PREcision Timed (PRET) Machines

Isaac Liu

1 Introduction

Cyber-Phyiscal Systems are systems that heavily interact with the physical environment. These systems are often hard real time systems, which exhibit safety critical properties and have stringent timing requirements. Examples of these include avionics or automotive systems. Sangiovanni-Vincentelli et al.[1] listed in particular three difficulties as we continue to scale up designs for these systems. *Timing Predictability* is the ability to predict timing properties of the system. *Dependability* is the ability to handle and contain faults. *Composability* is the ability to modify tasks but still preserve system level temporal properties. This works mainly deals with *Timing Predictability* and *Composability* from the computer architecture level.

Time Predictability [2, 3, 4] all outlined the importance, requirements and difficulties of designing timing predictable systems. Modern abstraction layers have abstracted away "time". Programming languages such as C or Java do not have any timing semantics associated with the syntax. Thus, when designing systems where timing needs to be guaranteed, an analysis is needed to determine the worst case execution time (WCET). However, it is difficult to analyze the execution time of a program. Wilhelm et al. [5] described the abundant amount of research and effort that has been put into determining the WCET. Not only is determining the worst case program flow a challenge, but the precision and usefulness of the analysis also depends on the underlying architecture[6]. Conventional architectures have introduced speculation and prediction units that target improvement of average case execution time (ACET) at the expense of WCET. As a result, it's extremely complex, if not impossible to obtain precise analysis of the execution time on modern architectures.

Composability Modern designs use a "*Federated Architecture*" when designing systems that integrate many features. A Federated Architecture develops functions and features on physically separate architectures which are later integrated through an interconnect or system bus. This allows each feature to be designed and verified separately, since they won't interfere when integrated together. However, each unit only has a specialized function, thus they are often idle during run-time. In order to reduce resources (power, area, weight), there is a shift towards "*Integrated Architectures*" where multiple features are instead integrated onto shared hardware resources. [7, 8] both outline the challenges to switch from a Federated to an Integrated Architecture for avionics and automotive systems. Of those challenges, maintaining the ability to independently develop and verify features is most crucial for the continued scaling of systems. The shared platform must allow interference free integration of independent features.

Contribution The goal of this work is to propel design of cyber physical systems with stringent timing requirements. The key challenges we contribute to are the difficulties in designing time predictable systems, and the difficulty of integration when designing complex large scale systems. ISA extensions are proposed to allow temporal specifications at the instruction level. A timing predictable computer architecture is provided to allow for simple timing analysis at the architecture level. The architecture also provides interference free execution of multiple contexts with low context switch overhead, to allow for simple and efficient integration of multiple independent tasks.

2 ISA Extensions

In order to incorporate temporal semantics into the modern abstraction layers, we build upon Ip and Edwards'[9] proposal to extend the Instruction Set Architecture (ISA) with instructions that allow timing specification. Being part of the ISA, these timing specifications are transparent to the compiler, and enforced by the architecture. To allow for more expressiveness of temporal properties of the program, we introduce four variants of timing instructions that enable specifications for minimum execution time, worst case execution time, exact execution time and guaranteed execution time of code blocks. The first three are enforced at run time, while the last variant is a compile time constraint. With these extensions to the ISA, we enable timing control at the programming language level, compiler optimization towards user specified timing properties, portable timing behaviors across architecture families (when the ISA including the timing extension is preserved), but most importantly, we expose temporal properties in the abstraction layer to allow higher level design of cyber physical systems.

3 Precision Timed Machine

Lee et al.[10] proposed a paradigm shift in the design of computer architectures, focusing on timing predictability instead of average case performance. In this time predictable architecture, we employ thread interleaved pipelines with scratchpads.

Thread interleaved pipelines [11, 12] interleave multiple hardware threads through the pipeline in a round robin order. Every cycle a different thread is fetched into the pipeline. This removes the control and data hazards within the pipeline, allowing a simple pipeline design without hazard detection and speculation units. For memory accesses, scratchpads are used instead of caches. Scratchpad is an on chip memory in which the allocation of data from off chip memory is done in software. Thus, the contents of the on chip memory is known by the software, and memory accesses could be categorized as misses or hits to the on chip memory.

With this architecture, each instruction is independent of its execution context, resulting in a simple architectural timing analysis of a program. Furthermore, the execution of multiple contexts are independent, because of the static thread scheduling policy. This allows for composable resource sharing of separate tasks or functions, for simple and efficient integration of large scale systems.

4 Conlusion

[13, 14, 15, 16, 17, 18] have all proposed modifications to modern architectures in attempt to improve timing predictability with some average case performance penalty. This work goes in the opposite direction, focusing on timing predictability as the main objective, and only improving performance when predictability is not sacrificed. We propose ISA extensions that expose temporal properties in the abstract layers, and an architecture that is timing predictable and allows composable resource sharing of independent tasks. These features enable design at higher levels and are essential as we continue to scale in the design of cyber physical system.

References

[1] A. Sangiovanni-Vincentelli and M. D. Natale, "Embedded system design for automotive applications," *Computer*, vol. 40, pp. 42–51, 2007.

[2] T. A. Henzinger, "Two challenges in embedded systems design: Predictability and robustness," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 366, issue 1881, pp. 3727–3736, 2008.

[3] E. A. Lee, "Absolutely positively on time: What would it take?" *Computer*, vol. 38, pp. 85–87, 2005.

[4] L. Thiele and R. Wilhelm, "Design for time-predictability," in *Perspectives Workshop: Design of Systems with Predictable Behaviour*, ser. Dagstuhl Seminar Proceedings, L. Thiele and R. Wilhelm, Eds., no. 03471. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2004. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2004/2

[5] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem—overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 1–53, 2008.

[6] R. Heckmann, M. Langenbach, S. Thesing, and R. Wilhelm, "The influence of processor architecture on the design and the results of wcet tools," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1038–1054, 2003.

[7] R. Obermaisser, C. El Salloum, B. Huber, and H. Kopetz, "From a federated to an integrated automotive architecture," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 28, no. 7, pp. 956–965, 2009.

[8] W. C.B., R. Walter, G. Aviation, and G. Rapids, "Transitioning from federated avionics architectures to integrated modular avionics," *26th Digital Avionic Conference*, October 2007.

[9] N. J. H. Ip and S. A. Edwards, *A Processor Extension for Cycle-Accurate Real-Time Software*, 2006, pp. 449–458.

[10] S. A. Edwards and E. A. Lee, "The case for the precision timed (pret) machine," pp. 264–265, 2007.

[11] B. J. Smith, "Architecture and applications of the hep mulitprocessor computer system," pp. 342–349, 2000.

[12] E. Lee and D. Messerschmitt, "Pipeline interleaved programmable dsp's: Architecture," *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, vol. 35, no. 9, pp. 1320–1333, 1987.

[13] J. Yan and W. Zhang, "A time-predictable VLIW processor and its compiler support," *Real-Time Systems*, vol. 38, no. 1, pp. 67–84, 2008. [Online]. Available: http://dx.doi.org/10.1007/s11241-007-9030-5

[14] J. Whitham and N. Audsley, "Predictable out-of-order execution using virtual traces," pp. 445–455, 2008.

[15] T. U. Sascha Uhrig, Stefan Maier, "Toward a processor core for real-time capable autonomic systems." Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology, 2005.

[16] C. Rochange and P. Sainrat, "A time-predictable execution mode for superscalar pipelines with instruction prescheduling," pp. 307–314, 2005.

[17] A. El-Haj-Mahmoud, A. S. AL-Zawawi, A. Anantaraman, and E. Rotenberg, "Virtual multiprocessor: an analyzable, high-performance architecture for real-time computing," pp. 213–224, 2005.

[18] J. Barre, C. Rochange, and P. Sainrat, "A predictable simultaneous multithreading scheme for hard real-time," vol. 4934, pp. 161–172, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78153-0_13