Alternative Approaches
To
Concurrency
Outline

* Concurrency Basics
  -- Granularity
  -- SIMD/MIMD
  -- Supercomputers vs. Multi
  -- Data Flow vs. Control Flow

* Data Flow Basics
  -- Fire when ready
  -- Irregular parallelism
  -- Instances
  -- Example programs

* Single instruction stream
  -- SIMD (Vectors, Arrays)
  -- VLIW (now EPIc)
  -- DAE
  -- HPS

* MP Basics
  -- Metrics: Speedup, Redundancy, Efficiency
  -- Amdahl’s Law
  -- Cache Coherency (Consistency)
  -- Interconnection Networks
    (cost, latency, contention)

* NOT Single instruction stream
  -- cm* (NUMA)
  -- HEP (today, SMT)
  -- Hypercube
  -- Target-triggered (the MOV instruction)
  -- CMP
  -- Tiling the plane
    x early: nonVon, BVM, CM-1
    x today: TRIPS, Cell, Niagara, RAW, Wavescalar
Granularity of Concurrency

* **Intra-Instruction (Pipelining)**

* **Parallel Instructions (SIMD, VLIW)**

* **Tightly-coupled MP**

* **Loosely-coupled MP**
SIMD/MIMD

SISD  The Typical Pentium-Pro, for example
MISD
SIMD  Array Processor, Vector Processor
MIMD  Multiprocessor

and, Note:

Pipelined
SISD

SIMD
Pipelining

Pipelined:

\[
\begin{array}{cccc}
F_1 & D_1 & E_1 & S_1 \\
F_2 & D_2 & E_2 & S_2 \\
F_3 & D_3 & E_3 & S_3 \\
F_4 & D_4 & & \\
F_5 & & & \\
\end{array}
\]

Superscalar:

\[
\begin{array}{cccc}
F_1 & D_1 & E_1 & S_1 \\
F_2 & D_2 & E_2 & S_2 \\
F_3 & D_3 & E_3 & S_3 \\
F_4 & D_4 & E_4 & S_4 \\
F_5 & D_5 & E_5 & S_5 \\
F_6 & D_6 & E_6 & S_6 \\
F_7 & D_7 & E_7 & S_7 \\
F_8 & D_8 & E_8 & S_8 \\
F_9 & & & \\
\end{array}
\]

Superpipelined:

\[
\begin{array}{cccc}
F_1 & D_1 & E_1 & S_1 \\
F_2 & D_2 & E_2 & S_2 \\
F_3 & D_3 & E_3 & S_3 \\
F_4 & D_4 & E_4 & S_4 \\
F_5 & D_5 & E_5 & S_5 \\
F_6 & D_6 & E_6 & S_6 \\
\end{array}
\]
One Supercomputer

vs.

“The Multi”

(...Except Even Supercomputers have adopted the multi approach)

\[ 1 \times 2^n \quad 2^k \times 2^{n-k} \quad 2^n \times 1 \]

Why do we care?

-- Economic Answer
-- Strategic Answer
-- Scientific Answer

Scalability
Amdahl's Law

* Speed-up as a function of the parallelizability ($\alpha$) of the application

Speedup vs. Parallelizability for a given number of processors ($p$)

* Speed-up of an application as we add more and more processors ($p$)

Speedup vs. Number of Processors ($p$) for a given $\alpha$
MP vs. Multicomputer Network

* Shared memory vs. Message passing

* Easier for software, or easier for hardware

* No free lunch
  - Cache Consistency
  - Memory Contention
A Unit of Computation:
The Data Flow Node

OR,

| * | R | ARG1 | R | ARG2 | Dest. Of Result |

The Operation
(In Larger Granularity Systems, "The Compound Function")

Fires When Ready
The Firing Rule:

When all inputs have tokens

(Note: Safe vs. Queues)

*Conditional

*Relational

*Barrier Synch
An Example Data Flow Program: 
**Factorial** (Done, Iteratively)
Characteristics of Data Flow

* Data Driven Execution of Instruction-level Graphical Code
  -- Nodes are Operators
  -- Arcs are I/O

* Only REAL Dependencies Constrain Processing
  -- Anti-Dependencies Don't (write-after-read)
  -- Output Dependencies Don't (write-after write)
  -- NO Sequential I-stream (No PC)

* Operations Execute Asynchronously

* Instructions Do Not Reference Memory
  (at least, memory as we understand it)

* Execution Triggered By Presence of Data
  -- Safe vs. Queues
SIMD

Vector Processors, Array Processors

LD  *  @  ST

LD₁
LD₂ * 1
LD₃ * 2  @₁
LD₄ * 3  @₂  ST₁
   * 4  @₃  ST₂
       @₄  ST₃
           ST₄

1  2  3  4

LD₁  LD₂  LD₃  LD₄
   * 1  * 2  * 3  * 4
   @₁  @₂  @₃  @₄
ST₁  ST₂  ST₃  ST₄

time
VLIW

* Static Scheduling
  - Everything in lock step
  - Trace Scheduling

* Generic Model
Early Form of Decoupled - Access/Execute

* Andrew Plezskun, Univ. of Illinois
* SMA

* James E. Smith, Univ of Wisconsin
* DAE
HPS As Evolution

PE

a

PE PE PE ... PE

a a a ... a

PE PE PE ... PE

a β γ ... β

PE PE PE ... PE

a β γ ... β
HPS
(RESTRICTED DATA FLOW)

For example, the VAX instruction:

ADDL2 (R1) +, (R2)

VAX INSTRUCTION

\[ + 1 - 4 ! - 2000 \times \text{RD} \downarrow - 2000 \beta + \beta - \delta \times - \delta \text{RD} \delta \varepsilon - \varepsilon \text{WE} \approx \delta - \beta \]

Decoder

MERGER
```
\text{Decode

\begin{itemize}
\item \text{Mul R1, R3, R2}
\item \text{Addg (R1) +, (R2)}
\end{itemize}

\begin{itemize}
\item R1
\item R1
\item R2
\item WR
\item + 1-8
\item \text{RD}
\item R2
\item \text{RD}
\item \text{RD}
\item \text{RD}
\item \text{G+}
\item WR
\item OC
\item \text{RL}
\end{itemize}

\begin{itemize}
\item \text{Merge}
\end{itemize}

\begin{itemize}
\item \text{a la Tomasulo}
\item R1 1 - 2000
\item R2 \text{ax} -
\item R3 1 - 100
\end{itemize}

\begin{itemize}
\item + 1 - 8 1 - 2000 b (RD 1 - 2000 c RD \text{ax} - b \text{G+} \text{a} - \text{c} \text{WR} \text{OC} - \text{ax}
\end{itemize}

R1 \text{ax} -
R2 \text{ax} -
R3 1 - 100
```
HPS - What is it?

I-Stream

Active Window

MAchine LANG inst

Decoder

DATA FLOW GRAPH

MERCER

ISSUE

All the Nodes

\{LEVEL \}

\{FIRE\}

\{DIST\}

F.U.

\ldots

\ldots

\ldots

\ldots

Restricted Data Flow
Note: A well-meaning student told me to get rid of this slide. cm* is old. People will think you are an old man, and not take you seriously.
Cosmic Cube

(Example: k = 4)