

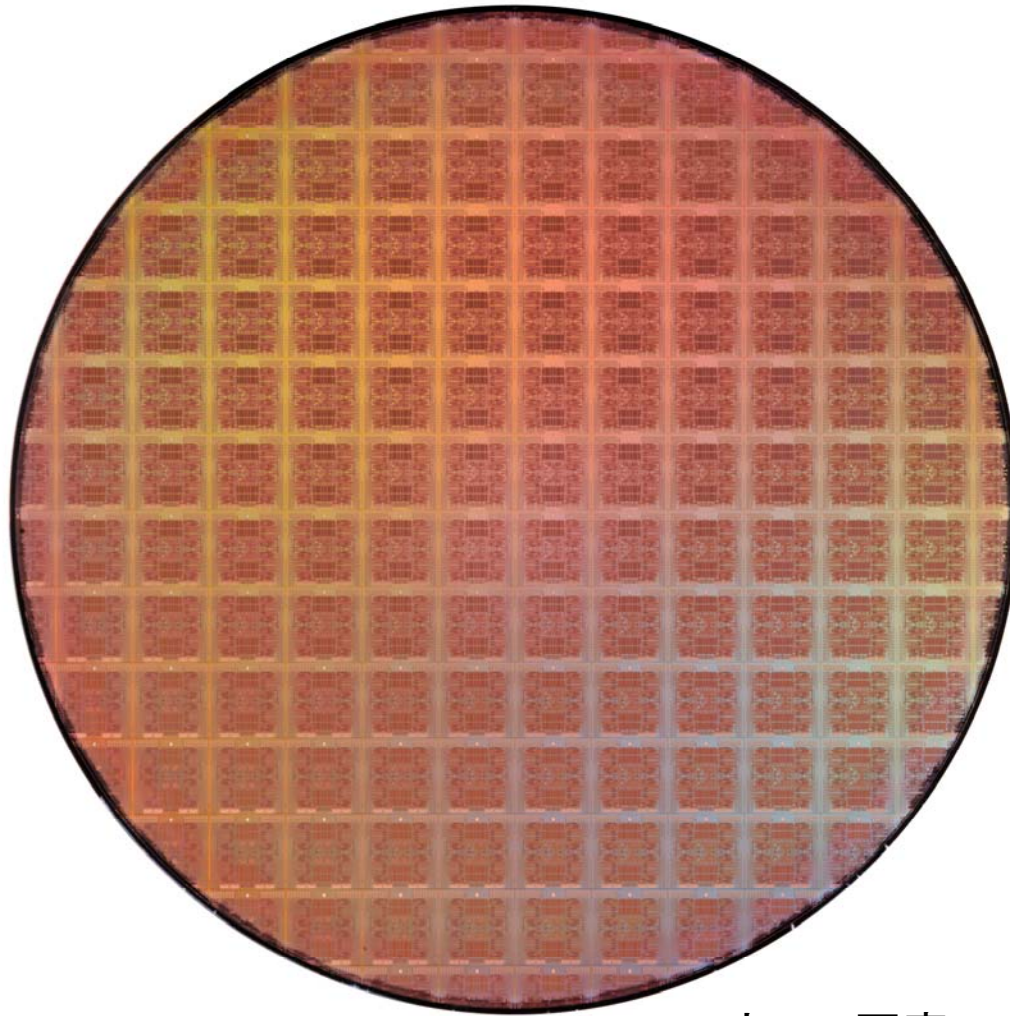
SPARC64TM VIIIfx

富士通次期スーパーコンピュータプロセサ

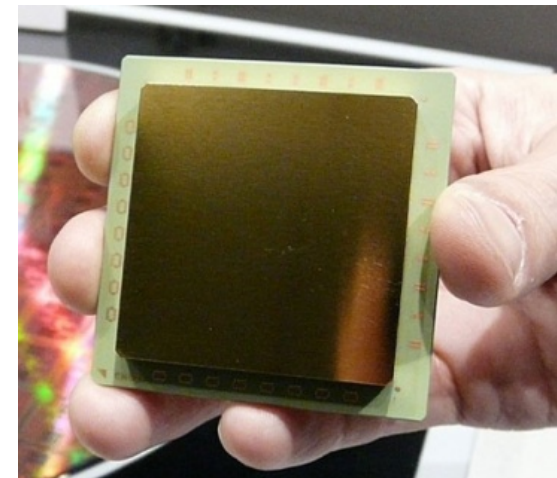
2010年1月28日

富士通株式会社
次世代テクニカルコンピューティング開発本部

山下 英男



ウェハ写真

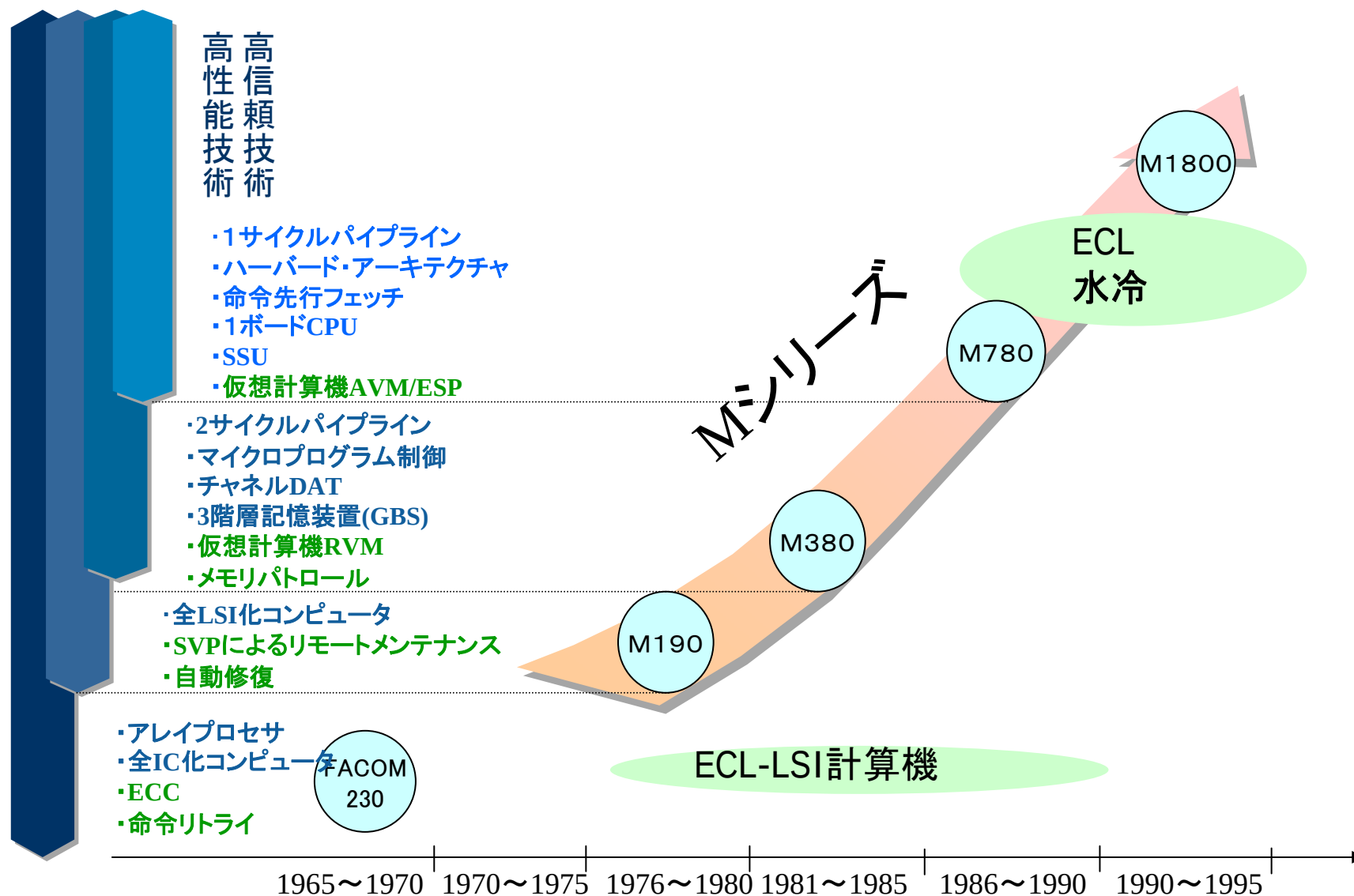


パッケージ写真

1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

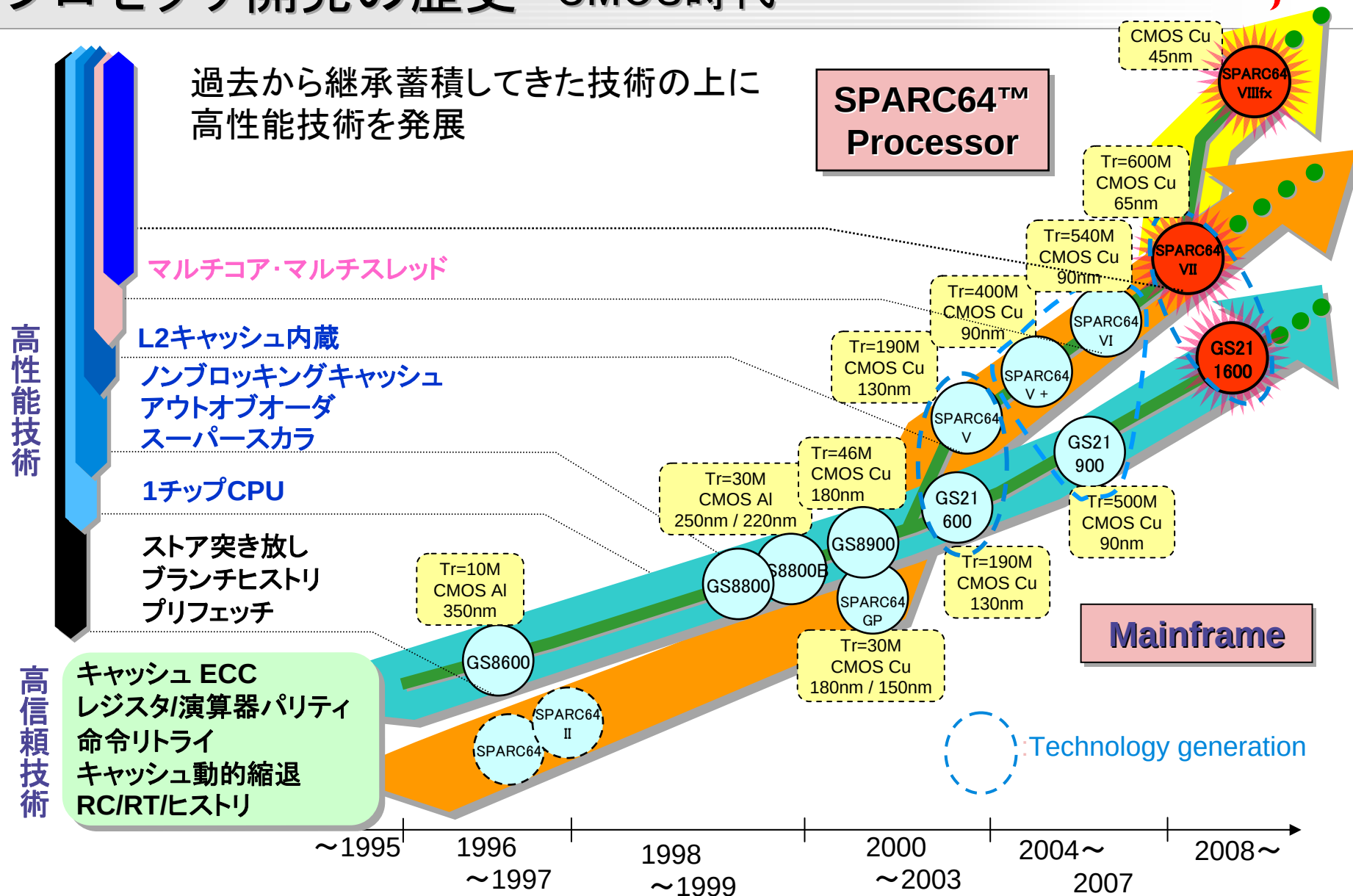
1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

汎用計算機開発の歴史



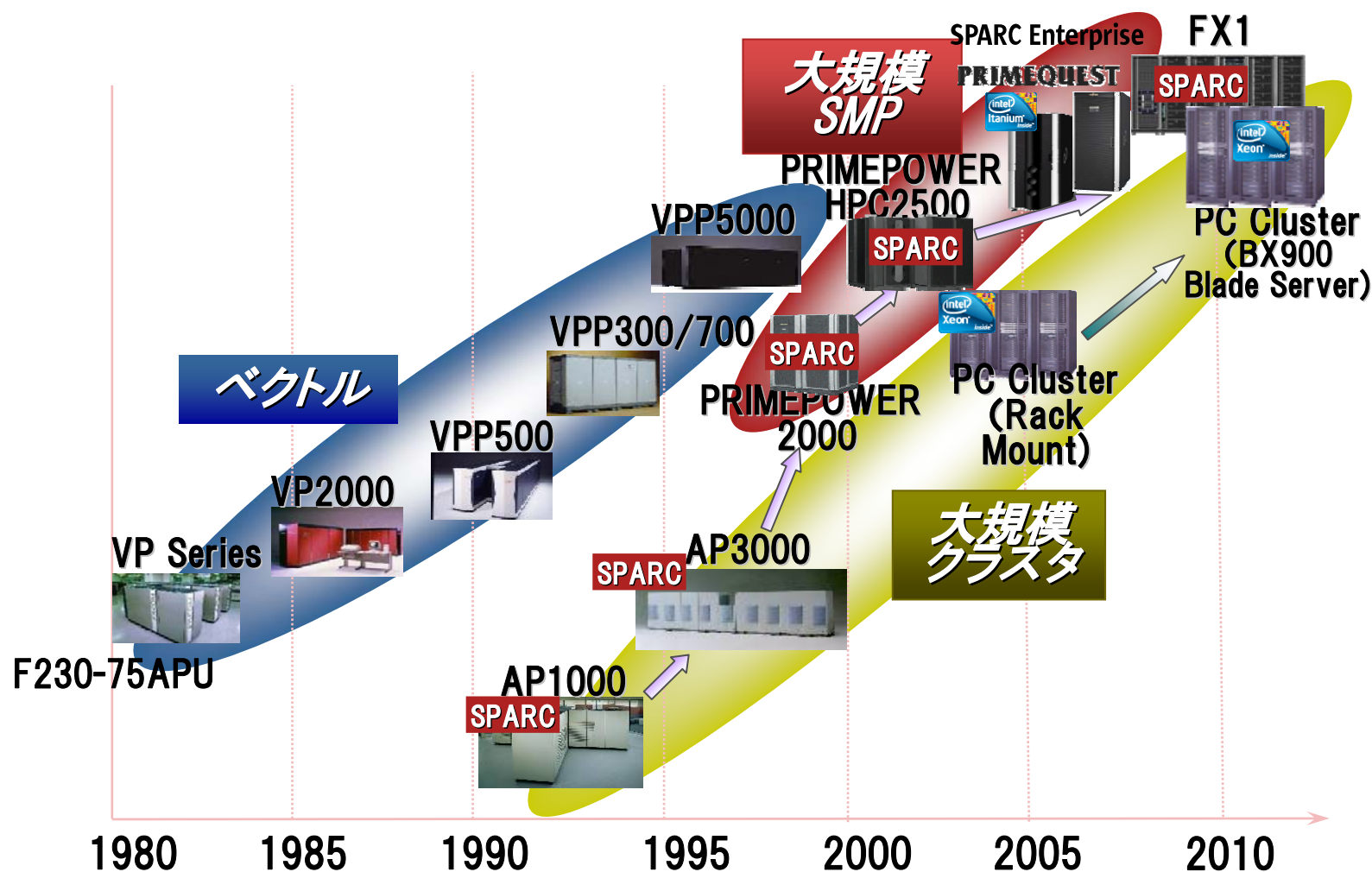
プロセッサ開発の歴史 CMOS時代

FUJITSU



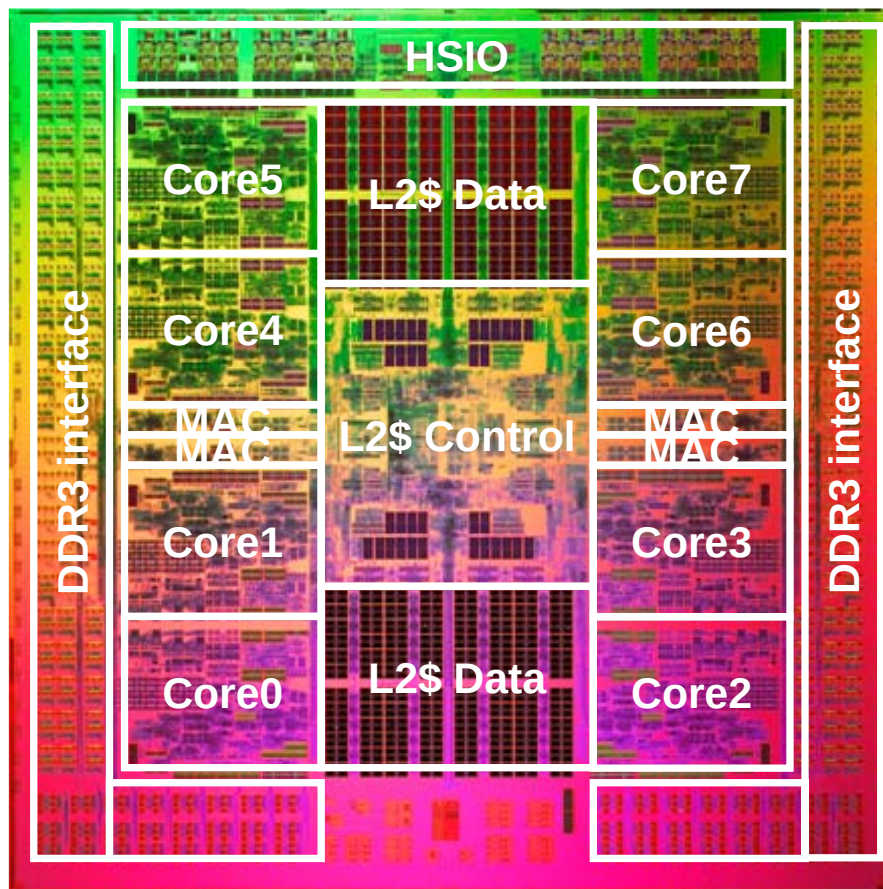
富士通スーパーコンピュータの歴史

- 1977年から現在まで世界最先端スーパーコンピュータを開発
- ベクトル式からSPARC64™ Vを搭載したスカラーシステムへ移行



1. 富士通のプロセッサ開発の歴史
- 2. SPARC64™ VIIIfxの概要**
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

SPARC64™ VIIIfx Chip 概要



◆ アーキテクチャ

- 8 コア
- 4命令同時発行/6命令同時実行
アウトオブオーダー, スーパースカラ
- 5 MB の共有L2キャッシュ
- メモリコントローラ内蔵
- 動作周波数 2GHz

◆ FMLの 45nm CMOS

- 22.7mm x 22.6mm
- 760M トランジスタ
- 信号ピン数 1271

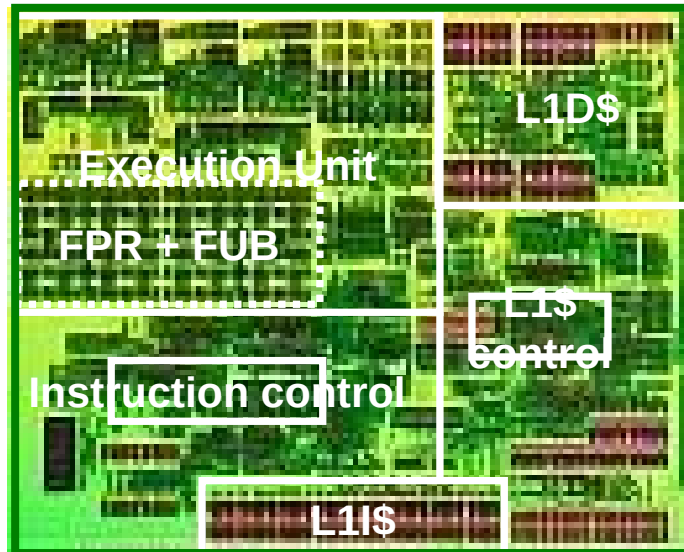
◆ ピーク性能

- 演算性能 128GFlops
- メモリスループット 64GB/s

◆ 消費電力

- 58W (TYP, 30°C)

SPARC64™ VIIIfx コア概要



演算器構成	固定小数点系	浮動小数点系
	ALU x2 SHIFT x2 MULT x1 DIVIDE x1 AGEN x2	FMA x4 (2SIMD) COMPARE x2 DIVIDE x2 VIS x1
レジスタ数	188 (GPR) 32 (GUB)	256 (FPR) 48 x2 (FUB)
L1キャッシュ	L1I\$ 32KB/2way L1D\$ 32KB/2way	

◆ 富士通の次期スーパーコンピュータに搭載するプロセサとして、 高性能 かつ省電力を実現

- 従来のプロセサ設計手法だけでは実現困難な目標

＜従来手法＞

マイクロアーキテクチャの改善

高い動作周波数

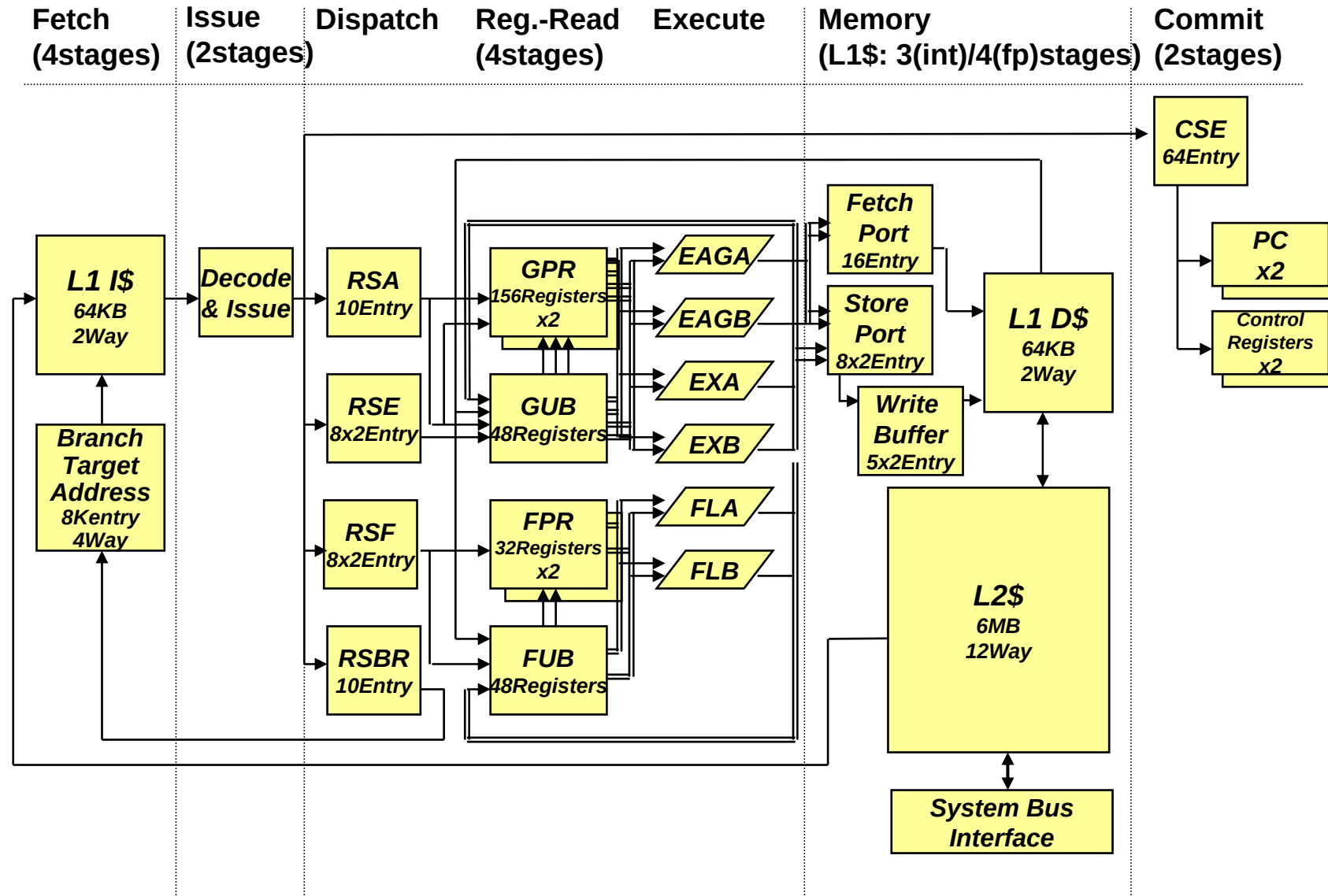
サイズの大きいTrによる高速化

→ 命令セットアーキテクチャ (Instruction Set Architecture)の大幅な
変革が必要

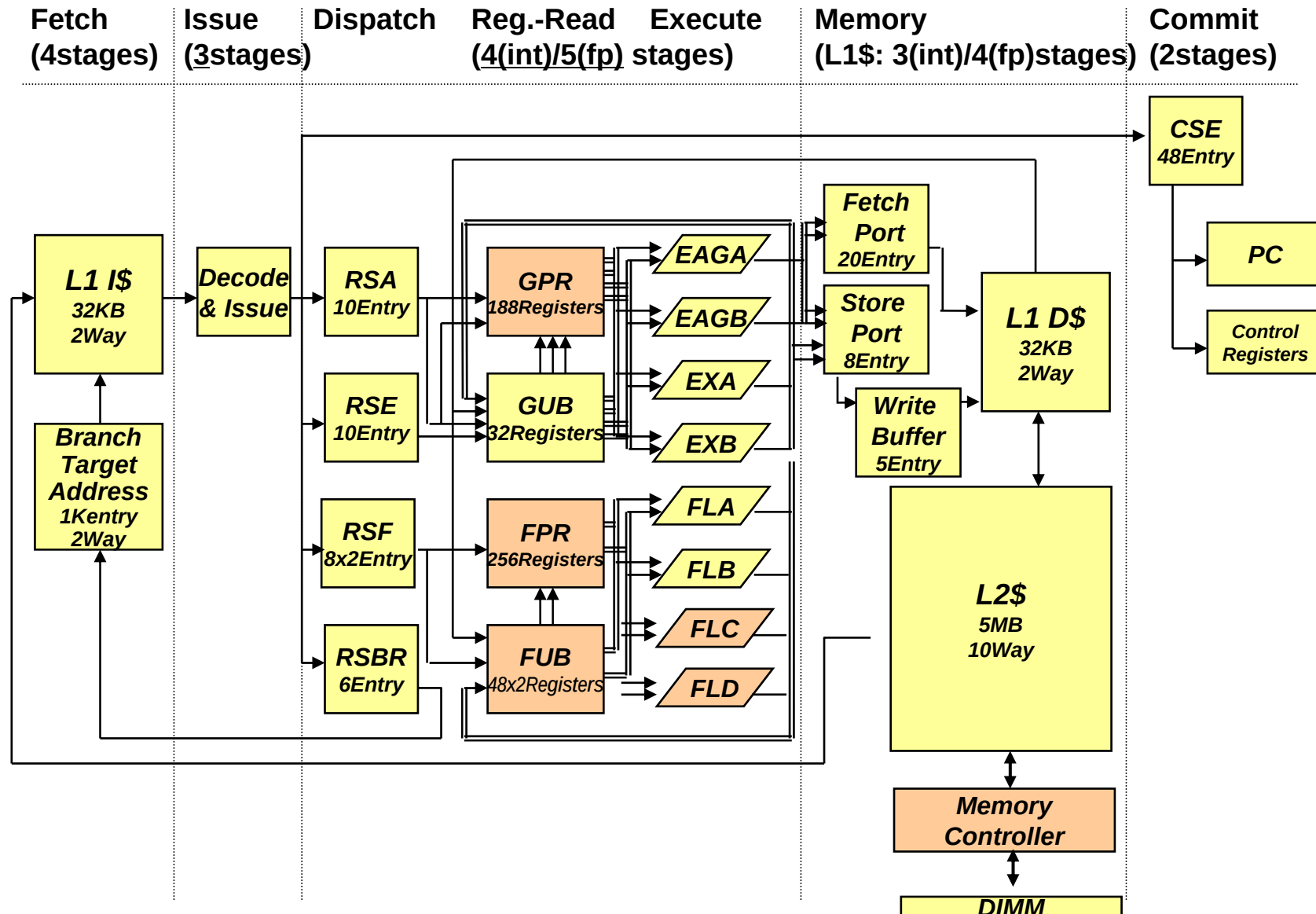
- 高密度実装: SoC (System on Chip)
- 高信頼性

◆ 適用可能な箇所では、SPARC64™ VIIのデザインを再利用

SPARC64™ VII パイプライン



SPARC64™ VIIIfx パイプライン



1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

◆ SPARC準拠仕様

- SPARC-V9 仕様
- JPS (Joint Programmer's Specification): SPARC-V9拡張仕様

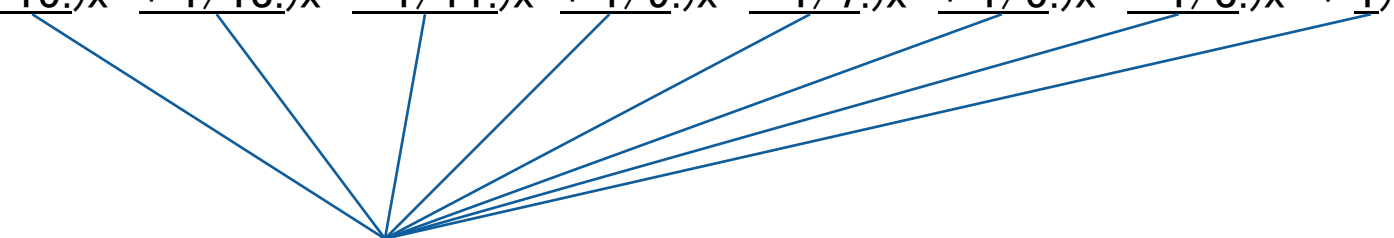
◆ HPC-ACE: 富士通独自のHPC向け命令セット拡張

“High Performance Computing – Arithmetic Computational Extensions”

- 新規命令 —HPC分野にて有用な命令を新規に追加
 - 三角関数 (ftrimadd)
 - 条件付き命令実行 (move/store)
 - 条件設定浮動小数点比較
 - 逆数近似 (frcpa, frsqr)
 - 大小選択 (fmax, fmin)
 - 前置命令 (sxar)
- レジスタ拡張
- SIMD命令
- セクターキャッシュ

◆ 三角関数を高速に求める命令

■ sin(x)をテーラー展開近似

$$\begin{aligned} \sin(x) &\approx x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \frac{1}{9!}x^9 - \frac{1}{11!}x^{11} + \frac{1}{13!}x^{13} - \frac{1}{15!}x^{15} \\ &= x \left(\left(\left(\left(\left(\left(\left(\left(0 - \frac{1}{15!} \right) x^2 + \frac{1}{13!} \right) x^2 - \frac{1}{11!} \right) x^2 + \frac{1}{9!} \right) x^2 - \frac{1}{7!} \right) x^2 + \frac{1}{5!} \right) x^2 - \frac{1}{3!} \right) x^2 + 1 \right) \end{aligned}$$


■ ftrimaddd

■ 処理内容: $rs1 \times \text{abs}(rs2) + T[\text{index}] \rightarrow rd$

■ 使い方:

```
# 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
ftrimaddd S, X2, 0, S  # S * X2 + 1/1! → S
fmuld     S, X, S      # S * X → S
```

◆ 条件付き命令実行

- ‘if’ を含むループの実行を効率化
 - 以下の条件付き命令を使うことで、条件分岐命令を削除することが可能
 - FPLレジスタを比較して結果をFPLレジスタに書き込む
 - FPLレジスタの値に基づいて、選択的にFPLレジスタ間でデータ転送
 - FPLレジスタの値に基づいて、選択的にFPLレジスタの値をメモリストア
- ソフトウェアパイプラインによる最適化が可能

◆ 除算/平方根の逆数近似

- 丸め誤差 $<1/256$ で逆数近似を計算
- パイプライン動作により除算/平方根を高速に計算

◆ 浮動小数点数の大小選択

- レジスタの値の大小を選択してレジスタに代入

＜逆数近似命令使用例＞

%f2 / %f10 → %f0

frcpad %f10, %f6

fmuld %f2, %f6, %f2

fnmsubd %f6, %f10, 1.0, %f6

fmuld %f6, %f6, %f0

fmadd %f6, %f6, %f6, %f4

fmadd %f0, %f0, %f6, %f0

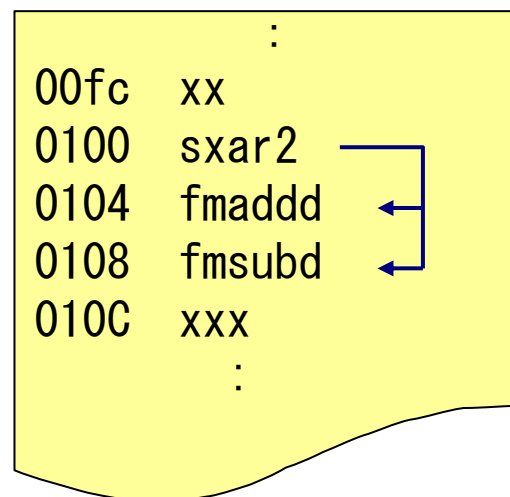
fmadd %f4, %f2, %f2, %f4

fmadd %f0, %f4, %f2, %f0

◆ 前置命令(SXAR)

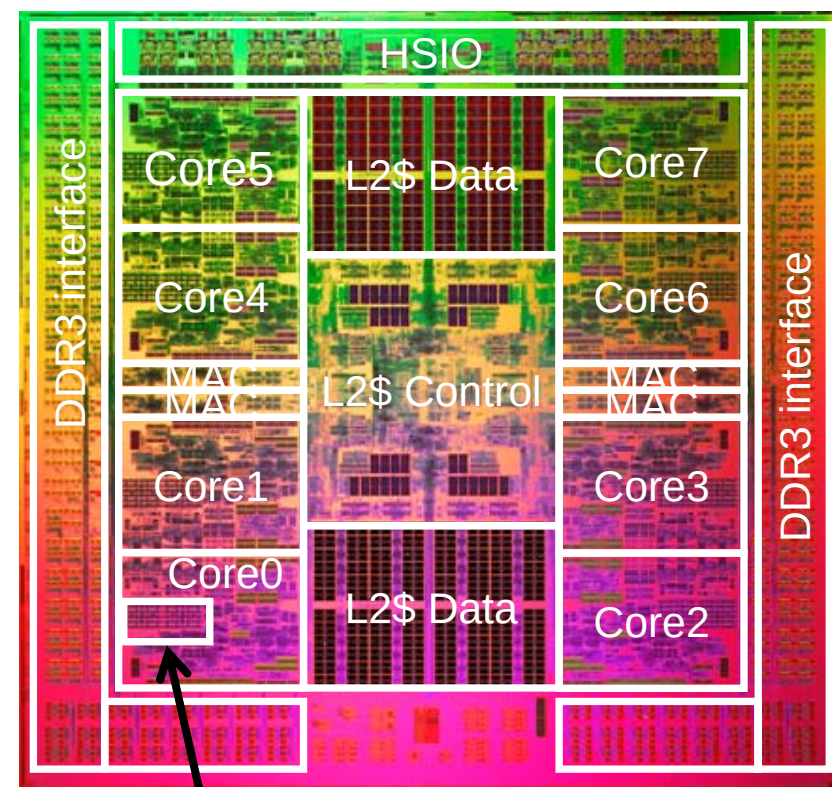
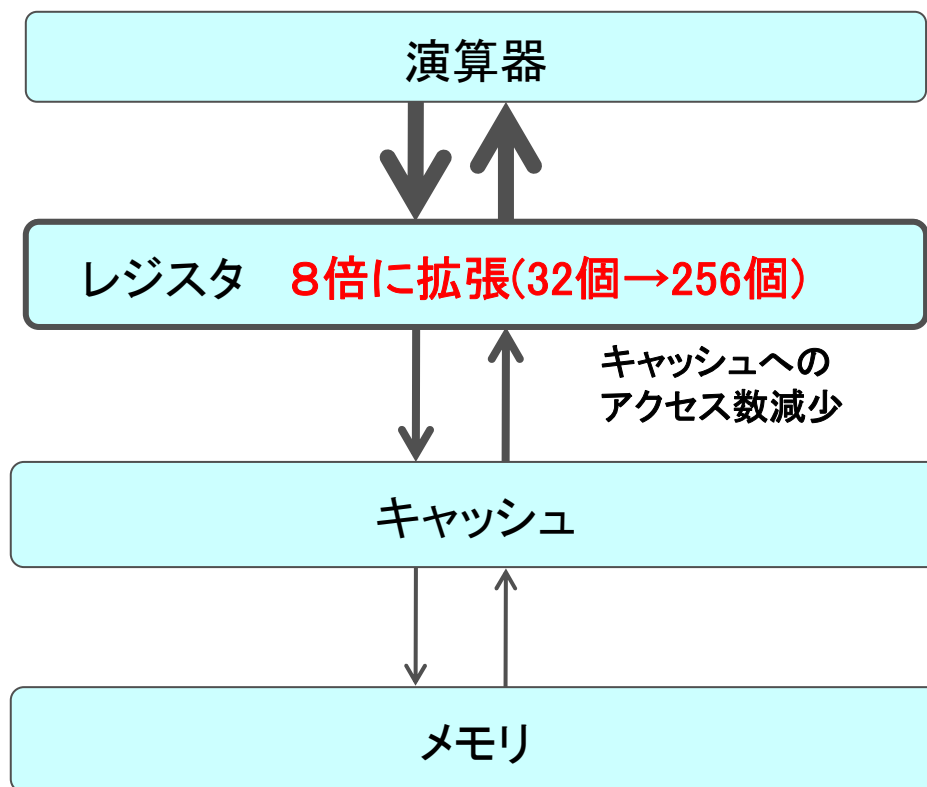
- XAR: 拡張演算レジスタ(Extend Arithmetic Register)へ後続命令の付加情報をセットする
- デコード時に本命令の付加情報と後続命令がパックされ、後続命令が実行される。

- SXAR1: 後続の1命令のXARをセット
- SXAR2: 後続の2命令のXARをセット
- 付加情報
 - 拡張レジスタアドレス
 - SIMD指定
 - セクタ指定



RISCアーキテクチャによる命令制限の壁
を打ち破る！

浮動小数点レジスタ拡張



SPARC64™ VIIIfx

拡張したレジスタ部分

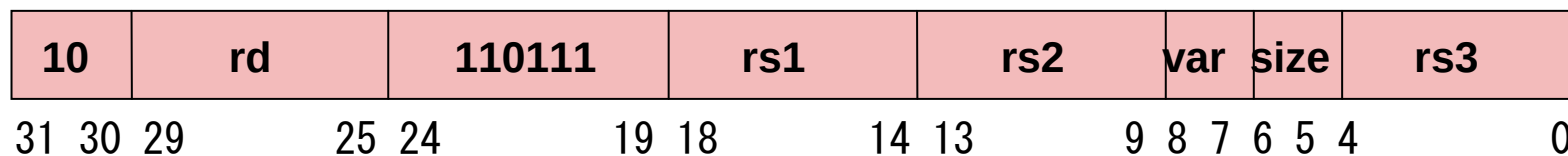
レジスタ数を大幅に拡張したことにより,
「計算処理の高速化」を実現

◆ RISCアーキテクチャ

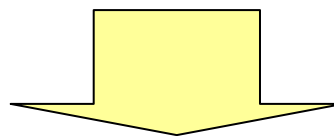
命令長: 32ビット

(例)

fmadd



オペコード + 4つのレジスタの指定



レジスタ指示ビットを増やせないため
レジスタ拡張は不可能！

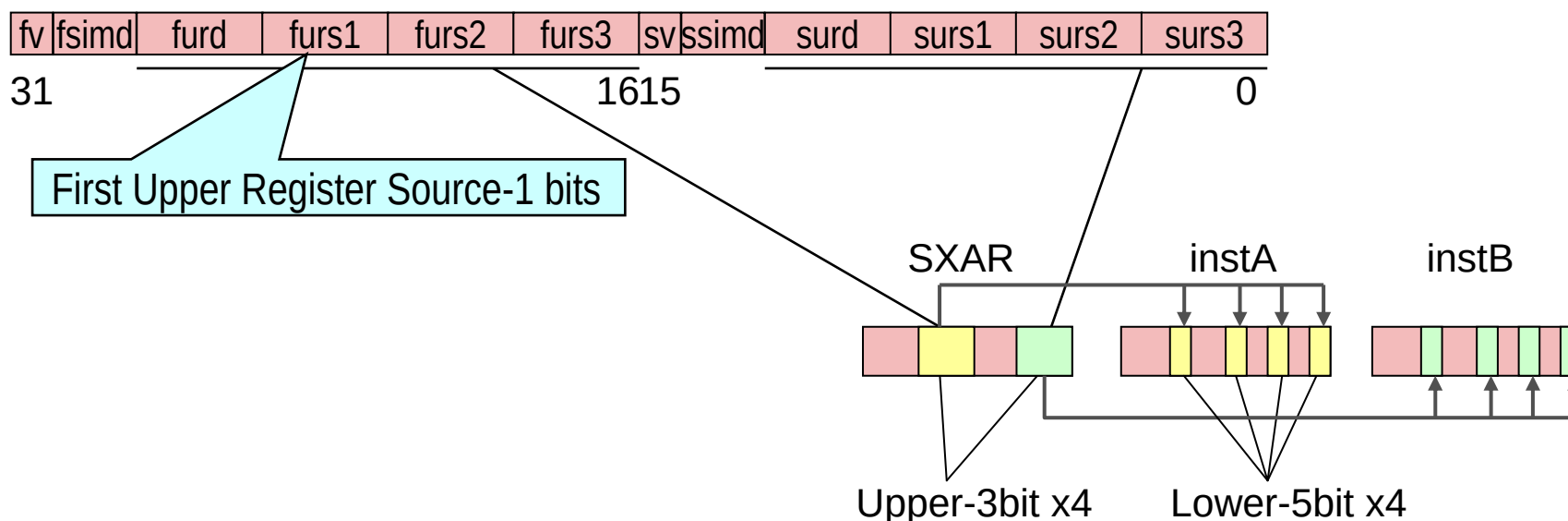
SXARによるレジスタ拡張指示

◆ XAR: レジスタ拡張

- SXAR命令により、後続命令のレジスタアドレス上位3ビットを指定

◆ 256個の浮動小数点レジスタのための命令フォーマット

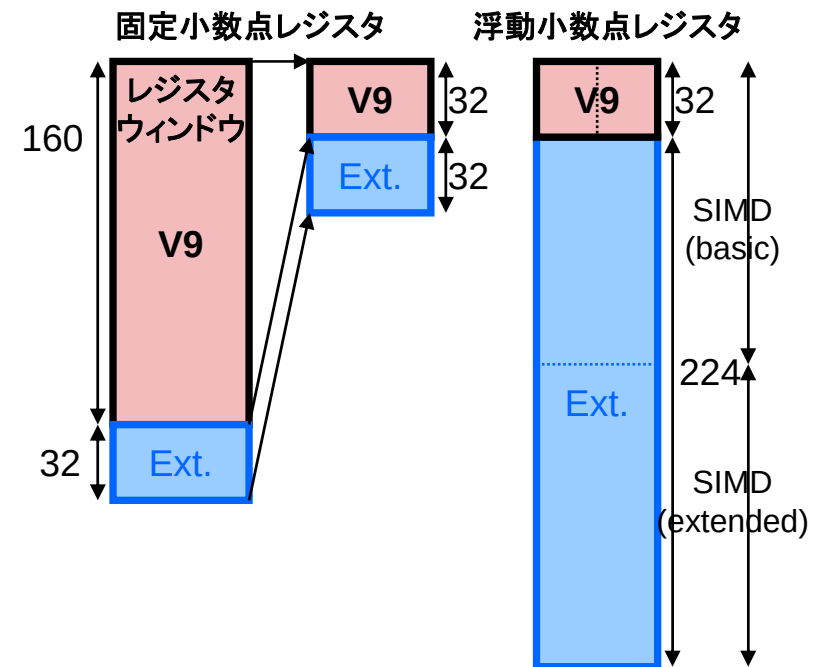
- 8 bit x 4 (3 read + 1 write) のレジスタ番号フィールドがFMA (Floating-point Multiply and Add) 命令に必要



拡張レジスタの構成

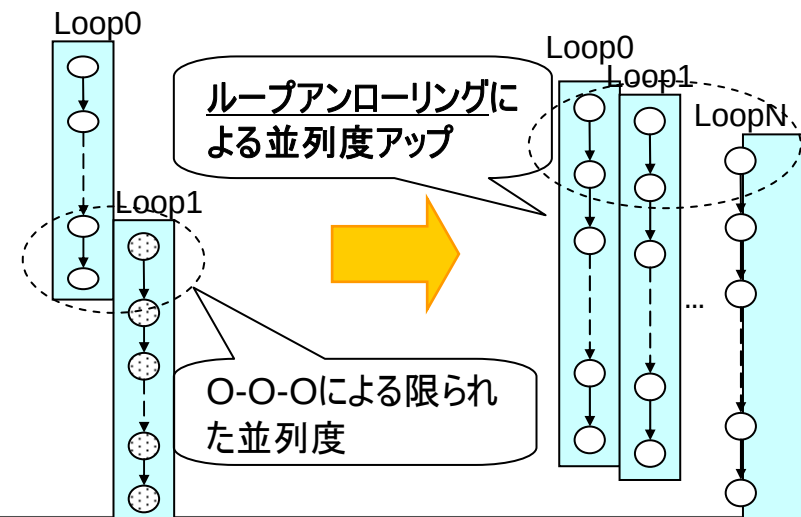
◆ SPARC-V9からレジスタを拡張

- 固定小数点レジスタ 32→64
- 倍精度浮動小数点レジスタ 32→256
 - 下位の32個のレジスタは SPARC-V9と共通
 - 浮動小数点レジスタはすべて同等。拡張浮動小数点レジスタは非SIMD命令からもアクセス可能。



◆ 拡張の効果

- レジスタ数による制約で制限されていた並列度の向上
- レジスタのspill/fillによるオーバーヘッドを削減

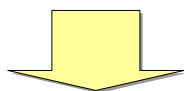


◆ SIMD (Single Instruction Multiple Data)

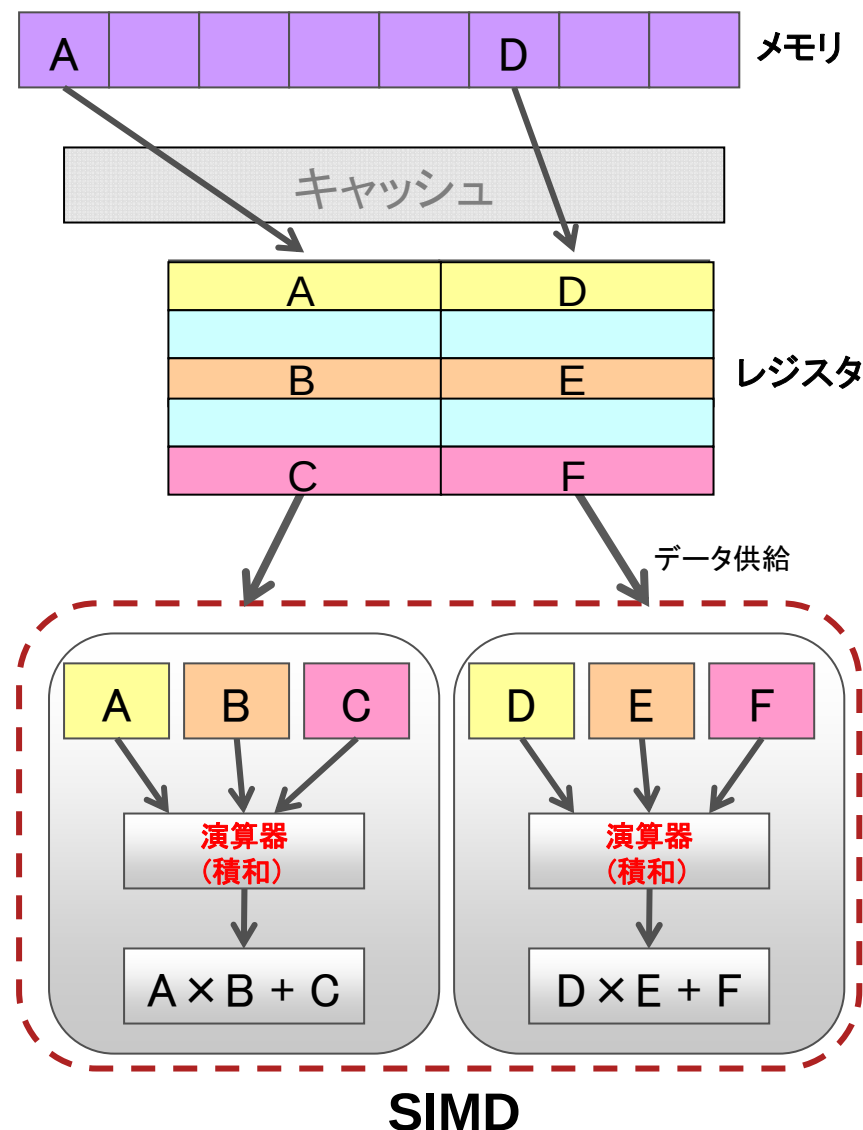
- 1つの命令で、複数のデータを並列処理すること

◆ SPARC64 VIIIfx のSIMDの特徴

- 1命令で2つの演算を並列処理
- 積和演算をサポート
- 1つのコアで2つのSIMD命令を同時に実行可能

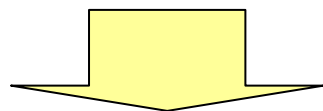


1つのコアで8個(1CPUで64個)の演算処理を同時に実行可能

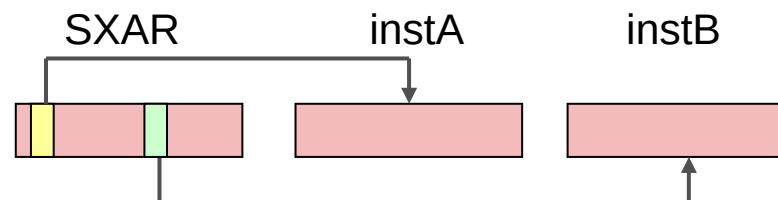
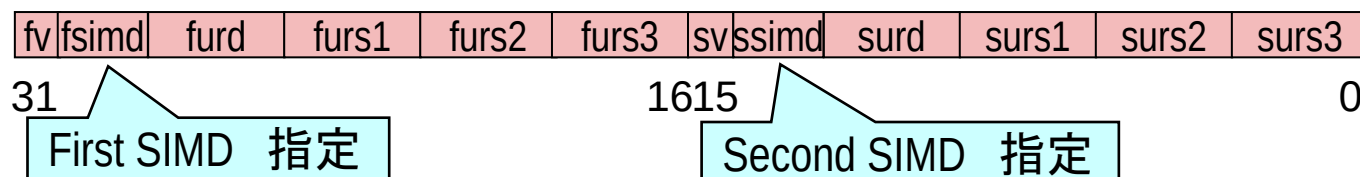


◆ XAR: SIMD拡張

- SXAR命令により、後続命令のSIMD実行を指示



SIMD実行のために新規オペコードを用意する必要がない



SIMDレジスタアクセス

◆ SIMD動作

- 1つのSIMD命令で2つの単精度あるいは倍精度の浮動小数点演算を実行
- レジスタ番号

extended Reg# = basic Reg# + 128

```
fmadd %f20, %f32, %f84, %f120
```

f20 x f32 + f84 → f120

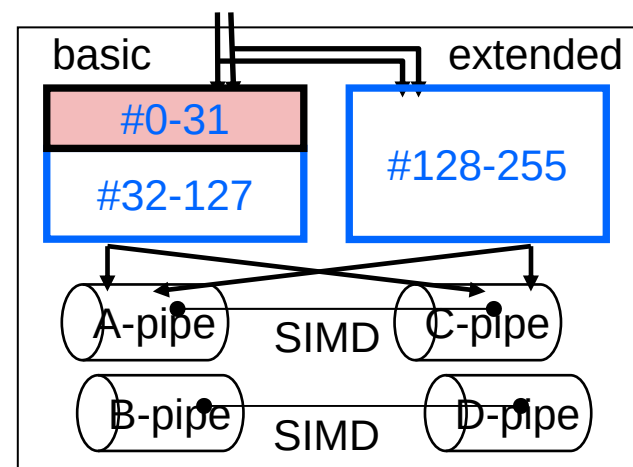
f148 x f160 + f212 → f248

◆ SIMD ロード/ストア

- 1つのSIMD ロード/ストア命令でメモリ上の連続する2つの単精度あるいは倍精度浮動小数点データにアクセスする
- 倍精度のSIMDロードについては、通常の16バイト境界ではなく、8バイト境界で実行
- 2つの独立した浮動小数点動作を拡張浮動小数点レジスタ指定と一緒にまとめることが可能

→ SIMD命令に関してソフトウェアによる最適化が可能

16byte Load x2



通常の SIMD 命令の使用例

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

2つのFP動作を SIMDで実行

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

- ◆ 性能を最適化するために、キャッシュコヒーレンスを維持したまま、ソフトウェアでキャッシュを制御

- ◆ キャッシュを2種類(セクタ)に分割使用

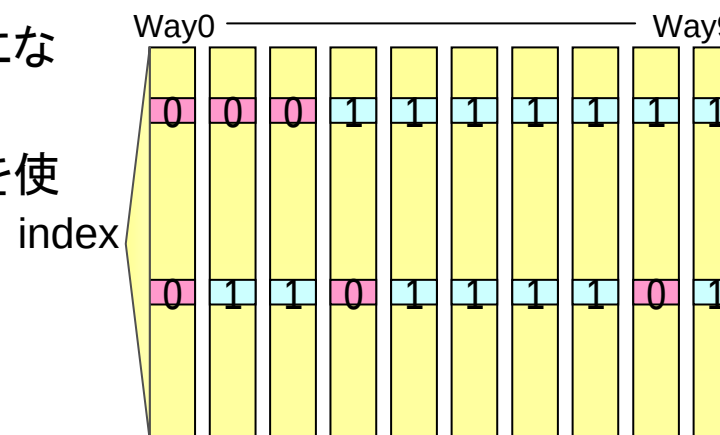
- SXAR 命令でセクタを指定
 - セクタ 0:命令フェッチ/通常のオペランドアクセス(デフォルト)
 - セクタ 1:明示的に指定されたオペランドアクセス
- セクタキャッシュ構成レジスタ
同一indexにおけるセクタ0/1の割合を指定

- ◆ ハードウェア実装

- キャッシュライン単位にセクタを管理
 - ラインリプレイス時、指定されたセクタ0/1割合になるようリプレイスウェイを選択
- ソフトウェアが、データの再利用性に応じてセクタを使い分けることが可能

L2キャッシュ構成

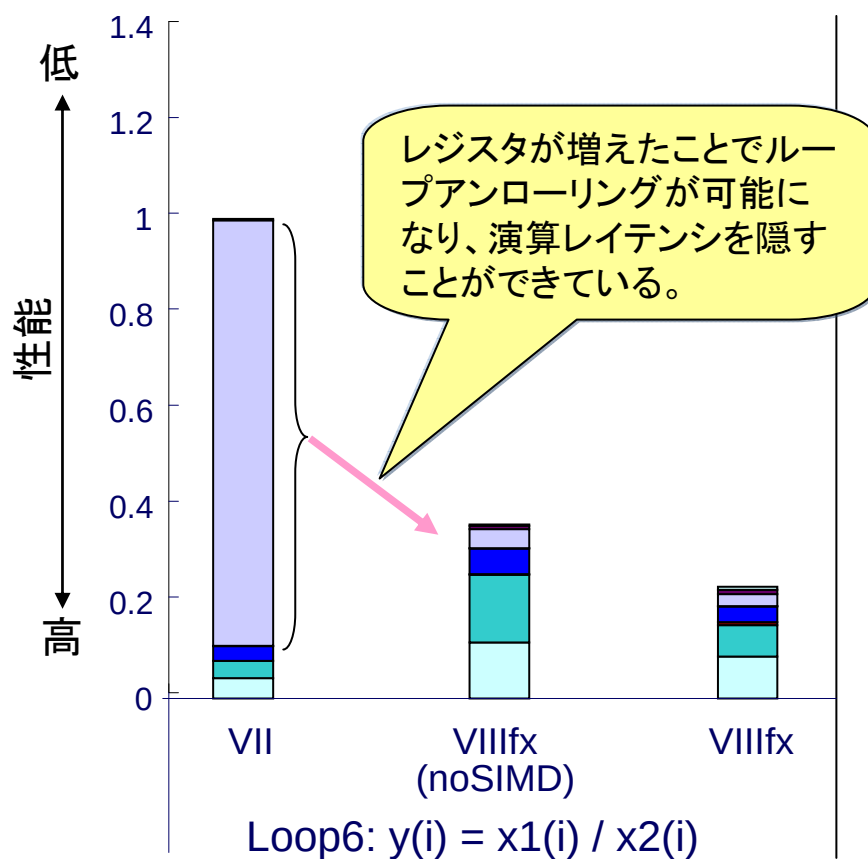
セクタ0/1の割合=3:7の例



レジスタ拡張の効果

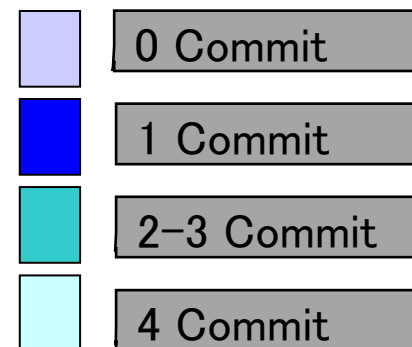
実行時間
SPARC64™ VII比

ベンチマーク: Euroben + 社内ツールで実測



SPARC64™ VIIIfx@2.0GHz

SPARC64™ VII@2.5GHz

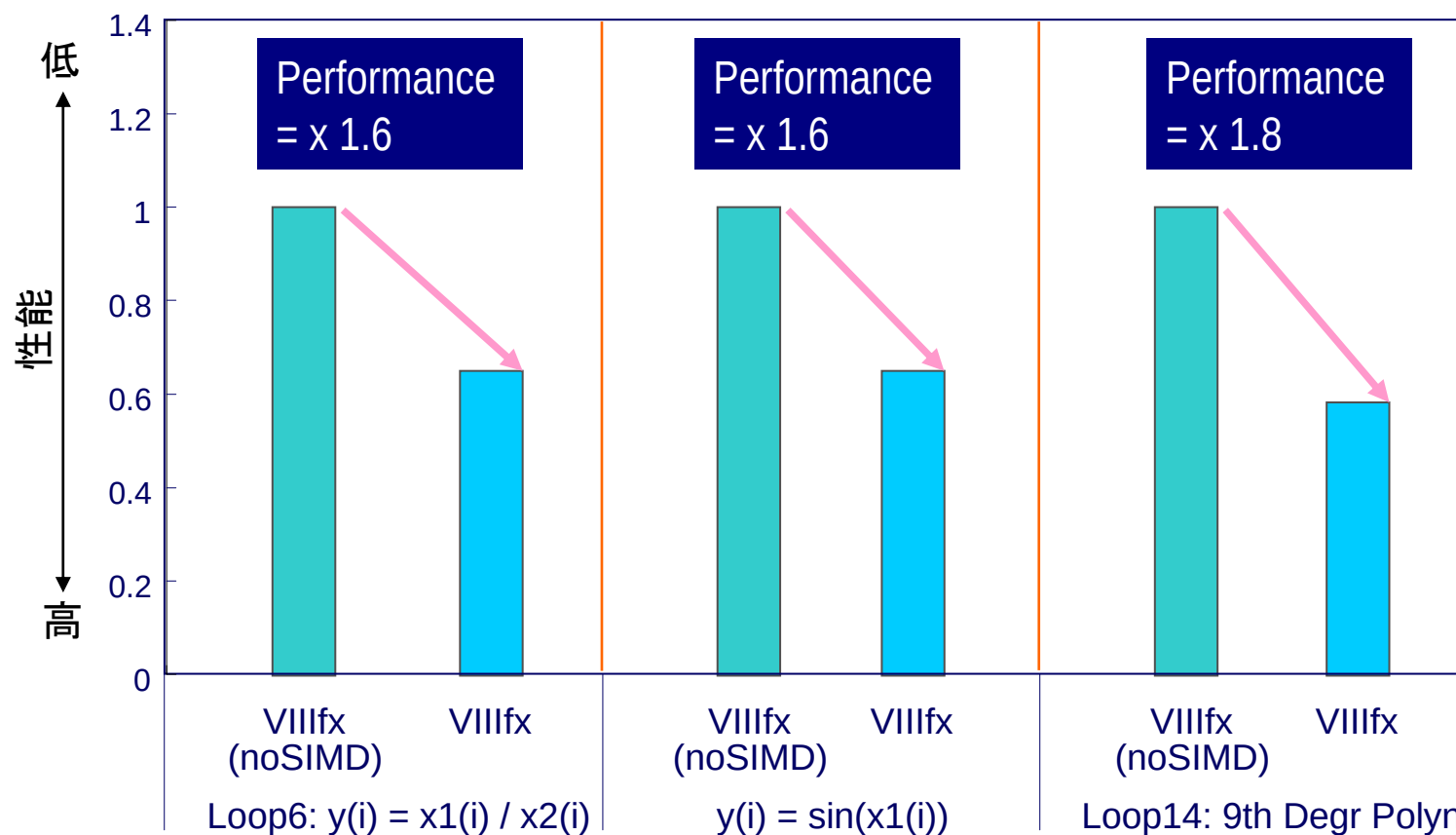


SIMDの効果

SPARC64™ VIIIfx@2.0GHz

実行時間
SIMD無し比

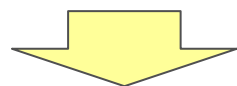
ベンチマーク: Euroben + 社内ツールで実測



1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

エラー検出・修復機能

宇宙線の衝突などにより
データが変化 (ex. 0→1)



間違った計算結果を出力

ハードによるエラー検出, 自己修復が可能

演算器	パリティ レジジュ	エラー検出 領域
レジスタ	パリティ ECC	
キャッシュ	ECC	

SPARC64™ VIIIfx



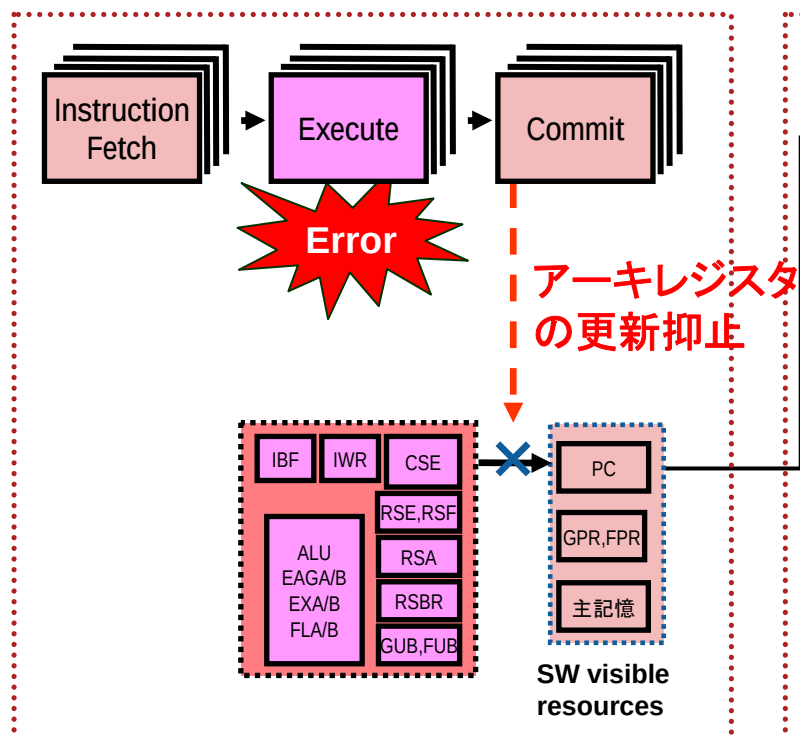
- ハードによるエラー検出
+ 自己修復可能領域
- ハードによるエラー検出
領域
- エラーが実運用に影響
しない領域

SPARC64™は, 広範囲なエラー対応を実現

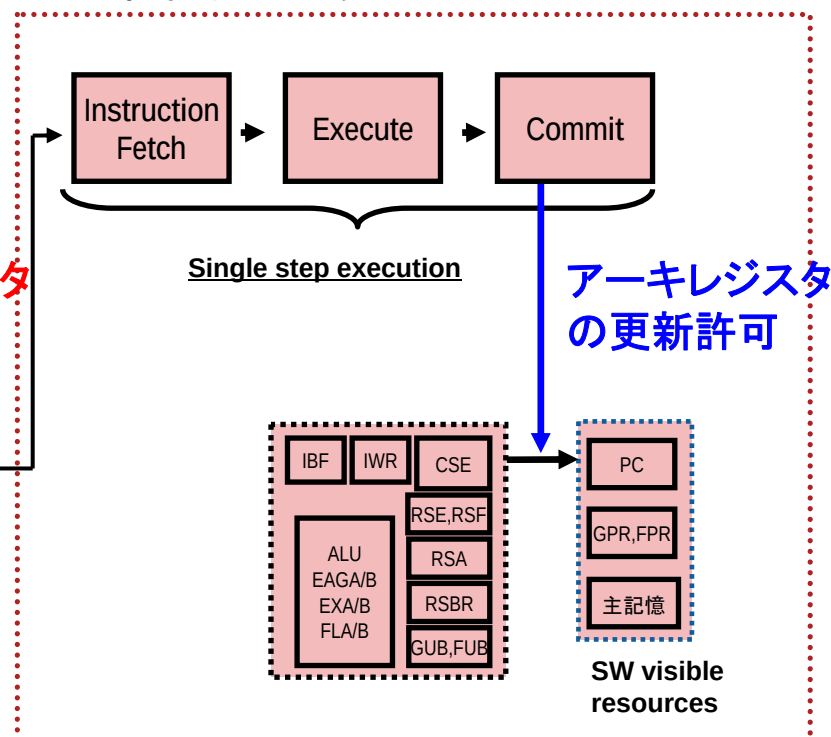
命令リトライ機能

- ◆ 間欠故障に対してはハードウェア命令リトライで救済する
(ソフトから見えずOS, アプリケーションによるサポートは不要)

①通常実行中にエラー発生



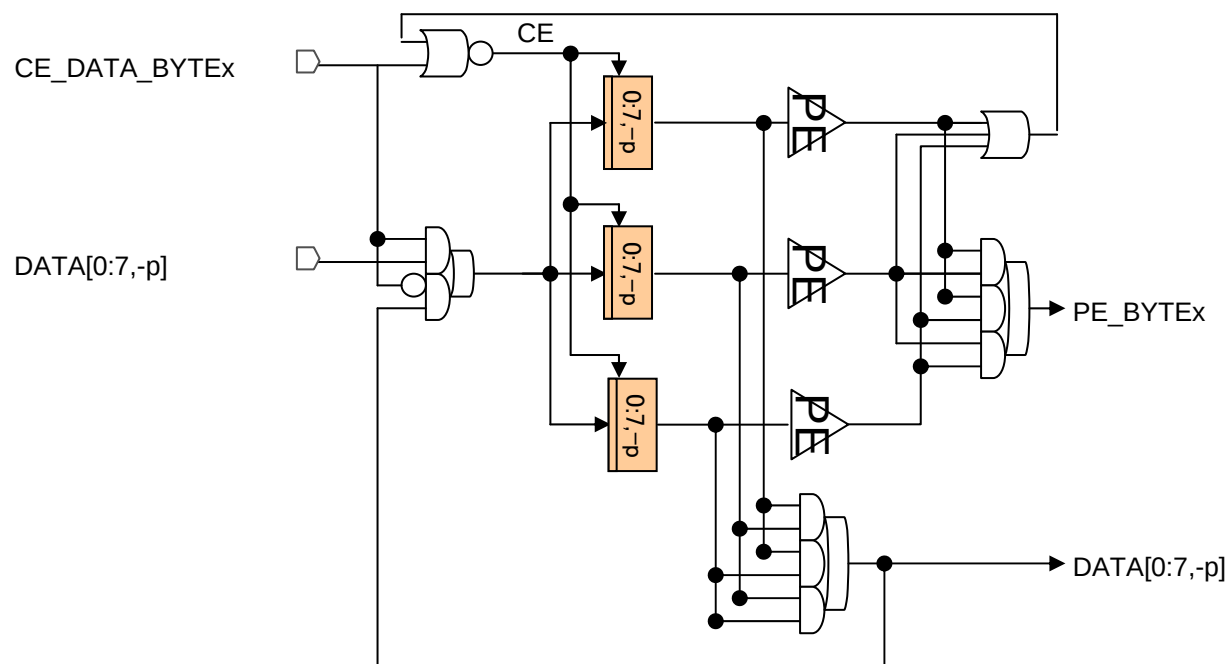
②更新前のPCが示す命令から
1命令ずつ再実行



③再実行した命令がエラーの発生なく
完了した場合は、通常の実行に戻る

ソフトウェア対策(レジスタの保護)

- ◆ 設定信号など特に重要なレジスタに対してソフトウェア対策を実施
 - 多重化回路の採用
 - ソフトエラー対策ラッチの採用



3重化多数決回路(TMR: Triple Module Redundancy)

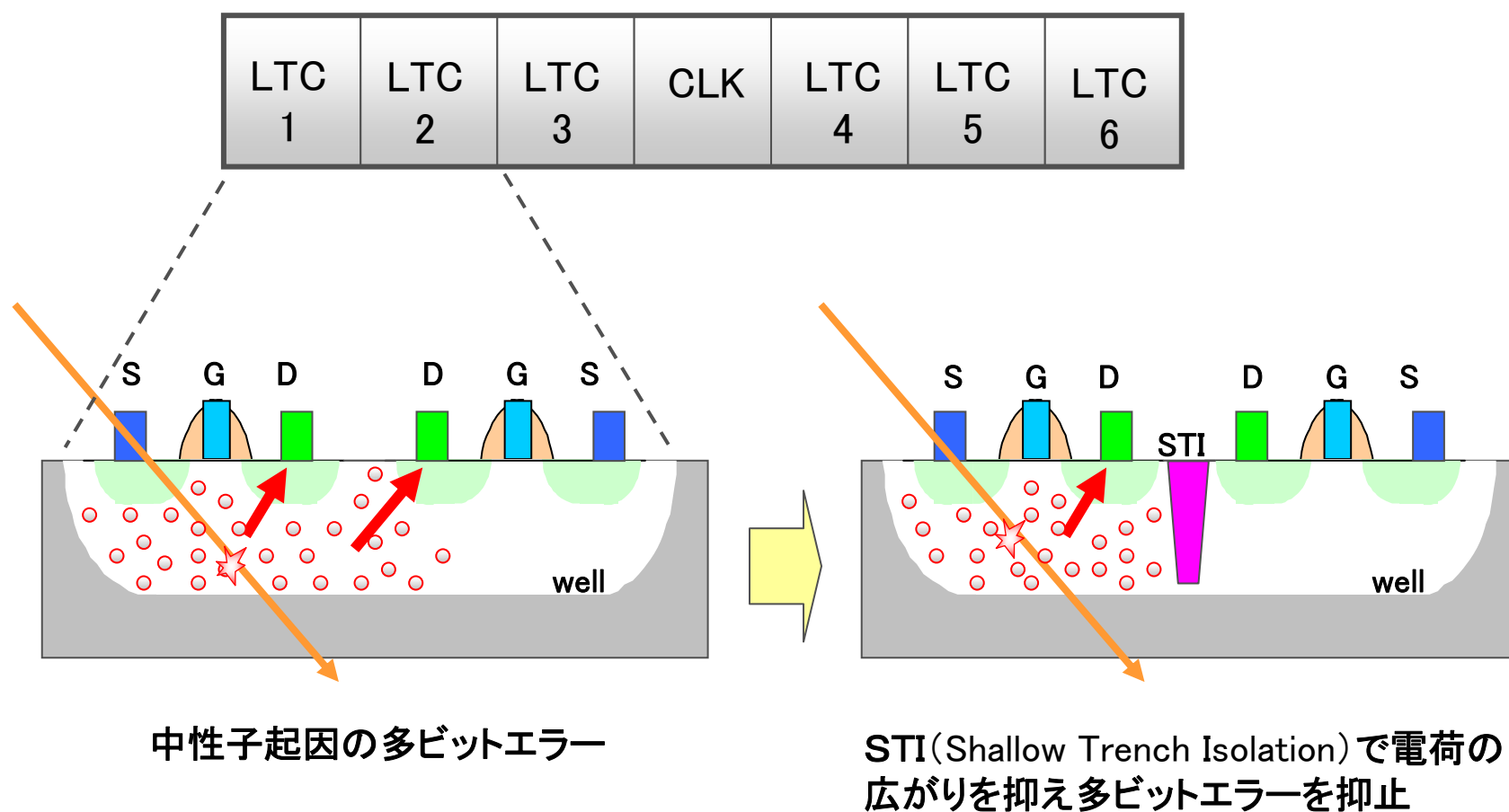
三重多数決で出力を決める。

ERR検出時に、多数決結果をラッチに書きもどす

ソフトウェア対策(ラッチレイアウト)

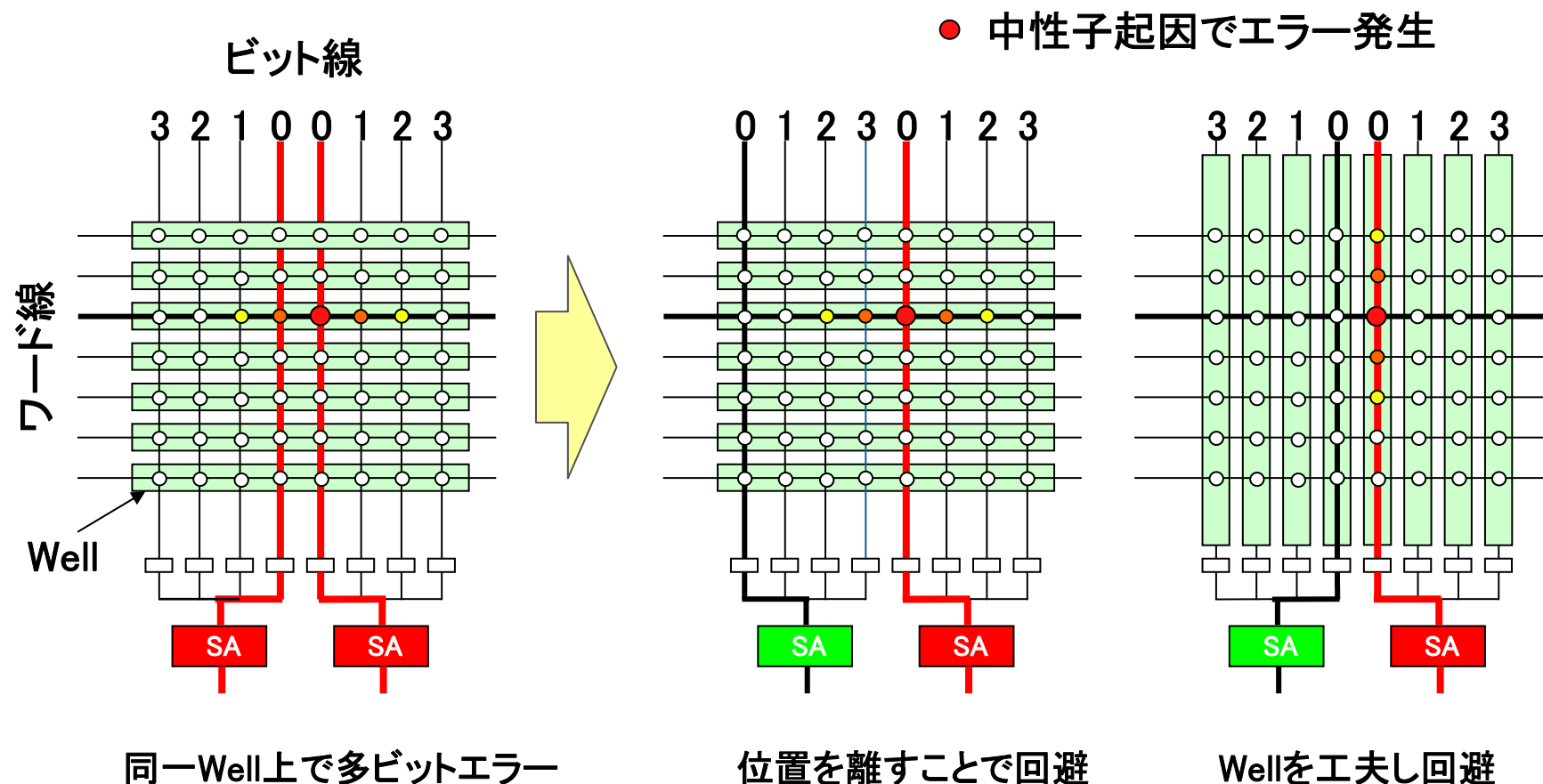
- ◆ 中性子起因のソフトウェアではエネルギーが大きいため、多ビットラッチの複数ビットが同時に反転する可能性がある

⇒ 多ビットエラーを意識したレイアウト設計



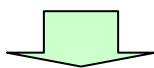
ソフトウェア対策(RAMのレイアウト)

多ビットエラー対策として、同時にアクセスするセルを近傍に配置しない
レイアウト設計

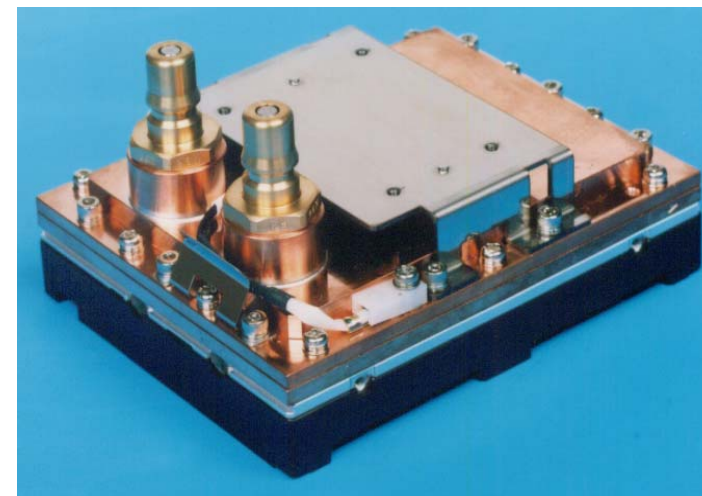


◆ 半導体の特性

- ジャンクション(接合部分)の温度を下げると部品寿命が向上



温度を10度下げれば寿命は約2倍向上



コールドプレート

◆ システム実装・冷却技術

- 水冷方式を採用
- 発熱体近傍まで冷水を敷設して部品と冷媒の温度差を高く維持し、熱変換効率を向上
- 高発熱部品はコールドプレートを密着して冷却

水冷方式の採用により部品寿命が向上し、「故障率低減」を実現

1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

- ◆ 水冷方式の採用により, リーク電流を削減
 - リーク電流は温度に依存

- ◆ 設計プロセスを見直し, 消費電力削減を極限まで追求
 - テクノロジ
消費電力が少ないトランジスタを採用
 - アーキテクチャ
「レジスタ拡張」, 「セクターキャッシュ」により無駄なアクセスを削減
 - 回路設計の工夫
 - 無駄な回路動作を抑制
 - 適切なサイズのマクロを使用
 - レジスタファイルの構造

従来のCPUに比べて消費電力を大きく削減

◆ 無駄な回路動作を抑制

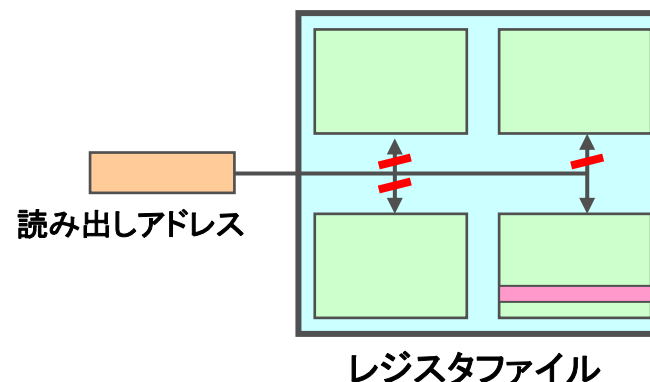
- 不要時におけるクロックバッファ、ラッチのクロック抑止
- 値が変化した時にのみラッチにクロックを印加し値を取り込む
- 未使用タイミングにおけるバスの値変化の抑止

◆ 適切なサイズのマクロを使用

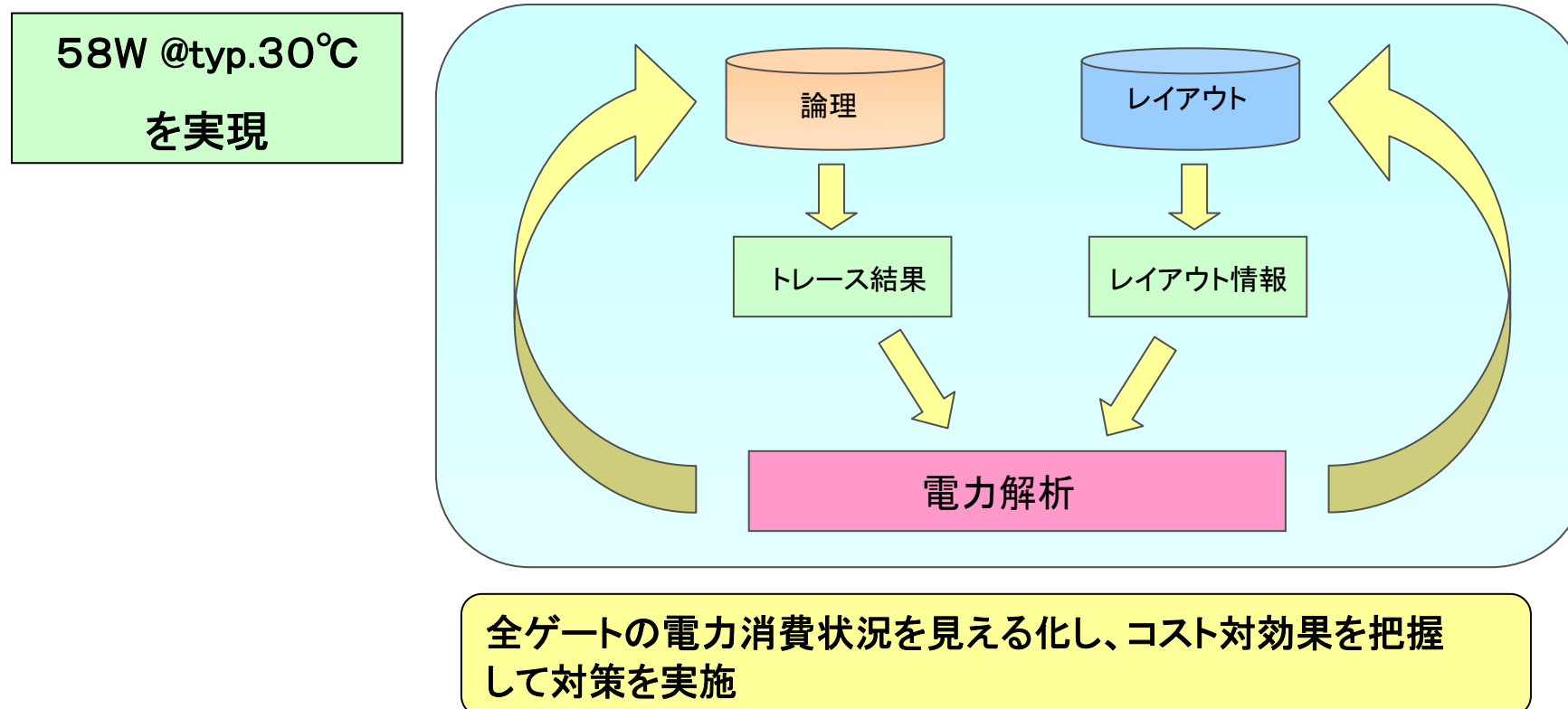
- タイミングの余裕値を把握し、余裕のあるパスについては小さいサイズのマクロを使用

◆ レジスタファイルの構造

- 読み出すエントリを含むブロック以外のアドレスを固定し、読み出しデータ遷移を抑制。



- ◆ 論理ブロック単位での消費電力状況を算出
- ◆ 論理ブロック単位での消費電力バジェットを設定し、それに向け各メンバが競って対策を実施
- ◆ 『論理変更+レイアウト変更 → 電力解析』のループをひたすら回し電力削減を実施

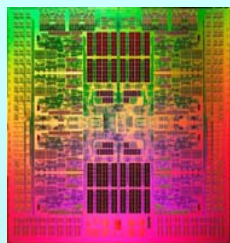


1. 富士通のプロセッサ開発の歴史
2. SPARC64™ VIIIfxの概要
3. 新アーキテクチャ HPC-ACE
4. 高信頼性技術
5. 省電力技術
6. まとめ

SPARC64™ VIIIfx まとめ



- ◆ 富士通の次期スーパーコンピュータ、文部科学省殿が進める次世代スパコンに搭載されるプロセサSPARC64™ VIIIfxを開発
- ◆ SPARC-V9 アーキテクチャの制限を越えるためのHPC-ACE 命令セットを採用
- ◆ 高性能と低消費電力を同時に実現
- ◆ 現在チップは実験室でシステム検証中



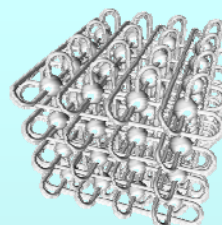
CPU



System Board



Rack



3次元トラスネットワーク
(概念図)



次世代スパコンシステム



