Adaptive Resource Partitioning in Simultaneous MultiThreading Architectures

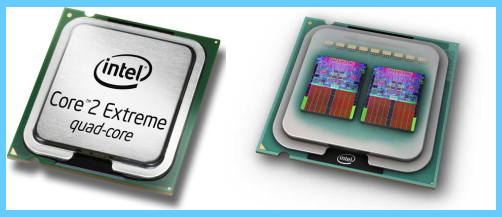
Chen Liu

Prof. Jean-Luc Gaudiot, Electrical Engineering & Computer Science Dept.

The Henry Samueli School of Engineering

University of California, Irvine

What is a general purpose processor ?



- Here is one of Intel's most recent commodity processors

What do we want?

- Execute as many instructions as we can and as fast as we can

Houston, we have a problem!

- How do we distribute the system resources among threads to achieve best performance?

This is Houston!

Two choices:

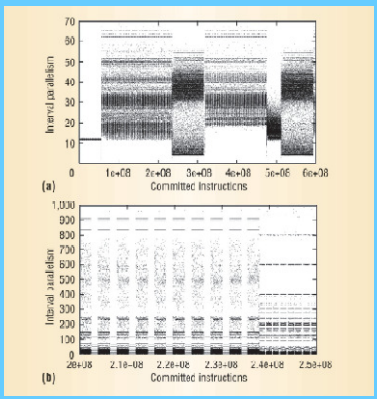
Dynamic Sharing: threads compete for resources among themselves

Problem: what if a thread grab many resources and then have a cache miss?

OR

Static Partitioning: each thread has a pre-assigned cap on resources

Can we do better?



- The behavior of a program can be divided into different phases, based for instance on the number of committed instructions
- The need for system resources at different phases is different
- Static partitioning fails to meet the varying needs of the threads at different phases

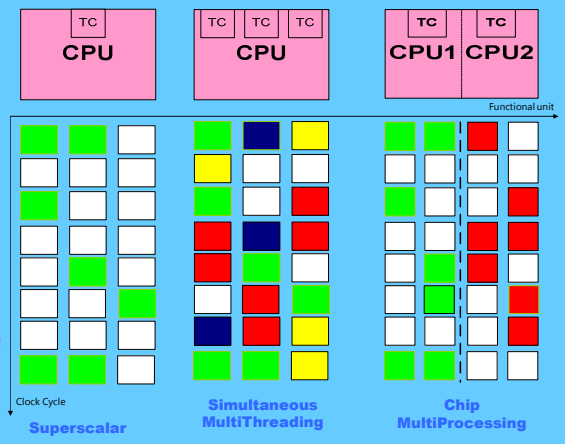
Can we do that?

Due to the sequential programming model, we can only exploit limited Instruction-Level Parallelism (ILP), because of:

- Data hazards
- Control hazards
- Structure hazards

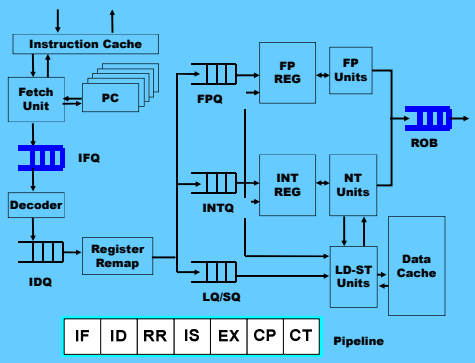
What can we do then?

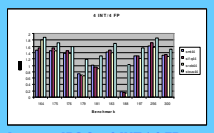
- Exploit Thread-Level Parallelism (TLP) in addition to ILP



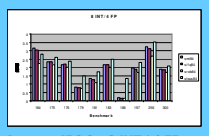
- SMT achieve the BEST resource utilization (different color represents different threads)

Let's see what happens.

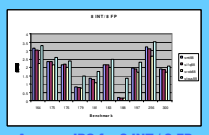




Average IPC for 4 INT / 4 FP functional units configuration



Average IPC for 8 INT / 4 FP functional units configuration



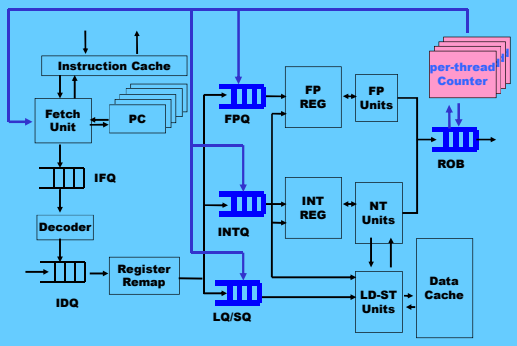
Average IPC for 8 INT / 8 FP functional units configuration

- Compare dynamic sharing of Instruction IFQ and ROB with static partitioning
- The performance gain comes from the ability in static partitioning to avoid pipeline clogging

What do we really need?

Adaptive Resource Partitioning: Resource distribution varies according to the performance of the thread, approaching optimal performance

- Behavior divided into Commit-bound and Issue-bound phases based on a per-thread counter
- For threads in a commit-bound phase, assign more entries in the ROB buffer, fewer entries in the Instruction Queues
- For threads in an issue-bound phase, assign fewer entries in ROB buffer, more entries in the Instruction Queues
- Stall fetching from threads over quota



Processor of the future?

- Fit into heterogeneous multi-core systems with one core dedicated to resource usage analysis and resource distribution.

The PArallel Systems & Computer Architectures Lab
http://pascal.eng.uci.edu