Lecture 6 - HW and SW Parallelism

Mattan Erez

UT ECE

The University of Texas at Austin
Outline

• Parallel HW (multiple ALUs)
  – Analyze by shared resources
  – Analyze by synch/comm mechanisms
  – ILP, DLP, and TLP organizations

• Parallelism in SW
  – ILP/DLP/TLP?

• Parallel Programming
  – Design patterns
Parallel Execution

• Concurrency
  – what are the multiple resources?

• Communication
  – and storage

• Synchronization

• what is being **shared**?

• What is being **partitioned**?
Pipelining Summary

• Pipelining is using parallelism to hide latency
  – Do useful work while waiting for other work to finish
• Multiple parallel components, not multiple instances of same component
• Examples:
  – Execution pipeline
  – Memory pipelines
    • Issue multiple requests to memory without waiting for previous requests to complete
  – Software pipelines
    • Overlap different software blocks to hide latency: computation/communication
Resources in a parallel processor/system

- **Execution**
  - ALUs
  - Cores/processors

- **Control**
  - Sequencers
  - Instructions
  - OOO schedulers

- **State**
  - Registers
  - Memories

- **Networks**
Communication and synchronization

- **Synchronization**
  - Clock – explicit compiler order
  - Explicit signals (e.g., dependences)
  - Implicit signals (e.g., flush/stall)
    - More for pipelining than multiple ALUs

- **Communication**
  - Bypass networks
  - Registers
  - Memory
  - Explicit (over some network)
Organizations for ILP (for multiple ALUs)
Superscalar (ILP for multiple ALUs)

- sequencer
- scheduler
- memory hierarchy

How many ALUs?
Superscalar (ILP for multiple ALUs)

- Synchronization
  - Explicit signals (dependences)
- Communication
  - Bypass, registers, mem
- Shared
  - Sequencer, OOO, registers, memories, net, ALUs
- Partitioned
  - Instructions

How many ALUs?
SMT/TLS (ILP for multiple ALUs)
SMT/ TLS (ILP for multiple ALUs)

Why is this ILP? How many threads?
SMT/ TLS (ILP for multiple ALUs)

- **Synchronization**
  - Explicit signals (dependences)
- **Communication**
  - Bypass, registers, mem
- **Shared**
  - OOO, registers, memories, net, ALUs
- **Partitioned**
  - Sequencer, Instructions, arch. registers

Why is this ILP? How many threads?
VLIW (ILP for multiple ALUs)
VLIW (ILP for multiple ALUs)

How many ALUs?
VLIW (ILP for multiple ALUs)

- **Synchronization**
  - Clock+compiler

- **Communication**
  - Registers, mem, bypass

- **Shared**
  - Sequencer, OOO, registers, memories, net

- **Partitioned**
  - Instructions, ALUs

How many ALUs?
Explicit Dataflow (ILP for multiple ALUs)
Explicit Dataflow (ILP for multiple ALUs)
Explicit Dataflow (ILP for multiple ALUs)

- **Synchronization**
  - Explicit signals

- **Communication**
  - Registers + explicit

- **Shared**
  - Sequencer, memories, net

- **Partitioned**
  - Instructions, OOO, ALUs
DLP for multiple ALUs

From HW – this is SIMD
SIMD (DLP for multiple ALUs)
**SIMD (DLP for multiple ALUs)**

- **Synchronization**
  - Clock+compiler

- **Communication**
  - Explicit

- **Shared**
  - Sequencer, instructions

- **Partitioned**
  - Registers, memories, ALUs
  - Sometimes: memories, net
Vectors (DLP for multiple ALUs)

Vectors: memory addresses are part of single-instruction and not part of multiple-data
TLP for multiple ALUs
MIMD - shared memory (TLP for multiple ALUs)

- Sequencer
  - Scheduler
    - Memory hierarchy

Diagram showing the components of a MIMD architecture with shared memory and a scheduler for multiple ALUs.
MIMD - shared memory (TLP for multiple ALUs)

- Synchronization
  - Explicit, memory
- Communication
  - Memory
- Shared
  - Memories, net
- Partitioned
  - Sequencer, instructions, OOO, ALUs, registers, some nets

memory hierarchy
MIMD - distributed memory

sequencer

scheduler

memory hierarchy

sequencer

scheduler

memory hierarchy

sequencer

scheduler

memory hierarchy

sequencer

scheduler

memory hierarchy
MIMD - distributed memory

- Synchronization
  - Explicit

- Communication
  - Explicit

- Shared
  - Net

- Partitioned
  - Sequencer, instructions, OOO, ALUs, registers, some nets, memories
### Summary of Communication and Synchronization

<table>
<thead>
<tr>
<th>Style</th>
<th>Synchronization</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td>explicit signals (RS)</td>
<td>registers + bypass</td>
</tr>
<tr>
<td>VLIW</td>
<td>clock + compiler</td>
<td>registers (bypass?)</td>
</tr>
<tr>
<td>Dataflow</td>
<td>explicit signals</td>
<td>registers + explicit</td>
</tr>
<tr>
<td>SIMD</td>
<td>clock + compiler</td>
<td>explicit</td>
</tr>
<tr>
<td>MIMD</td>
<td>explicit signals</td>
<td>memory + explicit</td>
</tr>
</tbody>
</table>
## Summary of communication and synchronization

<table>
<thead>
<tr>
<th>Style</th>
<th>Synchronization</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLIW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataflow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIMD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Summary of sharing in ILP HW

<table>
<thead>
<tr>
<th>Style</th>
<th>Seq</th>
<th>Inst</th>
<th>OOO</th>
<th>Regs</th>
<th>Mem</th>
<th>ALUs</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td>S</td>
<td>P</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>SMT/TLA</td>
<td>P</td>
<td>P</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>VLIW</td>
<td>S</td>
<td>P</td>
<td>N/A</td>
<td>S</td>
<td>S</td>
<td>P</td>
<td>S</td>
</tr>
<tr>
<td>Dataflow</td>
<td>B</td>
<td>P</td>
<td>P</td>
<td>B</td>
<td>S</td>
<td>P</td>
<td>S</td>
</tr>
</tbody>
</table>
## Summary of sharing in ILP HW

<table>
<thead>
<tr>
<th>Style</th>
<th>Seq</th>
<th>Inst</th>
<th>OOO</th>
<th>Regs</th>
<th>Mem</th>
<th>ALUs</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Superscalar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMT/TLS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLIW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Summary of sharing in DLP and TLP

<table>
<thead>
<tr>
<th>Style</th>
<th>Seq</th>
<th>Inst</th>
<th>OOO</th>
<th>Regs</th>
<th>Mem</th>
<th>ALUs</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIMD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Summary of sharing in DLP and TLP

<table>
<thead>
<tr>
<th>Style</th>
<th>Seq</th>
<th>Inst</th>
<th>OOO</th>
<th>Regs</th>
<th>Mem</th>
<th>ALUs</th>
<th>Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector</td>
<td>S</td>
<td>S</td>
<td>N/A</td>
<td>P</td>
<td>S</td>
<td>P</td>
<td>B</td>
</tr>
<tr>
<td>SIMD</td>
<td>S</td>
<td>S</td>
<td>N/A</td>
<td>P</td>
<td>p</td>
<td>P</td>
<td>B</td>
</tr>
<tr>
<td>MIMD</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>S/P</td>
<td>P</td>
<td>B</td>
</tr>
</tbody>
</table>