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

半導体設計の現場からのコメント

ルネサス エレクトロニクス株式会社
第一事業本部　システムコア開発統括部
CPU開発第一部　亀井達也

2013/03/16　Rev. 1

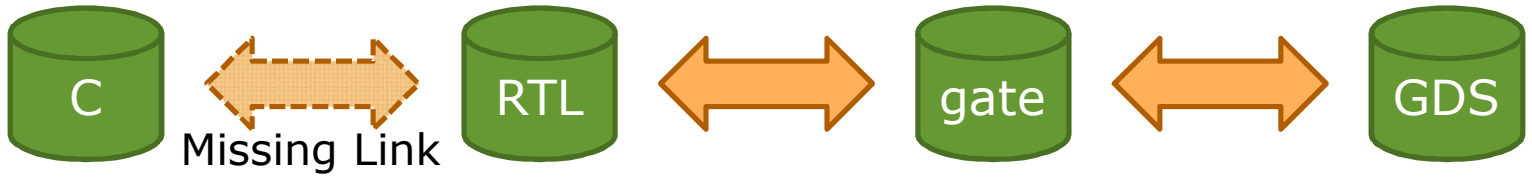
What’s Dependability?

- **設計品質**
  - **開発段階でバグを取る**
  - Si **になってから対処する**
- **動作時のエラー抑止**
  - **ソフト/ハードエラー耐性**
  - **ばらつき起因のタイミング問題**
- **社会的アカウンタビリティ**

どうやって設計品質を上げるか

- **形式的検証の活用**
  Simulation + Formal Verification **のベストミックス**
- **高抽象度の記述をマスタとしたい**
  "正しさ"の確認が容易
- **各抽象度間の等価性検証が必須**
  RTL **以降については既に実現**
  動作記述(C) vs RTL **がネック**
  これがうまくいかないと動作記述をマスタとして管理できない
- **実際よくある設計フローの例:**
  同一プロセス世代内では gate netlist **流用**＋ECO bug fix
  プロセス世代更新時に再合成
  →動作記述をマスタとした場合に、論理等価なものが再合成されるか？

本研究では、C vs C の範囲に留まる。
技術的に困難なのは重々承知だが、
あきらめずにここを追及してほしい。



そうはいってもバグは出る(Si になってからの対処が必要)

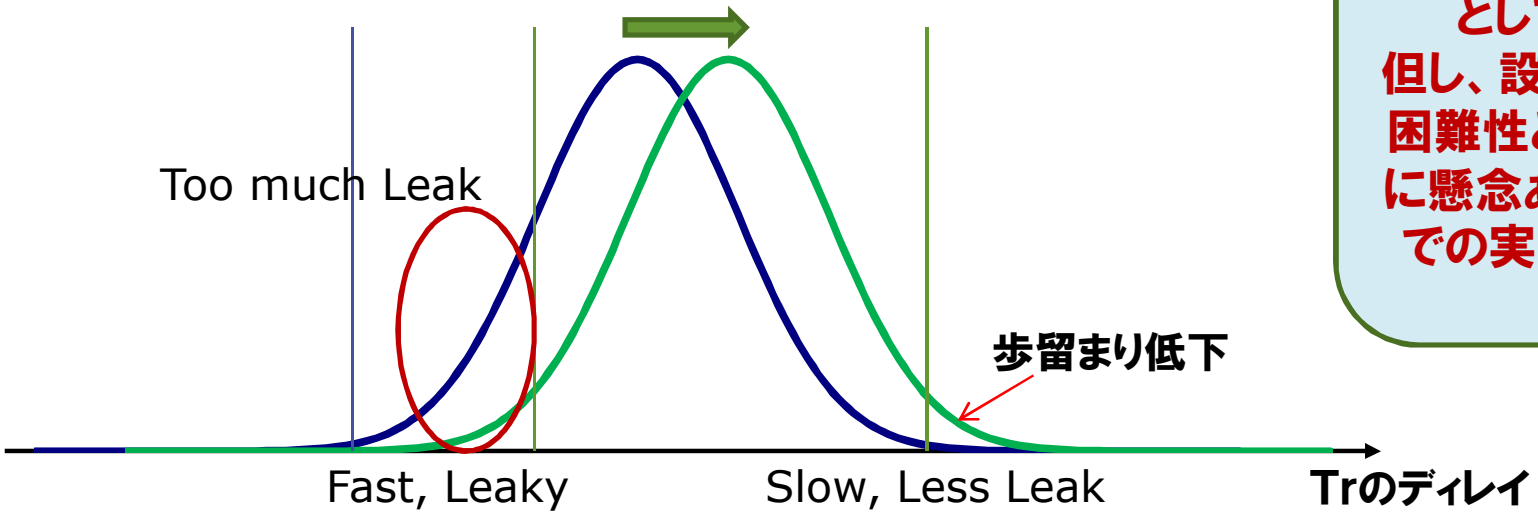
- IP/チップ/システムの並行開発
  きれいなウォーターフォールではビジネスにならない
- 問題の切り分けが難しい
  不安定動作の原因がどこにあるのかがわかりにくい
  HWのバグが非常に遅い段階で見つかることがままある
- 多段の対策が必要
  - SW Workaround を容易にする仕組み
    チキンビットの準備、形式的検証による
    WA の完全性の証明、etc.
  - パッチ可能なHW
    デジタル回路だけでなく、アナログ回路も
    STA 制約間違いによるタイミングバグなどもある
  - ECO 修正容易なチップ構成
    微細化、実装技術の高度化で、
    配線だけでは治らないケース多発
    ボーナスセル配置の最適化？

本研究の形式的手法を応用して、どういう修正用冗長回路を入れておくのが費用対効果がよいか、探索する方法は考えられないか

微細化により増大したばらつき起因のタイミング問題

- Setup ディレイにマージンを持たせるとリーク電流が大きくなりすぎる
  FF corner, Typical 共
  設計ターゲットウインドウを絞らざるを得ない→歩留まりが犠牲になる
- クロックの On-Chip Variation
  大量の Hold バッファによる無駄電力
  最悪 Setup と Hold 両方を満たせない場合も
- 動作時に一定確率でエラーが起きても、
  破綻しない設計が必要(Setup, Hold 両方)

本研究の耐タイミング・フォールト方式は、これを実現する方式として有望。
但し、設計時のSTA困難性とHold対策に懸念あり。実設計での実証に期待。



社会的アカウンタビリティ

本チームに限らない一般論ですが

- 技術的にディペンダブルであるだけではだめ
- なぜディペンダブルであるのかを、非専門家にわかるように（少なくともわかった気にさせるように）説明できないといけない
- 研究段階から説明性の確保が重要
- ステークホルダ間の役割/責任分担の合意/明確化がないと、ビジネスとして回せない

ディペンダブルにするためのコスト、責任ばかり増えると儲からない

ディペンダブルを“Value”として、儲けられる仕組みが必要

まとめ

- No Silver Bullet for Dependability
  設計/検証/製造/テスト/動作時、すべての段階に工夫が必要
- **形式的検証手法の更なる活用**
  適用可能箇所は色々あるはず
- **(ある意味での)発想の転換が必要**
  バグはあるもの、エラーは起きるもの、という前提に立ったシステム構築
  コンピュータアーキテクチャ的発想をリカバリに活用
  分岐予測、投機実行、トランザクション、etc.
- **“ディペンダブル”で儲かる仕組みを一緒に考えましょう**

ルネサス エレクトロニクス株式会社