FUJITSU
shaping tomorrow with you

CREST「ディペンダブルVLSIシステムの基盤技術」研究領域　第1回領域会議

パネル：　設計検証、テスト

2012年6月8日

富士通株式会社
高山浩一郎

「京」

- 2011年6月と11月にTOP500を連覇
  - LINPACK性能：10.51PFlops
  - 計算時間：　　約29時間
    ⇒　1ノード換算290年以上連続稼働

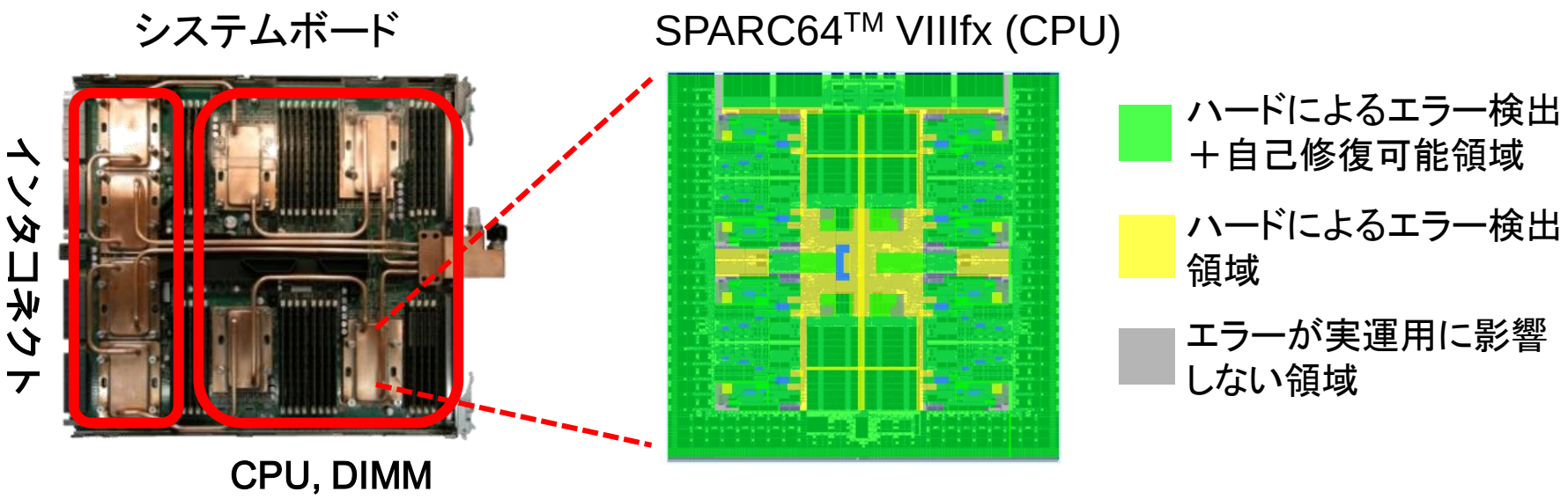
	System	Site	Performance [Pflops]	#Nodes	Time [hour]	Node-years
1	**K computer**	**RIKEN AICS**	**10.51**	**88,128**	**29.5**	**296.8**
2	Tianhe-1A	NSC in Tianjin	2.57	14,336	3.4	5.6
3	Jaguar	Oak Ridge N. L.	1.76	18,688	17.3	36.9
4	Nebulae	NSC in Shenzhen	1.27	9,280	1.9	2.0
5	Tsubame-2.0	Tokyo Inst. of Tech.	1.19	2,816	2.4	0.8

- 数万ノード規模のシステムを安定稼働させるために
  - 個々の部品の高い信頼性、可用性、保守性を実現

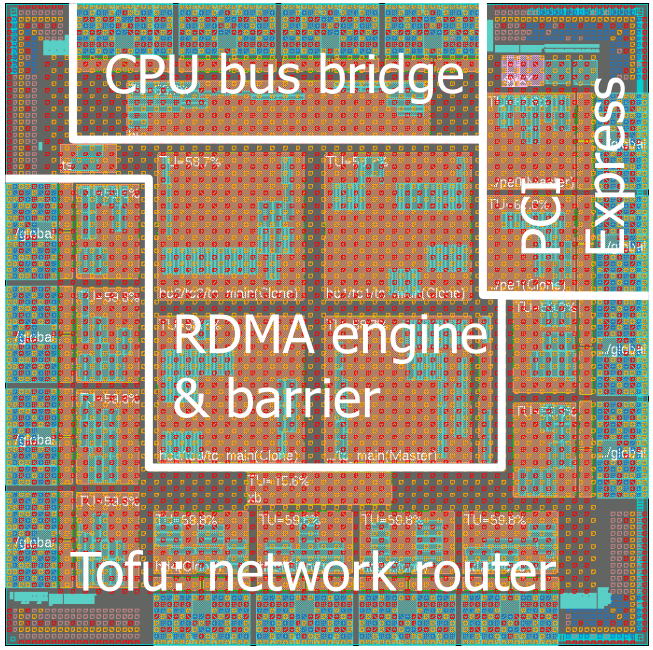
高信頼なVLSIの開発

FUJITSU

- VLSI (CPU, interconnect) の高信頼化
  - メインフレームの時代からのRAS技術の蓄積
  - 壊れない： 水冷、α線によるソフトエラー対策ラッチ
  - 壊れても間違わない（検出、訂正）：
    論理多重化、ECCによるエラー訂正、リトライ機構
    - エラーの検出、訂正を論理的に検証



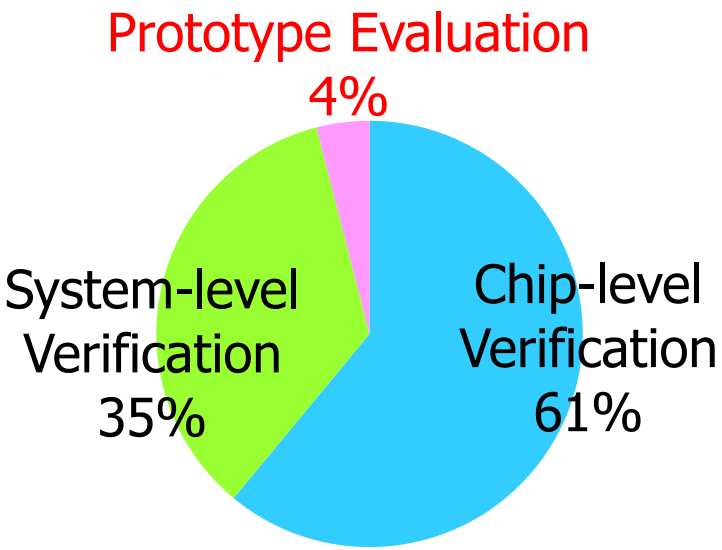
インターコネクトコントローラ (ICC)



- 基本機能
  - 4個のRDMAエンジン
  - 高機能バリアのハードウエアサポート
  - Tofu: 6次元メッシュ/トーラス
    - Virtual Cut-Through
  - PCI Expressルート機能
- デバイス
  - 65nm ASIC
  - 18.2mm x 18.1mm
  - 48Mゲート、12Mbitメモリ
  - クロック 312.5 MHz
- 高帯域
  - 1ポート当たり 10GB/s (双方向)

ICCの論理検証

- 仕様書を分析して、テスト項目を抽出
  - 正常テスト (エラーが無い状態)
  - エラー検出テスト (エラーを検出して処理が停止)
  - エラー訂正テスト (エラーを訂正して処理が完了)
- ICCは複数のトランザクションを並列処理
  - 複数のテスト項目の組合せ
- Verification step
  - Chip-level verification
    - ICC 1チップ/サブチップの検証
    - Simulation @ 10~100Hz
  - System-level verification
    - 3ノード (CPU + ICC) 構成の検証
    - Emulation @ 100k~1MHz
  - Prototype evaluation
    - 100ノード規模の試作機 (engineering model) による検証
    - 全体の4%の論理障害を検出

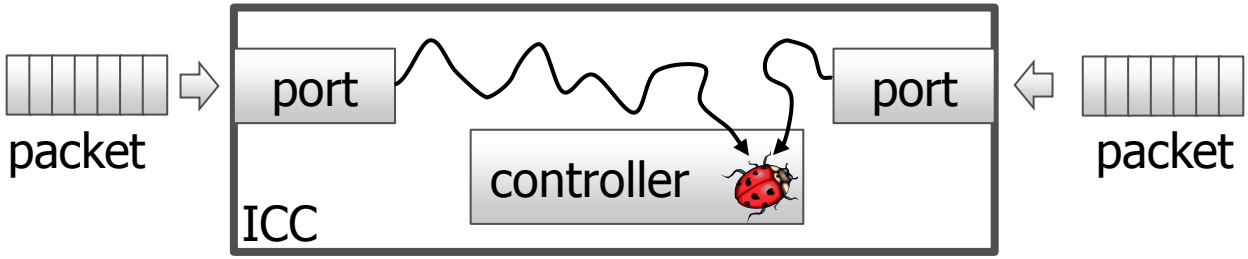


論理障害の割合

試作機で検出された論理障害の例

FUJITSU

- 複雑で稀なタイミング上のコーナーケース
  - エラー訂正テストにおいて、異なるポートに起きたイベントが、ある制御論理を誤動作させた。タイミング上、また、イベントの組合せ上稀なケースであり、設計者はそのような組み合わせを想定していなかった。
  - そのようなイベントとタイミングの組合せを仕様書から読み解くのは非常に困難であった。イベントやタイミングの単純な組み合わせではテスト項目数が指数的に増大する。
  - テストプログラム上で障害が発生するコードの特定に数日、シミュレータ上で再現可能なシーケンスを抽出し、障害箇所の特定に数週間を要した。
  - 形式的検証ツール (市販) は、設計規模が大きく(>1Mゲート)、反例シーケンスが長い(>100cycle)ことから障害を検出できなかった。



VLSIの論理検証の効率化に向けた課題

FUJITSU

- 検出困難な障害を早期に検出するために
- **何を検証すべきか**
  - 機能だけでなく性能や電力も検証対象
    - テスト項目抽出における機能やタイミングの組合せの絞り込み
    - 検証品質を判定するに足る機能カバレッジ指標の構築
- **どのように検証すべきか**
  - 形式的検証, プロトタイピング, ハード/ソフト協調検証
    - 最適な検証計画の立案： どのテスト項目をどの方法で検証するか
- **効果的なデバッグ**
  - Pre/post-silicon debug
    - 論理的/物理的に誤りやすい場所の判定
    - 障害箇所の特定, テストプログラム長の縮小

上記の項目の中には、既に研究報告されているものがある。
「京」のような大規模なシステムへの実用的な適用可能性を注意深く判断する必要がある。