

Levels of Parallelism within a Single Processor

- **ILP**: smallest grain of parallelism
 - Resources are not that well utilized (far from ideal CPI)
 - Stalls on operations with long latencies (division, cache miss)
- **Multiprogramming**: Several applications (or large sections of applications) running concurrently
 - O.S. directed activity
 - Change of application requires a context-switch (e.g., on a page fault)
- **Multithreading**
 - Main goal: tolerate latency of long operations without paying the price of a full context-switch

11/8/2001

Multithr. CSE 471 Autumn 01

1

Multithreading

- The processor supports several instructions streams running “concurrently”
- Each instruction stream has its own **context** (process state)
 - Registers
 - PC, status register, special control registers etc.
- The multiple streams are multiplexed by the hardware on a set of common functional units

11/8/2001

Multithr. CSE 471 Autumn 01

2

Fine Grain Multithreading

- Conceptually, at every cycle a new instruction stream dispatches an instruction to the functional units
- If enough instruction streams are present, long latencies can be hidden
 - For example, if 32 streams can dispatch an instruction, latencies of 32 cycles could be tolerated
- For a single application, requires highly sophisticated compiler technology
 - Discover many threads in a single application
- Basic idea behind Tera (now Cray) MTA
 - Burton Smith third such machine (he started in late 70's)

11/8/2001

Multithr. CSE 471 Autumn 01

3

Tera's MTA

- Each processor can execute
 - 16 applications in parallel (multiprogramming)
 - 128 streams
- At every clock cycle, processor selects a ready stream and issues an instruction from that stream
- An instruction is a “LIW”: Memory, arithmetic, control
- Several instructions of the same stream can be in flight simultaneously (ILP)
- Instructions of different streams can be in flight simultaneously (multithreading)

11/8/2001

Multithr. CSE 471 Autumn 01

4

Tera's MTA (c'ed)

- Since several streams belong to the same application, **synchronization** is very important (will be discussed later in the quarter)
- Needs instructions – and compiler support – to allocate, activate, and deallocate streams
- Compiler support: loop level parallelism and software pipelining
- Hardware support: dynamic allocation of streams (depending on mix of applications etc.)

11/8/2001

Multithr. CSE 471 Autumn 01

5

Coarse Grain Multithreading

- Switch threads (contexts) only at certain events
 - Change thread context takes a few (10-20?) cycles
 - Used when long memory latency operations, e.g., access to a remote memory in a shared-memory multiprocessor (100's of cycles)
- Of course, context-switches occur when there are exceptions such as page faults
- Many fewer contexts needed than in fine-grain multithreading

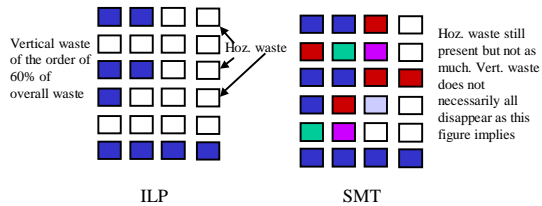
11/8/2001

Multithr. CSE 471 Autumn 01

6

Simultaneous Multithreading (SMT)

Combines the advantages of ILP and fine grain multithreading



11/8/2001

Multithr. CSE 471 Autumn 01

7

SMT (a UW invention)

- Needs one context per thread
 - But fewer threads needed than in fine grain multithreading
- Can **issue simultaneously** from distinct threads in the same cycle
- Can share resources
 - For example: physical registers for renaming, caches, BPT etc.
- Future generation Alpha was to be based on SMT
- Intel Hyperthreading is SMT with ... 2 threads

11/8/2001

Multithr. CSE 471 Autumn 01

8

SMT (c'ed)

- Compared with an ILP superscalar of same issue width
 - Requires 5% more real estate
 - More complex to design (thread scheduling, identifying threads that raise exceptions etc.)
- Drawback (common to all wide-issue processors): centralized design
- Benefits
 - Increases throughput of applications running concurrently
 - Dynamic scheduling
 - No partitioning of many resources (in contrast with chip multiprocessors)

11/8/2001

Multithr. CSE 471 Autumn 01

9

Speculative Multithreading

- A current area of research
- Several approaches
 - Try and identify "critical path" of instructions within a loop and have them run speculatively (in advance) of the main thread in further iterations
 - Span speculative threads on events such as:
 - Procedure call: continue with the caller and callee concurrently
 - In loops: generate a speculative thread from the loop exit
 - In hard to predict branches: have threads execute both sides
- Main advantage
 - Warm up caches and branch predictors.

11/8/2001

Multithr. CSE 471 Autumn 01

10

Trace Caches

- Filling up the instruction buffer of wide issue processors is a challenge (even more so in SMT)
- Instead of fetching from I-cache, fetch from a trace cache
- The trace cache is a complementary instruction cache that stores sequences of instructions organized in **dynamic program execution order**
- Implemented in forthcoming Intel Willamette and some Sun Sparc architecture.
- One way to do dynamic optimization of programs and to find critical path of instructions for speculative multithreading

11/8/2001

Multithr. CSE 471 Autumn 01

11