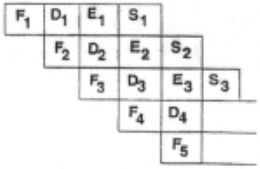
Single Thread Parallelism

Outline

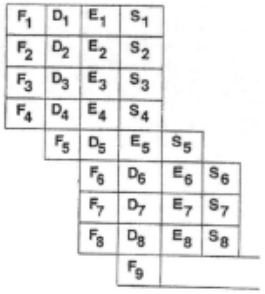
- ***Pipelining***
- ***SIMD (both vector and array processing)***
- ***VLIW***
- ***DAE***
- ***HPS***
- ***Data Flow***

Pipelining

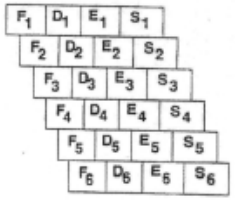
Pipelined:



Superscalar:



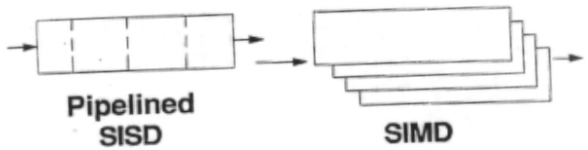
Superpipelined:



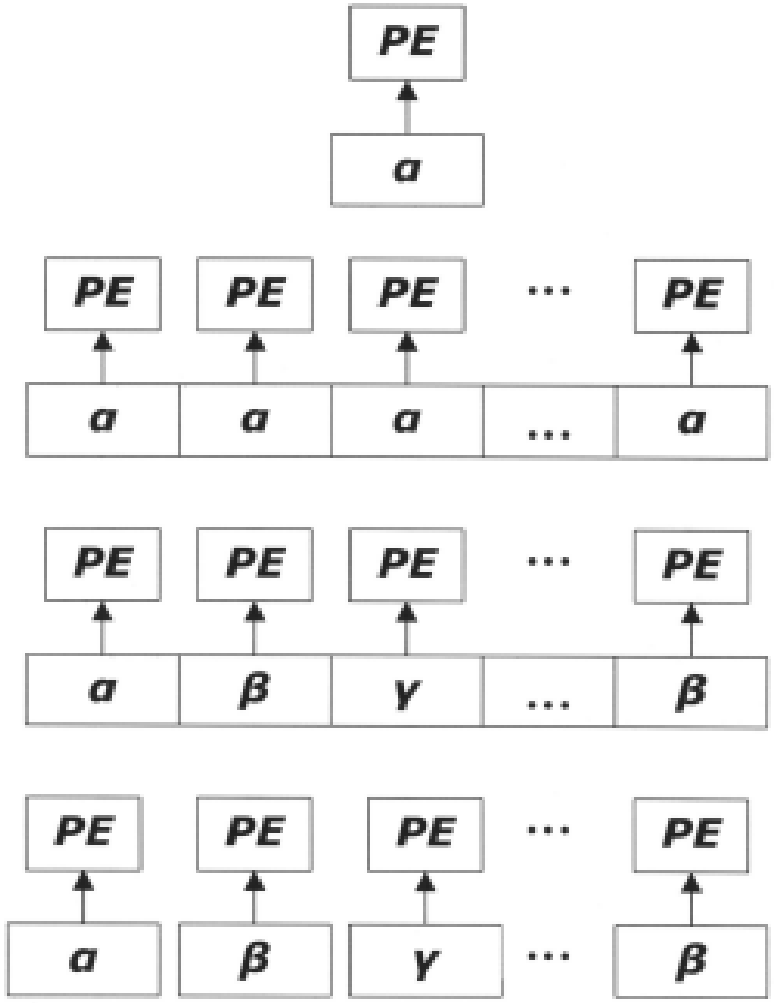
SIMD/MIMD

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

and, Note:

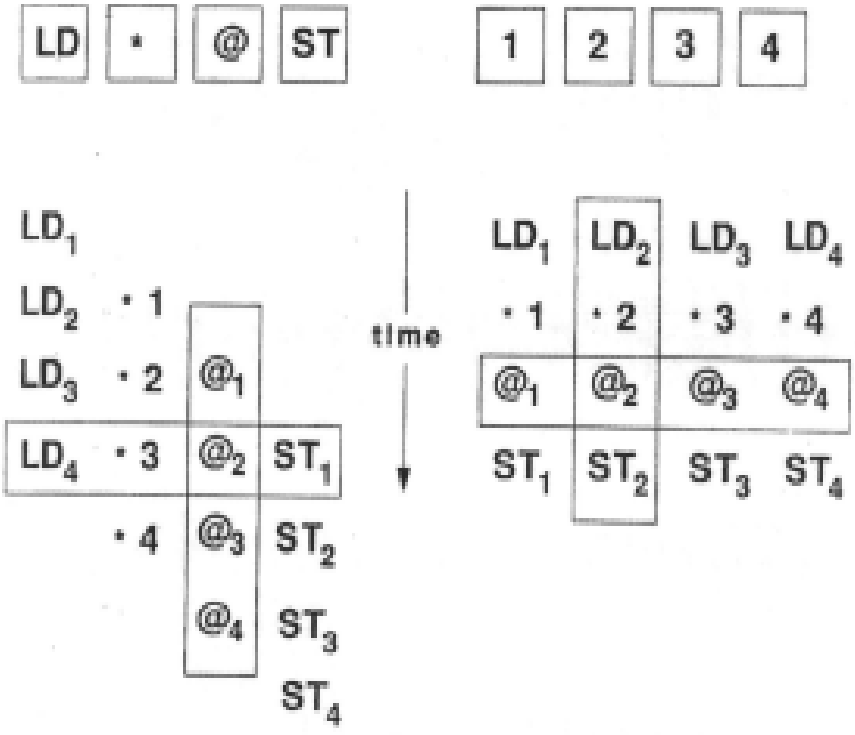


HPS As Evolution



SIMD

Vector Processors, Array Processors



An example: Vector processing

- ***The scalar code:***

  for i=1,50

  A(i) = (B(i)+C(i))/2;

- ***The vector code:***

  lvs 1

  lvl 50

  vld V0,B

  vld V1,C

  vadd V2,V0,V1

  vshf V3,V2,1

  vst   V3,A

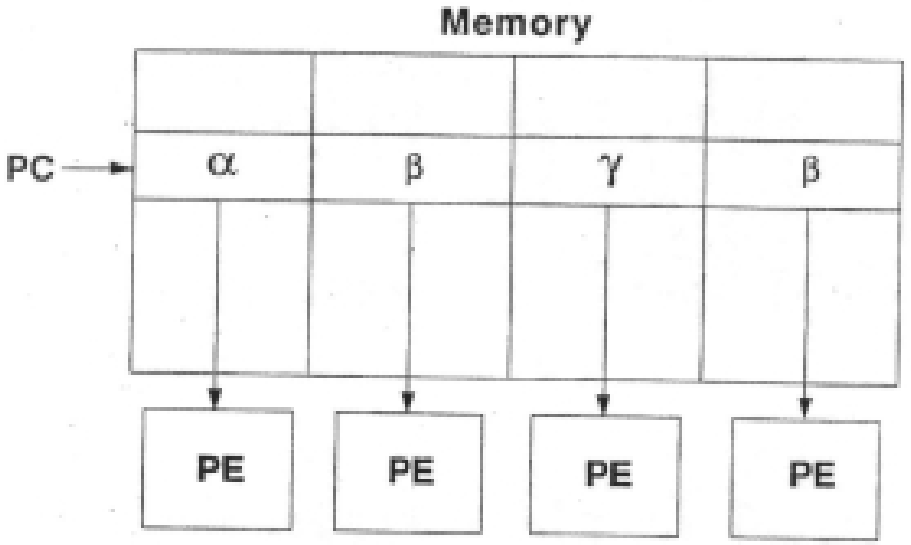
Vector processing example (continued)

- ***Scalar code (loads take 11 clock cycles):***
- ***Vector code (no vector chaining):***
- ***Vector code (with chaining):***
- ***Vector code (with 2 load, 1 store port to memory):***

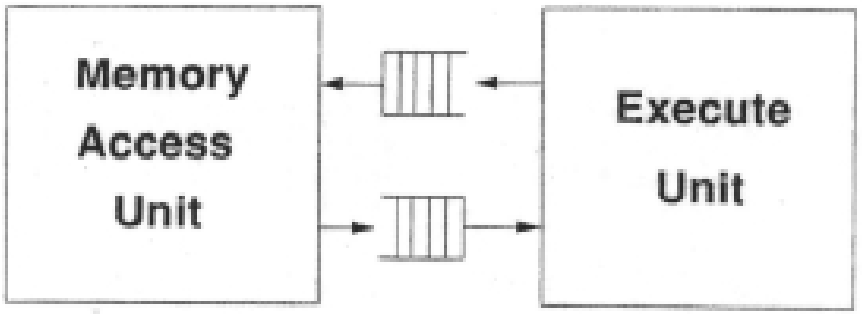
VLIW

* **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

The HPS Paradigm

- ***Incorporated the following:***
  - ***Aggressive branch prediction***
  - ***Speculative execution***
  - ***Wide issue***
  - ***Out-of-order execution***
  - ***In-order retirement***

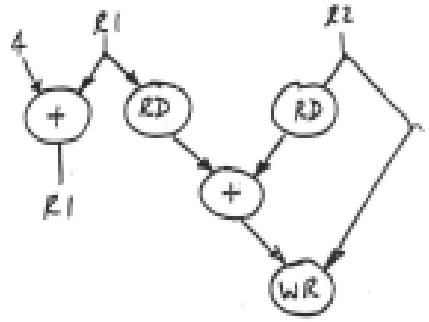
- ***First published in Micro-18 (1985)***
  - ***Patt, Hwu, Shebanow: Introduction to HPS***
  - ***Patt, Melvin, Hwu, Shebanow: Critical issues***

HPS

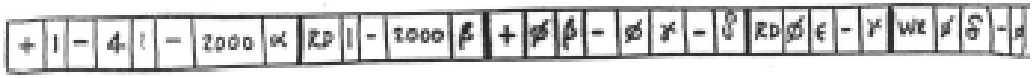
(Restricted Data Flow)

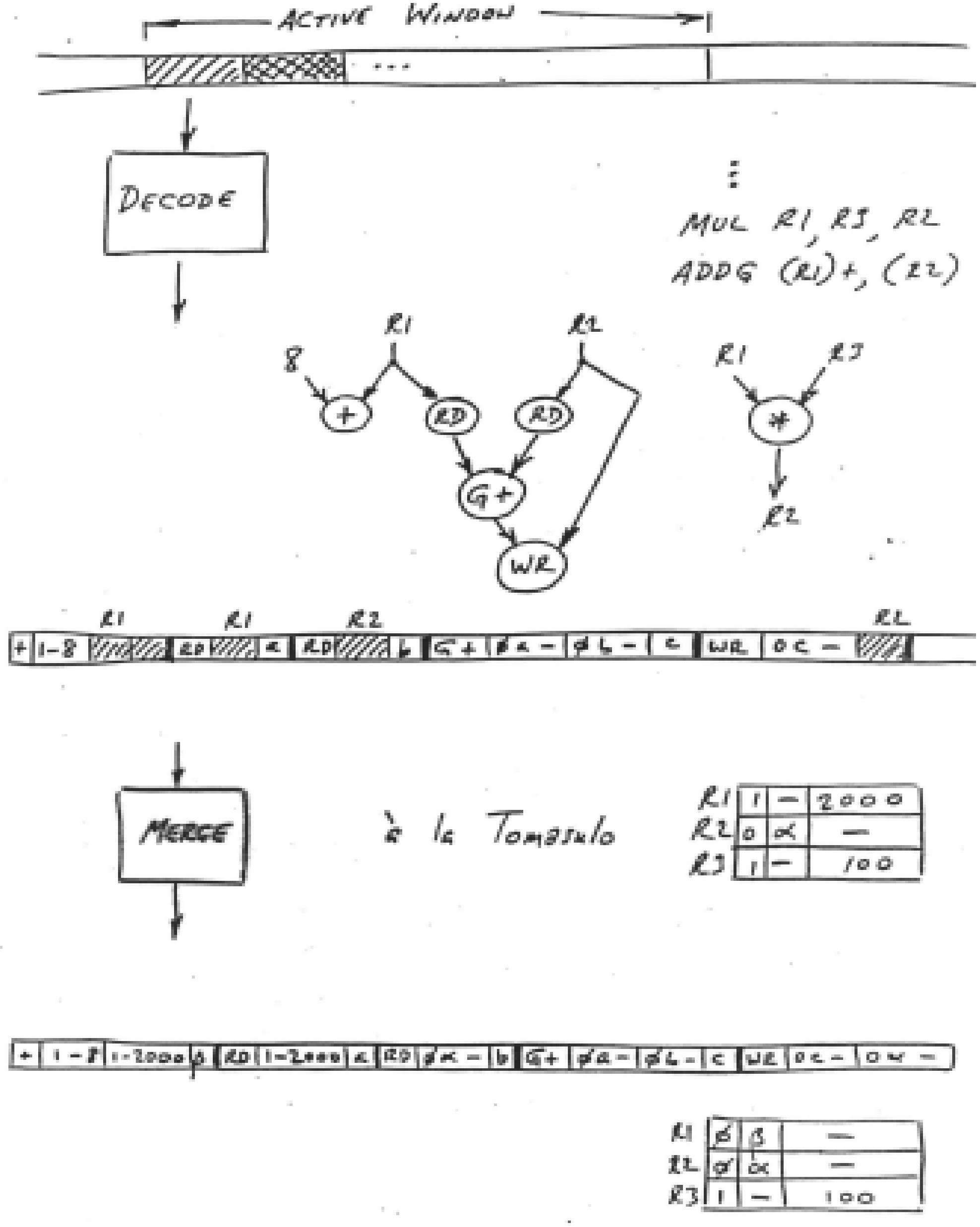
For Example, The VAX Instruction:

ADDL2 (R1)+, (R2)

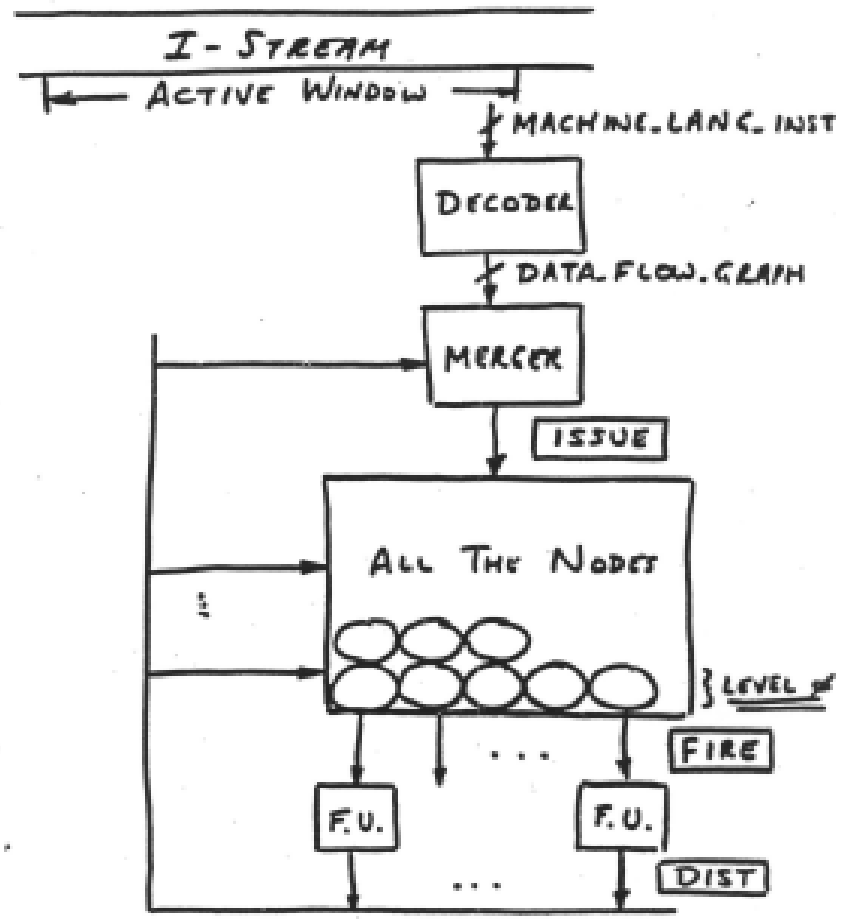


VAX Instruction



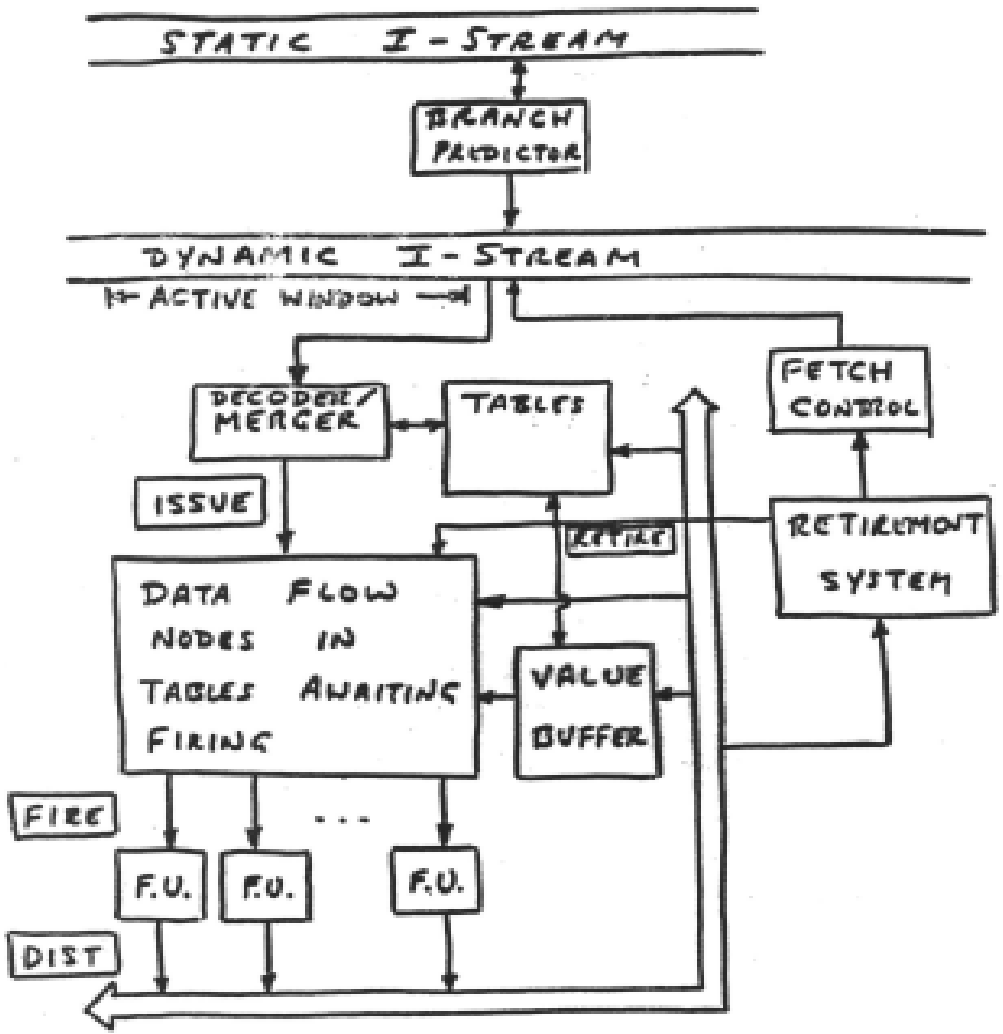

ACTIVE WINDOW
DECODE
MUL R1, R3, R2
ADDG (R1)+, (R2)
R1
R2
8
+
RD
RD
G+
WR
R1
R3
*
R2
MERGE
à la Tomasulo
R1 | 1 | — | 2000
R2 | 0 | α | —
R3 | 1 | — | 100
R1 | β | β | —
R2 | φ | α | —
R3 | 1 | — | 100

HPS – What Is It ?



* RESTRICTED DATA FLOW

HPS

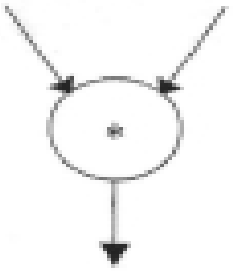

STATIC I-STREAM
BRANCH PREDICTOR
DYNAMIC I-STREAM
ACTIVE WINDOW
DECODER/ MERGER
TABLES
FETCH CONTROL
ISSUE
RETIREMENT SYSTEM
DATA FLOW NODES IN TABLES AWAITING FIRING
VALUE BUFFER
FIRE
F.U.
F.U.
F.U.
DIST

Data Flow

- ***Data Driven execution of inst-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 program counter)***
- ***Operations execute ASYNCHRONOUSLY***
- ***Instructions do not reference memory***
  - ***(at least memory as we understand it)***
- ***Execution is triggered by presence of data***

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")

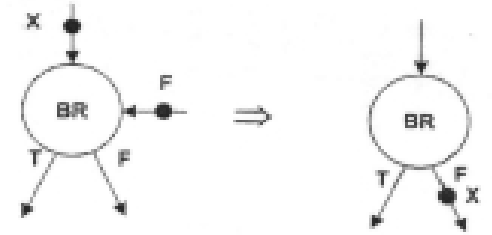
Fires
When
Ready

The Firing Rule:

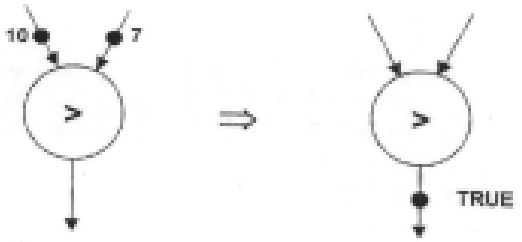
When all Inputs Have Tokens

(Note: Safe vs. Queues)

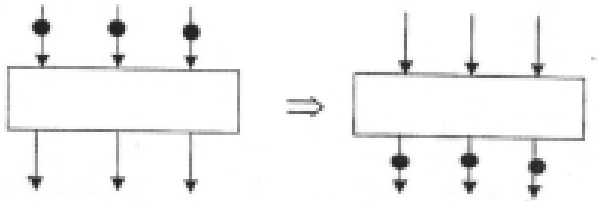
**Conditional*



**Relational*



*Barrier Synch



An Example Data Flow Program:

<u>Factorial</u> (Done, Iteratively)