# Reverse-Engineering a Processor

## Discovering Structural Hazards for Scheduling

# What the processor manual does not tell you...

Matthew J. Bridges  Neil Vachharajani  Guilherme Ottoni  David I. August

{mbridges,nvachhar,ottoni,august}@princeton.edu

Generation of high quality code for modern and embedded processors requires a compiler to maximize utilization of available resources. An essential step in this is the aggressive scheduling of instructions while avoiding stalls from structural hazards. To do this, an optimizing compiler must be aware of the processor's available resources and how these resources are utilized by each instruction. The typical process of manually discovering and specifying this information is both tedious and error-prone. Formal models can be used to automatically generate this information, however, there are situations where these are not used or available. Ideally, structural hazards would be determined automatically, without the need for a formal model.

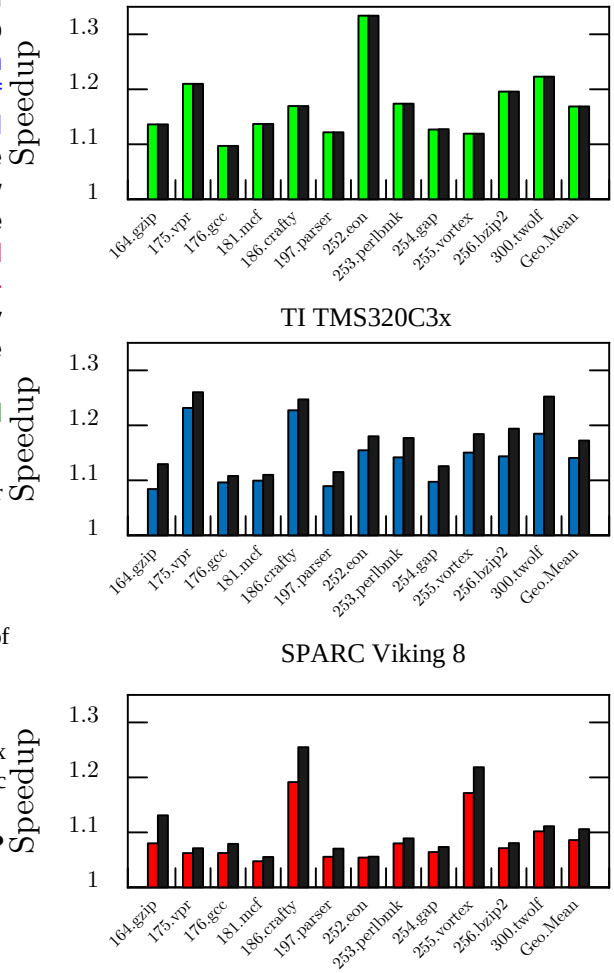
**Why is this needed?** First, machine information needed for compiler development is not always available or, when available, accurate. Second, evaluation with a tuned optimizing compiler is important during the design space exploration of a computer system, where it is necessary to rapidly explore several candidate designs. Since the quality of the system depends on the number of candidate designs that can be explored, the ability to rapidly retarget an instruction scheduler is critical. However, retargeting the compiler's instruction scheduler to a new candidate design is particularly tedious and time-consuming due to the many complex interactions between instructions and the tight coupling to specific design implementation details.

**Isn't reverse-engineering a processor impossible? Yes!**

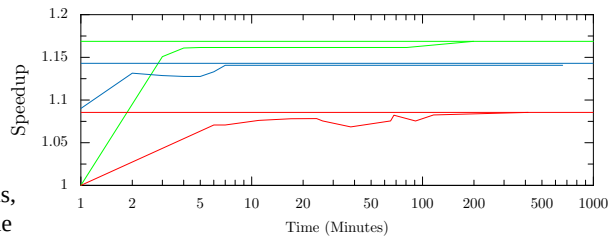
Formal models, such as used by ADLs, can allow a processor model to be reverse-engineered. However, it is impossible to automatically discover *all* structural hazards without such formal models. Our technique automatically reverse-engineers the *most* important structural hazards. This technique uses observations about the properties of processors to identify a subset of instruction schedules to explore. Structural azards found during exploration are used to infer the existence of other hazards. The subset of instruction schedules formed is based on three observations:

- A processor's functional units tend to be pipelined, allowing the depth of instruction schedules considered to be limited.
- Instructions tend to belong to categories defined by resource usage, allowing a representative to be chosen for exploration.
- Use of resources tends to be independent of an instruction's position within a cycle, allowing combinations, rather than permutations, of instructions to be explored.

**How well does it work?** The technique was used to build a *conflict database*, containing known and inferred structural hazards, which was then used for hazard detection during scheduling. As the graphs to the right show, the structural hazards we discover and infer are sufficient to obtain 80-100% of the performance of perfect resource maps when scheduling. Additionally, the bottom right graph shows the speedup of the technique over time. Almost all of the speedup is achieved within 3 hours, with almost all of the performance obtained in under 15 minutes.




*The speedup versus scheduling without knowledge of structural hazards for three machines. The speedup obtained by our technique is shown in color and the speedup obtained by perfect resource maps is shown in **black**.*




*The accuracy (measured as speedup) over time for our technique for the TI TMS320C3x, SPARC Viking 8, and Itanium 2. The horizontal line represents maximum speedup for each machine.*

**More Information:** http://www.liberty-research.org/Research/DSE or contact the Liberty Design Space Exploration Team at the addresses above.