SPARC64™

Presenter: Niteen Patkar

Manager, Processor Development

email: niteen@hal.com

HAL Computer Systems

Outline of the talk

- **SPARC Version 9**
- **Microarchitecture Evaluation**
- **Processor Microarchitecture**
  - ***CPU***
  - ***Cache***
  - ***MMU***
  - ***Summary***

Outline of the talk - contd.

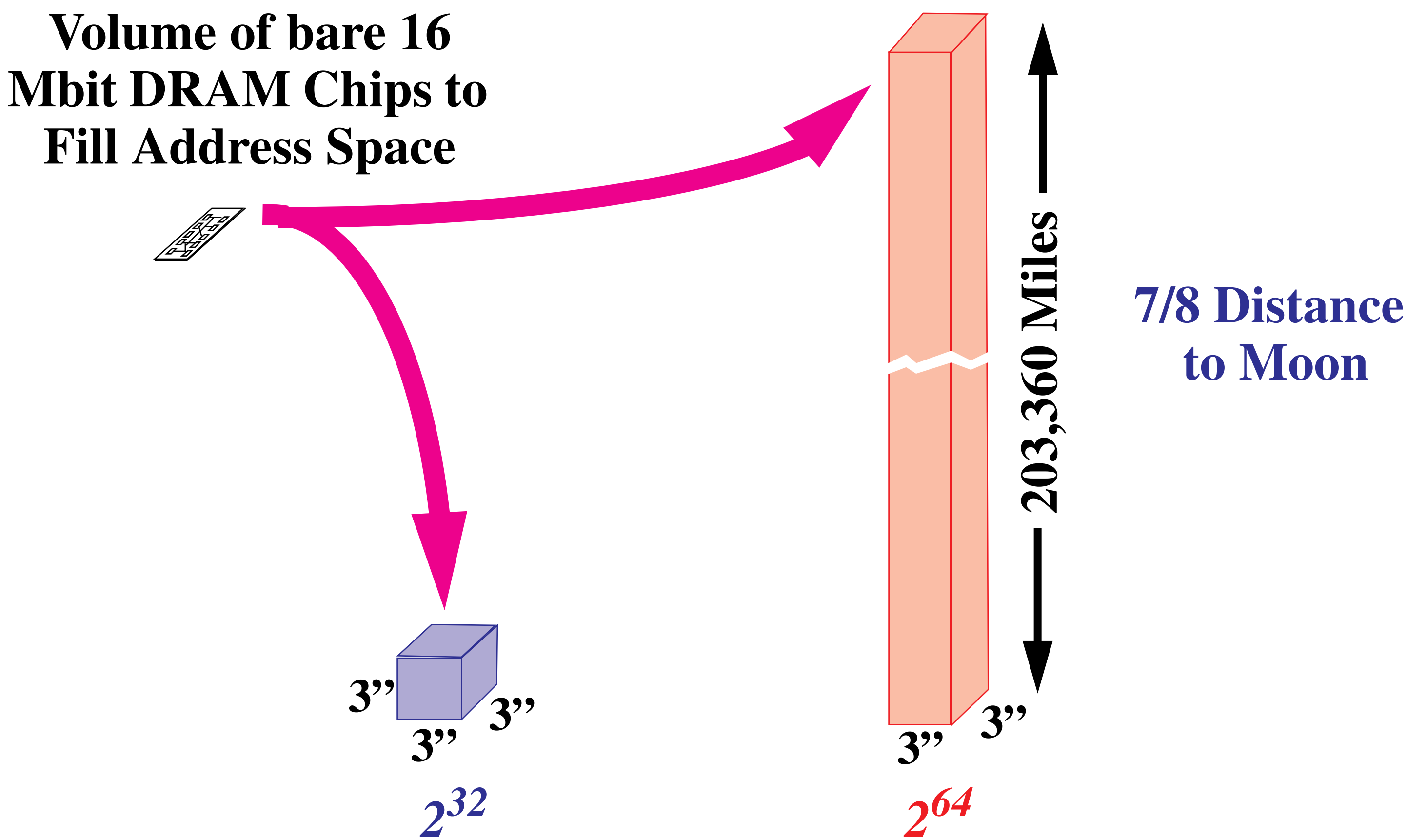
- Implementation trade-offs
- Verification
- Future Developments
- In Summary..

SPARC Version 9

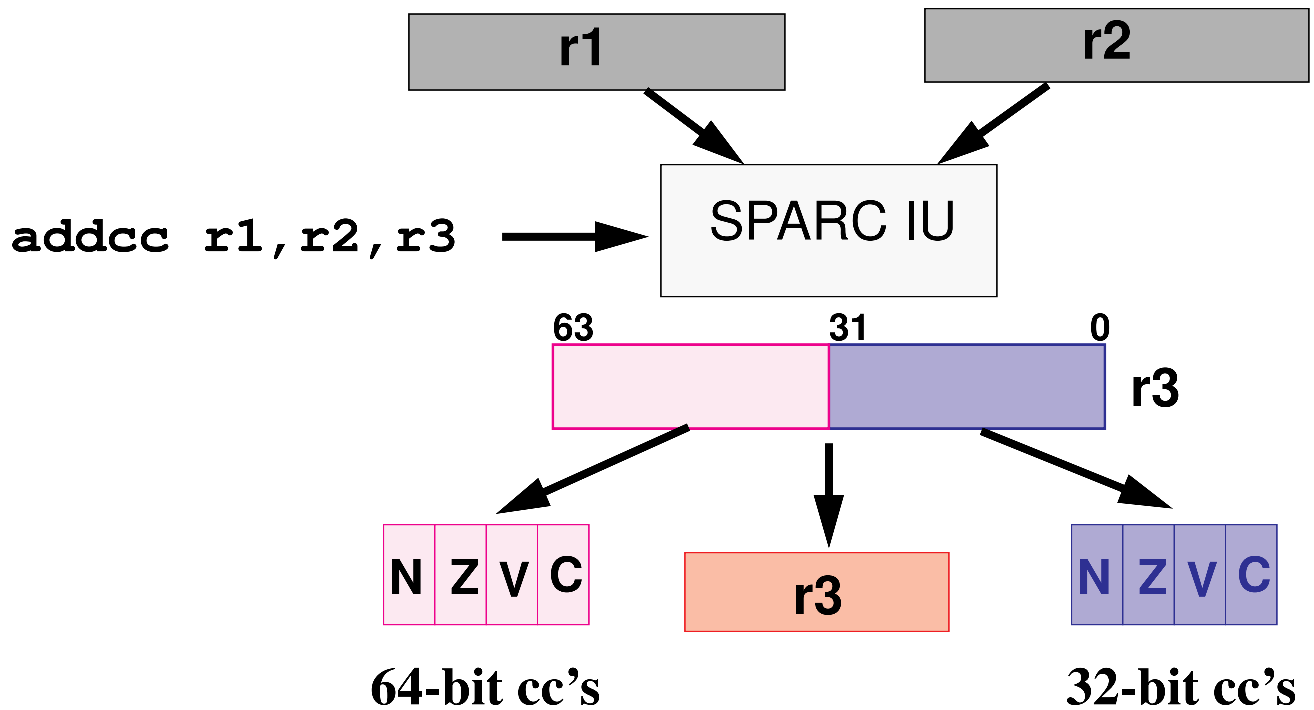
True 64-bit Architecture

- **Full 64-bit Addresses**
  - ***Linear and Unsegmented Address Spaces***
  - ***2^{64} is Big! (18,446,744,073,709,551,616 bytes) 18 Exabytes***
- **64-bit Integer Registers**
- **64-bit Loads, Stores, and Conversions**
- **ALU Operations Extended to 64-bits**
  - ***New Instructions Not Needed in Almost All Cases***

A 64-bit Address Space is Huge!



32 and 64 bit ALU Ops Coexist in SPARC-V9



Upward Compatible From 32-bit

- **All SPARC V7 & V8 User Programs Run Unmodified on SPARC V9**
  - ***No Mode Bit***
- **New SPARC V9 Application Binary Interface (ABI) Was Developed in Parallel With the Architecture**
- **Compatible With SPARC Compliance Definition 2.0 (SCD2.0)**

New Non-Privileged Registers

- **All Integer Registers Extended to 64-bits**
  - ***32 Integer Registers Per Register Window***
  - ***3 to 32 Register Windows***
- **16 Additional Double Floating-Point Registers.**
  - ***Aliased with 8 Quad Regs.***
  - ***Total 32 Single, 32 Double, 16 Quad***
- **64-bit Integer Condition Code Register (%xcc)**
  - ***Set Based on All 64-bits of the Result***

New Non-Privileged Registers - cont.

- **3 More Sets of Floating-Point Condition Codes**
  - *%fcc1, %fcc2, %fcc3*
- **Floating-Point Register State (%FPRS)**
  - *User Reg Enables and Dirty Bits Support Fast Thread Switches*
- **Free Running Counter Register (%tick)**
  - *Counts System Clocks*
  - *Guaranteed to Not Overflow for at Least 10 Years.*
  - *Used by User Programs for Performance Measurements*

New Insts. to Support New Impl.

- **New Implementations = Superscalar, NUMA**
- **64-bit Integer Multiplies and Divides**
- **New Branches with Software Prediction Bits**
  - ***Mispredicted Branches are Very Expensive in New Machines***
  - ***Based On: Integer CC's, FP CC's, or Int. Register Contents***
- **Conditional Register Moves**
  - ***Based On: Integer CC's, FP CC's, or Int. Register Contents***
  - ***Can Conditionally Move Integer or FP Registers***

Conditional Moves Save Branches

- Example: `if (x == 0) y = 5 else y = 10;`

Traditional Branch Code

```
    cmp    0,x
    mov    5,y
    beq    label
    mov    10,y
label:
```

Always One Branch

Conditional Move Code

```
    cmp    0,x
    mov    5,y
    movne  10,y
```

No Branches

New Instructions for Compiler Efficiency

- **Data Prefetches**
  - ***Get Data Nearer to the CPU to Avoid Memory Latency.***
  - ***Does NOT Load Data into a Register***
  - ***Can be Treated as NOP by Hardware***
  - ***Prefetches for 1 or N reads or writes***
- **Non-Faulting Loads**
  - ***Loads Data or Zero Into a Register***
  - ***Loads Zero if Load Causes a Fault***

New Instructions for Compiler Efficiency - cont.

- **Non-Faulting Load Example:**
  - ***Traversing a NULL Terminated Linked List.***
  - ***Loads for "Next" List Member Can Be Started Before the Pointer to "Next" is Validated.***
  - ***Compiler Guarantees Data Will Not be Used if "Next" Pointer is Bad.***
  - ***Used to Move Loads "Earlier" in Compiled Code***
  - ***30% Performance Improvements in Some Codes***

New Privileged Features For Higher Performance

- **Very Fast Register Window Spill/Fill Routines**
  - ***Save Registers on Stack When Hardware Windows Exhausted.***
  - ***19 Instructions Total***
- **Split Register Windows for Multiple Address Spaces**
  - ***Supports Threads, Client/Server Implementations***
- **Fast “User” Traps**

New Privileged Features For Increased Reliability

- **Hardware Trap Stack (≥ 4 Levels)**
  - ***Handles All But Multiple Catastrophic Errors***
  - ***Faster Too!***
- **Reset, Error and Debug Mode (RED Mode)**
  - ***Entered When Trap to Last Trap Level***
  - ***Very Protected Execution Environment for Recovery***
- **SPARC V9 Can be Used in Reliable Commercial Applications.**

Expanded Memory Models

- **New "Relaxed Memory Order" Option (RMO)**
- **RMO Allows For Faster Memory Hardware**
- **"Memory Barrier" Instructions Inserted for Memory Synchronization**
  - ***Only Inserted Where Required***
  - ***99+% of Code Runs Faster on RMO Model***
- **New Atomic "Compare and Swap" Instruction**
  - ***For Wait Free Synchronization***

Microarchitecture Evaluation

Processor Microarchitecture - CPU

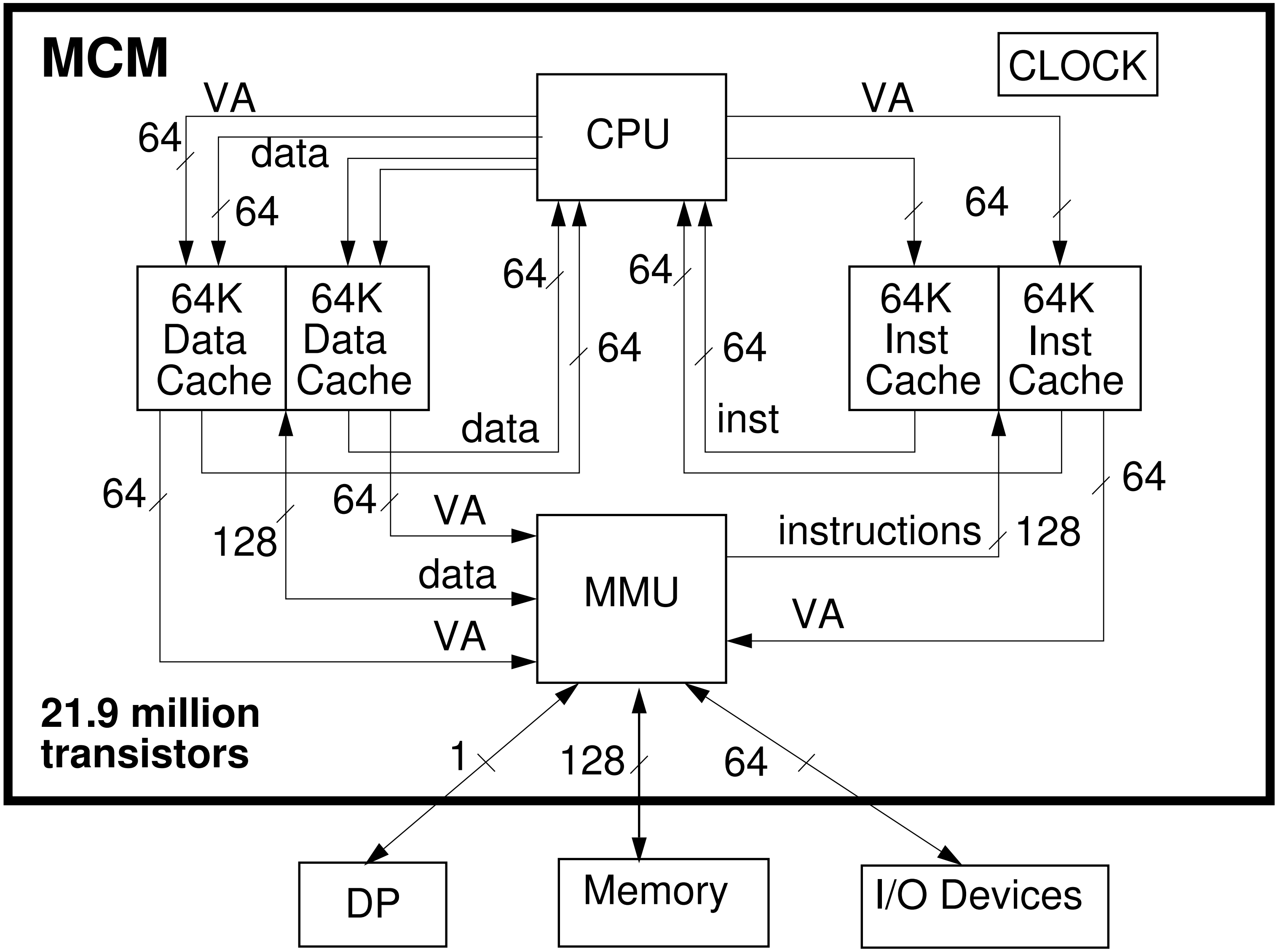
HAL SPARC64 Processor

- First implementation of the 64-bit SPARC-V9 standard - SPARC International certified
- Binary compliant with SPARC V9, V8 and V7 standards
- Full utilization of new V9 extensions

Overview of the SPARC64 Processor

- Superscalar out-of-order speculative execution processor
- Non-blocking Instruction and Data caches
- MMU/Interface chip
- Maintains Precise state
- Extensive features for RAS support
- Current SPARC binaries will run efficiently without recompilation

HAL SPARC64 Processor



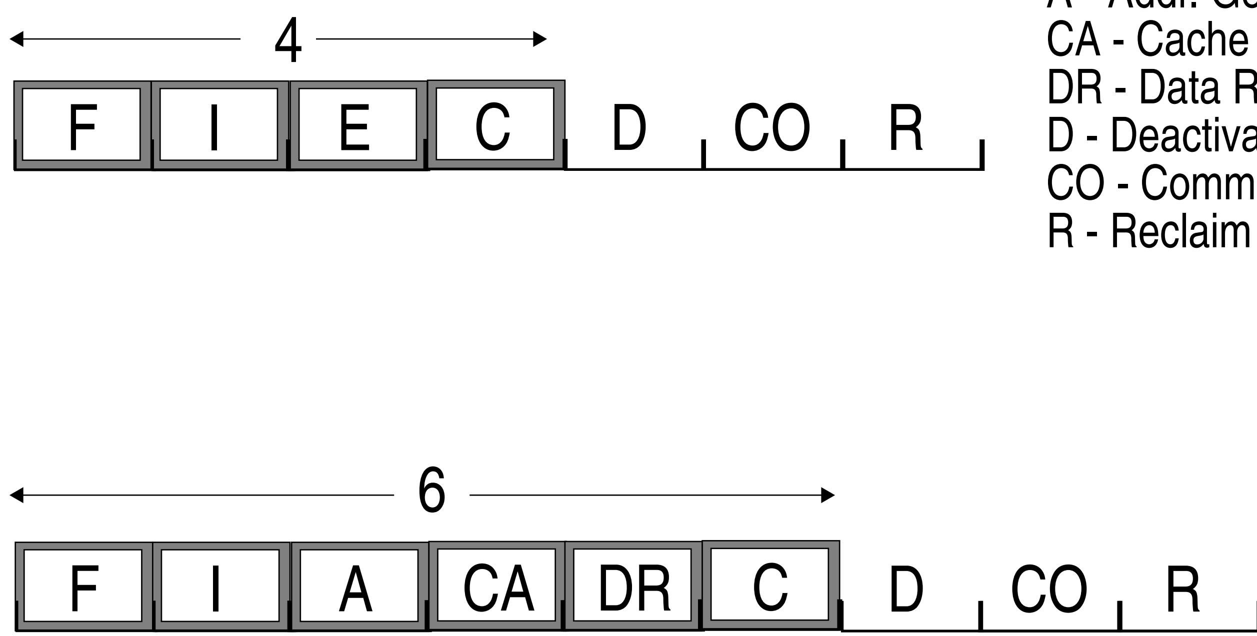
HAL SPARC64 CPU

- Issue upto 4 instructions/cycle- 4 Integer, 2 Load/ Store, 2 Floating Point, 1 Branch
- Data flow architecture
- Large active instruction window - 64
- 4 Reservation stations - multiple execution units
- Single cycle Register Renaming and Read - ISSCC 1995

Pipeline Diagram

Integer Instructions

Load Instructions



F - Fetch
I - Issue
E - Execute
C - Complete
A - Addr. Gen.
CA - Cache acc.
DR - Data Return
D - Deactivate
CO - Commit
R - Reclaim

Data flow helps data dependencies

- **Real Data Dependencies Hamper Multiple Execution:**

  add %r1,%r2,%r3 !r1 + r2 $\rightarrow$ r3

  ld [%r3+%r5],%r7 !real dependency on r3

- **These 2 instructions cannot be "executed" in same cycle.**
- **However they can be "issued" in same cycle.**
  - ***"Data flow" organization lessens effect of real dependencies.***

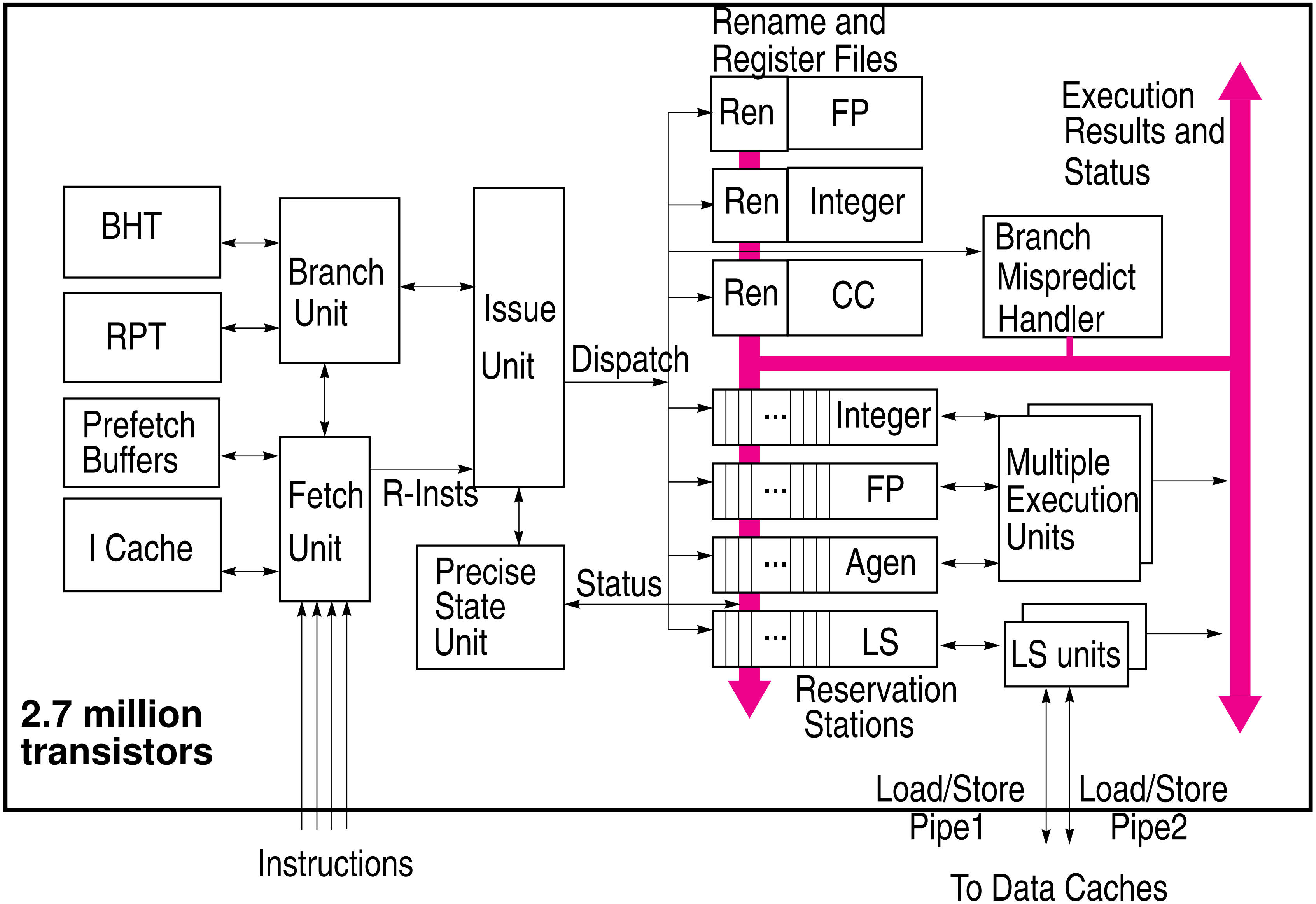
Register Renaming removes artificial dependencies

```
ld      [%r5 + %r6], %r2

add     %r2, %r8, %r30

add     %r9, %r10, %r2
```

= real dependency

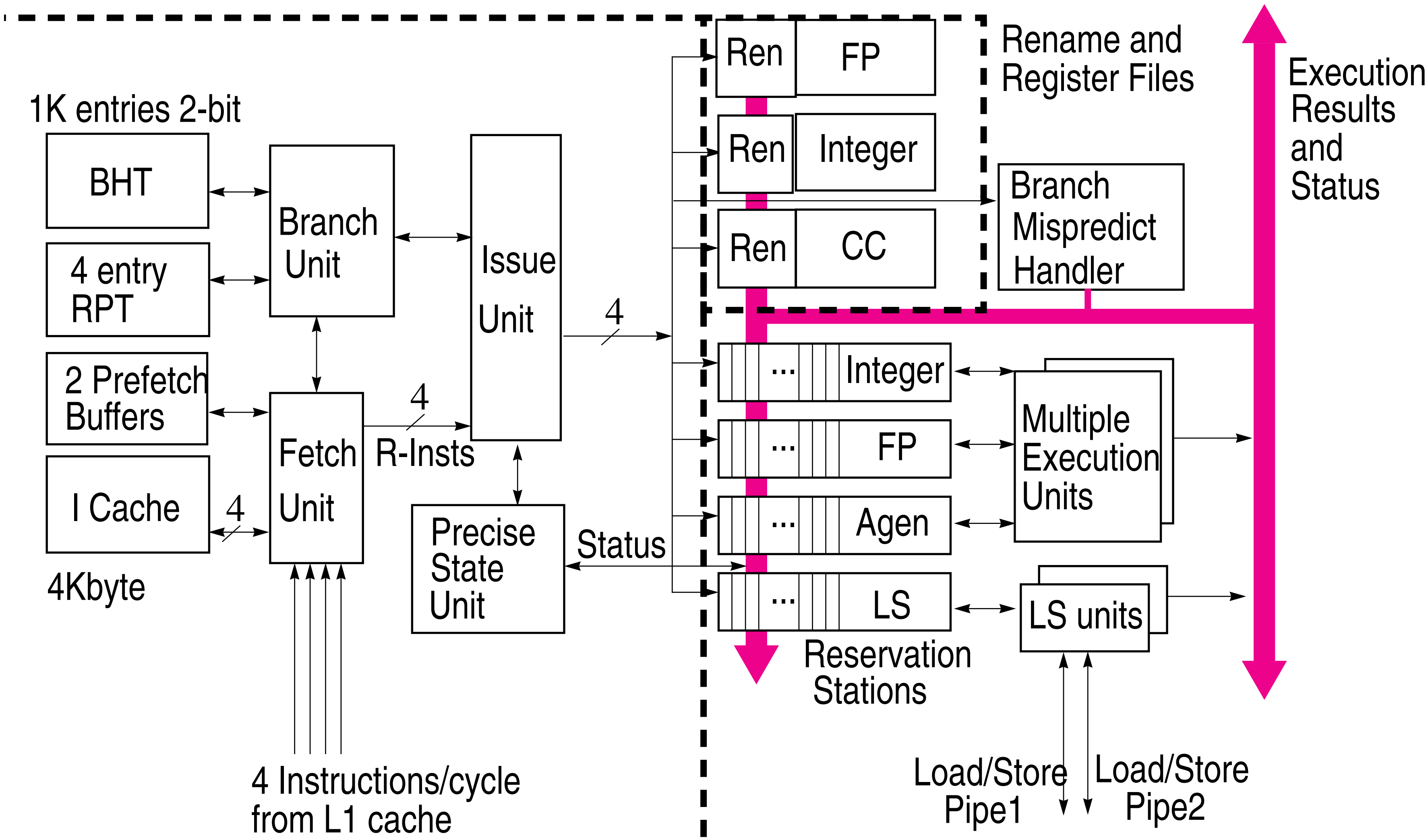
= artificial dependency

SPARC64 CPU

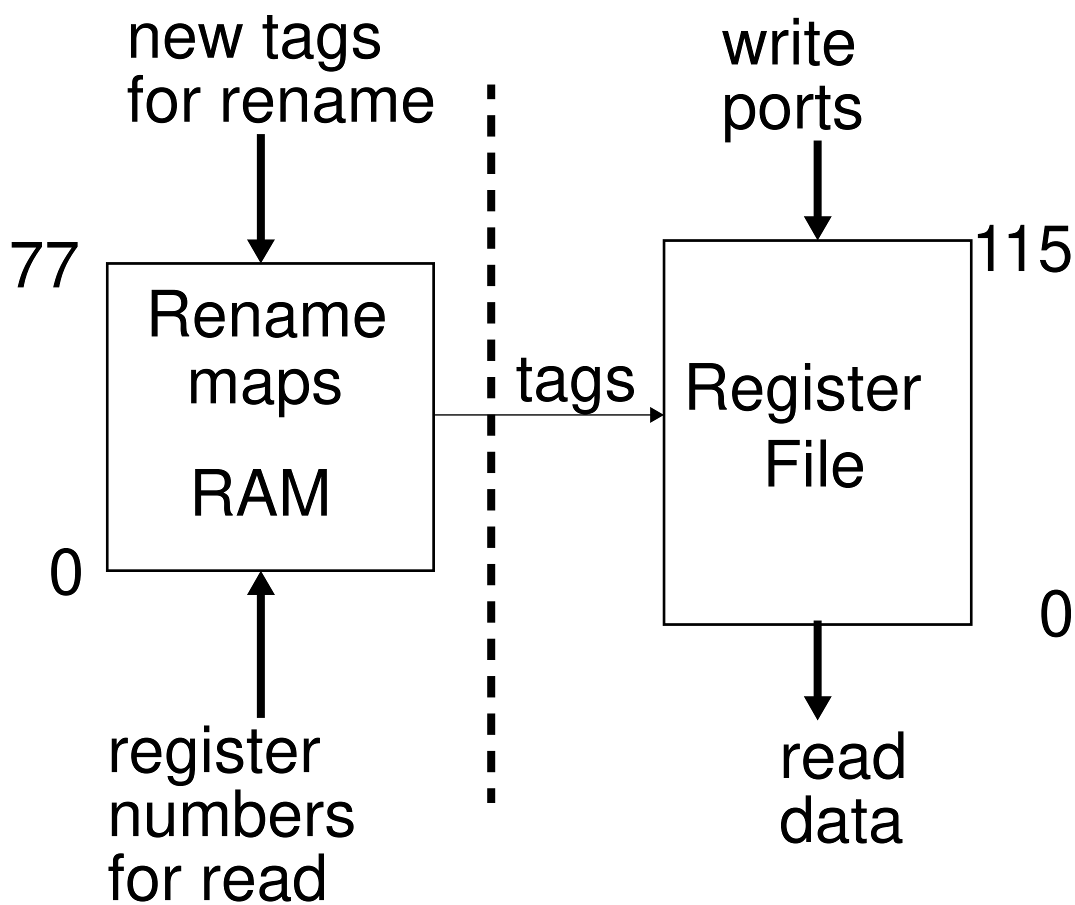


2.7 million transistors

Fetch, Issue and Dispatch Units

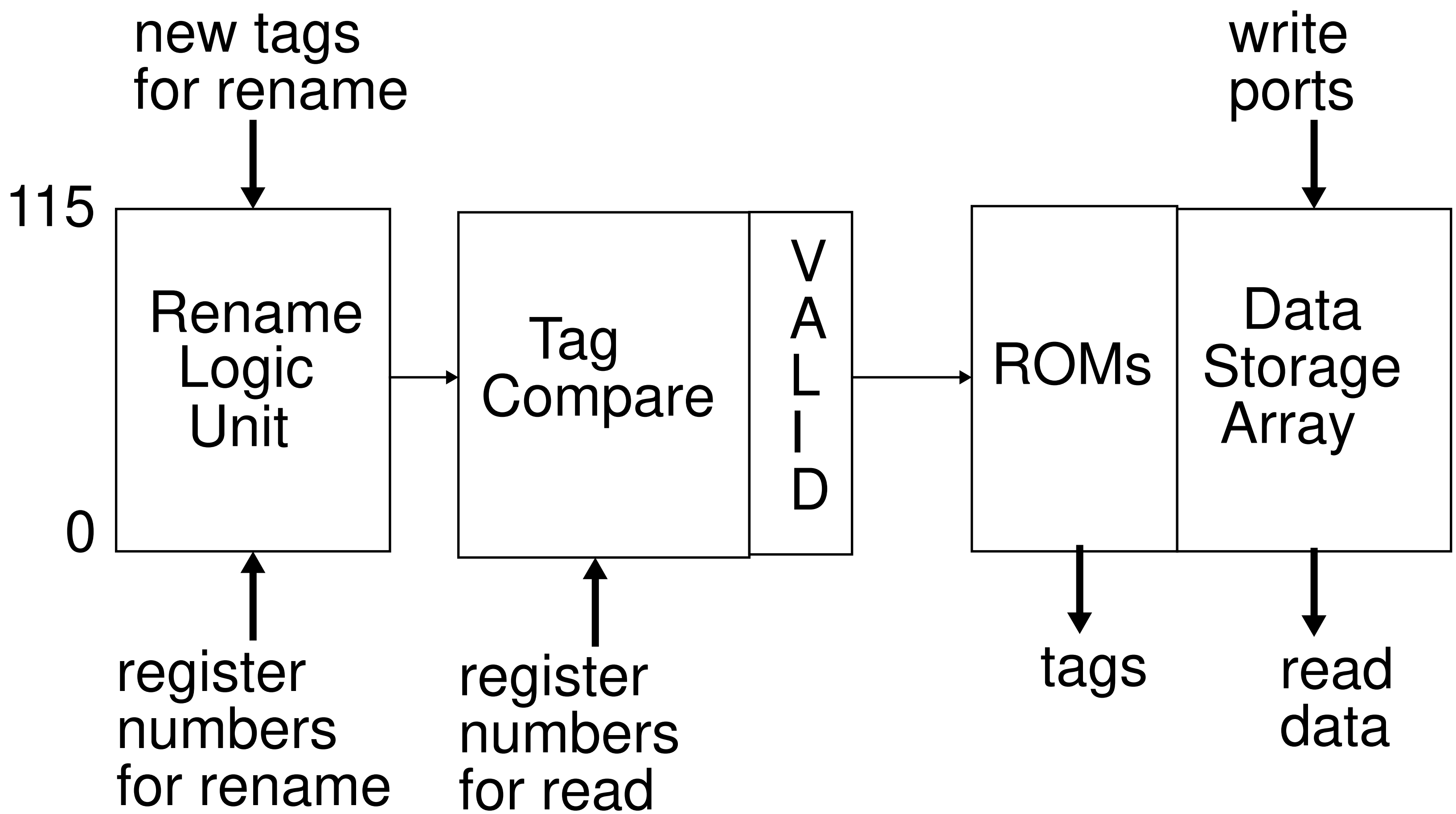


One way to rename



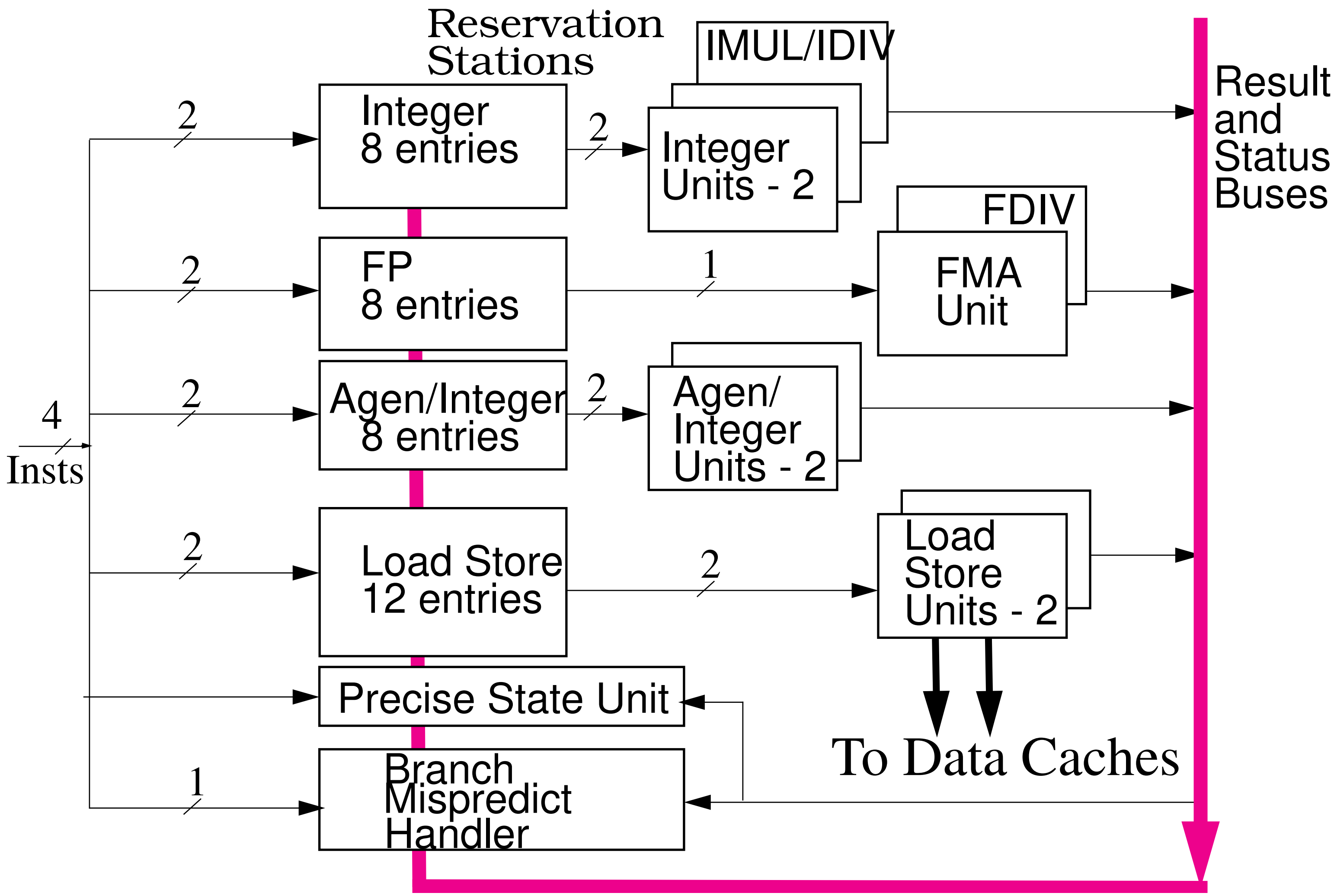
- **550 bits per checkpoint**

Renamed Register File

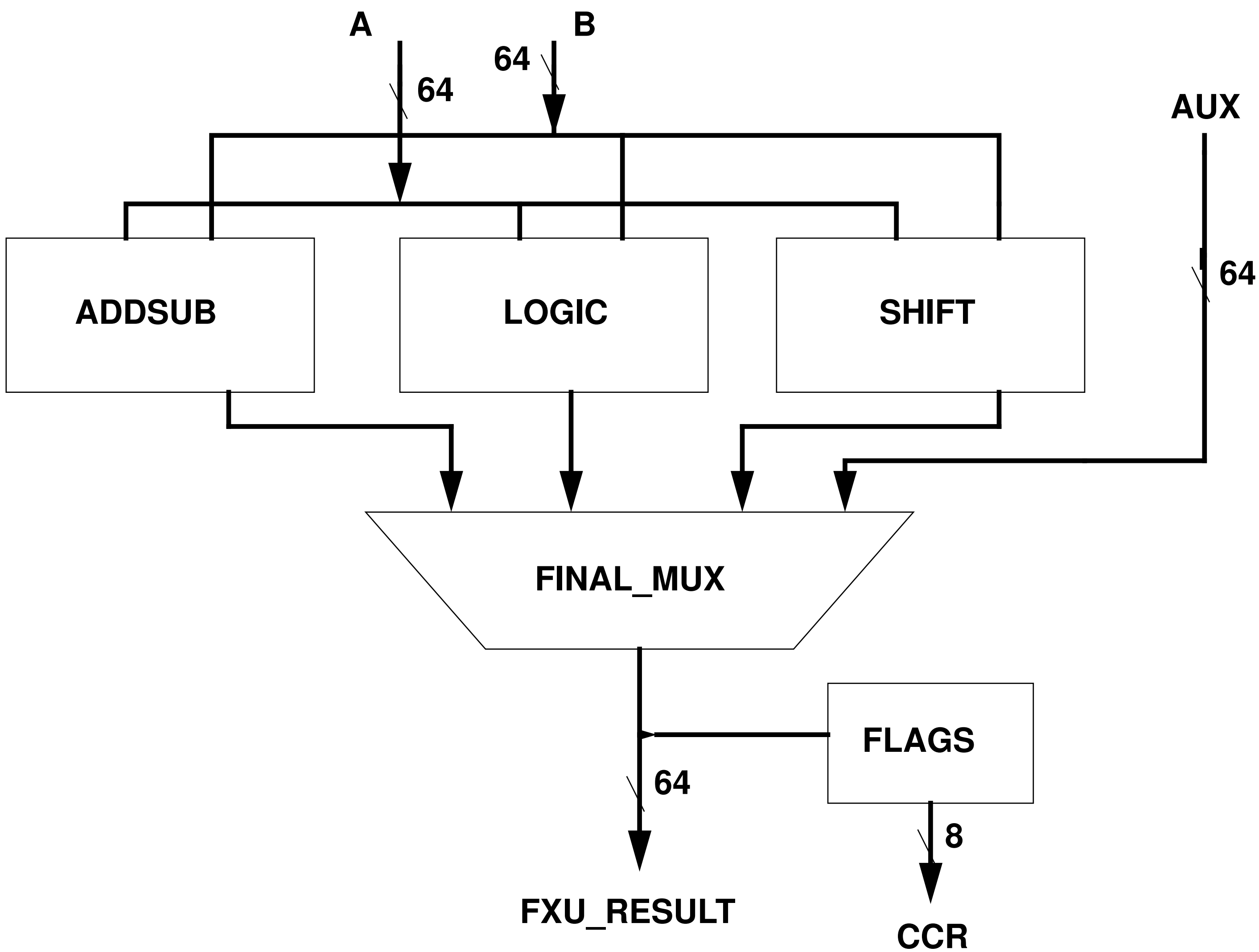


- **116 bits per checkpoint**

Reservation stations and Functional units



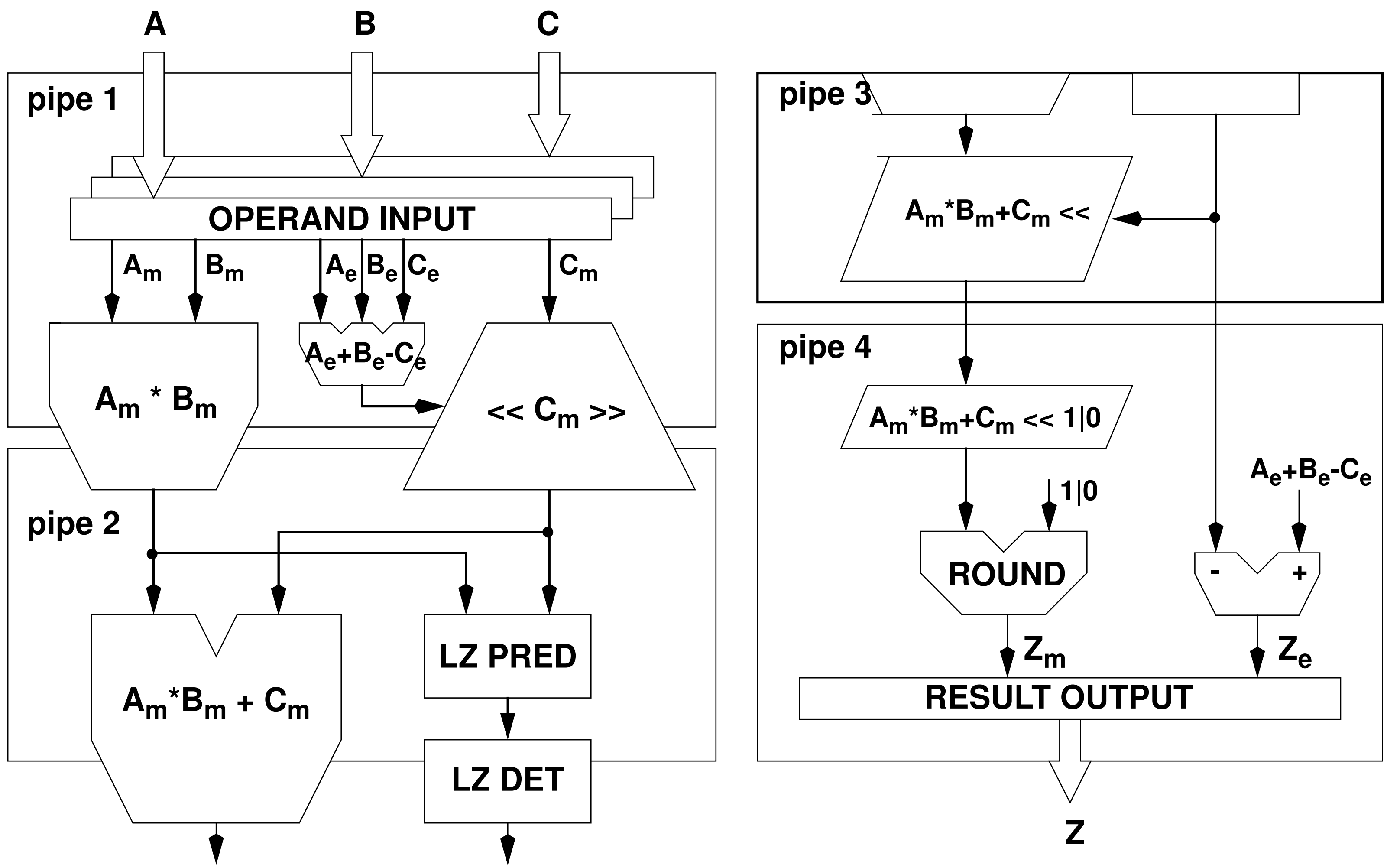
Integer Unit Block Diagram



Floating Point Multiply Add Unit (FMA)

- The FMA is optimized to execute an atomic multiply-add; All other operations are implemented as special cases of the atomic multiply-add
- 4 stage pipeline - Throughput is 1 per cycle.
- The FMA is fully compliant with the IEEE-754 Floating point standard

FMA Block Diagram



Floating Point Divider (FDIV)

- Self-timed divider using Precharged logic
- Uses Radix 2 SRT algorithm
- Can detect early completion of division

FDIV Block Diagram

Table 1: Function Unit Performance

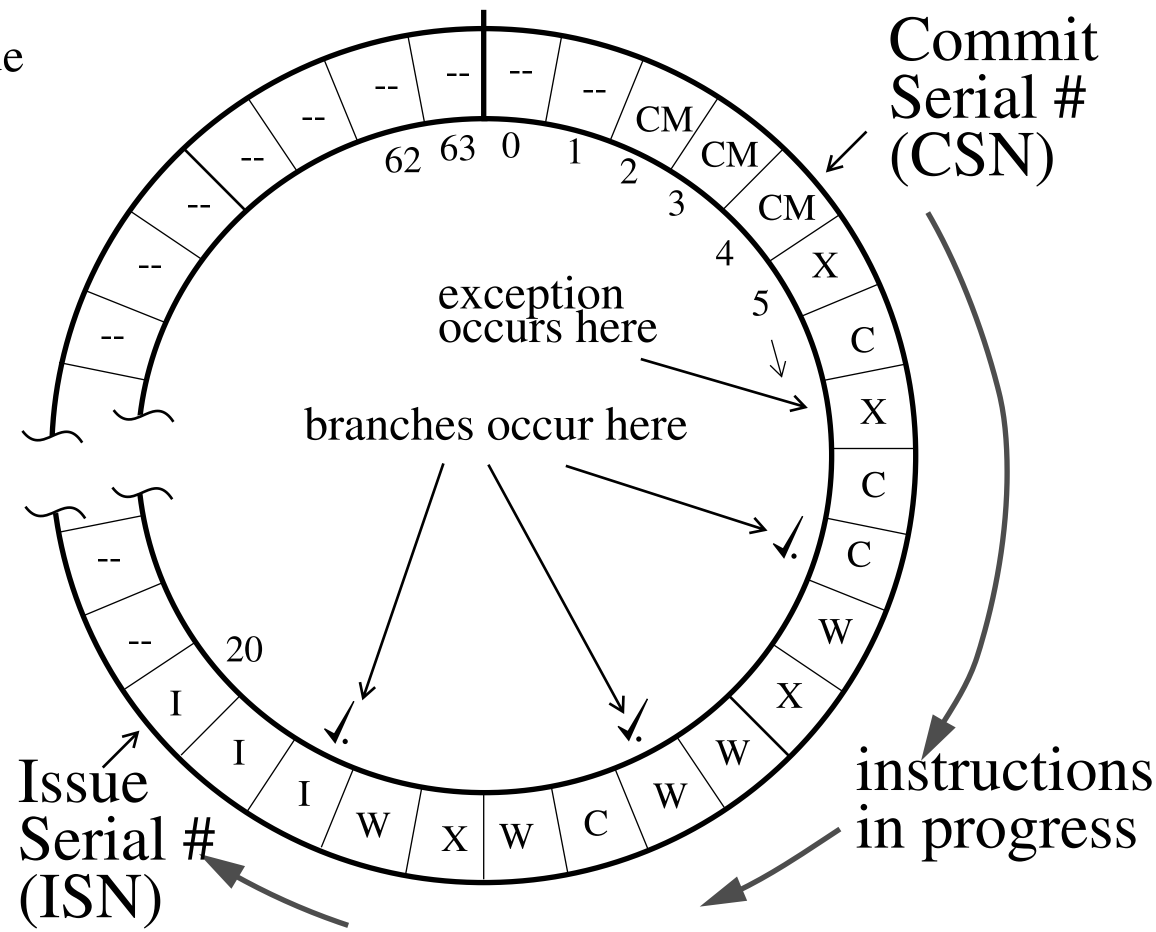
Unit	Throughput (cycles)	Latency (cycles)
Integer	1	1
IMUL	3-5	3-5
IDIV	2-39	2-39
FMA	1	4
FDIV	self-timed	~ 7

Load Store Unit

- Caches are treated as functional units
- Can do two independent 64-bit Loads or Stores per clock
- Three cycle pipeline for cache access
- Loads can be speculative and can occur out-of-order w.r.t other Loads and Stores
- Stores are executed in “Program Order”

Maintenance of Precise State

I = Issued
W = Waiting in Queue
X = Executing
C = Completed
CM= Committed



How many ??

- **Active instruction window - 64**
- **Each cycle, the CPU can**
  - ***Fetch upto 4 instructions***
  - ***Issue upto 4 instructions***
  - ***Speculate upto 16 branches***
  - ***Execute upto 10 instructions***
  - ***Evaluate upto 16 branches***
  - ***Complete upto 9 instructions***
  - ***Commit upto 8 instructions***
  - ***Retire upto 4 instructions***

Processor Microarchitecture - Cache

Cache design requirements

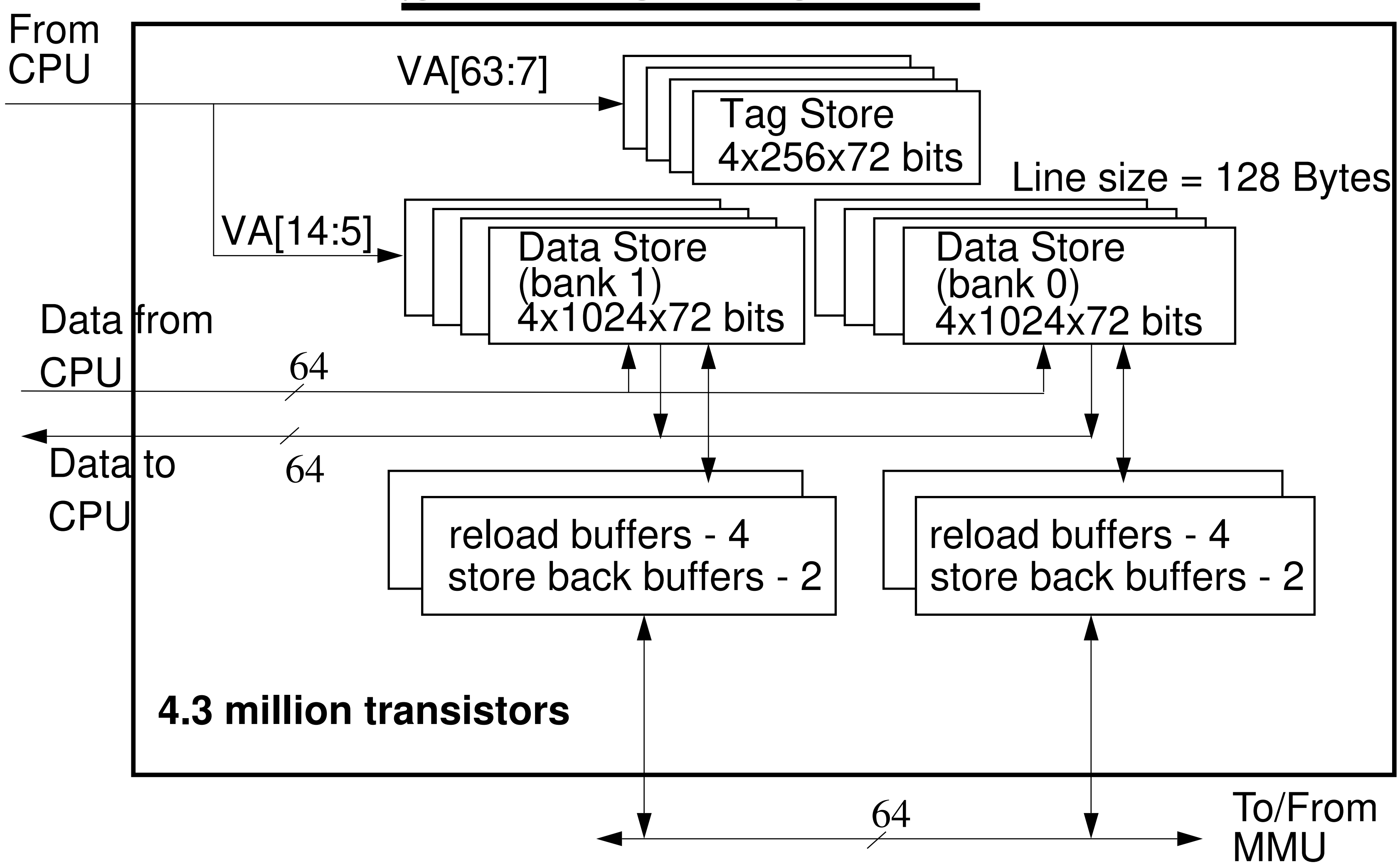
- **Support Out-of-Order and Speculative Execution**
  - *Transaction Based Protocols*
  - *Non-Blocking Design*
- **Provide High Bandwidth**
  - *Pipelined Cache Accesses*
  - *Odd/Even Caches Split Cache Line*
  - *Multiple Banks on Each Chip*
- **Satisfy RAS Requirements**

Cache Subsystem Features

- **Single design for both Instruction and Data Cache**
- **Virtually Indexed and Virtually Tagged with Address Space Tags**
- **64 Kilobyte/chip, 4-way Set Associative,128 Byte Cache Line Size**
  - ***Cache lines are split on 8-byte boundary between odd/even chips***
- **Two memory banks on each chip**
  - ***Allows simultaneous refill and CPU cache access***

- **Four Reload Buffers per chip**
  - ***Each Reload Buffer Corresponds to a Cache Line in Transition***
  - ***Each Reload Buffer Holds up to 8 Cache Misses***
  - ***Cache Hits Are Handled Regardless of Pending Misses***
- **Two Store Back Buffers on each chip**
  - ***Allows lazy flushback of modified lines***
- **RAS features**
  - ***ECC Protection for Data Storage***
  - ***Parity Protection for Signal Pins and for Tag Store***
  - ***Can turn off sets in the case of hard errors***

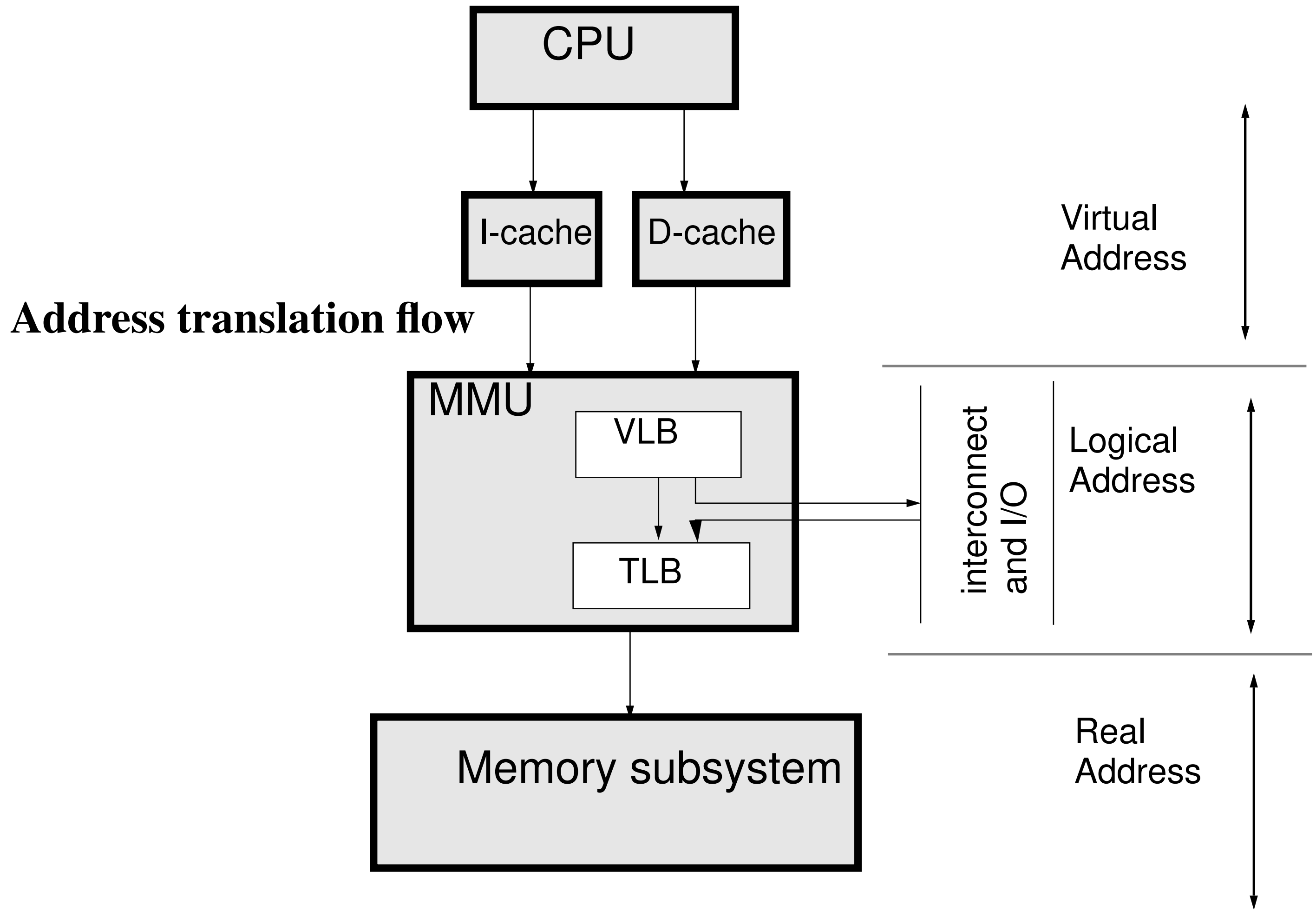
SPARC64 Cache



SPARC64 MMU Functions

- Address translations and hardware table walking
- Protection violation and software faults checking
- Data movement controls and bus interfaces among CPU/Caches, Memory subsystem, and I/O system
- Error Handling for MMU/Cache/Memory subsystem
- Memory coherence and diagnostic functions for MMU/caches/memory

SPARC64 Address Translation flow



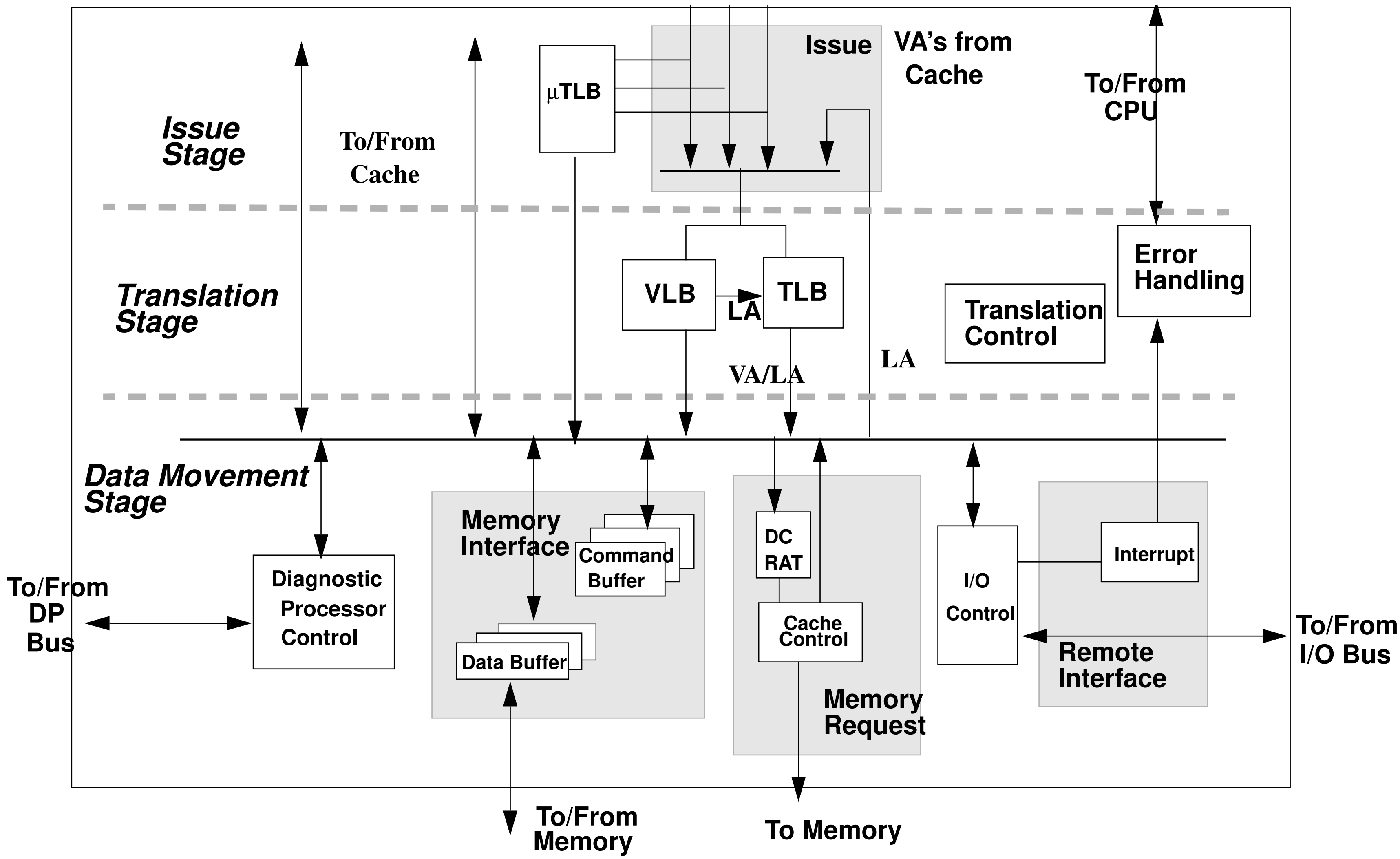
SPARC64 MMU Features

- **Three levels of address space.**
  - ***CPU and caches: Virtual Address space (VA)***
  - ***I/O and Diagnostic Processor: Logical Address space (LA)***
  - ***Memories: Real Address space (RA)***
- **VLB (*View Lookaside Buffer*):**
  - ***Fully-associative table with 128 entries, FIFO replacement policy. VA --> LA mapping***
- **TLB (*Translation Lookaside Buffer*):**
  - ***4-way set-associative table with 256x4 entries, LRU replacement policy. LA --> RA mapping***

SPARC64 MMU Features (contd.)

- **8-entry micro-TLB speeds up translation and is accessed in parallel with cache lookup**
- **DCRAT (*Data Cache Real Address Table*):**
  - ***4-way set-associative table with 256x4 entries that store real address tags of data cache lines.***
  - ***Used to maintain data coherency between data caches, memory subsystem and I/O***

SPARC64 MMU



Issue
VA's from Cache
To/From CPU
Issue Stage
To/From Cache
µTLB
Translation Stage
VLB
TLB
LA
VA/LA
LA
Translation Control
Error Handling
Data Movement Stage
To/From DP Bus
Diagnostic Processor Control
Memory Interface
Command Buffer
Data Buffer
DC RAT
Cache Control
Memory Request
I/O Control
Interrupt
Remote Interface
To/From I/O Bus
To/From Memory
To Memory

RAS Features and Error Handling

- **Three different mechanisms**
  - ***Low priority errors - Interrupts to CPU & logged***
  - ***High priority errors - SPARC V9 RED state***
  - ***Unrecoverable errors - STOP clocks to scan-out machine states***
- **Error handling for Translation Tables**
  - ***Parity check on Translation tables - A faulty entry will be invalidated and miss will be generated***
- **Error handling for Memory Subsystem:**
  - ***SEC/DED with parity-encoded address information***

Processor Microarchitecture - Summary

Table 2: Process Features

Technology	CMOS Twin-well, 3.3V
Process	0.4μm, 4-level metal
M1 pitch	1.8 μm
M2 pitch	2.1 μm
M3 pitch	2.5 μm
M4 pitch	12.0 μm

Table 3: MCM Features

MCM	Ceramic substrate
Power	50W @ 133MHz
MCM pins	Signal - 286, Power - 218
Package	PGA - 565 pins
CPU Die size	17.53mm x 16.92mm
CPU bumps	Signal - 817, Power - 1695
Cache Die size	14.0mm x 10.11mm
Cache bumps	Signal - 446, Power - 1408
MMU Die size	13.83mm x 11.78mm
MMU bumps	Signal - 817, Power - 1695

Reliability, Availability, Serviceability (RAS) Features

- All major arrays protected by Error Correcting Code (ECC) and BIST
- All chip crossing signals are protected by Parity
- Full-scan design - 99.9% fault coverage
- Special Programmable Watchdog registers

Implementation Trade-offs

Trade-offs that will be discussed

- **Free registers used for renaming**
  - *Fixed point Register file*
  - *Floating Point register file*
- **Number of entries in the reservation stations**
  - *Fixed point/Address Generation Q's*
  - *Floating point Q's*
  - *Load/Store Q's*
- **Execution selection mechanisms**

Register files

- **Consumed a lot of die area.**
- **Number of registers is directly proportional to size.**

Fixed Point RF **Floating Point RF**

Fixed and Floating point Reservation Stations

- Size is directly proportional to number of entries.
- Every entry adds to the cycle time

FXU Queue **FPU Queue**

Load/Store Reservation Station

- **Some extra constraints:**
  - ***Had to execute a window spill/fill trap (16 consecutive loads or stores) without stalling issue.***
  - ***Long latency of Cache misses.***

LSU Queue

Execution Selection

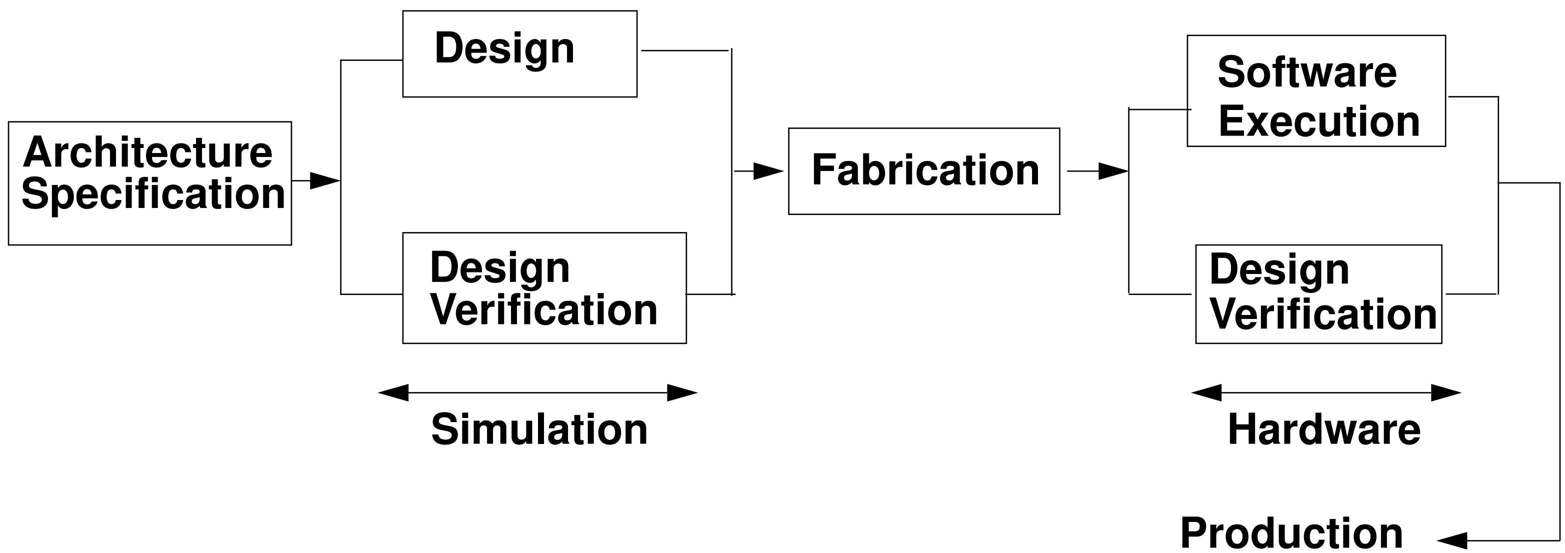
- **Cycle time played the major role.**
- **The ideal execution selection algorithm is:**
  - ***Choose the oldest ready instructions that modified the condition codes.***
  - ***Choose the oldest ready instructions.***
  - ***The first instruction being issued in the current cycle, if ready, could be sent to either execution unit.***
  - ***The second instruction being issued in the current cycle, if ready, could be sent to either execution unit.***

Execution selection - contd.

- **One of algorithms we implemented based on timing and physical constraints:**
  - ***The two oldest ready instructions whose final source dependency was met during the last cycle are selected.***
  - ***Two "random" instructions whose final source dependencies are being generated this cycle are selected.***
  - ***The first instruction being issued in the current cycle is allowed to be executed only in the first execution unit and the second instruction being issued is allowed to be executed only in the second execution unit if nothing is valid in the reservation station***

Verification

Processor design cycle



- Typical Development times are in the range of two to three years.

Critical Components

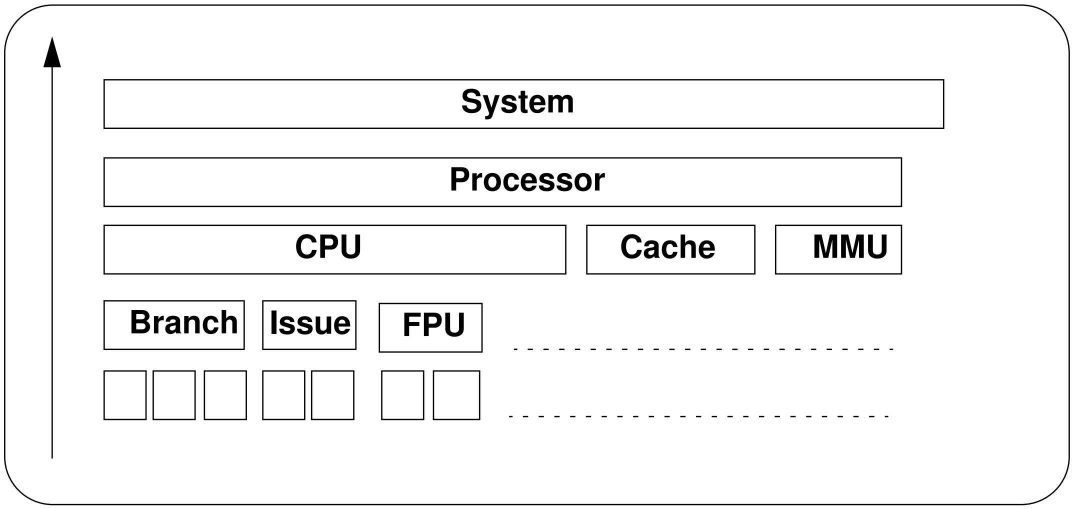
- **Quality of verification programs**
  - *Complete functional coverage.*
  - *Efficient result checking.*
- **Simulation resources**
  - *Simulation speed.*
  - *Optimal utilization.*
- **Quality of design**
  - *Design for verifiability.*
  - *Complexity control.*

Functional Classification of Programs

- **Architecture Verification Programs (AVPs)**
  - ***Validate SPARC-V9 compliance.***
- **Implementation Verification Programs (IVPs)**
  - ***Verify processor micro-architecture.***
  - ***Principle source of identifying design defects.***
- **Random Verification Programs (RVPs)**
  - ***Verify complex and hard to anticipate conditions.***
  - ***Utility in verification on simulation as well as real hardware.***
- **Real Software**

Hierarchical Testing

- **Smaller units verified before integration.**



Directed tests : All levels
Random tests: System level
Real Software : System level

- ***Quick isolation of defects.***
- ***Faster simulation.***
- ***High degree of controllability.***

Directed Tests

- **Unit internal tests**
  - ***Verification of the internal mechanisms of all functional units.***
- **Unit interaction tests**
  - ***Verification of interaction between functional units.***
- **Error injection tests**
  - ***Verification of error handling.***
- **Asynchronous activity tests**
  - ***Verification of asynchronous event handling.***

Random Program Generation

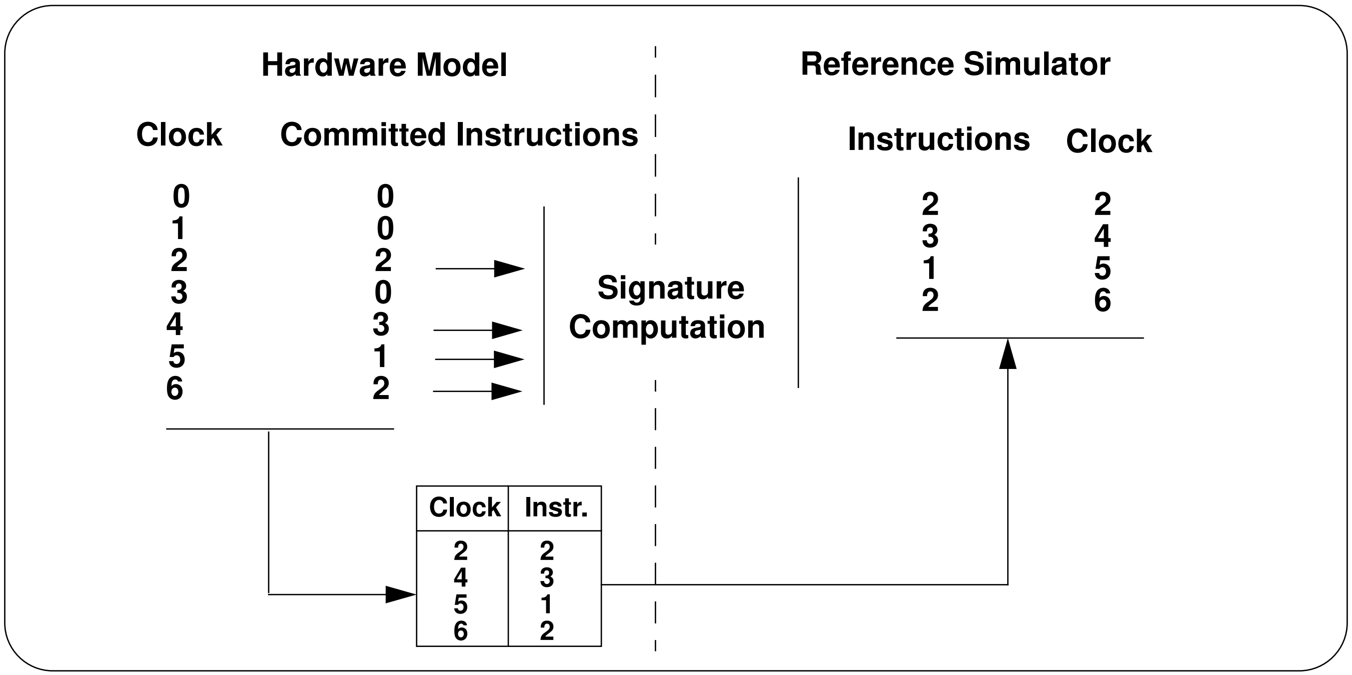
- **Random combinations of biased sequences.**
  - *Complex scenarios hard to envision.*
- **Random instructions with mix of branches.**
  - *High amount of speculative activity.*
- **Random combination of blocks of instructions.**
  - *Conditions similar to real software.*
- **Random combination of valid instructions.**
  - *Stress tests for functional units.*

Result Checking

- **Architectual state checking**
  - *Self checking - Expected results are compared with actual ones.*
- **Reference checking**
  - *Architectural state extracted from logic simulation is checked with that of a reference simulator running the same test.*
- **Internal state checking**
  - *Protocol violation checking - Protocols between functional units are checked for violations.*
- **Assertion checking**
  - *Internal states of the functional units are checked for consistency.*

Signature Compare Mechanism

- **Reference checking mechanism that reduces simulation overhead.**

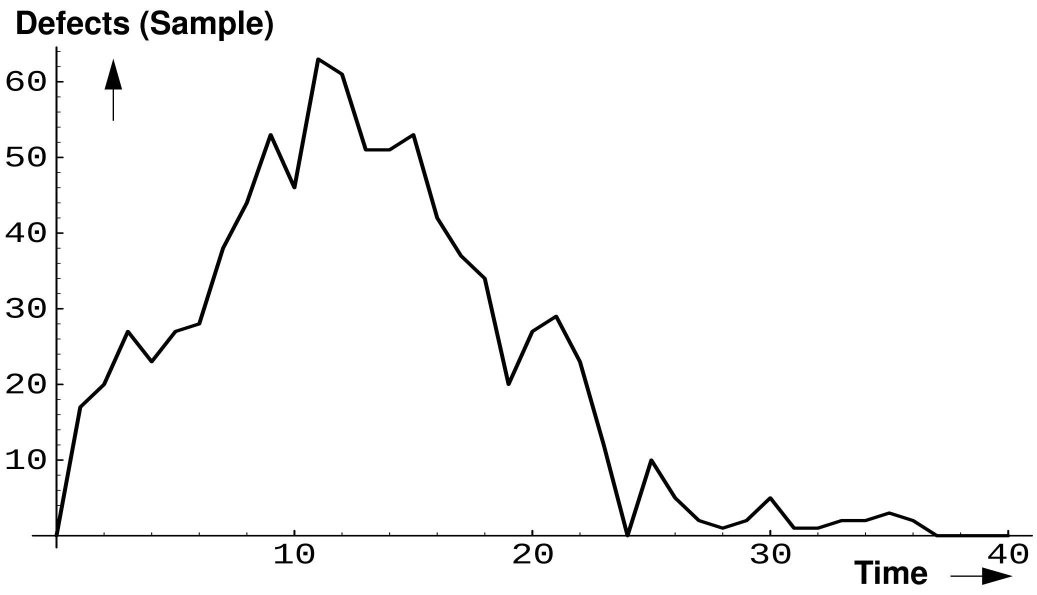


- ***Deferred checking.***
- ***Increases simulation speed.***

Coverage Analysis

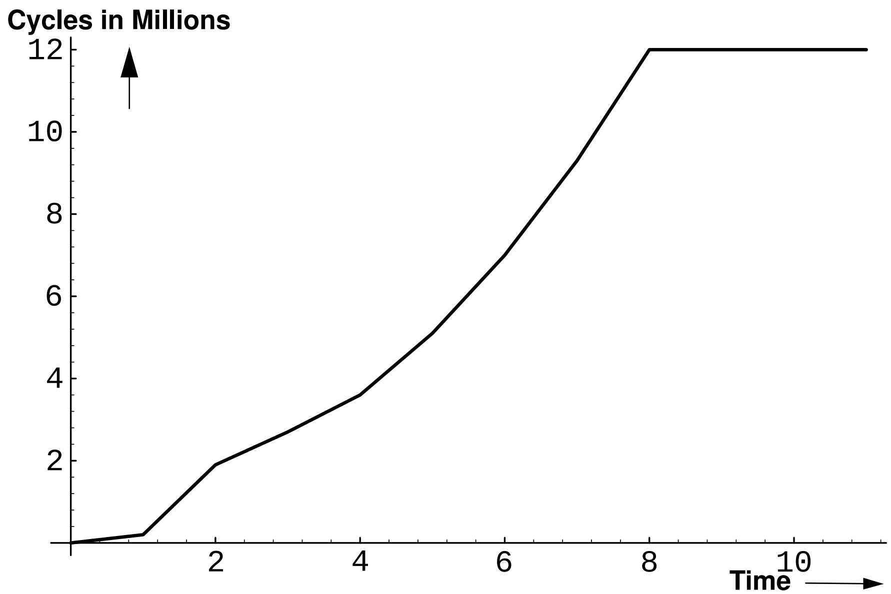
- **Coverage improved using a feedback process.**
- **Test specifications with coverage as the goal.**
  - ***Generation by enumerating various scenarios.***
  - ***Periodic peer review.***
  - ***Analysis of defects detected by non-directed tests.***
- **Validation of coverage.**
  - ***Post-processing of simulation traces.***
  - ***Event logging and analysis.***

Defect Discovery Rate



- **Last few defects take longer time to detect.**
- **Factors affecting discovery rate.**
  - ***Test coverage.***
  - ***Simulation speed.***
  - ***Defect backlog.***

Simulation Cycles



- **Factors affecting simulation cycle requirement.**
  - ***Stability of the design.***
  - ***Simulation cycles per defect.***
  - ***3 Billion cycles equivalent to 1.8 seconds on the hardware.***

Defect Discovery Categorization

- **92% by directed tests.**
  - ***Most effective way of identifying defects.***
  - ***Expensive, in terms of effort and time.***
  - ***High degree of human involvement.***
- **8% by random tests and real-life software.**
  - ***47% due to the ambiguities in the design specifications.***
  - ***27% due to defects in tests.***
  - ***26% due to scenarios not specified during test enumeration.***

Future Development SPARC64+

SPARC64+ Project Goals

- **Improvements over HAL's first generation SPARC64 Processor**
  - ***Improved Performance.***
  - ***Performance Monitor Support.***
  - ***Size Reduction.***
  - ***Increased Debug Visibility.***
  - ***Complete software compatibility with SPARC64.***

SPARC64+ Improvements (Performance)

- **Cycle Time speedup ~30% in CPU**
  - ***Fujitsu's faster CS60-process, smaller die size.***
  - ***Removed 2 levels of logic from SPARC64 CPU long paths.***
- **Improved/Fine-tuned Critical Timing Paths using**
  - ***Faster Fixed/Floating Point - Register Files***
  - ***Faster Caches, Macros(Circuits).***
  - ***Wider Instruction recoding (changed from 38 to 44 bits - requires less decoding in the fetch cycle).***
  - ***More aggressive Execution Selection Algorithm.***

SPARC64+ Improvements (Performance)..

- **Micro Architectural Changes to improve IPC (approx 8.0%) ***
  - ***Fixed Point register file modifications: +2.9%***
  - ***Physical register size increased from 116 to 128 (~10%)***
  - ***Number of Register Windows increased from 4 to 5 (~25%)***
  - ***CPU on-chip Instruction Cache Size - 4KB to 8KB: +3.0%.***
  - ***Branch History Table Size Increase (1K -> 2K) and improved Memory Instruction scheduling:+2.0%.***

****** measured by SPECINT92, All improvement predictions based on "Timer" - HAL developed Performance Evaluation Tool.***

SPARC64+ Improvements (Perf. Monitoring)

- **SPARC64 MMU Performance counters:**
  - ***observing Lookaside Buffer(s) performance.***
  - ***Cache hit/miss rates, Data cache replacement rates.***
- **In SPARC64+, CPU Performance Monitoring features added.**
  - ***HAL implementation specific architecture feature.***
  - ***Software visible and privileged.***

SPARC64+ Improvements (Perf.Monitoring)..

- **Performance Monitors/Counters in CPU:**
  - ***Count Instruction Issue stalls due to Fetch, Resource limitations (Execution units, Reservation Stations, Free Registers) and Precise State Exceptions.***
  - ***Measure Instruction Issue/Commit rates.***
  - ***Measure Total Latency of Memory accesses.***
  - ***Measure Data Cache Hit Rates.***
  - ***Measure Memory Access rates.***
  - ***Count Accumulation/Monitor interval period:100msec.***

SPARC64+ Improvements (Size Reduction)

- **SPARC64+, CPU die size: ~200mm2**
- **33% reduction over SPARC64 CPU due to**
  - ***reduction in metal pitch.***
  - ***reduced transistor feature size.***
  - ***addition of 5th metal layer for routing.***

In Summary...

Summary

- **Covered the following topics:**
  - ***SPARC V9***
  - ***Microarchitecture Evaluation***
  - ***Processor Architecture***
  - ***Implementation Trade-offs***
  - ***Verification***
  - ***Future Development - SPARC64+***

Summary (contd.)

- **4-Issue superscalar out-of-order execution processor. Instructions are**
  - *Issued in order*
  - *Executed out of order*
  - *Completed out of order*
  - *Committed in order*
- **Existing binaries will run efficiently**
- **Extensive RAS features**

Summary (contd.)

- **Two machines introduced**
  - ***HALstation 330 - 100 MHz - 181 Specint92, 230 SpecFP92***
  - ***HALstation 350 - 118 MHz - 212 Specint92, 271 SpecFP92***
- **Future processor developments beyond SPARC64+ are on-going.**

Acknowledgements

A special thanks to Joel Boney, David Chang, Chien Chen, Andrew Essen, Steven Goldstein, Mike Simone, Ravi Swami and Babu Turumella for their help with this presentation.

I would like to acknowledge the contributions of the entire processor design team and also recognize the contributions of Mike Shebanow. Additionally, I would like to thank the Architecture, Design Verification, Technology, and Tools teams, and especially the support of a team of Fujitsu employees working in Japan.

HAL
COMPUTER SYSTEMS
A Fujitsu Company