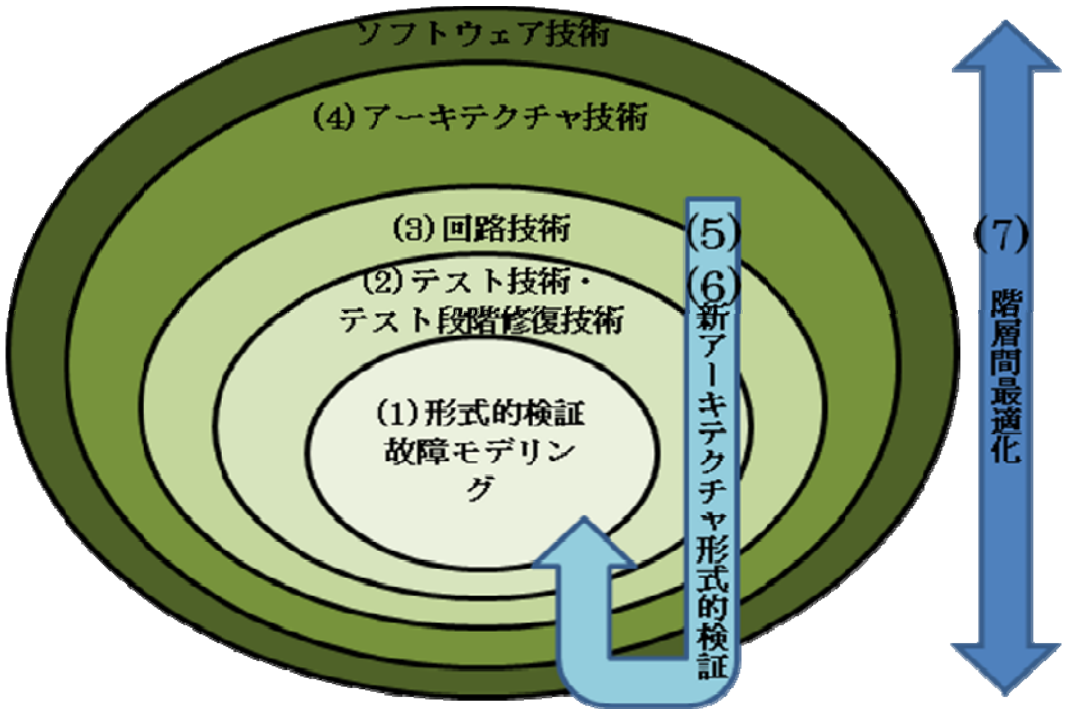
アーキテクチャと形式的検証の協調による超ディペンダブルVLSI

戦略的創造研究推進事業
「ディペンダブルVLSIシステムの基盤技術」

東京大学　大学院情報理工学系研究科　坂井　修一（代表者）
五島　正裕
東京大学　大規模集積システム設計教育研究センター（VDEC）　藤田　昌宏
東京工業大学　大学院情報理工学研究科　吉瀬　謙二
日本電気（株）　若林　一敏

全体マップ： ディペンダビリティ階層



それぞれの階層で技術開発
＋ 全体を通した最適化
＋ 最新アーキテクチャの検証

Best Effort Design
Run Time Recovery

前半3年： 方式検討、基本設計、実験システム構築・評価
後半2年： プロトタイプ試作と評価、要素技術の統合

2011/12/3

(1) 形式的検証手法

- 等価性検証ソフトウェア
- ハード・ソフト協調による検証高速化・エミュレータへの応用
- ボトムアップ・トップダウン協調検証
- 算術回路合成・検証・高速化
- 設計解析技術・デバッグ支援

(2) テスト技術・テスト段階修復技術

- テスト容易化・検証容易化を実現する設計手法
- プログラマブル素子自動挿入

(3) 回路技術

- タイミング制約緩和回路

(4) アーキテクチャ技術

- 故障検出・回復機構の提案・実現
- 耐永久故障FPGA
- 耐故障高機能ルータ
- 超ディペンダブルプロセッサ、超ディペンダブルメニコア

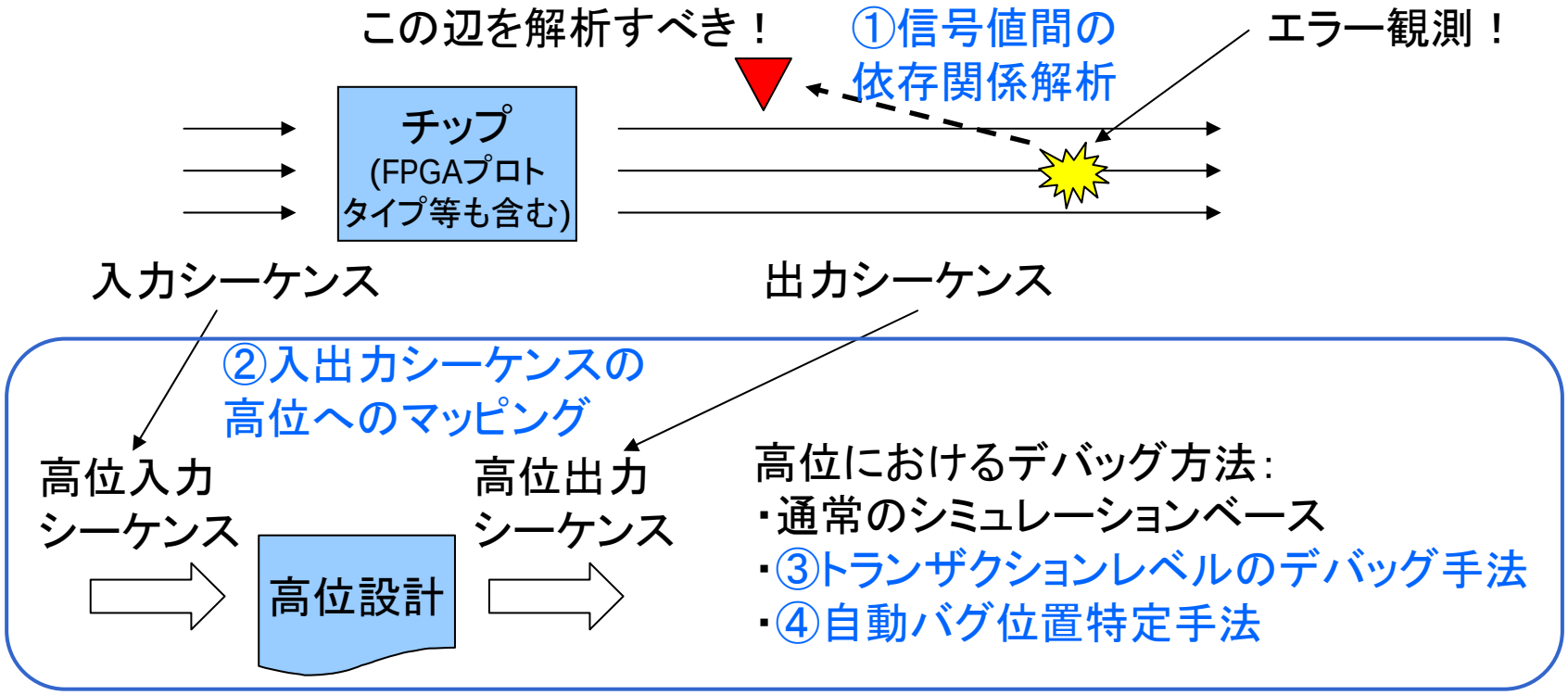
(5)(6) 新アーキテクチャ形式的検証

- ディペンダブルアーキテクチャ技術自体を形式的に検証
- 既存のアーキテクチャ、最新のアーキテクチャを形式的に検証

(7) 各設計階層間のディペンダビリティ役割分担を最適化

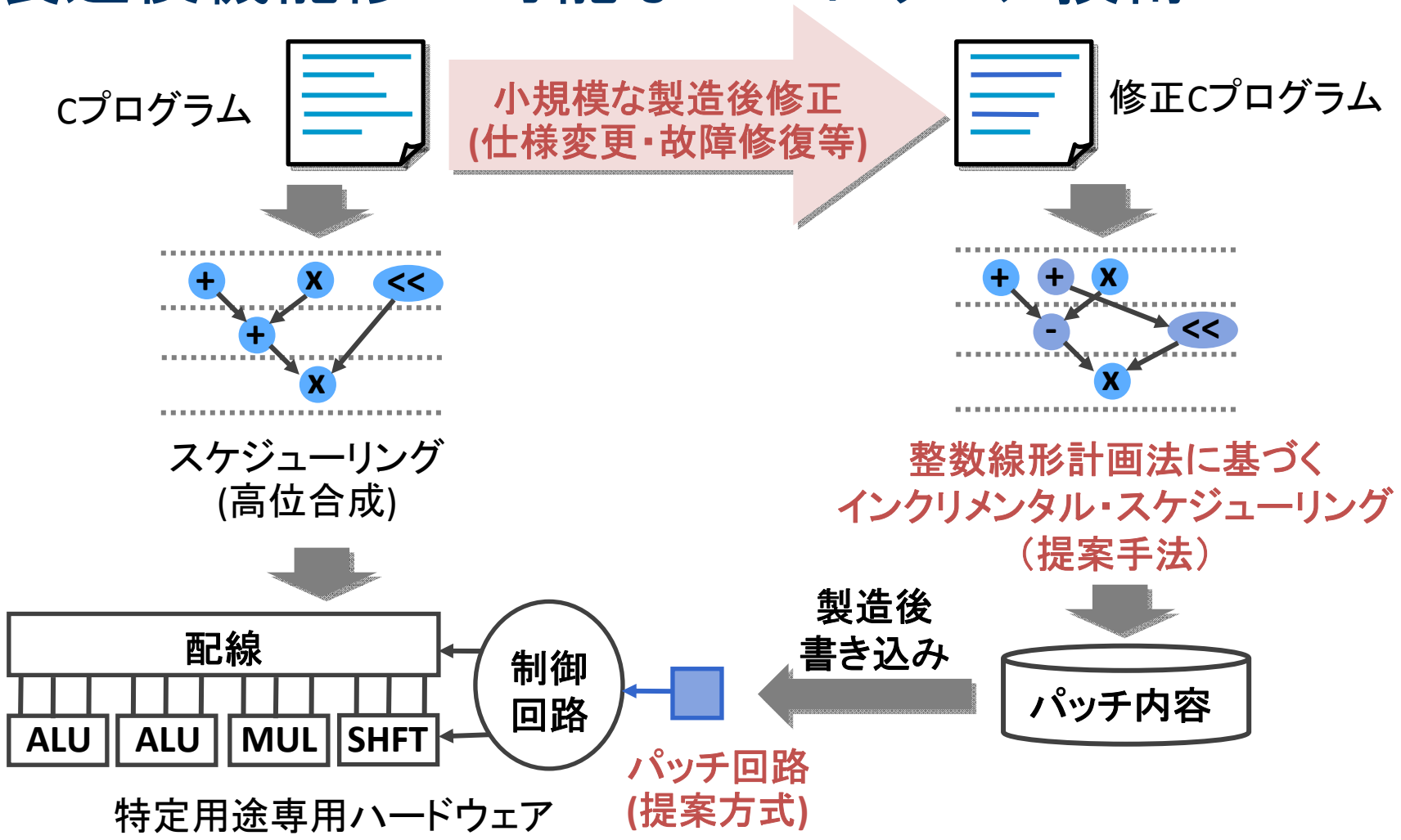
形式的検証（藤田、若林）

ポストシリコンデバッグ技術



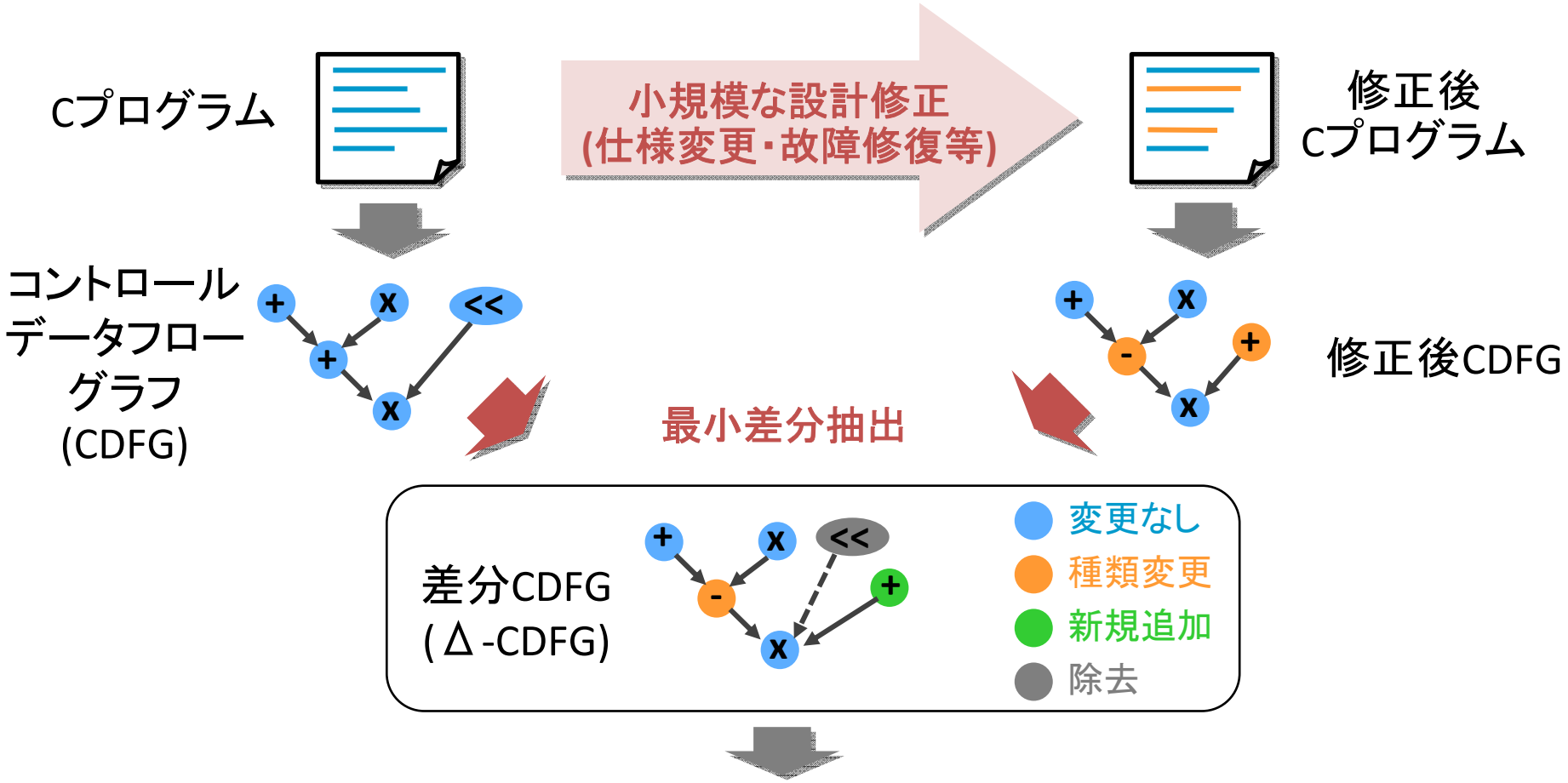
- ポストシリコンにおける「非常に長い反例シーケンス」「内部信号の低い可視性」を考慮したデバッグ支援手法
  - インタフェース部の信号線のみモニター、反例の解析部分を絞る、など
- 要素技術を組合せた全体枠組み構築に着手(ツール化含む)
- 企業ヒアリング(4社)を実施し、ニーズ・実用化における問題点を把握
  - 反例シーケンス上で～1000クロックサイクルに絞り込むまでが大変
  - トランザクションレベルでのバグ検出・デバッグ支援は期待大、など

Post-siliconデバッグ：
製造後機能修正可能なハードウェア技術



回路全体をプログラマブルにするのではなく、制御回路を部分的に
変更可能にすることで専用ハードウェア並みの効率・性能を実現

高位設計記述間の最小差分抽出



応用分野

- インクリメンタル高位合成（既設計の再利用）
- パッチ可能ハードウェア用パッチ生成（post-silicon修復）
- 高位設計記述の等価性検証（差異に基づく検証）

差分抽出問題へのアプローチ

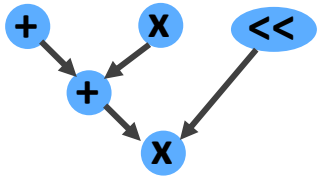
Cプログラム

プログラム間の文字列比較

名前変更などに弱い・
内部表現との対応が取れない

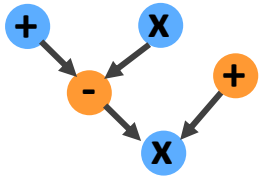
修正後
Cプログラム

コントロール
データフロー
グラフ
(CDFG)



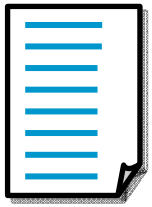
グラフ間の差分抽出問題

(共通部分グラフ問題)は
現実的に解決困難！



修正後
CDFG

中間命令列



提案アプローチ

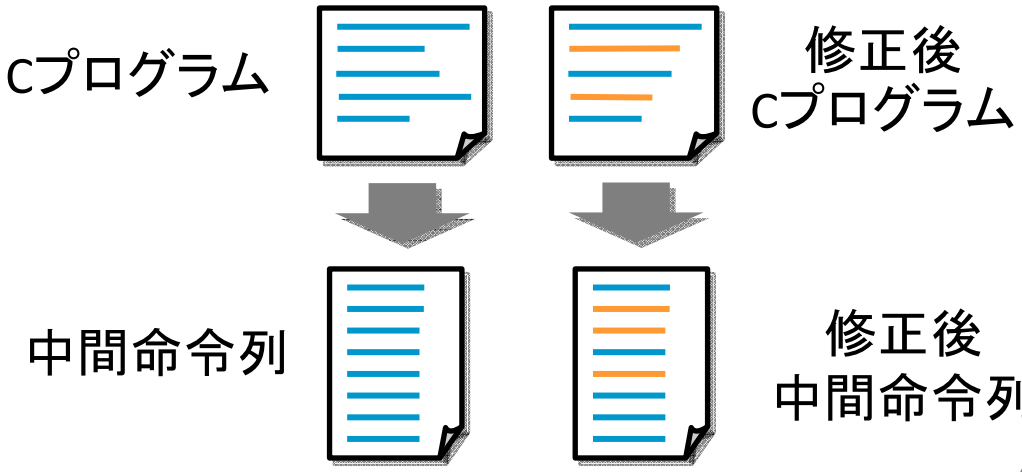
単純な命令の列に
変換してから差分を計算

修正後
中間命令列

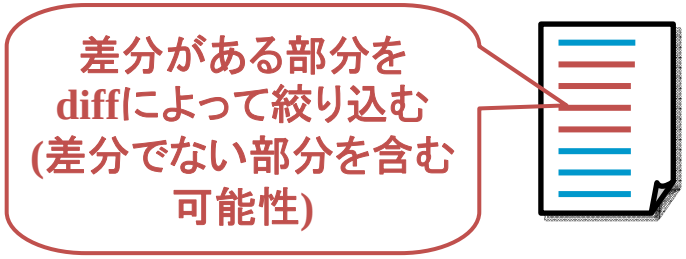
提案アプローチの利点

- 列同士の比較のため、グラフ間比較よりも容易
- 文字列表現にすることで効率の良い差分計算手法(diff)を利用可能
- 制御フローとデータフローを一貫して表現可能

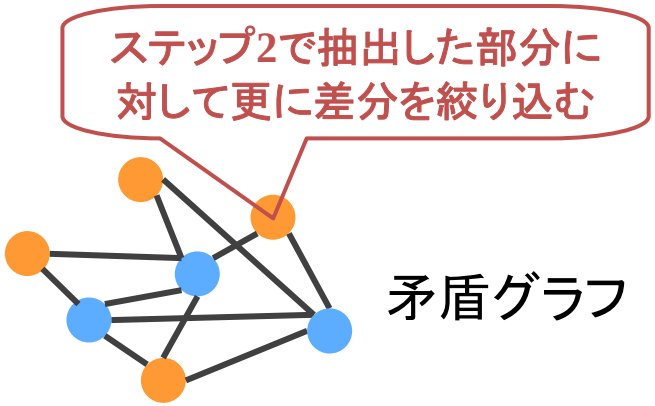
差分抽出の提案手法



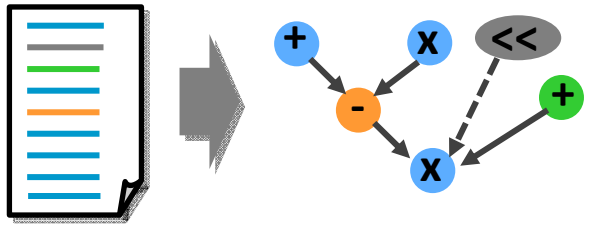
ステップ1: 中間命令列への変換



ステップ2:文字列比較(diff)
に基づく悲観的な差分抽出



ステップ3: 整数計画法に
基づく厳密手法



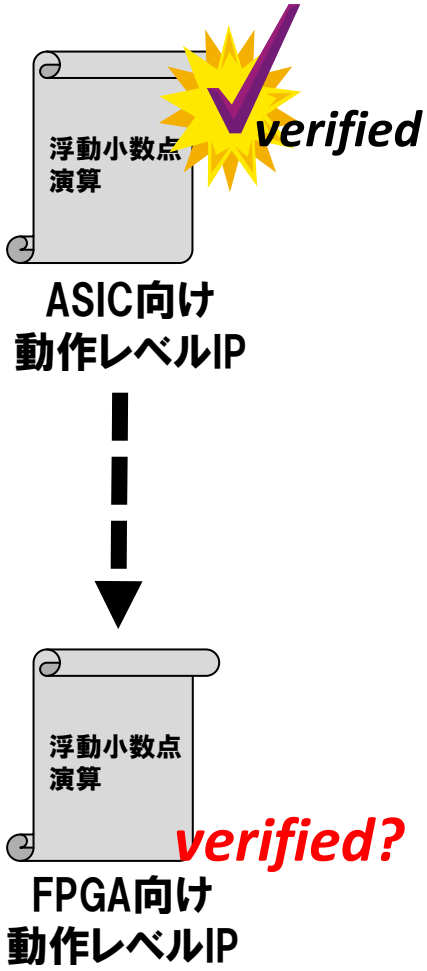
最小差分解

評価結果

- ■8x8 IDCTのCプログラム(約100行)の10行を修正し差分を抽出
  - 1秒以内に差分抽出が可能
- ■ヒューリスティックを組み合わせれば数千行でも差分抽出可能

FLEC実用化シナリオ

検証済動作レベルIPを設計変更した際の機能検証工数を大幅削減



検証済み動作レベルIP

例）CyberWareの浮動小数点演算器
数か月かけてロングラン検証済

機能を変えない設計変更

設計現場では頻繁に発生
例１） ASIC向け資産をFPGA向けに変更
例２） 動作周波数の向上・面積削減
例３） ソフトウェアアルゴリズムのハードウェア化

課題

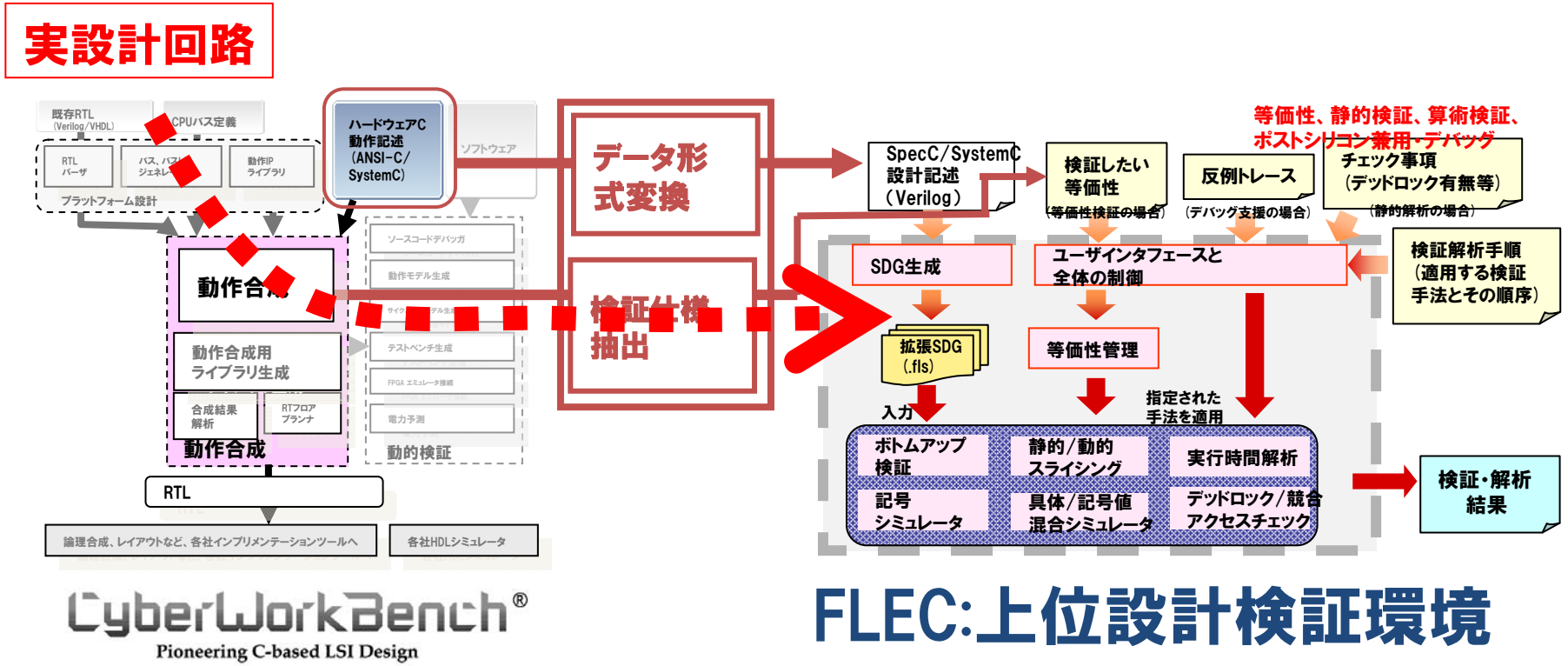
機能検証のコスト大（数週間～数か月）

FLECによる等価性検証

人手による検証パターン作成なし、検証時間大幅短縮
特に、設計変更前後に特化し、現在の手法の実用化を狙う

FLEC/CWB インタフェイスツール (cwb2flec)

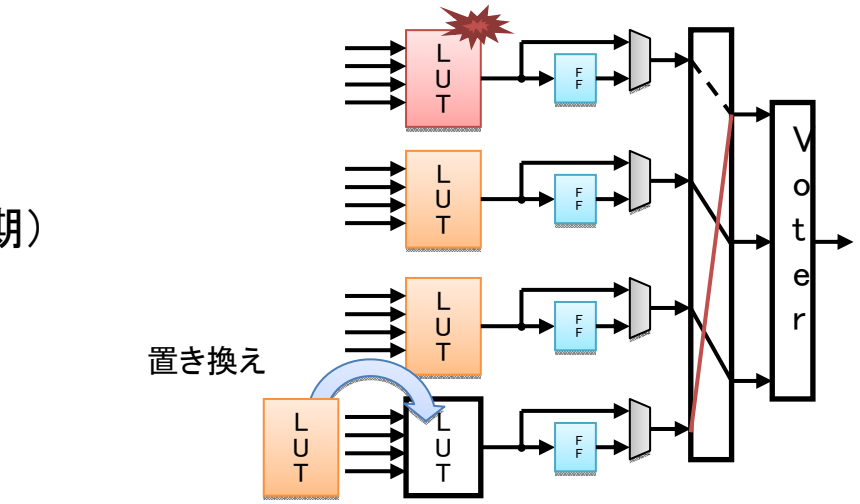
- ANSI-CをFLEC入力可能なSpecCに変換
- FLEC 制限構文(ポインタ・関数)を解消
- 設計変更部分のみを抽出する機能
- 階層的検証により規模問題に対処する機能

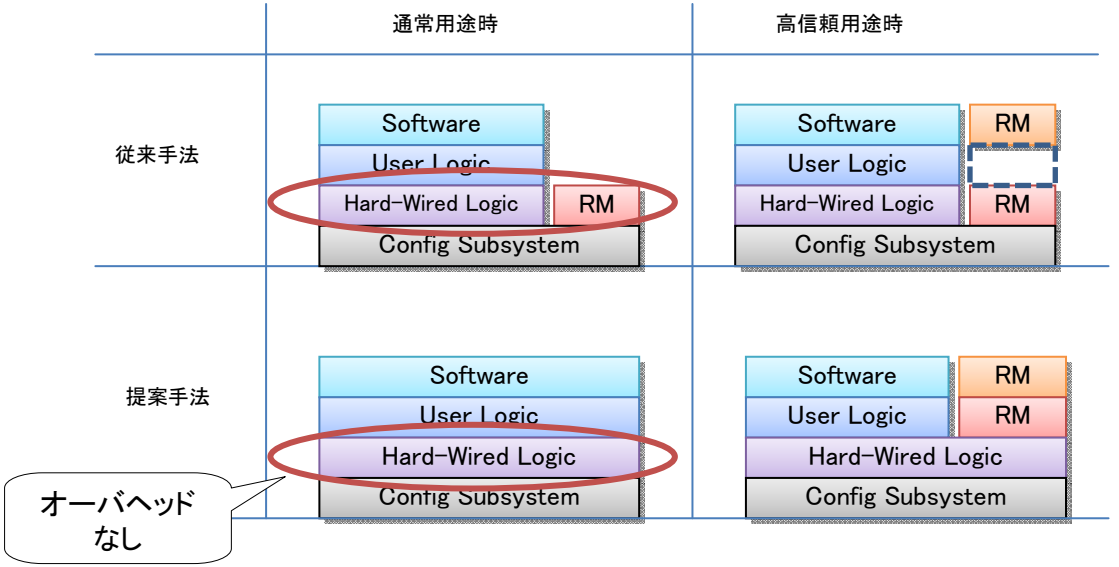


ディペンダブルアーキテクチャ（坂井、五島）

耐過渡・耐永久故障 FPGA アーキテクチャ

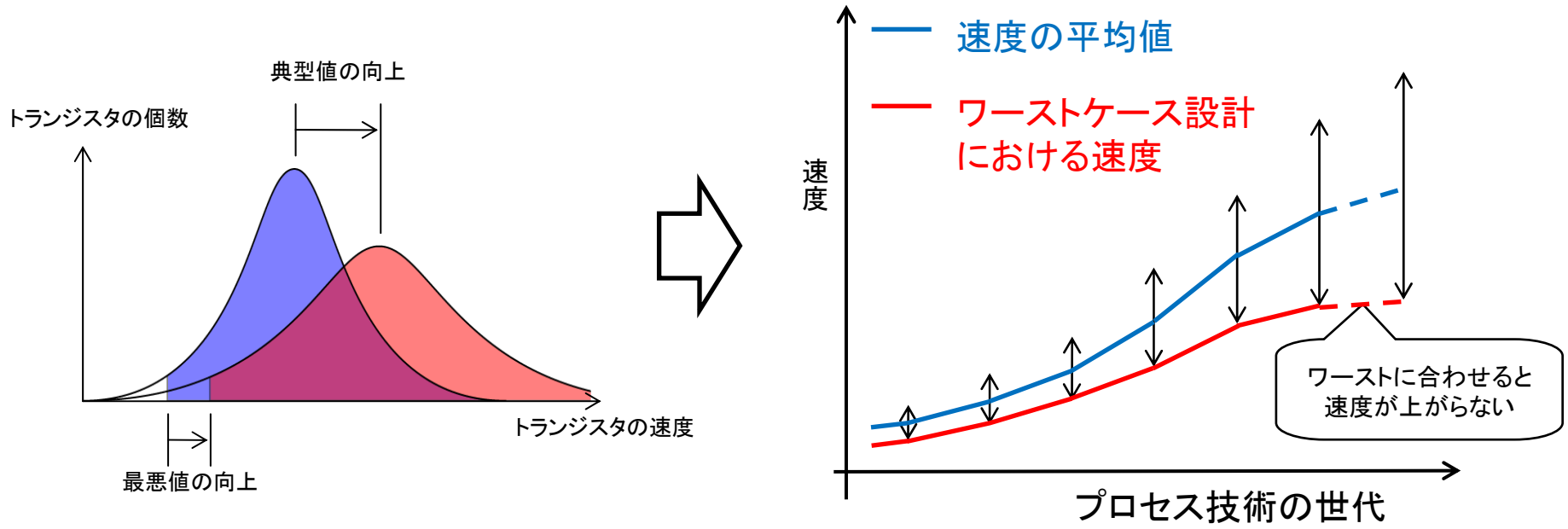
- 使いやすい高信頼 LSI の要求
  - 従来：MIL 規格品
    - 低性能，高価，低入手性（長納期）
  - 提案：FPGA＋TMR＋DPR
    - 高性能，安価，高入手性
- ポイント：通常用途との両立
  - 追加ロジックを最小化
- 回復を行う Recovery Manager
  - 従来方式：ハードワイアード
  - 提案方式：ユーザロジック
- 成果：
  - 「TMR は局所的がよい」
- 今後の予定：
  - 再配置配線の手法の開発





ばらつきの増大とワーストケース設計

- LSI のランダムばらつき
  - プロセスの進化 → ランダムばらつき大
- ワース・トケース設計は悲観的になり過ぎる
  - ワースト・ケースではなく『現物』の遅延に合わせる
  - ⇒ 動的タイミング・フォールト検出・回復



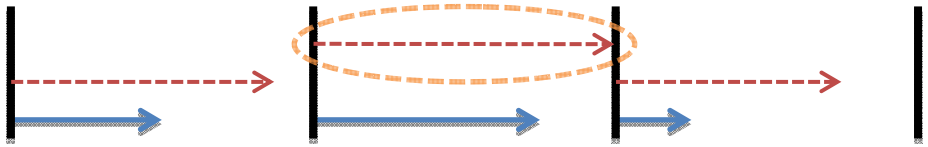
動的タイムボローイングを可能とするクロッキング方式

：ワースト遅延

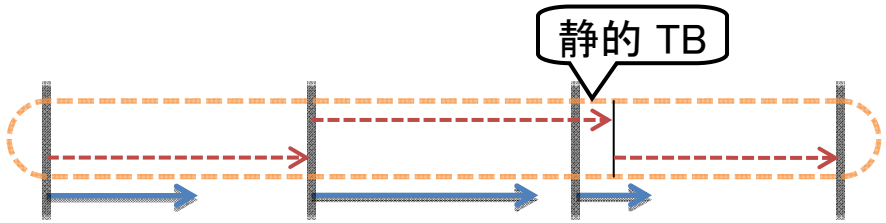
：実効遅延

動作周波数を決めるのは

単相 FF



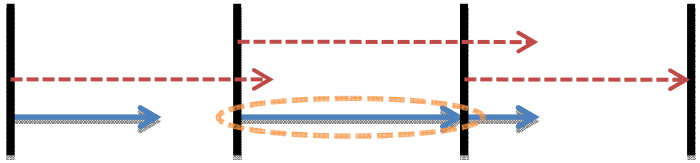
ワースト遅延のワースト



二相ラッチ

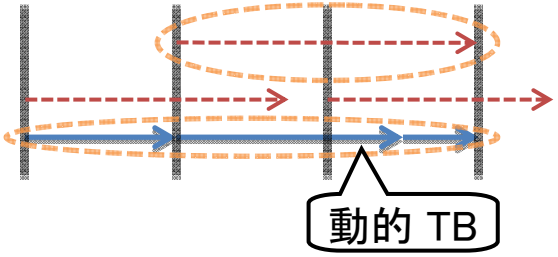
ワースト遅延の累積

Razor



実効遅延のワースト

提案 ＝
Razor ＋
二層ラッチ

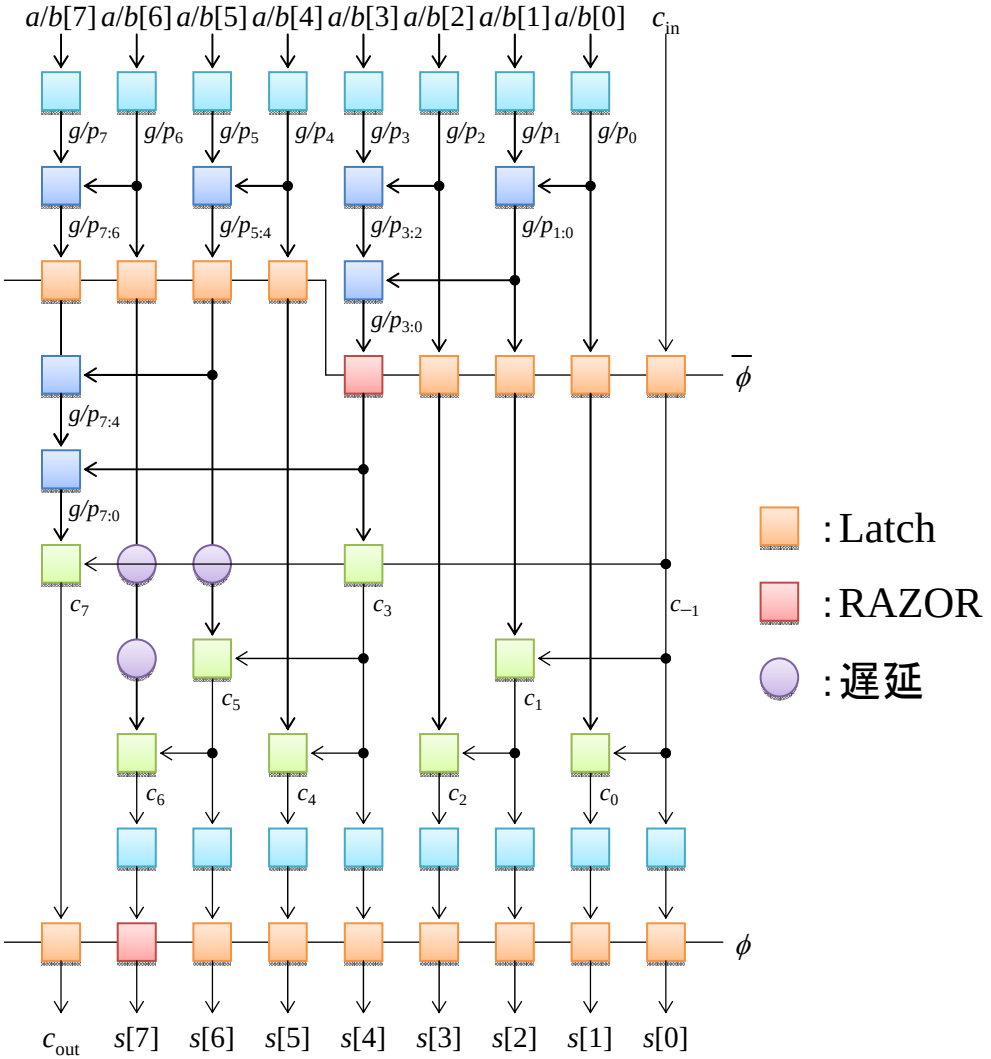


実効遅延の累積

ワースト遅延のワーストの半分

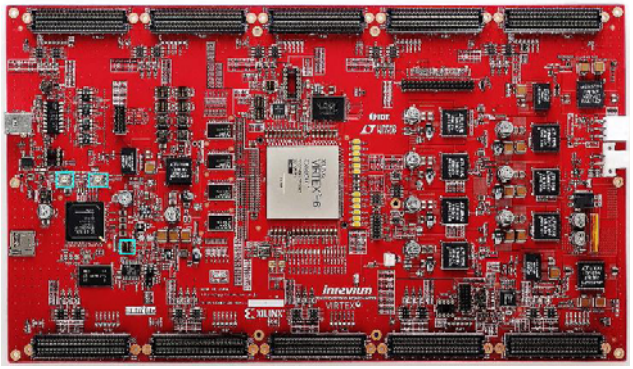
動的タイムボローイングを可能とするクロッキング方式

- 成果：
  - 動的タイムボロー可能なクロッキング方式の提案
  - リプル・キャリー/キャリー・ルックアヘッド・アダーのカウンタを FPGA 上に実装，動作を確認
- 今後の計画：
  - FPU，プロセッサに適用

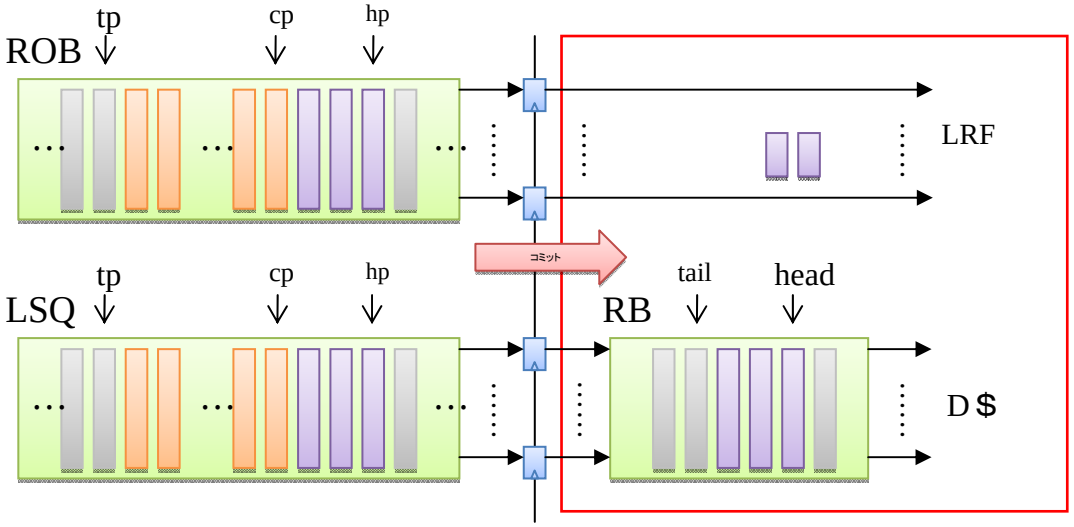


耐タイミング故障 OoO スーパスカラ・プロセッサ

- 成果
  - 故障検出・回復機能付き OoO スーパスカラ・プロセッサを FPGA で試作，基本動作を確認
  - LSQ に生じるタイミング故障に対応するコミット方式を提案
- 今後の計画： 試作・評価
  - TSMC 60nm プロセスを用いて，タイミング故障検出能力を持つキャッシュ・メモリを試作中
  - ハイパフォーマンス系の研究成果を詰め込んだ本格的なプロセッサを試作予定

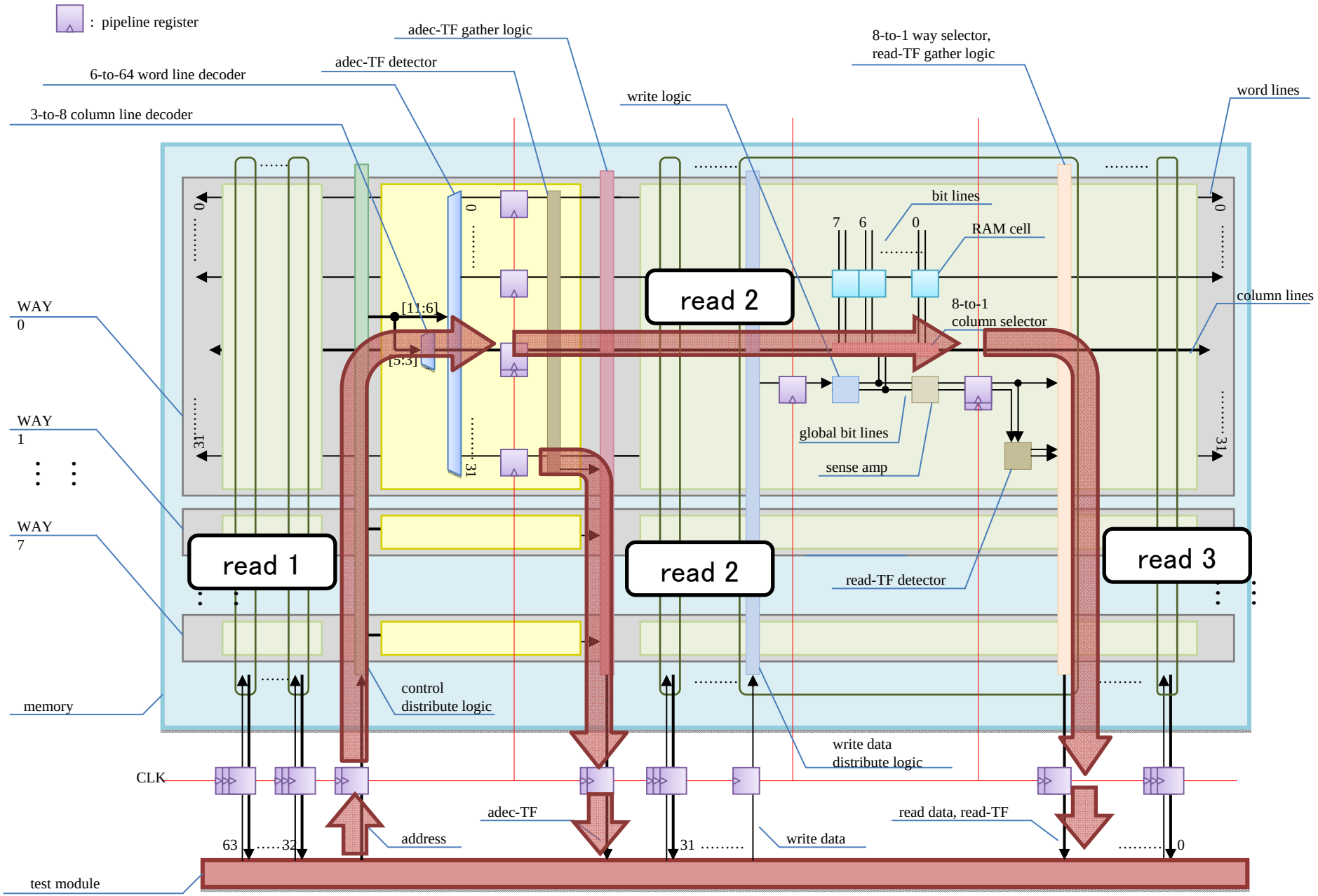


新規開発中のプロセッサを実装予定
の 大容量 FPGA 基盤



新コミット方式ブロック図

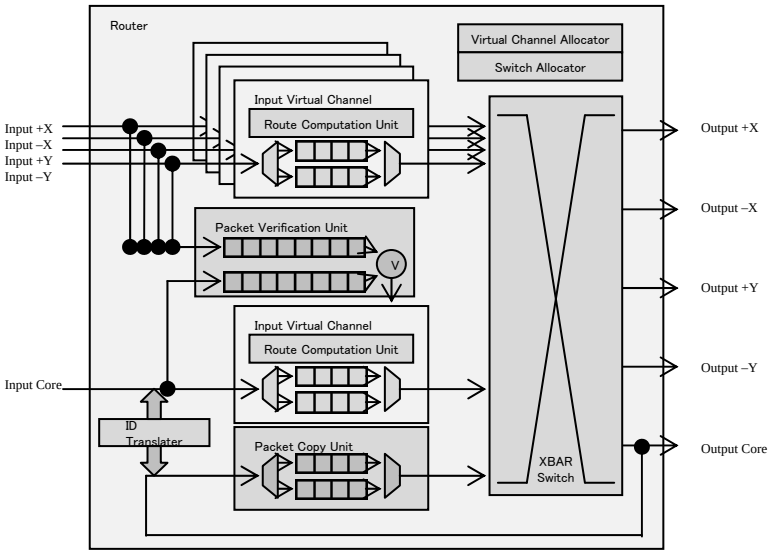
キャッシュと LSQ の間に Retire Buffer を追加
仮に LSQ でタイミング故障が発生しても Retire Buffer に正しいステートが保持されている


: pipeline register
adec-TF gather logic
adec-TF detector
6-to-64 word line decoder
3-to-8 column line decoder
write logic
8-to-1 way selector,
read-TF gather logic
word lines
bit lines
RAM cell
column lines
8-to-1
column selector
[11:6]
[5:3]
WAY
0
WAY
1
WAY
7
read 1
read 2
read 2
read 3
global bit lines
sense amp
read-TF detector
memory
control
distribute logic
write data
distribute logic
CLK
adec-TF
address
write data
read data, read-TF
test module

ディペンダビリティ支援ルータ（吉瀬）

超ディペンダビリティ支援高機能ルータグループ

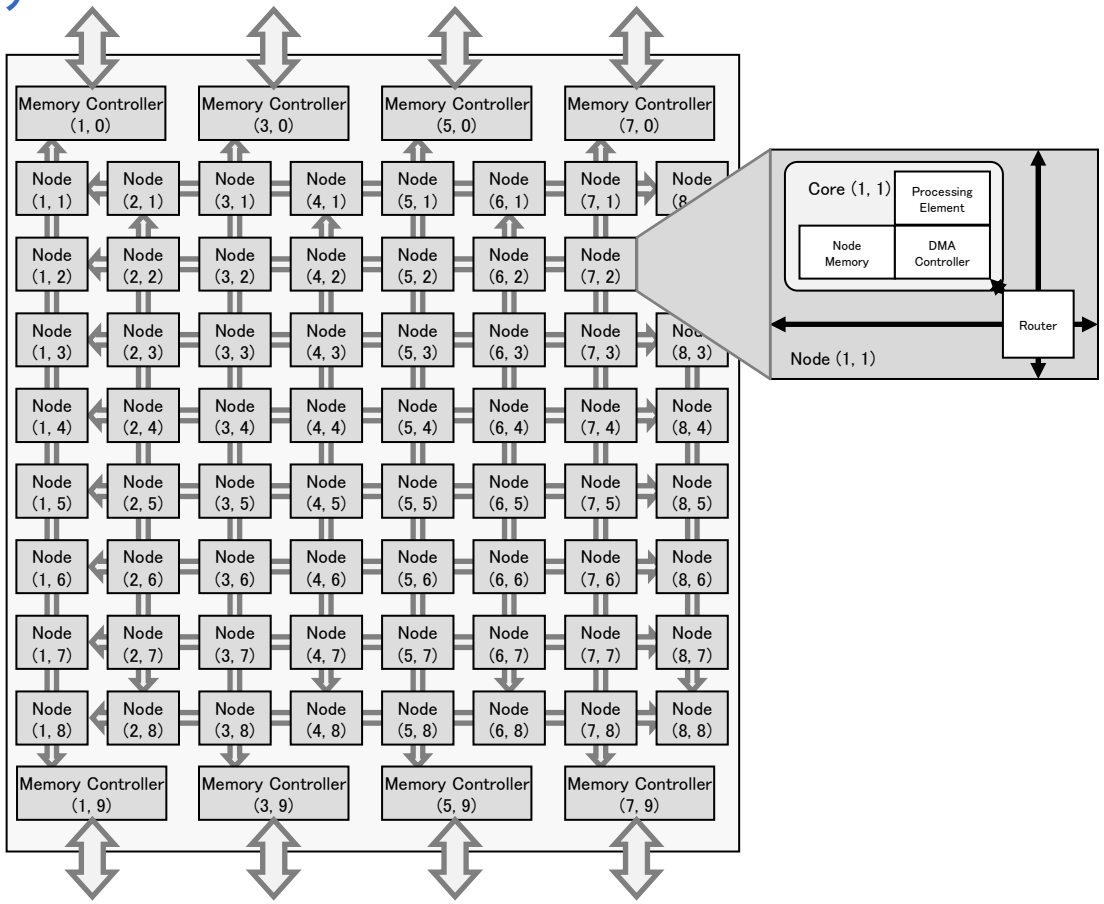
- SmartCoreシステム
  - 高機能ルータを核として送受信パケットのレベルで多重実行を実現するシステム
  - 多数のコアと高機能ルータによってディペンダビリティ向上と速度向上を目指す
- マルチコアシミュレータ，FPGAプロトタイプシステム
- マルチコア開発支援ソフトウェア
  - タスク配置手法



高機能ルータアーキテクチャ

冗長実行自動支援のためのパケットの複製、同一性検出、マージの機能を実現

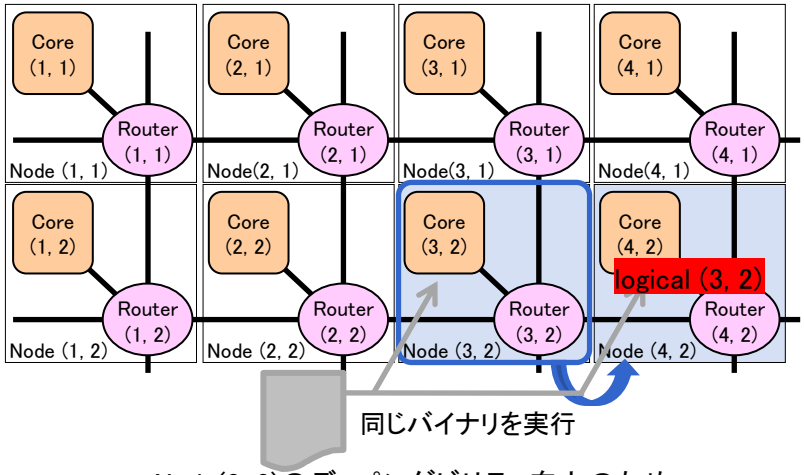
- パケットを送信する宛先を調整
- パケットを比較してエラー検出
- パケットを複製



高機能ルータをもつマルチコアシステムアーキテクチャ M-Core

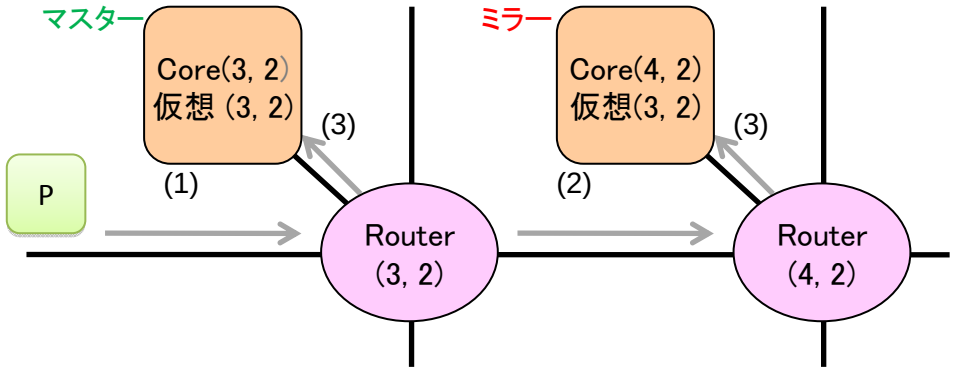
SmartCoreシステムによるメニーコアのディペンダビリティ向上

- ディペンダビリティ向上
  - Node(3,2)のディペンダビリティ向上の為Node(4,2)と多重実行
    - Node(3, 2): マスター
      - 通信をすべてミラーに転送
    - Node(4, 2): ミラー
      - マスターと同じ仮想ID
      - マスターからの通信のみで動作
  - Node(3, 2)のルータでCore(3, 2)とCore(4, 2)からのパケットを比較，エラー検出



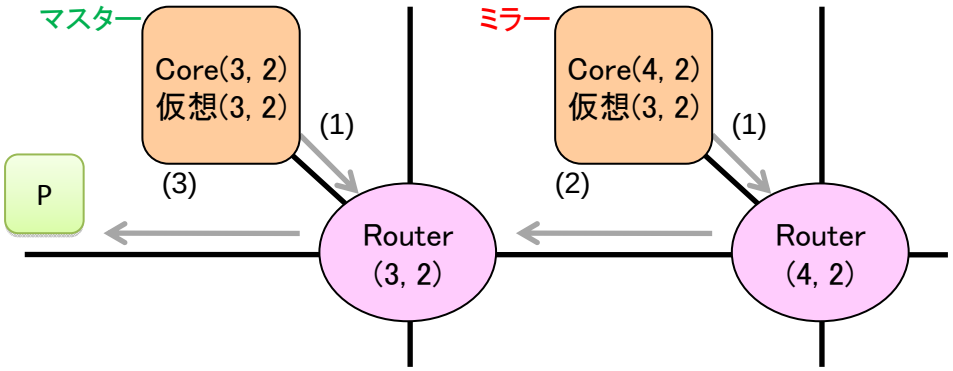
Node(3, 2)のディペンダビリティ向上のため
Node(3, 2)とNode(4, 2)で多重実行する様子

◆パケットの受信と転送



(1) Router(3, 2)にCore(3, 2)が受信するパケットが届く
(2) Router(3, 2)はパケットを複製しCore(4, 2)に送信
(3) 各Coreが同じパケットを受信

◆パケットのチェックと送信

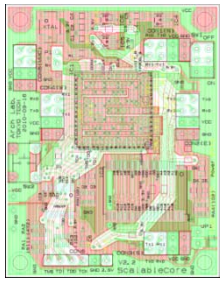


(1) 各Coreがパケットを送信
(2) Router(4, 2)はパケットの送信先を(3,2)に変更
(3) Router(3, 2)はCore(3, 2)とCore(4, 2)からのパケットを待ち合わせ，比較。エラーがなければ送信

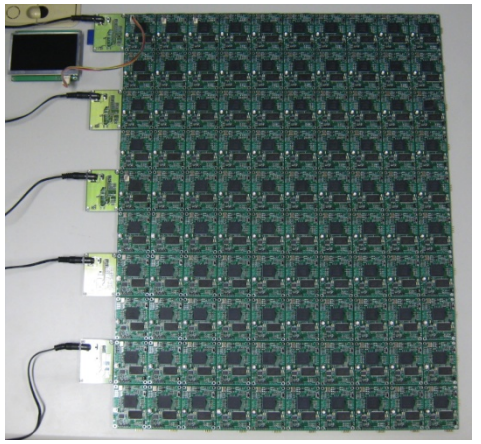
高機能ルータ評価のためのFPGAプロトタイプシステム

FPGAシステムによる評価環境ScalableCoreシステム

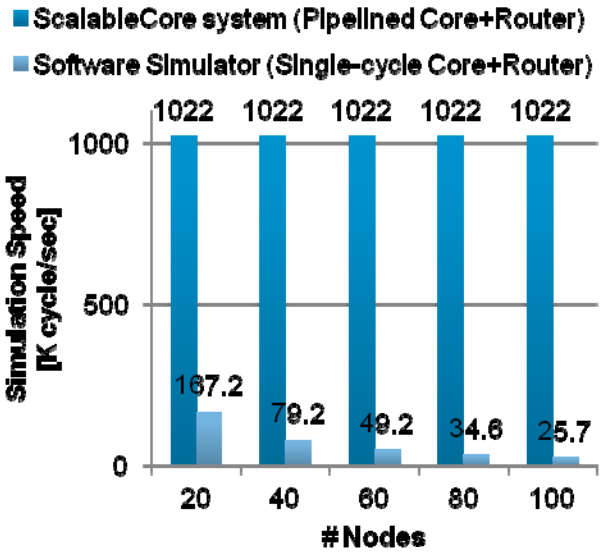
- SmartCoreシステム評価のベースシステムとして
- 新型ScalableCoreシステム3.0
  - より現実的な構成のプロセッサの動作を確認
    - FPGAにパイプラインコアおよびルータを実装
  - SWシミュレータと同等のプログラムの実行が可能
- SWシミュレーション環境との比較
  - 100コアシミュレーション時に約40倍の高速化

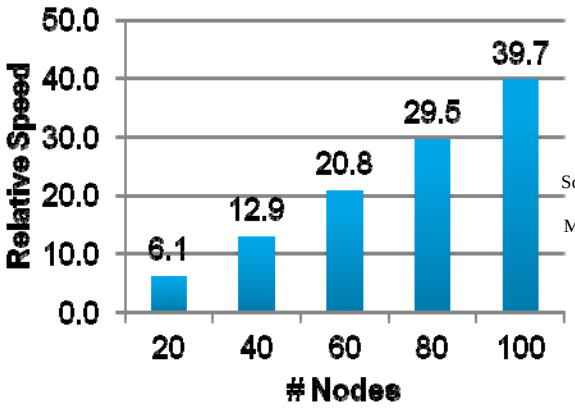


設計基板

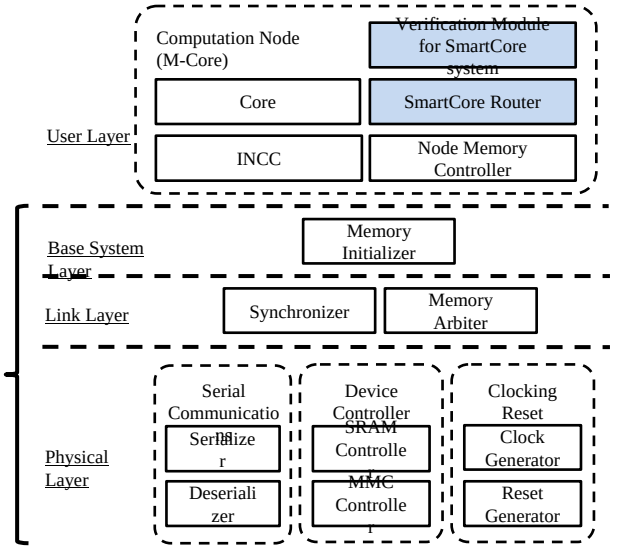


100ノード構成の
ScalableCoreシステム3.0





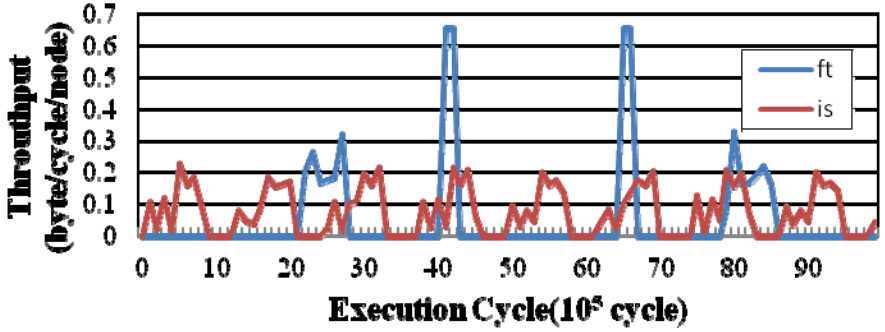
ノード数を変更したときのシミュレーションスピード



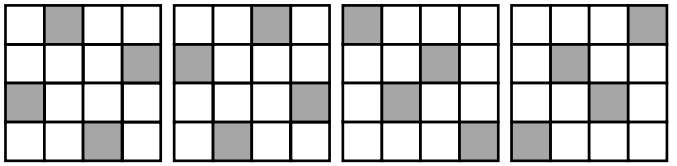
ScalableCoreシステムをベースとする
SmartCoreシステム評価環境の構築モデル

メニーコアプロセッサのためのタスク配置手法

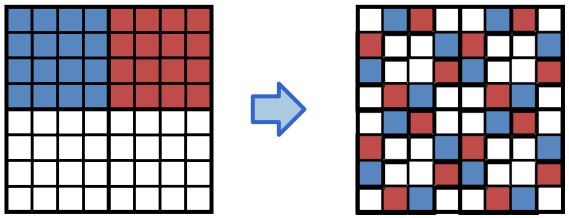
- メニーコアプロセッサのためのタスク配置手法 RMAP を提案，評価
  - 速度向上のために通信衝突の削減が重要
  - アプリケーションごとの通信の時間的局所性を利用
  - XY次元順ルーティングの性質に着目し，通信衝突を削減するタスク配置手法 RMAP Xnを提案
    - 4−ルーク問題の解を利用
    - n個のアプリケーションを重ね合わせる
  - NAS Parallel Benchmarksによる評価
    - RMAP X4で最大11%，平均4.4%の性能向上
  - 今後の課題: 多重実行およびSmartCoreを考慮したタスク配置手法の検討



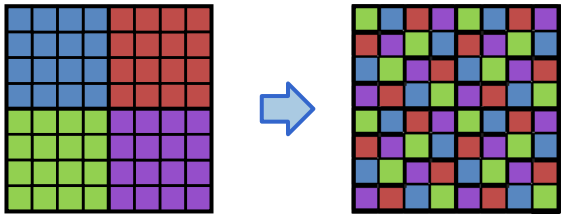
NAS Parallel Benchmarks の ft, is の通信挙動



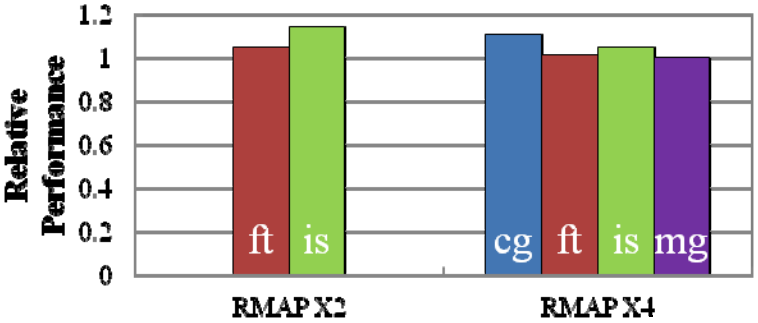
4−ルーク問題の解を利用したタスク配置の4種類の例
（16コアに4個のタスクを配置）



提案手法適用前(2 app)　提案手法 RMAP X2



提案手法適用前(4 app)　提案手法 RMAP X4



タスク配置手法の評価(256コアに複数のアプリケーションを配置,各アプリケーションは64並列で実行)

出口に関する考え方

- 形式検証
  - 同一チーム内共同開発： 藤田（東大）＋若林（NEC）
    - CYBERへの組込試験： 有用性検証
- RTL設計支援
  - 外国EDAベンダとの共同開発に向けた相談
- ポストシリコンデバグ
  - 国内ベンチャーと相談
- 耐タイミング故障アーキテクチャ
  - IP化： PC、サーバに加えて携帯、タブレットも （組込VLSI複雑化）
- FPGA
  - 宇宙
- マルチコア
  - 車

(問題) ディペンダビリティのコスト

- 1%？ 5%、10% 100%？