

Module-4
CSEN 3104
Lecture 23
02/09/2019

Dr. Debranjan Sarkar

Data Flow Architecture

Comparison of Dataflow & Control-flow computers

- Example: (Show the Dataflow graph)

input d, e, f

$c_0 = 0$

for i from 1 to 8 do

begin

$a_i := d_i \text{ div } e_i$

$b_i := a_i * f_i$

$c_i := b_i + c_{i-1}$

end

output a, b, c

Comparison of Dataflow & Control-flow computers

- 24 instructions (8 divides, 8 multiplies and 8 adds)
- Let 'add' takes 1 cycles, 'multiply' takes 2 cycles 'divide' takes 3 cycles
- Sequential execution on a control flow uniprocessor takes $(3+2+1)*8 = 48$ cycles
- Execution on a 4-processor dataflow computer takes 14 cycles (Show figure)
- The output c_8 is the last one to produce, due to its dependence on all the previous c values (c_1 to c_7)
- Parallel execution of the same set of computations on a shared-memory 4-processor system takes 14 cycles (Show figure) → no time advantage of dataflow execution over control flow execution
- Shared memory is used to hold intermediate results (s_i and t_i for $i = 1, 2, 3, 4$)

Control Flow vs. Data Flow Computers

- Control Flow Computers

- Conventional Von Neumann machines are “Control Flow Computers”
- Instructions are executed sequentially as controlled by a program counter
- Sequential program execution is inherently slow
- Data is passed between instructions via references to shared memory cells
- Synchronous computations are performed, using centralized control

Control Flow vs. Data Flow Computers

- Data Flow Computers

- Basic concept of data-driven computations is to enable execution of an instruction whenever its required operands are available
- No program counters are needed
- Initiation of instructions depends on data availability, and not on the physical location of an instruction in the program
- Data-driven concept means asynchrony, that is many instructions can be executed simultaneously and asynchronously
- Data Flow Computers exploit maximum parallelism in a program
- The only constraint is the availability of hardware resources

Advantages and Disadvantages of DFA

- Advantages:
 - Very high potential for parallelism
 - High throughput
 - No memory sharing between instructions → No side effects (inconsistency)
- Disadvantages
 - Time lost waiting for unneeded arguments
 - High control overhead
 - Difficulty in manipulating data structures

Salient features of Data Flow Architecture

- The execution of an instruction is driven by the data availability instead of being guided by a program counter
- Theoretically, any instruction should be ready for execution whenever operands become available
- The instructions in a data-driven program are not ordered in any way
- Data is not stored separately in main memory, rather data are directly held inside instructions
- Intermediate or final results are passed directly as data tokens between instructions
- The data generated by an instruction will be duplicated into many copies and forwarded directly to all the instructions requiring this data

Salient features of Data Flow Architecture

- Data tokens, once consumed by an instruction, will no longer be available for reuse by other instructions
- Requires special mechanism to detect data availability
 - To match data tokens with needy instructions
 - To enable the chain reaction of asynchronous instruction execution
- No memory sharing between instructions results in no side effects
- As soon as the operands are available, the instructions are executed if the functional units are available
- Asynchrony implies the need for handshaking or token-matching operations
- Exploits fine-grained parallelism at the instruction level

Fine, coarse and medium grained parallelism

- Given 10 X 10 image for processing
- Processing of the 100 pixels is independent of each other
- **Fine-grained parallelism:**
 - 100 processors responsible for processing the 10*10 image
 - Ignoring the communication overhead, the 100 processors can process the 10*10 image in 1 clock cycle
 - Each processor is working on 1 pixel of the image and then communicates the output to other processors
- **Medium-grained parallelism:**
 - 25 processors processing the 10*10 image
 - The processing of the image will now take 4 clock cycles
- **Coarse-grained parallelism:**
 - Number of processors is reduced to 2
 - The processing will take 50 clock cycles
 - Each processor need to process 50 elements which increases the computation time
 - But the communication overhead decreases as the number of processors which share data decreases

Example Dataflow Processor (Dennis machine)

- Two data flow projects at MIT, USA
 - Dennis Machine (static architecture)
 - Arvind machine (dynamic architecture)
- Dennis machine
 - Must follow a static execution rule
 - Only one data token can occupy an arc at an instant
 - Static firing rule that an instruction is enabled if
 - A data token is present on each of its input arcs, and
 - No token is on any of its output arcs
 - So, the program graph contains control tokens as well as data tokens, both contributing to the enabling of an instruction
 - The control tokens act as an acknowledgement signals when data tokens are removed from output arcs

Example Dataflow Processor (Arvind machine)

- Based on dynamic architecture
- Data tokens are tagged (labelled or coloured) to allow multiple tokens to appear simultaneously on any input arc of an operator node
- No control tokens are needed to acknowledge the transfer of data tokens among instructions
- Rather, the matching of token tags is performed to merge them for instructions requiring more than one operand token
- Additional h/w required to attach tags onto data tokens and to perform tag matching
- It uses an $N \times N$ packet switch network for inter-PE communication
- It consists of N numbers of PEs, where each PE is a complete computer with an instruction set, a memory, tag-matching hardware etc.

Thank you