an URI / NEU collaboration

LevoSim

or

the HPCA experience July 2001

student **David Morano**

advisors **Professor David Kaeli**

Professor Augustus Uht

NUCAR talk 01/08/03

Agenda

- **LevoSim background**
- **preparing for HPCA**
- **speed**
- **memory**
- **Levo machine bugs**
- **plan summary**

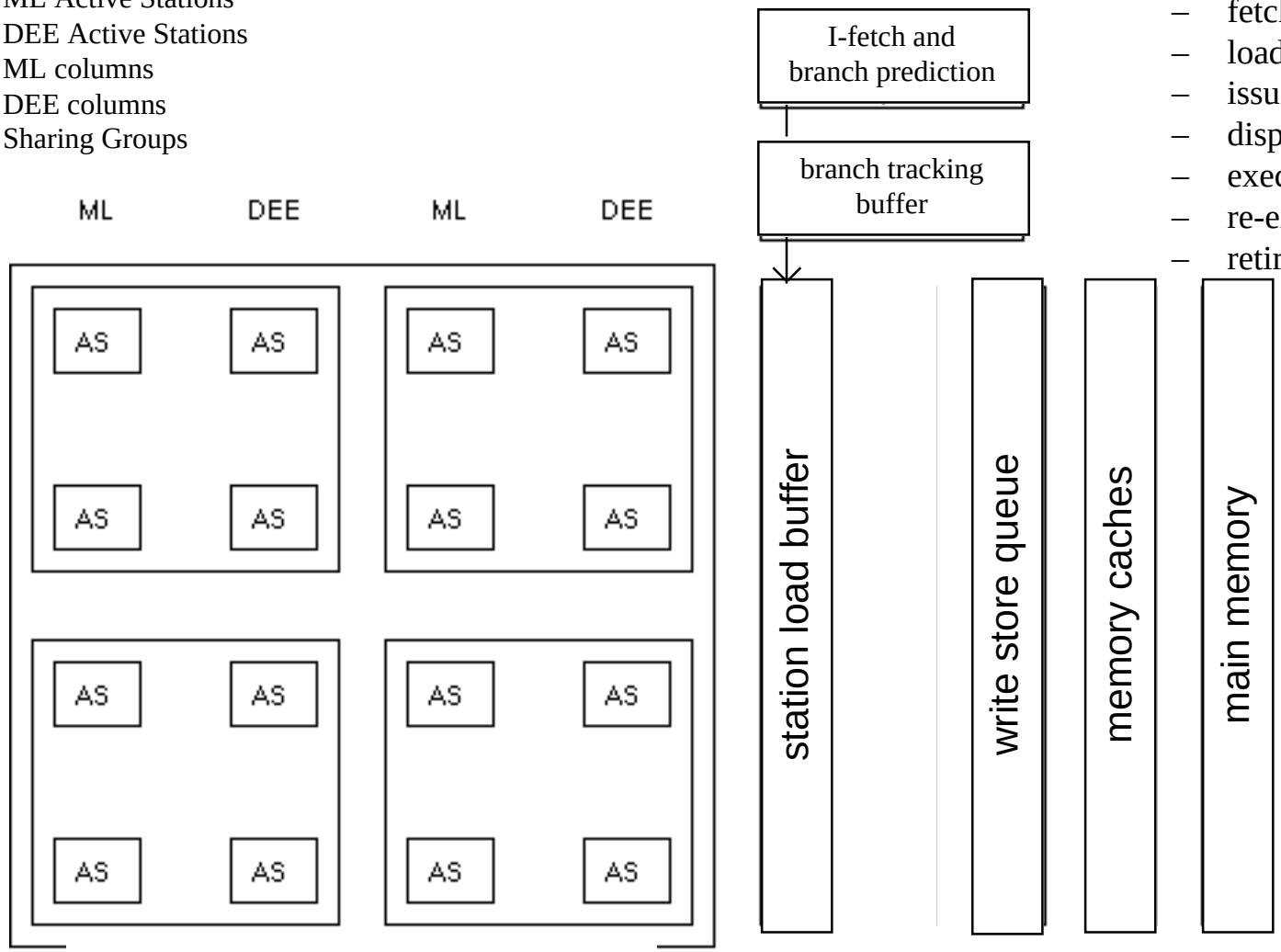
LevoSim background

- **hardware model for a Levo machine**
- **most high-level Levo hardware components are modeled**
  - **we modeled all of the ones needed for a minimal machine**
  - **no predicate forward units (PFUs)**
  - **no memory filter units (MFUs)**
- **most familiar components are modeled minimally if at all**
  - **caches**
  - **memory**
  - **instruction decode**
  - **instruction execution**
- **modeled structurally at the highest levels**
- **modeled behaviorally at all lower levels**
- **example "add" instruction (actual C language from the code) :**

```
dst1 = src1 + src2 ;
```

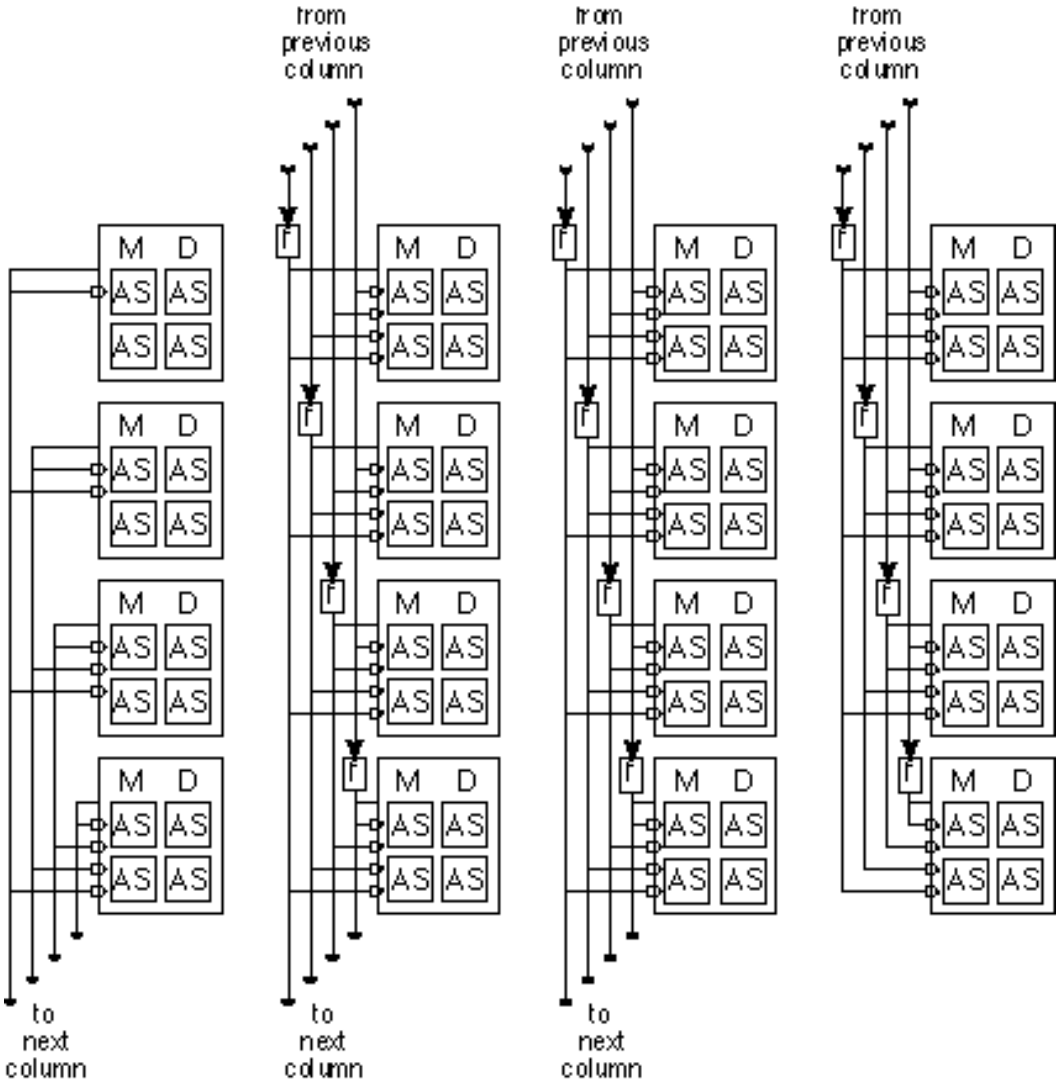
high-level block diagram

8 ML Active Stations
8 DEE Active Stations
2 ML columns
2 DEE columns
4 Sharing Groups



- **execution flow**
  - fetch
  - load
  - issue
  - dispatch
  - execute
  - re-execute !
  - retire

sharing group forwarding buses



preparing for HPCA

- LevoSim really failed to produce many results
- it can do most all configurations of Dhrystone (single-loop)
  - below a certain size Levo machine
- it failed on Dhrystone 100-loop
- it failed on all other Spec-2000 benchmark programs
  - there are also possible fetch trace flaws due to DBX
- it did produce three data points for BZIP2-maryam
  - only got something over 19k instructions before deadlocking
- there were memory size problems
- there were Levo machine bugs
- there were possible fetch trace flaws

speed issues

- LevoSim is slow !
- the simulation speed is related to the size of the Levo machine
  - types and numbers of components used
- the time is spent in the Levo specific components
- example machine 32 x 8 ASes w/ span of 32, 128-way memory
  - each AS has to snoop every clock :
    - 128 memory forwarding buses
    - 128 memory backwarding buses
    - 256 predicate forwarding buses
    - 32 register forwarding buses
    - 32 register backwarding buses
    - 1 PE result bus
  - most things have to process forwards, backwards, or execution requests most every clock

speed issue plan

- don't clock DEE path ASes when DEE paths are not being used
  - already doing this
  - also reduces memory resident size by allowing DEE path ASes to get paged out
- don't clock or otherwise service DEE path related components, sub-components, or registers when DEE paths are not being used
- work towards getting better IPC numbers out of the machine
  - less clocks needed for the same instructions executed
- find ways to reduce the number of transactions used
  - less processing burden on ASes
- implement MFUs and PFUs to reduce snooping burden on ASes
  - will probably also give an IPC performance increase

memory issues

- LevoSim uses up a lot of memory !
- we allocate memory only as needed and were very frugal
- the amount of memory needed is related to the size of the Levo machine being simulated
  - types and numbers of components
- we run out of heap space first
- we also can run out of swap space in certain circumstances
- we can run out of memory due to the limit of physical memory installed in the host computer
  - we need our entire working page set in core

memory issue plan

- get everything out of the heap as much as reasonably possible
  - will do this with hardware components at the higher structural levels of the Levo machine
  - I already did some of this while trying to get something for HPCA
- create our own swap space for all of our own managed memory objects
  - Levo hardware structural objects
  - target program writable data and stack
  - target program read-only memory is already being swapped back to its own object file
- do not allocate DEE path resources where possible (not always easy)
  - obviously not useful when we need DEE path stuff

Levo machine bugs

- we do not have core dumps anymore (we did in the past)
  - keep your fingers crossed !
- the remaining errors are design bugs in the Levo machine itself
  - unfortunately all hardware components must actually perform properly or the machine fails in some way
- large Levo machines (which sometimes tend to exhibit more Levo machine bugs) have bugs that can be difficult to track and find
- LevoSim currently has some restrictions that require specialized procedures
  - we currently can only execute from a fetch trace
  - the fetch trace must be correct

bug plan

- **develop tools to help with verification of correct program execution**
  - **SimpleSim**
  - **trace comparison and conversion tools**
  - **visualization ?**
- **develop a strategy for tracking bugs more like what logic analyzers do using trigger conditions**
- **obviously start by investigating the bugs that show up in smaller target programs and smaller Levo machine configurations**

plan summary

- finish current tool development
  - SimpleSim
  - trace stuff
  - integrate new stuff back into LevoSim
- work on some debugging code within LevoSim to facilitate bug exploration and investigation
- get some or all of the bugs out (the hard part !)
- work on reducing the memory footprint of LevoSim
- work on ways to increase the speed of LevoSim