修士論文

タイミング制約を緩和するクロッキング方式

A clocking scheme with relaxed timing constraints

平成22年2月9日提出

指導教員
五島正裕 准教授

東京大学大学院 情報理工学系研究科
電子情報学専攻

48-086405 喜多貴信

概要

半導体プロセスの微細化に伴って，回路遅延のばらつきの増加が，回路設計における大きな問題となりつつある．これは，トランジスタや配線のサイズが原子のサイズに近づくためであり，原理的に避けることができない．ばらつきが増大していくと，従来のワースト・ケースに基づいた設計手法は悲観的になりすぎる．今後の半導体産業の発展には，ばらつき対策技術が重要になる．

本研究では，動的タイミング・フォールト検出・回復技術の一つ Razor の考え方を更に推し進める．Razor では信号がサイクル・タイムを超えて遅れた場合は，すぐに検出されて回復処理を行ってしまう．提案手法では，動的タイム・ボローイングを利用することにより，信号がサイクル・タイムを超えることを許し，遅延が一定以上蓄積するまでは検出・回復を行わない．また提案手法は，入力依存の遅延ばらつきも含めて，統計的ワーストケースを見積もり，動作周波数を決定する．これによってばらつきを吸収して，その個体のその時の動作環境における実際の遅延に基づいた動作を実現することができる．結果的に従来よりも高クロック・低電圧な動作を可能とする．

目 次

図 目 次

表 目 次

第1章　はじめに

半導体プロセスの微細化に伴って，回路遅延の**ばらつき**の増加が，回路設計における大きな問題となりつつある．これは，トランジスタや配線のサイズが原子のサイズに近づくためであり，原理的に避けることができない．

ばらつきが増大していくと，従来のワースト・ケースに基づいた設計手法は悲観的になりすぎる．**図 1.1** は，トランジスタの速度の分布を模式的に表したものである．微細化したプロセスでは，微細化前のプロセスより，平均値は向上している一方で，ばらつきの増大により，分散が大きくなっている．そのため，製品歩留まりを一定とすると，出荷される LSI のトランジスタの遅延のワースト値は，平均値ほどには向上しない．**図 1.2** は，横軸に LSI の世代をとり，縦軸に遅延の平均値とワースト値をプロットしたものである．微細化が進むにつれて，ばらつきの増大により，平均値とワースト値の差は広がっていく．ワースト・ケース設計では，同図のように，LSI の動作速度が向上しなくなってしまうことも考えられる．

ばらつき対策手法　ばらつきの増大に対処するため，ワースト・ケースではなく，より実際の遅延に基づいた動作を実現する手法が提案されている．

設計段階では，**統計的静的タイミング解析** (Statistical Static Timing Analysis: **SSTA,** [1]) が用いられつつある．SSTA では，遅延のばらつきを統計的に扱うことによって，ワースト・ケースほど悲観的ではない遅延見積もりを得ることができる．SSTA については，3.1 節で詳しく述べる．

LSI テスト時にスピード・グレード選定を行うことが一般的に行われている．このことも，ワースト・ケースではなく，その個体の実際の遅延に基づいて動作させる手法の一つと理解することができる．

動的タイミング・フォールト検出・回復　タイミング・フォールトを検出し，回復する動的な手法も，ばらつき対策として有効である．

?? 章で詳しく説明するが，**Razor**[3, 2] は，**図 2.2** に示す FF を用いて，**タイミング・フォールト**を動的に検出し，回復する．

Razor のような動的タイミング･フォールト検出･回復手法に，**DVFS** (Dynamic Voltage and Frequency Scaling) を組み合わせると，その個体のその時の動作環境における実際の遅延に基づいた動作を実現することができる．以下，本稿では，DVFS における電源電圧と動作周波数の組を**動作点**と呼ぶことにする．動

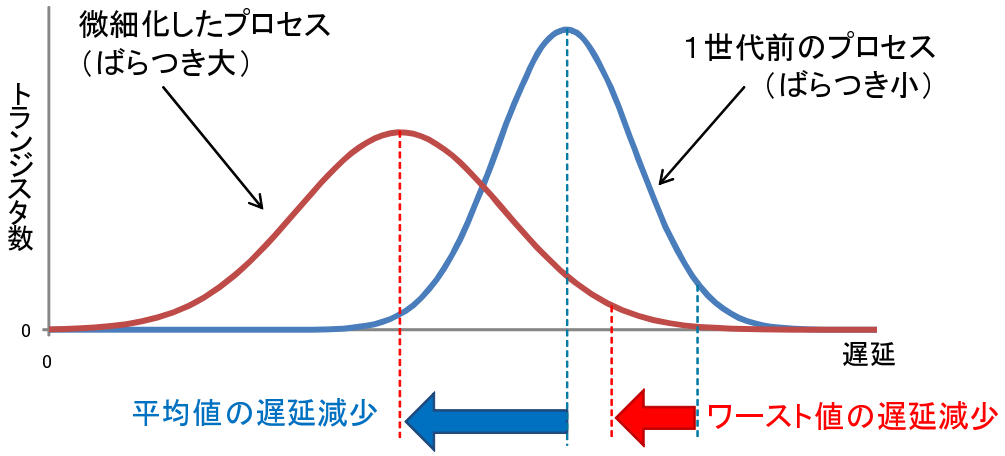


図 1.1: ばらつきが引き起こす，微細化の効果の減少

作点を厳しくする，すなわち，電源電圧を下げる，または，動作周波数を上げて行くと，回路はいずれタイミング・フォールトを生じ，Razor FF によって検出される．検出直前の動作点が，その個体のその時の動作環境における実際の遅延に応じた動作点である．タイミング・フォールトが頻発しないように動作点を調整すれば，その個体のその時の動作環境の実際の遅延に基づいた動作を実現することができる．

回復手法は検出ロジックと独立して組み合わせることができ，[3] では予備の FF から復旧する方法，[2] ではタイミング・フォールトに影響された命令を再実行する方法，[8] ではコアをリセットする方法，が提案されている．カナリア FF[9, 10] は，タイミング・フォールトを発生前に予測することで，Razor に比べてやや保守的な変動幅になるが回復ロジックなどの追加ハードウェアを削減している．

本稿の提案　本稿では，Razor の考え方を更に推し進めたもので，**動的タイム・ボローイング**を許す方式である．後で詳しく述べるが，2 相ラッチ方式などで可能になるタイム・ボローイングは，設計時における静的なものであり，本提案の動作における動的タイム・ボローイングとは異なる．

あるステージの遅延がクロック周期を超えることを，遅延の「借金」と考えると，従来の方式，前述した，動的にタイミング・フォールトを検出・回復する方式，提案方式は，以下のように説明することができる：

従来の方式　従来のワースト・ケースに基づくクロッキング方式は，言わば，借金の概念がない方式である．借金が生じたこと自体が分からず，生じれば silent error となる．そのため，絶対に借金が生じないように設計する必要がある．

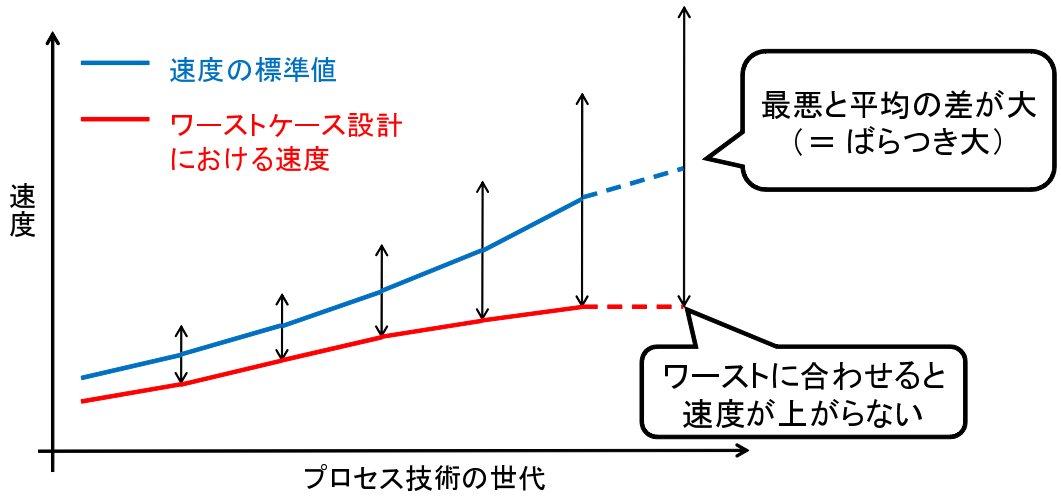


図 1.2: 悲観的すぎるワースト・ケース設計

動的タイミング・フォールト検出・回復　動的にタイミング・フォールトを検出・回復する方式は，借金は許されないが検出はできる方式と言える．借金が生じれば，検出され，破綻処理を行い，動作を再開する．

提案方式　提案する手法は，借金を許す方式であると言える．提案方式では，あるステージで生じた借金を次のステージで「返済」することができる．連続して赤字が続き資金繰りに行き詰まった時にはじめて破綻処理を行い，動作を再開する．

提案方式はまた，入力によって回路遅延が異なるという**入力ばらつき**を最大限に利用している．

これらのことにより，提案手法では，Razor の最大 2 倍の動作速度を実現することができる．

以下，2 章では，背景的知識を説明する．3 章では，入力ばらつきについて加算器の例を用いて説明した後に，SSTA の統計的ワーストに対して，入力ばらつきを考慮したワースト見積もりが向上することを示す．4 章では，既存のクロッキング方式とばらつき耐性について議論する．特に 2 相ラッチは提案手法のベースとなっている方式である．5 章で提案手法の構成・動作を示す．

第2章　背景

半導体プロセスが微細化するにつれて，ばらつきの問題が深刻化してきている．従来のワースト・ケース設計ではこの問題に対処することは難しくなりつつあり，今後の半導体産業の発展には，ばらつき対策技術が重要になる．

2.1　ばらつきの要因

ばらつきは要因ごとに，下記の 5 種類 **(PVTAI)** に分類できる：

Process　製造ばらつき．

- 露光精度の低下，エッチングや平坦化などの工作精度の低下による各加工寸法のばらつき
- 不均一な不純物密度によるトランジスタの閾値電圧のばらつき

Voltage　電源電圧のばらつき．

- IR ドロップによる電圧降下
- 雑音
- チップへの電圧供給源におけるばらつき

Temperature　温度のばらつき．

- 動作頻度が高い回路と低い回路の温度ばらつき
- 環境温度や，ファンなどの冷却装置に起因する温度ばらつき

Aging　経年劣化によるばらつき．

- トランジスタの損耗
- Electromigration

Input　入力ばらつき．
入力（計算内容）の違いによる遅延ばらつき

これら 5 つのうち，製造ばらつきは製造時に，その他 4 つは動作中に生じる．前者は静的，後者は動的なばらつきであると言える．

これらの要因により，トランジスタの閾値電圧や ON/OFF 抵抗値，配線の抵抗値がばらつき，回路の遅延と消費電流量がばらつくことになる．また，その結果，動作時の消費電流量のばらつきがさらに増大する負のフィードバックが存在する．これらは，トランジスタや配線のサイズが原子のサイズに近いたことによるものであるので，悪化を完全に抑えることは原理的に難しい．

2.2 悲観的すぎるワースト・ケース設計

従来の最悪値に基づいた設計/製造は，以下のように進められる：

1. 回路遅延が設計に対して変動する要因には，上に挙げた 5 つばらつき (PVTAI) がある．設計時には，まず，それぞれについて許容範囲を設定する．例えば，電源電圧は 1.10V ± 5%，温度は 0°C〜85°C，そして，製造ばらつきについては,「ロジックは 3σ，メモリは 6σ 以内」といった具合である．

2. その上で，すべての動作条件 —— 特に最悪条件において，正しく動作するように設計する．

3. それでも，製造ばらつきが許容範囲外であるようなチップが製造されることは原理的に避けられない．そのようなチップは出荷検査によって取り除かれる．

したがって，出荷後には，PVTAI のばらつきが設定された許容範囲内に収まっている限り，タイミング・フォールトは発生しないことになる．

しかし，ばらつきが増大していくと，従来のワースト・ケースに基づいた設計手法は悲観的になりすぎる．図 1.1 に示した例のように，微細化したプロセスでは，微細化前のプロセスより，平均値は向上している一方で，ばらつきの増大により，分散が大きくなっている．そのため，製品歩留まりを一定とすると，出荷される LSI のトランジスタの遅延のワースト値は，平均値ほどには向上しない．微細化が進むにつれて，ばらつきの増大により，平均値とワースト値の差は広がっていく．ワースト・ケース設計では，図 1.2 のように，LSI の動作速度が向上しなくなってしまうことも考えられる．

これまでは，微細化によって LSI の動作速度が向上したため，製造企業は新プロセスを導入するたびに，付加価値の高い（高クロック/低電圧など）商品を製造・販売して利益を得ることができた．しかし LSI の動作速度を向上させることができなくなると，既存の製品は次第に市場での販売価格が落ちてしまう．チップ面積減少によってチップ単価が下がるものの，付加価値の高い製品

を販売できないと，製造企業は新プロセスへの投資（工場や製造機器などのための費用）を回収できなくなってしまう．これによって，プロセスの微細化という方向性自体が，LSI 業界から消滅してしまう可能性すらある．

2.3 ばらつき対策手法

ばらつきが増大していくと，前述した最悪値に基づく設計，製造手法（ワースト・ケース設計）は悲観的になりすぎることになる．そのため，より実際に近い遅延に基づいて，回路を動作させる手法が提案されている．

静的な手法 （P：実際的, VTAI：ワースト）

- 統計的静的タイミング解析 (SSTA: Statistical Static Timing Analysis)
- テスト時のスピード・グレード選別
- テスト時のスキュー調整

動的な手法 （P + VTAI：実際的）

- 動的タイミング・フォールト検出・回復，予報

静的な手法は，製造ばらつきに対して実際の遅延に基づいた見積もりを行うが，動的なばらつき（VTAI）に対してはワーストに基づいた見積もりを行う．後述する SSTA では，遅延のばらつきを統計的に扱うことによって，ワースト・ケースほど悲観的ではない遅延見積もりを得ることができる．LSI テスト時にスピード・グレード選定を行うことが一般的に行われている．このことも，ワースト・ケースではなく，その個体の実際の遅延に基づいて動作させる手法の一つと理解することができる．

タイミング・フォールトを検出し，回復する動的な手法も，ばらつき対策として有効である．動的な手法では，前述したような動作時に生じるばらつきについても，その個体の，そのときの動作環境における，実際の遅延に基づいた動作を行うことができる．提案手法も動的タイミング・フォールト検出・回復を利用している．

2.4 動的タイミング・フォールト検出・回復，予報

Razor[3, 2] は，図 2.2 に示す FF を用いて，**タイミング・フォールト**を動的に検出し，回復する．

Razor のような動的タイミング・フォールト検出・回復手法に，**DVFS** (Dynamic Voltage and Frequency Scaling) を組み合わせると，その個体のその時の動作環境における実際の遅延に基づいた動作を実現することができる．

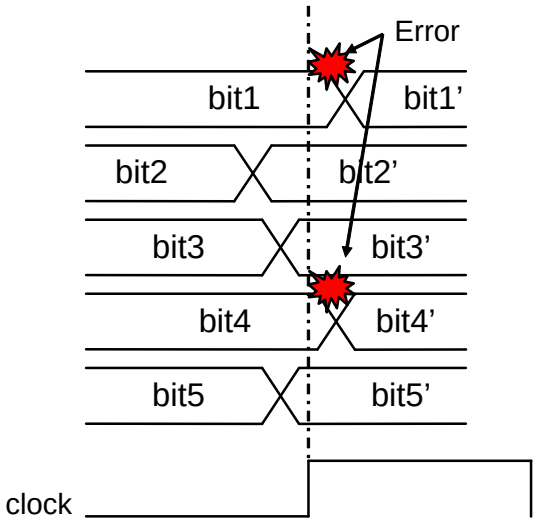


図 2.1: タイミング・フォールトの例

2.4.1 タイミング・フォールト

タイミング・フォールトとは，回路の遅延の増減により, 設計時の想定外の挙動を起こす過渡故障である. 同期回路における典型的なタイミング・フォールトは，遅延の増加によってロジックの出力がクロック期間内に安定せず，前のサイクルの出力がサンプリングされる事である．**図 2.1** に，そのような例を示す．この例では，bit1 と bit4 を生成するロジックの遅延が何らかの原因で増加し，前のサイクルの値が取り込まれている．

2.4.2 検出 (Razor1, 2)

図 2.2 は Razor の回路図とタイミング・チャートを示している．Razor は通常のフリップ・フロップ (main FF) に加えて，もう一つ shadow latch がセットになっている．shadow latch には，main FF へのクロックより位相が遅れたクロックが供給されている．

タイミング・チャート上で動作を示す．図は，縦軸に時間，横軸にロジックをとっている．4 章で詳細するが，単相フリップ・フロップでは，クロックの立ち上がりの一瞬にデータがサンプリングされ，それが 1 サイクル以内に次段のフリップ・フロップに到達しなくてはならない．しかし，Razor ではこの制約に違反して信号が 1 サイクル以上遅れた場合でもタイミング・フォールトとして検出できる．

main FF は通常のフリップ・フロップと同じく，クロックの立ち上がりでデータをサンプリングする．shadow latch には，main FF よりも遅れたクロックが

供給されているため，サンプリング・タイミングが遅い．図中では位相が 180 度遅れている場合を仮定しており（つまり反転クロックが shadow latch に供給されている），このとき shadow latch は main FF に 0.5 サイクル遅れてデータをサンプリングする．

図中赤線は，ワースト・ケースの遅延を示している．(図中で傾きが急であることは，信号の遅延が大きいことを意味する.）この信号は遅延 1 サイクルを超えており，main FF のサンプリング・タイミングに間に合っていない．しかし，shadow latch には正しくサンプリングされる．このとき，main FF がサンプリングした値と shadow latch のサンプリングした値を XOR で比較することにより，タイミング・フォールトを検出することができる．タイミング・フォールトが検出されると，後述する回復機構によって回復される．

時間をお金に例えれば，通常のフリップ・フロップはお金が絶対に不足しないように設計しなければならず，Razor は仮にお金が足りなくなっても（タイミング・フォールトが発生しても）破綻・再生処理を行うことによって，正常な動作を再開できるような手法であると言える．

ただし，shadow latch が正しい値をサンプリングするためには，ロジックの最短パスを通った信号が 2 回目のサンプリング点よりも遅く到達するように設計しなくてはならない．さもないと，最短パスを通った信号が前フェーズの遅れた信号と混ざってしまい，2 回目のサンプリング点で shadow latch が正しい値をサンプリングできなくなってしまう．この場合では，2 回目のサンプリングは 0.5 サイクル遅れて行われるから，ロジックの最小遅延が 0.5 サイクル以上になるように，ショート・パスに遅延素子を挿入しなくてはならない．

DVFS 制御との組み合わせ　DVFS (Dynamic Voltage and Frequency Scaling, **図 2.3)** は，LSI の電圧や周波数を動作環境に合わせて動的に変動させる技術である．本稿では，DVFS における電源電圧と動作周波数の組を**動作点**と呼ぶことにする．例えば Intel Speed Step では，設計時に用意されたいくつかの動作点の中から，ドライバが OS のポリシーに従い動作点を選択・設定する．

動的タイミング・フォールト検出・回復を用いない DVFS では，周波数や電源電圧の変動幅は,（SSTA）ワースト・ケースに基づいて決定する．動作点を厳しくする，すなわち，電源電圧を下げる，または，動作周波数を上げて行くと，回路はいずれタイミング・フォールトを生じる．タイミング・フォールトが発生すると，回路の正常動作を保証できない．結果的に，いかなる個体・いかなる動作環境においても絶対にタイミング・フォールトが発生しないような，保守的な変動幅で DVFS を行わざるをえない．ばらつきが大きくなると，保守的な動作のためのマージンも大きくせざるを得なくなってしまう．

Razor のような動的タイミング・フォールト検出・回復手法を用いれば，タイミング・フォールトが発生しても回路の正常動作を保証できるようになる．動作点を厳しくしていくと，回路はいずれタイミング・フォールトを生じ，Razor

FF によって検出される．検出直前の動作点が，その個体のその時の動作環境における実際の遅延に応じた動作点である．タイミング・フォールトが頻発しないように動作点を調整すれば，その個体のその時の動作環境の実際の遅延に基づいた動作を実現することができる．

タイミング・チャート上における緑線は，その個体のその時の動作環境における実際のワースト遅延を示している．これが 1 回目のサンプリング点と一致するように動作点が設定される．このとき，信号がタイミング・フォールトを発生することなく通ることができる範囲は，最小遅延が 0.5 以上という制約も含め，0.5〜1 サイクルとなり図中水色の領域のようになる．

2.4.3 リセットによる回復

回復手法は検出ロジックと独立して組み合わせることができる．[3] では予備の FF から復旧する方法，[2] ではタイミング・フォールトに影響された命令を再実行する方法，[8] ではコアをリセットする方法，が提案されている．本節では，特にリセットにより回復する手法（**図 2.4**）について述べる．

リセットによる回復手法は，タイミング・フォールトの影響を受けた間違った結果を，アーキテクチャ状態（PC，論理レジスタ・ファイル，データ・キャッシュ）へコミットしないように制御する．そして，タイミング・フォールトの影響を受けた可能性のあるロジックに対してリセットを書ける．その後，正しいアーキテクチャ状態を復元して動作を再開する．この手法の利点は，[2] などの回復手法と異なり，制御系パスで発生したタイミング・フォールトも含めて包括的に回復可能である，ということである．

Razor の検出した error 信号は，パイプラインに沿って伝搬される．error 信号は，必ず命令の実行結果のコミット前にコミットステージに到達するように設計される．（例えば，コミットの前に stabilize ステージを設けて，error 信号がコミットステージに到達するまで，コミットを遅らせることで実現できる．）

2.4.4 予報 (Canary FF, Replica path)

Razor のようにタイミング・フォールトが発生した時に，検出・回復するのではなく，タイミング・フォールトが発生しそうな兆候を検知してタイミング・フォールト発生を予報する．予報を元に，タイミング・フォールト発生前に動作点を緩くすることにより，回復機構などの追加ロジックを省略することができる．ただし，タイミング・フォールト発生前に確実に予報をしなくてはならないため，Razor に比べて保守的な範囲で動作点を変動させることになる．

図 ??にカナリア FF の回路構成とタイミング・チャートを示す．カナリア FF も通常のフリップ・フロップ (main FF) に加えて，もう一つ canary latch をセットに持っている．Razor と同じく，この二つのフリップ・フロップのサンプリ

ングした値を比較することにより，タイミング・フォールト予報を行う．Razor では shadow latch には遅延クロックを供給してサンプリング・タイミングをずらしていた．一方カナリア FF では，canary latch がサンプリングするタイミングは main FF と同じだが，canary FF へのデータ入力は遅延素子によって main FF よりも遅れている．

タイミング・チャート上で動作を説明する．図中赤線は，ワースト・ケースの遅延を示している．Razor ではこれが 1 サイクルを超えることができたが，カナリア FF では必ず 1 サイクル以内になるように設計しなくてはならない．例えば，canary latch へ挿入された遅延素子が信号を 0.3 サイクル遅延させるとする．クロックの立ち上がりにおいて main FF は必ず正しい値をサンプリングし，canary latch は 0.7 サイクルの時点において到達していた信号値をサンプリングする．この２つのサンプリング値を比較して異なっていたら，これをタイミング・フォールト発生の兆候とみなし，予報信号を発生する．予報信号は，DVFS 制御部へ伝搬されて，動作点を緩くするフィードバックをかける．しばらく予報が発生していなければ，少し動作点を厳しくし，予報が発生したら少し動作点を緩くする，という動的な制御を行うことにより，その個体のその時の動作環境における実際の遅延に基づいた動作を実現することができる．

カナリア FF には Razor にあったような最小遅延制約はない．その代わり，（この場合）0.7 サイクルより信号が遅れると予報を発してしまうから，信号は 0～0.7 サイクルの範囲で到達することを要求され，図中水色の領域となる．タイミング・チャート上における緑線は，その個体のその時の動作環境における実際のワースト遅延となる．

カナリア FF では個々のフリップ・フロップに細工をすることにより予報を行っていた．これに対し，ロジックの最長パスを模した Replica path[4] に対してのみカナリア FF と同様のタイミング・フォールト予報を行う手法もある．この手法では，カナリア FF よりもハードウェアを小さくできる．ただし，Replica path はロジック中のあらゆるクリティカル・パスよりも長い遅延となるように設計しなくてはならない．さもないと，Replica path で予報する前に，ロジック中のクリティカル・パスでタイミング・フォールトが発生してしまい，正常動作を保証できないからである．ばらつきが大きくなると，Replica path に大きなマージンを見込まなければならないため，動作点の変動幅が保守的になってしまう問題点がある．

お金の例えを用いれば，タイミング・フォールトを予報する手法は，お金（時間）が不足しそうになったら未然に対策する手法である，といえる．

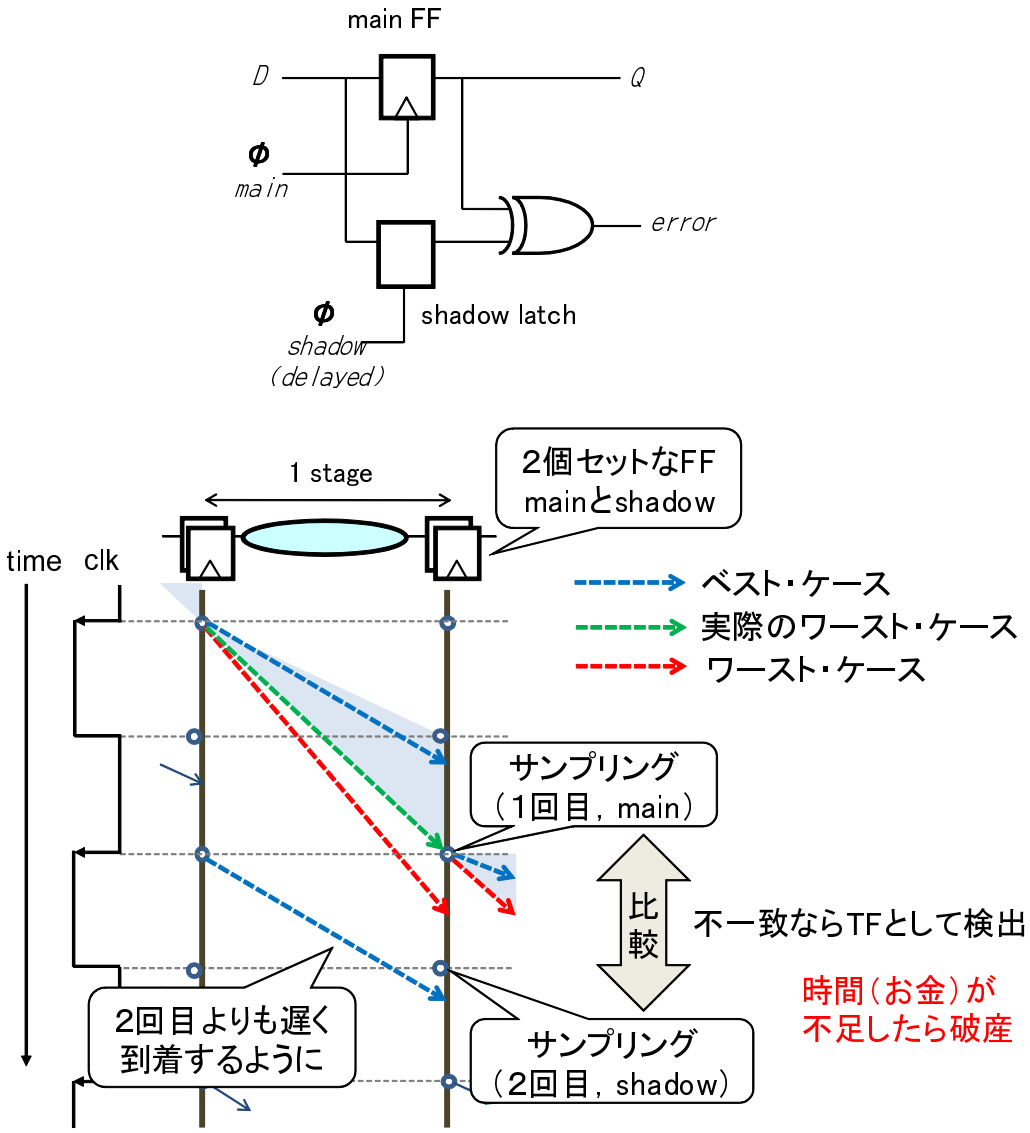


図 2.2: Razor の回路図（上）とタイミング・チャート（下）

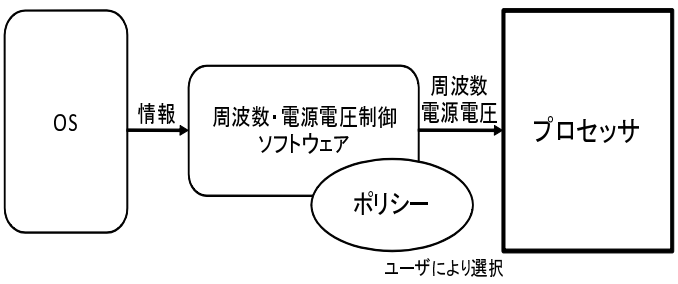


図 2.3: DVFS 制御

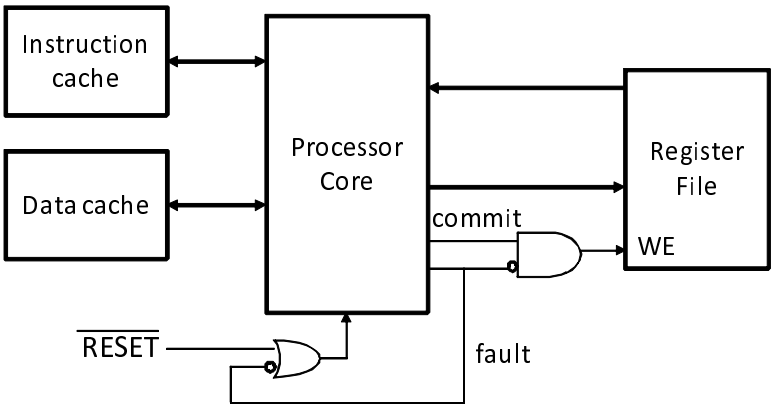


図 2.4: リセットによる回復手法

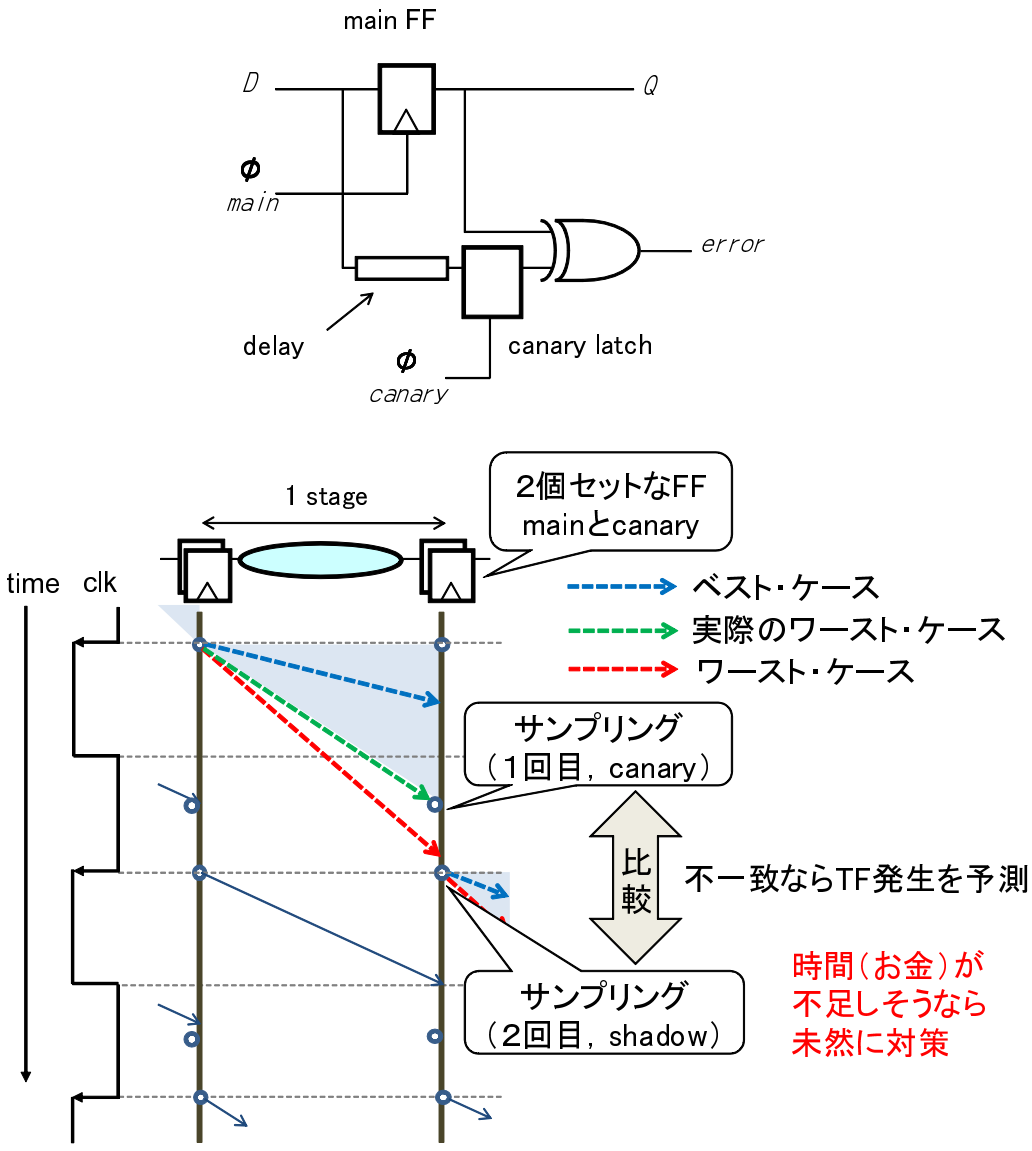


図 2.5: カナリア FF の回路図（上）とタイミング・チャート（下）

第3章　ロジックの遅延のばらつき

本章では，ロジックの遅延のばらつきについて述べる．

3.1　統計的静的遅延解析 (SSTA)

パス上のトランジスタ，配線の遅延がすべてワースト・ケースであるような確率は高くない．**統計的静的遅延解析** (Statistical Static Timing Analysis: SSTA) では，確率的にロジックの遅延を見積もる．

図 3.1 は，連続する 2 個のトランジスタからなるパスの遅延の確率分布を表している．各トランジスタの遅延は，平均が m_1 で，標準偏差 σ_1 の正規分布 $N(m_1, {\sigma_1}^2)$ に従うものとする．

各トランジスタの遅延の分布の $+6\sigma_1$ をカバーする遅延をワースト値とすると，ワースト値を満たさないトランジスタが製造される確率は（正規分布に従うとすると）100 万分の 3.4... となる．

図 3.1 上は，ワースト・ケース設計におけるワースト・ケース遅延見積もりを表している．ワースト・ケース設計では，各トランジスタの遅延のワースト値を単純に 2 つ足して，$2m_1 + 12\sigma_1$ をパスのワースト遅延とする．

図 3.1 下は，SSTA におけるワースト・ケース遅延見積もりを表している．SSTA では，まずパス全体の遅延の分布を求め，それに対して遅延見積もりを行う．前記の分布 $N(m_1, {\sigma_1}^2)$ を確率的に足しあわせて得られる分布は，正規分布 $N(m_2, {\sigma_2}^2)$，$m_2 = 2m_1$，$\sigma_2 = \sqrt{2}\sigma_1$ となる．

パスの遅延の分布の $+6\sigma_2$ をカバーする遅延をパスのワースト値とすると，ワースト値を満たさないパスが製造される確率は，同じく 100 万分の 3.4... となる．その場合ワースト値は，$m_2 + 6\sigma_2 = 2m_1 + 6\sqrt{2}\sigma_1$ と見積もることになる．そのため，SSTA におけるワースト値は，ワースト・ケース設計のそれより $(12 - 6\sqrt{2})\sigma_1$ だけ減少する．

このように，パスを構成するトランジスタが多いほど，SSTA によるワースト・ケース値の見積もりはワースト・ケース設計のそれより改善されることになる．

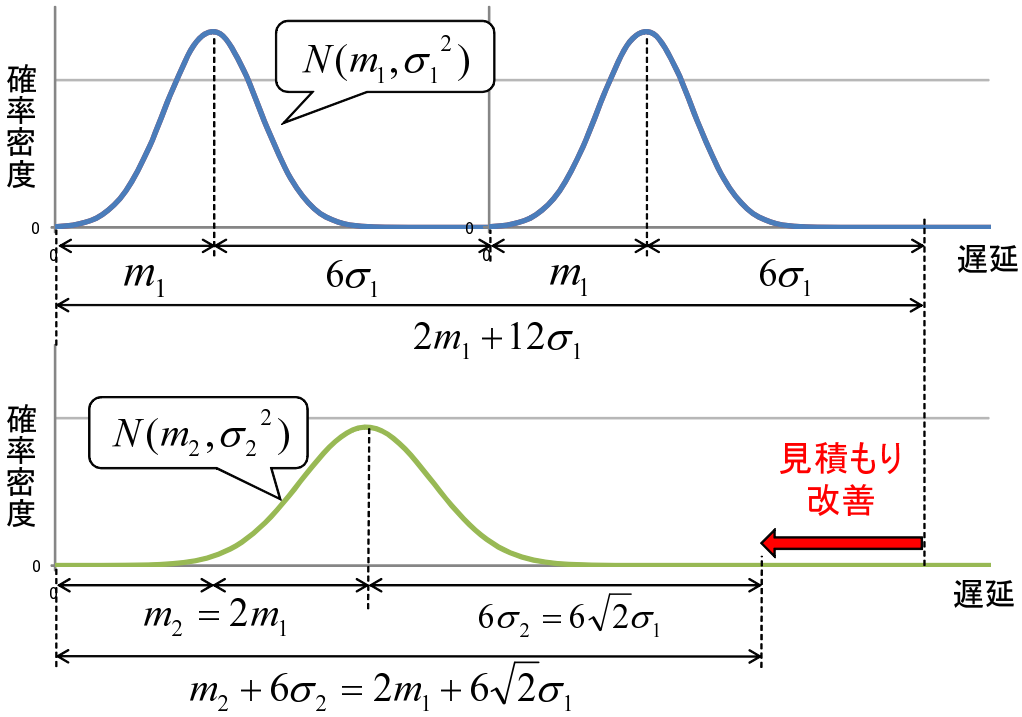


図 3.1: SSTA による，製造ばらつきの見積もり改善

3.2 入力ばらつき

3.2.1 クリティカル・パスとショート・パス

図 3.2 に示す桁上げ先見加算器は，教科書 [11] に載っているようなシンプルな構成である．Generate(G) と Propagete(P) から 4bit の和の桁上がりを求める CarryGenerator4（CG4）を基本ユニットとして，それをツリー状に組み合わせて CG16，さらにそれを組み合わせて CG64 を形成している．最終的に，和 $S(n) = A(n) + B(n) = P(n) \oplus C(n-1)$ と計算される．

桁上がりが生成される遅延時間は，桁ごとに異なる．下位 4bit の桁は，ひとつの CG4 の中での計算だけで求まるため，遅延は小さい．一方上位ビット (63:16) の桁上がりは，図の赤線で示した経路のように，ツリー状に並ぶ 3 つの CG4 の間を往復して信号が伝搬するため，遅延が大きい．このように，ロジック内の遅延時間は信号の伝達経路によって異なる．一般にあるロジック内で，遅延の大きい経路をそのロジックの**クリティカル・パス**，小さい経路を**ショート・パス**と呼ぶ．

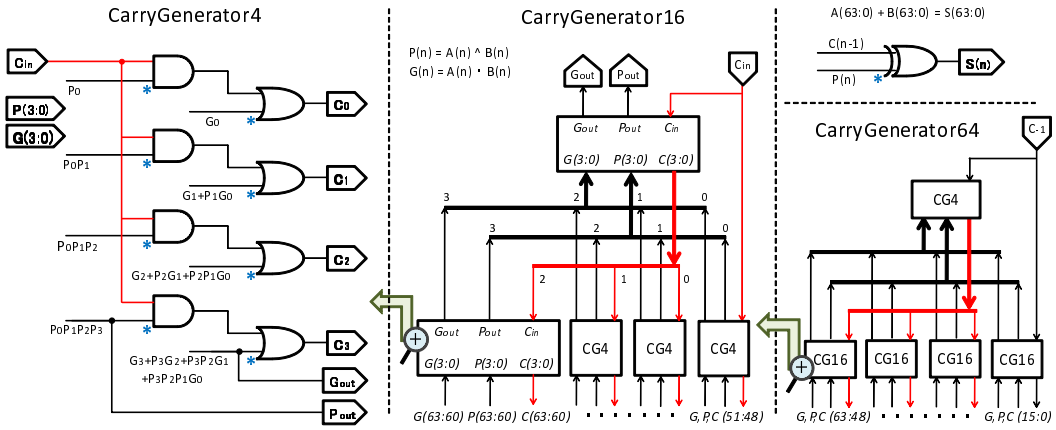


図 3.2: 64bit 桁上げ先見加算器（赤線経路がクリティカル・パス）

3.2.2 パスの活性化と実効遅延

ロジックの遅延はクリティカル・パスのみによって決まるのではない．**図 3.3** の，2 入力 OR ゲートを例に説明する．この OR ゲートの入力の一方は段数の少ないロジックに，もう一方は段数の多いロジックに，それぞれに接続されており，前者がショート・パス，後者がクリティカル・パスの一部となっている．初期状態では，2 つの入力は両方とも 0 で，結果 OR ゲートの出力も 0 となっている．あるサイクルにおいて，ショート・パス側の入力が 1 に変化し，しばらくしてクリティカル・パス側の入力も 1 に変化したとしよう．OR ゲートの出力は，ショート・パス側の入力が変化した時に 1 に変わる．しかし，クリティカル・パス側の入力が変化した時には，その変化は**マスク**され，出力は変化しない．この場合，このロジックのこのサイクルの遅延を決定したのは，クリティカル・パス側ではなく，ショート・パス側であると言える．

あるサイクルにおいて，ロジックの出力の変化をもたらしたパスは**活性化**されたと言う．あるサイクルにおいて最後に活性化されたパスの遅延を，このロジックのこのサイクルの**実効遅延**と呼ぶことにする．実効遅延は，入力の系列に依存する．このことを**入力ばらつき**と呼ぶ．

特に，ロジックの出力が 1 度も変化しなかった場合には，実効遅延は 0 となることに注意されたい．

桁上げ先見加算器の例　遅延の大きいクリティカル・パスは，遅延の小さなショート・パスよりも，**活性化率**が低い傾向がある．[7] これは，多数のゲートを通るとマスクされる確率が上がることや，クリティカル・パスがロジックの例外的な動作を司ることが多いなどの理由による．

図 3.4 は，64bit 桁上げ先見加算器の最上位桁の出力の実効遅延の確率分布を示している．

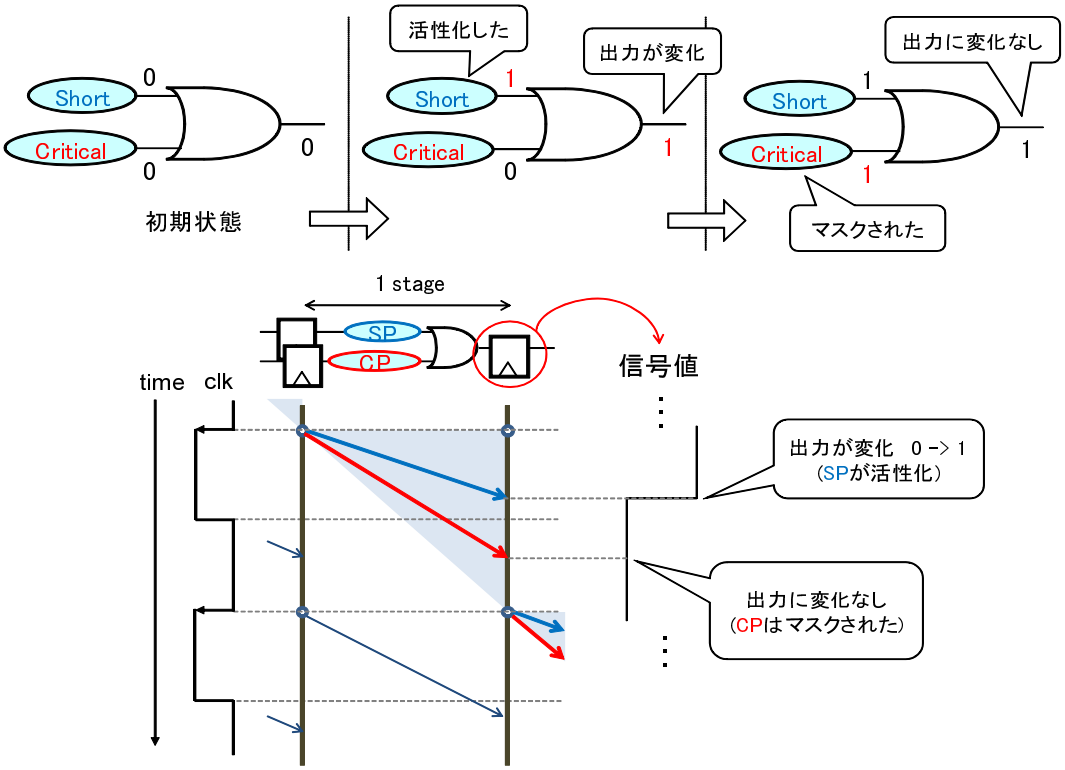


図 3.3: パスの活性化と実効遅延

桁上げ先見加算器では，4bit 桁上げ先見器の 3 階層のツリーによって，桁上げを計算している．そのため，入力によって桁上げが伝搬する桁が異なり，活性化されるパスが異なる．

まず，シミュレーションにより，ALU における各パスの活性化率を求めた．ベンチマークとしては，SPEC CPU2000 の gcc を用いた．結果，各パスの活性化率よって，同図左上のような離散的な分布を得る．前述したように，遅延 0 は，出力が直前のサイクルから変化しなかった場合を示す．この遅延 0 の場合は，47.9%にも上る．また，最長パスの活性化率は 0.022 であった．

次に，各ゲートの遅延を平均 10ps，標準偏差 3ps の正規分布として，SSTA により実効遅延の確率分布を得る．3.1 節で述べたように，段数が多いパスほど，分散は大きくなり，カーブは扁平になる．また，遅延 0 の場合は，ゲートに依らず，ばらつきの影響を受けないので，積分値 0.479 のインパルスとして表現される．

3.3 Input-variation-aware SSTA（ISSTA）

通常の SSTA では，入力ばらつきを考慮していない．クリティカル・パスの遅延の分布に対してワースト・ケース値を計算する．

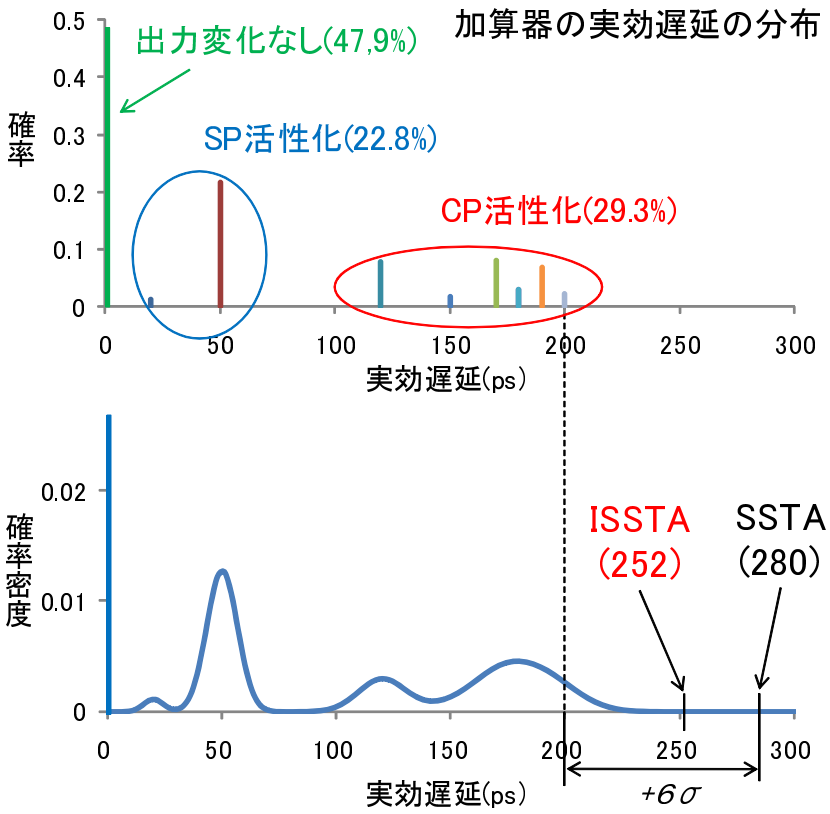


図 3.4: 64bit 桁上げ先見加算器における実行遅延の分布 (ベンチマークは SPEC CPU2000 gcc, 最上位桁)

SSTA　通常の SSTA では，クリティカル・パスの遅延の分布の，例えば $+6\sigma$ をカバーする遅延を，ワースト値とする．すると，パスの遅延がワースト値を満たさない確率は（正規分布に従う場合）100 万分の 3.4... となり，それらを不良品として廃棄することになる．図 3.4 では，最も右にあるカーブの $+6\sigma$ をカバーする遅延をワースト値とすることになる．計算すると，その値は 280ps となった．

クリティカル・パスの遅延の分布に対してワースト値を見積もるということは，入力ばらつきを考慮していないことになる．

ISSTA　動的にタイミング・フォールトを検出する方式では，クリティカル・パスの遅延の分布ではなく，ロジック全体の遅延の分布に対する確率が重要となる．図 3.4 の場合，クリティカル・パスが活性化する確率は 0.022 であったので，実効的な遅延が SSTA で求めたワースト値 280ps となる確率は，100 万分の 3.4... の更に 0.022 倍程度に過ぎない．全体の遅延の分布に対して $+6\sigma$ を与える遅延は，計算すると 252ps となり，280ps に比べて 1 割程度短い．動

的にタイミング・フォールトを検出する方式でこのロジックを周期 252ps で動作させた場合，100 万周期に 3.4 . . . 回，タイミング・フォールトが発生し，回復処理を行うことになる．

このように，入力ばらつきを考慮して求めた実効遅延の統計的ワースト値を，**Input-variation-aware SSTA**（**ISSTA**）によるワースト値と呼ぶことにする．

SSTA は,「最悪の個体の，最悪の動作環境における，最長パスの遅延」に基づいてワーストを見積もる．このため，出荷検査をパスしたならば，実効遅延は必ず SSTA ワースト以下であることが保証される．一方 ISSTA は,「ある個体の，そのときの動作環境における，100 万分の 3.4 . . . の確率でタイミング・フォールトが発生するようなライン」に基づいてワーストを算出する．動的タイミング・フォールト検出・回復手法を併用すれば，ISSTA ワーストに基づいた動作点で回路を動作させることができる．タイミング・フォールトは非常に低確率でしか発生しないため，そのときにだけ（数十から数百サイクル程度のオーバヘッドを伴いながら）回復処理を行えばよい．

第4章　さまざまなクロッキング方式

同期式回路の構成方法をクロッキング方式という．ロジックの出力が確定するタイミングについて制約（**タイミング制約**）があり，これを満たすように設計しなければ，回路が正しく動作しない．本章では，さまざまなクロッキング方式のタイミング制約を示し，そのばらつき耐性について議論する．

以下，$D_{\min}, D_{\max}$ ：ばらつきなしと仮定した，実行遅延のベスト/ワースト，$V_{\min}, V_{\max}$ ：最長/最短パスの遅延ばらつき量, とする．(単位はサイクル，すべて正の値) **図 4.1**，**図 4.3** は，縦軸に時間，横軸にロジックをとったタイミング・チャートである．

4.1　単相フリップ・フロップ

フリップ・フロップ（**図 4.2**）は設計が容易なため，最も広く用いられている．マスターとスレーブの２つのラッチが連続して並んだ構造をしている．スレーブに供給されるクロックは，マスターのクロックを反転したものとなっている．

クロックが立ち上がる一瞬にロジック間のデータの受け渡しを行う．タイミング制約は，$D_{\max} + V_{\max} < 1$ となる．タイミング・チャート図 4.1 上では，信号の通ることができる領域は，水色部分のようになる．ばらつきが増加して，$V_{\max}$ が大きくなると，タイミング制約が非常に厳しくなってしまう．

4.2　単相ラッチ

単相ラッチは，クロックが L の時に前段ロジックからの信号を通す．フリップ・フロップと異なり，信号の受け渡しのタイミングに自由度があり，次節で述べるタイム・ボローイングの性質を持っている．

$D_{\max} + V_{\max} < 1.5$ であれば，信号を遅れず次段に渡すことができる．フリップ・フロップよりも $D_{\max}$ にかかる制約が緩く，$V_{\max}$ の増加に強い．ただし，$D_{\min} - V_{\min} < 0.5$ も同時に満たさなければ，早すぎる信号が前フェーズの信号に混じってしまい，正しく動作しない．このため，ショート・パスに遅延素子を挿入して，わざと遅らせる必要がある．タイミング・チャート図 4.1 上では，信号の通ることができる領域は，水色部分のようになる．

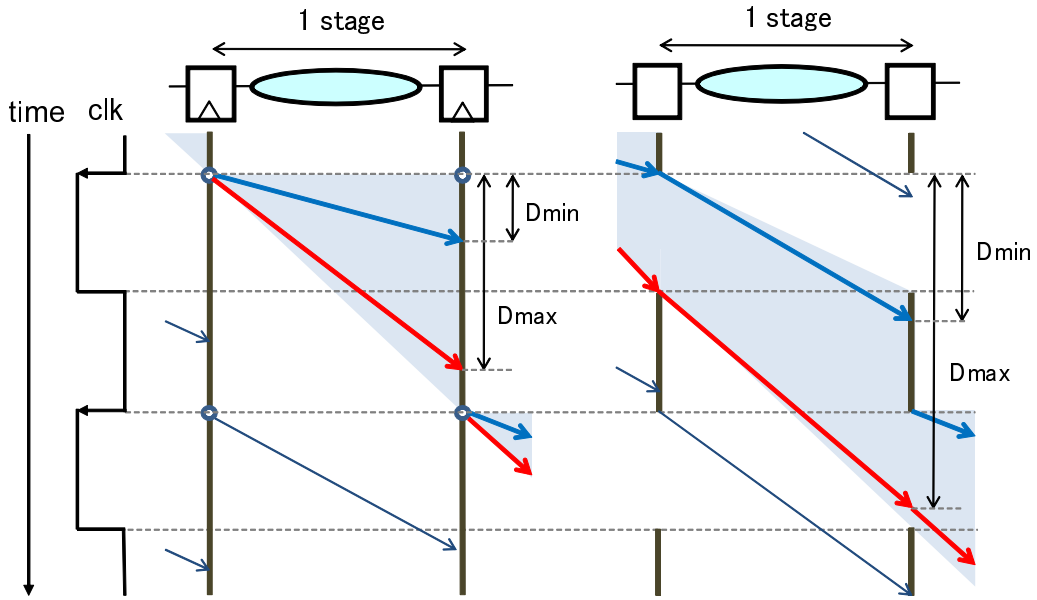


図 4.1: フリップ・フロップ (左) と単相ラッチ（右）のタイミング・チャート

クロックのデューティ比を変えて，L の期間を短くすることにより，$D_{\max}$ の制約は厳しくかわりに，$D_{\min}$ の制約を緩くできる．このような工夫をしたものを，パルス・ラッチという．

4.3　2 相ラッチ

2 相ラッチ（あるいはトランスペアレント・ラッチ，[6]）は，フリップ・フロップを構成する 2 つのラッチ（マスタ，スレーブ）のうちの 1 つを，ロジックのちょうど真ん中に移動したものである．(図 4.2)

互いに逆相のラッチが交互に並んでいるため，信号の通る領域は階段状になる．(図 4.3) フリップ・フロップ方式の 1 ステージ相当するロジックをラッチが二分する形になるため，ステージ数は 2 倍になる．前段のラッチが開いている時に，今段のラッチが閉じていて，早すぎる信号をブロックするため，単相ラッチにあった $D_{\min}$ の制約はない．$D_{\max} + V_{\max} < 1$ がタイミング制約となる．単相ラッチよりも，ラッチ数は増えるが，ロジックの大きさがフリップ・フロップ方式の半分なので，ばらつきが増加しても $V_{\max}$ の増加が小さいため，ばらつきに強い．また，クロック・スキューに対する耐性にも効果がある．[5]

図 4.2: フリップ・フロップ（左）と 2 相ラッチ（右）

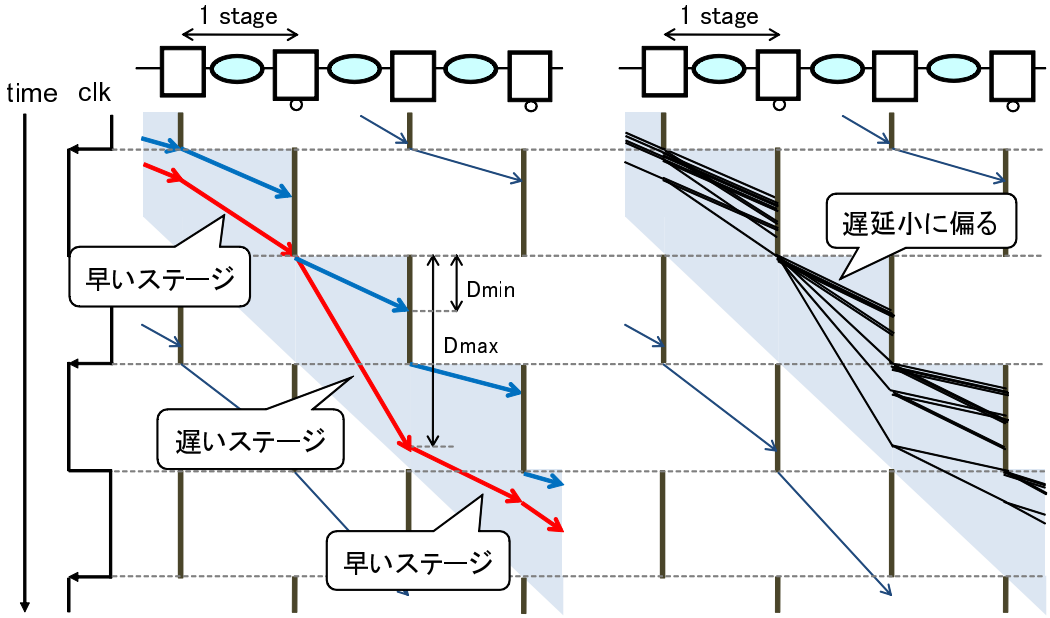


図 4.3: 2 相ラッチにおける静的タイム・ボローイング（左）と実効遅延の様子（右）

4.3.1　静的タイム・ボローイング

2 相ラッチはフリップ・フロップと異なり，信号の受け渡しのタイミングに自由度がある．あるステージが設計の都合上，前後のステージより最大遅延が 1.5 倍長くなってしまったとしても，そのステージが 0.6 サイクル，前後のステージが 0.45 サイクル，となるように周波数を決定すれば，回路を動作させることができる．このように，設計時に前後のステージ間で時間を融通する手法を静的タイム・ボローイングという．

静的タイム・ボローイングにおいてはあまり意識されないが，ステージ間の時間の融通は動作時にも行われている．図 4.3 右は，さまざまな入力に対してロジックが取り得る信号到達の線を多数サンプリングして，(信号波形のアイパターンのように）全て重ねて描いたものである．

入力ばらつきにより実行遅延は遅延小に偏る性質があるから，傾きの緩い（ショート・パスの）線は本数が多く，逆にクリティカル・パスの線は少数である．図から分かるように，３ステージ通過するための信号（線）の組み合わ

せは，早い信号*3，早い信号*2+遅い信号*1，など多数ある．このように動作時にも，遅延の異なる信号同士が，時間の融通を行っている．各ステージの線の中で最も下に位置する線を結んだもの（最遅*3 の組）が，SSTA ワースト・ケースとなる．

4.4 複数ステージにわたるロジックの実効遅延

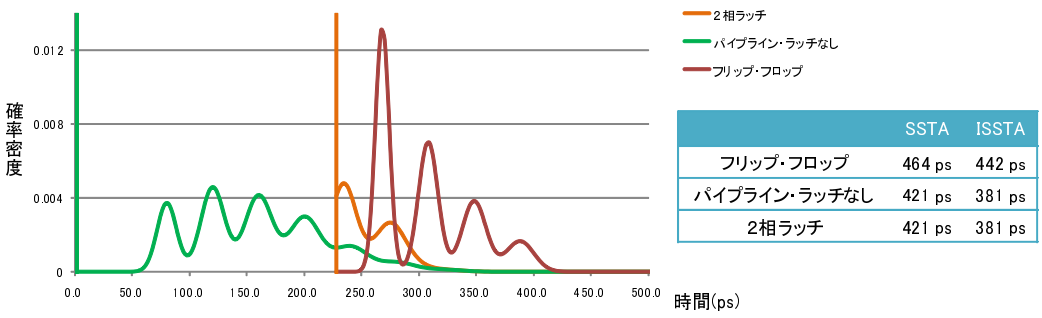
SPEC2000 gcc における加算器ロジックの実行遅延の情報をベースに，分布を簡略化した実効分布モデルを作成し，それが仮に２ステージ連続に配置されていた場合に，２ステージのロジックの実効遅延の分布がどのように変化するのか，を評価した．（図 4.4，１ステージ目への入力が確定した時刻を 0）１つの加算器ロジックの SSTA ワースト 232ps に基づき，これを１サイクルとした．

単相フリップ・フロップの場合　２ステージ目への入力が確定する時刻は 232ps なので，２ステージの実行遅延の分布は，加算器ロジックの分布を 232ps シフトさせた形になる．２ステージ目の出力が前サイクルと変わらない確率は，１つの加算器ロジックの時と同じ 0.5 であり，これが時刻 232ps の位置にインパルスとして現れる．

パイプライン・ラッチなしの場合　加算器ロジックの分布を２回，足し合わせることにより計算できる．１ステージ目で出力が確定すると，パイプライン・ラッチがないため，すぐに２ステージ目が始まる．このため４つのうちで，最も遅延小に偏る．時刻 0 には，前サイクルから変化なしの場合（確率 0.5）のインパルスが現れる．SSTA ワーストはクリティカル・パスが２連続で活性化した場合の山（平均 320ps）に対する $+6\sigma$ で 421ps，ISSTA ワーストは全体の面積の 100 万分の 3.4 . . . で 381ps となる．

２相ラッチの場合　１ステージ目で 116ps（0.5 サイクル）以内に出力が確定すると，ラッチにブロックされる．同様に，２ステージ目で 232ps(１サイクル) 以内に出力が確定しても，ラッチにブロックされる．時刻 232ps には，前サイクルから変化なしの場合のインパルスが現れる．

SSTA ワーストは 421ps，ISSTA ワーストは 381ps となる．パイプライン・ラッチなしの場合よりも遅延大に偏るため，ISSTA ワーストも悪化するが，それは無視できるほど小さい．ISSTA ワーストに支配的な影響を与える最右の山（平均 320ps，図では見えないくらい小さい）が，２者ともほぼ同じ形であるからである．それ以外の山は最右の山に対して十分離れているため，高さに差があっても影響はほとんどない．右から２番目の山は，ISSTA ワーストにおける確率密度 1.839E-7 に対して，1.63E-12 しか影響を与えない．



	SSTA	ISSTA
フリップ・フロップ	464 ps	442 ps
パイプライン・ラッチなし	421 ps	381 ps
2相ラッチ	421 ps	381 ps

図 4.4: ２ステージ分のロジックの実効遅延と，ISSTA によるワースト見積もり改善

タイム・ボローイングの有効性　図 3.4 では，SSTA ワーストと ISSTA ワーストの差は $232 - 210 = 22ps$ であった．２ステージのタイム・ボローイングを行うことで，２相ラッチの場合，差が $421 - 381 = 40ps$ に広がった．３ステージでは，SSTA，ISSTA ワーストが 605ps，522ps となり，差がさらに拡大する．タイム・ボローイングを利用して ISSTA と SSTA のワーストの差を拡大することにより，動作点を大きく引き上げることができる．

第5章　提案手法

提案手法は，動的タイミング・フォールト検出・回復を行い，実際の遅延(ISSTA ワースト) に基づいて動作する．ISSTA は，入力ばらつきを考慮しない SSTA よりも実際的なワースト遅延を見積もることができ，これに基づいて動作点を動的に設定することにより，高クロック化/低電圧化を達成できる．ごく稀に発生するタイミング・フォールトは Razor を用いて検出・回復する．

従来の動的タイミング・フォールト検出・回復手法と異なる点は，”借金”ができることである．Razor では，信号がサイクル・タイムを超えて遅れた場合は，即座に検出・回復されていた．つまり，お金が不足したら破綻した後に再開する．しかし，提案手法は信号が遅れても，それを”借金”として扱い，後続の速いステージでそれを”返済”するチャンスが与えられる．返済が滞り，借金が一定以上に貯まった時点ではじめて，タイミング・フォールトとして検出・回復する．

5.1　アプローチ

提案手法では，2 相ラッチをベースとした回路構成をとり，**動的タイム・ボローイング**を行う．前後のステージ間で時間の貸し借りができるようになり，動的に生じた借金を，動的に返済することができる．動的タイム・ボローイングは，2 相ラッチの（設計時に行う）静的タイム・ボローイングとは異なるものである．また，既存の動的タイミング・フォールト検出・回復を利用した手法は，PVT ばらつきのためのマージンを削って動作することを主眼に置いており，入力ばらつきと動的タイム・ボローイングの相乗効果を利用していなかった．

そして，動的タイミング・フォールト（稀な最悪ケース）の検出・回復を併用した上で，ISSTA ワースト（平均的ケース）に基づいて動作周波数/電圧を動的に変動させる．

1. ショート・パスの最小遅延が 0.5 サイクルになるように遅延素子を挿入して，クリティカル・パスの信号が最大 1.5 サイクル遅れても，Razor が正しい信号をサンプリングできるようにする．これにより，検出・回復可能な最大遅延を 1.5 サイクルまで拡大する．

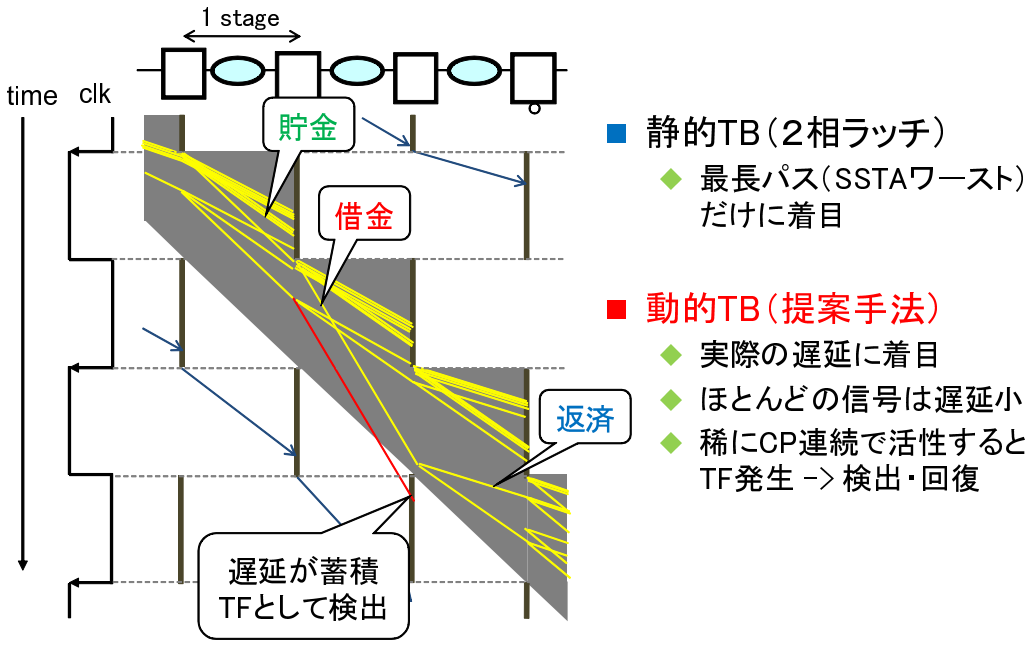


図 5.1: 動的タイム・ボローイング

2. 遅延挿入するとロジックの入力ばらつきが遅延大に偏ってしまう．これを補うために，ロジックの出力を高速に予測する回路を付加する．予測が正確であれば，ロジック全体の入力ばらつきはさらに遅延小に偏る．

3. タイミング・フォールト検出のためのサンプリング点を，クロックのデューティ比を変化させることで，ロジックごとに動的に調整する．遅れた信号が，ISSTA ワースト以内であることを正確に判定して，次段に通すことができる．

図 5.1 は，複数の実効遅延を重ねて描いたもので，動的タイム・ボローイングが行われる様子を表している．中央段の「借金」の吹き出しに指されている信号はクリティカル・パスが活性化したため，大きく遅れている．しかし，前段が早くに結果確定し（貯金し），後段もショート・パスが活性化（返済）しているため，この場合はタイミング・フォールトが発生していない．中央段の赤線のようにクリティカル・パスが連続して活性化することにより，遅延が蓄積することも低確率で発生しうる．このときはタイミング・フォールトとして動的に検出・回復される．

5.2　構成と動作

提案手法の構成と動作について述べる．

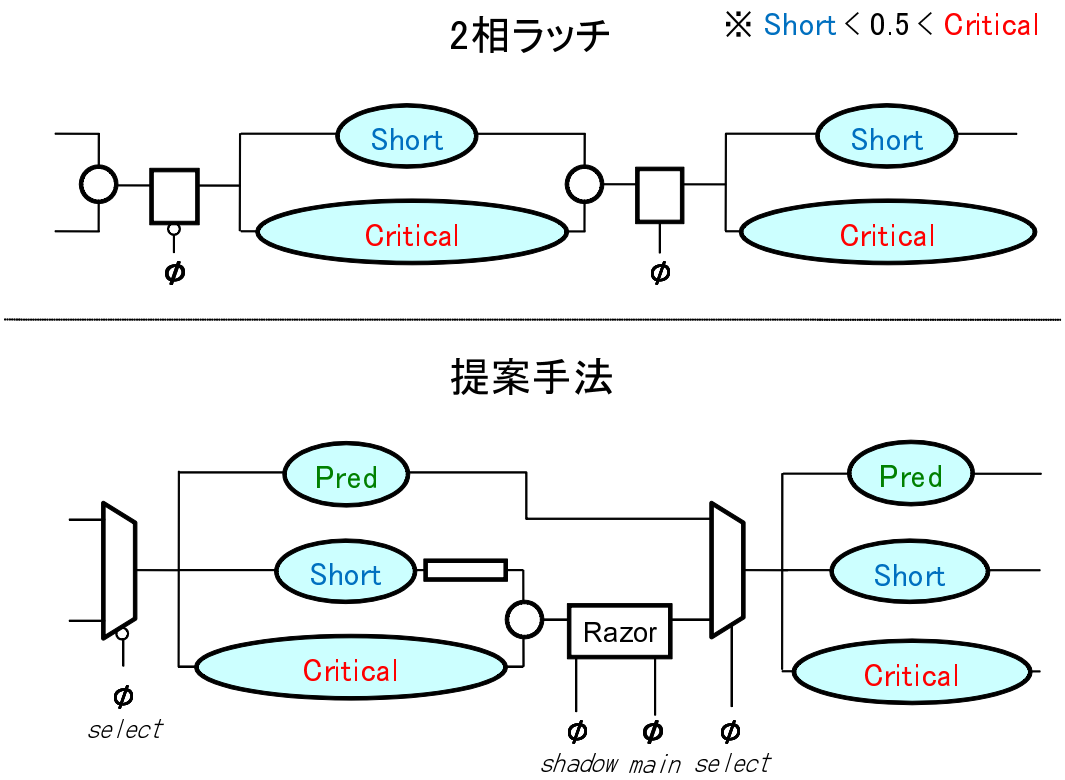


図 5.2: 提案手法の回路（下）と 2 相ラッチ（上）

5.2.1 回路構成

図 5.2 上は 2 相ラッチの回路を示し，ロジック中のショート・パスとクリティカル・パスがなんらかのゲート（図中〇形）で合流した後にラッチに接続されている．2 相ラッチ中でラッチ間に位置する部分を，以下オリジナル・ロジックと呼ぶ．

図 5.2 下は提案手法の回路で，オリジナル・ロジックの出力を高速に予測する回路（Pred）が付加されている．デューティ比の調整されたクロック (ϕ_{select}) に従って，予測結果とオリジナル・ロジックの出力を周期的に切り替える．オリジナル・ロジック中のショート・パスには遅延素子（図中□形）が挿入される（後述）．タイミング・フォールト検出のために Razor1[3] を付加する．Razor には ϕ_{main} と ϕ_{shadow} の 2 種類のクロックが供給されている．

5.2.2 動作

図 5.3 右は，提案手法のタイミング・チャートである．1 段のロジックに，2 本の線が描かれている．下側の線は SSTA ワーストの線，上側は ISSTA ワー

ストの線である．前述したように，タイム・ボローイングを続けるにつれて，ISSTA ワーストと SSTA ワーストの差が広がっていく．

タイミング・フォールトの検出　Razor が main FF にサンプリングするタイミングが，ISSTA ワーストのタイミングである．そして，shadow latch にサンプリングするタイミングが SSTA ワーストのタイミングである．従って出荷検査をパスしたならば，shadow latch の値は必ず正しいことが保証される．shadow latch にサンプリングした時点で，その値が main FF の値と異なっていたら，タイミング・フォールトとして検出・回復される．

提案手法においては，ロジックごとに Razor の main FF へのサンプリング・タイミングを動的に調整する．実行中のプログラムによって入力ばらつきが異なるため，いくつかの組み合わせの中から最適なサンプリング・タイミングに設定する．Razor に分配されるクロックのデューティ比を，ロジックごとに変更することにより実現する．

Razor にはロジックの最小遅延に関して制約がある．shadow latch に正しい値をサンプリングするためには，次フェーズのショート・パスを通った信号がサンプリング前に到達して上書きしてはならない．このために，図 5.2 のように，オリジナル・ロジックのショート・パスには遅延素子を挿入して，最小遅延が 0.5 サイクルより大きくなるよう設計する．ただしこれにより，オリジナル・ロジックの実効遅延が遅延大に偏ってしまう．これを補うために，次に述べる予測回路を用いる．

予測回路の動作　予測回路 (Pred) は，オリジナル・ロジックの出力を高速に予測する回路である．例えば，出力が 0 に偏向していれば単に GND を予測回路にできるし，オリジナル・ロジックのショート・パスを複製して構成することもできる．

オリジナル・ロジックには遅延素子を挿入するが，予測回路は高速に設計する．オリジナル･ロジックがタイミング･フォールト検出を行っている間（フェーズの前半）に，予測回路の出力を後段へ入力する．タイミング・フォールト検出が終了したら（フェーズの後半），セレクタをオリジナル・ロジックに切り替えて，その結果を後段へ入力する．予測が正しければ，切り替え時に値は変化しないため,「貯金」あるいは「前フェーズの借金の返済」が成功したことになる．これにより，予測回路を使って今フェーズの結果を後段に伝えると同時に，前フェーズの結果の正当性をオリジナル・ロジックで検証することができる．２つ併せて１つのロジックと考えた場合，このロジックの入力ばらつきは 2 相ラッチの時と同じかそれ以上に遅延小に偏るため，タイム・ボローイングに有利となる．

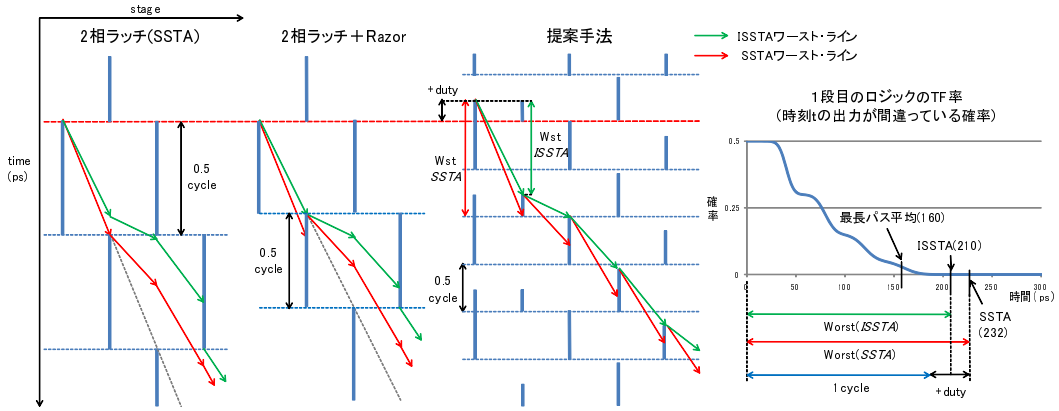


図 5.3: 提案手法（右），2 相ラッチ+Razor（中央），2 相ラッチ（左）のタイミング・チャートの比較

5.3　2 相ラッチとの比較

SSTA を用いて 2 相ラッチを設計する際（図 5.3 右）には，起点とするステージに最も遅く入力が到達したと仮定して，その点（図中ラッチが閉じる瞬間）から SSTA ワーストのラインを引いていき，そのラインが全ステージにおいてラッチが閉じるよりも早くなるように，動作周波数を決定する．

2 相ラッチに Razor を付加した場合（図 5.3 中央）では，全ステージ中最もクリティカルなパス上において，非常に低確率でタイミング・フォールトが発生するくらいにまで，周波数を動的に向上できる．ISSTA ワーストが，100 万分の 3.4 . . . の確率でタイミング・フォールトが発生するラインであるとすれば，ラッチが閉じるタイミングをこれと一致させることにより，SSTA のみの場合と比べて，図中では約 20% 高クロック動作している．

提案手法（図 5.3 左）では，起点とするステージの最大遅延が大きいため，前段においてタイミング・フォールト検出のためのサンプリング点を早めている．これによって，この点より遅い時刻から入力が来る可能性を考えなくてもよくなり，最大遅延の大きなロジックをステージに含むことができるようになり，2 相ラッチ+Razor の場合の倍近い，大幅な高クロック化を可能としている．

図 5.3 右のグラフは，１段目のロジックにおけるタイミング・フォールト率が時刻によってどのように変化するかを示している．時刻 0 は前段におけるサンプリング点で，このときの確率は 0.5 である．これは，１段目ロジックの出力が前フェーズから変化する確率と同じである．時間とともに出力が間違っている可能性は減少していき，ISSTA ワースト 210ps において 100 万分の 3.4 となる．１段目ロジックの終端にある Razor が main フリップ・フロップにサンプリングするタイミングをこれと一致させ，shadow フリップ・フロップへは SSTA ワーストでサンプリングする．このとき，shadow フリップ・フロップの

値は必ず正しく，main フリップ・フロップと値を比較して不一致となる（タイミング・フォールトとして検出する）確率が 100 万分の 1 となる．後段のロジックにおいても，ISSTA ワースト・ラインに沿うように main フリップ・フロップへのサンプリングが行われる．

第6章　考察・検討

本章では提案手法に関する定量的・定性的な考察を行う．

6.1　本研究の与えるインパクト

提案手法は，既存の2相ラッチに比べて1.5倍ものばらつき耐性を持っている．従来ではタイミング・フォールトにより不良品となっていた製品を出荷できるようになり，歩留まりの大きな向上が見込める．ばらつきに強くなった分，電圧を下げて低消費電力化することもできるし，クロック周波数を上げて高速化することもできる．

高クロック化と低電圧化に対する効果を試算してみる．まず高クロック化について，1.5倍の遅延を許容できるならば1.5倍の周波数で動作することになる．次に電圧について．ロジック遅延は$V_{dd}/(V_{dd}-V_{th})^{1.3}$（$V_{dd}$:電源電圧，$V_{th}$:閾値電圧）におおよそ比例する．よって，$V_{dd}=1$V, $V_{th}=0.4$V で動作するロジックがあったときに，これに提案手法を適用すると$V_{dd}=0.75$V で動作する計算となる．動的消費電力はV_{dd}^2に比例するので，40%以上も動的消費電力を削減することができる．実際には問題はこれほど単純ではないものの，提案手法は非常に大きな効果を期待できる．

6.2　オーバーヘッド

提案手法のオーバヘッドについて考察する．

回路面積の増加　3章で例のため実装した桁上げ先見加算器の回路を例に提案手法のオーバヘッドを考察する．この場合，$S(63:16)$の桁上がり生成経路に合計$(64-16)*3=144$個のラッチを挿入することにより提案手法を適用できる．各ロジックのトランジスタ数を，AND:6，OR:6，XOR:8，FF:16，ラッチ:10として計算した時，桁上げ先見加算器自体のトランジスタ数は5188個となる．（回路の最適化は考慮していない）フリップ・フロップで構成すれば$5188+16*64=6212$個，2相ラッチで構成すれば$5188+10*(64*2)=6468$個，提案手法で構成すれば$5188+10*(144+64)=7268$個，のトランジスタ数となる．一般に提案手法が追加するラッチによって，2相ラッチで構成した

場合と比べて，ロジックあたり 10〜20%程度トランジスタ数が増加すると推測している．

汎用プロセッサ・チップにおいてはキャッシュがチップ面積の約半分，分岐予測テーブルやレジスタ・ファイルなどの RAM も相当な面積を占めている．このため，ロジック部はチップ面積の 30〜40%程度に留まる．その中で，フリップ・フロップが占める面積はロジック部の 10〜20%程度，つまりチップ面積の 3〜8%である．提案手法はタイミング・クリティカルなロジック部分に対してのみ適用することを考えているが，Razor の論文がチップを試作した結果，タイミング・クリティカルなフリップ・フロップは 10%程度であるという．つまり単純に試算すれば，提案手法の適用対象となるフリップ・フロップはチップ面積の 1%未満にすぎず，ラッチの数の増加による面積増加は無視できる．Razor などの検出・回復機構を追加することを考慮しても，やはり面積増加は問題にならないと考えられる．

追加ロジックの消費電力の影響　上述したとおり追加要素はわずかで，全体から見た消費電力は低電圧化することでむしろ大幅に削減できる．

回路設計の効率に与える影響　ロジックを分割してラッチを挟むという操作は，スーパー・パイプライニングを施すこととよく似ている．スーパー・パイプライニングはパイプライン構成などのアーキテクチャレベルの変更を伴うため，自動化することは難しい．一方，2 相ラッチや提案手法の適用はアーキテクチャレベルには変更がなく，単にロジックを分割するだけなので，自動化することが可能である．ロジック内でのラッチの挿入位置や個数を最適化したり，そのために一部の回路を複製したりということも自動的に処理させることができる．従って，提案手法を導入するにあたって，回路設計が著しく難しくなることはないと考えている．ばらつき吸収の効果により，むしろ設計は容易になる．

第7章　おわりに

半導体プロセスの微細化に伴って，回路遅延のばらつきの増加が，回路設計における大きな問題となりつつある．これは，トランジスタや配線のサイズが原子のサイズに近づくためであり，原理的に避けることができない．ばらつきが増大していくと，従来のワースト・ケースに基づいた設計手法は悲観的になりすぎる．今後の半導体産業の発展には，ばらつき対策技術が重要になる．

本研究では，動的タイミング・フォールト検出・回復技術の一つ Razor の考え方を更に推し進める．Razor では信号がサイクル・タイムを超えて遅れた場合は，すぐに検出されて回復処理を行ってしまう．提案手法では，動的タイム・ボローイングを利用することにより，信号がサイクル・タイムを超えることを許し，遅延が一定以上蓄積するまでは検出・回復を行わない．

入力ばらつきを考慮した ISSTA は，SSTA よりも実際的なワーストを見積もることができる．タイミング・フォールト検出・回復を併用して正常動作を保証した上で，プロセッサの動作点を ISSTA に基づいて動的に変動させれば，高クロック化/低電圧化を達成できる．これにより，最大で SSTA を用いて設計された 2 相ラッチの２倍近い高クロック化を達成できる．

7.1　今後の予定

現在，OpenSPARC の FP Adder に対して提案手法を適用している．EDIF フォーマットのネットリストを読み込んで，自動的に回路を解析・提案手法の適用を行うツールを作成し，これを用いて適用する．これの目的は以下の通りである．

1. 実際の回路上でどのように適用可能なのかを調査する
   回路遅延が問題になる部分はどこか，パスの分離の仕方，予測回路の構成方法，など

2. 適用した回路のクロック周波数・電圧を変動させた場合の挙動の分析
   適用の仕方・適用した数と，変動可能範囲の関係，など

3. 救済できる限界を超えた遅延が発生した場合，タイミング・フォールトとして検出・回復できるか

4. 面積増加がどれくらいになるかを確認する

5. 上記の調査結果をフィードバックして最適な回路構成を検討する

現在研究室の別プロジェクトで，フルカスタムに近いプロセッサチップを設計中であり，LSI 試作サービス VDEC にて試作予定である．順調に研究が進んだ場合，このチップに提案手法を組み込むことを考えている．

チップ試作の目的は，実際に使われている intel などのプロセッサ上でも提案手法が有効であると示すためである．このチップは，研究室の今までの研究成果を数多く実装したチップで，ARM などの商用プロセッサを凌ぐ性能を出すことを目標としている．実用的な性能を狙うこのチップ上において，ばらつき耐性向上・高クロック化・低電圧化の効果を実証できれば，本研究は極めて工学的価値の高いものとなる．

回路遅延が問題となる部分を調べて，作成したツールを利用して提案手法を適用していくつもりである．

試作チップを用いて，クロック周波数や電圧の変動可能範囲について，実際のデータを測定する．同時に，試作の経験を活かして，より実用的な構成方法を研究する．対外発表の際には,「どうすればより多くのプロセッサに成果を使ってもらえるか」を特に意識して発表したい．

付録

シグマ・レベルとワースト・ケース

平均が m で，標準偏差 σ の正規分布 $N(m, \sigma^2)$ と表記される．標準偏差 σ は，平均値からのばらつき具合を表す．表 7.1 は，平均から N シグマ離れたときに，それに含まれる割合と外れる割合を示している．例えば，平均 $\pm 1\sigma$ の範囲には，全体の 68.2689492%が含まれる．

回路設計におけるワースト・ケースは，歩留まりの目標を考慮した上で，経験則をもって決められる．本稿では，平均 $+6\sigma$ をワースト・ケースに設定した．これは，品質管理手法として一般的であるシックス・シグマ法に基づいている．シックス・シグマ法では,。「100 万回の作業を実施しても不良品の発生率を 3.4 回に抑える」ことを目標にしている．

しかし，表 7.1 において，6σ を外れる割合は，0.0000001973%となっており，100 万分の 3.4 とは大きく異なる．シックスシグマの考え方においては、様々な長期的変動要因を考える必要から「経験的に得られた 1.5σ 程度の変動」を加味し、また良い結果は不問として、「-4.5σ 以上（偏差値で言えば 5 以下）」の発生確率を目標値としている。すなわち 100 万分の 3.4 と言う数字を使っている。

z σ	percentage within CI	percentage outside CI	ratio outside CI
1 σ	68.2689492%	31.7310508%	1 / 3.1514871
1.645 σ	90%	10%	1 / 10
1.960 σ	95%	5%	1 / 20
2 σ	95.4499736%	4.5500264%	1 / 21.977894
2.576 σ	99%	1%	1 / 100
3 σ	99.7300204%	0.2699796%	1 / 370.398
3.2906 σ	99.9%	0.1%	1 / 1000
4 σ	99.993666%	0.006334%	1 / 15,788
5 σ	99.9999426697%	0.0000573303%	1 / 1,744,278
6 σ	99.9999998027%	0.0000001973%	1 / 506,800,000
7 σ	99.9999999997440%	0.0000000002560%	1 / 390,600,000,000

表 7.1: シグマ・レベルとそれに含まれる割合，外れる割合

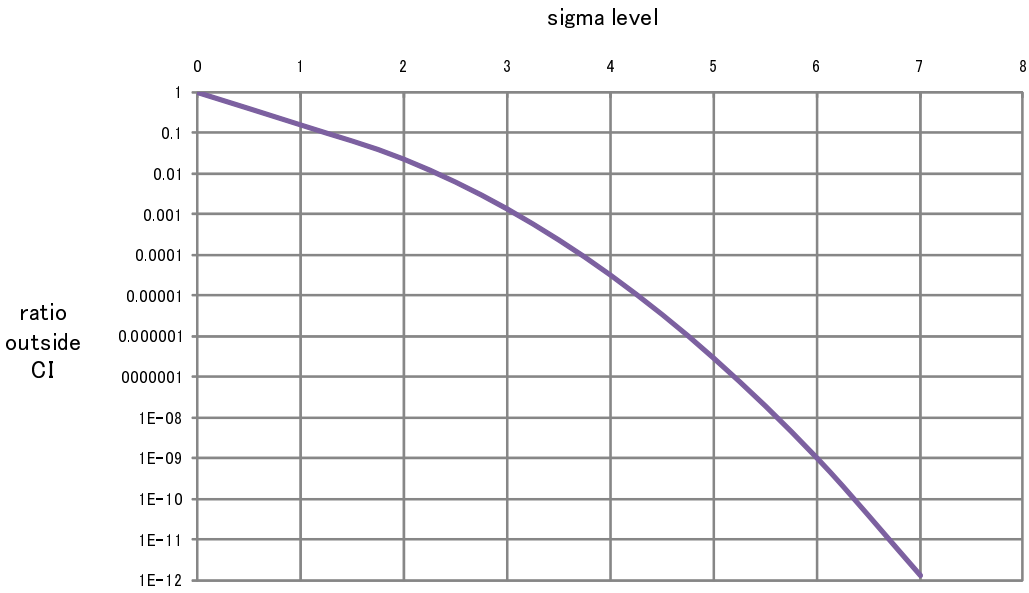


図 7.1: シグマ・レベルとそれを外れる割合

関連図書

[1] Srivastava Ashish, Sylvester Dennis, and Blaauw David. Statistical Analysis and Optimization for VLSI: Timing and Power. *ISBN: 978-0-387-25738-9*, 2005.

[2] D. Blaauw, S. Kalaiselvan, K. Lai, Wei-Hsiang Ma, S. Pant, C. Tokunaga, S. Das, and D Bull. Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance. In *Int'l Symp. on Solid-State Circuits Conference (ISSCC)*, 2008.

[3] D.Ernst, N.Kim, S.Das, S.Pant, T.Pham, R.Rao, C.Ziesler, D.Blaauw, T.Austin, and T.Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Int'l Symp. on Microarchitecture (MICRO)*, pp. 7–18, 2003.

[4] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, V. Pokala, and Austin IBM. A distributed critical-path timing monitor for a 65nm high-performance microprocessor. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pp. 398–399, 2007.

[5] David Harris. Skew-tolerant circuit design. *Morgan Kaufmann Publishers*, 2001.

[6] I. E. Sutherland. Micropipelines. *Communications of the ACM*, Vol. 32, pp. 720–738, 1989.

[7] K. Usami, K. Nogami, M. Igarashi, F. Minami, Y. Kawasaki, T. Ishikawa, M. Kanazawa, T. Aoki, M. Takano, C. Mizuno, M. Ichida, S. Sonoda, M. Takahashi, , and N. Hatanaka. Automated low-power technique exploiting multiple spply voltages applied to a media processor. *IEEE J. Solid-State Circuits*, Vol. 33, pp. 463–472, 1998.

[8] 杉本健. タイミング・フォルト耐性を持つスーパスカラ・プロセッサ. 東京大学修士論文, 2009.

[9] 佐藤寿倫. カナリア・フリップフロップを利用する省電力マイクロプロセッサの評価. 先進的計算基盤シンポジウム SACSIS, pp. 227–234, 2007.

[10] 佐藤寿倫, 国武勇次. ばらつき耐性を持つカナリアＦＦを利用したデザインマージン削減による省電力化. 情報処理学会論文誌, pp. 2029–2042, 2008.

[11] 五島正裕. ディジタル回路. 数理工学社, 2007.

研究業績

主著論文

1. 予測ミスをした命令の実行を継続するアーキテクチャ
   喜多貴信
   卒業論文，東京大学 工学部 (Feb. 2008)

2. 予測ミスした命令の実行を継続する投機手法
   喜多貴信, 塩谷亮太, 入江英嗣, 五島正裕, 坂井修一
   情報処理学会研究報告 2008-ARC-178, pp. 7-12pp. (2008).

3. タイミング制約を緩和するクロッキング方式の予備評価
   喜多 貴信, 樽井 翔, 塩谷 亮太, 五島 正裕, 坂井 修一
   電子情報通信学会研究報告 CPSY2009-26, pp. 61-66 (2009).

4. タイミング制約を緩和するクロッキング方式の提案
   喜多貴信，塩谷 亮太， 五島 正裕，坂井 修一
   情報処理学会 第 72 回全国大会 (2010)

5. タイミング制約を緩和するクロッキング方式
   喜多貴信，塩谷 亮太， 五島 正裕，坂井 修一
   先進的計算基盤システムシンポジウム SACSIS 2010
   （投稿中）

共著論文

1. 耐永久故障 FPGA アーキテクチャ
   岡田 崇志, 喜多 貴信, 五島 正裕, 坂井 修一
   情報処理学会研究報告 2009-ARC-184, No. 4 (2009).

2. 過渡故障耐性を持つ Out-of-Order スーパスカラ・プロセッサの評価
   有馬 慧，岡田 崇志，堀尾 一生，喜多 貴信，塩谷 亮太，五島 正裕，坂井 修一
   情報処理学会 第 72 回全国大会 (2010)

受賞

1. 予測ミスをした命令の実行を継続するアーキテクチャ
   平成 20 年度　優秀卒業論文賞，東京大学 電気系３学科（2008）

2. 予測ミスした命令の実行を継続する投機手法 (ARC178-2)
   平成 21 年度 山下記念研究賞, 社団法人 情報処理学会 (2009).

謝辞

非常に多くの方々から多大なご指導、ご協力、励ましを頂き、本論文を完成させることができました。この場を借りて、感謝の意を表したいと思います。

指導教官である五島正裕准教授には学部 4 年からの 3 年間に渡って多くのご指導，ご助言を頂きました．坂井修一教授からも，大変多くのご指導を頂きました．ここに深く感謝の意を表します.

塩谷亮太氏には，ミーティングなどでいつも指導を行って頂いたのを始めとして，様々な点でお世話になりました.

八木原晴水さん，長谷部環さん，伊世知代さんには，研究室における設備の導入や各種事務手続きなど，研究室で過ごすための様々なご支援を頂きました.

また，ここでは紹介しきれなかった研究室のメンバーの皆様にも様々な形でお世話になりました．本当にありがとうございます.