

# Research Statement

Chengmo Yang

Computer Science and Engineering Department  
University of California, San Diego  
*c5yang@cs.ucsd.edu*

## 1 Motivation and vision

The continued improvements in VLSI fabrication technologies have placed chip multiprocessors center stage as the dominant architectures in the next computer generation. Our ability to exploit such an extant computational power, however, is checkmated not only by parallelism extraction limitations of our current techniques, but furthermore by increasing levels of execution uncertainty within the system. As device feature sizes scale towards nanoscale, not only would a high occurrence of device faults be expected at run-time, but potential thermal stress would also make cores temporally unavailable during execution. Such an unreliable platform is used to concurrently hold an increasing number of applications that constantly vie for execution resources on the computational fabric, thus furthermore requiring resource demands to be frequently renegotiated at run-time.

The unreliability in the electronic fabric and the unpredictability in the execution process are coalescing to make *execution adaptivity* the cornerstone of the research challenges in multicore platforms. Unfortunately, the increasing level of execution uncertainty cannot be surmounted through incremental optimizations of traditional pure run-time fault tolerance and resource management techniques, as they suffer from inevitable overhead either in comparing and checkpointing execution states, or in collecting and communicating workload information respectively. Instead, future multicore systems necessitate aggressive support to quickly detect a resource variation due to execution faults, heat buildup, or application competition, as well as to quickly reconfigure the execution upon a resource variation. Such aggressive tasks need to be addressed through a set of system-level software & hardware co-optimization techniques, thus requiring the consideration of *execution adaptivity* as a primary design constraint.

## 2 Multicore platform design challenges

While relentless scaling of CMOS technology has provided a steady increase in processor performance, it also adversely affects long-term chip lifetime reliability. Currently, the leading CMOS technology has reached 45 nanometers in scale, yet the rate of transient and intermittent faults has increased by three orders of magnitude (from  $10^{-6}$  to  $10^{-3}$ ) as technology has advanced from 180nm to 45nm [1, 2]. On the other hand, the shrinking is projected to reach beyond 32 nanometers in scale by 2013 [3]. The ultra low voltage and the extremely high frequency, as a result, will lead to significantly reduced noise margins, thus making nanoscale devices more sensitive to temperature, background noises, and crosstalk effects. This reliability issue is furthermore worsened by heat buildup, which not only accelerates the chemical process taking place inside the chip, but also diminishes the switching speed of the transistors.

Despite the increasing level of unreliability in the device world, the complexity of current applications still grows dramatically. As reported by ITRS [3], multicore platforms will be rapidly employed for applications including defense, office, portable/consumer, medical, and networking/communication. Efficient utilization of the ample hardware resources requires these envisioned applications to be decomposed into fine-grained concurrent threads, with the frequency and volume of inter-thread communications increasing super-linearly as the number of threads grows. The concurrently executed applications would also constantly vie for execution resources on the platform, as each of them individually exhibits unpredictable resource utilization intensity. Resource demands therefore need to be constantly renegotiated at run-time, thus requiring flexible resource reallocation and consequent execution rescheduling.

The increasingly pessimistic news on the device reliability and the execution predictability fronts requires the consideration of *execution adaptivity* as a primary design constraint for multicore systems. The increasing level of uncertainty imposed by intermittent failures, heat buildups, frequent communications, and resource competitions furthermore argues for techniques of highly-constrained and predictable performance and power overhead. Unfortunately, pure run-time fault tolerance and scheduling techniques suffer from inevitable overhead either in comparing and checkpointing execution states, or in collecting and communicating workload information respectively. The fundamental culprit for this situation is the *generality* that these techniques aim to attain. Specifically, to detect and recover from

arbitrary execution faults, traditional duplication-based fault detection techniques usually buffer and compare all the load/store values, although it is unnecessary to check a value stored in the cache if it would not be written back to the memory. Similarly, run-time scheduling techniques need to collect resource usage information through a centralized message channel or shared memory. These traditional communication and synchronization protocols ensure that any producer can send data to any consumer, despite localized communications being the most common case as forced by interconnect costs.

While the generality of pure run-time techniques provides flexibility and ease of implementation, it on the other hand limits the applicability of performance- and area- improving techniques that exploit static, compiler-derived information. Run-time scheduling approaches, as an example, fall short of exploiting deterministic program information regarding the varying amounts of resource requirement. If an execution fault-induced resource variation occurs, the scheduler can only make locally optimal rescheduling decisions, thus possibly causing unpredictable impact on the performance of individual applications that frequently exhibit real-time constraints, such as multimedia, stream, automotive and game applications, to name a few.

The identification of the common limitations of traditional run-time optimization techniques provides a starting point for the incorporation of light-weight *fault detection*, predictable *execution reconfiguration*, as well as cheap *inter-thread communication* support into future multicore platforms to surmount the increasing level of execution uncertainty. Such identified limitations designate **two** optimization directions, namely, exploiting statically extracted program information to guide run-time optimization techniques, and fine-tuning architectural components in such a way that the most common cases can be accelerated through dedicated techniques.

### 3 Doctoral work

The objective of my Ph.D thesis is the definition of a new multicore architecture with aggressive *execution adaptivity* support so as to effectively surmount the execution uncertainty imposed by intermittent failures, heat buildups, frequent communications, and resource competitions. To achieve this goal, my doctoral work has mainly focused on **four** tightly coupled topics that conceptually address the development of fine-tuned architecture components as well as the algorithmic support for statically extracting useful program information. Specifically, the exploitation of the program information regarding inter-thread data dependences enables the generation of adaptive static execution schedules, as well as the encoding of inter-thread communication directions. Such information can be transferred to, and efficiently utilized by the multicore architecture. Meanwhile, the cache organization and the memory topology of the platform can be redesigned so as to minimize the performance and power overhead of fault detection, execution reconfiguration, and inter-thread communication.

Future multicore platforms will suffer from variations in resource availability due to either intermittent faults, or thermal stress, or resource competition between applications. The highly unpredictable yet frequent occurrence of such variations furthermore argues for flexible execution migration, capable of delivering predictable impact on individual applications. To compensate for the lack of global program visibility at run time, a set of possible execution schedules can be compactly captured during compile-time, thus delivering regular and predictable response to various resource availability constraints [4]. By imposing conceptually a block & band structure on the static task schedules, a regular reassignment capability can be attained through performing a group transfer of the whole band of tasks upon a dynamic resource variation. Within each band the relative position of each task is retained intact, thus naturally preserving the spatial and temporal locality of the tasks within each band after reconfiguration. Moreover, through the incorporation of a set of soft constraints into the scheduling process [5], the inherent flexibility regarding the core order can be utilized so as to effectively eradicate the performance overhead introduced by reconfiguration while retaining all the concomitant benefits on static schedules.

The compiler-directed predictable execution reconfiguration technique is furthermore supported by a light-weight fault detection mechanism so as to enhance the reliability of the target multicore platform. Through exploiting the inherent redundancy of cache access patterns produced by two identical processes/threads, arbitrary execution faults can be detected within minimum cost in both performance and hardware [6]. Specifically, the sharing of a single cache between a task and its duplicate enables the duplicate to directly check values written by the task into the cache, while only confirmed results are written to the memory so as to strictly protect it from being polluted by execution faults. Moreover, as the task is allowed to write unconfirmed results into the cache, the strict instruction-by-instruction synchronization model can be relaxed. The task thus can run ahead of its duplicate in execution, thus offering the additional benefit of workload balancing since only the task encounters misses in the shared cache yet only the duplicate needs to compare store values. Meanwhile, to further enlarge the amount of run-ahead offset, the

cache design can be extended so as to selectively buffer the old value of a block upon a cache block dependence, thus enabling both thread copies to run independently.

A light-weight communication protocol in future multicore platforms is indispensable not only for dependent threads to frequently exchange data, but also for a run-time resource manager to quickly collect workload information. Rather than employing a generic solution that allows any producer to send data to any consumer, a cost-efficient solution needs to differentiate neighborhood-centered communications from long-distance communications and accelerate the former. To attain this goal, a light-weight communication technique [7] has been developed, wherein two cores are allowed to directly communicate through a shared small register file or cache structure. The application information regarding inter-thread dependences furthermore enables the semantically correct access order to any shared variable to be statically identified and encoded, thus completely eliminating the continuous polling of explicit synchronization variables. Moreover, as a thread only communicates in most cases with a small and fixed subset of the rest of the threads, global dependence information can be encoded in arbitrary access contexts within only negligible overhead, thus furthermore enabling a highly-efficient implementation.

The incorporation of the aforementioned execution reconfiguration, fault detection, and communication techniques into a multicore platform necessitates a reconsideration of the underlying memory topology. Since store values, data to communicate, and checkpointed execution states need to be shifted among cores, a *sharable yet scalable* memory organization is necessitated. On the other hand, as execution migration and inter-thread communication only involve a limited set of computation nodes, shareability only needs to be established within a neighborhood, resulting in a *locally sharable* memory model being an appealing compromise. Here, memory is still organized in a distributed form, while sharing is achieved through each core directly accessing multiple memory units and each unit directly accessible to a set of adjacent cores. Meanwhile, this model exhibits an inherent redundancy in that there exist multiple paths for any pair of cores to communicate through, thus enabling the entire platform to retain its connectivity in the case of single failures of either core or communication link. Such a locally sharable property is furthermore independent of a particular topological structure, thus allowing distinct 2-dimensional topologies to be adapted for diverse application sets so as to match parallelism characteristics and resilience needs of the application.

## 4 Future Research Directions

As process technologies continue to evolve, the semiconductor integration density and consequent execution uncertainty would reach even higher levels. The design of multicore systems with aggressive yet predictable *execution adaptivity* support creates a highly exciting research dimension with various disciplines of computer science and engineering closely involved, including architectures, compilers, operating systems, VLSI design, on-line testing and diagnosis, among others. I am eager to carry out my future research work in pushing the development of this new area with extensive communication and cooperation with the research experts across multiple disciplines and universities.

My doctoral work has conceptually addressed the achievement of execution adaptivity in multicore platforms from the perspectives of architectural component redesign and algorithm development for static program information utilization. In my future work, I will continue to bring more concreteness to the fundamentals of my doctoral work, while at the same time exploring the connections and interactions among the aforementioned various disciplines. My vision of the interdisciplinary examination to expedite the development of adaptive multicore systems focuses on the following **short-term research directions**:

- The role of the **compiler** is of great importance for the attainment of execution adaptivity within multicore systems. I will work and collaborate in this area so as to establish new directions of utilizing and adjusting advanced compiler optimization techniques, such as *speculation* and *predication*. I will explore the impact of various compiler optimization techniques on several issues, including the encoding of communications within data transfer instructions, the generation of compact static execution schedules with reconfigurability embedded, as well as the construction of statistical task latency models for improving the accuracy of static schedules.
- The **Operating System** has an active role in identifying resource availability and adjusting resource allocation footprints. Real-time OSs furthermore determine the performance of real-time systems for which determinism and responsiveness are important product requirements. I plan to research the development of aggressive real-time OSs which, based on statically extracted resource requirement information, can attain resource reallocation within a small and predictable timing overhead. Additionally, hardware components that would facilitate the OS to efficiently dispatch the statically generated schedule blocks to a cluster of cores will be explored.
- The various types of device unreliability impose a crucial obstacle in multicore systems, namely, ambiguity in fault manifestation rates and in fault types. **On-line testing and diagnosis** techniques can be applied on the

ambiguity set to identify the faulty component and furthermore differentiate transient from permanent faults. Meanwhile, application information regarding execution invariants can be used for property checking which may provide precise identification for system integrity. I plan to extend the development of the adaptive execution environment in these two directions so that different recovery techniques can be applied upon a cost-effective identification of the faulty component and the fault type.

- **VLSI design** techniques can significantly impact various aspects of the proposed adaptive multicore platform, including cost-effective heat removal, design for testability and reliability, and design for minimization of performance/power sensitivity to device variability and layout styles. I plan to work and collaborate in order to explore VLSI architectures and design principles that will additionally boost the applicability of the proposed execution adaptivity framework. One of the main focal points is the development of an advanced interconnect network that addresses performance, power, and reliability issues with functional diversity.

As CMOS scaling is approaching its physical limits, nanotechnology has been widely acknowledged as the foundation for the next generation of computer systems. While the hardware constraint becomes less stringent in nanoelectronic system design, the level of execution uncertainty is ever higher. I am highly interested in carrying out a number of **long term research directions** regarding the issues of fault detection, execution reconfiguration and communication cost reduction in such systems, however, with particular considerations of nano-system characteristics, such as the strict interconnect constraint and the clustered fault behavior as a result of the bottom-up fabrication.

**Communication Model Development:** Interconnection is one of the dominant constraints in a nanoelectronic system in terms of area, delay and power consumption, thus forcing localized communication to become a critical criterion in the nanoelectronic environment. Efficient topology and structure for such nanoelectronic systems, together with power-aware and reliable ways to communicate data across the chip, constitute significant obstacles that need to be overcome. I will base my new research work on the locally shareable memory model that I have developed and investigate *novel system topologies* and *communication models* by collaborating with related areas of networks-on-chip and nano fabrication. Such new topologies and models certainly will contribute to the formation of the future nanoelectronic systems in terms of computation unit allocation and execution reconfiguration regularization.

**Fault behavior characterization:** The fabrication process in nano environments is prone to defects due to the small scale of devices and the bottom-up self-assembly process. Not only is the fault rate projected to be high, but also a high variance in the fault rate can be expected. Such variations lead to significant differences in performance, robustness, as well as noise immunity among the devices. This clustering behavior should be considered in the development of fault tolerance approaches, together with other effects such as transient/permanent characteristics, temperature-induced fault rate increases, and testing-induced device damages. I am anticipating to work with the research groups in the nanoelectronic device level in *characterizing fault behavior* of various nano devices, so as to enable the development and precise parameterization of fault tolerance schemes for nanoelectronic systems.

## References

- [1] P. Shivakumar, S. W. Keckler, D. Burger, M. Kistler, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in *International Conference on Dependable Systems and Networks (DSN)*, June 2002, pp. 389–398.
- [2] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The Impact of Technology Scaling on Lifetime Reliability," in *International Conference on Dependable Systems and Networks (DSN)*, June 2004, pp. 177–186.
- [3] International Technology Roadmap for Semiconductors (ITRS), "ITRS 2007 Edition: Executive Summary," <http://www.itrs.net/Links/2007ITRS/Home2007.htm>
- [4] C. Yang and A. Orailoglu, "Predictable Execution Adaptivity through Embedding Dynamic Reconfigurability into Static MPSoC Schedules," in *International Conference on Hardware Software Codesign (ISSS-CODES)*, September 2007, pp. 15–20.
- [5] C. Yang and A. Orailoglu, "Towards No-cost Adaptive MPSoC Static Schedules through Exploitation of Logical-to-physical Core Mapping Latitude," in *Design, Automation and Test in Europe (DATE)*, April 2009.
- [6] C. Yang and A. Orailoglu, "A light-weight Cache-based Fault Detection and Checkpointing Scheme for MP-SoCs Enabling Relaxed Execution Synchronization," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, October 2008, pp. 11–20.
- [7] C. Yang and A. Orailoglu, "Light-weight Synchronization for Inter-processor Communication Acceleration on Embedded MPSoCs," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, September 2007, pp. 150–154.