FLINT
Presented 1/15/2009
RAMP Retreat, Berkeley, CA

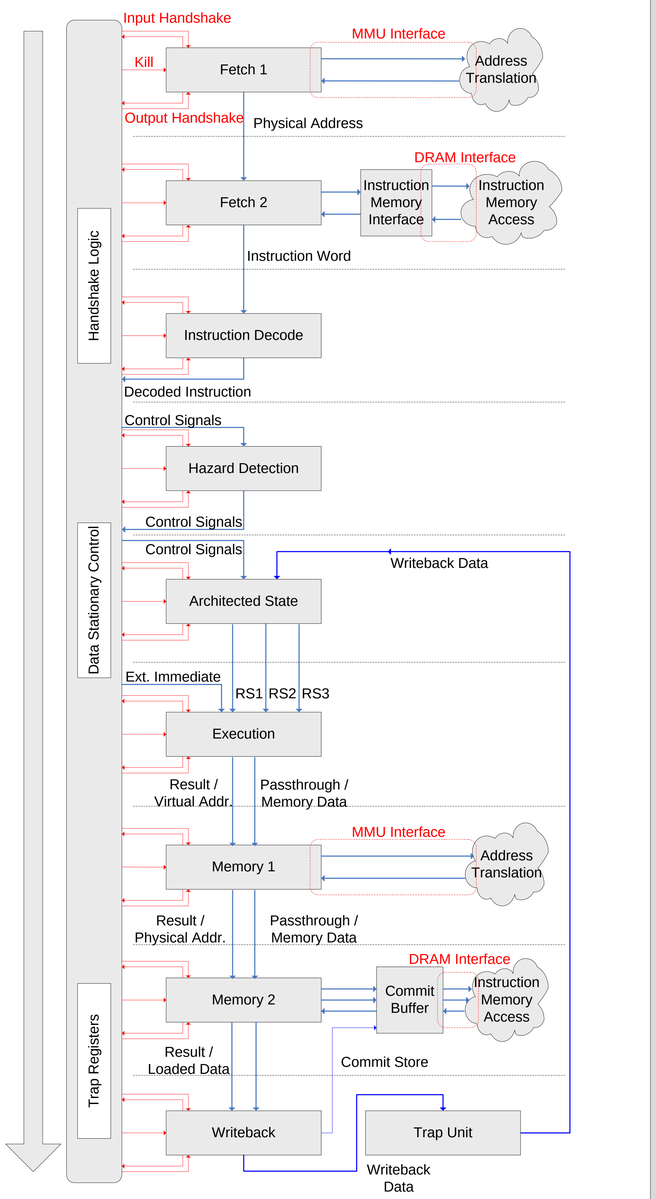
flint
A Flexible, Latency-Insensitive SPARC implementation

Ilia Lebedev (ilial@berkeley.edu)
Alex Williams (awilliams@berkeley.edu)
Chris Fletcher (http://cwfletcher.net)

Special Thanks to the UC Berkeley RAMP Group, Greg Gibeling, John Lazarro

A SPARC V8 implementation optimized for the Virtex5 FPGAs, readily modifiable and extensible through a clear separation of functional blocks and standard interfaces. FLINT provides a framework to quickly implement a CPU model with a range of parameters for quick experimentation.

Architecture



Nine-stage pipeline

Fetch 1

Performs address translation

Fetch 2

Pulls instruction word from memory

Instruction Decode:

Generates controll signals

Hazard Detection:

Stalls on data hazards

Architected State:

Reads status and register file

Execution:

Operates on data

Memory 1

Performs address translation

Memory 2

Perform/queue memory operation

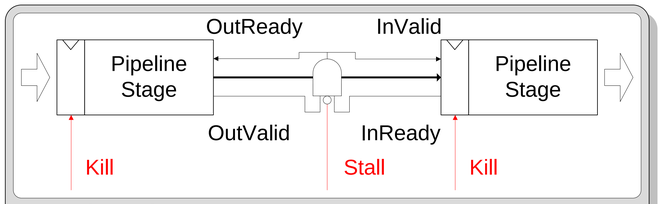
Writeback

Commit buffered memory operation, handle traps

Tolerance for variable latency of each stage.

Support for atomic, complex instructions. (microcode)

Pipeline Stage Interface



Pipeline stages expose a standard interface (Ready/Valid).

Functional blocks such as the ALU, branch predictor, memory interface, instruction decoder, etc. can easily be swapped out.

Functional blocks can be developed in isolation.

Framework to quickly implement a broad range of CPUs.

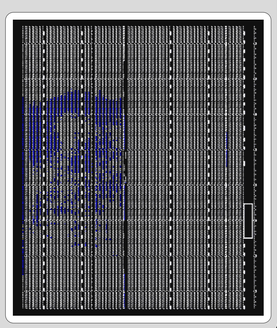
Progress

Most SPARC V8 instructions. (some use microcode)

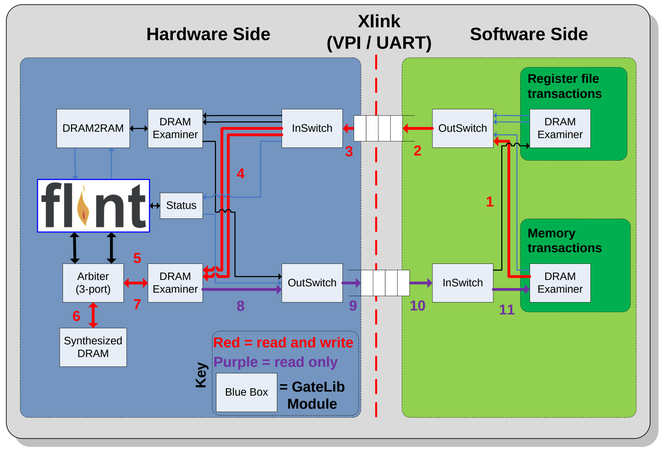
Support for traps.
Compact design with higher CPI.

~100 MHz
~2K slices
1 Block Ram

Goals:
Use DSP blocks.
Optimize for size.



Test Harness



Test Framework