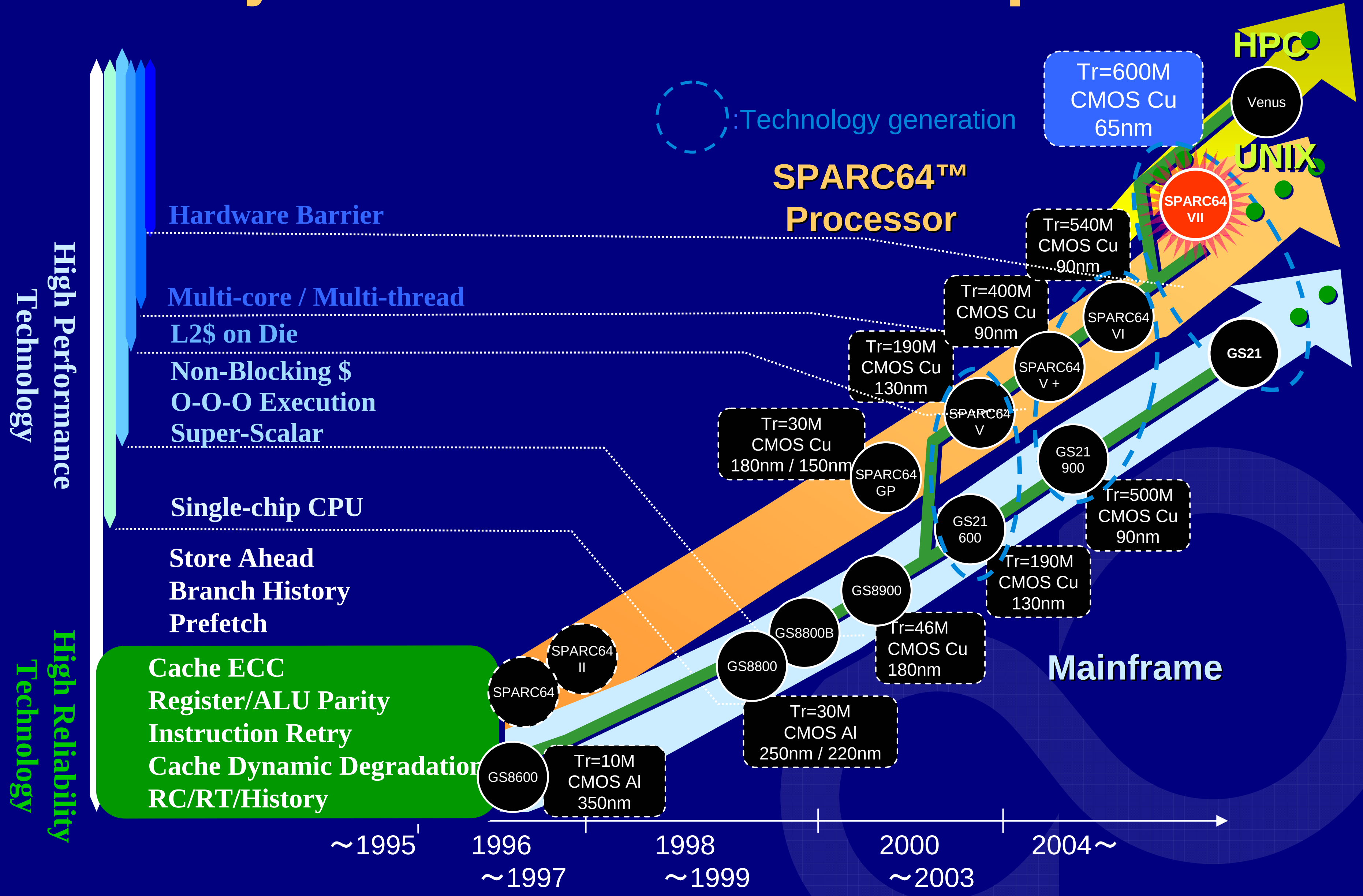
SPARC64™ VII Fujitsu's Next Generation Quad-Core Processor

August 26, 2008

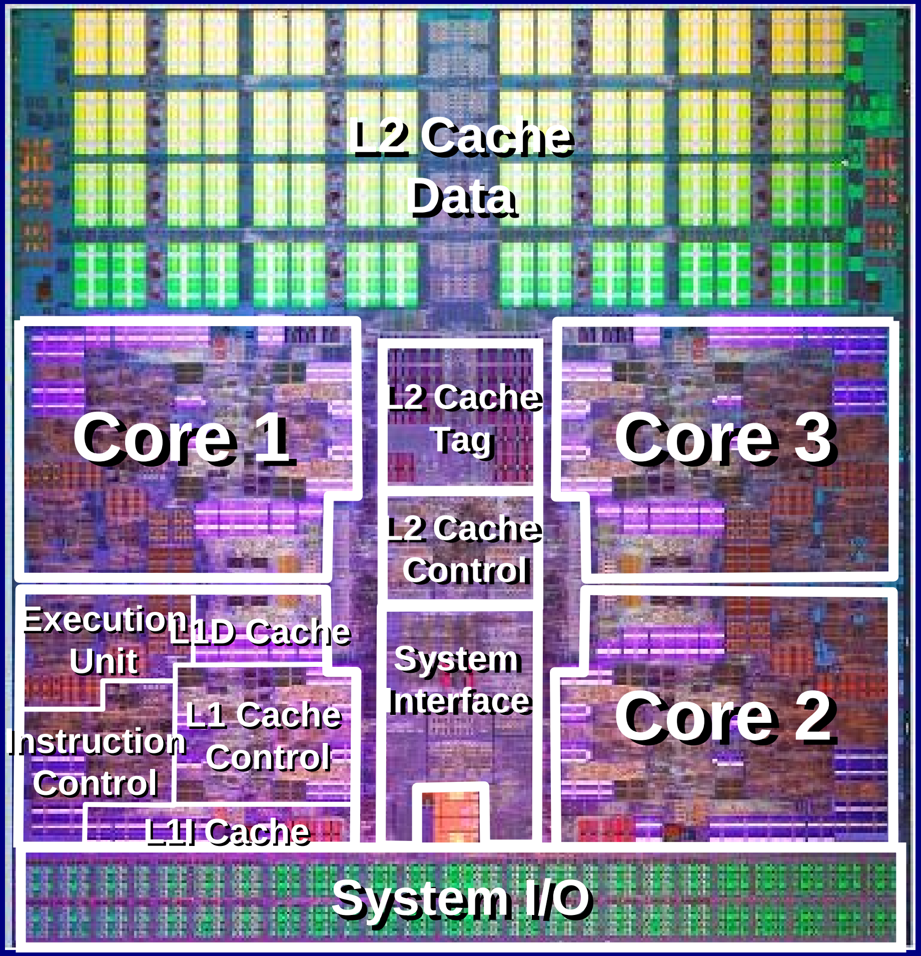
Takumi Maruyama

LSI Development Division
Next Generation Technical Computing Unit
Fujitsu Limited

Fujitsu Processor Development



SPARC64™ VII Chip

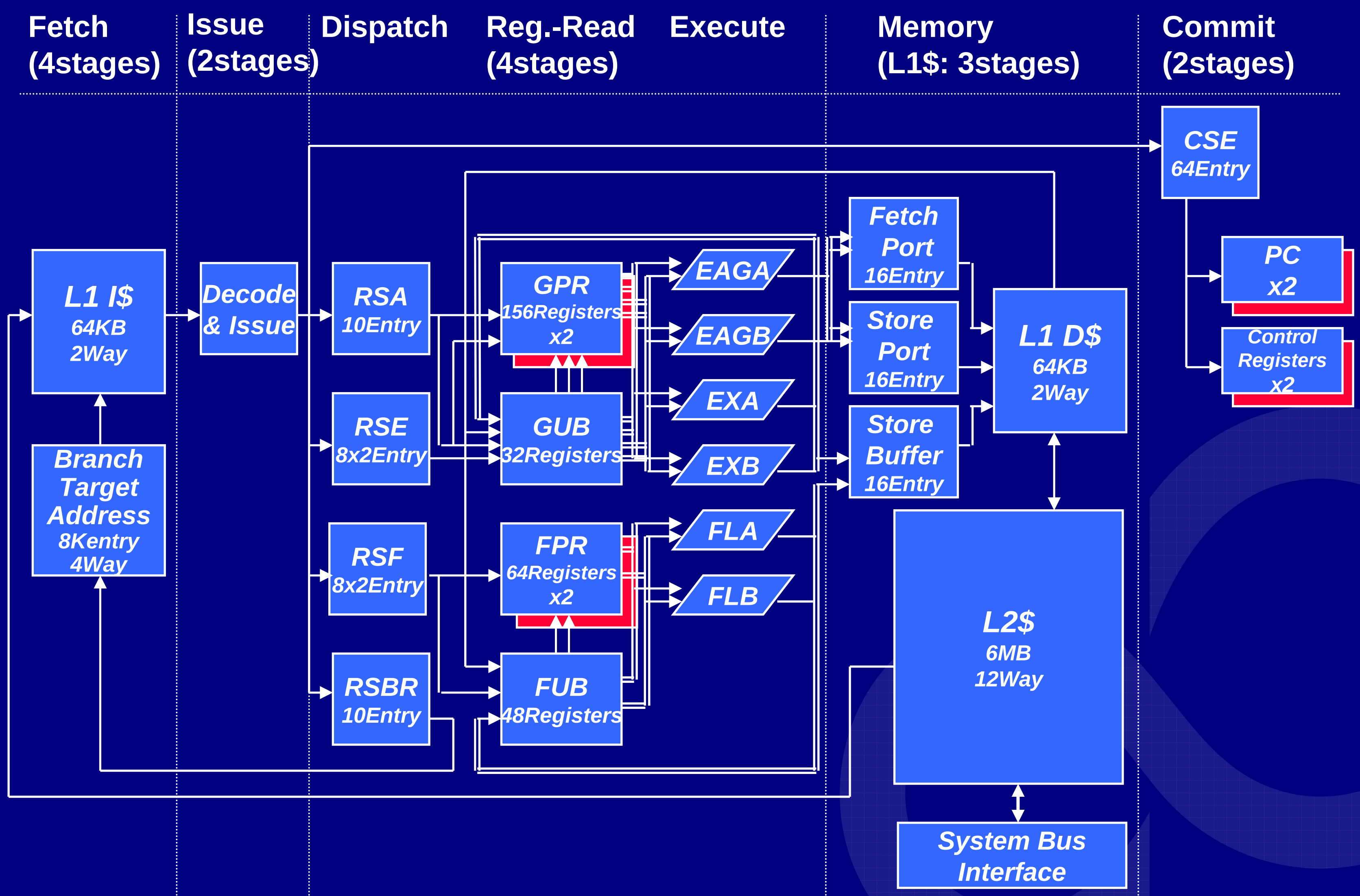


- Architecture Features
  - 4core x 2threads (SMT)
  - Embedded 6MB L2$
  - 2.5GHz
  - Jupiter Bus
- Fujitsu 65nm CMOS
  - 21.31mm x 20.86mm
  - 600M transistors
  - 456 signal pins
- 135 W (max)
  - 44% power reduction per core from SPARC64^{TM} VI

SPARC64™ VII Design Target

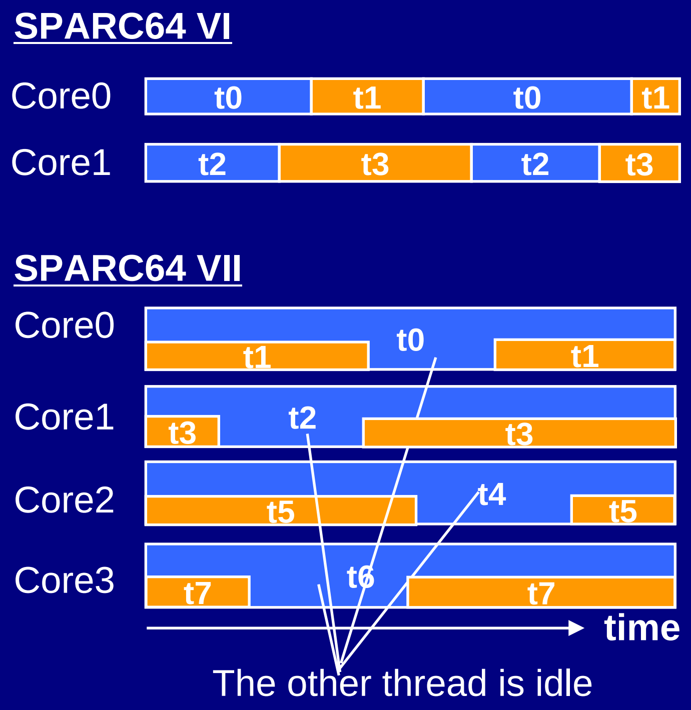
- Upgradeable from current SPARC64™ VI on SPARC Enterprise Servers
  - ➔ Keep single thread performance & high reliability by reusing SPARC64™ VI core as much as possible
  - ➔ Same system I/F: Jupiter-Bus
- More Throughput Performance
  - ➔ Dual core to Quad-core
  - ➔ VMT (Vertical Multithreading) to SMT (Simultaneous Multithreading)
- Technical Computing
  - ➔ Shared L2$ & Hardware Barrier
  - ➔ Increased Bus frequency (on Fujitsu 'FX1' server)

Pipeline Overview



SPARC64™ VII

Simultaneous Multi-Threading



- Fine Grain Multi-thread
  - Fetch/Issue/Commit stage: Alternatively select one of the two thread each cycle
  - Execute stage: Select instructions based on oldest-ready independently from threads.

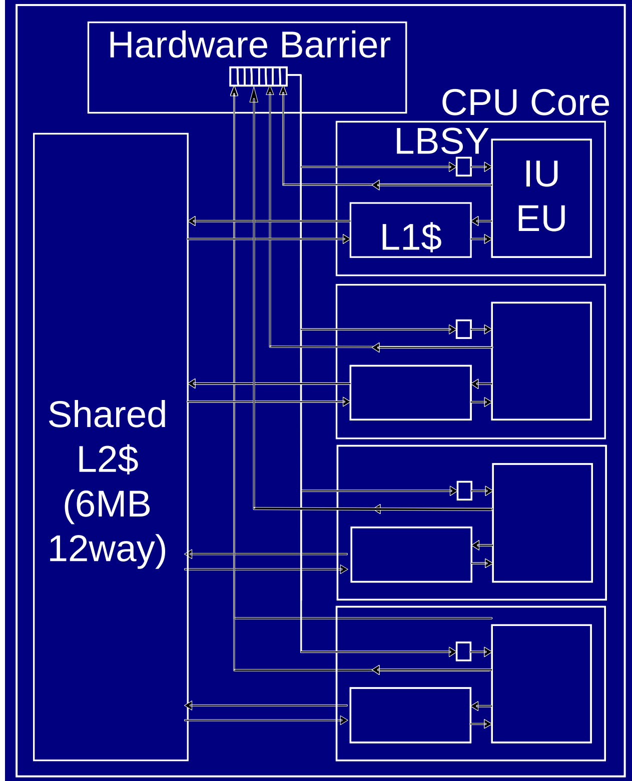
➔ SMT throughput increase
  - x 1.2 INT/FP
  - x 1.3 Java
  - x 1.3 OLTP

- Split HW Queues

#Active threads	IB	Reservation Station.		Rename Registers		Port		CSE
		RSE	RSF	GUB	FUB	FP	SP	
Two	4+4	8x2	8x2	24+24	24+24	8+8	8+8	32+32
One	8	8x2	8x2	48	48	16	8	64

- To avoid interaction between threads.
- Automatically combined if the other thread is idle.

Integrated Multi-core Parallel Architecture



- L2 cache
  - Shared by 4 Cores to avoid false-sharing
  - B/W between L2 cache and L1cache (2x of SPARC64™ VI)
    - L2$→L1$: 32byte/cycle/2core x 4core
    - L2$←L1$: 16byte/cycle/core x 4core
- Hardware Barrier
  - High speed synchronization mechanism between cores in a CPU chip.
  - Reduce overhead for parallel execution

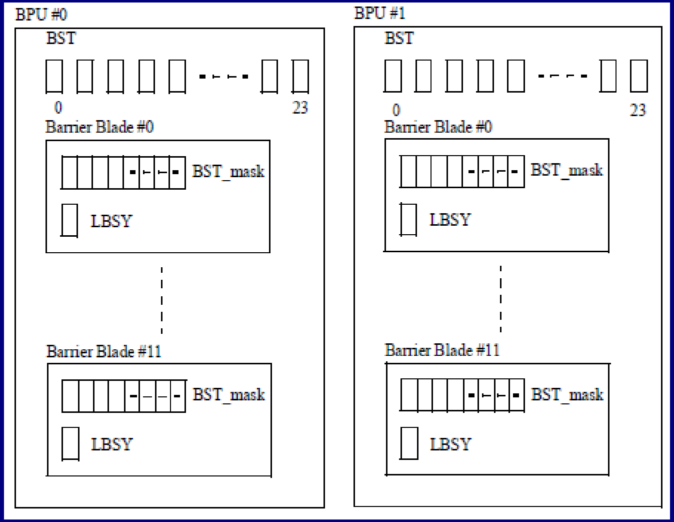
➔ Handles the multi-core CPU as one equivalent fast CPU with Compiler Technology (Fujitsu's Parallelnavi Language Package 3.0)

  - Automatic parallelization
  - Optimize the innermost loop

```
  DO J=1,N
P  DO I=1,M
P    A(I,J)=A(I,J+1)*B(I,J)
P  END
  END
```

SPARC64™ VII

Hardware Barrier



- Barrier resources
  - BST: Barrier STatus
  - BST_mask: Select Bits in BST
  - LBSY: Last Barrier synchronization status
  - Synchronization is established when all BST bits selected by BST_mask are set to the same value.

Sample Code of Barrier Synchronization

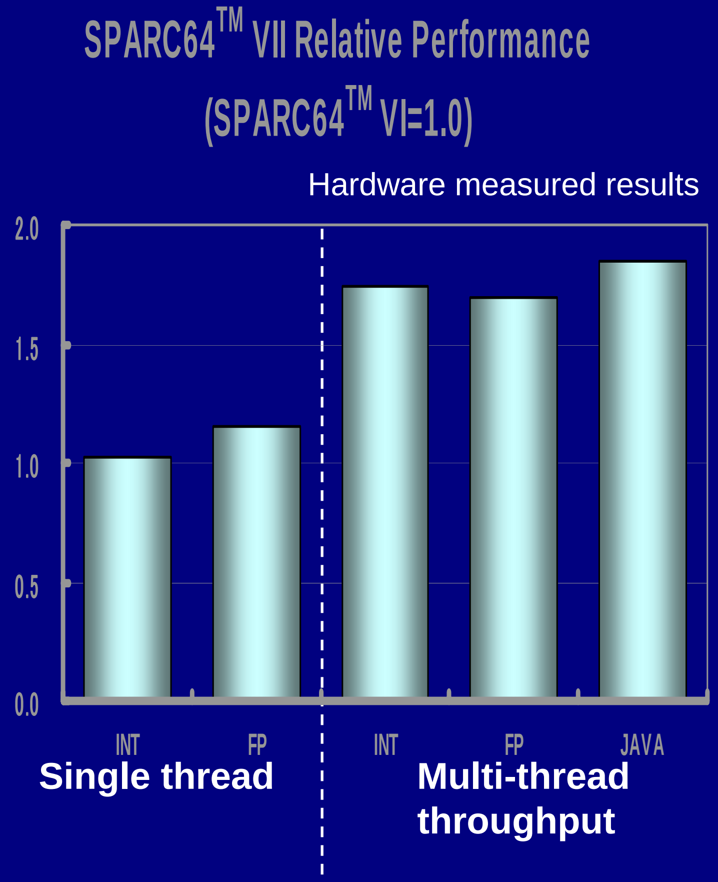
```
/*
* %r1: VA of a window ASI, %r2:, %r3: work
*/
ldxa [%r1]ASI_LBSY, %r2          ! read current LBSY
not %r2                          ! inverse LBSY
and %r2, 1, %r2                  ! mask out reserved bits
stxa %r2, [%r1]ASI_BST           ! update BST
membar #storeload ! to make sure stxa is complete
loop:
ldxa [%r1]ASI_LBSY, %r3          ! read LBSY
and %r3, 1, %r3                  ! mask out reserved bits
subcc %r3, %r2, %g0              ! check if status changed
bne,a loop
sleep                ! if not changed, sleep for a while
```

- Usage
  - Each core updates BST
  - Wait until LBSY gets flipped

➔ Synchronization time: 60ns
  - 10 times faster than SW

SPARC64™ VII

Other performance enhancements



- Faster FMA (Fused Multiply-Add)
  - 7cycles→6cycles
  - DGEMM efficiency: 93% on 4 cores
  - ➔ LINPACK=2,023Gflops with 64CPU
- Prefetch Improvement
  - HW prefetch algorithm further refined
  - SW is now able to specify “strong” prefetch to avoid prefetch lost.
- Shared context
  - Virtual address space shared by two or more processes
  - Effective to save TLB entries
- New Instruction
  - Integer Multiply-Add with FP registers
- etc…

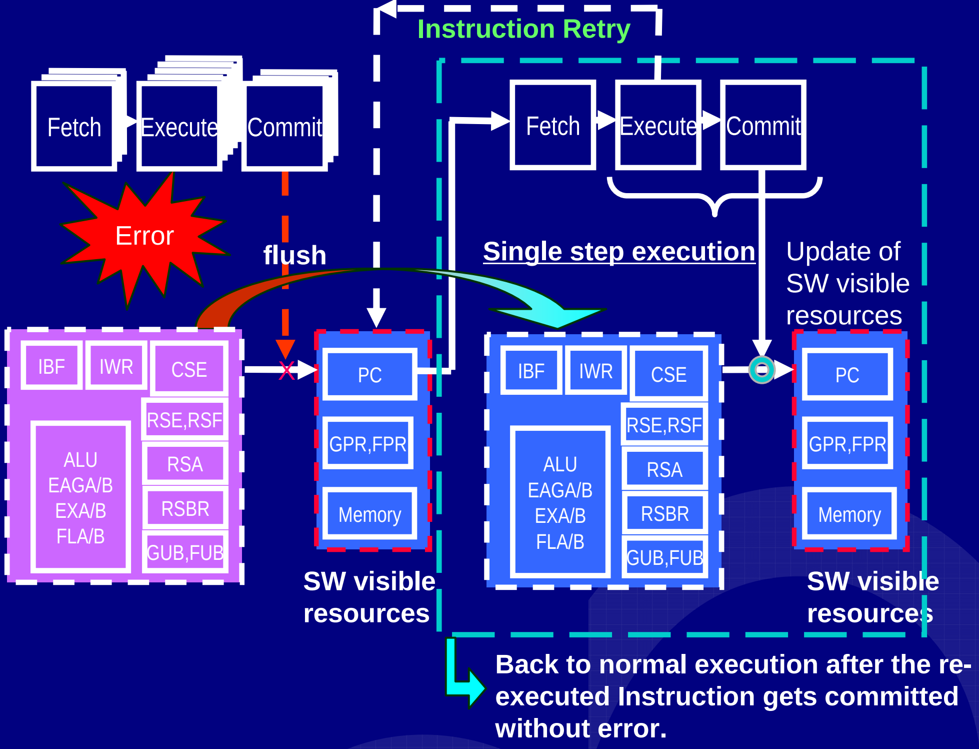
➔ Performance relative to SPARC64^{TM} VI
  - Single thread performance : x1.0 - x1.2
  - Throughput: x1.7 - x1.9

SPARC64^{TM} VII

Reliability, Availability, Serviceability

- Existing RAS features

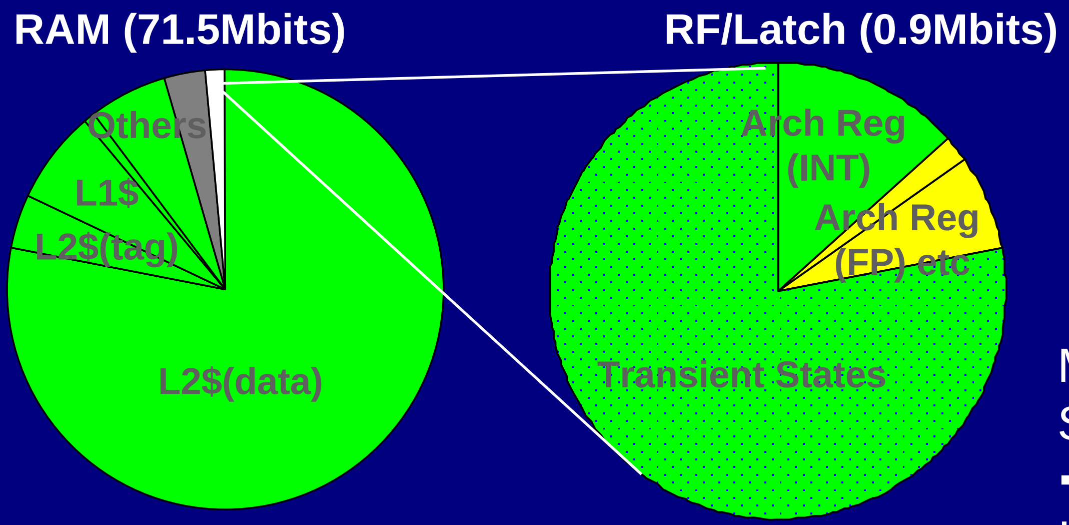
Cache Protection	Tag	ECC (L2$) Dupicate & Parity (L1$)
	Data	ECC (L2$, L1D$) Parity (L1I$)
Cache Dynamic Degradation		Yes
ALU/Register		ECC (INT Regs) Parity (FP Regs, etc)
HW Instruction Retry		Yes
History		Yes



- New RAS features of SPARC64™VII
  - Integer registers are ECC protected
  - Number of checkers increased to ~3,400

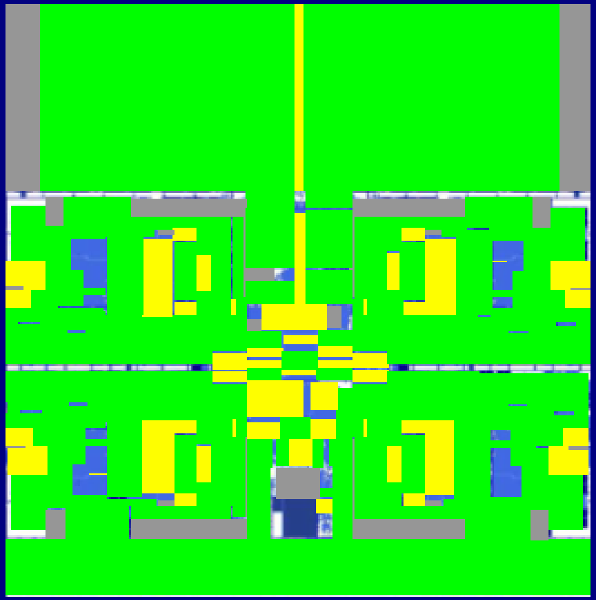
SPARC64™ VII

RAS coverage



Green: 1bit error Correctable
Yellow: 1bit error Detectable
Gray: 1bit error harmless

More than 70% of Transient States are 1bit error detectable.
➔Recoverable through HW Instruction Retry



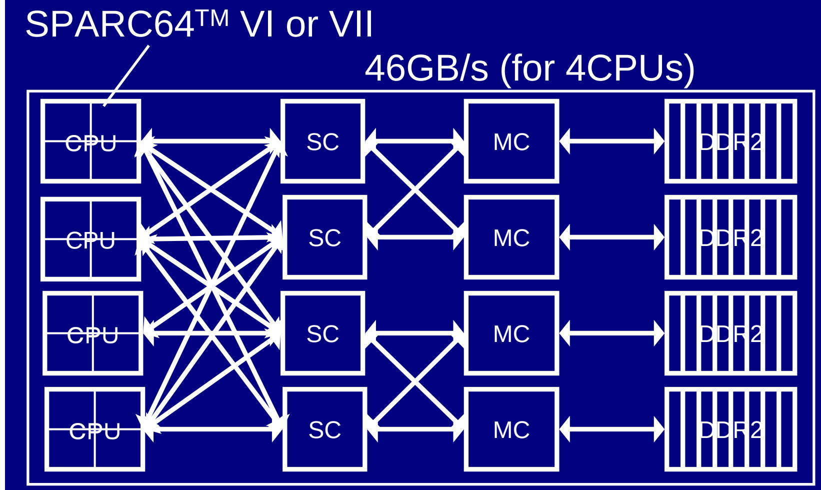
- All RAMs are ECC protected or Duplicated
- Most latches are parity protected

➔Guaranteed Data Integrity

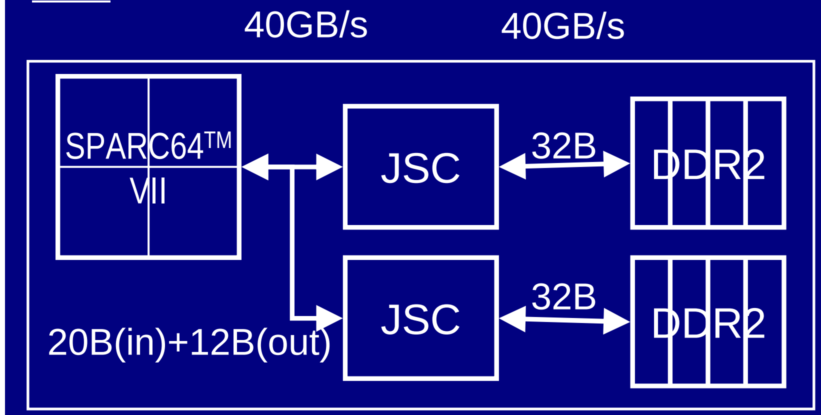
SPARC64™ VII

Servers based on SPARC64^{TM} VII

SPARC Enterprise M8000



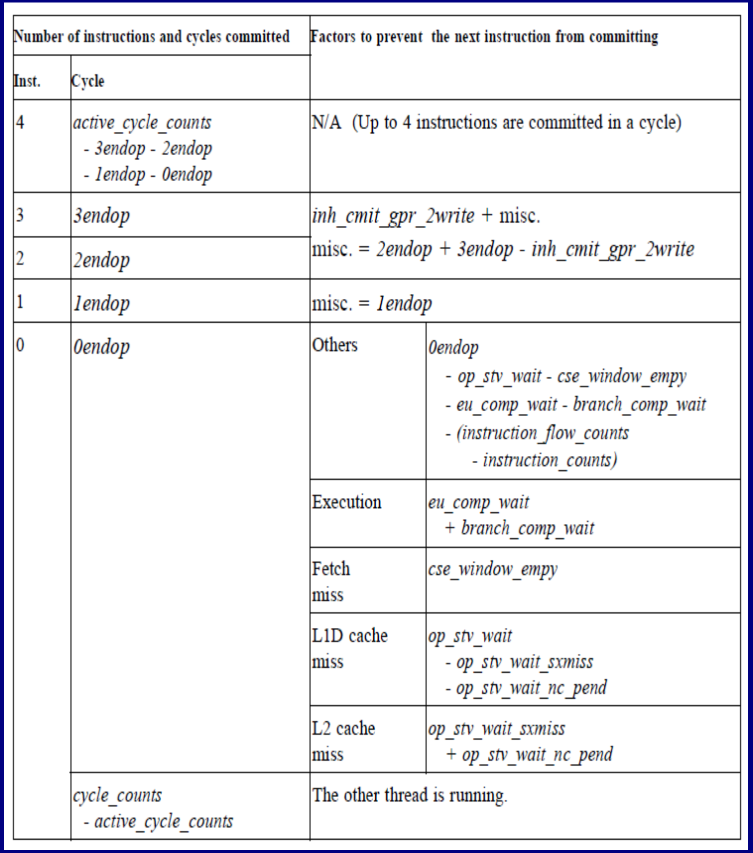
FX1



- **SPARC Enterprise: UNIX**
  - Max 64CPUs / SMP
  - Upgradeable from SPARC64^{TM} VI
  - Possible to mix SPARC64^{TM} VI and SPARC64^{TM} VII within the same SB
- **FX1: Technical Computing**
  - Single CPU node
  - Increased Jupiter Bus freq = 1.26GHz
  - System chip is newly designed to realize
    - Higher memory throughput
    - Lower Memory latency
    - Smaller footprint
  - Performance
    - FP throughput/socket: x1.5
    - STREAM benchmark: >13.5GB/s

SPARC64^{TM} VII

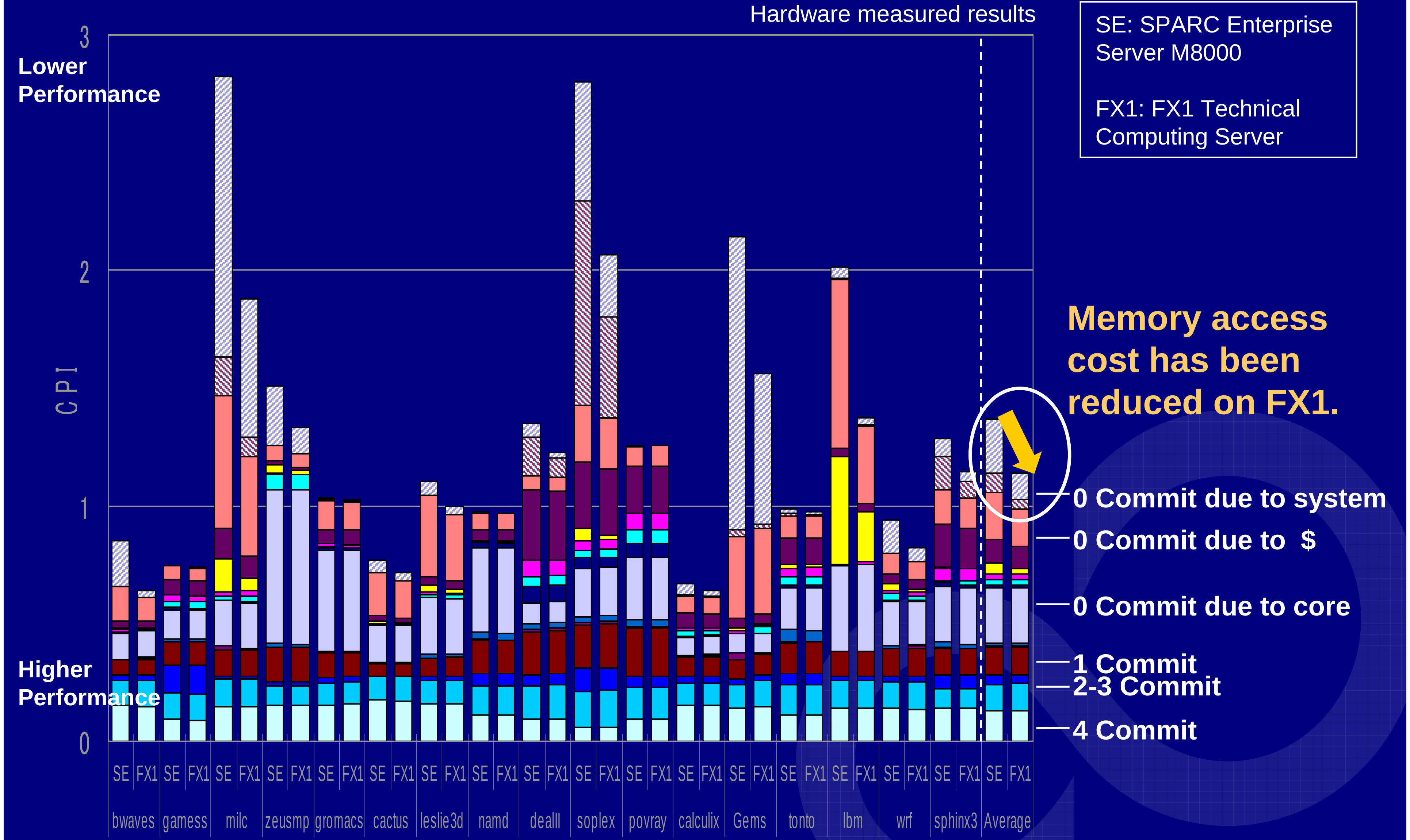
Performance Analysis



Number of instructions and cycles committed		Factors to prevent the next instruction from committing	
Inst.	Cycle		
4	*active_cycle_counts* - *3endop* - *2endop* - *1endop* - *0endop*	N/A (Up to 4 instructions are committed in a cycle)	
3	*3endop*	*inh_cmit_gpr_2write* + misc.	
2	*2endop*	misc. = *2endop* + *3endop* - *inh_cmit_gpr_2write*	
1	*1endop*	misc. = *1endop*	
0	*0endop*	Others	*0endop* - *op_stv_wait* - *cse_window_empy* - *eu_comp_wait* - *branch_comp_wait* - *(instruction_flow_counts* - *instruction_counts)*
		Execution	*eu_comp_wait* + *branch_comp_wait*
		Fetch miss	*cse_window_empy*
		L1D cache miss	*op_stv_wait* - *op_stv_wait_sxmiss* - *op_stv_wait_nc_pend*
		L2 cache miss	*op_stv_wait_sxmiss* + *op_stv_wait_nc_pend*
	cycle_counts - *active_cycle_counts*	The other thread is running.	

- Performance Analyzer
  - About 100 performance events can be monitored
  - Available through cputrack() and cpustat()
  - 8 performance events can be gathered at the same time.
- Commit base performance events
  - Number of commit instructions each cycle.
  - The cause of no commit
    - L2$miss
    - L1D$miss
    - Fetch miss
    - Execution Unit busy
    - …

SPARC64^{TM} VII CPI (Cycle Per Instruction) Example



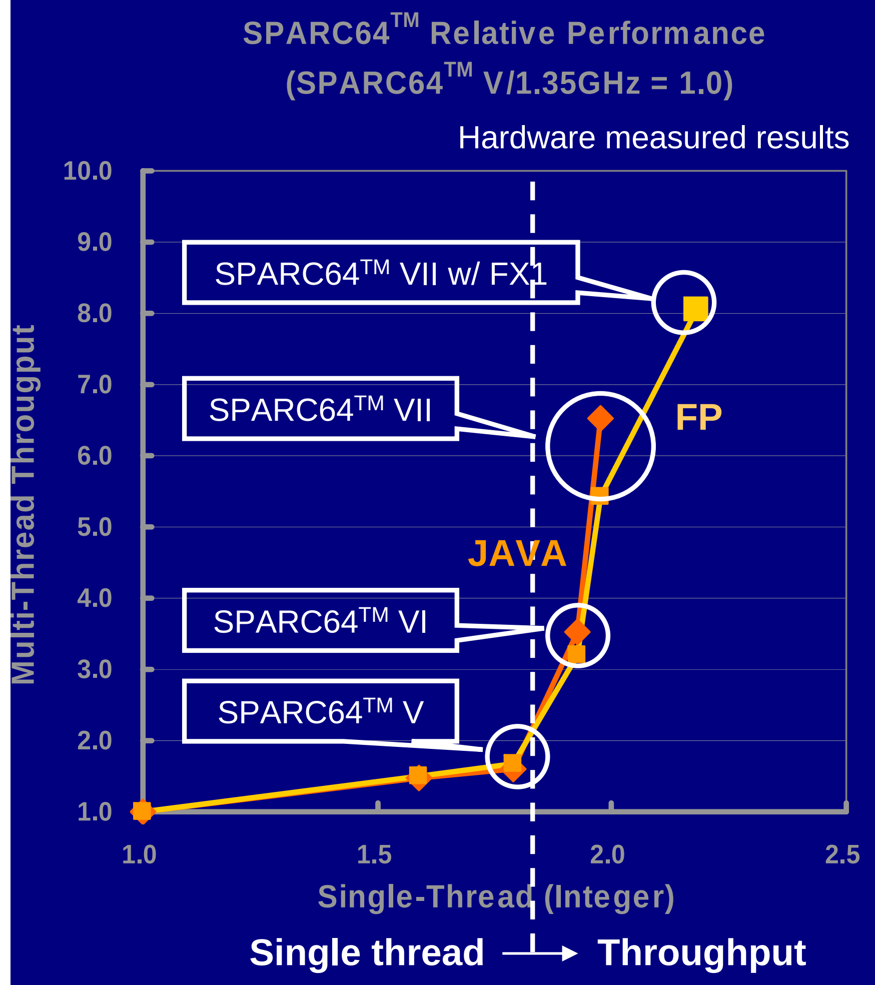
SE: SPARC Enterprise Server M8000

FX1: FX1 Technical Computing Server

Memory access cost has been reduced on FX1.

SPARC64^{TM} VII

SPARC64™ Future



- Design History:
  Evolution rather than revolution

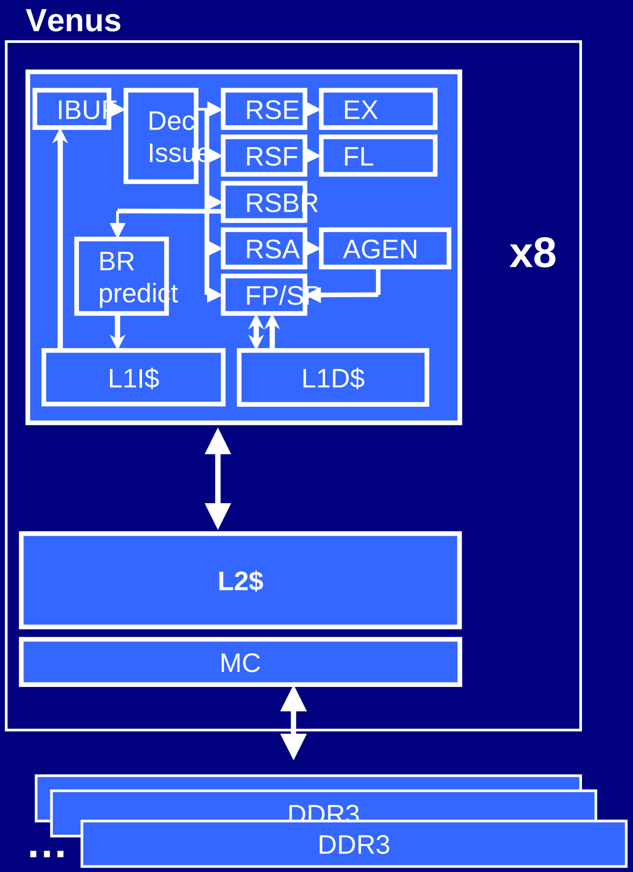
- SPARC64™ V (1core)
  - ✓RAS
  - ✓Single thread performance
- SPARC64™ VI (2core x 2VMT)
  - ✓Throughput
- SPARC64™ VII (4core x 2SMT)
  - ✓More Throughput
  - ✓High Performance Computing

- What's next?

SPARC64^{TM} VII Summary

- The same system I/F with existing SPARC64^{TM} VI to protect customer's investment
- 4core x 2SMT design realizes high throughput without sacrificing single thread performance
- Shared L2$ and HW barrier makes 4 cores behave as a single processor with compiler technology.
- The new system design has fully exploited potential of SPARC64^{TM} VII.
- Fujitsu will continue to develop SPARC64^{TM} series to meet the needs of a new era.

Venus



- For PETA-scale Computing Server
  - ✓ 8core
  - ✓ Embedded Memory Controller
- SPARC-V9 + extension (HPC-ACE)
  - ✓ SIMD
- Fujitsu 45nm CMOS

➔ 128GF@socket

Abbreviations

- SPARC64^{TM} VII

– RSA:	Reservation Station for Address generation
– RSE:	Reservation Station for Execution
– RSF:	Reservation Station for Floating-point
– RSBR:	Reservation Station for Branch
– GUB:	General Update Buffer
– FUB:	Floating point Update Buffer
– GPR:	General Purpose Register
– FPR:	Floating Point Register
– CSE:	Commit Stack Entry
– FP:	Fetch Port
– SP:	Store Port

- Chip-sets

– SC:	System Controller
– MC:	Memory Controller
– JSC:	Jupiter System Controller
– SB:	System Board

- Others
  - DGEMM: Double precision matrix multiply and add