統合的高信頼化設計のための モデル化と検出・訂正・回復技術

研究代表者　安浦　寛人
（代理　松永　裕介）
九州大学 システムLSI研究センター

研究の目標とアプローチ

- さまざまな種類のエラー（製造故障、ソフトエラー、タイミングエラー、設計誤り、不完全な仕様に基づく誤り、悪意のある攻撃など）に対して、統一的な視点からディジタルVLSIシステムのディペンダビリティを確保するための設計技術の確立を目指す。
- ディペンダビリティの解析と対策回路の合成を行うEDAツールを核とした、ディペンダブルLSI向け設計フローを構築する。
- 具体的な問題から、一般化、ツール構築、フロー構築へと展開する。

ソフトエラーやセキュリティを
考慮したEDAツールが存在しない

22年度の研究計画

（1）ソフトエラー対策を考慮した設計技術の確立

ソフトエラーに対する信頼性の指標としては、VLSIやシステム全体に対するソフトエラー率が考えられるが、個々のデバイスにおけるソフトエラー率から全体のソフトエラー率を厳密に算出することは計算量的に非常に困難であり、現実的ではない。そこで、設計の各階層においてソフトエラーの振る舞いの適切なモデル化を行い、解析ツール（アナライザ）および合成・最適化ツール（エンハンサ）の開発を進める。

（2）タイミングエラー対策を考慮した設計技術の確立

個々のトランジスタの遅延ばらつきが論理回路でのタイミング違反に至る過程を評価するための指標を検討する。それを与えるための解析技術も重要である。特にタイミング違反回避FFの頑強度の解析を検討し、ツール化を視野に入れた解析モデルの構築を目指す。

（3）LSIの搭載している価値や信用を守るための設計技術の確立

21年度に開発した機密情報の漏洩のしにくさを表すモデルを用いて、他の暗号回路へ応用と面積やテスタビリティとの関係を示す。

22年度の研究体制

研究代表者

主たる共同研究者

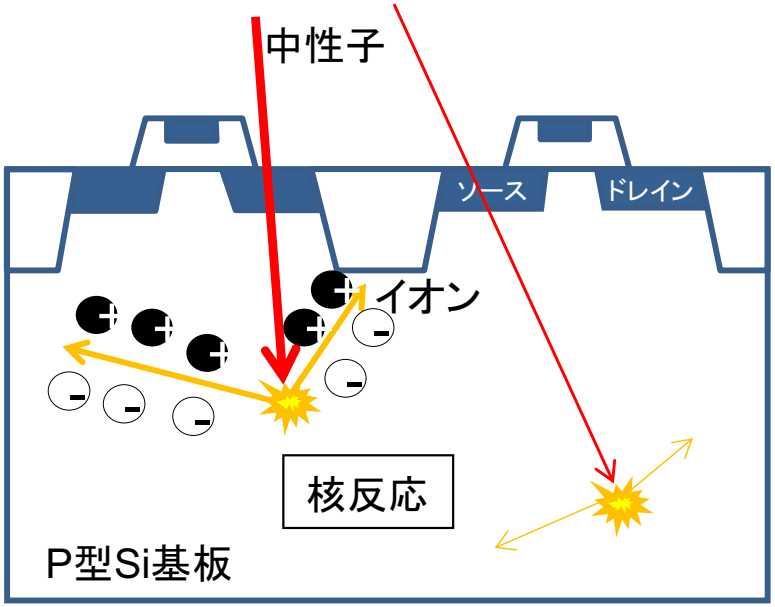
九州大学 安浦教授	九州大学 松永准教授	福岡大学 佐藤教授	豊橋技術科学大学 杉原准教授
馬場准教授	吉村助教 研究員2名 学生7名	林田助教 学生1名	学生5名
担当分野			
(1)ソフトエラー (3)価値と信用	(1)ソフトエラー (2)タイミングエラー (3)価値と信用	(1)ソフトエラー (2)タイミングエラー	(1)ソフトエラー

ソフトエラー

放射線によってLSIの動作不良が引き起こされる現象

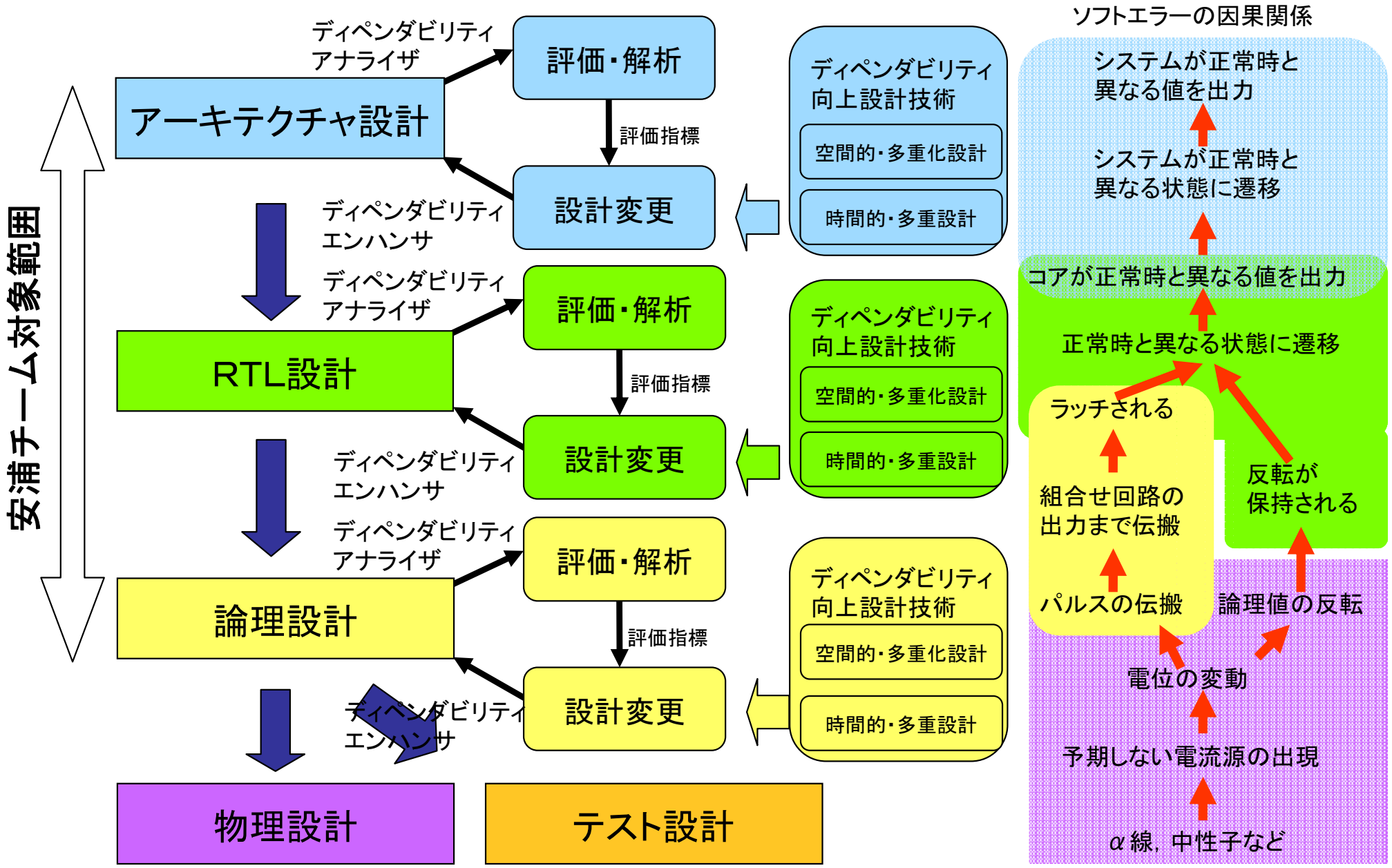
- 放射線による，LSI内の論理ゲートの異常パルスの発生やレジスタのビット反転（エラー）は，誤った最終出力（Failure）を発生させる可能性がある
- ハードエラー（LSIそのものの不良による恒久的な故障）とは異なる対策と評価手法が必要となる
- トランジスタの寸法縮小，低消費電力化による臨界電荷量の低下によるエラーの増加（2004年の報告:1GBのメモリで1～5日に一度の頻度で発生）

Failureの増加



回路の設計者は回路の信頼性を保つため，ソフトエラーから回路を守る手法を考える必要がある

設計ツールとフローの構築



評価ツールの重要性

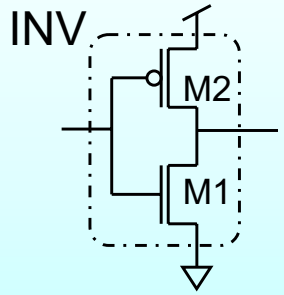
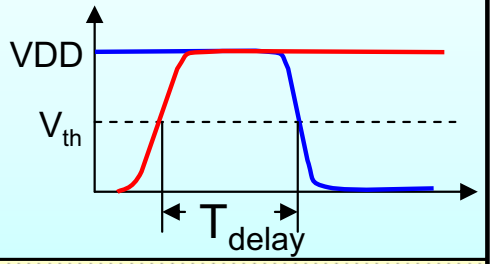
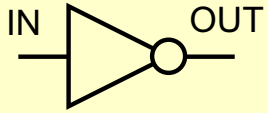
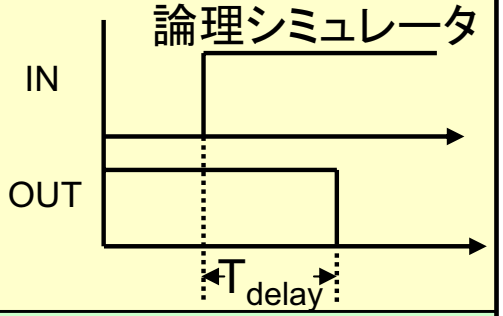
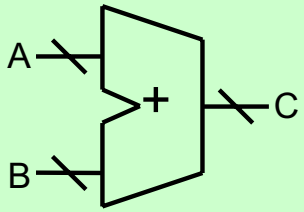
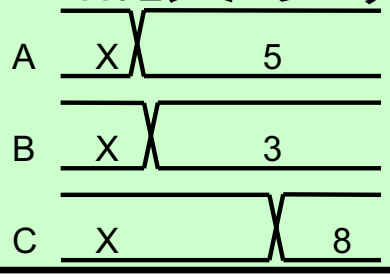
- LSIの設計では、試作⇒評価・解析⇒改良のサイクルを仮想的にEDAツールを用いて行う。
- さらに、ソフトエラーの場合、実際の現象を観測し、解析することは非常に難しい。

  ⇔遅延時間や電力との相違

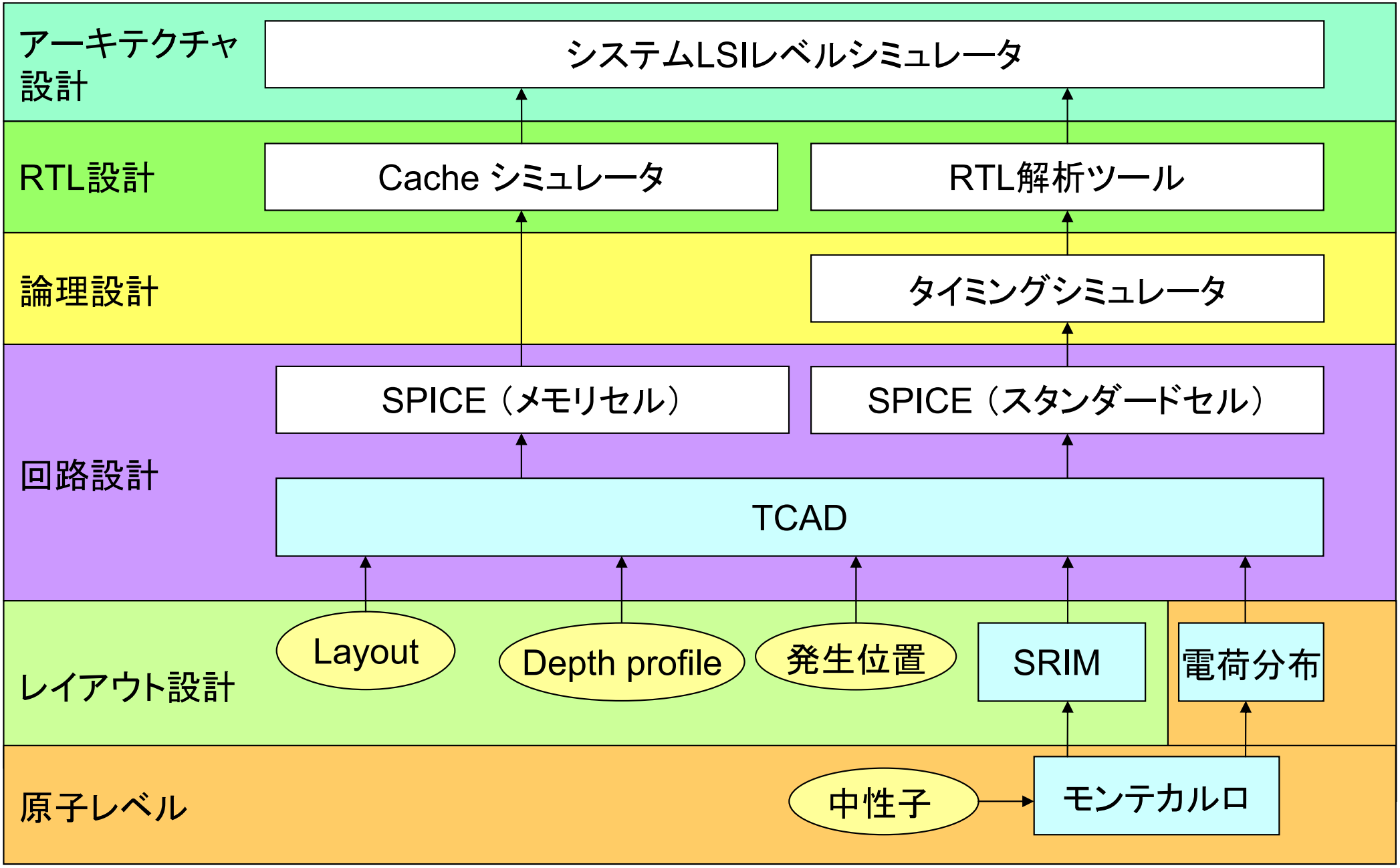
ソフトエラー発生のメカニズムを忠実にモデル化した、上位階層から下位階層まで整合性のとれた評価ツールの確立を目標とする。

階層設計に基づいた設計フロー

- 各階層における評価ツール（詳細レベルのBB化による高速化）
- 階層間を結ぶ合成・最適化ツール（抽象レベルでの最適化）
- 階層設計のフローとの整合と評価の効率化（タイミング、電力）

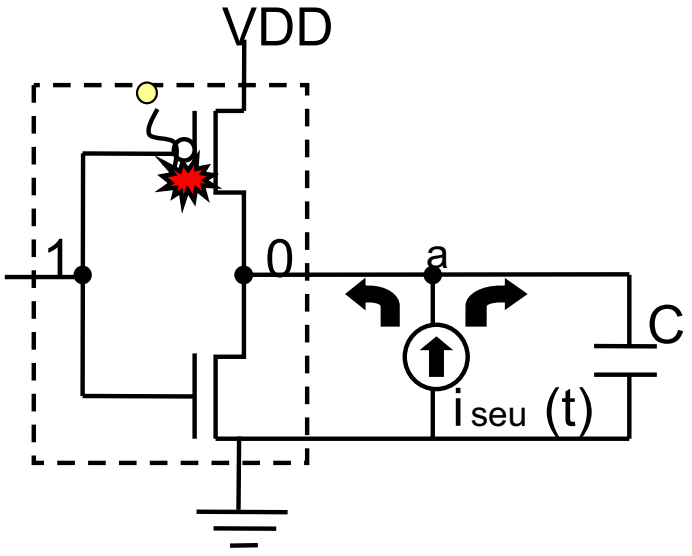
設計の抽象度	評価ツール	高速化
回路レベル 	回路シミュレータ 	x 10,000～
論理レベル 	論理シミュレータ 	x 10～100
RTレベル 	RTLシミュレータ 	

ツールチェイン


アーキテクチャ設計
システムLSIレベルシミュレータ
RTL設計
Cache シミュレータ
RTL解析ツール
論理設計
タイミングシミュレータ
回路設計
SPICE（メモリセル）
SPICE（スタンダードセル）
TCAD
レイアウト設計
Layout
Depth profile
発生位置
SRIM
電荷分布
原子レベル
中性子
モンテカルロ

解析手法の比較

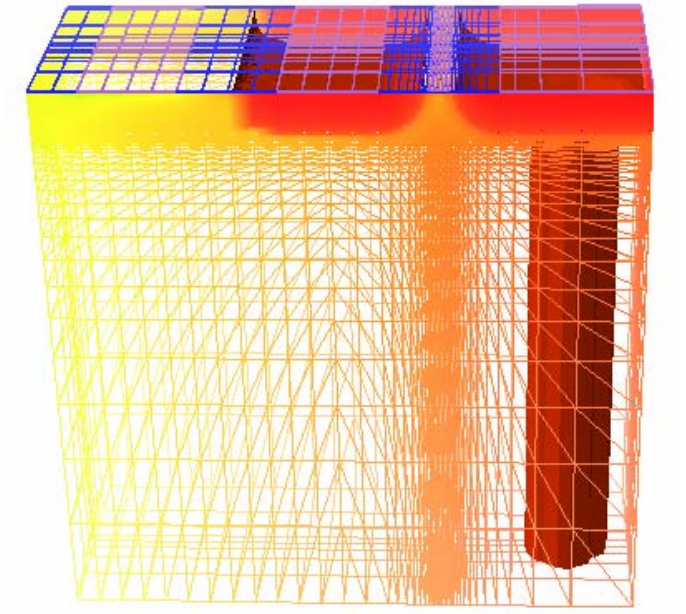
- 回路シミュレーション
  - 入力: ネットリスト、ノイズ電流源
  - 出力: 各ノードの電流値、電圧値
  - 実行時間: 短い（ex. 65nmインバータで1分程度）
- デバイスシミュレーション
  - 入力: トランジスタのデバイスモデル、電子・正孔の対（ehp）の分布
  - 出力: 各ノードの電流値、電圧値（→パルス幅）
  - 実行時間: 長い（ex. 65nmインバータで約3時間）



※電流が分かれば正確な解析が可能

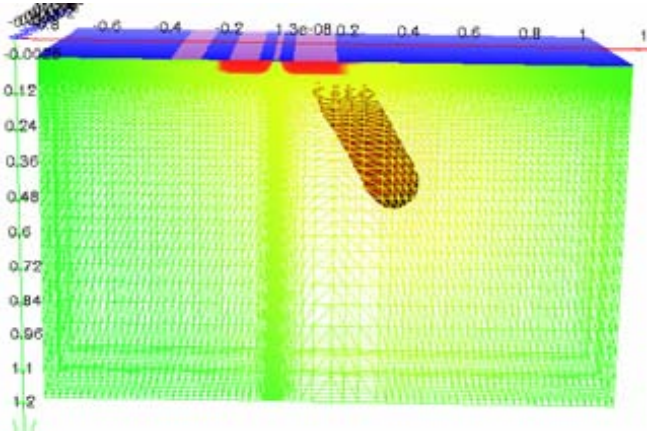
解析手法の比較

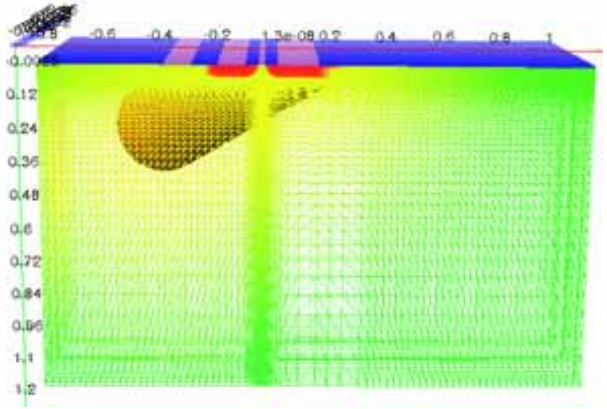
- 回路シミュレーション
  - 入力: ネットリスト、ノイズ電流源
  - 出力: 各ノードの電流値、電圧値
  - 実行時間: 短い（ex. 65nmインバータで1分程度）
- デバイスシミュレーション
  - 入力: トランジスタのデバイスモデル、電子・正孔の対（ehp）の分布
  - 出力:　各ノードの電流値、電圧値（→パルス幅）
  - 実行時間: 長い（ex. 65nmインバータで約3時間）



※ehpの分布を与えれば正確な解析が可能

TCADを用いたパルス幅の計算





- どの場所に、
- どれだけのエネルギーをもった

中性子が衝突するか、の情報は確率的にしか決められない。

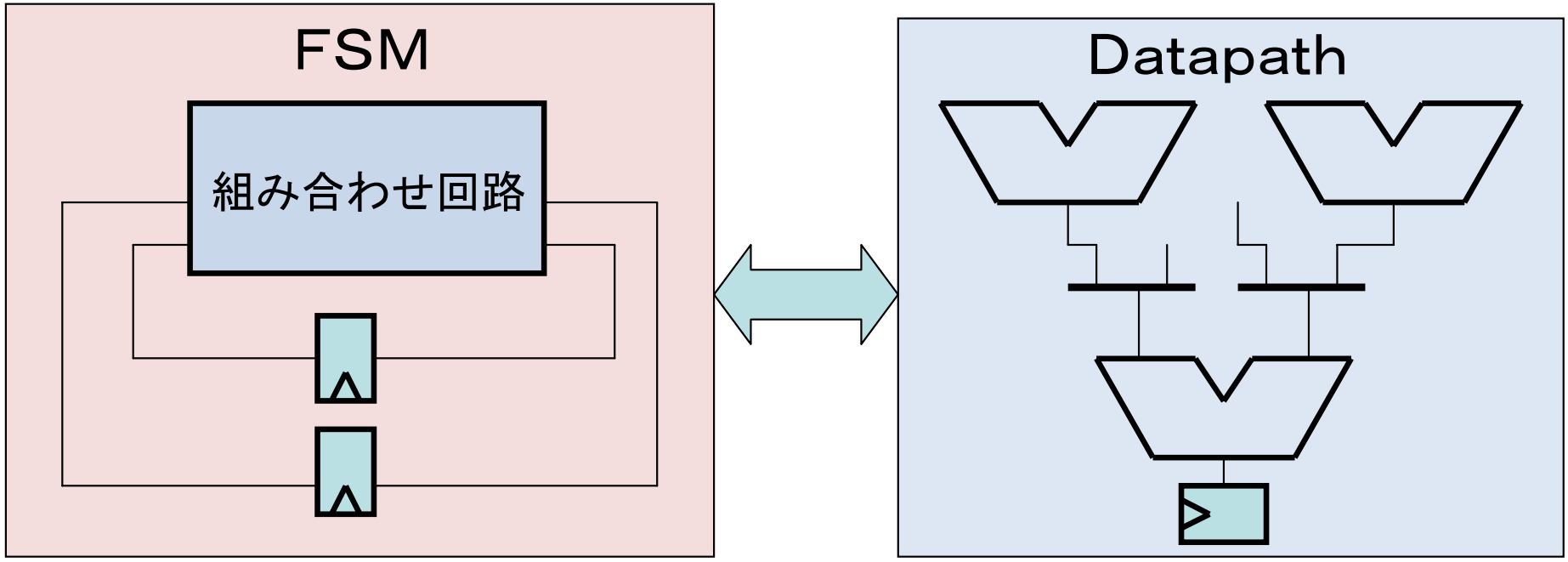
上記の情報を決めてもその結果、電子・正孔対がどのように分布するかは確率的にしか決まらない。

- 大まかに言って2重の確率統計処理が必要

ナイーブに数多くのランダムサンプリングシミュレーションを行っていたら埒があかない。
⇒統計科学的なアイデアが必要

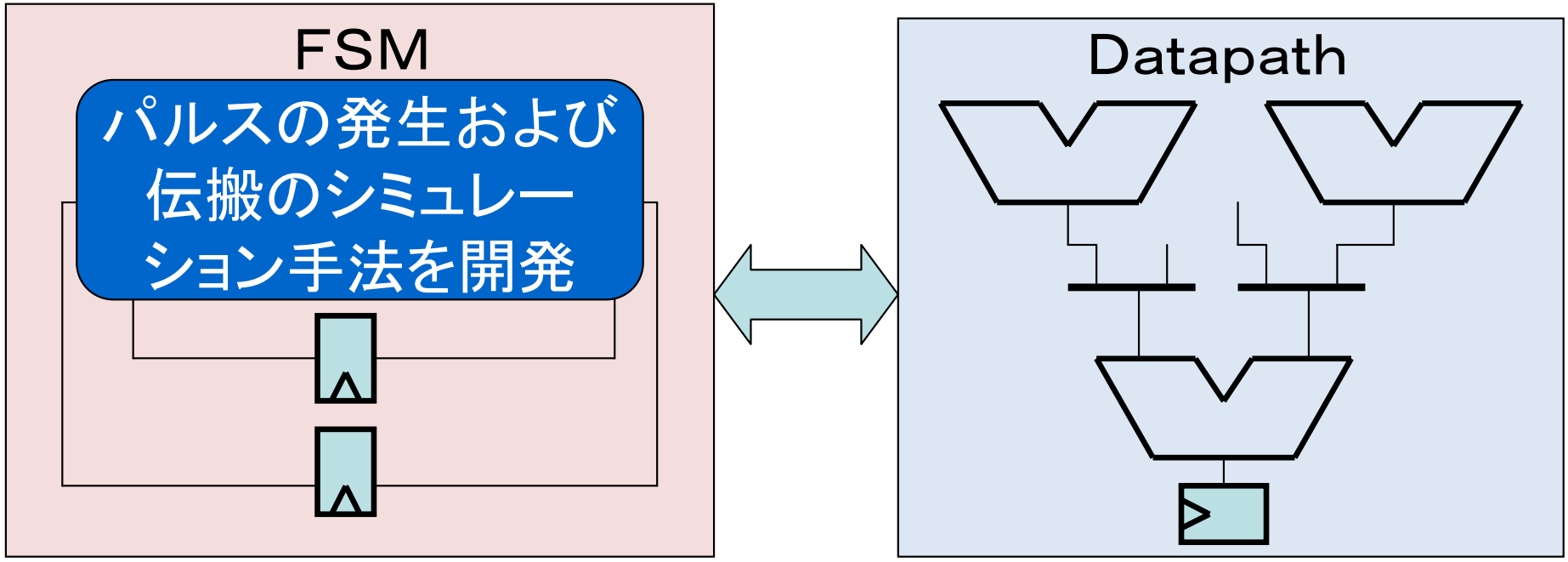
RTレベルの解析

- 回路を制御回路(FSM)とデータパスに分割
- FSM部は順序回路の解析手法を適用
- データパス部には束線をまとめて抽象化する手法を適用



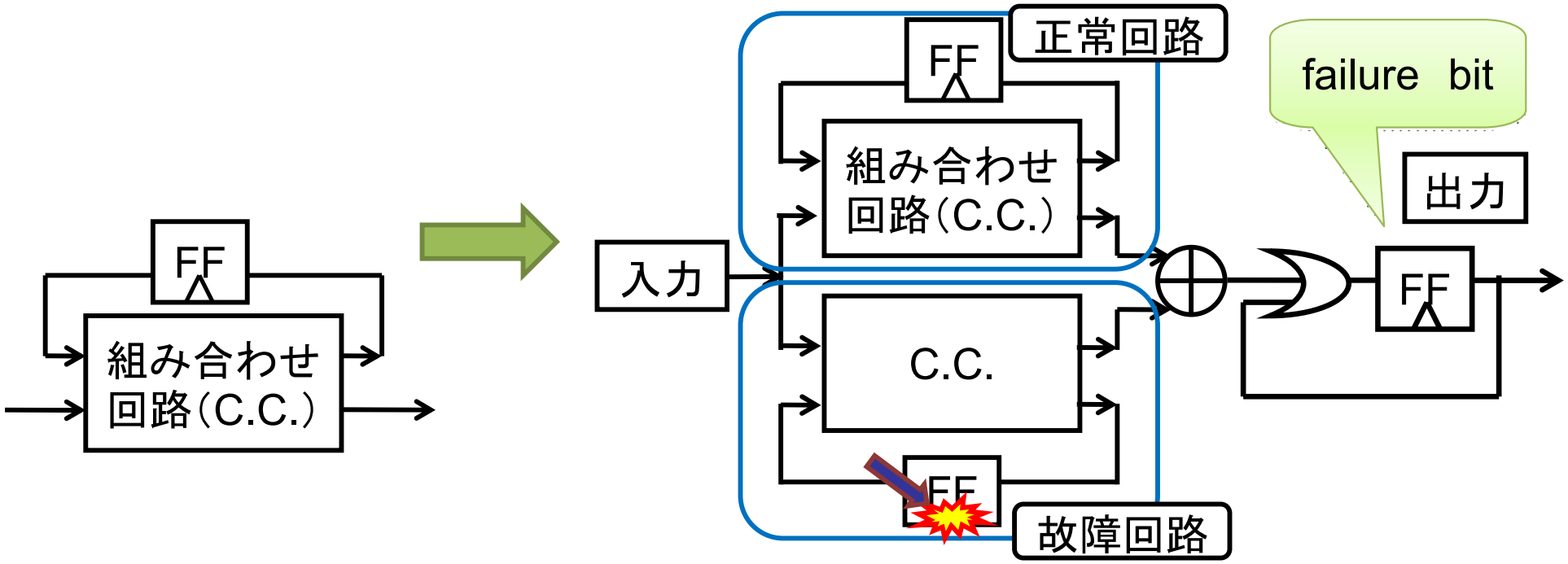
RTレベルの解析

- 回路を制御回路（FSM）とデータパスに分割
- FSM部は順序回路の解析手法を適用
- データパス部には束線をまとめて抽象化する手法を適用

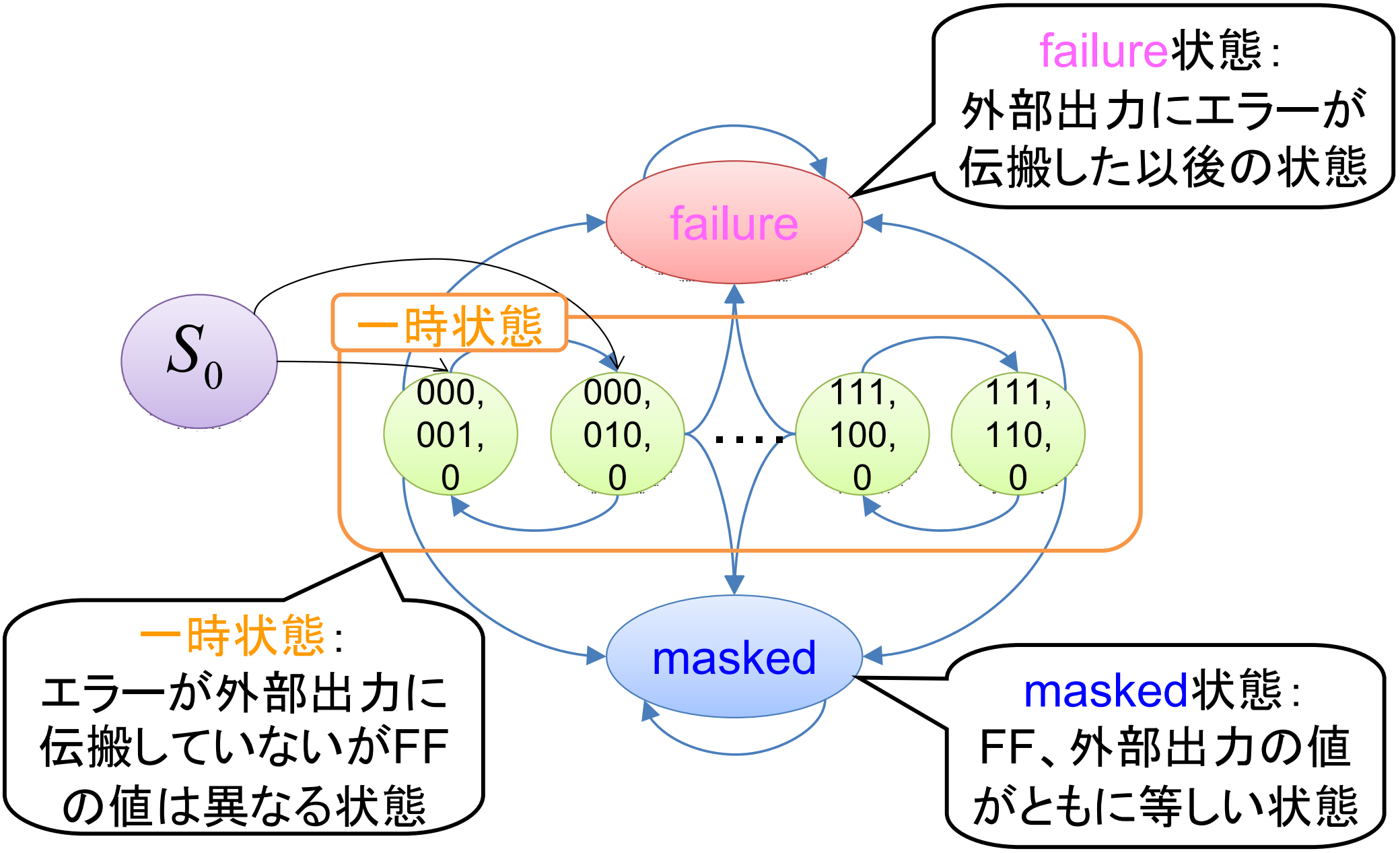


順序回路の回路対

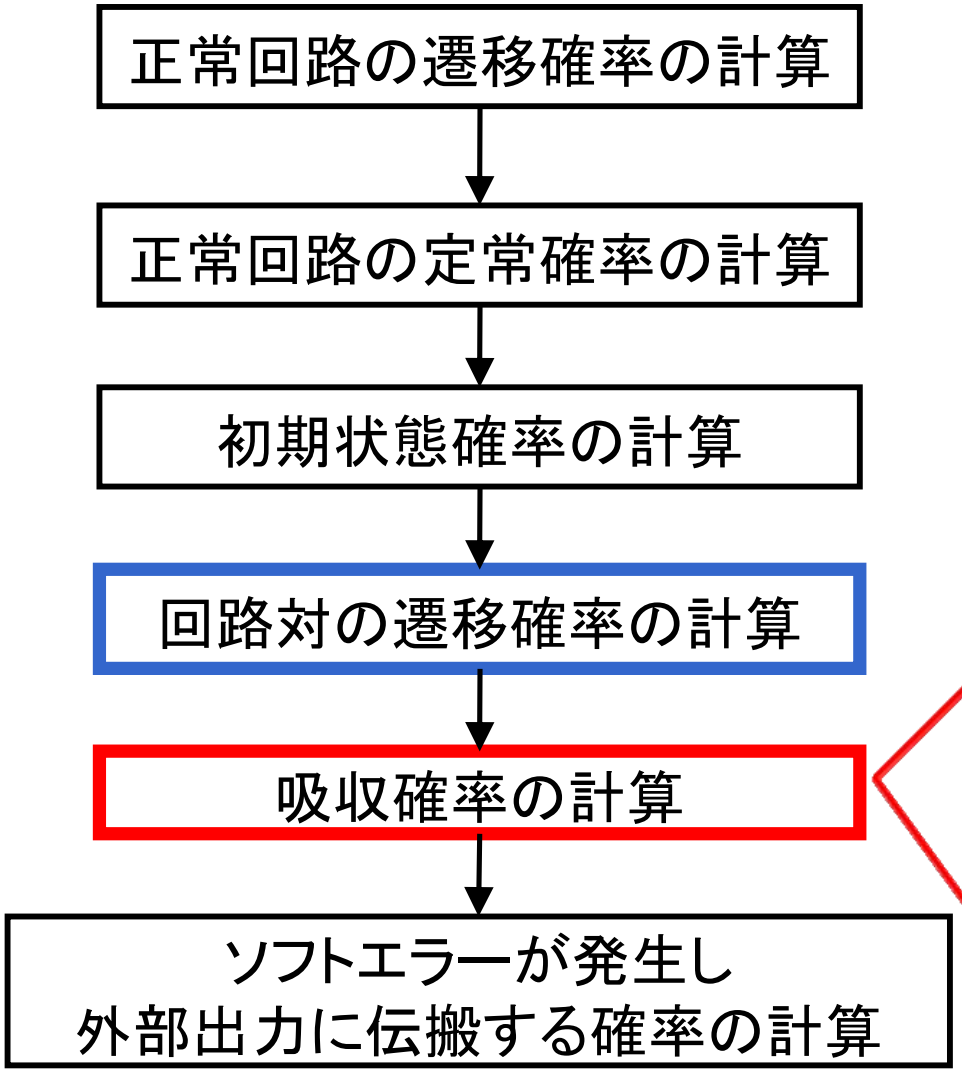
- 正常回路と故障回路(ソフトエラーの発生を仮定)の対
  - 外部出力値、FFの値を比較して回路の振る舞いを調査
    - failure bitは外部出力へのエラーの伝搬以後は1
  - 入力が確率的ならば回路対の状態はマルコフモデルとなる



回路対の状態


failure状態：
外部出力にエラーが
伝搬した以後の状態
failure
S_0
一時状態
000,
001,
0
000,
010,
0
....
111,
100,
0
111,
110,
0
masked
一時状態：
エラーが外部出力に
伝搬していないがFF
の値は異なる状態
masked状態：
FF、外部出力の値
がともに等しい状態

処理の枠組みと計算時間



連立方程式を解く

- ガウスの消去法では元数の3乗に比例した時間を要する
- 元数＝回路対の状態数
- 回路対の状態数 $\leqq 2^{2k}$

(k： 正常回路のFF数)

(つまり 2^{6k} に比例した時間)

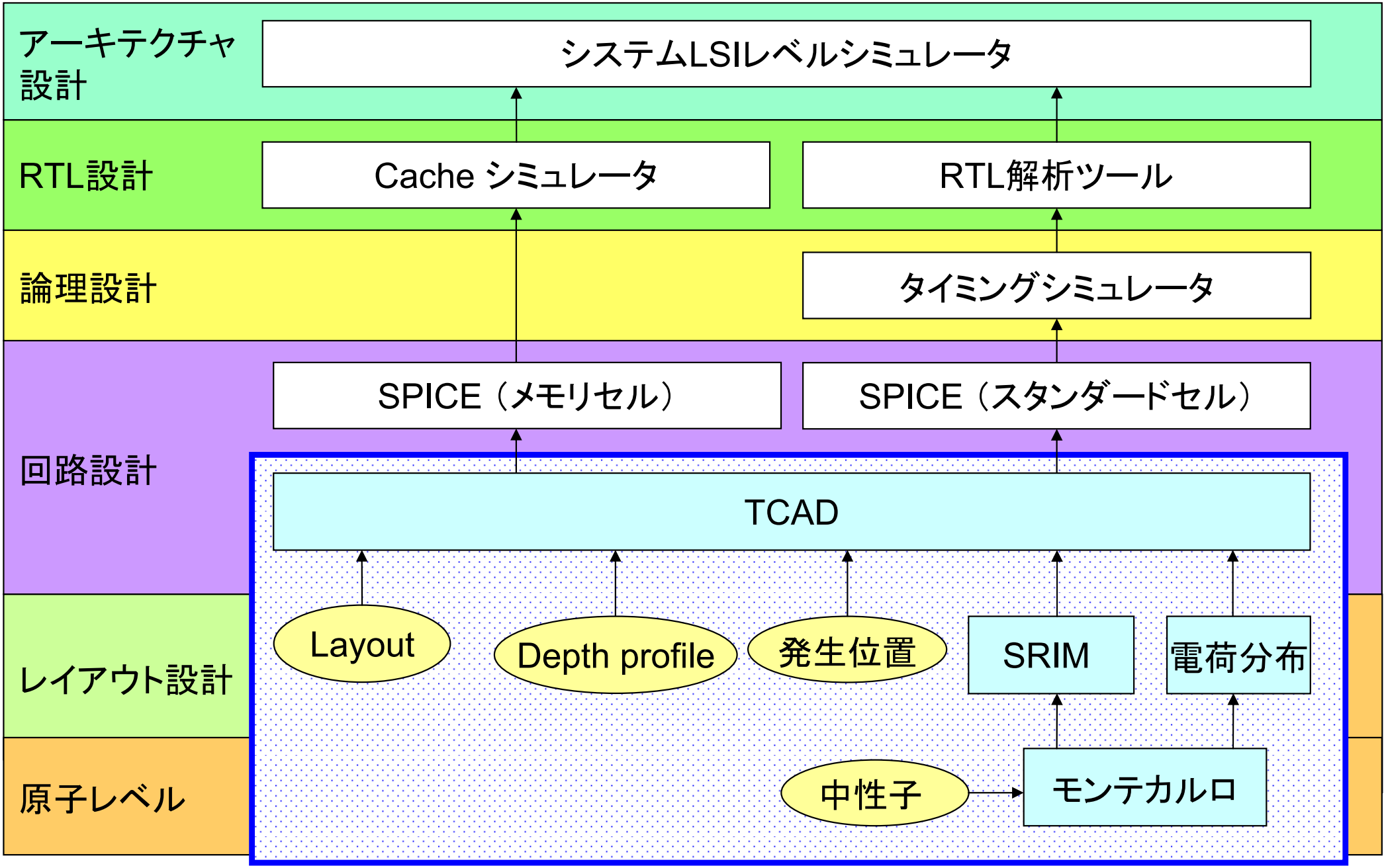
例：s382

- 全体 ：89398秒 (≒24時間)
- 吸収確率の計算：87834秒

厳密解法と近似解法

- 厳密解法：
  - フリップフロップ数21（回路対の状態数＝2^{42}）の回路の確率計算に約4万秒
- 近似解法：
  - Kクロックサイクル以内に正しい状態に戻らない場合には外部出力にエラーが到達したとみなす手法では850秒
  - 多くの場合、誤差は極めて小さい（1%以下）
- 今後の課題：より大規模（FF数：1,000～10,000）に適用可能な近似手法の開発

ツールチェイン


アーキテクチャ設計
システムLSIレベルシミュレータ
RTL設計
Cache シミュレータ
RTL解析ツール
論理設計
タイミングシミュレータ
SPICE（メモリセル）
SPICE（スタンダードセル）
回路設計
TCAD
Layout
Depth profile
発生位置
SRIM
電荷分布
レイアウト設計
中性子
モンテカルロ
原子レベル

タイミングエラーの検出・回復技術（九大G・福岡大G）

- タイミングエラーの原因
  - ばらつき（プロセス，電圧，温度，入力）
  - 経年劣化（NBTI，PBTI）
- 検出機構
  - Razor（発生後に発見）
  - カナリア（発生前に予報）
- 回復
  - 周波数降下
  - 電源電圧昇圧

プロセッサ以外のLSI一般にも適用したい⇒カナリアを選択

これまでは原因を特定せず，タイミングエラーの検出と回復技術を検討してきた
↓
今回は特定の原因（NBTI）にフォーカスする

NBTI

- pMOSで閾値電圧$|V_{th}|$が増大
- ストレスが無くなると，回復する特徴がある

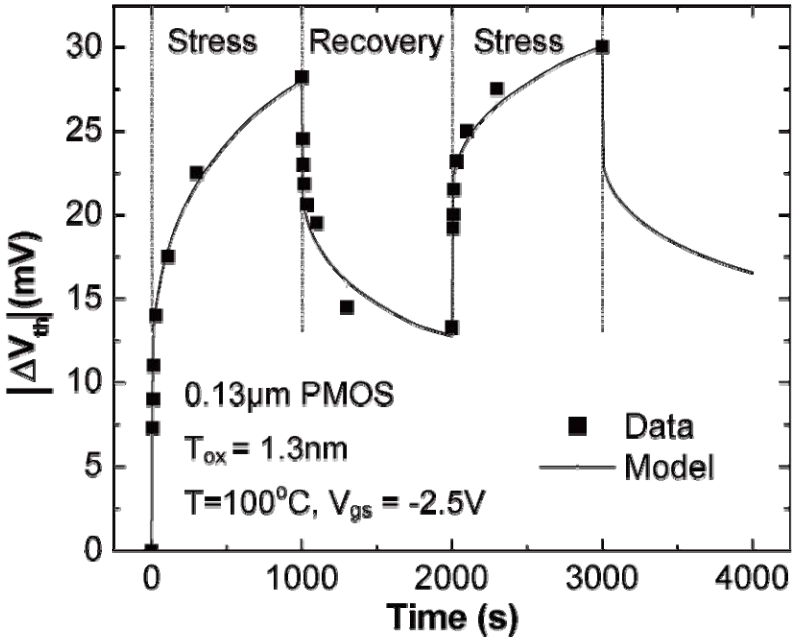
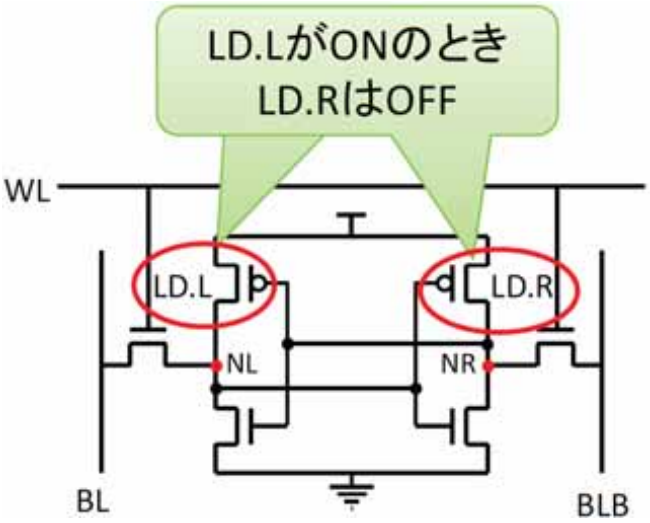


Figure 4. ΔV_{th} during dynamic NBTI [14].

- SRAMセルでのNBTI

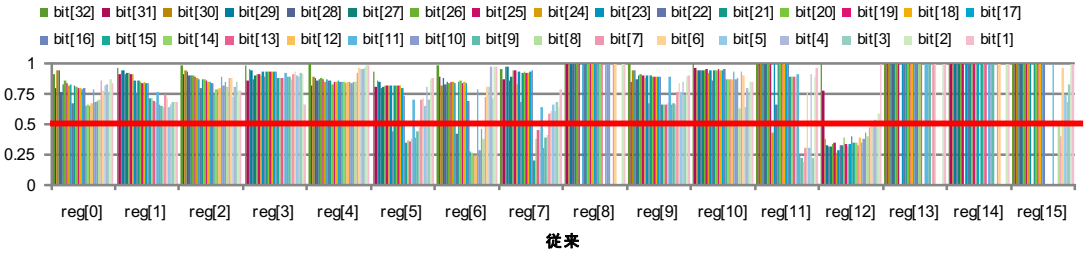


- 必ずどちらかのLDでストレス
  - $|V_{th}|$増大⇒SNM悪化
- SRAM使用のブロック
  - レジスタファイル，キャッシュ

[14] R. Vattikonda, et al. “Modeling and Minimization of PMOS NBTI Effect for Robust Nanometer Design”, DAC, 2006.

レジスタファイルでのNBTI対策

- SNMの悪化⇒動作不良
  - 組合せ回路とは別の対策が必要
  - 検出・回復⇒予防技術
- 各セルの0/1が時間的に均等になるように対策
  - 回復効果の利用
- モード切り替え
  - 書き込み時に反転
  - 読み出し時に反転書込み



偏り大⇒ストレス大

ISQED2010で発表済み．キャッシュへの応用をASQED2010で発表予定．ジャーナル投稿準備中

シグネチャ検査技術

- 1980年代に活発に研究された技術。
- ソフトエラーやNBTIに起因する信号線誤りを検出する技術。
- 一部の信号線が誤る前提。
- アーキテクチャレベル、あるいはソフトウェアレベルのように、繰り返し処理が存在するレベルで適用可能な技術。

Control Flow Checking

コントロールフローエラーの有無を確認すること

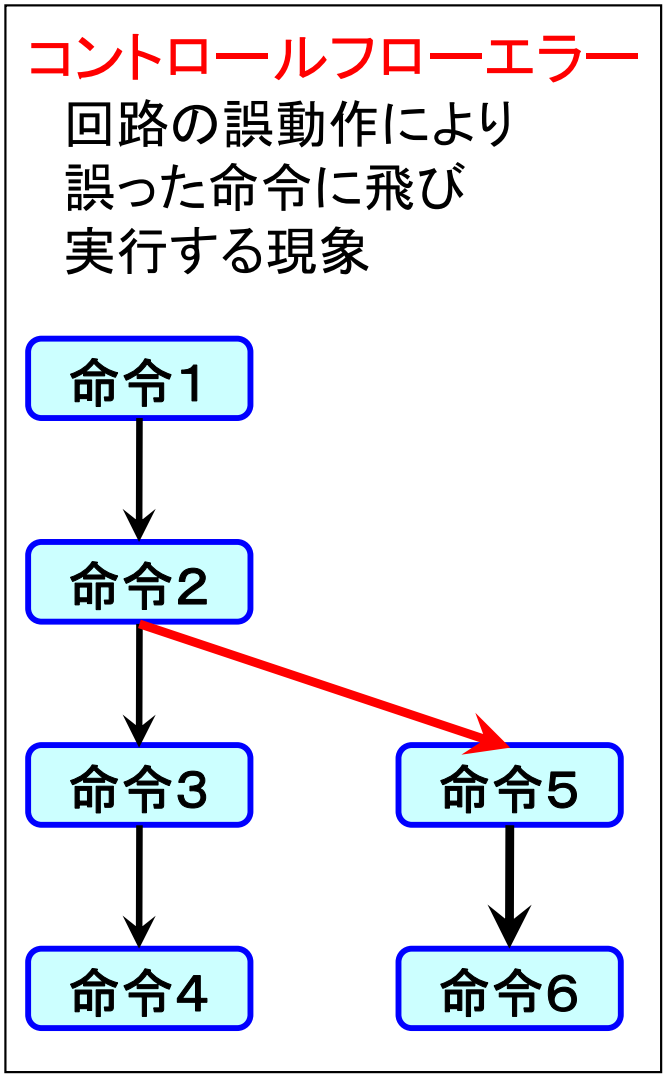
信号値系列をある値（シグネチャ）に対応付け、シグネチャとその期待値を比較することでコントロールフローエラーの有無を確認する。

$$S_i = f(S_{i-1}, W_{i-1})$$

f :シグネチャ関数

S_i :時刻iでのシグネチャレジスタの値

W_i :時刻iでの信号値



信号値系列からのシグネチャの生成

命令1 $S_1 = f(S_0, W_0)$

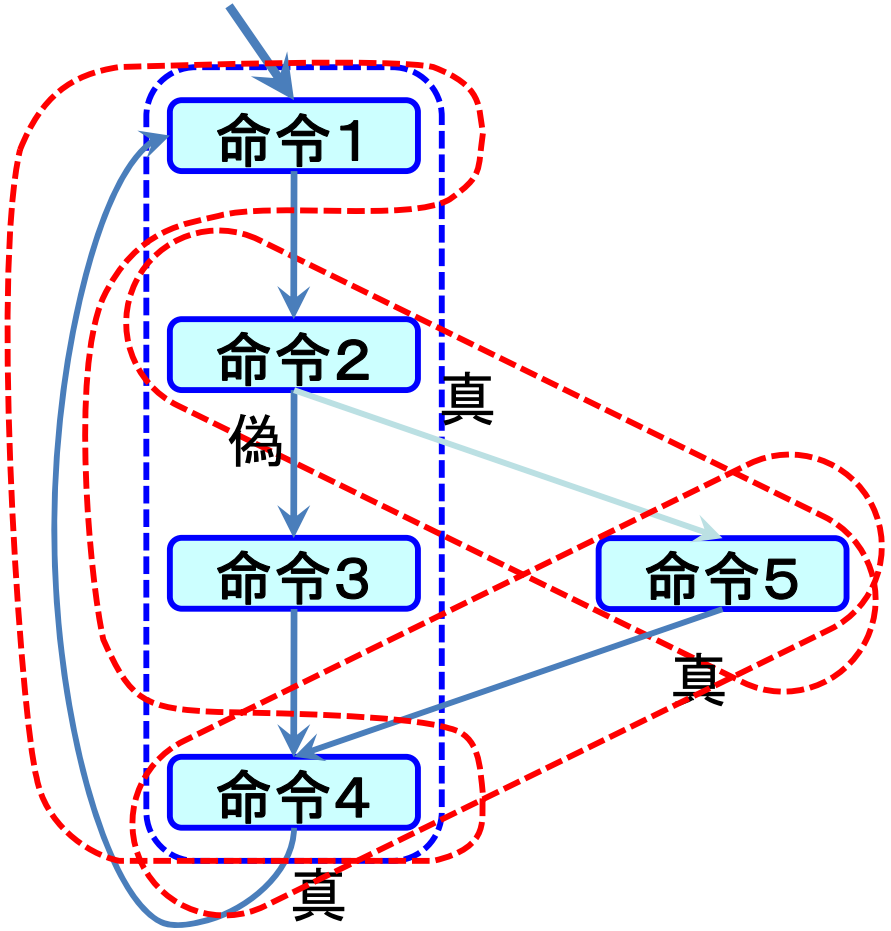
命令2 $S_2 = f(S_1, W_1)$

命令3 $S_3 = f(S_2, W_2)$

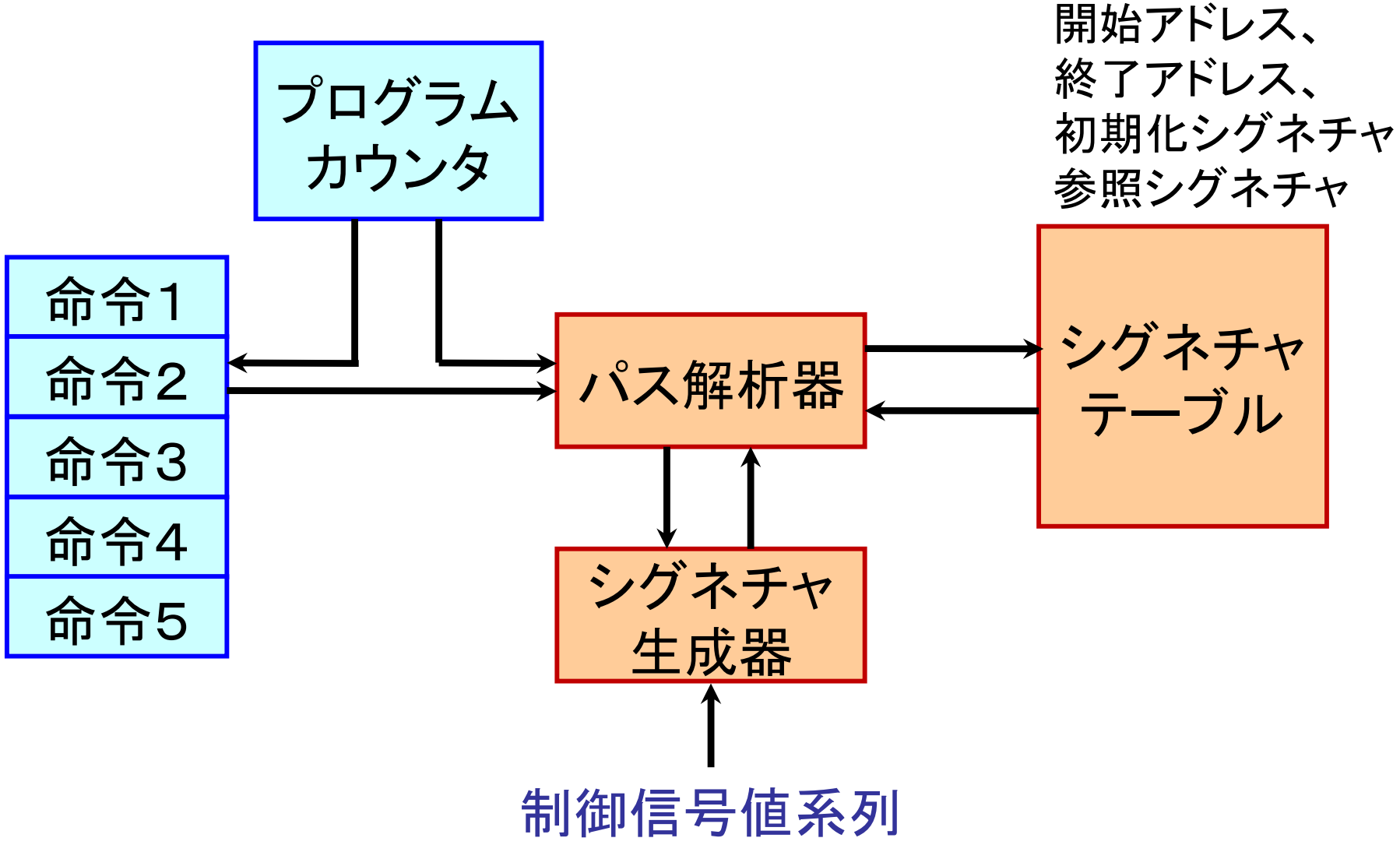
命令4 $S_4 = f(S_3, W_3)$

初期化シグネチャ: S_0
参照シグネチャ:
　シグネチャの期待値
実行時シグネチャ:
　実行時に生成するシグネチャ

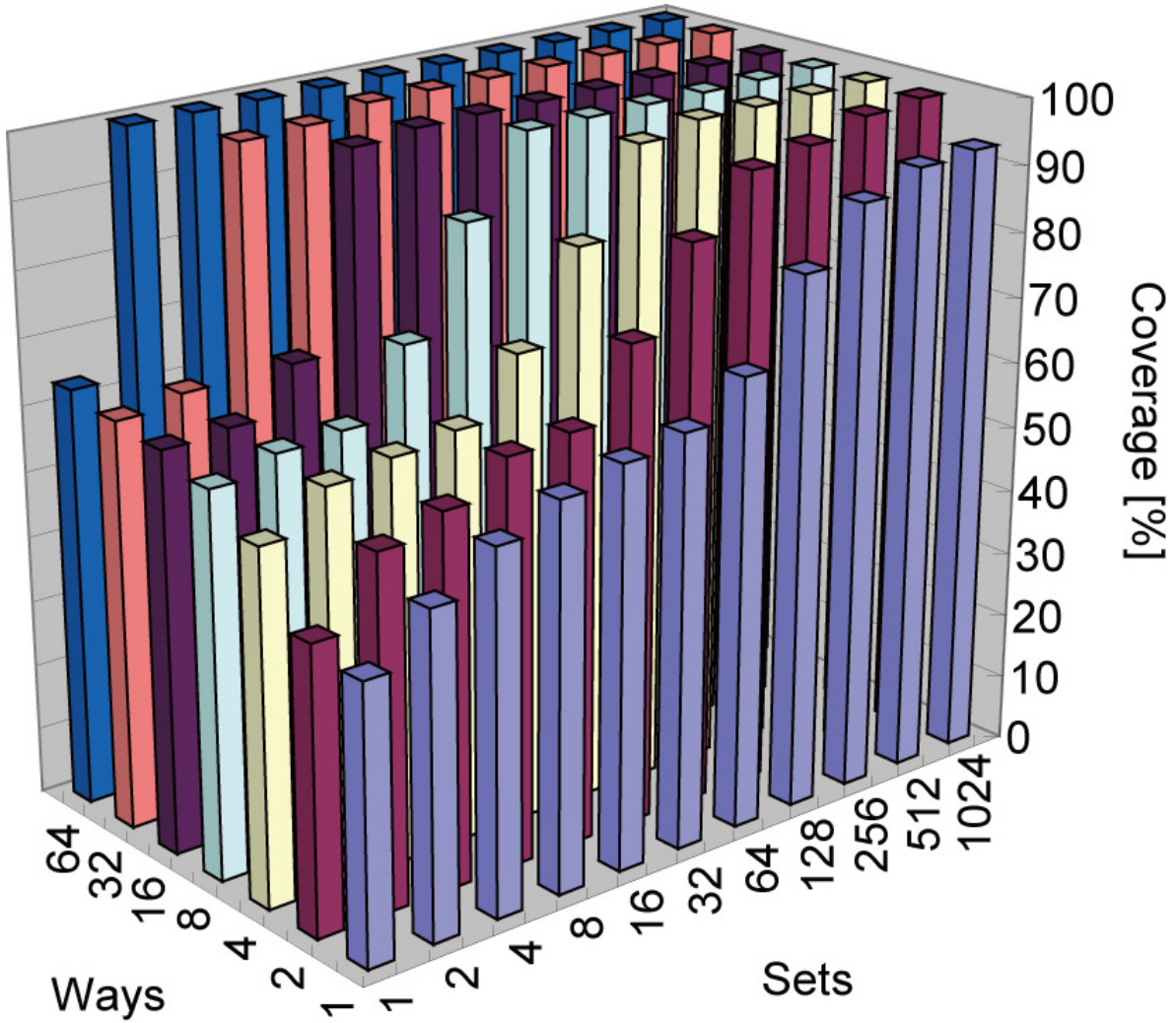
パス:シグネチャ生成・比較の
対象とする命令系列



動的CFC技術


プログラム
カウンタ
命令1
命令2
命令3
命令4
命令5
パス解析器
シグネチャ
生成器
開始アドレス、
終了アドレス、
初期化シグネチャ
参照シグネチャ
シグネチャ
テーブル
制御信号値系列

命令カバー率



$$\frac{\text{パスの実行命令数}}{\text{全実行命令数}}$$

- シグネチャテーブルのサイズを変更し、命令カバー率を調査。
- セットアソシアティブ
- LRU (least recently used)
- 41.8~100.0%命令カバー率

特許申請済、EUROMICRO DSDで発表予定。ジャーナル投稿準備中。