Parallel Processing SIMD, Vector and GPU's – cont.

EECS4201 Fall 2016 York University

1

Multithreading

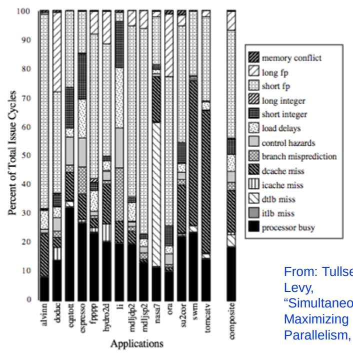
- First, we start with multithreading
- Multithreading is used in GPU's

2

Thread Level Parallelism

- ILP is used in straight line code or loops
- Cache miss (off-chip cache and main memory) is unlikely to be hidden using ILP.
- Thread level parallelism is used instead.
- Thread: process with own instructions and data
  - thread may be a process part of a parallel program of multiple processes, or it may be an independent program
  - Each thread has all the state (instructions, data, PC, register state, and so on) necessary to allow it to execute

3



From: Tullsen, Eggers, and Levy, "Simultaneous Multithreading: Maximizing On-chip Parallelism, ISCA 1995.

4

Thread Level Parallelism

- Multithreading: multiple threads to share the functional units of 1 processor via overlapping
  - processor must duplicate independent state of each thread e.g., a separate copy of register file, a separate PC, and for running independent programs, a separate page table
  - memory shared through the virtual memory mechanisms, which already support multiple processes
  - HW for fast thread switch; much faster than full process switch ≈ 100s to 1000s of clocks
- When to switch?
  - Alternate instruction per thread (fine grain)
  - When a thread is stalled, perhaps for a cache miss, another thread can be executed (coarse grain)

5

Fine-Grained Multithreading

- Switches between threads on each instruction, causing the execution of multiples threads to be interleaved
- Usually done in a round-robin fashion, skipping any stalled threads
- CPU must be able to switch threads every clock
- Advantage is it can hide both short and long stalls, since instructions from other threads executed when one thread stalls
- Disadvantage is it slows down execution of individual threads, since a thread ready to execute without stalls will be delayed by instructions from other threads
- Used on Sun's T1

6

Coarse-Grained Multithreading

- Switches threads only on costly stalls, such as L2 cache misses
- Advantages
  - Need to have very fast thread-switching
  - Doesn't slow down thread, since instructions from other threads issued only when the thread encounters a costly stall
- Disadvantage is hard to overcome throughput losses from shorter stalls, due to pipeline start-up costs
  - Since CPU issues instructions from 1 thread, when a stall occurs, the pipeline must be emptied or frozen
  - New thread must fill pipeline before instructions can complete
- Because of this start-up overhead, coarse-grained multithreading is better for reducing penalty of high cost stalls, where pipeline refill << stall time

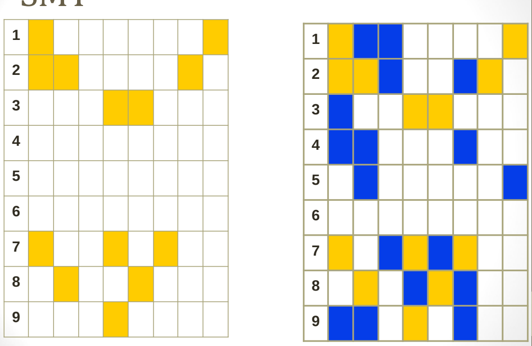
7

Simultaneous Multithreading

- Fine-grained multithreading implemented on top of multiple-issued dynamically scheduled processor.
- Multiple instructions from different threads.

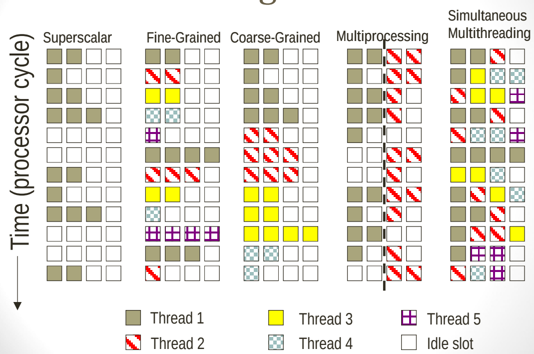
8

SMT



9

Multithreaading



10

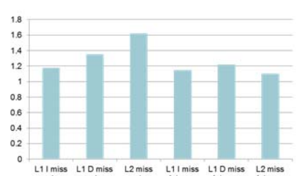
SUN T1

- Focused on TLP rather than ILP
- Fine-grained multithreading
- 8 cores, 4 threads per core, one shared FP unit.
- 6-stage pipeline (similar to MIPS with one stage for thread switching)
- L1 caches: 16KB I, 8KB D, 64-byte block size (misses to L2 23 cycles with no contention)
- L2 caches: 4 separate L2 caches each 750KB. Misses to main memory 110 cycles assuming no contention

11

SUN T1

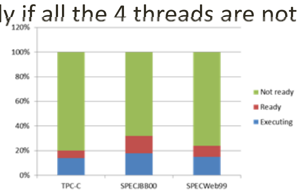
- Relative change in the miss rate and latency when executing one thread per core vs 4 threads per core (TPC-C)



12

SUN T1

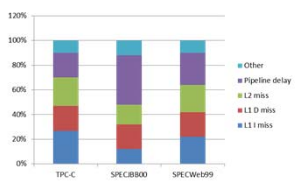
- Breakdown of the status on an average thread. Ready means the thread is ready, but another one is chosen – The core stalls only if all the 4 threads are not ready



13

SUN t1

- Breakdown of the causes for a thread being not ready



14

Graphics Processing Unite

- Started as graphics accelerators
- Given the investment we made in GPUs, can we use it for other things?
- Offers multithreading, SIMD, MIMD and ILP.
- NVIDIA developed CUDA (Compute Unified Device Architecture) to generate code for CPU (host) and GPU (device).
- SIMT (Single Instruction Multiple Threads) SIMD is not exposed to the programmer

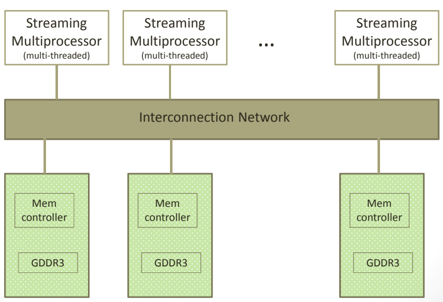
15

Programming the GPU

- CUDA uses __device or __global and __host
- Functions defined as __device or __global are allocated to GPU
- Functions defined as __host are allocated to the CPU

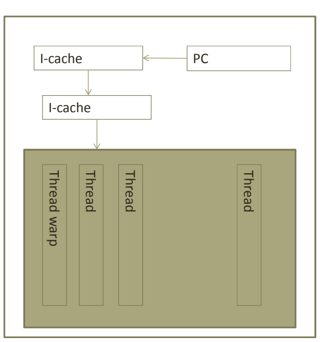
16

GPU



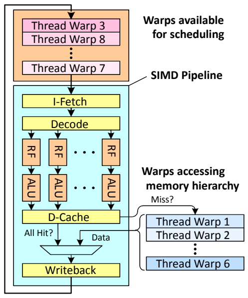
17

Streaming Multiprocessor

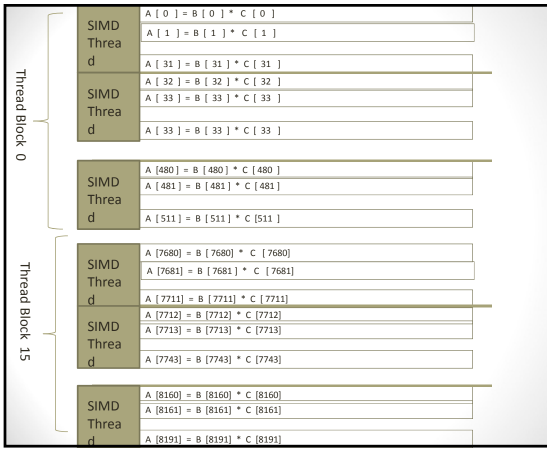


18

Streaming MultiProcessor



19



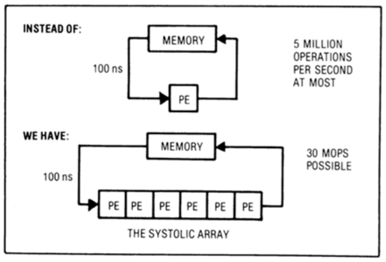
20

Systolic Arrays

- Idea: memory is a bottleneck, once we access a datum from the memory, fully utilize it before return it to the memory again
- Data flowing in a "rhythm" being processed
- Similar to an assembly line
  - Different people work on the same car
  - Many cars are assembled simultaneously
  - Can be two-dimensional
- Advantages:
  - Simple, regular designs (keep # unique parts small and regular)
  - High concurrency → high performance
  - Balanced computation and I/O (memory access)
  - Used in accelerators

21

Systolic Arrays



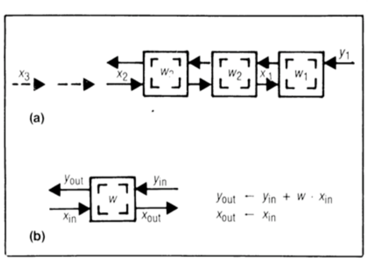
22

Example -- Convolution

- $Y_i = w_1 x_i + w_2 x_{i+1} + w_3 x_{i+1} + \ldots$

23

Convolution



24