Structural Integrity: Safety in miniature technology

Greg Hoover, Forrest Brewer, and Timothy Sherwood
University of California, Santa Barbara
{ghoover, forrest}@ece.ucsb.edu, sherwood@cs.ucsb.edu

Abstract

Micro-Electro-Mechanical (MEMS) sensing and actuation devices are poised to reinvent the cyber-physical world by providing environmental connectivity in unprecedented ways. MEMS accelerometers and transducers open the door to sensor networks embedded in physical structures, armed with the ability to monitor structural integrity. Such applications are the hallmark of deeply embedded systems, where autonomy, communication, and cyber-physical interaction play pivotal roles in system effectiveness. Realization of these next-generation embedded systems will require expertise spanning a variety of domains, further exacerbating the already significant integration effort of modern design. To this end, we propose a system-level design methodology that promotes functional correctness through specification of latency-insensitive (or event-based) behavior.

1 Motivation

Micro-Electro-Mechanical-Systems (MEMS) are positioned to be fundamental components in future cyber-physical systems, by allowing deeply embedded systems to interact with the physical environment. Widespread availability of devices such as accelerometers, gyroscopes, pressure transducers, and even complex arrays of thousands of micromirrors[1], enables system miniaturization and mobility at unprecedented levels. Despite the promise of MEMS devices, realizing practical, portable, and autonomous systems is remarkably difficult given the effort to actuate, interface, and control with current digital controller technology. This is largely due to scaling of physical time constants, which can be several orders of magnitude faster than macro-scale devices; in addition to ensemble behaviors including communication, power management, and computation, all at vastly lower power levels than conventional design.

Existing platforms designed around commercially available microcontroller devices have provided the means for exploring new areas in network sensors. While valuable demonstrations, these architectures do nothing to directly address the growing complexity of system-level design or the specific issues of MEMS. Rather, system complexity has been pushed to the software abstraction level, where real-time operating systems and clever programming are expected to fill the gap. In this space, sporadic, environment-driven execution are often well represented by event-based models. In our work, we propose to address the growing complexity in specifying *correct* system behavior through use of latency-insensitive, or event-based, semantic models. Such models are applicable at hardware, software, and system architecture levels of abstraction. We argue that an applicative, rule-based language[4] is a natural fit for this semantic model and that this flow enables a single specification to target a range of hardware implementations that tradeoff power and performance without sacrificing behavioral correctness.

2 Architecting Systems

Meeting system-level constraints in modern embedded environments has become a daunting task, and in the field of MEMS, requires high-performance custom ASIC implementations[2] to support control functionality. Novel architectures offer viable alternatives to commodity offerings by exploiting characteristics of both the application environment and execution patterns[5]. Unfortunately, architecture exploration is a costly and complex task which often mitigates extensive use of custom architectures. Much of the difficulty in the architectural design process lies in temporal integration of components. Interfaces tend to be both physically and temporally diverse, requiring significant effort to construct candidate designs meeting performance goals and to validate design correctness.

2.1 IP Integration

With the advent of true systems-on-chip came the need for retargetable IP solutions that provide common functionality with drop-in simplicity. Unfortunately this dream is far from the reality of current IP solutions, where the end-designer is responsible for correctness across design spaces that he has no direct influence on. This makes behavioral closure (correctness) challenging since a variety of ad-hoc methods may be employed for interfacing and communication. Moreover, such interfaces are typically built with intrinsic temporal constraints that must be satisfied. Latency-insensitivity, on the other hand, allows interface functionality to be decoupled from time, allowing arbitrary delays without affecting behavioral correctness. In this model, design correctness is insulated from temporal behavior, allowing composition of functionality and simplified debugging.

As a specification medium, rule-based languages are a natural fit since the description format closely represents the behavior of the underlying execution model[4]. Unfortunately, current synthesis techniques target conventional machine organization with global control and data accessibility, resulting in artificial critical paths and unnecessary communications. As an alternative to this, we propose the use of a set of tiny, composable connector circuits that implement a 2-signal communication protocol preserving elasticity throughout the control network[3]. In this format, control and data are tightly coupled, allowing distributed implementations where all interfaces are guaranteed to be composable. This composability exists at all levels of design abstraction, allowing system composition that is guaranteed to be behaviorally correct and thus significantly reducing the effort of system integration.

In software, this execution model is not a large departure from event-based methodologies currently employed. The latency-insensitive model can be realized as transaction-process or thread-based software implementations where functional tasks are dynamically scheduled for execution on available resources. The inherent parallelism available at the software abstraction level is typically more limited than in hardware, thus requiring a centralized mechanism for resource allocation. Despite this, it is the case that processes are always safe to run in parallel, and therefore take advantage of as much physical parallelism is made available by the architecture. A key aspect of this is the ability to identify data dependencies during synthesis and optimize inter-process communication and shared state.

The commonality between hardware and software models opens the door for a variety of implementation tradeoffs that are not available to conventionally composed systems. Elasticity in the interfaces of both hardware and software provide a common mechanism for communication, obviating the need for complex device drivers. The symbiosis of these technologies offers the potential for scalable system design in ways presently infeasible and offers opportunities for architectural exploration to identify efficient architectures with minimal effort. Consider the ability to automatically synthesize behavioral components to hardware in an effort to meet system constraints, all without rewriting any of the system specification.

2.2 Meeting System Constraints

Latency-insensitive design relies on the decoupling of temporal constraints and behavioral functionality. This is a contrarian view to conventional embedded design where system correctness is frequently tied to real time constraints. Here we emphasize that design correctness *should* be paramount throughout the specification processes, while system timing and power constraints *should* be optimization criteria and constraints. This allows a large number of implementations to be realized from a single specification, all with equivalent behavior. Powerful optimizations are possible via localized transformations which can reduce resource execution and communication, and lower power consumption. At the hardware level, power savings are also possible through selective clocking controlled by the token-based composition protocol.

3 Research Directions

Realizing cyber-physical systems that couple next-generation MEMS technology requires expertise spanning a multitude of disciplines from embedded systems to control theory. While our proposed methodology provides a feasible path to reach design constraints and eases much of the design burden, there are open issues related to execution modeling, efficient control partitioning, and physical interfacing. RTL synthesis provides an accurate foundation for modeling system latency and throughput in hardware. In software, however, the ability to model execution latency is more complicated. To this end, we must have mechanisms for establishing execution bounds by which to direct optimizations. In the world of MEMS, many are skeptical that all-digital solutions can provide the required performance while meeting system power and size constraints. While novel architecture have shown that it is feasible to achieve such constraints, software opportunities also exist for control partitioning and efficient resource-based execution strategies. And at the low-level, novel physical interfaces need to be developed for plug-in connectivity between high-voltage MEMS drivers (30V), low-current sensors (1nA), and practical integrated digital logic.

References

[1] D. J. Bishop, C. R. Giles, and G. P. Austin. The lucent lambdarouter: Mems technology of the future here today. In *IEEE Communication Magazine*, volume 40, pages 75–79, Mar. 2002.

[2] P. Chu, I. Brener, P. Chuan, L. Shi-Sheng, J. I. Dadap, P. Sangtae, and K. B. et al. Design and nonlinear servo control of mems mirrors and their performance in a large port-count optical switch. *Journal of Microelectromechanical Systems*, 14(2), Apr. 2005.

[3] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of synchronous elastic architectures. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 657–662, New York, NY, USA, 2006. ACM Press.

[4] J. C. Hoe and Arvind. Hardware synthesis from term rewriting systems. In *VLSI '99: Proceedings of X IFIP International Conference on VLSI*, 1999.

[5] G. Hoover, T. Sherwood, and F. Brewer. Towards understanding architectural tradeoffs in mems closed-loop feedback control. In *CASES '07: Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems*, 2007.