Adaptive Resource Partitioning in Simultaneous MultiThreading Architectures

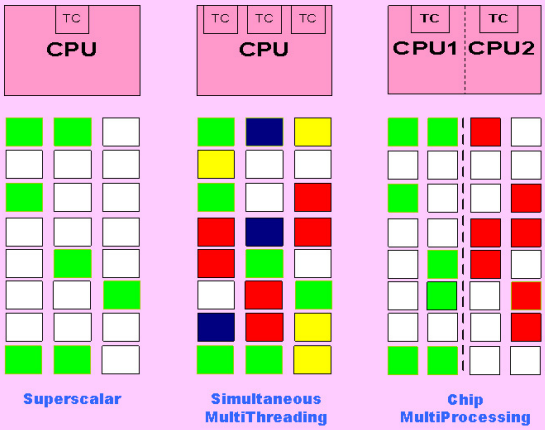
Chen Liu

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

The Henry Samueli School of Engineering

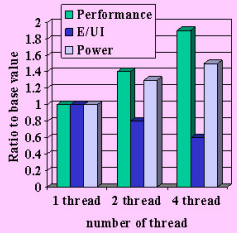
University of California, Irvine

General Processor Architecture

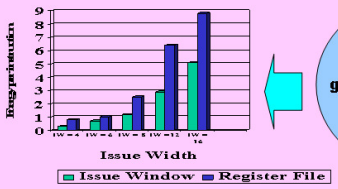


- SMT achieves BEST system resource utilization
- The key is to intelligently distribute the system resources among threads to achieve best performance

Power Concerns



Both the power consumption and performance grow though the energy per useful instruction decreases



Power consumption grows quadratically with the issue bandwidth

- To save on power, we must limit the issue bandwidth
- In the meantime, we want to maximize resource utilization

Research Goals

- To improve performance through intelligently distributing the resources among threads
- To serve for the next generation general purpose processors

Different Partitioning Schemes

Static Partitioning:

- Each thread has a pre-assigned cap when sharing resources

Static Partitioning: No thread can grab resources exclusively

Problem: Cannot respond to the different needs in different phases of execution

Dynamic Sharing:

- Let the thread compete for resources among themselves without pre-defined cap

Dynamic Sharing: Resources are dynamically distributed among threads

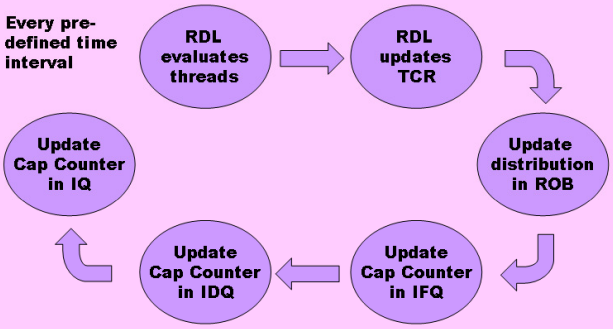
Problem: A thread may grab more resources it can use and clog the pipeline

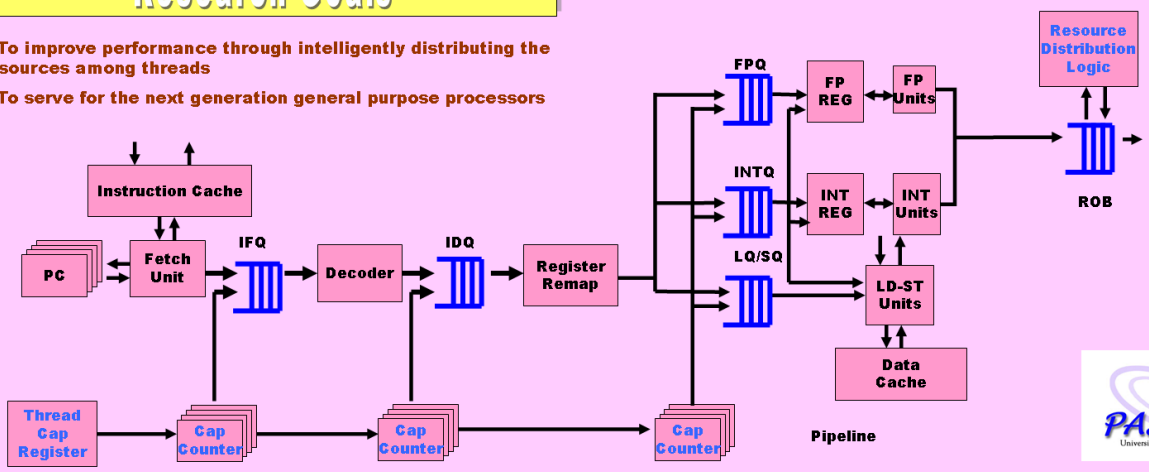
- However, the behavior of a thread varies with time
- No single scheme can be optimum all the time
- We need to distribute the system resources closely based on current performance of the thread

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

Adaptive Resource Partitioning

- Based on the ROB statistics, the Resource Distribution Logic (RDL) divides the threads into commit-bound and issue-bound, and updates the Thread-Cap Register (TCR).
- Commit-bound threads get fewer entries in the front-end queues, but more entries in ROB based on TCR
- Issue-bound threads get more entries in the front-end queues, but fewer entries in ROB based on TCR





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