Dataflow Model of Computation

(From Dataflow to Multithreading)

Guang R. Gao

ACM Fellow and IEEE Fellow
Endowed Distinguished Professor
Electrical & Computer Engineering
University of Delaware

ggao@capsl.udel.edu
Coarse-Grain thread- The family home model

Fine-Grain non-preemptive thread- The “hotel” model

Coarse-Grain vs. Fine-Grain Multithreading

[Gao: invited talk at Fran Allen’s Retirement Workshop, 07/2002]
Evolution of Multithreaded Execution and Architecture Models

Non-dataflow based

- CDC 6600 (1964)
- Flynn’s Processor (1969)
- Cosmic Cube (Seitz 1985)
- HEP (B. Smith 1978)
- J-Machine (Dally 1988-93)
- MASA (Halstead 1986)
- M-Machine (Dally 1994-98)
- Alwife (Agarwal 1989-96)
- Eldorado
- CASCADE

Dataflow model inspired

- MIT TTDA (Arvind 1980)
- LAU (Syre 1976)
- Manchester (Gurd & Watson 1982)
- Arg-Fetching Dataflow (DennisGao 1987-88)
- Monsoon (Papadopoulos & Culler 1988)
- Iannuci’s (1988-92)
- P-RISC (Nikhil & Arvind 1989)
- TAM (Culler 1990)
- SIGMA-I (Shimada 1988)
- MDFA (Gao 1989-93)
- MTA (Theobald 1990)
- Tera (B. Smith 1990-93)
- T/Start-NG (MIT/Motorola 1991-)
- Cilk (Leiserson)
- *T/Start-NG (MIT/Motorola 1991-)


9/22/2011
The Von Neumann-type Processing

```
begin
  for i = 1 ...
  ...
  endfor
end
```

Source Code

Compiler

Sequential Machine Representation

Processor

Load

CPU
A Multithreaded Architecture
McGill Data Flow Architecture Model (MDFA)
Argument -flow Principle

Argument -fetching Principle
A Dataflow Program Tuple

Program Tuple = \{ P-Code . S-Code \}

P-Code

N1: x = a + b;
N2: y = c – d;
N3: z = x * y;

S-Code

\[\begin{align*}
\text{n1} & \leftarrow 2 \\
\text{n2} & \leftarrow 2 \\
\end{align*}\]

\[\begin{align*}
a & \rightarrow \text{n1} \\
b & \rightarrow \text{n1} \\
c & \rightarrow \text{n2} \\
d & \rightarrow \text{n2} \\
\text{n1} & \rightarrow 2 \\
\end{align*}\]
The McGill Dataflow Architecture Model

Pipelined Instruction Processing Unit (PIPU)

Dataflow Instruction Scheduling Unit (DISU)

Enable Memory & Controller

Signal Processing

Fire

Done

9/22/2011
The McGill Dataflow Architecture Model

**Important Features**

Pipeline can be kept fully utilized provided that the program has sufficient parallelism.
The Scheduling Memory (Enable)

Dataflow Instruction Scheduling Unit (DISU)

Enabled Instructions
Waiting Instructions

Fire
Done

Count Signal(s)
Signal Processing

1

9/22/2011
Topic-Gao-Dataflow-part3
Advantages of the McGill Dataflow Architecture Model

- Eliminate unnecessary token copying and transmission overhead
- Instruction scheduling is separated from the main datapath of the processor (e.g. *asynchronous*, *decoupled*)
A sequence of instructions is “packed” into a macro-dataflow node.

Synchronization is done at the macro-node level.
Hybrid Evaluation Von Neumann Style Instruction Execution” on the McGill Dataflow Architecture

- Group a “sequence” of dataflow instruction into a “thread” or a macro dataflow node.
- Data-driven synchronization among threads.
- “Von Neumann style sequencing” within a thread.

**Advantage:**
Preserves the parallelism among threads but avoids unnecessary fine-grain synchronization between instructions within a sequential thread.
What Do We Get?

• A hybrid architecture model without sacrificing the advantage of fine-grain parallelism!

(latency-hiding, pipelining support)
A Realization of the Hybrid Evaluation

Pipelined Instruction Processing Unit (PIPU)

Dataflow Instruction Scheduling Unit (DISU)

Shortcut

Fire

Done

Von Neumann bit

1 2 \cdots k
Case Studies – Dataflow Model Insired Multithreading

• McGill Dataflow Model (1988 - 1993)

• EARTH Model (1993 – mid 2000s )

• The UHPC/Runnemede Model (2010 - )