How does an SMT work?

- What is private to each thread?
  - Context:
    - PC
    - Logical registers (RAT)
    - Address space (possibly shared)
  - Instruction buffer
  - Retirement (reorder)

SMT how does it work

- What is shared?
  - Functional units
  - Issue buffer
  - Reservation stations/physical registers
  - Branch predictor
  - Maybies:
    - Cache
    - TLB
    - Store buffer

What makes SMT a ‘good idea’?

- Optimal utilization of functional units
- Dynamic sharing of resources
- Throughput of instructions
- Flexible between singled and multithreaded performance
- +1 thread is “Free”

What is a CMP?

- Chip multiprocessor
- Bunch of processors on one die
- What do you share?
  - External bus
  - L2 cache
- What do you not share?
  - Functional units/cores/L1 caches
  - everything else

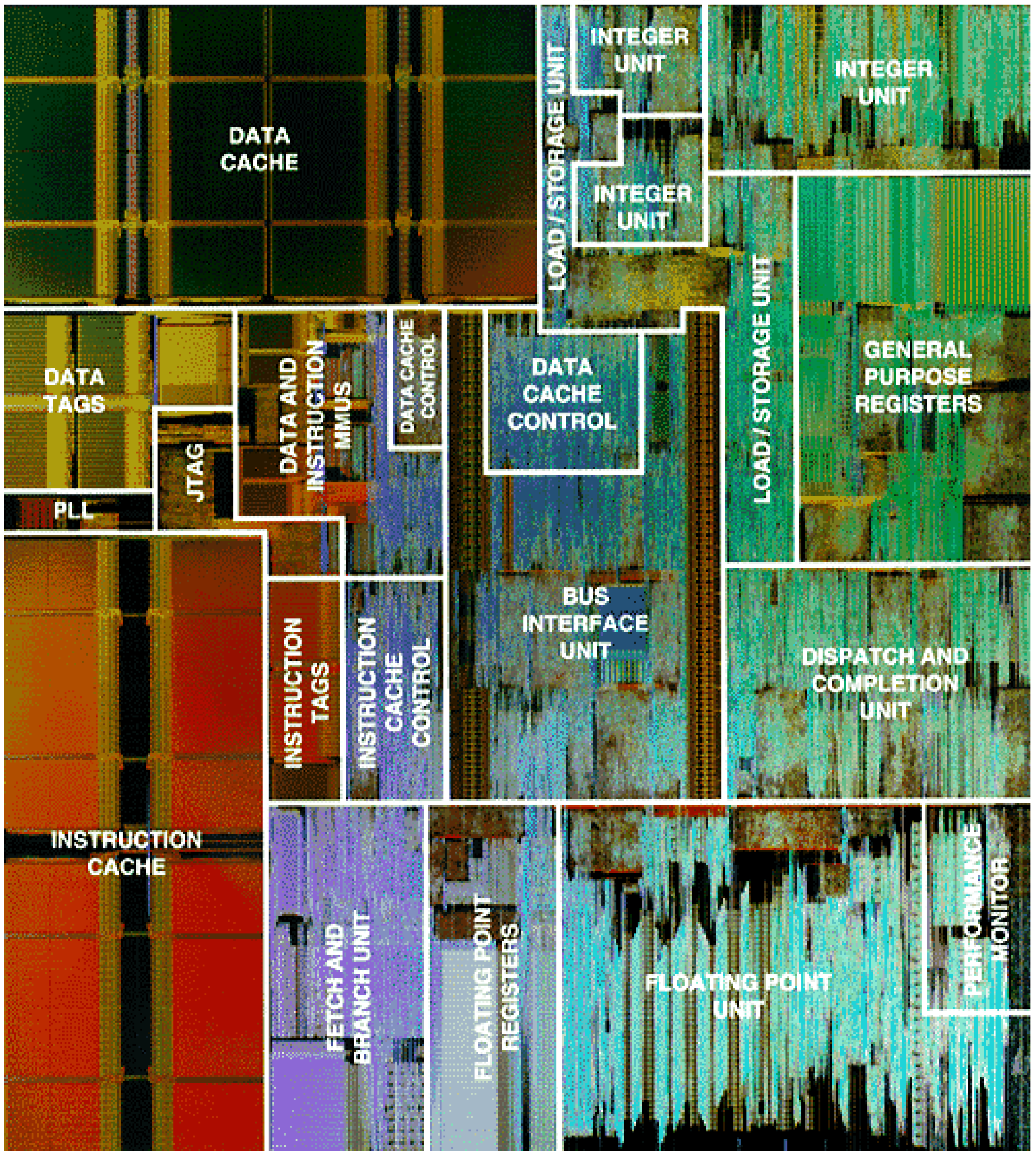
What makes CMP a ‘good idea’?

- Complexity control
- Clock distribution
- Scalability (some kinds)
- If you have a lot of threads… processses…
  - Webservers, databases, etc.

Where does all the silicon go anyway in a superscalar?

- Control logic
- Reservation stations
- Clock distribution
- Interconnect
- Cache
- Functional unit (5-6% today)

Motorola's PowerPC 604e™ RISC Microprocessor



What are the limits of SMT?

- 8
- 4
- Larger # threads => more parallelism
  - => more control logic
  - => more complexity
  - => more wire delay
- External ports to data caches are a problem
  - Can have a private cache to handle this

Does there have to be a distinction?

- Sharing => complexity
  - Gain: improved resource utilization
- Partitioning => simplification