ディペンダブルネットワークオンチッププラットフォームの構築

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」

研究代表者　米田友洋 (国立情報学研究所)
主たる共同研究者　今井　雅 (東京大学)
松本　敦 (東北大学)
齋藤　寛 (会津大学)

National Institute of Informatics
国立情報学研究所

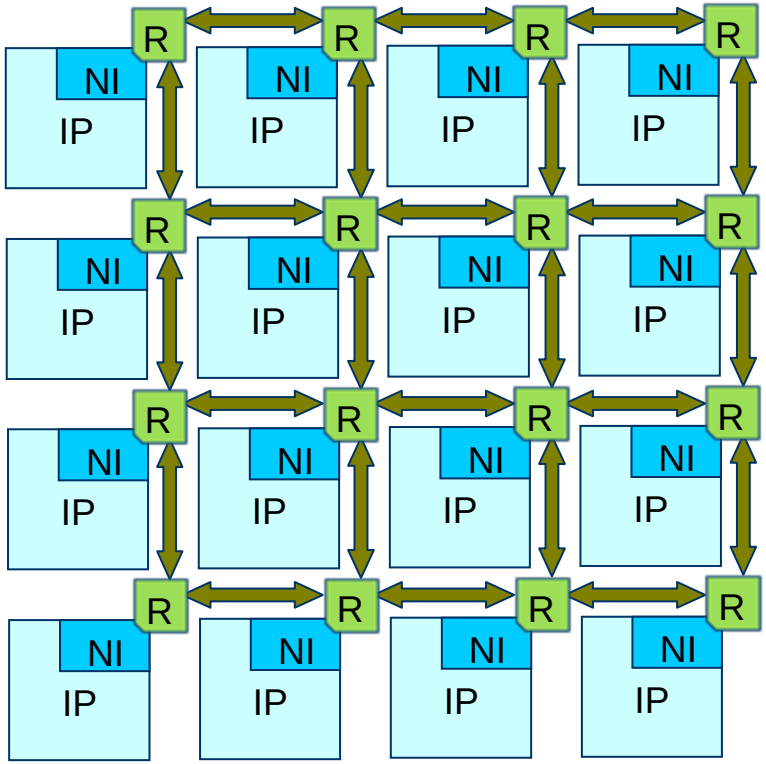
背景

- 一つのVLSIに実装が求められるコア数
  - 近い将来に必ず急激に増加する
- (例) 車載制御系システム
  - さまざまなタイプのECUが50個以上も混在
    - 実装方法にいくつかの課題
  - 新しいアプローチ
    - 集中型ECU
      - 各ECUを統合
    - X-by-Wire方式
      - センサー・アクチュエータ間を光ファイバなどで接続

「多数のコアが適応的に協調動作して異種多様なタスクを効率よく実行できるプラットフォーム」が必要

アプローチ

- ネットワークオンチップ(NoC)



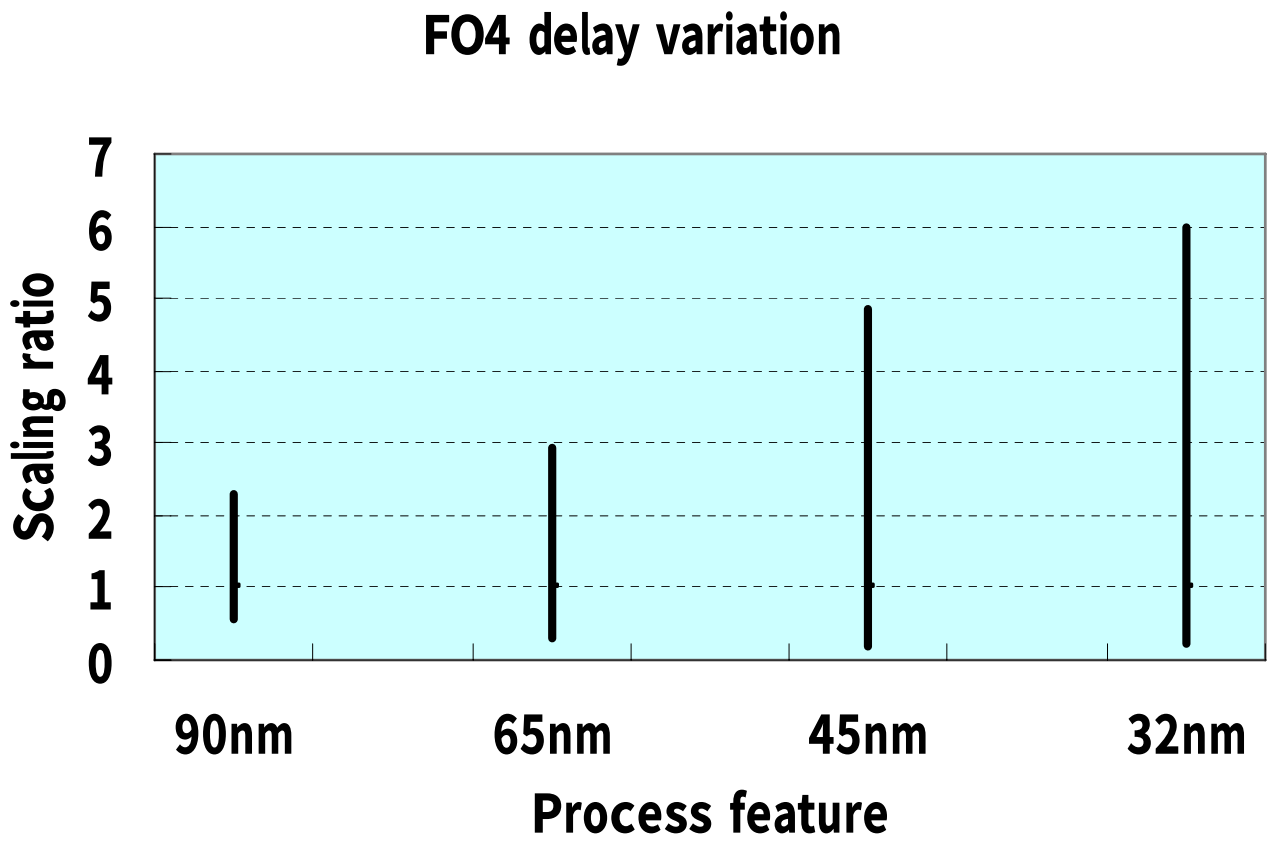
IP：CPUコアやアクセラレータ・メモリ等
NI: ネットワークインターフェース
R: ルータ

- 大域非同期局所同期(GALS)方式

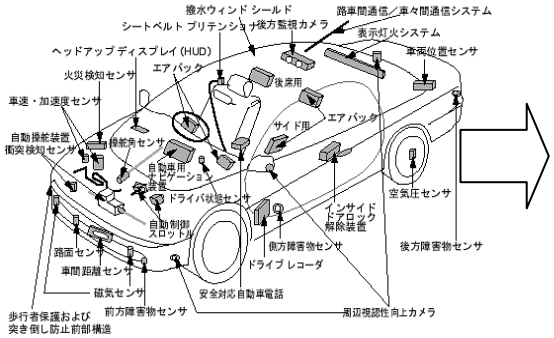
本研究課題で解決する問題

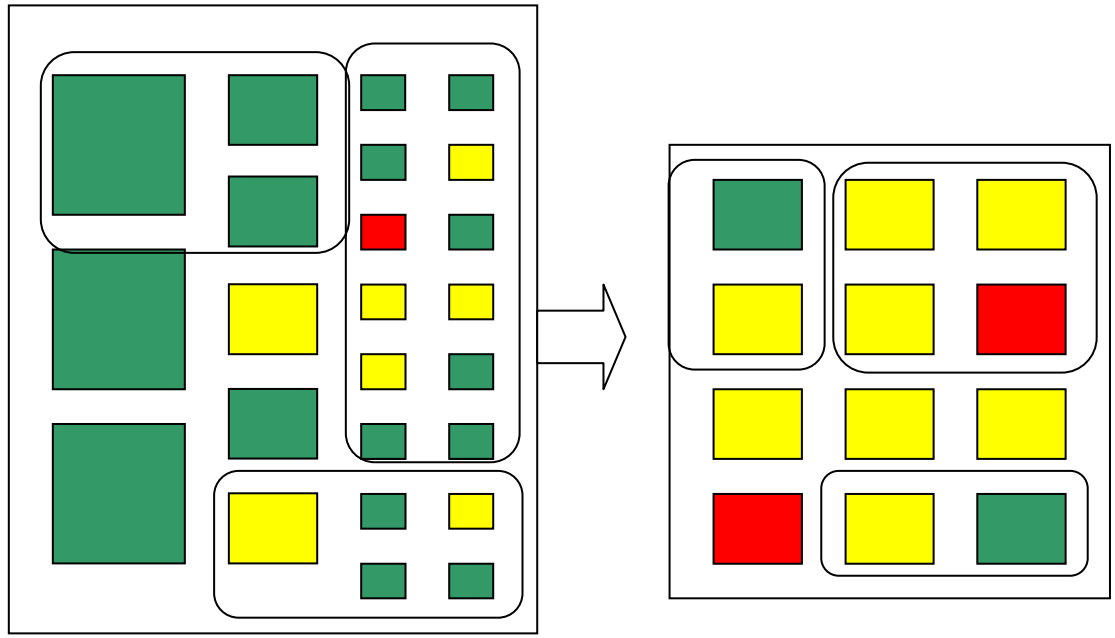
- GALS-NoC方式でも以下のような問題がある
  - 微細化に伴うコア内の局所的な性能劣化
    - 高ディペンダビリティ化が必要
  - コアの寄せ集めによる冗長・不要部分の増加
    - 高アダプタビリティ化が必要
  - GALSにおける通信の非同期化に伴う，速度および面積オーバーヘッドの増加
    - 高性能化の追求

微細化に伴うコア内の局所的な性能劣化



コアの寄せ集めによる冗長・不要部分の増加 (1)





点在するコアを集積

機能共有によるコア集約

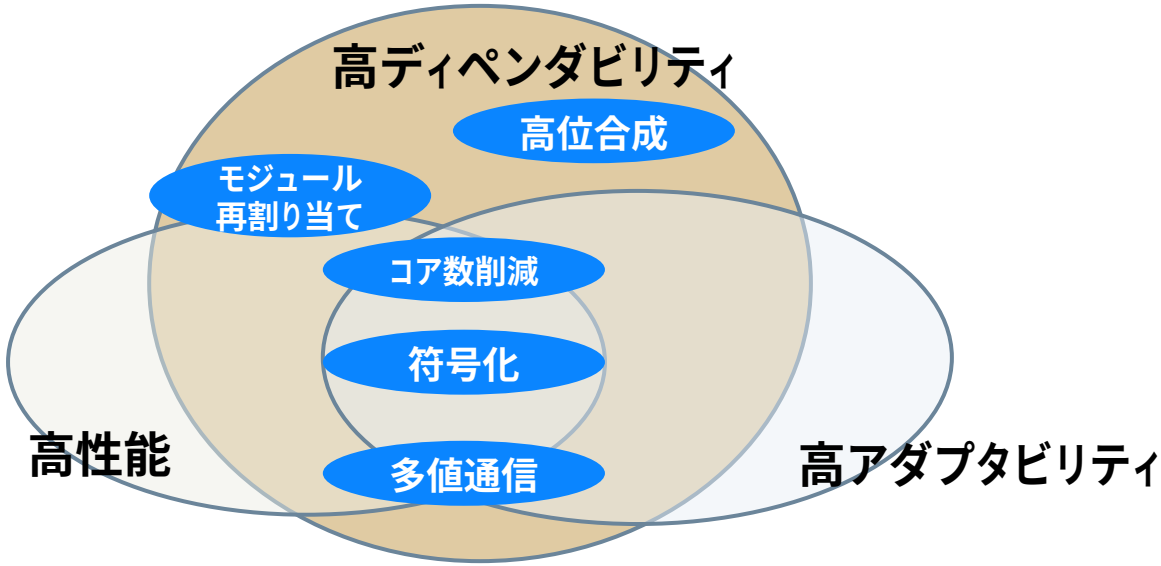
動作率：高　動作率：中　動作率：低

コアの寄せ集めによる冗長・不要部分の増加 (2)

- 従来手法
  - センサ・アクチュエータとECUの対応が固定
    - 能力が余っても他に流用不可
    - 劣化・故障時の対応が困難
- NoCアプローチ
  - センサ・アクチュエータの制御はどのコアも可
    - 利用可能な資源を有効に活用できる
    - 劣化・故障時には他のコアを利用できる

研究計画

- 高ディペンダビリティ化・高アダプタビリティ化・高性能化実現のための要素技術開発



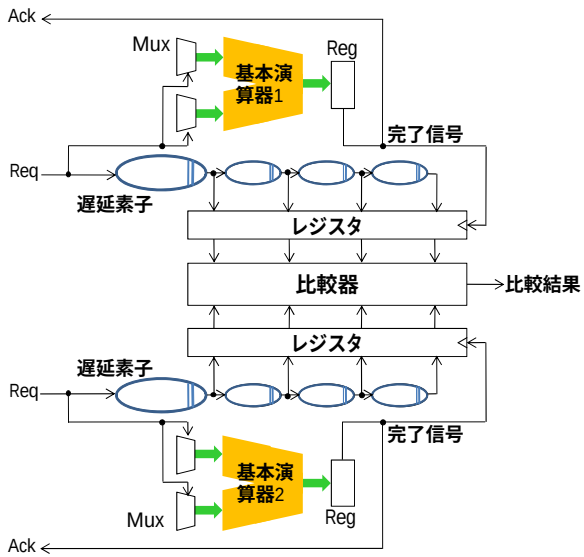
- NoCプロトタイプの設計・試作実験

要素技術開発（1）

- 耐劣化性の実現
  - 非同期式回路技術を利用
    - 演算完了信号　→　演算結果の格納
      - 性能劣化が生じても常に正しい検算結果が得られる
    - 平均遅延動作を行う
      - 劣化した部分は遅く，正常部分は早く動作
    - 突然の誤動作は起こらず，徐々に演算時間が延びる
      - 検出が容易
  - 劣化した部分を同定し，その部分に負荷の軽いタスクや基本演算を割り当て
  - いろいろなレベル
    - 静的／動的
    - コア／基本演算器

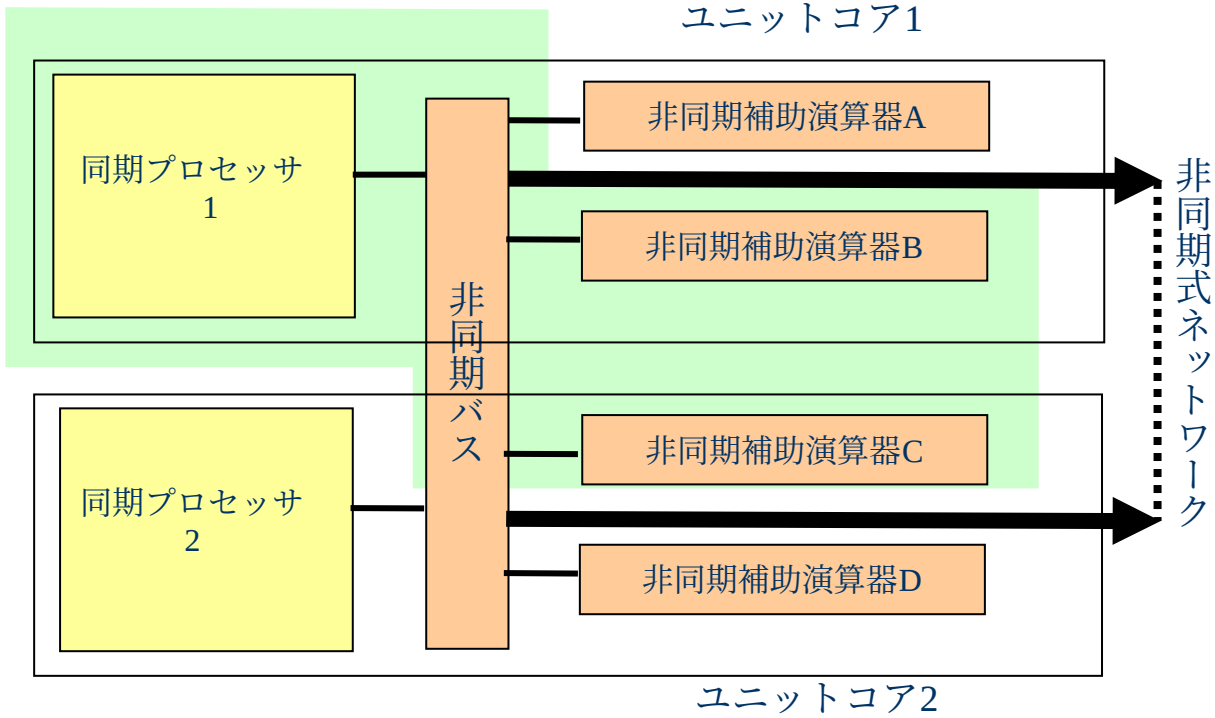
要素技術開発 (2)

- 静的な性能劣化
  - プロセスパラメータのばらつき等による
  - 製造時に性能測定 → タスク・モジュール割り当て決定
- 動的な性能劣化
  - NBTI: Negative Bias Temperature Instability等による
  - 動作中に劣化を検出 → タスク・モジュール割り当て変更
    - 劣化部に負荷の軽いタスクや基本演算を割り当て
  - 自律的に行え，オーバーヘッドの小さい方式が望ましい



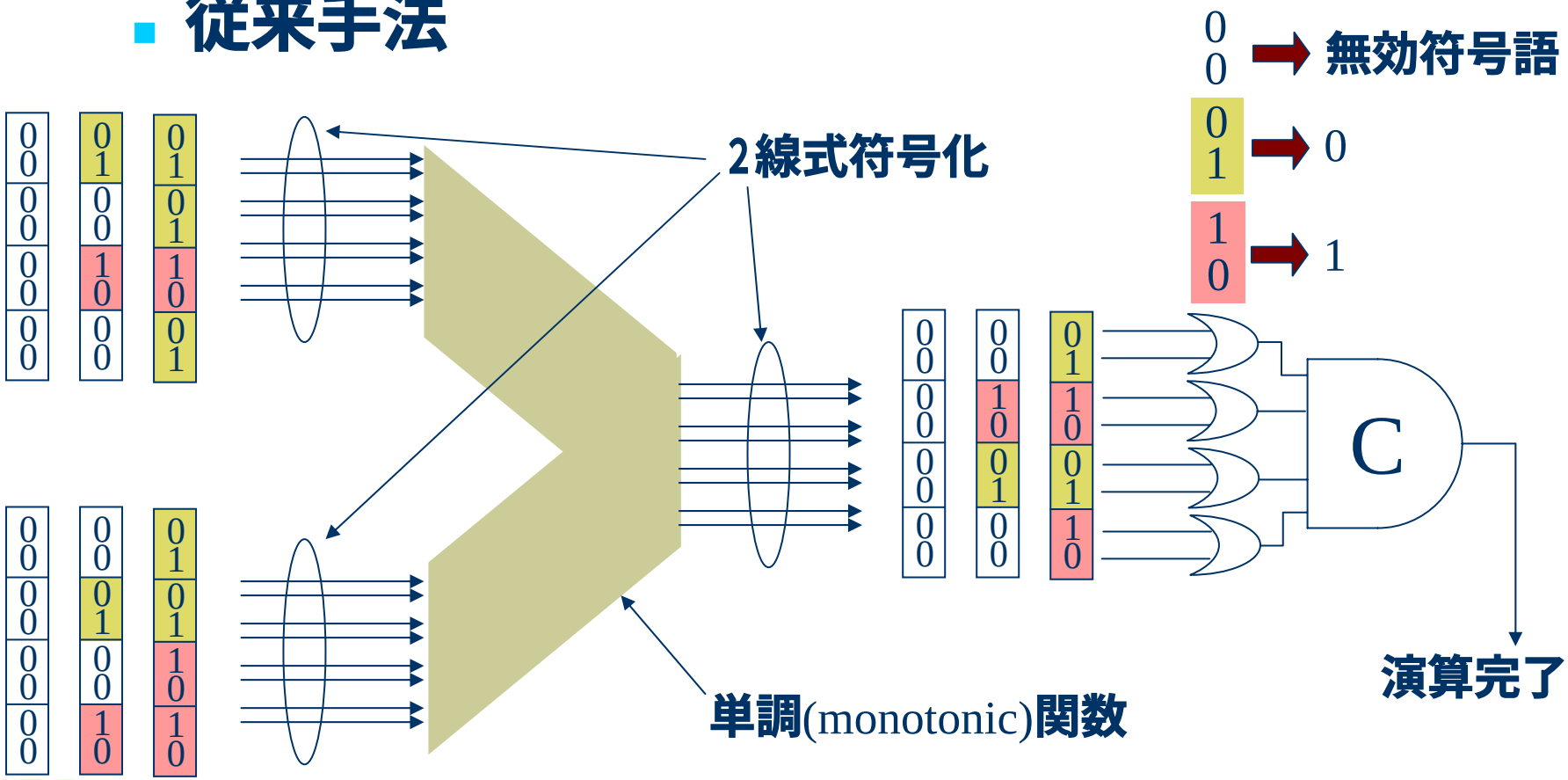
要素技術開発(3)

- 適応型コアの開発
  - 機能と性能を可変化する機構
  - 動作率によってタスクを適切にスケジューリング



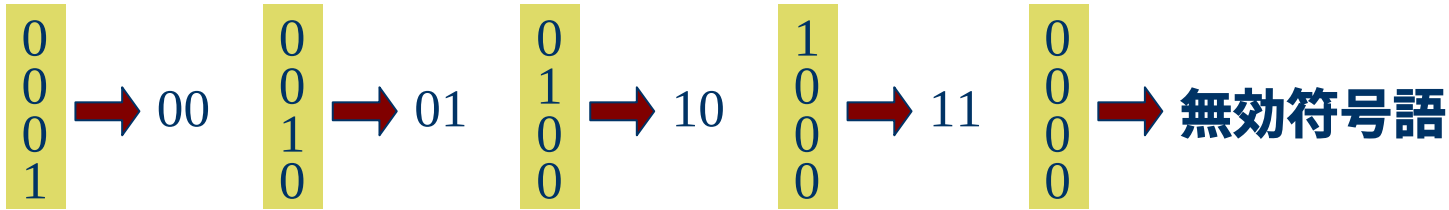
要素技術開発(4)

- 非同期式回路のための効率のよい符号化技術
  - 従来手法



要素技術開発(5)

- 非同期式回路のための効率のよい符号化技術
  - 1 out of 4符号の使用

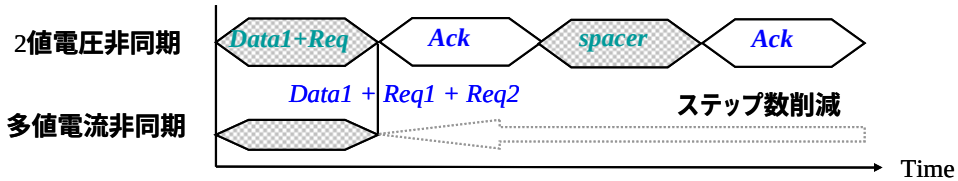


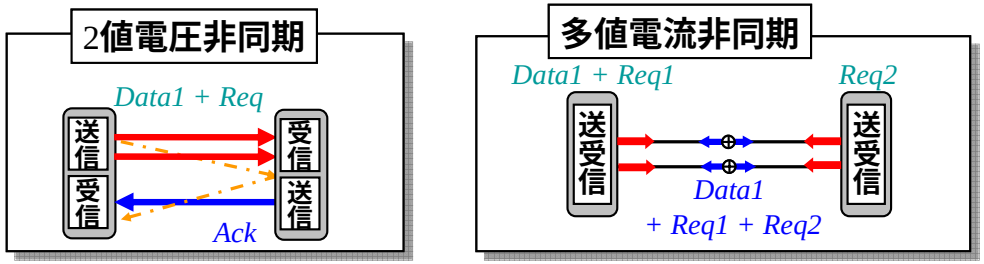
- 省電力化，高速化の可能性
- ただし，演算部分への適用にはツール整備が不可欠

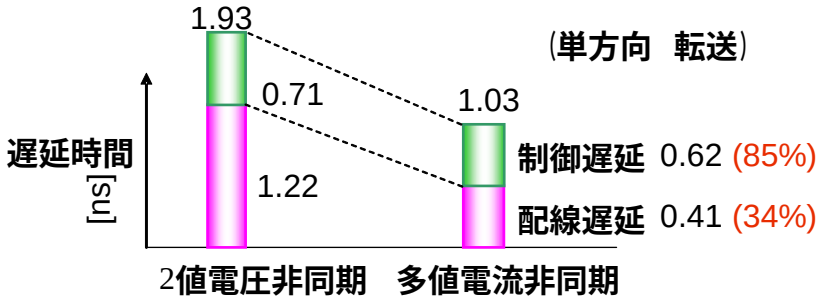
要素技術開発（6）

- 多値回路に基づく非同期式通信方式

［一回のデータ転送に必要なステップ数］







- 電流モード多値回路を用いた原理動作は実チップで実証済み
- 新しい非同期式データ転送（シングルトラックetc）と電流モード多値回路を融合させ，高速化･低消費電力化をさらに推進

NoCプロトタイプ設計・試作（1）

- 車載制御系システムを想定
  - 仕様
    - エンジン制御，ブレーキ制御等の計算量の大きい処理
    - マンマシン系の優先度が低い処理
  - 研究の初期の段階
    - ベンチマークによる仮想的なアプリケーション
    - ある程度の評価結果
  - カーメーカーや車載システムメーカに評価結果を提示
    - さらに具体的なアプリケーションの提示を受け，より具体的なアプリケーションを決定

NoCプロトタイプ設計・試作（2）

- プロトタイプの構成
  - 4台のCPUユニットコアモジュール
  - 有限要素法ハードウェアアクセラレータ
  - メモリモジュール，センサインタフェースモジュール，アクチュエータインタフェースモジュール等
  - リアルタイムOS

有限要素法用アクセラレータの構想

- 流れ，燃焼の解析
  - エンジン制御のため，リアルタイム実行の要求がある
  - 手法
    - 有限体積法
      - 計算量は少ないが，取り扱いが難しい
    - 有限要素法
      - 取り扱いはやさしいが，計算量は大きい → アクセラレータの必要性が大きいかもしれない

流れ解析における有限要素法 (1)

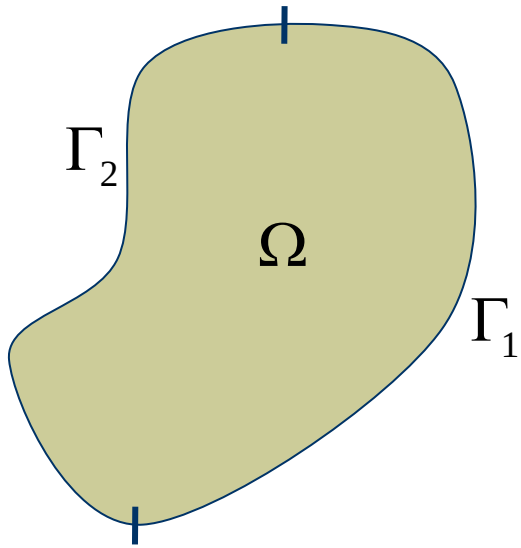
- **支配方程式の例**

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0 \quad (\Omega\text{内})$$

- **境界条件の例**

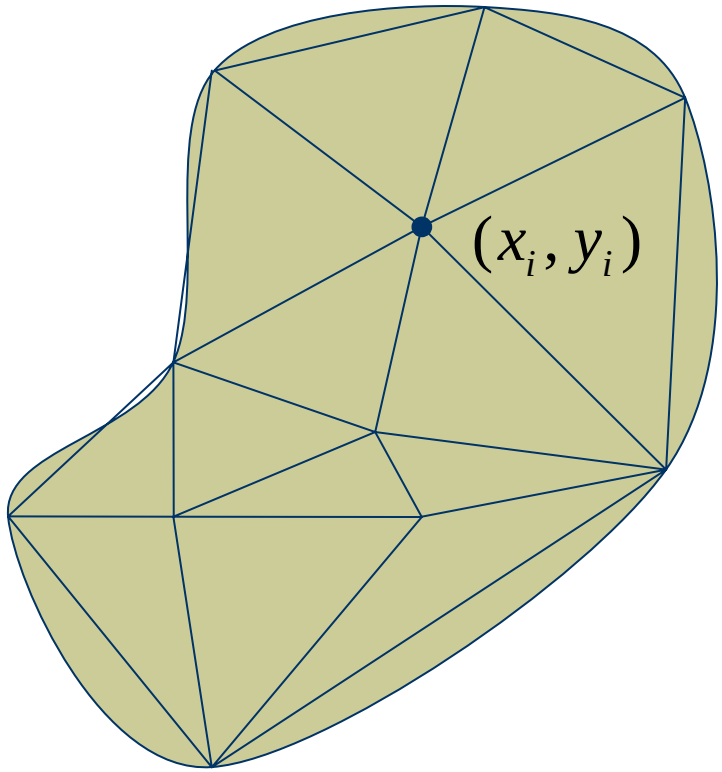
$$\frac{\partial \phi}{\partial n} = \hat{q}(x, y) \quad (\Gamma_1\text{上})$$

$$\phi = \hat{\phi}(x, y) \quad (\Gamma_2\text{上})$$



流れ解析における有限要素法 (2)

- 領域のメッシュ表現



各頂点の座標は既知

(x_1, y_1)

(x_2, y_2)

(x_3, y_3)

…

各頂点のϕ値が未知数

$\phi(x_1, y_1)$

$\phi(x_2, y_2)$

$\phi(x_3, y_3)$

…

流れ解析における有限要素法(3)

- メッシュの各要素中では，ϕがx,y,zの一次式と仮定することで，

$$A\phi^{n+1} = B\phi^{n} + C$$

の形となる．ただし，

ϕ^n等は時刻t^nにおけるϕ

A,Bは形状により決まる行列，

Cは境界条件により決まるベクトル

- これより，下記の形の線形1次方程式が解ければよい

$$A\phi = d$$

線形1次方程式のソルバ

- Gauss-Seidel法

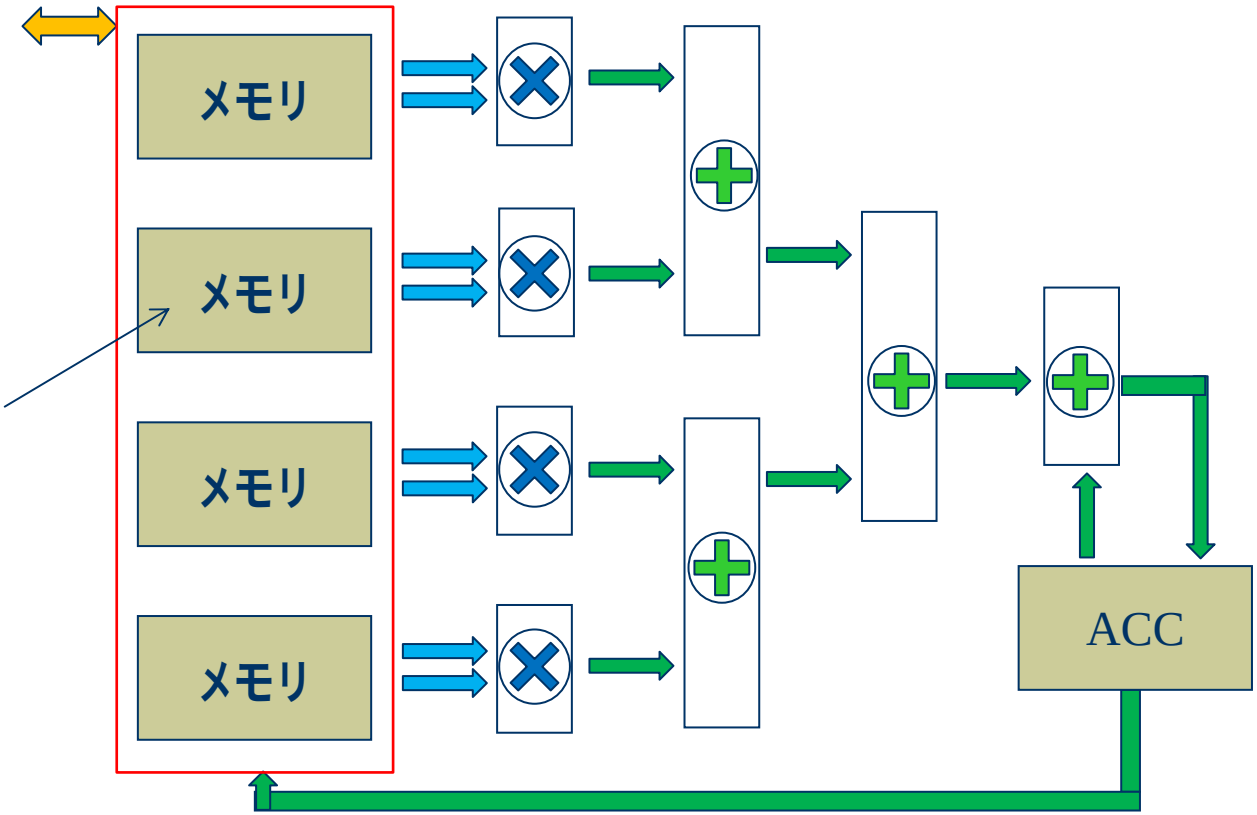
$$\begin{bmatrix} & & & & \\ a_{i1} & \cdots & a_{ii} & \cdots & a_{in} \\ & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right)$$

基本構成

外部との
インタフェース

行列，ベクトルを
分割して格納

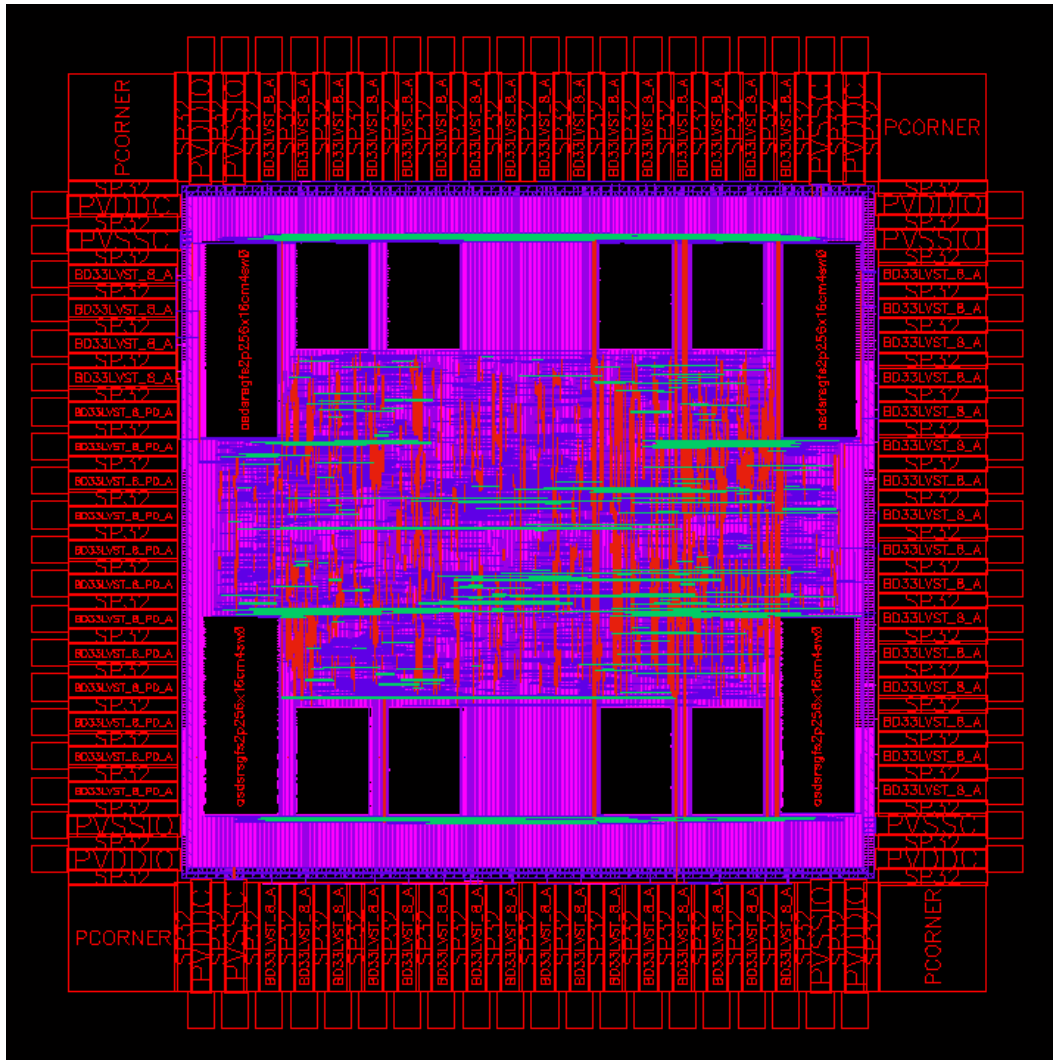


アクセラレータの特徴

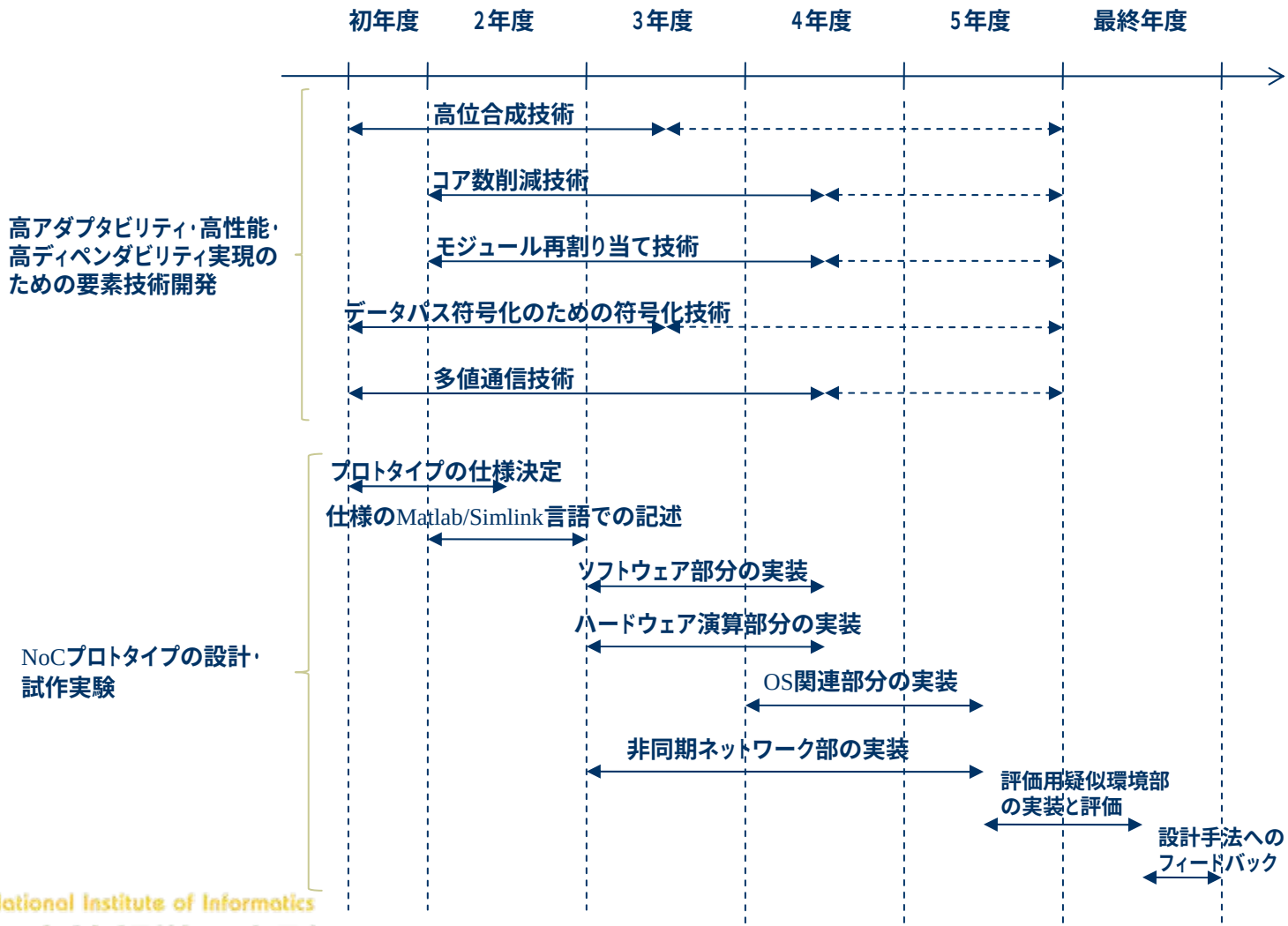
- 演算ユニットを複数用意し，並列実行
- 劣化した演算ユニットの同定，負荷の軽減は自律的に実行
- 完全非同期式設計

試作

- 32次元
- 16ビット変数
- 130nmプロセス
- 50μS以下で解



スケジュール


初年度
2年度
3年度
4年度
5年度
最終年度
高アダプタビリティ・高性能・高ディペンダビリティ実現のための要素技術開発
高位合成技術
コア数削減技術
モジュール再割り当て技術
データパス符号化のための符号化技術
多値通信技術
NoCプロトタイプの設計・試作実験
プロトタイプの仕様決定
仕様のMatlab/Simlink言語での記述
ソフトウェア部分の実装
ハードウェア演算部分の実装
OS関連部分の実装
非同期ネットワーク部の実装
評価用疑似環境部の実装と評価
設計手法へのフィードバック