Multithreaded Architectures
Lecture 3 of 4

Supercomputing ’93 Tutorial

Friday, November 19, 1993
Portland, Oregon

Rishiyur S. Nikhil

Digital Equipment Corporation
Cambridge Research Laboratory

Copyright (c) 1992, 1993 Digital Equipment Corporation
Overview of Lecture 3

Multithreading: a Dataflow Story

Dataflow graphs as a machine language

MIT Tagged Token Dataflow Architecture  Manchester Dataflow

Explicit Token Store Machines

MIT/Motorola Monsoon  ETL EM−4

P−RISC: "RISC−ified" dataflow

MIT/Motorola *T
Dataflow Graphs as a Machine Language

Code:
Dataflow Graph (interconnection of instructions)

Memories:
Two major components

Heap, for data structures (e.g., l-structure memory)

"Stack" (contexts, frames, ...)

- Conceptually, "tokens" carrying values flow along the edges of the graph.
- Values on tokens may be memory addresses
- Each instruction:
  - Waits for tokens on all inputs (and possibly a memory synchronization condition)
  - Consumes input tokens
  - Computes output values based on input values
  - Possibly reads/writes memory
  - Produces tokens on outputs
- No further restriction on instruction ordering

A common misconception:
"Dataflow graphs and machines cannot express side effects, and they only implement functional languages"
Encoding Dataflow Graphs and Tokens

Conceptual

Encoding of graph
Program memory:

<table>
<thead>
<tr>
<th>OpCode</th>
<th>Destination(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>op1 120L</td>
</tr>
<tr>
<td>113</td>
<td>op2 120R</td>
</tr>
<tr>
<td>120</td>
<td>+ 141, 159L</td>
</tr>
<tr>
<td>141</td>
<td>op3</td>
</tr>
<tr>
<td>159</td>
<td>op4 ...</td>
</tr>
</tbody>
</table>

Encoding of token:
A "packet" containing:

- 120R 6.847 Destination instruction address, Left/Right port
- Value

Re-entrancy ("dynamic" dataflow):

- Each invocation of a function or loop iteration gets its own, unique, "Context"
- Tokens destined for same instruction in different invocations are distinguished by a context identifier

- 120R Ctxt 6.847 Destination instruction address, Left/Right port
- Context Identifier
- Value
MIT Tagged Token Dataflow Architecture

- Designed by Arvind et. al., MIT, early 1980’s
- Simulated; never built

- Global view:

  Processor Nodes
  (including local program and "stack" memory)

  Interconnection Network

  "I-Structure" Memory Nodes (global heap data memory)

  Resource Manager Nodes

- Resource Manager Nodes responsible for
  - Function allocation (allocation of context identifiers)
  - Heap allocation
  - etc.

- Stack memory and heap memory: globally addressed
MIT Tagged Token Dataflow Architecture Processor

- **Wait-Match Unit:**
  - Tokens for unary ops go straight through
  - Tokens for binary ops: try to match incoming token and a waiting token with same instruction address and context id
    - **Success:** Both tokens forwarded
    - **Fail:** Incoming token → Waiting Token Mem, Bubble (no-op) forwarded
MIT Tagged Token Dataflow Architecture Processor Operation

Output Unit routes tokens:
- Back to local Token Queue
- To another Processor
- To heap memory based on the addresses on the token

Tokens from network are placed in Token Queue