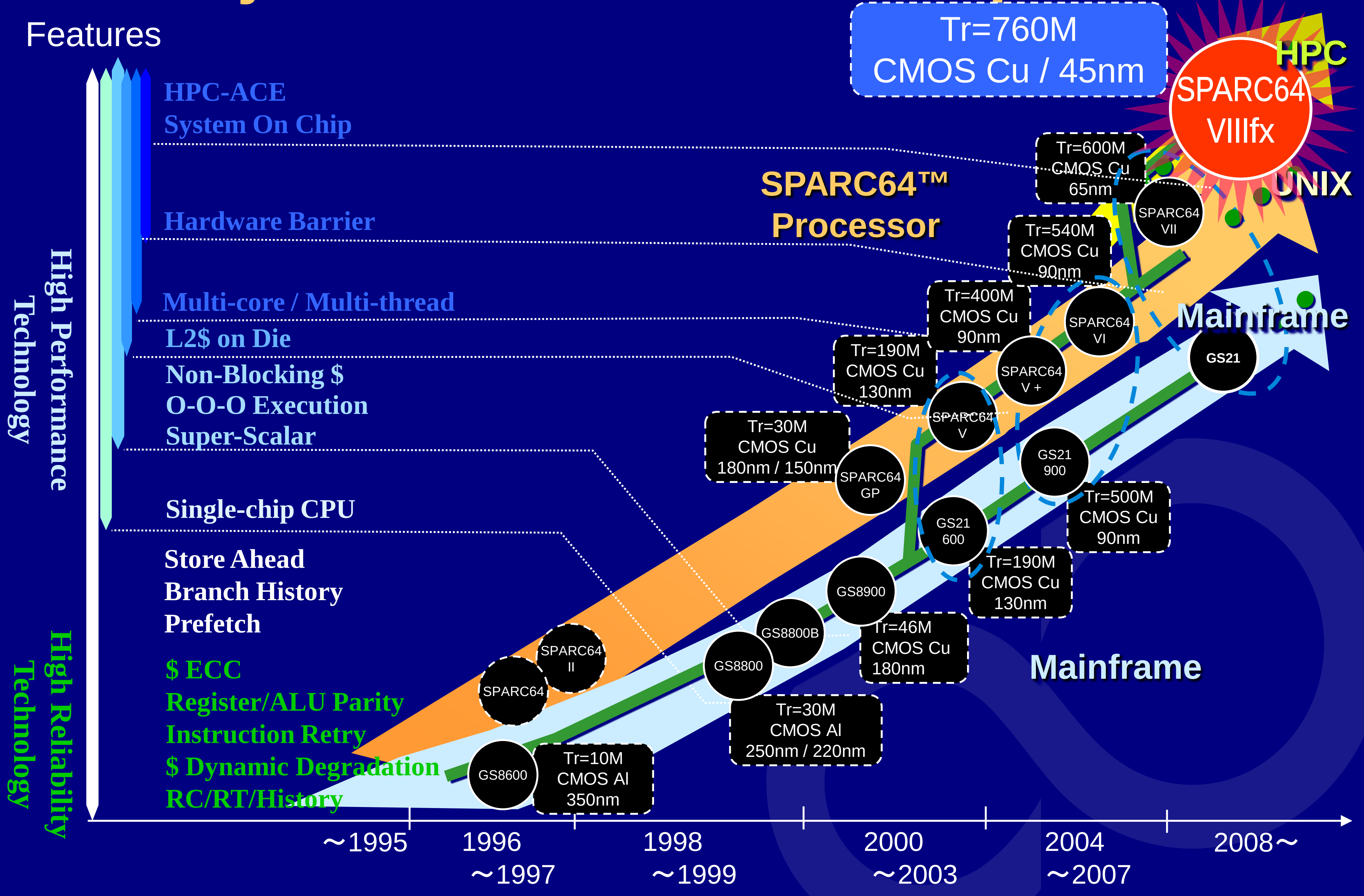
SPARC64™ VIIIfx: Fujitsu's New Generation Octo Core Processor for PETA Scale computing

August 25, 2009

Takumi Maruyama

LSI Development Division
Next Generation Technical Computing Unit
Fujitsu Limited

Fujitsu Processor Development



SPARC64™ VIIIfx Design Target

- Processor for Fujitsu's supercomputer for the peta-scale computing age, which realizes both **high performance** and **low power**
  - Unachievable goal with conventional processor design
    - More GF (Giga Flops)
    - More Efficiency
    - No more high frequency
  - → Large ISA (Instruction Set Architecture) extension is required

  Focus of this presentation

  - High Integration: SoC (System On Chip)
  - High Reliability

- Reuse SPARC64^{TM} VII design when applicable

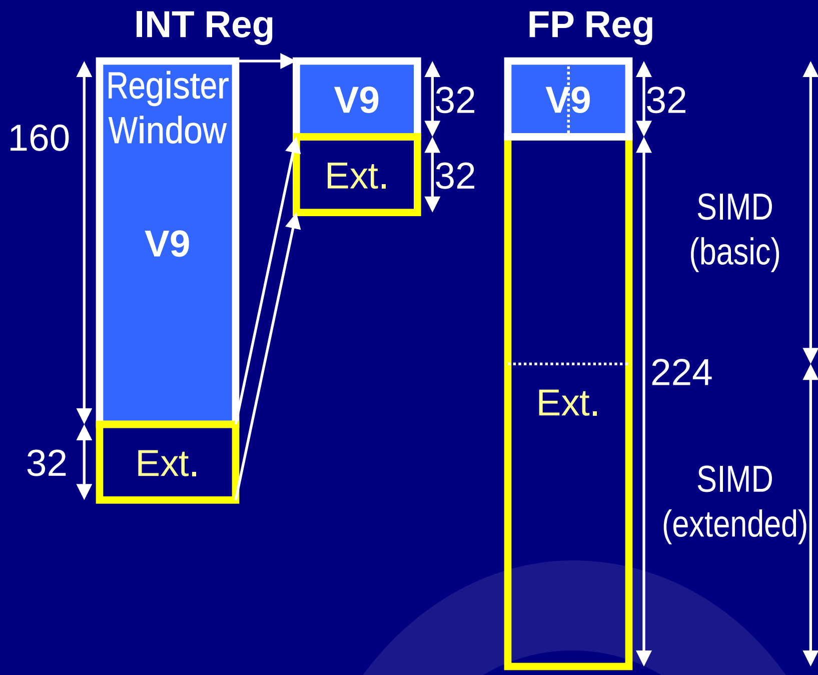
HPC-ACE
(High Performance Computing - Arithmetic Computational Extensions)

- ISA (Instruction Set Architecture) of SPARC64VIIIfx is:
  - Complies with
    - The SPARC-V9 standard
    - JPS (Joint Programmer's Specification): Extension to SPARC-V9
  - HPC-ACE: Fujitsu's unique ISA extension for HPC
    - Large register sets
    - SIMD (single instruction multiple data) instructions
    - etc

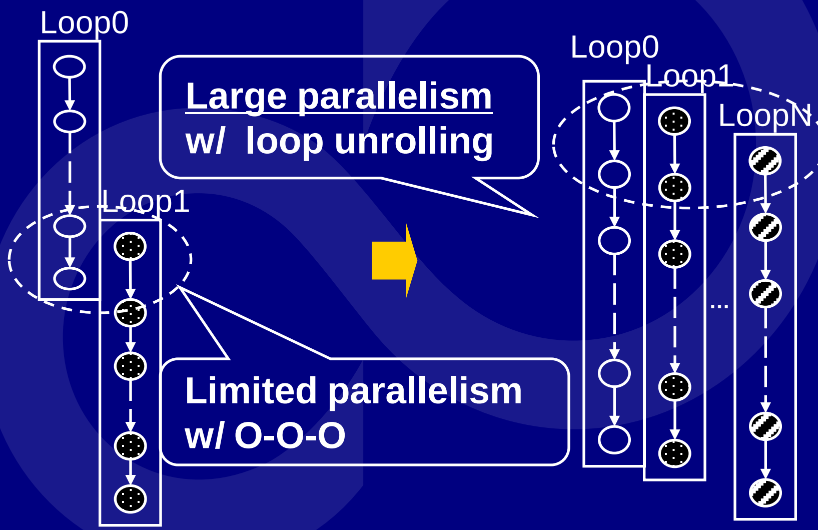
- You can download SPARC64™ VIIIfx ISA documents from http://jp.fujitsu.com/solutions/hpc/brochures/
  - The SPARC® Architecture Manual Version 9
  - SPARC® Joint Programming Specification (JPS1): Commonality
  - SPARC64 VIIIfx Extensions

Large register sets 1/2

- Register enhancements from V9
  - 32→64 INT registers
  - 32→256 FP registers (DP)
    - Lower 32 registers are the same with SPARC-V9's.
    - FP registers are "flat". Extended FP registers can be accessed with non-SIMD instruction as well.



- Why needed
  - To extract more parallelism currently limited by the number of Arch registers.
  - Reduce spill/fill overhead

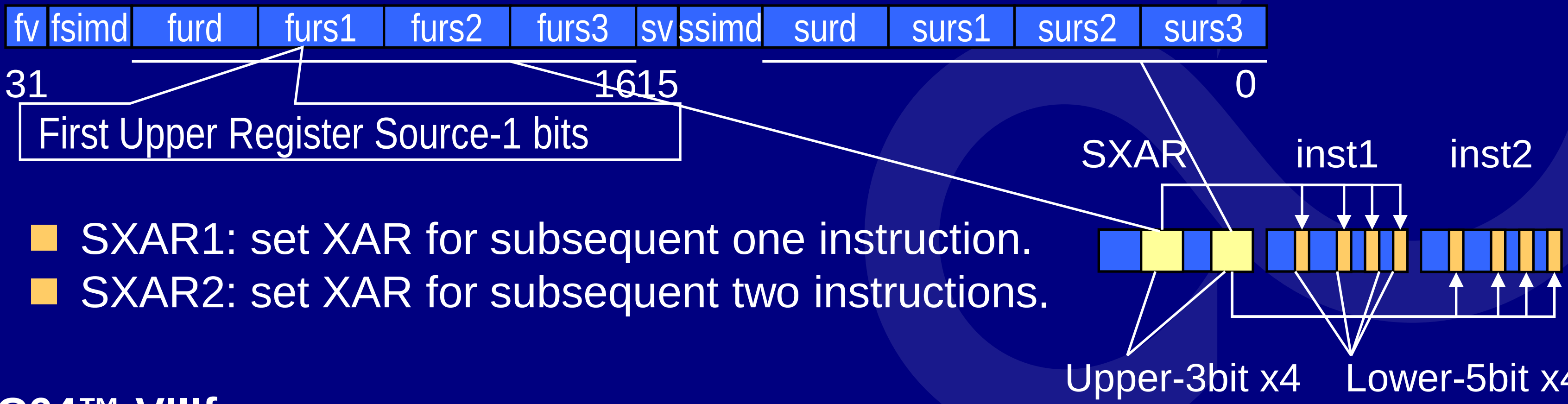

Loop0
Loop1
Limited parallelism w/ O-O-O
Large parallelism w/ loop unrolling
Loop0
Loop1
LoopN
…

Large register sets 2/2

- Instruction format for 256 FP registers
  - 8 bit x 4 (3 read +1 write) register number fields are necessary for FMA (Floating-point Multiply and Add) instruction.
  - But SPARC-V9 instruction length is limited (32bits – fixed)

➔ Defined a new prefix instruction (SXAR) to specify upper-3bit of register numbers of the following two instructions.

- SXAR (Set XAR) instruction
  - XAR: Extended Arithmetic Register
    - Set by the SXAR instruction
    - Valid bit is cleared once the corresponding subsequent instruction gets executed.

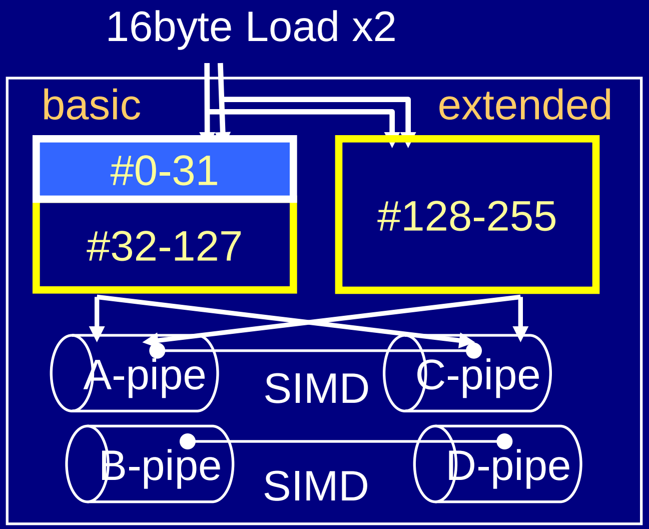


  - SXAR1: set XAR for subsequent one instruction.
  - SXAR2: set XAR for subsequent two instructions.

SIMD

- SIMD FP operation
  - 1 SIMD FP instruction executes two SP or DP floating-point operations
- SIMD Load/Store
  - 1 SIMD load/store instruction accesses two contiguous SP or DP data in memory
  - Data alignment requirement for SIMD-load (DP) is 8 byte rather than usual 16byte.
  - Combine two independent FP operation into one with the Flat FP registers
- SXAR instruction
  - Specifies SIMD operation of the subsequent two instructions.
  - No new opcode is required for SIMD operation

➔ More SW optimization possibility with SIMD



Conventional SIMD usage

SIMD-Load (Src1)
SIMD-Load (Src2)
SIMD-FP
SIMD-Store (Dst)

Combine two FP ops with SIMD

Load (Src1-A)
Load (Src1-B)
Load (Src2-A)
Load (Src2-B)
SIMD-FP
Store (Dst-A)
Store (Dst-B)

FP Trigonometric Functions

- Instructions for fast Trigonometric Function calculation

- Taylor series approximation of sin (x)

$$\sim= x - 1/3!x^3 + 1/5!x^5 - 1/7!x^7 + 1/9!x^9 - 1/11x^{11} + 1/13x^{13} - 1/15x^{15}$$

$$= x\,((((((((0 \underline{- 1/15}!)x^2 \underline{+ 1/13}!)x^2 \underline{- 1/11}!)x^2 \underline{+ 1/9}!)x^2 \underline{- 1/7}!)x^2 \underline{+ 1/5}!)x^2 \underline{- 1/3}!)x^2 + 1)$$

- New instruction - ftrimaddd
  - Function: rs1 $\times$ abs(rs2) + $\boxed{\text{T[index]}}$ $\rightarrow$ rd
  - Usage:

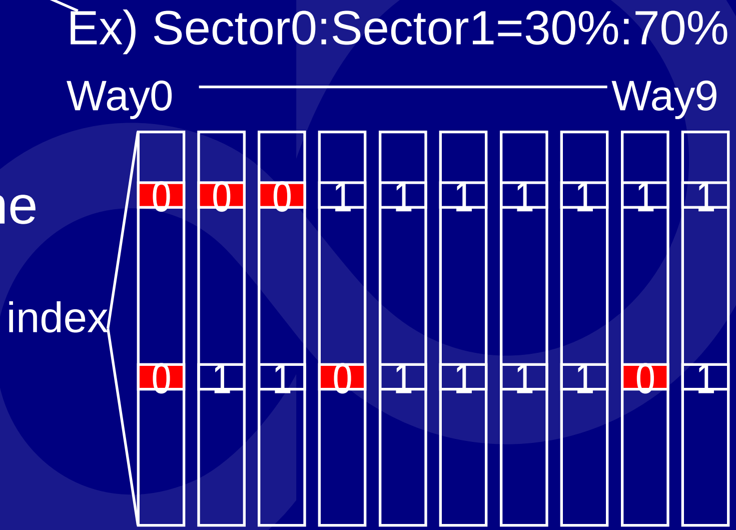
```
                                # S = 0
ftrimaddd S, X², 7, S    # S * X²  – 1/15! → S
ftrimaddd S, X², 6, S    # S * X²  + 1/13! → S
ftrimaddd S, X², 5, S    # S * X²  – 1/11! → S
ftrimaddd S, X² , 4, S   # S * X²  + 1/9! → S
ftrimaddd S, X² , 3, S   # S * X²  – 1/7! → S
ftrimaddd S, X² , 2, S   # S * X²  + 1/5! → S
ftrimaddd S, X² , 1, S   # S * X²  – 1/3! → S
Fmuld     S, X, S        # S * X → S
```

Software controlled Cache

- Give SW to ability to control cache to optimize performance while keeping cache coherency
- Divide Cache into 2 groups (sectors)
  - SXAR instruction specifies the sector
    - sector 0: Instruction fetch / normal operand access (default)
    - sector 1: operand access explicitly specified by SXAR
  - Sector cache configuration register
    Specifies the ratio of sector 0 and 1 at the same index
- HW Implementation
  - Keep sector info of each cache line.
  - Select the way to be replaced to meet the sector ratio on cache misses

➔ SW specifies a sector depending on temporal and spatial locality of data

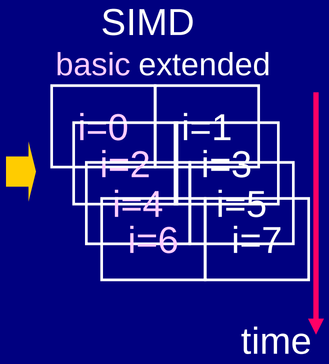
L2 Cache structure



Other HPC-ACE features

- Conditional operation
  - For efficient execution of Loop with 'if'
  - Conditional branch can be removed with the following conditional instructions.
    - FP Conditional Compare to Register
    - Move Selected FP-register on FP-register's condition
    - Store FP-register on FP-register's condition
  - ➔ Compiler can optimize the loop with SW pipeline

```
for (i=0; i<00; i++) {
  if (A(i)>0)
    FPop X(i)
  else
    FPop Y(i)
}
```



- FP Reciprocal Approximation of Divide/Square-root
  - Calculate reciprocal approximation with rounding error < 1/256
  - To achieve higher divide/square-root performance with pipelined operation
- FP Minimum and Maximum

```
Usage:
# %f2 / %f10 → %f0
frcpad   %f10, %f6
fmuld    %f2, %f6, %f2
fnmsubd  %f6, %f10, 1.0, %f6
fmuld    %f6, %f6, %f0
fmaddd   %f6, %f6, %f6, %f4
fmaddd   %f0, %f0, %f6, %f0
fmaddd   %f4, %f2, %f2, %f4
fmaddd   %f0, %f4, %f2, %f0
```

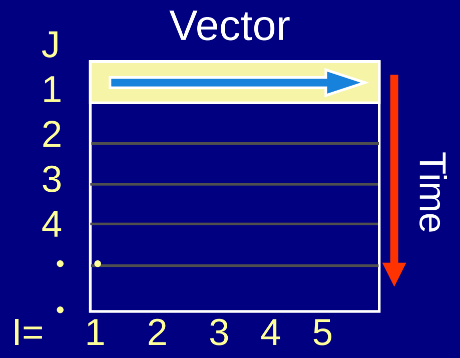
Integrated Multicore Parallel Architecture

- **SPARC64™ VIIIfx HW**
  - HPC-ACE
  - Shared L2 cache to avoid false sharing
  - Hardware barrier for fast inter-core synchronization
- **Fujitsu's compiler technology**
  - Automatic parallelization

➔ More possibility of optimization: <u>Parallelize the innermost loop</u>

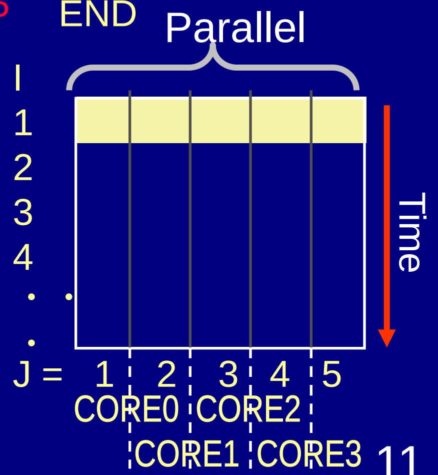
●Vector

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



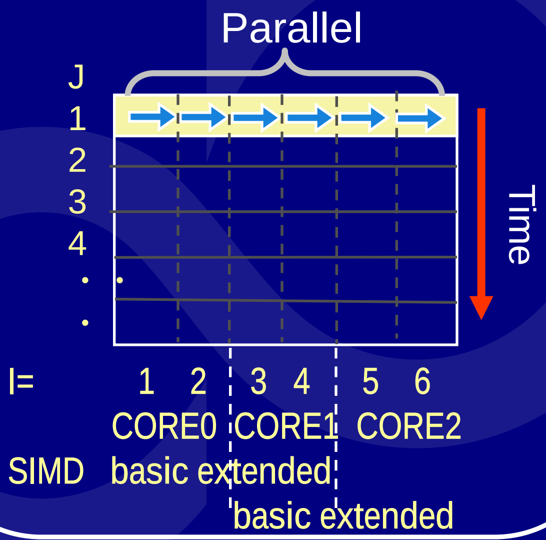
●Conventional Scalar

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

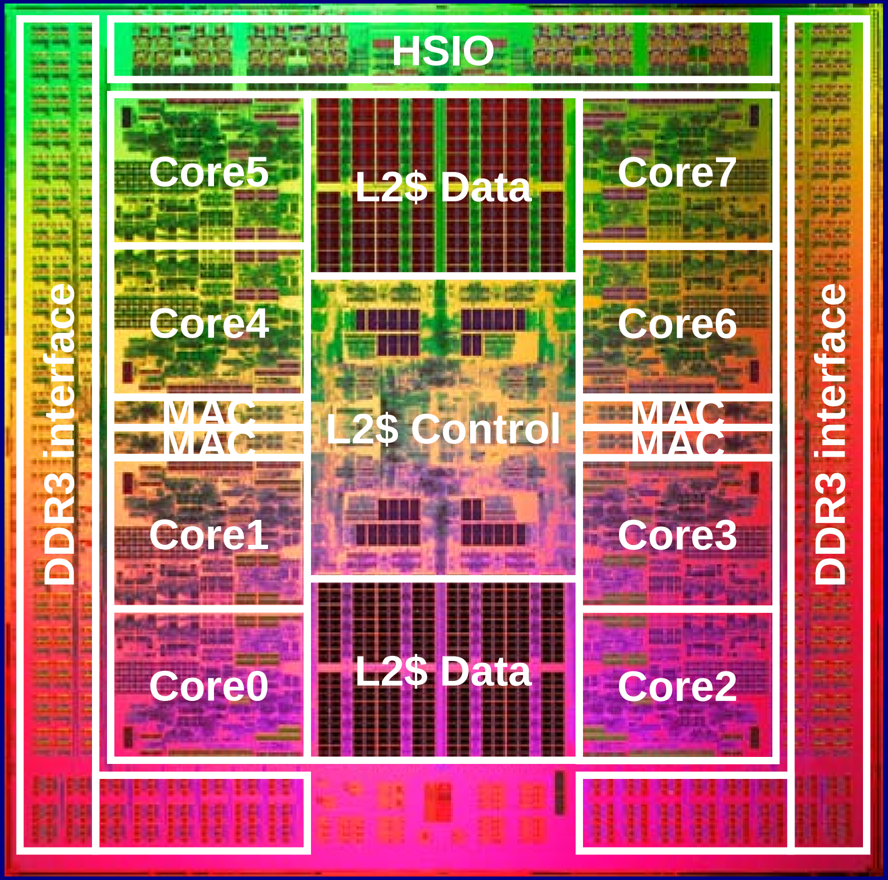


●SPARC64™ VIIIfx

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

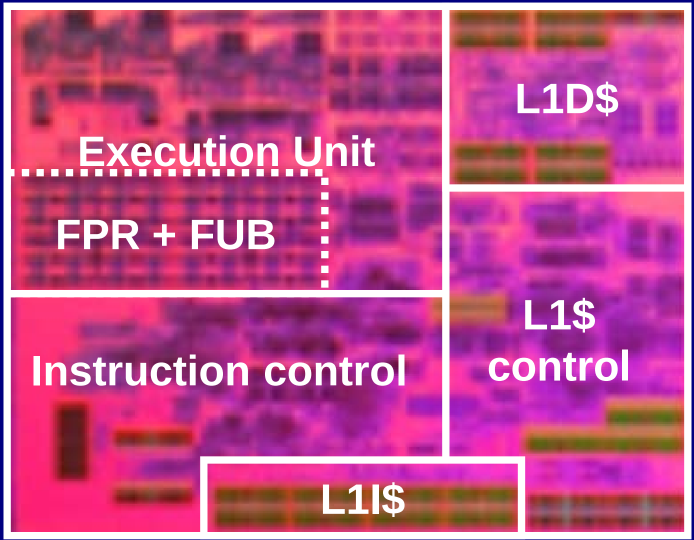


SPARC64™ VIIIfx Chip Overview



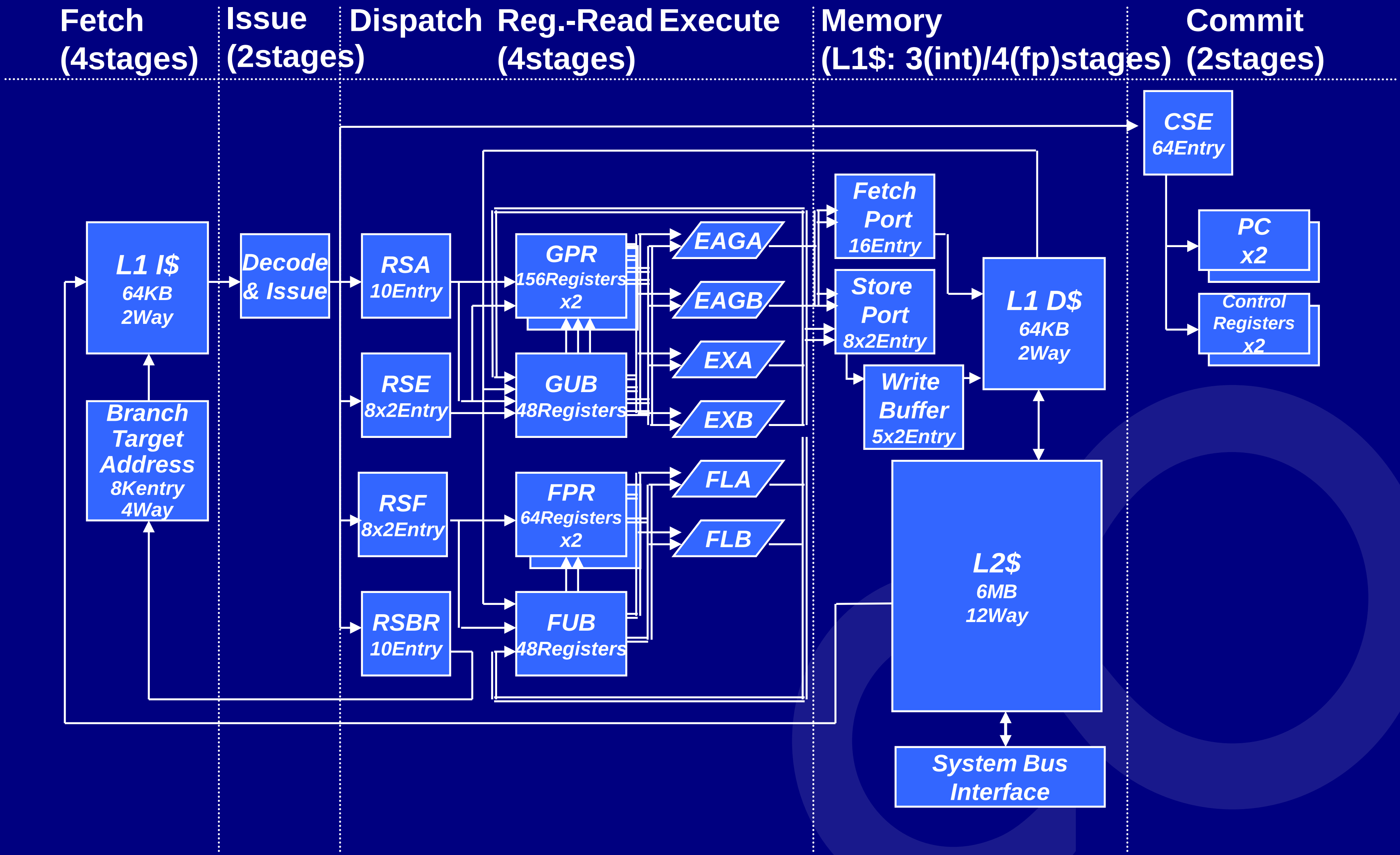
- **Architecture Features**
  - 8 cores
  - Shared 5 MB L2$
  - Embedded Memory Controller
  - 2 GHz
- **Fujitsu 45nm CMOS**
  - 22.7mm x 22.6mm
  - 760M transistors
  - 1271 signal pins
- **Performance (peak)**
  - 128GFlops
  - 64GB/s memory throughput
- **Power**
  - 58W (TYP, 30℃)
  - Water Cooling – Low leakage power and High reliability

SPARC64™ VIIIfx Core spec

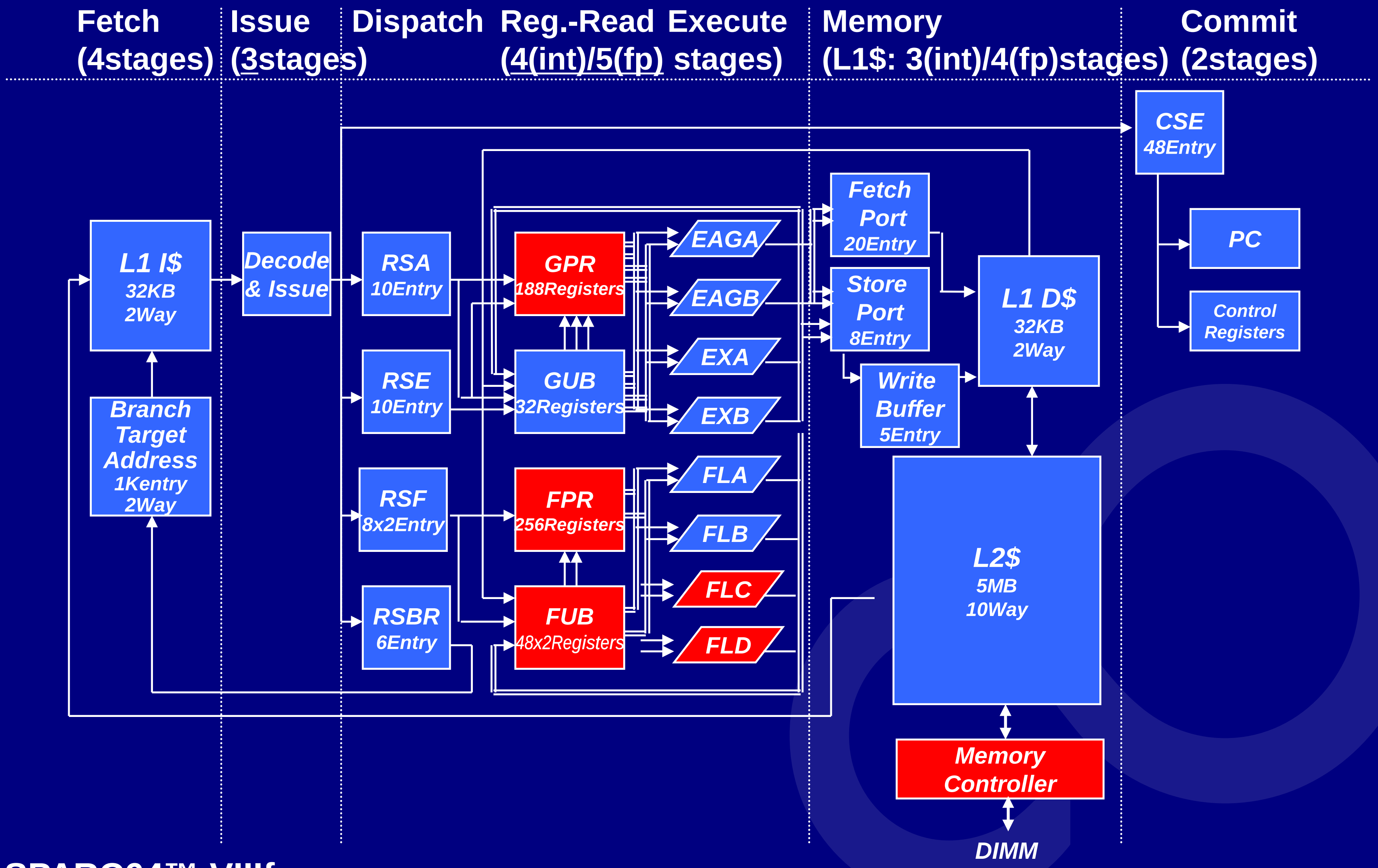


Instruction Set Architecture	SPARC-V9/JPS + HPC-ACE	
#FP-operations per clock	8 (= 4 Floating Multiply and Add)	
Execution units	<integer>	<floating-point>
	ALU x2 SHIFT x2 MULT x1 DIVIDE x1 AGEN x2	FMA x4 (2SIMD) COMPARE x2 DIVIDE x2 VIS x1
#Registers	188 (GPR) 32 (GUB)	256 (FPR) 48 x2 (FUB)
L1$	L1I$ 32KB/2way L1D$ 32KB/2way	

SPARC64^{TM} VII Pipeline



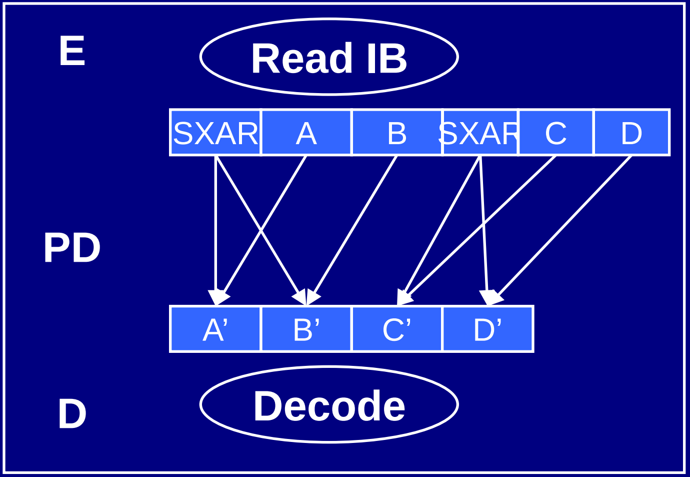
SPARC64™ VIIIfx Pipeline

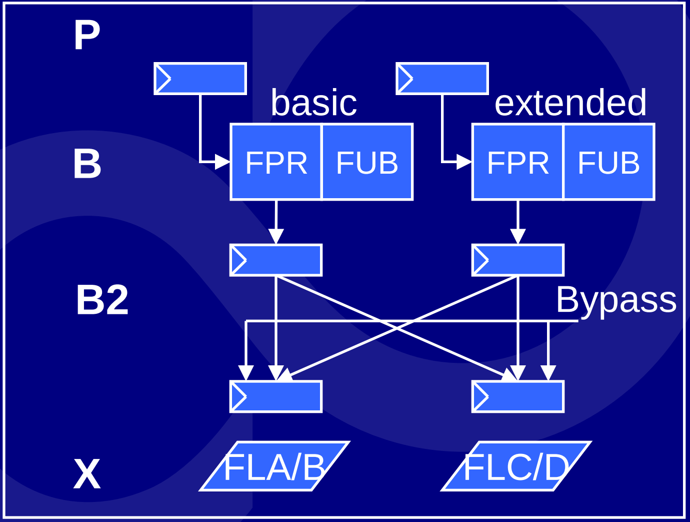


Changes in the pipeline

- Issue stage
  - Added “PD” stage to handle SXAR
  - 6-instructions packed into 4 at “PD”
  - An CSE entry is assigned at “D”
  - The packed instructions have
    - Extended register fields
    - SIMD operation attributes
  - 4 packed (=6 unpacked) instructions can be committed at the same time.

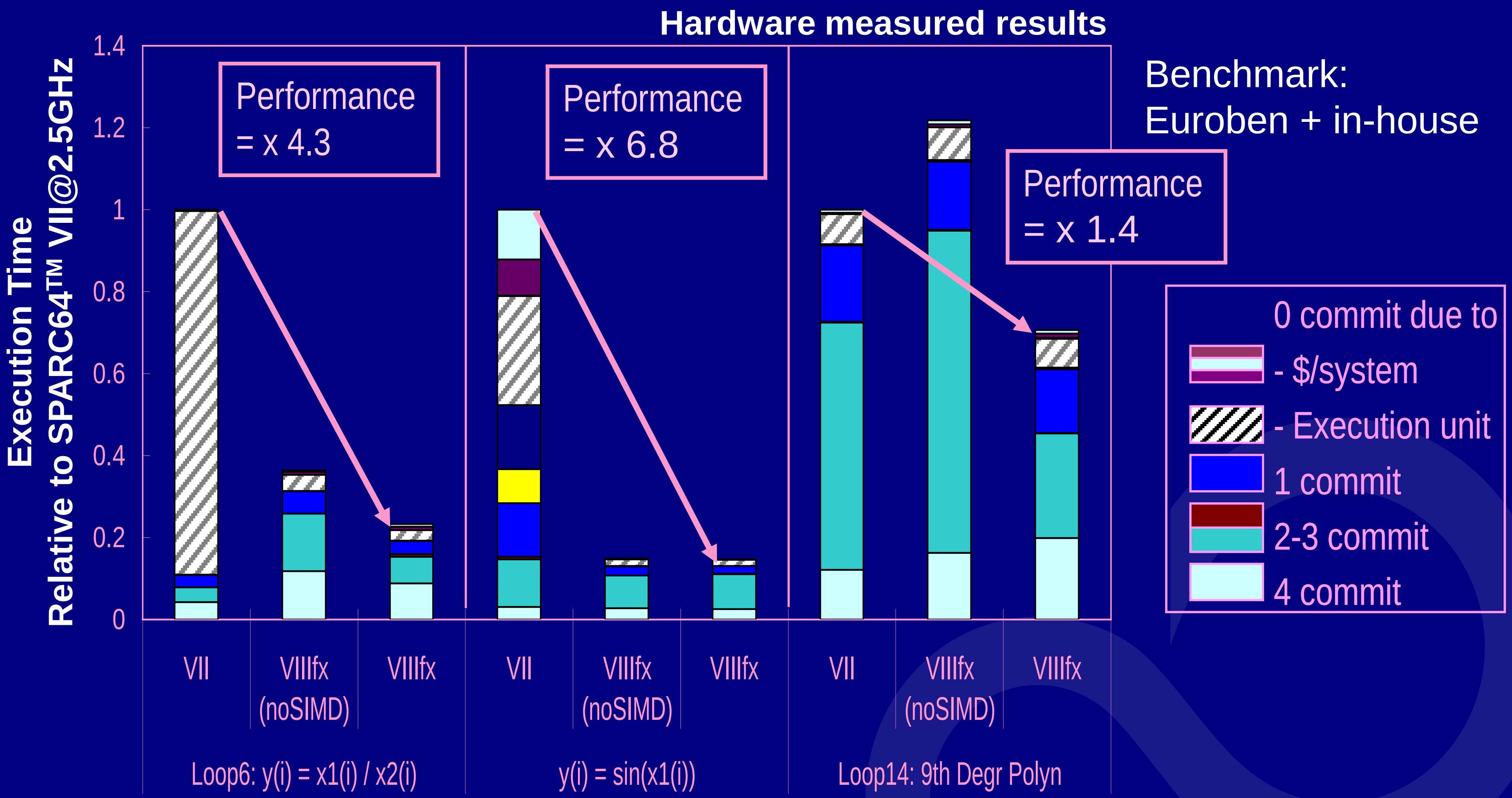
- Execution Stage
  - “B” stage has been split into two for flat Floating-point Register access





Performance Results – Core

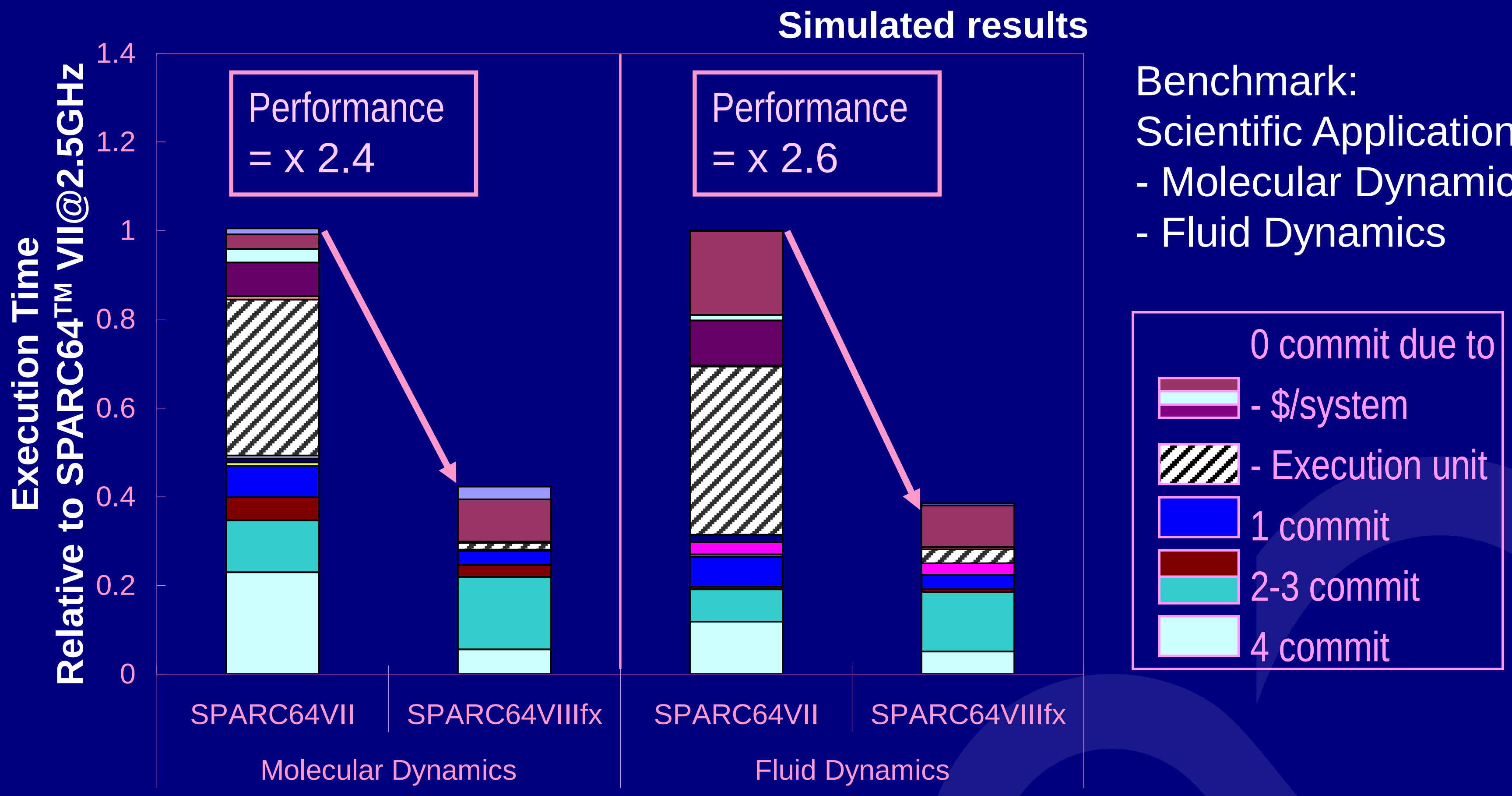
SPARC64^{TM} VIIIfx@2.0GHz



➔ SPARC64^{TM} VIIIfx realizes much higher core performance than SPARC64^{TM} VII thanks to HPC-ACE despite its lower frequency

Performance Results – Chip

SPARC64™ VIIIfx@2.0GHz



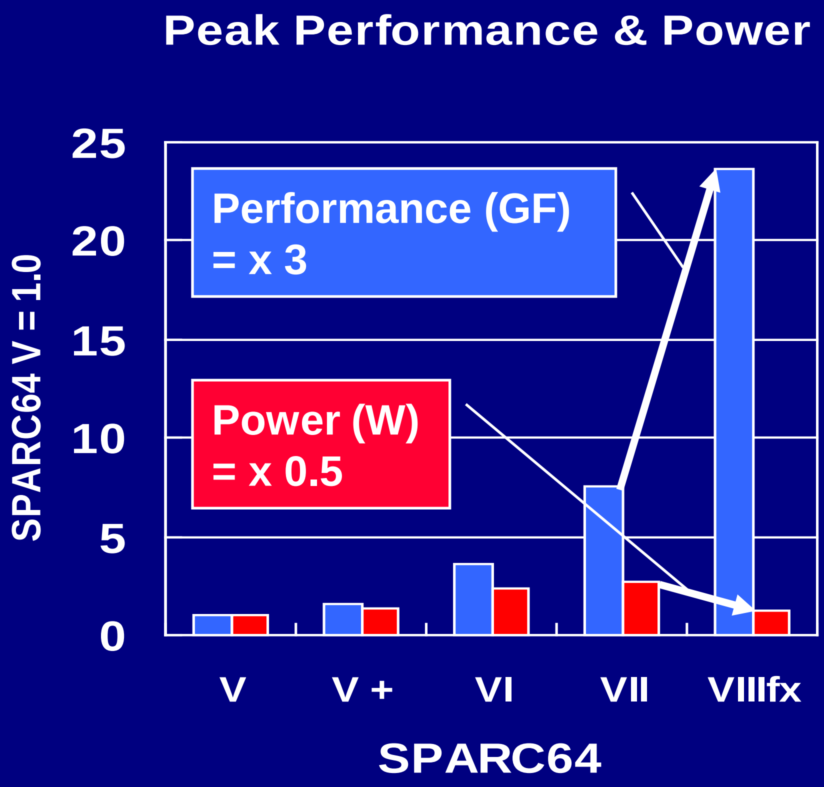
Benchmark:
Scientific Application
- Molecular Dynamics
- Fluid Dynamics

➔ SPARC64™ VIIIfx shows about x 2.5 performance of SPARC64™ VII
- Stall time due to the execution unit busy has been reduced dramatically.

➔ Expect x 3 performance with further compiler optimization

SPARC64™ History and Future

Evolution rather than revolution



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

- SPARC64™ VIIIfx (8 core)
  - High Performance Computing
  - Low Power
  - SoC

SPARC64^{TM} VIIIfx Summary

- SPARC64^{TM} VIIIfx has been designed to be used for Fujitsu's supercomputer for the PETA-scale computing age.
- HPC-ACE instruction sets overcomes the limitation of conventional SPARC-V9 architecture.
- SPARC64^{TM} VIIIfx has combined high performance and low power.
- SPARC64^{TM} VIIIfx chip is up and running in the lab.
- Fujitsu will continue to develop SPARC64^{TM} series to meet the needs of a new era.

Abbreviations

- SPARC64^{TM} VIIIfx

– IB:	Instruction Buffer
– 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