

Research Statement

Chengmo Yang

Computer Science and Engineering Department

University of California, San Diego

c5yang@cs.ucsd.edu

1 Motivation and vision

The continued scaling of silicon fabrication technologies has enabled the integration of dozens of processing cores on a single chip in the next computer generation. Our ability to exploit such computational power, however, is checkmated not only by limitations of parallelism extraction techniques, but furthermore by increasing levels of execution uncertainty within the system. As device feature sizes scale below 45nm, reliability has rapidly moved to the forefront of concerns for leading semiconductor companies, with the main challenge being the scaling of system performance while meeting power and reliability budgets. To make things worse, such an unreliable computational fabric is used to concurrently execute an increasing number of applications that constantly vie for execution resources, thus furthermore making the execution environment more dynamic and unpredictable.

The unreliability in the electronic fabric, in conjunction with the unpredictability in the execution process, results in frequent renegotiation of computational resources at run-time. **Execution adaptivity** thus becomes the cornerstone of the research challenges for future multicore systems. To attain this goal in a cost-effective manner while at the same time delivering maximum computation efficiency and predictable worst-case performance, I have proposed a set of *compiler-directed run-time optimization* techniques in my doctoral work. I extract intensive static program information to guide architectural component redevelopment and dynamic execution reorganization, thus effectively compensating for the lack of global program visibility at run-time within negligible cost. Specifically, I propose several tightly-coupled techniques for future multicore systems in order to **1)** maximally **mitigate** sources of uncertainty through cost-effective temperature reduction and communication acceleration techniques, **2)** precisely **detect** resource variations, especially the ones induced by device failures, and **3)** quickly **reconfigure** the execution upon a resource variation in a predictable manner with no reliance on spare units.

2 Multicore platform design challenges

While continued scaling of CMOS technology has provided steady improvements in processor performance and cost, it also adversely affects long-term chip lifetime reliability. The shrinking is projected to reach beyond 32nm in scale by 2013 [1]. Yet issues that were considered as second-order effects in the past, such as Soft-breakdowns (SBD) in device gate oxide, Negative Bias Temperature Instability (NBTI) in PMOS threshold voltage, Electro-Migration (EM) in copper interconnects, and dielectric breakdown in low- k materials, become clear threats for systems in near future technologies. Heat buildup furthermore accelerates the chemical process inside the chip and diminishes the transistor switching speed, engendering in turn functional and timing failures. This significant increase in fault rate (from 10^{-6} to 10^{-3} as technology has advanced from 180nm to 45nm [2]) in turn degrades manufacturing yield and product lifetime. Semiconductor companies therefore have started to incorporate reliability support, regardless of the expense, into their newly-released designs, such as Intel *QuickPath*, and IBM *Power6* [3].

Despite the increasing level of unreliability in the device world, the complexity of current applications still grows dramatically. As reported by ITRS [1], 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, engendering in turn super-linearly increased communications, in conjunction with resource competitions and preemptions. Resource demands therefore need to be constantly renegotiated at run-time, thus requiring flexible resource reallocation and consequent execution rescheduling.

The projected degradation in device reliability, in conjunction with the unpredictable resource utilization intensity, imposes stringent requirements of *resource variation tolerance* for future multicore systems. It is essential to incorporate aggressive technical support to precisely detect and identify resource variations, and then rapidly recover and reconfigure the execution. To make things more complicated, these functions need to be attained in conjunction with the many challenges that designers already face, such as computation efficiency, power and thermal management, and predictability of worst-case performance. Circuit level replication techniques, such as Razor [4], have proven successful in defect tolerance for microprocessors. Yet their high cost and inflexibility in responding to highly variable manifestations of the faults limit their applicability to multicore systems in near future technologies. It is thus essential to complement such techniques by *system-level* approaches to efficiently and flexibly attain **execution adaptivity**. Computation on a core can be temporarily suspended upon the detection of a resource reduction. Conversely, cores can be pulled into execution once local heat buildup has abated, intermittent fault durations have elapsed, or resource competition pressures have diminished.

While one may envision incremental optimizations of traditional pure run-time fault tolerance or resource management techniques can prove successful in delivering *execution adaptivity*, a crucial limitation of either is *brittleness*: the pre-optimized solutions become useless upon an unpredictable event, while dynamic reactions waste significant computation power and, due to their sub-optimal nature, unpredictably impact each application. The fundamental causes for this situation are the lack of global program visibility at run-time, as well as the *generality* that these techniques aim to attain. This generality, although it offers flexibility and ease of implementation, limits the applicability of performance- and area- improving techniques that exploit static, compiler-derived information. Pure run-time scheduling approaches, as an example, fall short of exploiting deterministic program information regarding the regularity of inter-task dependences. Similarly, traditional duplication-based fault detection techniques usually compare all the store values, thus failing to exploit the fact that it is unnecessary to check a store value if it would pollute neither memory nor other cores.

To compensate for the aforementioned common limitations of traditional run-time optimization techniques, intensive static program information needs to be coupled with run-time optimizations in future multicore platforms. I have thus proposed to establish a *compiler-directed run-time optimization framework*, wherein regularity can be embedded into the system, and program information can be extracted and synthesized statically. Such information is to be transferred to, and efficiently utilized by the architecture, so that superior dynamic decisions can be made and architectural components can be fine tuned.

3 Doctoral work

The objective of my Ph.D thesis is the definition of a new multicore architecture with aggressive compiler-directed, run-time *execution adaptivity* support to effectively surmount the increasing level of execution uncertainty imposed by device failures, heat buildups, unexpected communications or resource competitions. The fundamental key solution is to statically extract useful program information to guide run-time decision-making and architectural component redesign. Specifically, my doctoral work has focused on **five** tightly coupled topics that address execution uncertainty from the perspectives of *mitigation* (i.e., temperature reduction and communication acceleration), *detection*, *recovery* (i.e., execution reconfiguration), as well as *architectural reorganization* for a cost-effective implementation of all these functions.

Predictable Execution Reconfiguration: Due to the highly unpredictable yet frequent occurrence of resource variations, future multicore platforms necessitate flexible execution migration, capable of delivering predictable worst-case performance. To compensate for the lack of global program visibility at run-time, a set of possible execution schedules are compactly captured during compile-time, with each of these schedules making full utilization of the available resources [5]. By imposing conceptually a shiftable band structure, tasks can be transferred between adjacent cores in a regular manner upon run-time resource variations, with no reliance on any rescheduling decisions being made on the fly. Within each band the relative position of each task is retained intact, thus naturally preserving most of the inter-task dependences during reconfiguration. Moreover, through the incorporation of a set of soft constraints into the scheduling process [6], the inherent flexibility regarding the logical-to-physical core binding order can be utilized so as to effectively eradicate the performance overhead while retaining all the concomitant benefits of schedule reconfigurability.

Cost-effective Fault Detection: The compiler-directed predictable execution reconfiguration technique needs to be supported by a light-weight fault detection mechanism so as to enhance overall system reliability. Yet traditional fault tolerance techniques incur significant overhead either in checkpointing execution results or in constantly synchronizing the execution for value checking. To reduce such overhead while at the same time delivering full fault detection capability, a single fault-tolerant cache [7] is shared between a pair of threads, so that one thread can directly verify the results of the other. Protection of the main memory against contamination by execution faults is thus attained, thus drastically reducing the checkpointing overhead. Meanwhile, as unconfirmed results are allowed to be written into the cache, two redundant threads can run independently without constantly synchronizing for value checking. The cache design can be extended to selectively duplicate a constantly-updated cache block and skip the comparison of the intermediate values, thus further reducing comparison overhead and hence relaxing synchronization requirements.

Heat-driven Deterministic Register Shuffling: As the system fault rates exponentially increase as peak temperature rises, overall system reliability can be improved through reducing the peak temperature of the most overheated module in each core, to wit, the register file. Detailed examinations indicate that register accesses display a highly unbalanced distribution, and the challenge in temperature reduction is to physically remap the heavily accessed registers before heat gets locally accumulated, yet to do so with *no reliance* on any hardware renaming table. To attain this goal, I have proposed a compiler-directed iteration-based register shuffling technique [8]. Through the exploitation of application-specific register usage information, the compiler can embed regularity into register names, thus allowing register mapping to be deterministically shuffled across loop iterations for access balance without a renaming table. The extremely low hardware overhead minimizes the associated power dissipation, thus enabling an easy incorporation of this dynamic remapping support into multicore systems to enhance the reliability of the entire chip.

Localized Communication Acceleration: 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 rapidly collect workload information. Traditional communication protocols employ a generic solution that allows any producer to send data to any consumer, failing to exploit the fact that in most cases a thread only communicates with a small and fixed subset of neighbors. In contrast, I have proposed a cost-effective solution [9] to differentiate neighborhood-centered communications from long-distance communications and accelerate the former. Two adjacent cores are allowed to directly communicate through dedicated buffers or shared cache structure, while semantically correct access order can be precisely identified and encoded through the static extraction of communication patterns, thus completely eliminating the continuous polling of explicit synchronization variables. The limited communication patterns furthermore allow localized communications to be encoded in arbitrary access contexts within only negligible overhead, thus in turn enabling a highly-efficient implementation.

Locally Shareable MPSoC topologies: As execution results and computation states need to be frequently transferred among cores, future multicore platforms need a *shareable* yet *scalable* memory model. Yet the outlined execution reconfiguration, fault detection, and communication techniques only involve a limited set of computation nodes, implying that shareability only needs to be established within a neighborhood. I have thus proposed a locally shareable memory model wherein memory is organized in a distributed form, and sharing is attained through making each memory unit directly accessible to a set of adjacent cores. This model exhibits an inherent redundancy in that there exist multiple paths for any pair of cores to communicate through, thus ensuring the connectivity of the entire platform in the case of single failures of either core or interconnect. 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.

In sum, the successful completion of the fine-grained and predictable adaptive multicore platform that I have proposed and worked on during my doctoral work, I believe, will engender *adaptive, scalable architectures that can seamlessly reshape execution paths and schedules in an amortizable, high-volume, fixed-silicon fabric*, thus providing avenues for effectively addressing thermal buildups, possible fault occurrences and even resource competition among multiple applications executing simultaneously.

4 Future Research Directions

As process technologies continue to evolve, the issue of execution uncertainty is exacerbated as we negotiate the end of the CMOS era and move onto the world of nanoelectronics. The design of multicore systems with aggressive yet predictable execution adaptivity support creates a highly exciting research dimension. My doctoral work has addressed this issue mainly from the perspectives of architectural component redesign and algorithm development for static program information utilization. Yet a complete realization of such a fully adaptive system requires the exploration of interactions between architectures and various other disciplines of computer science and engineering, including *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 collaboration with the research experts across multiple disciplines and universities.

To expedite the development of adaptive multicore systems, 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. More concretely, the exploitation of *compiler* optimizations and *on-line testing* techniques enables further reductions in run-time decision-making overhead, while interactions with OSs and VLSI design can be exploited to further improve the determinism of the overall system. My vision of the interdisciplinary examination thus focuses on the following research directions that I would like to start on immediately upon joining academia:

- The role of the **compiler** is of great importance for the attainment of execution adaptivity within multicore systems, especially for application specific embedded systems. I will work and collaborate in these areas 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 models of task execution time for improving the accuracy of static schedules.
- The **operating system** has an active role in dynamically identifying resource availability and adjusting resource allocation footprints. Real-time OSs furthermore determine the efficiency of design and maintenance 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 information regarding variations in resource requirements, 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. Precise identification of the faulty component and the fault type requires the use of **on-line testing and diagnosis** techniques. Additionally, application information regarding execution invariants can be used for property checking which may provide precise identification for system integrity in a cost-effective manner. I plan to extend the development of the adaptive execution environment in these two directions so that faulty components can be precisely identified, transient and permanent faults can be distinguished, and diverse fault recovery schemes can be adaptively applied.
- **VLSI design** techniques can significantly impact various aspects of the proposed adaptive multicore platform, including cost-effective heat removal, design testability and reliability enhancement, and the achievement of execution predictability in the face of device variability. I plan to work and collaborate in order to explore VLSI architectures and design principles that will additionally boost the advantages 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 while at the same time delivering 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. Yet the level of execution uncertainty in nanoelectronic systems is ever higher, since the fabrication process in nano environments is prone to defects due to the small

scale of devices and the bottom-up self-assembly process. Meanwhile, interconnect is one of the dominant constraints in a nanoelectronic system in terms of area, delay and power consumption. With particular consideration of these nano-system characteristics, 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 nanoelectronic systems.

- **Reliable and efficient nano computing:** The fundamental challenge in constructing a workable nanoelectronic system is to enable reliable computations despite the severe unreliability imposed by the underlying nanoelectronic devices. In such systems, not only is the fault rate projected to be high, but also a high variance in the fault rate can be expected. 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 damage. I am anticipating to collaborate with the research groups in the nanoelectronic device level so that based on *fault behavior characterization* of various nano devices, the most cost-effective configurations of fault tolerance schemes can be precisely identified and adaptively tuned.
- **Communication model development:** The strict interconnect constraint in a nanoelectronic system forces localized communication to become a critical criterion. 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 will contribute to the formation of the future nanoelectronic systems in terms of computation unit allocation and execution reconfiguration regularization.

References

- [1] International Technology Roadmap for Semiconductors (ITRS), "ITRS 2007 Edition: Executive Summary," <http://www.itrs.net/Links/2007ITRS/Home2007.htm>
- [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] M. J. Mack, W. M. Sauer, S. B. Swaney, and B. G. Mealey, "IBM POWER6 reliability," in *IBM Journal of Research and Development*, vol. 51, no. 6, November 2007, pp. 763–774.
- [4] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: a Low-power Pipeline Based on Circuit-level Timing Speculation," in *International Symposium on Microarchitecture (MICRO)*, December 2003, pp. 7–18.
- [5] 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.
- [6] 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, pp. 63–68.
- [7] C. Yang and A. Orailoglu, "A light-weight Cache-based Fault Detection and Checkpointing Scheme for MPSoCs Enabling Relaxed Execution Synchronization," in *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, October 2008, pp. 11–20.
- [8] C. Yang and A. Orailoglu, "Processor Reliability Enhancement through Compiler-Directed Register File Peak Temperature Reduction," in *International Conference on Dependable Systems and Networks (DSN)*, June 2009, pp. 468–477.
- [9] 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.