修士論文

動作時信号遷移監視に基づく
遅延補償フリップフロップの研究

Delay-Adaptation Flip-Flops
Featuring
In-Situ Signal-Transient Monitoring

指導教員　　坂井修一 教授

2008年 2月 4日

東京大学大学院 情報理工学系研究科
情報理工学系研究科 電子情報学専攻
48-66437

廣瀬健一郎

論文概説

半導体集積回路は、その発明以降劇的な発展を続けており、高い性能や利便性は現在もその応用分野を次々に広げていっている。そのような半導体集積回路のめざましい発展は、集積回路製造技術の進歩のよる微細化・高集積化によって下支えされてきたといえるが、その極端な追及によっていくつかの問題点も顕在化し始めている。主要な例としては、トランジスタの微細化・高集積化による動的・静的消費電力密度の増大、プロセスバラツキによる製造後性能の正確な見積もりの難化とそれに伴う設計マージンの増大と設計の難化、低電源電圧化によるノイズ耐性の低下などがあげられる。半導体集積回路においては、従来からの不断の性能追及とともに、現代の情報社会のインフラストラクチャーとして高い信頼性が求められるようになってきているにもかかわらず、このような問題点はその信頼性を著しく損なう可能性をはらんでいる。

これらの問題に対して統一的に対処する方法として、本稿では信号遷移タイミング監視に基づく遅延補償フリップフロップの提案を行う。このフリップフロップでは、データ入力信号の遷移タイミングを動作時に監視し、タイミングエラーの発生を検出あるいは予測する。そして、タイミングエラーの発生が検出されると、フリップフロップが誤ったデータを取り込まないようにデータの取り込みタイミングを調整する。こうしてえられる回路動作時のプロファイルをもとに、電源電圧や動作周波数をちょうせいすることによって、電源揺らぎや、製造時のプロセスバラツキによって発生するい予測しがたいタイミングエラーを回路動作時に自動的に回避することが可能となる。また、動作時にエラーの発生状況をモニタリングするので、これらの外乱に対する設計段階でのマージンを削減することが可能であり、動作環境に合わせて電源電圧を調整できることは高い省電力化効果ももつことになる。さらに、動作時信号監視システムを実現する既存手法と比べて、遅延補償フリップフロップは用いられる場合は回路のタイミング設計自由度を高く保つことができる。また、信号遷移タイミングを監視するため、タイミングエラーだけでなく、ディープサブミクロンプロセスのテクノロジを用いた際に顕著になってきているソフトエラー耐性にも優れている。遅延補償フリップフロップは以上のような、多くの特徴をもつ。

本論文では上記のように、タイミングエラー耐性・省電力特性・ソフトエラー耐性の 3 点に関する回路動作レベルでの評価検討を行い、既存手法との対比も行いながらその特長を明らかにする。その結果、電源電圧の揺らぎに対するタイミングエラー耐性の高さや、クロックゲーティング・ＤＶＦＳなどの既存省電力化技術と併用した場合の高い省電力化効果を確認した。ソフトエラー耐性においても、とくに SET 耐性の高さが確認できた。また実際に集積回路システム中に遅延補償フリップフロップを適用する場合にひつようとなるであろう、遅延補償の影響やメモリセルにおけるエラー監視などの要素技術についても考察を行った。

半導体集積回路は、近年の爆発的な用途の拡大に伴って、処理性能だけではなく、省電力特性や信頼性、堅牢性などディペンダビリティに対する性能要求も多岐にわたる。本研究は、ディペンダブル VLSI を実現するための要素技術の提案の一つとして行われたものであり、処理性能を維持しつつ、なるべく少ないオーバーヘッドで集積回路のエラー耐性や、省電力特性を高めることを目的とするものである。

目次

図目次

表目次

第I部

研究の背景と提案手法

第1章

Introduction

1.1　研究の背景

1970 年代以降、マイクロプロセッサは劇的な性能向上をとげ、それに伴ってその使用分野を急速に拡大してきた。この性能向上に大きく貢献した要因の一つとして、半導体デバイスの集積度と動作速度の指数関数的な向上が挙げられる。“半導体集積度は 1.5 年で 2 倍になる” という傾向は More の法則 [1] として知られており、数十年という長きにわたり、多岐にわたる科学技術の発展の原動力の一つとして貢献してきた (図 1.1) 。近年一般的に用いられている集積回路は MOS トランジスタによって構成されているが、MOS トランジスタはスケーリング則にそって微細化することで、より動作が高速に、より省電力に、より高集積になるという性質があり、求める性能間に trade-off の関係が無いという大きな特徴がある [2]。この非常に望ましい性質が、長年の集積回路素子の微細化・高集積化に関する研究開発を強力に推し進める一つの要因となってきたということはきわめて容易に想像できる。しかしながら、トランジスタのゲート長が 100nm をきった頃から、この理想的な性質にかげりが見え始めてきており、単純に集積回路の高機能化のために、構成半導体素子を微細化して集積度を向上させるだけでは、さまざまな問題が近い将来表面化してくることが予想されている。

従来から半導体集積回路に関する研究開発はおもに演算処理の高性能化を追求してきたということがいえるが、近年その用途の拡大に伴って、演算性能以外の面に対する要求も強くなってきている。たとえば、携帯機器の普及や、環境問題への意識の高まりから、省電性に優れることが求められたり、多くの重要な社会基盤に利用され、多くの人が利用できるようになったため、高い信頼性を有する必要があることなど、社会からの性能要求は非常に多岐にわたる。もはや半導体集積回路は、多くの人が手軽に利用する現代の情報社会のインフラストラクチャーであり、人や社会が真に依存しうるものかという、総合的な評価さえも求められるようになってきている。このような要求から、広い意味での安全・安心な情報システムを構築する礎のとなるようなディペンダブルプロセッサの研究が盛んになってきている。しかしながら、上述のように高性能化に伴う、極端な微細化が引き起こす問題は、このような信頼性などを悪化させることにつながるようなものも多く、高い性能を維持しながら、ディペンダビリティを追求することは容易なことではない。

本論文では、このように近年問題となってきているような種々の点に関して、なるべく多くの点に対して統一的に対応しつつ、信頼性・消費電力・処理性能を引き出せるようなシステムとして、動的な信号遷移監視にもとづく遅延補償フリップフロップを提案する。また、この遅延補償フリップフロップの構成と、以下に挙げる最先端集積回路の問題点への優れた性質を示し、システムへの実装に関する考察から超ディペンダブル VLSI への応用可能性について考える。以下、本章では近年の集積回路に関する問題のうち代表的なものを例示し、次章以降の論文の構成を示す。

1.2　近年の集積回路において顕在化し始めている課題

本研究は、近年顕在化してきている集積回路の問題を統一的に扱うための技術として、遅延補償フリップフロップを提案し、その適用例と効果を示すことを目的とするものである。そこで、まず研究の背景となる集積回路の一般的な問題について本節で簡単に触れる。

- 素子性能のばらつき
  近年の半導体製造技術では、非常に微細な素子が実現できるようになってきており、トランジスタのサイズが原子数十個というオーダーで表されるようなものも存在する。そのような場合、トランジスタ中の不純物の個数などは場所によって確率的に大きく変動することになり、同じチップ内でも素子の性能にばらつきが生じる [5] [6] [7]。
- 消費電力の増大

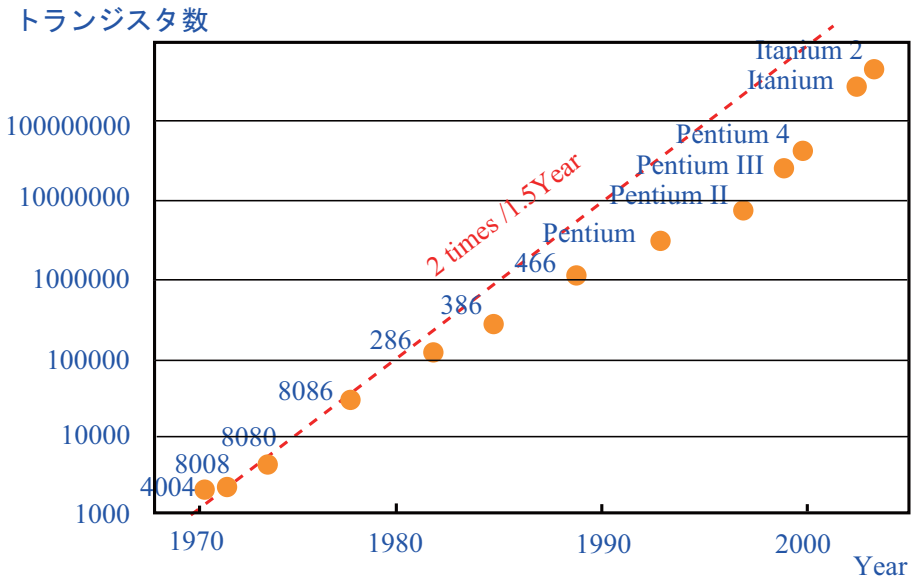


図 1.1: 商用プロセッサにみられるムーアの法則 (G. Moore, ISSCC 2003 [3] を元に作成)

待機時のトランジスタのリーク電流は、素子の微細化に伴って指数関数的に増加している [3]。また、処理の高速化、素子の高集積化は消費電力のエネルギー密度の増大も招いている。

- 配線遅延の相対的な増加
  古典的なスケーリング則によればトランジスタのサイズの指数関数的なスケーリングに比例してトランジスタの動作速度は高速になることが示されている。このゲート遅延に対して配線遅延は分布定数回路の抵抗・容量による遅延であると考えることができ、L を配線長とする一次近似で

$$wiredelay \simeq \frac{1}{2}RCL^2 \tag{1.1}$$

  と表すことができる。配線の大きさ（配線長・配線幅・配線間距離）が理想的に縮小した場合ですら、抵抗値は配線幅と配線間距離に比例し配線長に反比例するため、線形的な改善しか望めない。近年、この配線遅延の影響は顕在化してきており、プロセスの微細化に伴って配線遅延の大きさはさらに増加することが予想される。
- 低電圧化によるノイズ耐性の減少
  最先端の集積回路は非常に低い電圧で駆動するようになってきている。これは省電力の観点からは望ましいことであるが、ノイズが相対的に大きくなることになるため、ノイズ・エラーに対する脆弱性が増してきているといえる。
- 設計マージンの増大と設計の難化
  上述したいくつかの内容とも関連するが、ノイズや性能ばらつきが大きくても、正しい動作が保証できるように、設計時にとられるマージンはかなり大きくなっている。これによって、本来の限界性能よりも低い性能しか発揮できないということが起こりうる。

1.3　遅延補償フリップフロップの適応範囲

本論文において提案されている遅延補償フリップフロップは、これらのうち、配線遅延の相対的な増加以外の問題を統一的に扱うことを目的としている。遅延補償フリップフロップの特徴としては、タイミングエラーの監視を、実際に入力される信号の遷移のタイミングを直接監視して行っていることと、エラー発生時にデータの取り込みを遅らせ、正しい値を取り込めるようにフリップフロップ自身が自律的に調整

を行いつつ、エラーを監視することである。DVFS（Dynamic Voltage and Frequency Scaling）による動的な動作周波数や供給電圧の最適化と協調することで、個々のチップが独立してその性質と動作環境における最適な動作状況が選択できるため、プロセスばらつきキに強く、積極的な低消費電力化が達成できる。また、これに関連して、プロセスばらつきに対するマージンや動作環境からの外乱に対する設計段段階で挿入されるべきマージンを最小化することができる。さらに遅延補償フリップフロップは従来技術に比べて優れたソフトエラー耐性をもつことも特徴である。このような、動作時にエラー監視技術では、近年の集積回路設計の難化の問題の一つである、False Path Problem の緩和に対する効果も期待できる。本論文では遅延補償フリップフロップの仕組みとこれらの性質に対する詳説を、いくつかの既存研究との比較を用いながら行うことを目的とする。

1.4　本論文の構成

以下、第 2 章においては、通常のフリップフロップとタイミングエラーに関する基礎的な話題をふまえて、提案手法と同じくフリップフロップを起点として動作時エラー監視を行う既存技術の説明を行う。ここでは Razor と Canary Flip Flop を既存技術として紹介し、以降本論文全体を通してこれらの技術との比較を通じ提案手法の特徴をより明確に示す。その後第 3 章において、本論文で提案する遅延補償フリップフロップの具体的な動作と構造を解説する。さらに、第 4 章では動作時タイミングエラー検出、第 5 章では省電力特性に関して、類似の目的を持つ既存手法との比較を行いつつ、説明を行う。第 6 章では遅延補償フリップフロップのもつソフトエラー耐性を述べる。

第 7 章では、実際に遅延補償フリップフロップをプロセッサに応用するにあたって考慮すべき話題に関してのべる。

全体を通じての考察、まとめと展望などは第 8 章において述べることとする。

また第 3 章から第 6 章までは個々の章が遅延補償フリップと深く関連するが基本的には性質の異なるトピックを扱っている。このため、これらの章はさらにこまかく、研究の背景、提案手法がもつ特性、関連研究などの部分を個々に持つ構成となっている。

本論文の全体の構成のつながりを図1.2 として図示する。

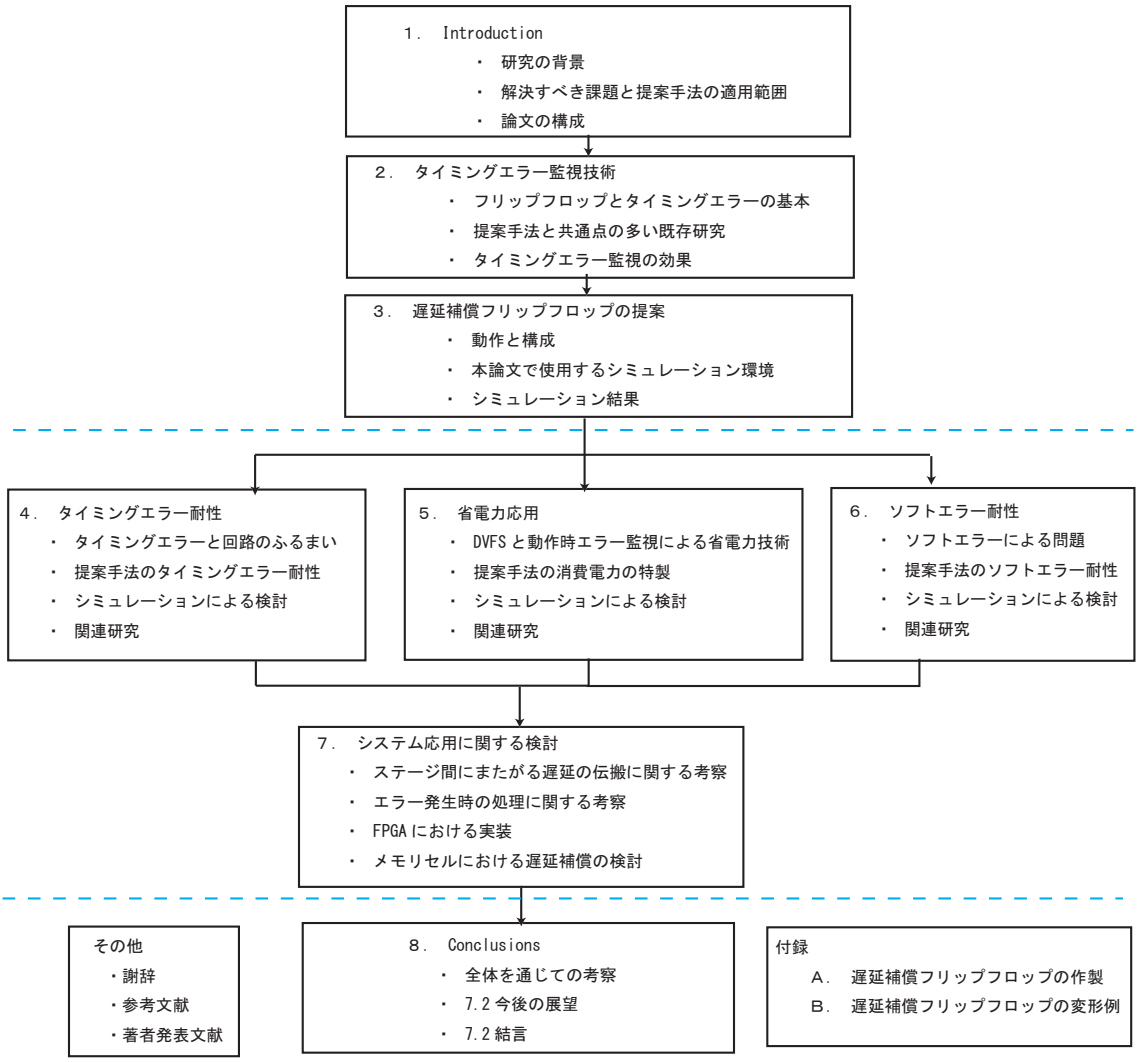

1．Introduction
・研究の背景
・解決すべき課題と提案手法の適用範囲
・論文の構成
2．タイミングエラー監視技術
・フリップフロップとタイミングエラーの基本
・提案手法と共通点の多い既存研究
・タイミングエラー監視の効果
3．遅延補償フリップフロップの提案
・動作と構成
・本論文で使用するシミュレーション環境
・シミュレーション結果
4．タイミングエラー耐性
・タイミングエラーと回路のふるまい
・提案手法のタイミングエラー耐性
・シミュレーションによる検討
・関連研究
5．省電力応用
・DVFS と動作時エラー監視による省電力技術
・提案手法の消費電力の特製
・シミュレーションによる検討
・関連研究
6．ソフトエラー耐性
・ソフトエラーによる問題
・提案手法のソフトエラー耐性
・シミュレーションによる検討
・関連研究
7．システム応用に関する検討
・ステージ間にまたがる遅延の伝搬に関する考察
・エラー発生時の処理に関する考察
・FPGA における実装
・メモリセルにおける遅延補償の検討
その他
・謝辞
・参考文献
・著者発表文献
8．Conclusions
・全体を通じての考察
・7.2 今後の展望
・7.2 結言
付録
A．遅延補償フリップフロップの作製
B．遅延補償フリップフロップの変形例

図 1.2: 本論文の構成

第2章

フリップフロップを基点とする既存の動作時エラー監視技術

本章では、フリップフロップの動作とタイミングエラーについての一般論について簡単に触れた後、提案手法同様にフリップフロップに何らかの工夫を施して動作時のタイミングエラー監視を行っている既存研究として Razor [8] [9] [10] と、カナリアフリップフロップ [18] という方式について説明する。この二つの関連研究は、提案手法と比較的類似性の高い既存技術そして論文全体を通して提案手法との比較対象に用いている。また、1 章で非常に簡単にふれた近年の集積回路が抱える課題のうち、動作時エラー監視技術や提案手法と特に密接にあるものについて、再度より詳細にその問題点と、どのようにこれらの技術と関連性をもつかについて述べる。

2.1　Flip-Flop とタイミングエラーに関する基本事項

本論文においては、全体を通じて Flip-Flop の動作と、タイミングエラーが考察の基点となる。そのため、まずフリップフロップの動作に関する基本的な事項を本節で整理する。フリップフロップは通常、回路全体の同期をとっているクロック信号の立ち上がりのタイミングに合わせて、前段のデータを取り込むと同時に取り込んだデータを保持し、保持したデータを後段に伝える働きをするものである。一般的には互いに逆相のクロックで動作するラッチを二つ直列につなげたような構成で実装されることが多い。ラッチとは、クロックなどの制御信号が High のときに入力されたデータをとりこみ、Low の間はそのデータを保持するものであるが、この実装方法には非常に多くの種類があるため、フリップフロップの実装方法も非常に多くの選択肢が考えられる。

本論文で扱うフリップフロップとしては、シンプルな構成である図 2.1 のようなものを扱うことを前提として評価を行う。これは PowerPC 603 などで用いられている、一般的なフリップフロップの実装法の一つである [27]。

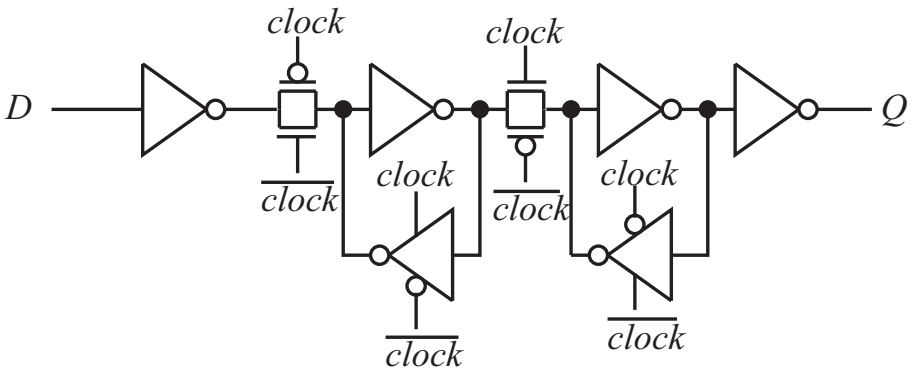
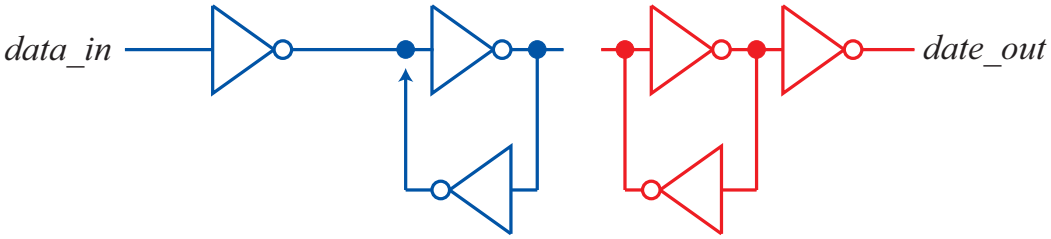


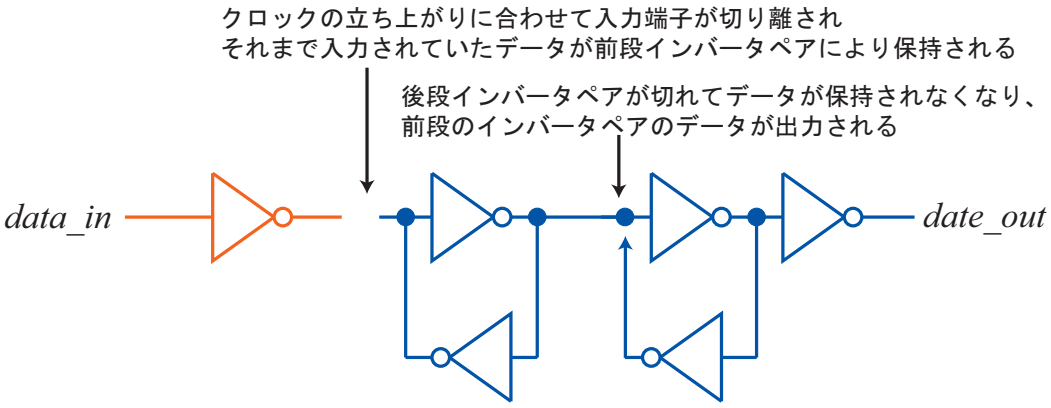
図 2.1: フリップフロップの基本構成

以下に動作の説明を加える。

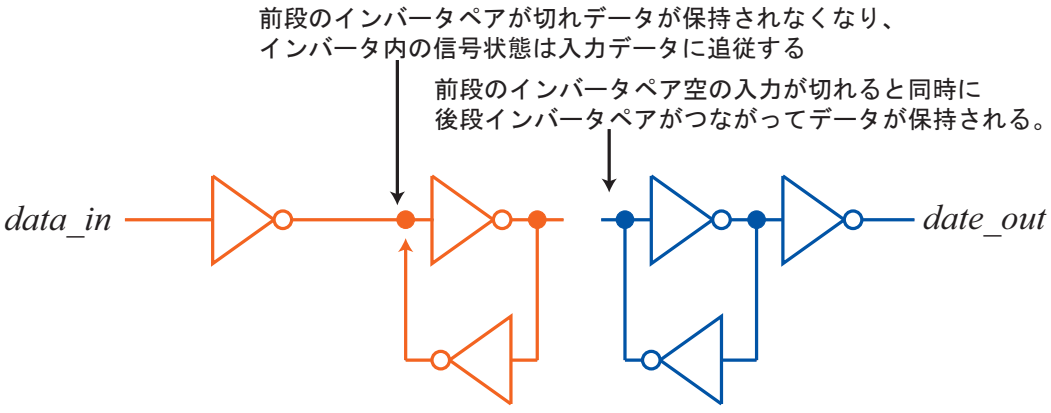
図 2.2 に示すように、ラッチを構成する 2 段のインバータの入力がクロックの立ち上がりのタイミングで切れて入力のインバータに対するドライブ能力がなくなると同時に、インバータペアがつながってそれまで入力されていたデータを保持するという点が動作の肝となっている。このため、クロックが変化する瞬間の内部状態はシビアなものであり、この間に入力が変化すると内部状態の正当性が保障されなくなってしまう。このため、フリップフロップではクロック信号の立ち上がりの瞬間に入力データは安定していなければいけないという制約が生じる。この条件を表すものとして、Setup time と hold time と呼ばれるものがあり、Setup time はクロックの立ち上がりよりどれだけ前に入力信号が安定していなければならないか、hold time はクロックの立ち上がりの後どれくらいの間入力信号が安定していなければならないかを表す指標となっている。



(1) Clock = 0



(2) Clock = 1



(3) Clock = 1

図 2.2: フリップフロップの動作の概念

つまり、通常動作においては、2 つのフリップフロップの間にある論理回路は、クロックの立ち上がりタイミングにおいて入力側のフリップフロップからデータが入力されると、その処理を次のクロック信号の立ち上がりよりも Setup time の分だけ早いタイミングまでに終わらせて、出力側のフリップフロップに渡さなければいけない。もし、処理がこのタイミングに間に合わなかった場合はフリップフロップに正しい値が取り込まれることが保証されず、タイミングエラーとなって、その後の回路動作の正当性がなくなる。また、これとは逆に処理が速く終了しすぎて、後段のクロックの hold time よりも前にデータが届いてしまうことがあり、これもタイミングエラーの一つである。以下、本論文では、この 2 種類のタイミングエラーを区別して、前者を過大遅延エラー、後者を過小遅延エラーと呼ぶことにする。

このようなタイミング設計は、回路設計における最も大事なステップの一つであるが、近年の回路動作周波数の劇的な高速化や、プロセスばらつきによる回路動作速度の揺らぎの顕在化等の影響によって、正確に設計することが大変難しくなってきている。

2.2　既存の動作時タイミングエラー監視フリップフロップ

提案手法の詳細について述べる前に、本節では同様にフリップフロップに工夫を凝らすことによって、動作時にタイミングエラーを検出する既存手法を示す。これらは、本論文を通じて、提案手法の特徴をより明確にするための比較対象としてもちいることとする。

2.2.1　Razor

Razor [8] [9] [10] ではタイミング違反を動作時に検出するために、タイミングがクリティカルになる部分のフリップフロップに、図 2.3 のようなシャドーラッチと呼ばれるものが用いられている。シャドーラッチには遅延クロックが供給されており、メインのフリップフロップとシャドーラッチとの間での保持されている値が異ならないかを動作時に監視する。もし、両者の内容が異なる場合は通常のクロックにデータの入力が間に合わずタイミング違反を起こしたということがわかる。

また Razor では DVFS(Dynamic Voltage and Frequency Scaling) との協調により、限界まで電源電圧を低下させ、低消費電力化を実現している。

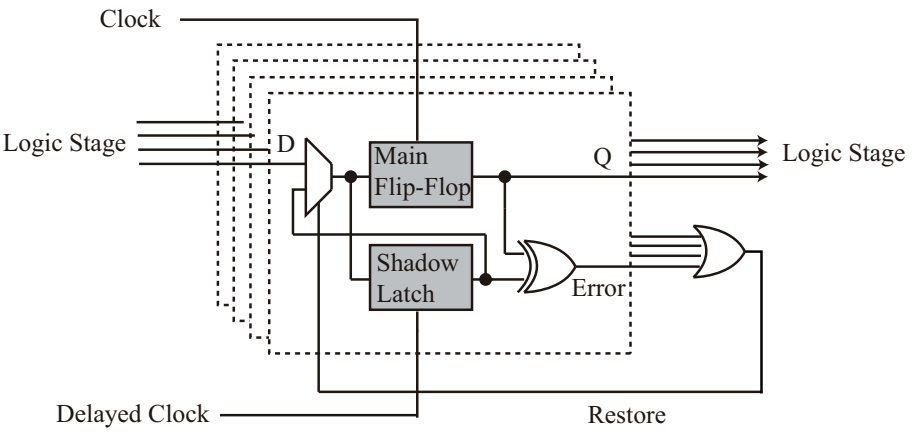


図 2.3: Razor Flip-Flop

ショートパス問題　前述のとおり、一般的なフリップフロップではクロックの立ち上がり前後にsetup time、と hold time と呼ばれるタイミング制約があり、正確な書き込みを保証するためこれらのタイミングの間では入力信号は安定していなければならない。つまり、前段の論理ステージでもっとも遅延が小さくなる

パスが活性化したとしても、後段のフリップフロップの hold time 以前にフリップフロップの入力にデータが到着してはいけない。このような最短パスの設計制限を min-delay constraint と呼ぶ。Razor では、順序回路設計時における min-delay constraint が非常に厳しくなる問題がありショートパス問題と呼ばれている。これは遅延クロックを使用していることが原因であり、遅延クロック側の hold time よりも後続データの到着を遅らせる必要があるからである。仮に遅延クロック側の hold time よりも後続データの到着が早くなってしまうと、shadow latch 側の値が本来書き込まれるべきデータの後続のデータとなってしまい、タイミングエラーの誤検出を起こすばかりか、本来 shadow latch に蓄えられている復旧すべきデータの破壊をも引き起こしてしまうことになる (図 2.4)。このような状態に陥った場合、メインフリップフロップとシャドーラッチの保持しているデータが異なるときはシャドーラッチが保持している値が常に正しい、とかていされている Razor では間違った値がメインフリップフロップに書き戻されてしまい、回復不能なエラーとなってしまう。

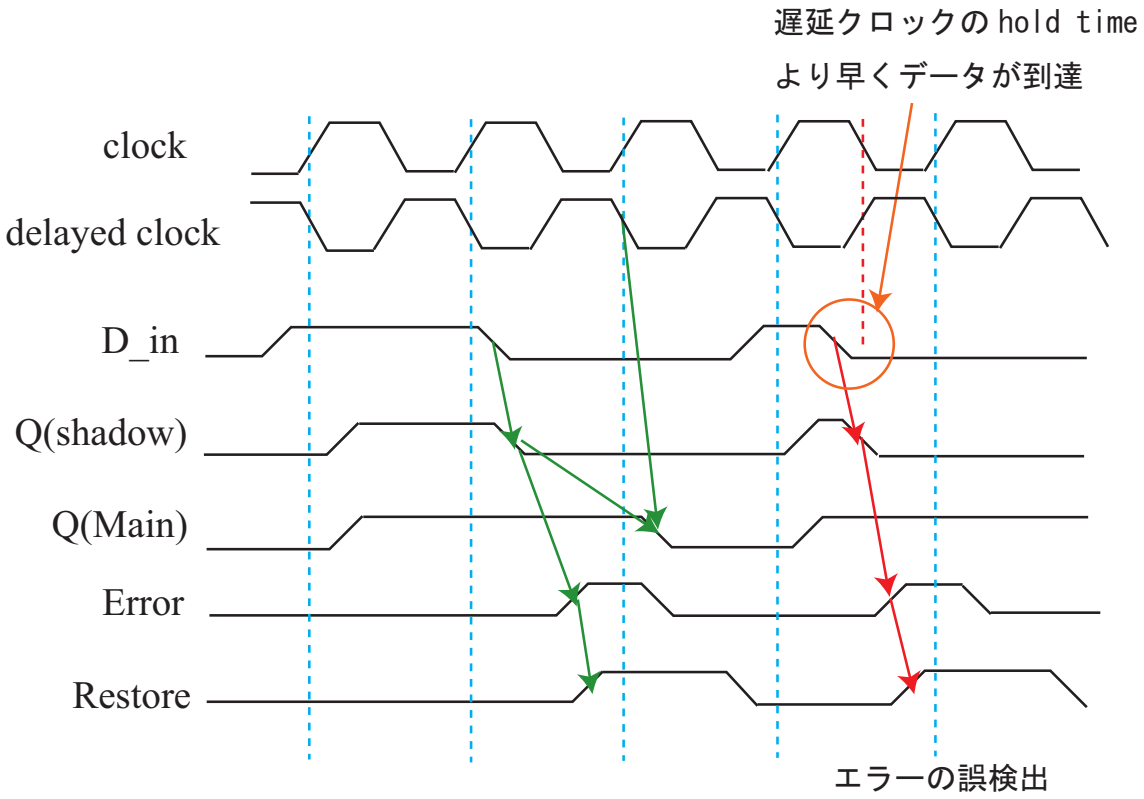


図 2.4: Conceptual timing diagrams.

Metastability の回避　また、Razor では Main Flip-Flop がエラーを実際に起こし、Shadow Latch が保持しているデータとの間に齟齬が生じるまでエラーは検出されない。このため、徐々にタイミングが遅れてくるような場合、Main のフリップフロップでは正常状態から、エラー状態にいたる過渡状態として、クロック信号とデータ信号が同時に動いてメインフリップフロップの値が中途半端な値で浮いてしまう状態、(Metastablity) に陥ってしまう可能性がある。このような状況になってしまうと、フリップフロップに保持されているデータとエラー検出の正しさが保証できなくなるため、Metastability を回避する機構が必要となる。

図 2.5 は Razor の実装の詳細を示しているが、図 2.3 に示される概念図と比較して特徴的な要素として、Metastability Detector がある。Metastability Detector では Skew の異なる 2 つのインバータをペアにし、入力が VDD/2 程度になったとき出力が後段のインバータを逆方向に駆動するようにしている（図 2.6）。

このため、後段のインバータの出力が一致していないときはインバータの入力が metastable であるということが検出できる。

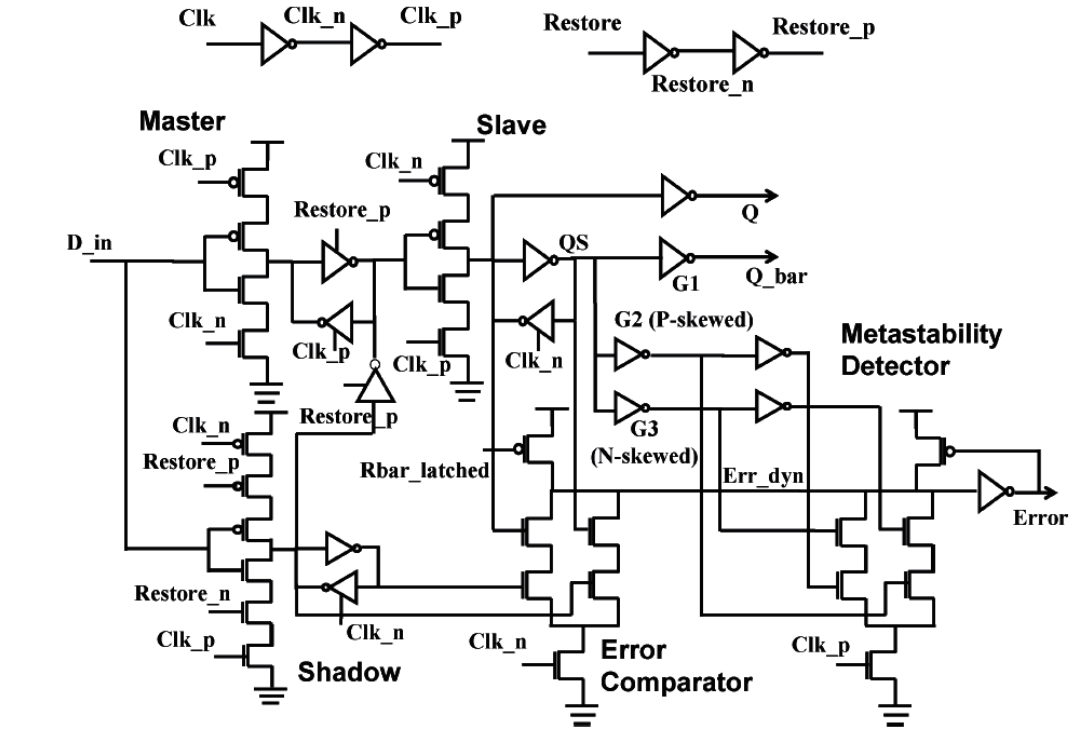
タイミングエラーからの回復機構　通常タイミングエラーは最も長いパスに連結されているフリップフロップで起こることが考えられるが、どの 1 ビットがエラーを起こしても 1 つのデータを表すすべてのビットはそれに同期した挙動を示す必要がある。このため各ビットのエラー信号の論理和をとって描き戻し信号の生成に使用することで、処理されているデータのなかで 1 ビットだけが遅れるなどといったエラーが起こらないようにしている。このために必要な回路を図 2.7 に示す。Razor では高速に多ビットのエラー信号の論理和を取るためにダイナミック論理が使用されていることがわかる。このような一般的にドミノ論理と呼ばれるようなタイプのダイナミック回路では CMOS 論理で実装すると p MOS が大量に直列に挿入される多ビットの論理和を演算する場合の実装に動作速度とハードウェア量の点から優れるが、1 ビットでも入力が '1' になるたびに、多くの nMOS トランジスタにつながっているノードの充放電がおこるため、頻繁に回路が動作する場合には消費電力が大きくなりがちであるという欠点がある。しかしながら、Razor のエラー検出信号の頻度はそう多くない（供給電圧を過度に低くして省電力を狙っても、エラーによる再実行のオーバーヘッドが増えて結局は消費電力が増加してしまうため、通常はエラー率は非常に小さい状態に電源電圧が調整される）ため、ダイナミック回路の利点のみが生かされている。

また、第 1 章でも少しふれたが、近年の集積回路では素子の動作やクロックの高速化に伴って、相対的に配線遅延が増大している。つまり、1 クロックの間に配船中を信号が伝搬できる距離が短くなってきている。このため、ある部分でエラーが生じたとした場合、正しい値をシャドーラッチからメインフリップフロップに書き戻しを行っているサイクルでは、回路全体が後段にデータを送らないようにストールしなければならないが、チップ全体にストールを指示する信号を瞬時に供給することは難しくなってきている。このような、近年の高周波数動作集積回路にも対応できるように Razor では、カウンターフローパイプライン [16] [17] を利用した回復機構を用いてエラーを回復させる手法も紹介されている（図 2.8）。この回復手法は回路のスケーラビリティを考慮した点で優れた手法であるが、逆に非常に複雑なエラー回復機構を必要とするという欠点もある。

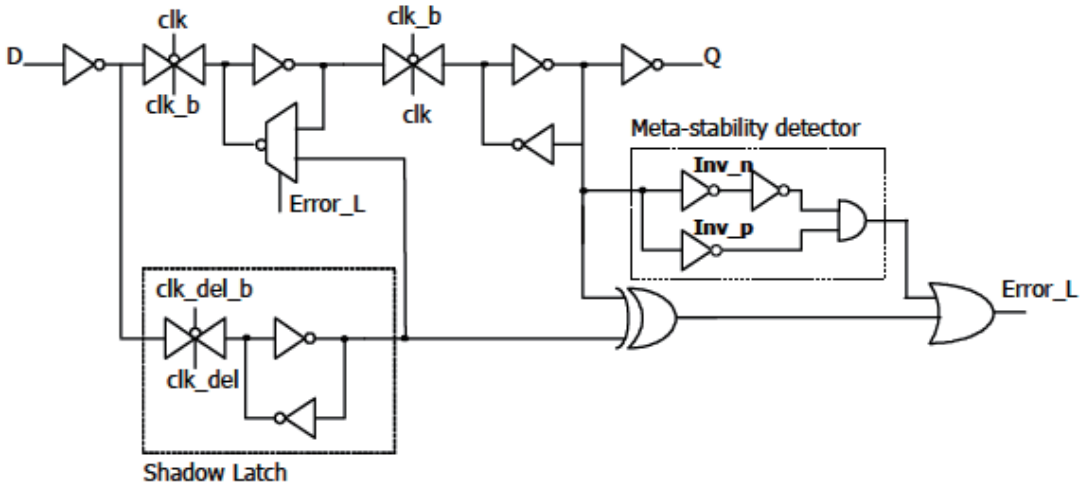
2.2.2　カナリア flip-flop

このようなショートパス問題を回避する方法としてカナリア方式 [18] と呼ばれるものがある。この方式では図 2.9 のように冗長化したフリップフロップの片側の前に遅延素子を入れる。これによって、入力が遅れた側のフリップフロップのほうが先にタイミングエラーを発生することになるため、2 つのフリップフロップの値の整合性を監視していれば、メインのフリップフロップでエラーが発生する直前にエラーを予測することができる。遅延クロックを用いていないためショートパス問題を防ぐことができる上に、エラーを発生前に検知することにより複雑なエラー回復機構が必要なくなるという利点がある。その反面、最大遅延に対する許容範囲は通常のフリップフロップより狭くなるため、電源電圧降下能力や、クロックの高周波数化による処理の高性能化などを志向する場合には、Razor に劣るということもできる。

このカナリアフリップフロップでは図 2.10 に示すような回路テスト用のスキャン回路 [15] を有効利用して実装する方法がていあんされている。図 2.11 に示すような実装を用いることによって、通常の回路動作時には従来役割のなかったスキャンフリップフロップを利用し、カナリアフリップフロップを利用することによる面積オーバーヘッドを抑えることができる。



(a) Razor Flip-Flop の回路図



(b) 回路図（a）のブロック図

図 2.5: Razor Flip-Flop の実装の詳細 （文献 [10] より）

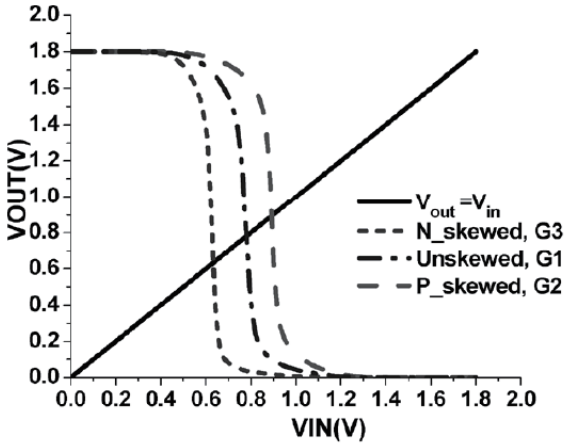


図 2.6: Metastability Detector で利用されている Skew の異なるインバータペアの挙動 （文献 [10] より）

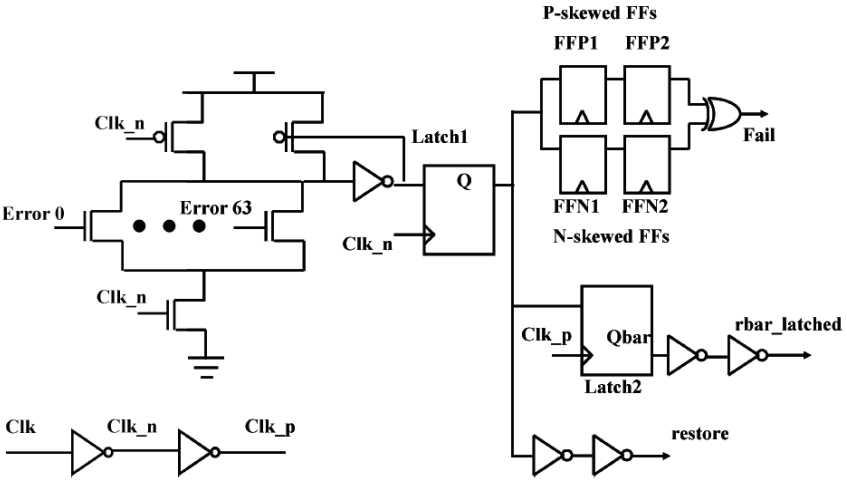


図 2.7: Razor の restore 信号生成機構 （文献 [10] より）

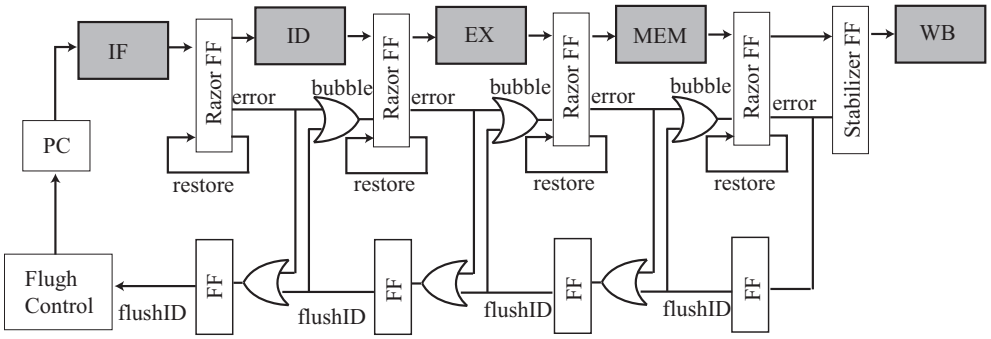


図 2.8: カウンターフローパイプラインを利用したデータ回復機構

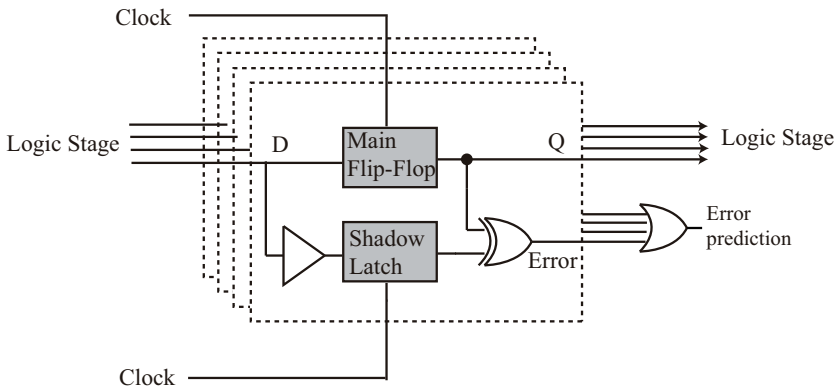


図 2.9: Canary Flip-Flop

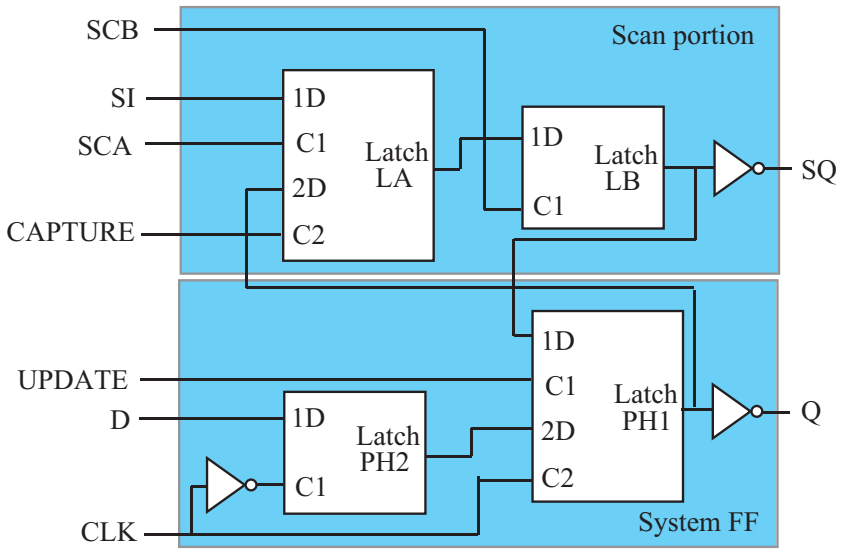


図 2.10: Scan Flip-Flop

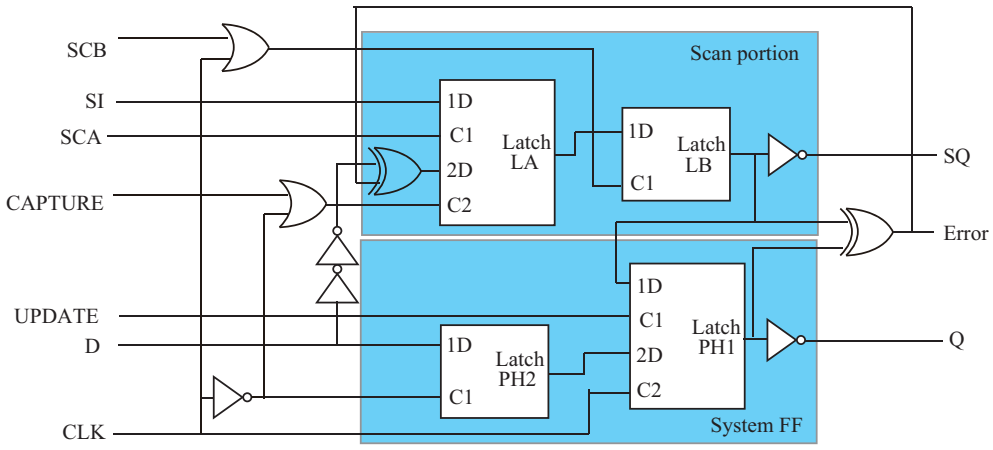


図 2.11: Canary Flip-Flop implemented with Scan Flip-Flop

2.3　近年の集積回路のもつ課題と動作時タイミングエラー監視の効果

半導体素子の極端な微細化に伴って表面化してくる代表的な問題としては第一章に挙げたようなものがあるが、このうち動作時タイミングエラー監視を行いながら DVFS を用いることによって改善が見込まれるものについて、より詳細に述べる。

2.3.1　素子性能のばらつきによる問題の軽減

半導体産業の劇的な進歩の裏には、薄膜作製・リソグラフィ・エッチングという集積回路作製の基本サイクルによって、製品が pre-assembled な状態で製造されるという特性があったと判断できる。このようなウエハ全体のパタンを一括で作製するという方法でなければ、配線などに膨大な時間がかかってしまうため、集積回路はこれほどまでに普及しなかったであろう。しかしながら、ウエハ全体に一括で製造処理を行うといっても、微視的には膜厚・プロセス液の濃度・不純物の濃度などの確率的なばらつきは当然避けられないものである。たとえば、原子直径がおよそ 4 Å の Si に対して 32nm 以降のプロセスではトランジスタのサイズが原子数十個というオーダーで表されるようになる。そのような状況ではトランジスタ中の不純物の個数などは場所によって確率的に大きく変動することになり、図 2.12 に示すように素子の性能にばらつきが生じるようになってしまう [5] [6] [7]。極端に微細化された集積回路の設計においてはこのような素子のばらつきも考慮の対象として外すことのできない要素となっている。このような確率的に素子の性能がばらつくランダムばらつきのほかにも、製造装置の特性や、作成されるパタンの密度などによって、ウエハ上の位置的に一定の傾向をもってばらつきが生じるシステマチックばらつきと呼ばれるものも問題となっている。動作時にエラーを監視しながらそれに基づいて電源電圧などの動作条件を変更するということは、製造後の回路がここの性質、動作環境に合わせて自律的に動作条件を調整するとみなせる。したがって、個々の回路が持つ性能のばらつきは一定の範囲内で隠ぺいされる効果がある。

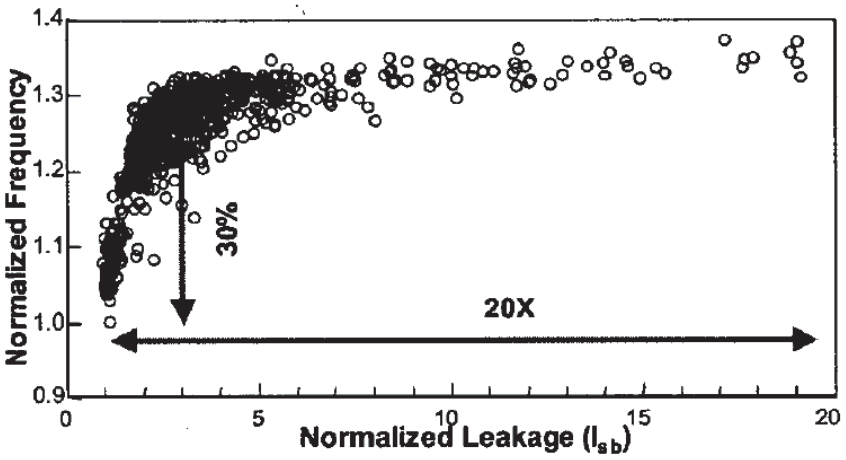


図 2.12: Leakage and frequency variations [6]

2.3.2　リークによる消費電力の増大

集積回路を構成するトランジスタのサイズが微細化し、ゲート長が短くなると、スレッショルド電圧のロールオフ、S パラメータの劣化、Drain Induced Barrier Lowering (DIBL) といった短チャネル効果を抑えることが難しくなってくる [19] [20] 。この結果、トランジスタが OFF 状態であっても流れてしまうリーク電流が大きくなってしまう。実際に、図 2.13 に示すように、待機時のトランジスタのリーク電流は、素子の微細化に伴って指数関数的に増加しており、リーク消費電力が無視できなくなってきている [3]。省電

力化に関する検討を行っている第 5 章において関連研究は詳述するが、リークによる消費電力を抑えるための研究は非常に多くなされてきており、電源電圧を低くする手法・高速大リークと低速少リーク 2 種類のトランジスタを用意する方法・基板バイアスをコントロールしトランジスタの閾値をコントロールする方法 [21]・スリープトランジスタ、シャットダウントランジスタなどと呼ばれるトランジスタを挿入し利用し、チップ上でアクティブでない部分の電源を遮断する方法などがある。また、通常の MOS FET ではなく、リーク電力の抑制効果が大きい FIN FET などの複雑な形状のトランジスタを用いるなど、デバイス面からの研究も多い。しかしながら、性質の異なるトランジスタを用意する方法は製造工程が複雑になったり、演算には直接関係のないスリープトランジスタを挿入することで性能が低下するなどの影響が指摘されている。また、基板バイアス効果による閾値の最適化に関しても、短チャネル効果や、基板不純物濃度の低下が顕著になる世代のプロセスでは効果が小さくなるといわれている [4]。

動作時にエラーを監視しながら電源電圧を調整できるということは、その時その時の環境と素子特性にあった電源電圧の最適化が行えるということである。これは、設計段階で見積もられたマージンの範囲よりも、よりアグレッシブに電源電圧を降下させ消費電力の削減に寄与できる可能性がある。たとえば、多くの場合では、回路は設計段階で想定されたマージンの最悪条件で動作していないため、設計段階で設定された動作電圧よりも低い電源電圧で処理を行っても、問題のないことが多いからである。

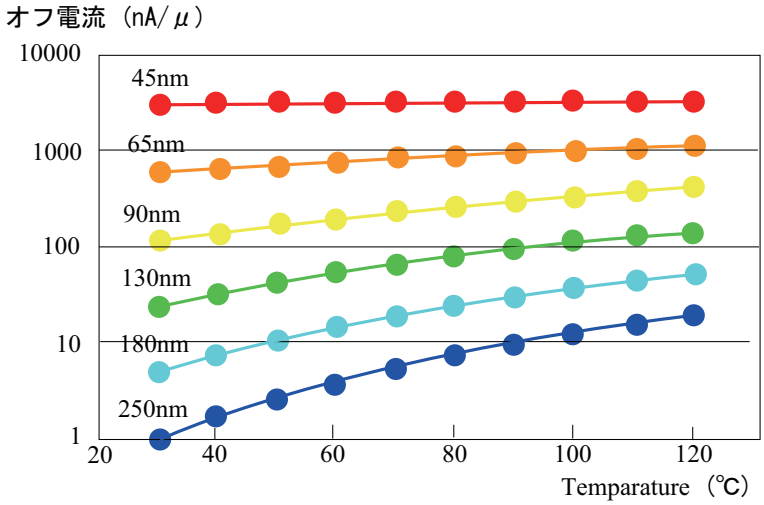


図 2.13: 各プロセス世代におけるリーク電流の傾向 （[4] より作成）

2.3.3　低電圧化によるノイズ耐性の減少

半導体集積回路はその用途を拡大し、携帯電話や PDA（Personal Degital Asisstants）など多くの携帯機器に用いられるようになってきており、このような携帯機器においてはバッテリーによる連続駆動時間が重要な性能指標の一つとなりうる。このような、近年の省エネルギー化に対する要求や、集積度向上による電力密度の上昇、さらにはスケーリング則から導かれる理論的な最適動作電圧の問題などから、集積回路の電源電圧は低下傾向にある。これにより、GND（接地電位）から VDD（電源の電位）までの連続的な値を利用するアナログ回路では、ノイズに弱くなってきているだけでなく、十分な階調が確保できないため必要な演算精度を達成することが難しくなってきている。さらに、デジタル回路でさえも、電源電圧の揺らぎやソフトエラーなどさまざまなノイズ対する脆弱性が増してきているなど、外乱に対する脆弱性は大きな問題となってきている。プロセスばらつきや外乱の影響を受けやすい SRAM セルなどでは電源電圧の下降傾向は近い将来終焉に向かうだろうといわれており、このような脆弱性に対するさまざまな対策技術が講じられている。

表 2.1: 動作環境によって、要求される動作マージンの一例 [22]

Corner	Voltage	Temparature
Fast	1.98 V	0 ° C
Typical	1.8 V	70 ° C
Slow	1.62 V	125 ° C

外乱によって発生するノイズに由来する問題として典型的なものに、ソフトエラーがある。ソフトエラーとは、宇宙線などの荷電粒子が半導体に入射した場合に発生する電気的なノイズが原因となるエラーである。前節で挙げたような既存の動作時タイミングエラー監視技術ではこのエラーに対する信頼性は通常のフリップフロップとほとんど変わらないが、本論文で提案している遅延補償フリップフロップは既存のタイミング鰓検出フリップフロップと比べて優れたソフトエラー耐性を持つ。遅延補償フリップフロップのソフトエラー耐性については第 6 章で詳述する。

2.3.4　動作を保証するためのマージンの増大と設計難化の抑制

上述したいくつかの内容とも関係することであるが、素子微細化・電源電圧低下などによって、電源揺らぎや温度揺らぎなどの影響が相対的に大きくなってきていることや、プロセスばらつきによって、同じように設計・製造を行ったトランジスタの性能にもばらつきが生じているということは、すでに述べたとおりである。このようなトランジスタ特性のばらつきは、同一ダイ内の各部の性能ばらつき、あるいはダイ間、ウエハ間の性能のばらつきとして顕在化し、これらの個々の性能を正確に予測することは非常に困難である。このため、設計時にとられるマージンは、動作環境などからの外乱に対するもの、ばらつきを考慮したものなど多岐にわたり、非常に大きいものとなっている。一例として表2.1 に、どの程度のマージンがデザイン時に必要になってくるかをしめす。

この表2.1 は商用の集積回路が満たすべきとされているものであるが、当然ながら、航空・宇宙用途で用いられているものなどは、より過酷な環境下での動作が保証される必要がある。設計時には、このうち最悪なケースである、動作温度も高く電源電圧も低いという状況でも回路が動作するように考えなければならない。さらに、実際にはこれだけでなく、ノイズによる影響、プロセスばらつきの影響などを多くの条件を考慮して、そのそれぞれの要素が最も悪条件でも、回路は正常に動作するように設計されなければならない。このような悲観的な見積もりに基づいてマージンを設定した設計は、外乱による悪影響を楽観的に見積もった場合と比べると、かなり無駄に性能を抑制させてしまうことになることは、想像に難くない。

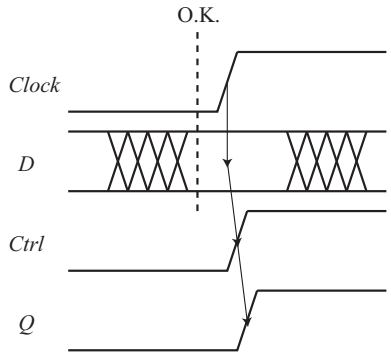
また、近年製造されているような集積回路では、高集積化によって利用可能な素子数などがもはや人手によって現実的な期間で設計できる規模をはるかにこえており、CAD（Computer Aided Design）による設計支援が必要不可欠になっている。Verilog、VHDL や System C などの高位のハードウェア記述言語で動作を記述し、自動的にレイアウトまで合成するといったツールもあり非常に生産性の高い環境が、CAD によってもたらされているといえる。しかしながら、そのようなコンピュータの力をかりた設計の自動化の発展の速度よりも、半導体集積回路の集積度の向上速度は速く、利用可能な素子数にたいして（種々の制約化で利用可能な人・物・時間で）設計可能な回路の規模が追いついていない現状がある。回路規模が年率約 58% で増大しているにもかかわらず、設計生産性は年率約 21% 程度の向上にとどまっており、両者のギャップは年率約 37% で拡大している。このような事態は “VLSI 設計の危機” などとよばれ、近年顕在化してきている問題である。

動作時エラー監視と DVFS の協調したシステムにおいては、個々のチップの性能や、動作環境に合わせて最適な電源電圧・動作周波数が動的に選ばれることや、仮に設計を楽観的に見積もりすぎていて予期せ

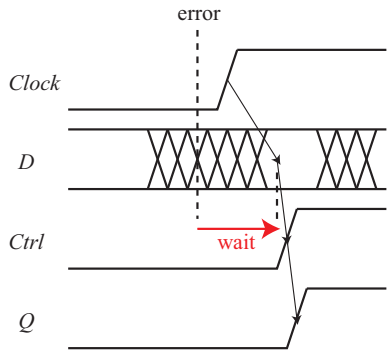
ぬエラーが発生したとしても、適切な回復機構を用いることで、クリティカルなエラーに陥ることなく処理を継続できることなどから、これらの設計・検証を容易にし、必要とされるマージンを削減することができる可能性がある。

第3章

Delay-Adaptation Flip-Flop



(a) 通常時



(b) タイミングエラー発生時 r

図 3.1: Ctrl 信号のふるまい

本章では、本論文での提案手法である遅延補償フリップフロップの動作、構成、特徴について説明を行う。また、本論文を通して用いた、シミュレーションによる評価環境についての説明を行い、実際に遅延補償フリップフロップの動作について回路シミュレーションを行った結果を示す。

3.1　遅延補償フリップフロップの動作

本研究ではクロックの立ち上がりの前後にデータ信号が変化してしまった場合をエラーとして検出するような、遅延補償フリップフロップという記憶素子を提案する。遅延補償フリップフロップは入力されるクロック信号と、データ信号の遷移のタイミングを常に監視し、クロックの立ち上がり遷移タイミングの前後でデータ信号が変動しない場合は通常状態、クロック信号立ち上がりとほぼ同時にデータ信号が変動した場合は、タイミングエラーとみなす。また遅延補償フリップフロップでは、システム全体の同期をとっているクロック信号とは別に、個々のフリップフロップのデータの取り込みを実際に制御する信号をフリップフロップごと生成する。以下ではこのフリップフロップ制御信号を Ctrl 信号とよぶこととする。

図 3.1 のように Ctrl 信号は通常動作時にはクロックと全く同じ挙動をするが、入力データの到着が遅れ、タイミングエラー検出状態になると、当該フリップフロップに入力されているデータが安定するまで信号の立ち上がりを遅らせる。こうすることによって、フリップフロップはメタスタビリティ状態などになることなく、正しい値が取り込まれる。フリップフロップによるデータの取り込みを後ろにずらした場合、後段のステージのデータは当然遅れて出発するため、当該フリップフロップ前段で発生した遅延は、そのまま後段に伝搬してしまうことはさけられない。しかしながら、このようなタイミングクリティカルなパス

が実際に活性化する可能性は低いといわれている [23] ため、後段の回路の最長パスが実際に活性化する可能性は低い。このため、高確率でこの遅延は後段のロジックステージで吸収されることになる。したがって、従来手法ではエラーが発生した場合、何らかの回復手法がとられることになるのが通常であるため処理のオーバーヘッドが生じていたが、本手法では処理オーバーヘッドの発生を抑制することができる。この手法は、回路動作時にあるステージの遅延を、後段のタイミングマージンで動的に補償することが可能であるという点で非常に革新的であるといえる。なお、種々の大きさの遅延が発生した場合の回路の挙動の概念図を図 3.2、 3.3、 3.4 に示す。

3.2　遅延補償フリップフロップの構成

以下に遅延補償フリップフロップの構成をより詳細に述べる。提案するフリップフロップは PowerPC 603 などで用いられているような基本的な構成の FlipFlop [27] に過大遅延エラー検出機構、フリップフロップ制御信号調整機構、過小遅延エラー検出機構を設けたものである。また、これらの追加される各機構では、信号遷移監視機構が使用されている。

3.2.1　信号遷移監視機構

遅延補償フリップフロップでは、信号の変化のタイミングを直接監視する必要があるため、信号遷移検出回路が必要となる。従来、信号の遷移検出では、対象となる信号とその信号を少し遅延させたものの XOR を取るという構成が一般的である。しかしながら本研究では、この信号遷移監視に基づいてフリップフロップ制御信号をクロックから生成しているためなるべく高速に反応するエッジ検出回路を用いることが望ましいといえる。

そこで、ＲＦ集積回路で高周波の発信信号を発生させる用途に用いられている、パルスジェネレータ回路の考え方を利用した [28]。これは、ある入力信号の立ち上がりのタイミングで、インバータ遅延の間だけパルスを立てるというものであり、非常に急峻に入力信号の変化に対応する（図 3.6）。こうして、入力信号の変化を示す信号とクロックの立ち上がり方向の変化を示す信号の AND をとることで限界状態を検出できることになる。なお、立下り側はインバータをはさんで入力をしているため、インバータ一段分無駄なマージンが発生していることになるが、図 3.6 の回路と相補的な回路を用意するよりも最終的に得られる波形の応答などが優れているという結果がシミュレーションから導かれた、また、立ち上がり、立ち下がり両方の遷移を監視する必要があるデータ入力信号については図 3.7 のような双方向の信号遷移検出を可能とする回路を用いた。なお、電子回路シミュレータによる信号遷移検出回路の動作波形シミュレーションの結果を図 3.8 に示す（このときに用いたシミュレーション環境は次節で詳述する）。

3.2.2　過大遅延エラー検出機構

過大遅延エラー検出機構においては、クロック信号の立ち上がりタイミングを表すパルスと、入力データの立ち上がり立下りタイミングを表すパルスの AND を取ることで、クロックと入力データが同時に動いてないかを監視する。この AND 論理の出力信号を Detect 信号とする。なお、Setup 制約を満たすために、入力データの信号遷移監視機構のパルス持続時間は最低でも Setup time と上記 AND 回路の遅延の和よりも少し長い必要がある。エラー検出信号に関しては、Ｒａｚｏｒなどと同様にダイナミック回路をもちいて全ビットでの論理和をとっている。

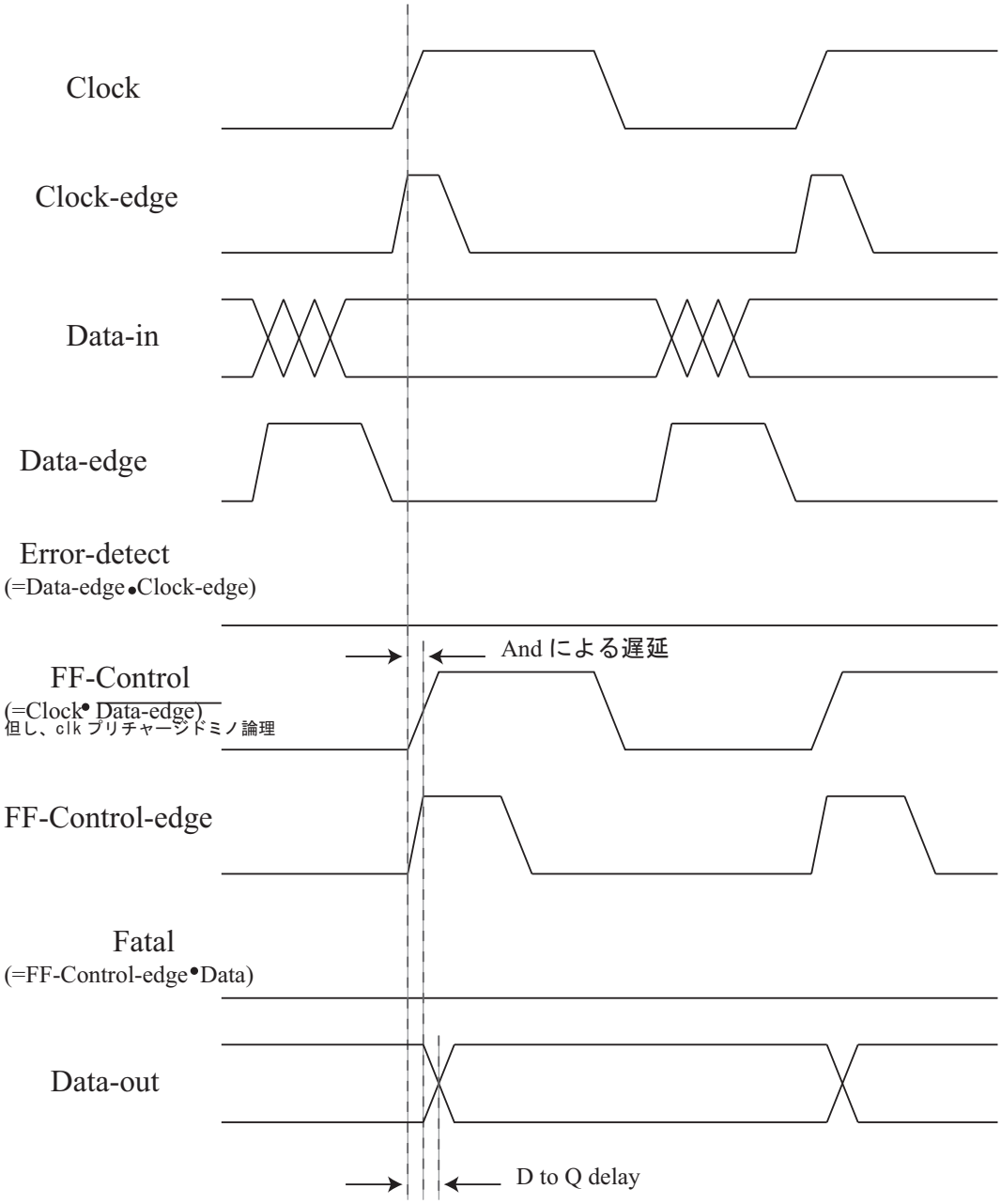


図 3.2: 遅延補償フリップフロップの通常動作時の挙動。タイミング制約に余裕がある場合は通常のフリップフロップとほぼ同様の挙動を示す。

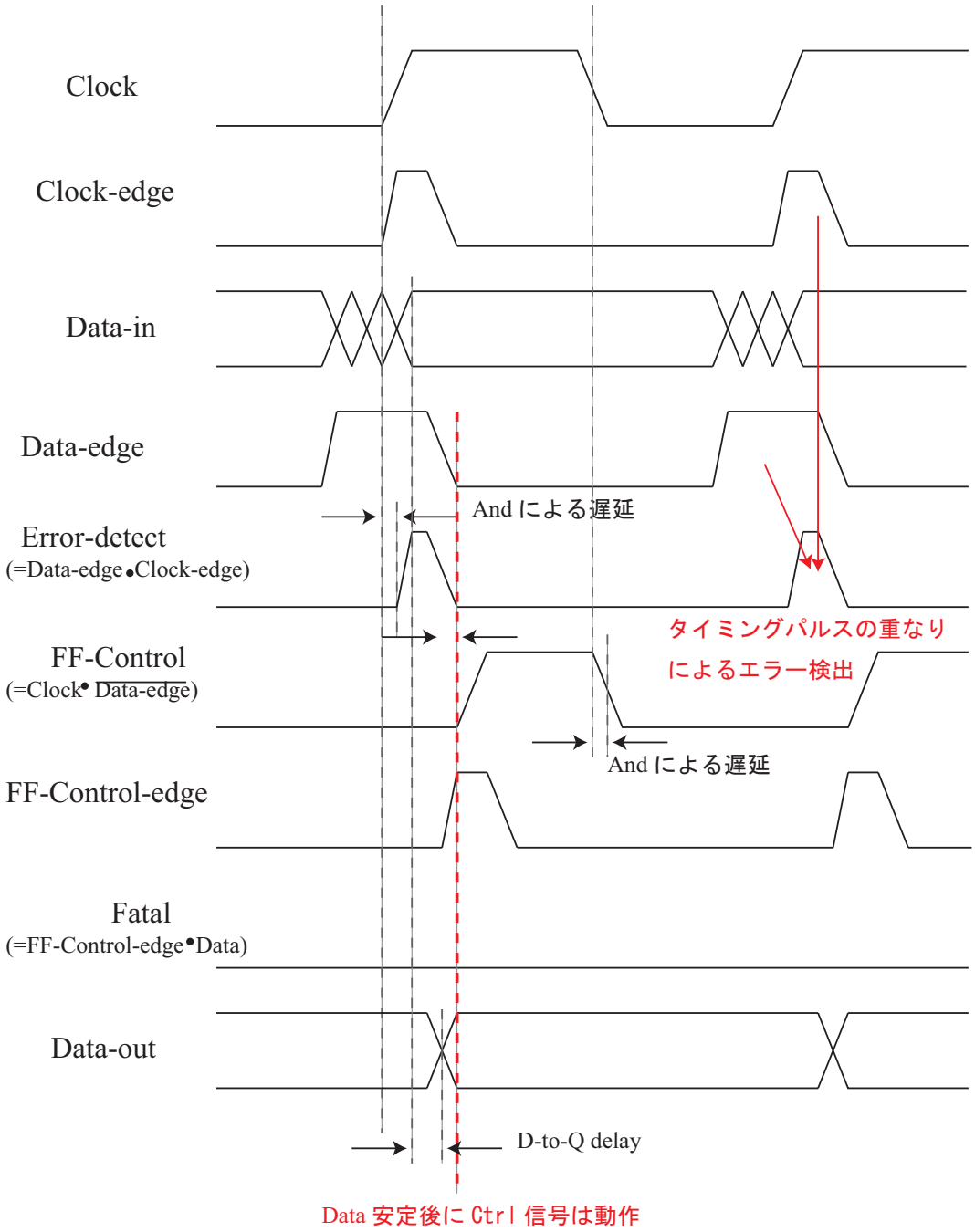


図 3.3: 遅延補償フリップフロップの過大遅延発生時の挙動。データの遅れがあり、クロックの立ち上がりタイミングになってもデータが安定していない場合には、それぞれのデータの変動を示すパルス信号の重なりから、タイミングエラーが発生したと検出する。また、入力データが安定するまでフリップフロップ制御信号の立ち上がりを遅らせることで、正しい値を保持するようにする。

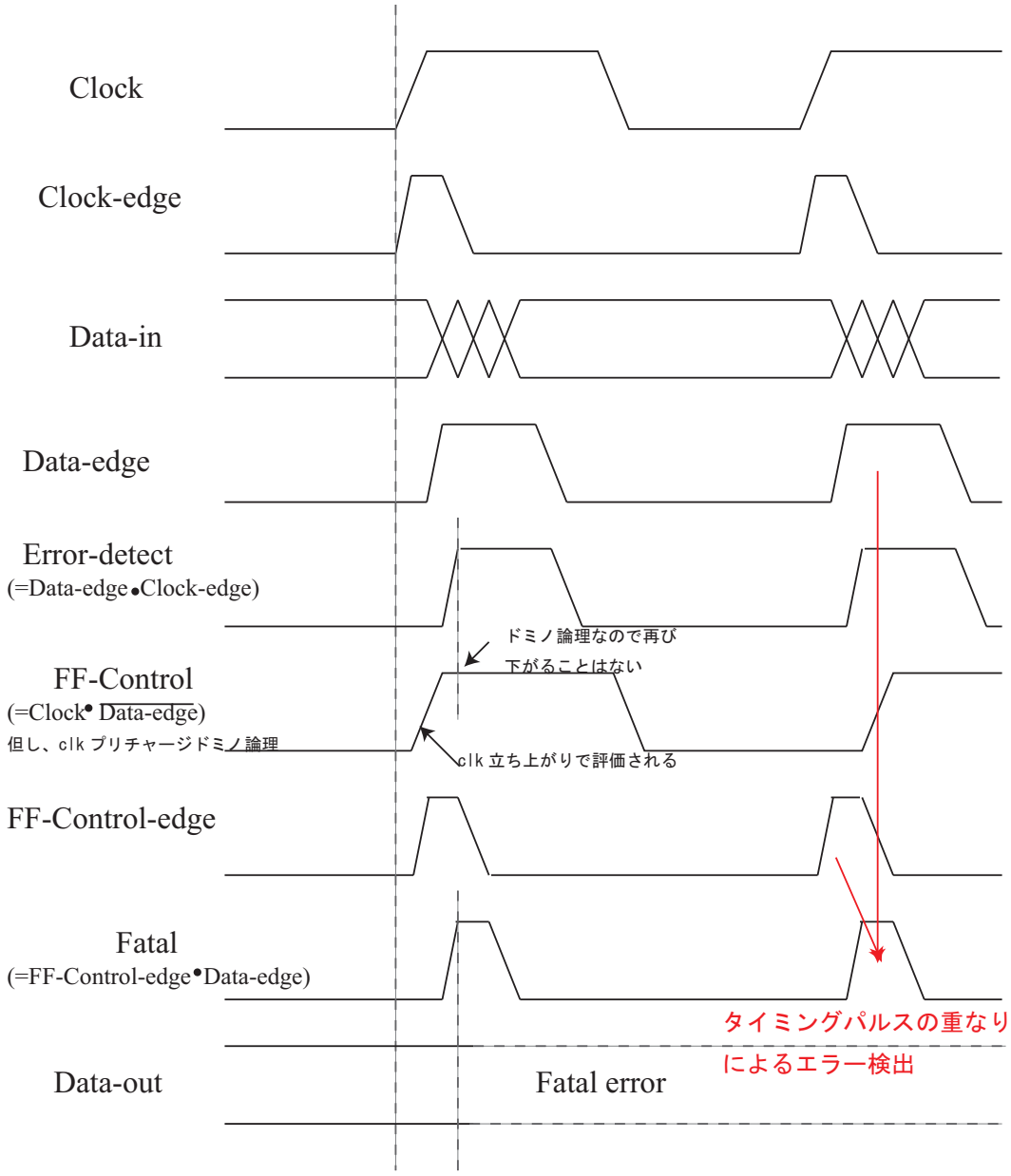


図 3.4: 過小遅延が発生した場合の遅延補償フリップフロップの挙動。フリップるフロップ制御信号が立ち上がって十分に安定するまでにデータの変動が起こった場合には、最短パス遅延設計制約が満たされなかった可能性があるとして、検出する。注意すべきはこの状態での波形は、補償不可能なほど大きい遅延が発生した場合と同様になる点である。

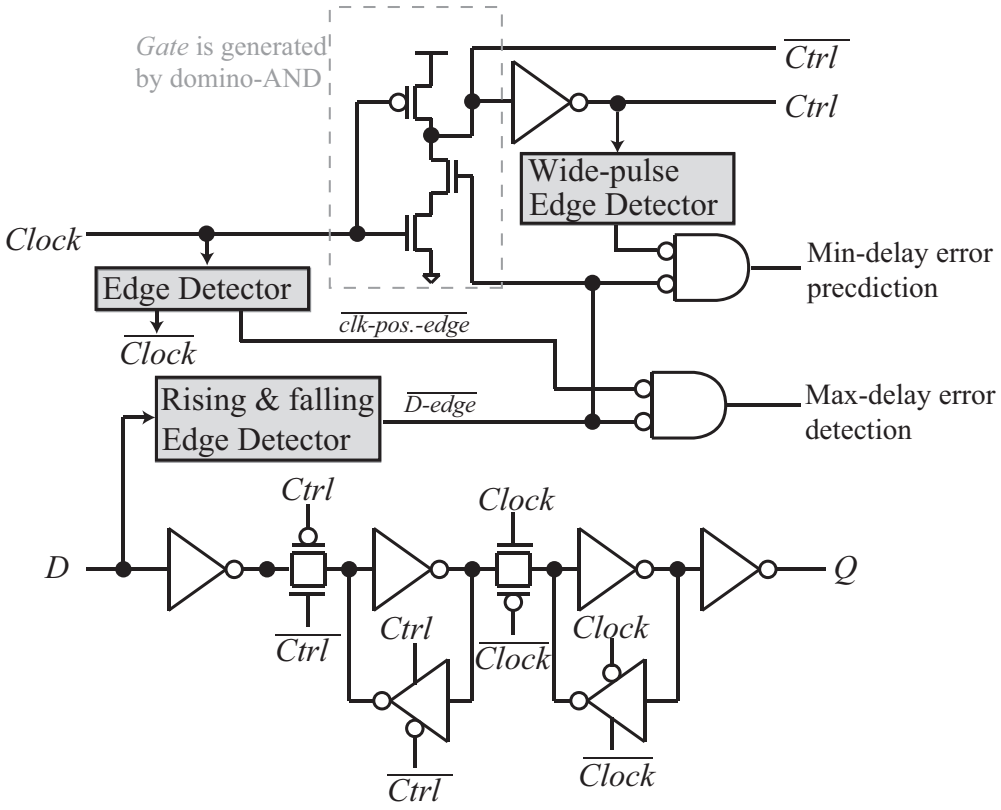


図 3.5: Edge-detector based on RF pulse generator

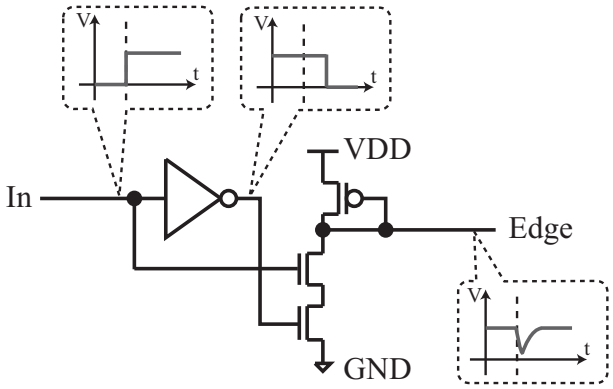


図 3.6: Rising-edge detector based on RF pulse generator

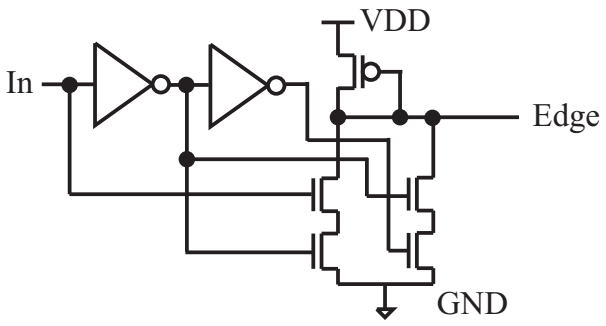


図 3.7: Rising-and-falling edge detector based on RF pulse generator

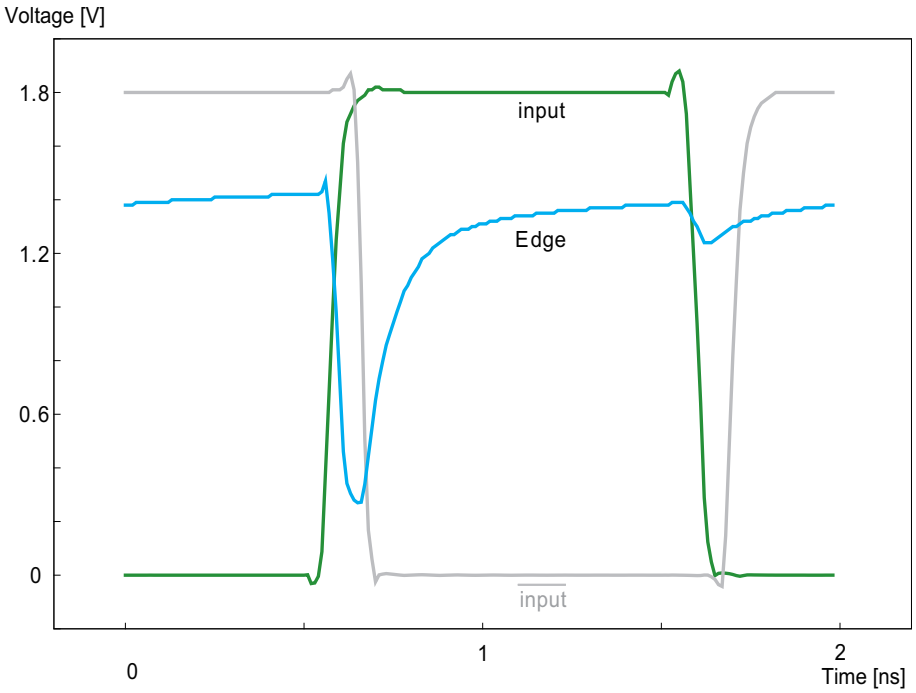


図 3.8: 信号遷移検出機構の動作波形

3.2.3　フリップフロップ制御信号調整機構

上記過大遅延エラー検出機構で生成された Detect 信号に基づいて実際にフリップフロップを制御する信号（以下 Ctrl 信号と呼ぶ）の立ち上がりタイミングを調整する。Ctrl 信号はクロック信号と Data 遷移信号の負論理との論理積によって生成する。たとえば過大遅延エラーが発生していないときは Data 遷移信号の負論理の値は常に 1 であるため、クロックの立ち上がりとほぼ同じタイミングで Ctrl 信号は立ち上がる。しかしながら過大遅延エラーが発生している間は、Data 遷移信号の負論理は 0 であるため、データが安定するまで Ctrl 信号は立ち上がることができない。このため、ふるまいとしては、データの安定を待ってからフリップフロップがデータを取り込むということになるため、フリップフロップが Meta-stable な状態になるなどして、書き込まれた値の正当性が失われることを回避できある。

ただし、データの遅延が大きすぎて、クロックのエッジまでにデータが変化しなかった場合も Data-edge 信号が 0 となってしまい、Ctrl 信号は立ち上がってしまう。このような場合は、フリップフロップに保存されているデータの正しさが保障されないので、単純に後段に遅延を伝搬させて補償することができない。このような場合の処理については次節で延べる。

3.2.4　過小遅延エラー検出機構

クロック信号と同様に Ctrl 信号の立ち上がり遷移タイミングも監視を行う。Ctrl の立ち上がりタイミングパルスとデータの遷移タイミングパルスの論理積を取ることによって、hold time 制約の違反が検出できることになる。この論理積の出力を Fatal 信号と呼ぶ。つまり、hold time 制約の確認を行うために、Ctrl 信号の立ち上がりタイミングを示すパルスは、hold time の長さと AND 論理の遅延分より長い必要があることがわかる。実際には hold time の制約が完全に破られる前にエラーが検出できるように、Ctrl 信号の遷移検出パルその幅を調節しておくことで、実際に過小遅延エラーが起こる前に、電源電圧の調整などの処理を行うことが可能となる。こちらの過小エラー予測信号に関しても、同様にダイナミック回路をもちいて全ビットでの論理和をとっている。

3.2.5　過小遅延エラー検出機構と補償不能な遅延エラーの取扱い

遅延補償フリップフロップでは、基本的にある程度より大きな遅延が発生しそうな場合には、DVFS などの既存技術と協調して電源電圧や周波数をコントロールすることによって、正常に近い動作状態をさせるようにすることを前提としている。非常に大きな遅延が起った場合にも対応できるように設計することも可能であるが、この場合は遅延補償フリップフロップだけではなく、別に回復機構が必要になる。また、このような場合に注意しなければいけない点は過小遅延エラー検出機構は補償不能なほどおおきな過大遅延エラーを同時に検出することである。上述したように、データの変化がクロックの立ち上がりまでに開始しないと、Ctrl 信号が立ち上がってしまう。このとき、Ctrl の遷移と入力データの遷移が同時におこるため、過小遅延エラー検出信号が生成される。つまり、遅延補償不可能なほど大きな過大遅延は過小遅延エラー検出機構で検出されることになる。

したがって、過小遅延エラーが発生した場合には、原因として 2 つの可能性がある。ひとつはデータ入力が早く到達しすぎて、先行するデータを上書きしてしまった場合。もう一つは、データ入力の変化の開始が遅く到達しすぎて正しい値がフリップフロップに取り込まれなかった場合である。このようなエラーが起こった場合には、正しい値がどこにも保存されていない状態が発生する上に、どちらが原因でエラーが発生したかの特定すらできないという状態に陥ってしまう。このため、このような場合にはプロセッサの処理をリセットして再実行を行うなどの大掛かりな復旧を行う必要がある。この復旧法に関しては、全体システムの中で検討する必要があり、第 7 章で詳述する。

3.2.6　フリップフロップ本体

本節前半の図 2.1 で挙げたような、2 つのラッチを用いた通常のフリップフロップを用いる。このとき、マスター側のラッチにはデータ入力に合わせてタイミングが調整された Ctrl 信号を、スレーブラッチには通常の回路全体の動作の基準となっている clock 信号を供給する。このようにするのには以下のような二つの理由がある。

- フリップフロップ入力から出力への伝搬遅延の最小化
  遅延補償フリップフロップでは、フリップフロップにデータが遅れて到着した際にも、後段のタイミングマージンによって遅れが取り返せることをみこんで、遅延を後段に伝搬させていることを特徴としているのは前述のとおりである。当然ながら、このときに伝搬してしまう遅延はなるべく少ないほうが、後段のタイミングマージンで問題なく補償される可能性が高くなるため、望ましいことはいうまでもない。そこで、ある信号がクロックの立ち上がり（あるいは Setup time）に対して遅れて到着し Ctrl の立ち上がりが遅れた場合を考える。このとき、後段に供給されているクロックは Ctrl 信号よりも一足先に立ち上がるため、入力から出力までが 4 段のインバータで単純につながった状態が生じる (図 3.9)。Setup 条件を満たすために、マスター側のラッチの帰還インバータの出力が安定するまで Ctrl 信号は立ち上がれないが、この間に入力されているデータは、スレーブラッチのインバータを超えることが可能となる。このため、図 3.10 で示すように、遅れてきたデータの入力は、フリップフロップに到着すると同時に、Ctrl 信号の状態とは無関係にフリップフロップ出力に向けて伝搬される。もし Ctrl 信号がマスター・スレーブの両ラッチに供給されているとすると、Ctrl がたち上がった時に初めてデータはマスタースレーブ間のノードに達するため、Ctrl 信号の立ち上がりに要する時間と、インバータ 1 段と、CMOS パスゲート動作の分だけ後段にデータが出発するのが遅くなる。
- Ctrl 信号の動作速度の向上
  マスター・スレーブのラッチにそれぞれ、Ctrl 信号、クロック信号を供給する場合について考えると、フリップフロップ本体が必要とする信号入力数は Ctrl 信号 2、nCtrl 信号 2、Clock 信号 2、nClock 信号 2 個となる（n は各信号の NOT を表すとする）。これに対して、Ctrl 信号ですべてを駆動する場合

には必要なフリップフロップの信号入力数は Ctrl 信号 4、nCtrl 信号 4 となり、Ctrl 信号の駆動部には要求されるドライブ力が大きくなる。Ctrl 信号も Clock 同様なるべく急峻に信号遷移が起こることが望ましいが、マスターラッチのみに Ctrl 信号を供給する場合と同程度の信号特性を望む場合、大きな素子が必要となる。スレーブラッチにクロックを供給する場合、クロックの信号遷移監視部から副産物として nClock 信号は得られるため、こちらのほうが実装規模が少なくて済むことは明白である。

以上の動作速度、実装規模の両観点より、このようにマスターラッチとスレーブラッチで別のクロックを供給するという形をとることとした。

3.3　遅延補償フリップフロップ特性のシミュレーション環境

次節で遅延補償フリップフロップの基本動作波形を示すのに先立って、本研究で用いたシミュレーション環境について簡単に述べる。これ以降、本論文全体を通して遅延補償フリップフロップや回路の特性を示すための多くのシミュレーション結果を示しているが、ここでシミュレーション環境について簡単に述べる。このシミュレーション環境は必要に応じて適宜微調整されているが、本論文の検証結果はすべて以下に説明するようなシミュレーション環境で行ったものである。

3.3.1　シミュレーション対象となる回路の設計

まず、シミュレーション対象となる回路を準備した方法について説明する。これには、厳密性・設計容易性の観点から大まかに 2 種類の方法によって作成を試みた。まず第一は、遅延補償フリップフロップのように、比較的小規模であるが、本論文においてなるべく正確なシミュレーションを行いたい部分である。このような部分に関しては、Cadence 社の集積回路設計 CAD である Integrated Circuit Frontend Backend (ICFG) を用いて、トランジスタレベルで回路素子を並べたものから電子回路シミュレータ Simulation Program with Integrated Circuit Emphasis (SPICE) 形式のネットリストを合成した。また、さらに厳密なシミュレーションが必要と思われる部分には回路レイアウトを同 CAD 上に作製し、そこから Layout Parameter Extraction (LPE) ツール Synopsys 社 Hercules, Star-RCextract などによって、配線の寄生容量などのレベルまで正確なネットリストを作成した。逆に、回路規模が比較的大きく、本研究においてその部分のシミュレーションが重要ではないと思われる部分ではハードウェア記述言語 Verilog HDL を用いて回路を記述し、論理合成ツール Synopsys 社 Design Compiler を用いて Verilog ゲートレベル記述を作成した。これらのレイアウトのデザインルール・論理合成の際のセルライブラリなどは、VDEC を通じローム社から提供されている 0.18-μm 5-metal process の設計パラメータを用いた。

3.3.2　シミュレーション対象の回路への入力信号の準備

また、必要に応じて、一般的な CPU ベンチマークを実際にプロセッサが処理したときと同様の挙動を模倣するために、CPU シミュレータを利用して実行トレースをとり、そのトレースを適宜整形してシミュレーションの入力ベクトルとしている。本研究で提案する遅延補償フリップフロップは、同期式集積回路すべてを適応の対象とするものであるが、特に実際に集積回路中で使用される場合となるべく近い条件で挙動をシミュレーションすべく、同期式集積回路の典型例でもある汎用プロセッサに適用された場合を想定するときにこの実行トレースを利用した。本論文中では、このような実行トレース情報を得るために、東京大学大学院情報理工学系研究科電子情報学専攻坂井・五島研究室で開発されたプロセッサシミュレータである Onikiri II [29] を利用している。プロセッサシミュレータには様々な種類のものがあるが [30] [31]、Onikiri II は標準でサポートされている実行トレースの出力情報が多いこと、cycle accurate であること、などの点から、回路要素の入力データトレースを準備する場合に利用が容易であるという理由で使用した。

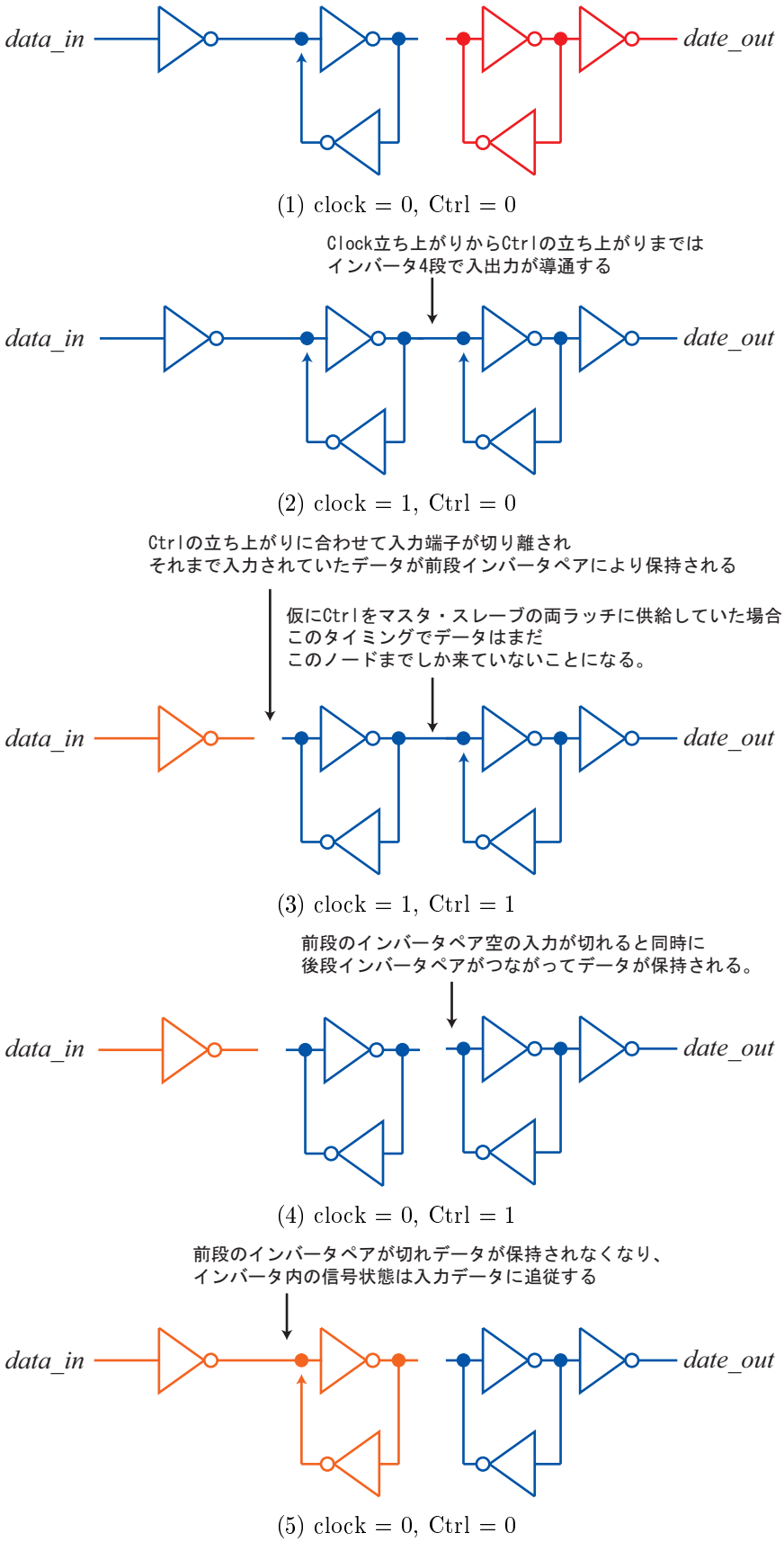
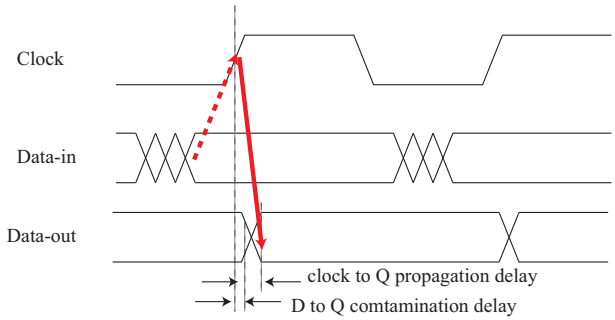
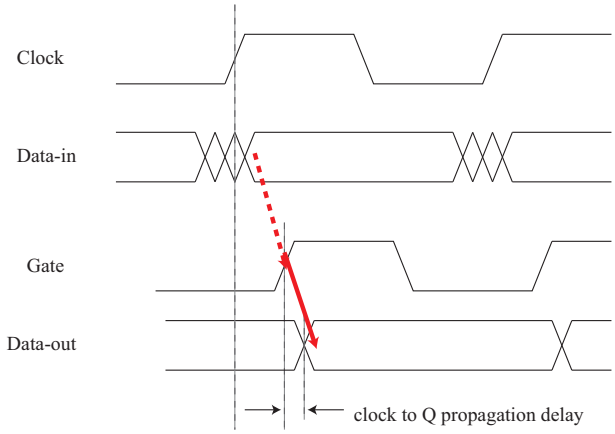


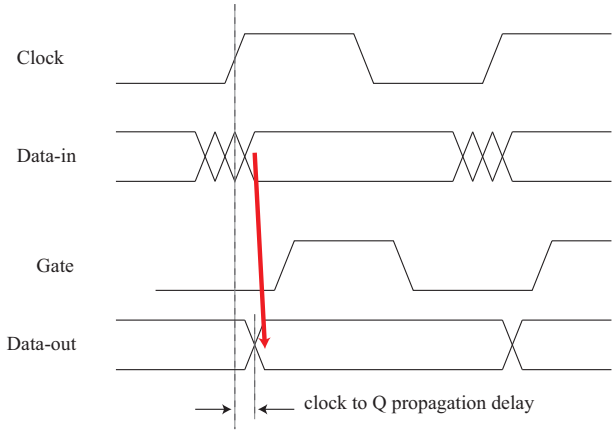
図 3.9: 遅延補償フリップフロップの動作の概念



(a) 通常のフリップフロップを用いた場合
出力データが利用可能になるタイミングはクロックに依存する。



(b) 遅延補償フリップフロップにおいて、フリップフロップ本体に Ctrl 信号のみを供給した場合
出力データが利用可能になるタイミングはデータの遅延に依存して決まる Ctrl 信号のタイミングに依存するため、2 段階に依存関係が存在する



(c) 遅延補償フリップフロップにおいて、フリップフロップ本体に Clock と Ctrl 信号を供給した場合
出力データが利用可能になるタイミングはデータの遅延のみに依存して決まる。

図 3.10: 前段のマスターラッチに Ctrl 信号を、後段のスレーブラッチに Clock 信号を入力することで、Clock 信号や Ctrl 信号の状態によらず、フリップフロップに入力されたデータは到達ごただちに出力に反映される。

表 3.1: 実行トレースを抽出した際のプロセッサシミュレータの主要パラメータ

Technology	Fanout of 4 inverter delay
way 数	2
命令セット	Alpha ISA
機能ユニット	(numbers) iALUs: 2, iDIV: 1, iMUL: 1, iBC: 1, Addr: 1, fBC: 1, fALU: 1, fMul: 1l (Latency) iALU: 1, iMUL: 7, iDIV: 14, iBC: 1, Addr: 1, fALU: 4, fBC: 4, fMul:4 fDiv:15:
命令ウィンドウ	Integer: 32, Float:16, Memory:16, Address: 16 entries
Register File	Integer: 128, Floating Point: 64
L1 Cache	Size: 12KB, Way: 4, Latency: 3 cycles,
L2 Cache	Size: 4MB, Way: 8, Latency: 10 cycles,
Main Memory	Latency : 200

なお、基本的にはシミュレータ　Onikiri II のパラメータは Alpha 21264 (DEC)　[32] に似せて設定しており、シミュレーション対象となる回路も、可能な限りパラメータと整合するような構成としている。なお、実際に消費電力のシミュレーションなどに用いた際のプロセッサシミュレータの主要パラメータを 3.1 にしめす。

3.3.3　回路シミュレータ

本研究では、電源電圧による遅延や、消費電力、タイミングエラーなど、回路レベルでのシミュレーションが適当と思われる部分が多いため、シミュレーション対象の回路規模などの観点から 2 種類の回路シミュレータを用いた。対象とする回路の規模が小さい部分では、Synopsys 社 HSPICE シミュレータを用いてより正確な評価を行うようにした。逆に、対象とする回路の規模が大きい場合やシミュレーション時間が長い場合、ゲートレベル Verilog 記述が対象回路の記述として含まれる場合には、Synopsys 社の Nanosim を用いて、デジタル・アナログ混合シグナルシミュレーションを行った。これらのシミュレーションにおいては、対象回路設計の際に用いたものと同様に、ローム社から提供されている 0.18-μm 5-metal process のデバイスパラメータを用いた。なお、特に 180nm よりも微細プロセスでのシミュレーションを行った部分では、Predictive Technology Model (PTM)　[33] を用いている。したがって、本論文のシミュレーション結果はこれらのデバイスパラメータに基づくものである。

なお、以上の評価環境についてまとめた図3.11 を以下に掲載する。なお、入出力データに互換性のないツール間のデータの変換などは、自作の変換プログラムなどによって行った。

3.4　遅延補償フリップフロップの基本動作のシミュレーション

以下に実際に設計を行った遅延補償フリップフロップの、HSPICE における動作シミュレーション波形をあげる。図 3.12、3.13、3.14 はそれぞれ、通常動作時、過大遅延発生時、過小遅延発生時のシミュレーション結果であり、これは先にあげた概念図 3.2、3.3、3.4 で示した動作が、設計を行った回路で実現できていることを示している。

また、前節のシミュレーション環境の項目で、大規模回路では HSPICE ではなく Nanosim を用いてより高速に処理を行うとしたが、その際のシミュレーション精度の決定においても、HSPICE でアナログ的にシミュレーションを行ったこれらの結果と比較して十分によく近似されていることを確認して行っている。

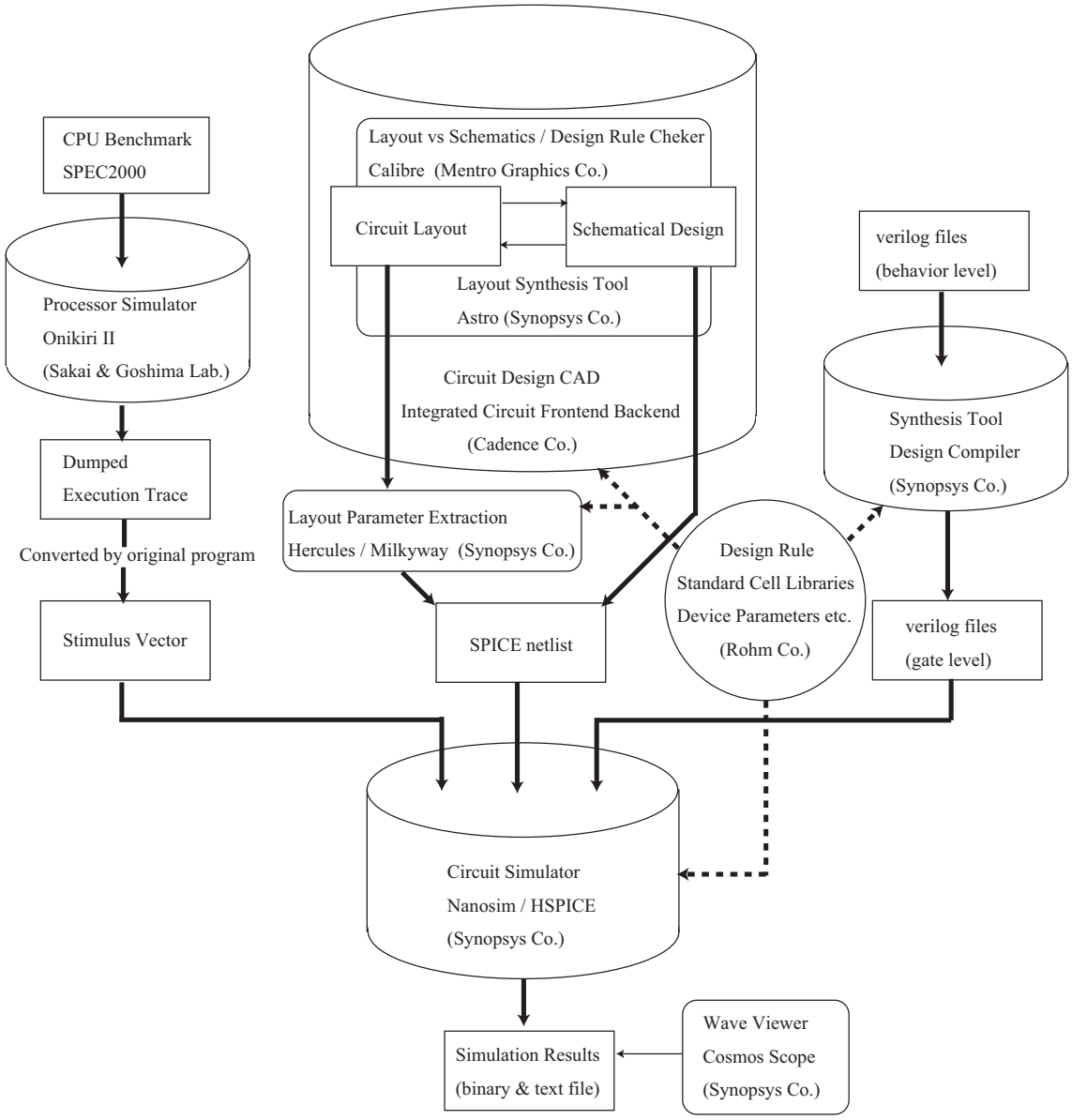


図 3.11: 本論文中で用いられている遅延補償フリップフロップの特性の評価環境

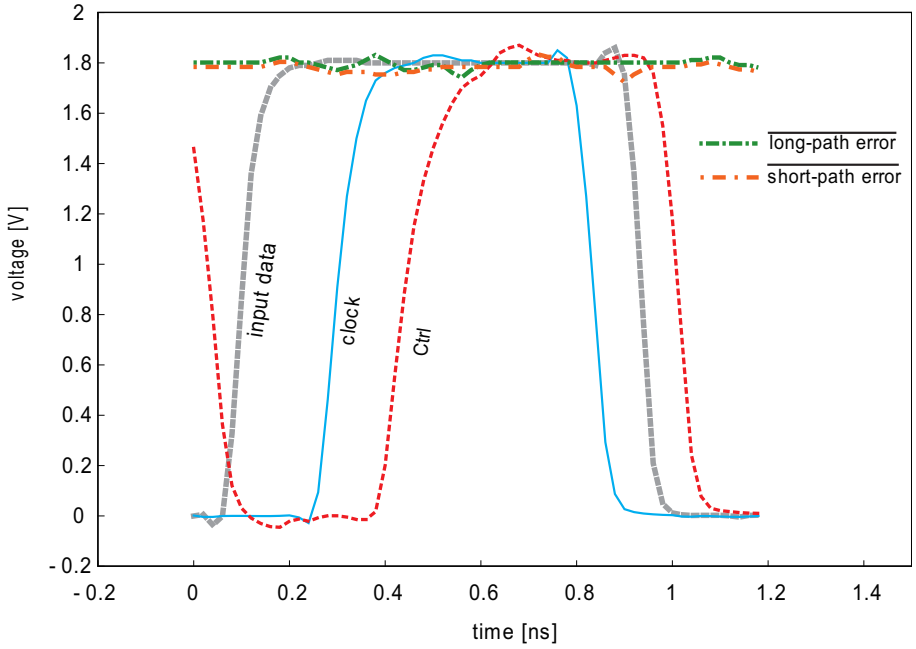


図 3.12: 通常動作時における遅延補償フリップフロップの各信号波形のシミュレーション結果

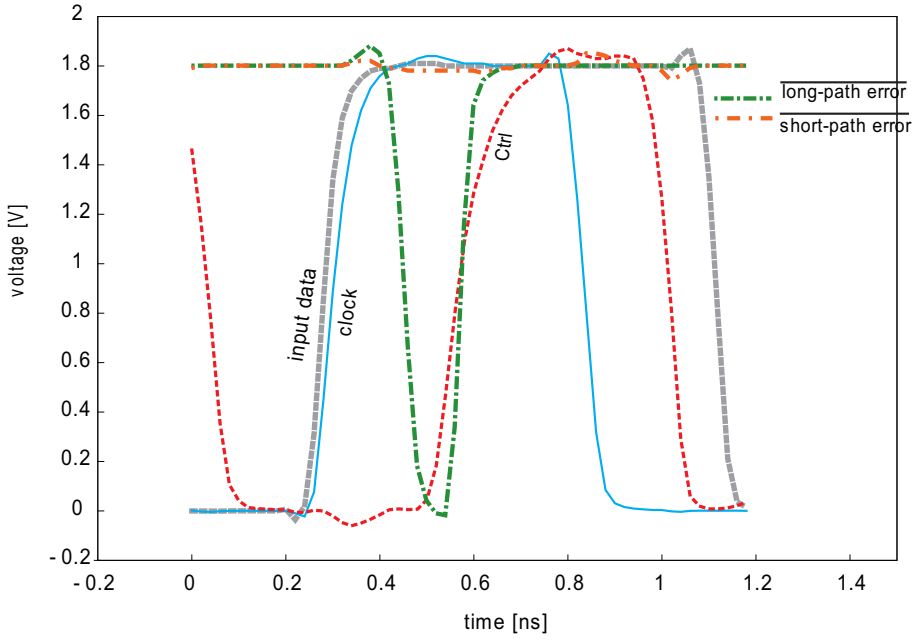


図 3.13: 過大遅延エラー発生時における遅延補償フリップフロップの各信号波形のシミュレーション結果

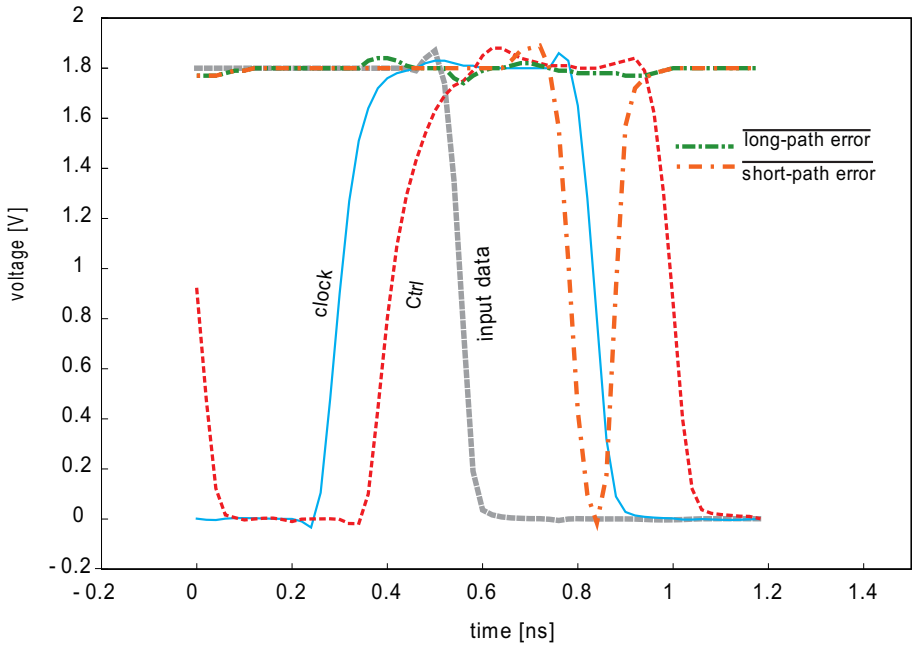


図 3.14: 過小遅延エラー発生時における遅延補償フリップフロップの各信号波形のシミュレーション結果

第II部

提案手法の特性と評価

第4章

タイミングエラー耐性

動作時にタイミングエラー監視するための従来技術では、システムに何らかの冗長化を施し、冗長化部のタイミングマージンを互いにずらすことで、部分的にエラーが起こりにくい場所（手法によっては起こりやすい場所）を作り出し、それらの処理の結果が等しいことを確認するという方法がとられているものが多かった。本研究で提案する遅延補償フリップフロップでは、何らかの冗長化を施すということはせずに、信号の遷移タイミングを直接監視する方法をとる。冗長化要素を用いる手法では、高信頼化の対象としていない部分のタイミング制約に関しては、複数あるユニットのうちで耐性の低い方の影響を大きく受け、Razor[8] のショートパス問題などのように、全体として正常動作するタイミングが狭くなってしまうということが起こりうる。これに対して、冗長化要素を持たない遅延補償フリップフロップではこのような理由による設計制約の狭小化は起こりがたい。また、従来手法ではエラーが発生した場合、何らかの回復手法がとられることになるのが通常であり、このときに処理のオーバーヘッドが生じる。遅延補償フリップフロップでは、一般的なデジタル回路では実際にタイミングクリティカルなパスが活性化する可能性は低い [23] という性質を利用して、なるべく処理オーバーヘッドなく処理を続行するようにしている。以下に、既存研究と比較しながら、遅延補償フリップフロップがもつタイミング特性について述べる。

4.1　タイミングエラー発生要因と回路の振舞い

遅延補償フリップフロップの持つタイミングエラー耐性の特徴について説明する前に、まずタイミングエラーがなぜ発生するかについて簡単に考察を行う。フリップフロップなどを利用した同期式回路は、2 つのフリップフロップに挟まれた回路中のすべての組み合わせ論理回路のうちで、最も処理に時間がかかるものよりも、クロックサイクルは十分に長くとられている。これは、回路を構成するありとあらゆる組み合わせ回路は、出力側に存在するフリップフロップのデータの取り込みに間に合うように、その処理が終了するように設計しなければならないためである。しかし、実際に作製されるチップは必ずしも設計通りの性能を動作時に発揮するとはいえない。この原因としては、上述してた回路素子の性能ばらつきなどの製造済みチップの特性によるものだけではなく、動作環境によるものもあげられる。たとえば、回路を構成している金属配線は回路の発熱や、外気の温度などの要因によって高温になると、抵抗値が大きくなる。このことは配線遅延の主な要因である RC 遅延を悪化させるため、全体として回路の動作が遅くなる。また、意図的かどうかにかかわらず、電源電圧は、過渡的・定常的に揺らぎを起こす。電源電圧が低下すると、トランジスタの動作速度が低下するので、これも全体として回路の遅延を増大させる結果となる。非常に代表的なものだけを以上にあげたが、回路の処理速度の変動を引き起こす要因は非常に多く、設計段階でタイミングマージンを導入する場合に見込まれている影響よりもよりも、実際の外乱の影響が大きいとタイミングエラーを引き起こすことになる。このため、近年はこの設計段階で挿入されるマージンの割合が肥大化してきていることが問題となっていることも前述のとおりである。

同じステージのフリップフロップに供給されるクロックごとにプログラマブルな遅延を挿入しておき、製造後にチップごとに最適なクロック配分を行う技術の研究はすでに多くされている [24] [25]。これは、製造後個体ごとの特性のばらつきを考慮して最適化しているものであるといえる。これに対して遅延補償フリップフロップなどの動作時エラー監視では実際に動作時にエラーを観測しながら、必要に応じて DVFS などと協調して回路を正常動作させることを目的としているため、電源電圧や温度の変化など動作時における外乱に対する調整技術であるとすることができる。したがって、これらの技術は相反するものではなく、協調して用いられることが望ましいと考えられる。つまり、製造後クロックタイミング調整によって粗調整を行い、動作時エラー監視によって微調整を行うようなイメージで、これらの技術を共存させることで、より効果的に堅牢なシステムの構築に貢献できることが考えられる。

上で述べたように電源電圧などの変化では、個々の素子・配線の遅延が全体として回路動作の遅れを引き起こしているわけであるが、それはつまり、処理のパス上に存在する素子数が多いほどその影響が大き

くなるということは予想に難くない。図 4.1 は、さまざまな電源電圧下で 32bit Kogge-Stone Adder‘を動作させた場合の最長パス遅延と最短パス遅延の変化を示している。このように同じような外乱に遭遇しても回路はその機能や入力によって影響を受ける大きさが異なってくることがわかる。

遅延補償フリップフロップでは、このようなパスの長さによる遅延の変化を積極的に利用することを特徴としている。

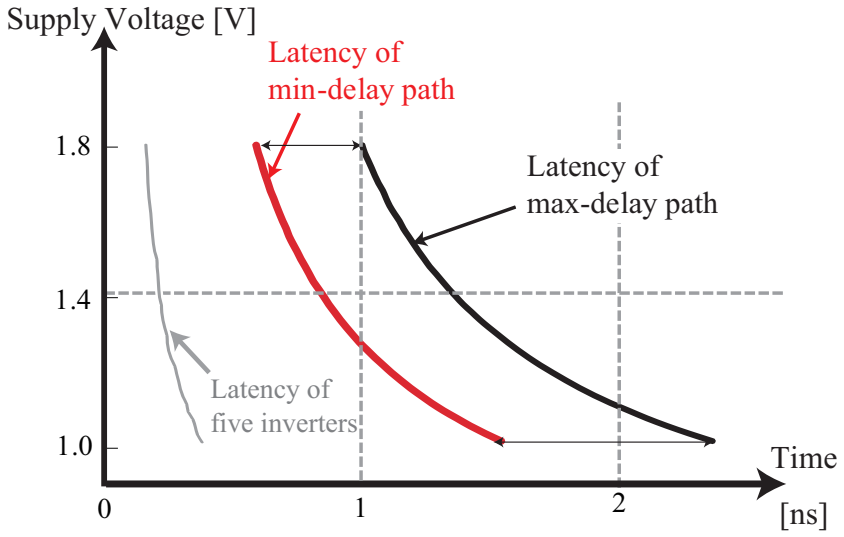


図 4.1: 電源電圧と最大遅延パス・最小遅延パスの処理時間の関係

4.2　従来技術との比較

本節では、提案している遅延補償 Flip Flop の特性を明らかにするために通常の Flip Flop、カナリア Flip Flop、Razor Flip Flop と比較しながら説明を行う。以下で詳述するが、図 4.2 に示すように、遅延補償フリップフロップは、Min-delay constraint、Max-delay constraint によって制限されるタイミング設計の自由度が他の手法に比べてに広いことが特徴である。まずこのタイミング制約に対する設計の自由度に関する性質について述べる。

4.2.1　Max-delay constraint

Max-delay constraint はショートパス問題の紹介の部分で説明した Min-delay constraint と対になる制約であり、これは前段のロジックブロックの最長パスのタイミング設計に関する規則である。通常のフリップフロップでは最大遅延が setup time に間に合うように設計する必要があることは、これまでも述べているとおりであるが、カナリアフリップフロップでは setup time より挿入する遅延の分だけ早くデータが到着するように設計する必要がある。そうしなければ、つねにカナリア側のフリップフロップにはデータが間に合わないため、エラーの予測が生じ続けることになってしまい、通常の処理ができない。Razor フリップフロップの場合には原理的には遅延クロックの setup time に間に合えばよいが、通常のクロックに遅れた際はエラーの回復に伴うオーバーヘッドが必要となる。これに対して、遅延補償フリップフロップでは、基本的には通常のフリップフロップにデータが間に合うことを前提とするが、なんらかの影響で処理が遅れた場合でも、非常に高い確率で復旧オーバーヘッドを生じずに対処することができる。このとき、どの程度の遅延までを許すかということは設計上の問題となるが、設定された限界の範囲内でフリップフロップの Setup time が動的に変化するため、max-delay constraint はあたかも回路の処理速度に合わせて動的

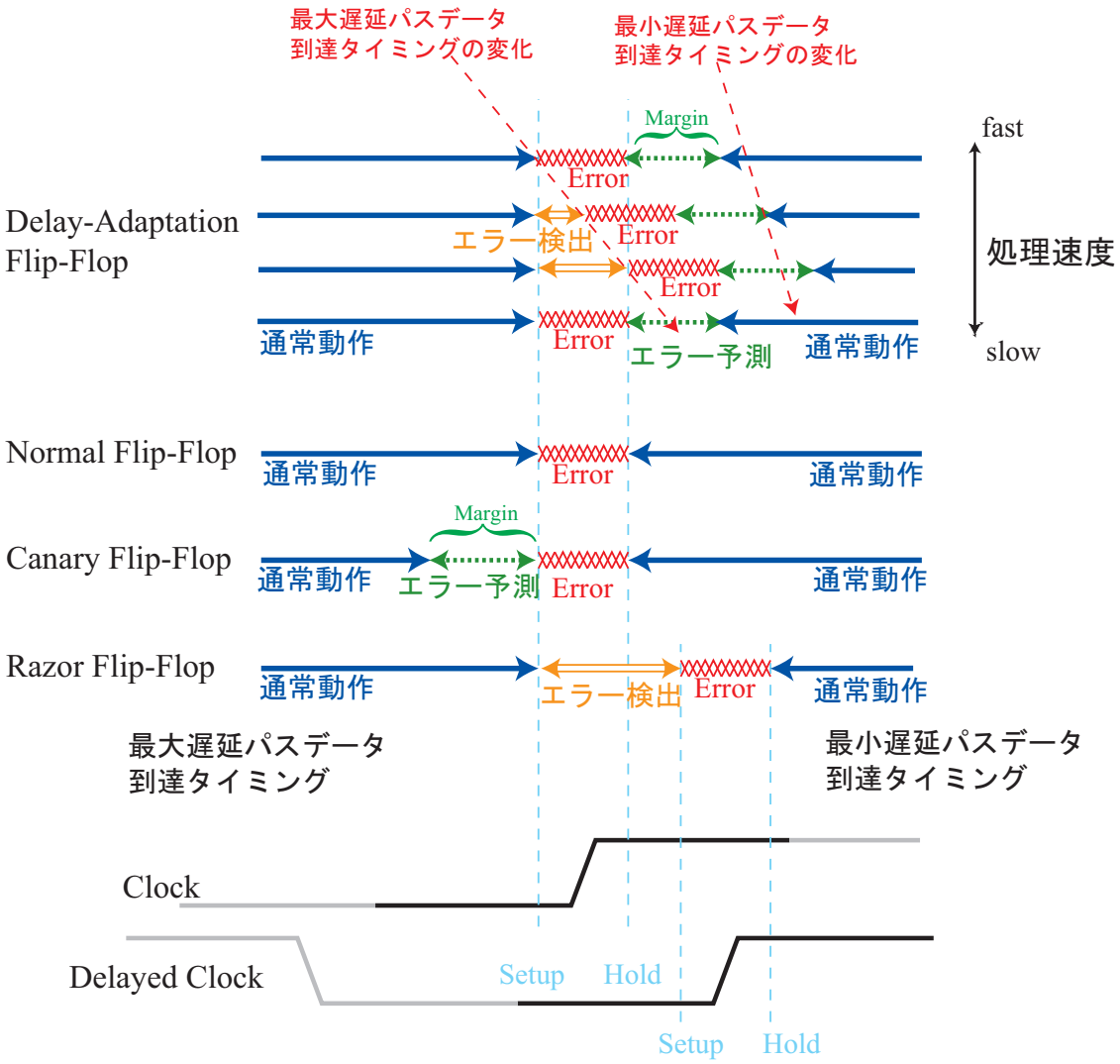


図 4.2: 各フリップフロップのタイミング制約の比較。データの到着が遅れるに従って、エラーとなる範囲が逃げていくような挙動をするのが遅延補償フリップフロップの特徴である。このような動的な変化をする特性から、設計段階でのタイミング制約の自由度は他のフリップフロップよりも高く設定できる。

に変化しているかのようにふるまうことになる。

4.2.2　Min-delay constraint

通常のフリップフロップと、カナリアフリップフロップでは最小遅延パスを通るデータは、通常のクロックの hold time よりも遅れて到着する必要がある。また、前述のとおり Razor フリップフロップでは最小遅延は遅延クロック側の hold time よりも遅れて到着する必要がある。提案手法では、遅延してきたデータによって遅れた Ctrl 信号の規定する setup time タイミングよりも後続の信号の到着が遅れている必要がある。提案手法では、このコントロール信号の立ち上がりタイミングが動的に変化することは何度も述べたとおりであるが、このように何らかの理由で最長パス遅延が大きくなっている場合には、図 4.1 で示した通り、最短パスの動作速度も当然ながら一定の割合で遅れていることが考えられる。このため、図 4.3 に示すように供給電圧などの状態によって適応的に Min-delay constraint が動いていくため、設計時において最短パスのタイミングは動作時の最短パスよりもゆるい制約下設計できることになる。これによって、無駄な遅延素子の挿入を削減できる可能性がある。

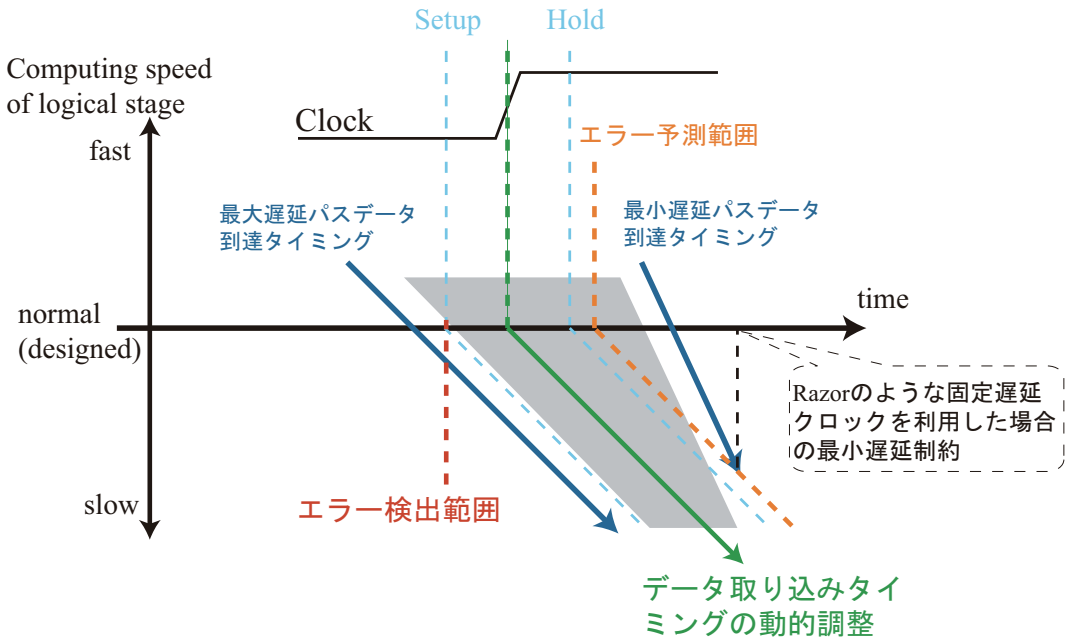


図 4.3: フリップフロップが正しく値を取り込める範囲と、回路の動作速度の関係

4.2.3　エラー検出時の復旧

カナリアフリップフロップではエラーではなくエラーの予報を検出して、実際にエラーが発生する前に周波数や電源電圧などをコントロールする。このため、複雑エラー回復機構を必要としないことを特徴としているが、逆にエラー予報を検出したらすぐに対応しなければ、実際にエラーが発生してしまった場合に復旧できなくなる可能性があるからである。また、最適な電源電圧を達成するために、供給電圧は常に低くする方向に動かしておく必要があるため、供給電圧は経過時間に対して振動することになる。Razor ではエラーを検出した場合にはパイプライン処理をとめて Shadow latch から main flip-flop へ正しい値を書き戻す必要がある。このとき、古典的な手法では 1 クロックのパイプラインストールが発生する。また、スケーラブルなエラー回復手法を用いる場合にはこのときのオーバーヘッドはさらに大きくなる。提案手法では正規のタイミングに遅れたものも、多くは後段で遅れが回復するものと仮定しているためオーバー

ヘッドの発生確率は低い。また、正規のタイミングに遅れたものの頻度をモニタリングしておくことでペナルティ無く DVFS などに必要な現在の回路動作の厳しさを知ることができる（Razor では実際にエラーが起こった場合の頻度を利用しているため、タイミングエラーの発生状況の情報を得るためにオーバーヘッドが必要となる。）。

提案手法、既存手法とも、DVFS などによる電圧・周波数の動的な最適かと協調することを前提としているが、電圧・周波数を変化させる場合にはそれに伴う比較的大きなオーバーヘッドが生じることは、どの手法においても同様である。このような DVFS にともなうオーバーヘッドをなるべく少なくするためには、できる限り電源電圧が振動するといった状態を避けることが望ましいと考えられるが、過大遅延エラー検出と過小遅延エラー予測を分けて検出する遅延補償フリップフロップでは、回路動作タイミングの厳しさをより定量的に把握しやすいため、電源電圧・周波数を最適な位置で固定しやすいと考えられる。Razor でもエラーの発生頻度などからこれらの情報を得ているが、このような性質は DVFS を効果的に行う上での利点となること考えられる。

4.3　シミュレーションによるタイミングエラー耐性の評価

4.3.1　シミュレーション結果

実際に 36bit Kogge-Stone Adder を設計し、その動作範囲を調べた。入力としては、最長パスが活性化した直後に、最短パスが活性化するような入力ベクトルを用いた。これは、図 4.3 からわかるように、“最長パスが活性化しフリップフロップコントロール信号が遅れて立ち上がった後、最短パスが活性化しすぐに信号が到着する状況” が、遅延補償フリップフロップに対してタイミング的に最も厳しい状況であることによる。また、遅延補償フリップフロップがどの程度の遅延まで対応するかは設計上の問題となるが、今回のシミュレーションでは、遅延補償フリップフロップを構成する論理回路が動く範囲で最も短い幅のパルスを信号遷移検出回路が発生するように設計したものでシミュレーションを行っている。

実際にシミュレーションを行った結果、種々の電源電圧で遅延補償フリップフロップがどのような挙動を示すかについてまとめた図を、図 4.4 に示す。

この遅延補償フリップフロップと、カナリアフリップフロップを用いて、SPEC2000 ベンチマークを動かした際に整数系のＡＬＵにおいてエラーを起こさない（遅延補償フリップフロップでは遅延補償ができなくなる限界、カナリアフリップフロップではエラーを予測しない限界）最も電源電圧を低い電圧を、図 4.5 にしめす。遅延補償フリップフロップのほうが、カナリアフリップフロップに比べて、低い電圧までタイミングエラーを起こさずに動作を行えることが見れる。

なお、このときのベンチマークは１Ｇサイクルスキップしたのち、10 万サイクルにわたってシミュレーションしたものである。また、ベンチマークに対する入力データは train を使用している。シミュレーションを行った電圧 (1.8V 以下　1.0 V 以上) の範囲内で、カナリアフリップフロップ、遅延補償フリップフロップともに一つのエラーも検出しなかったベンチマークに関しては、図 4.5 に掲載していない。

4.3.2　シミュレーション結果に関する考察

シミュレーション結果によると、電源電圧が下がり、入力データの到達タイミングが遅れた場合に、フリップフロップコントロール信号よりも、実際にデータ出力が利用可能になるタイミングのほうが早いことがわかる。これは、遅延補償フリップフロップの基本動作の項目で説明したとおり、クロックが立ち上がった時点で、入力されたデータは後段に出力されることによる。フリップフロップコントロール信号も数段のゲートによって構成されているため、その動作速度は電源電圧の低下の影響を受けることになる。とくに、低電源電圧化では信号変化の波形がなまるため、通常高速動作することが特徴であるドミノ論理回

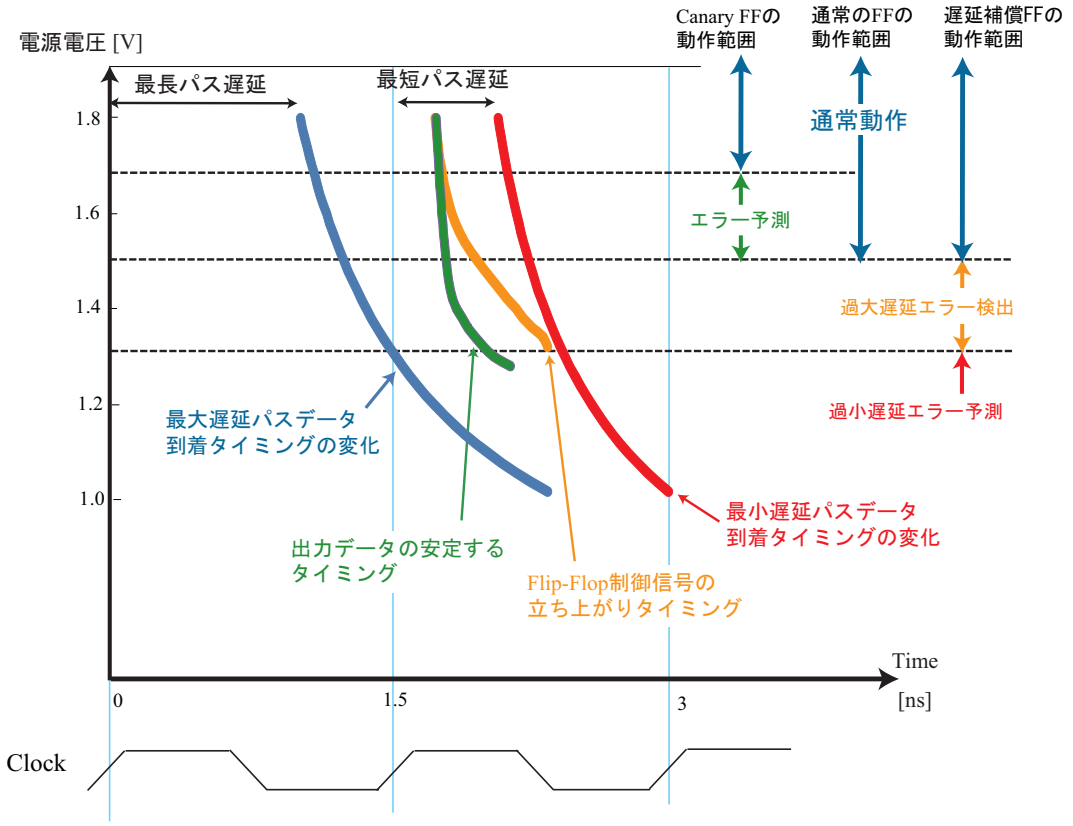


図 4.4: 32bit Kogge-Stone adder の後段に遅延補償フリップフロップを挿入した場合の各信号の動作タイミング。図の線はそれぞれ、0ns に入力されたデータが最長パスを通ってフリップフロップに到達したタイミング、到達したデータが取り込まれフリップフロップの出力が正しい値に安定したタイミング、次のクロックの立ち上がり (1.5ns) に入力されたデータが、最小遅延パスを通ってフリップフロップに達したタイミングである。遅延補償フリップフロップでは、このように、最大遅延パスの活性化に引き続いて、最小遅延パスが活性化した場合が、最悪ケースとなる。

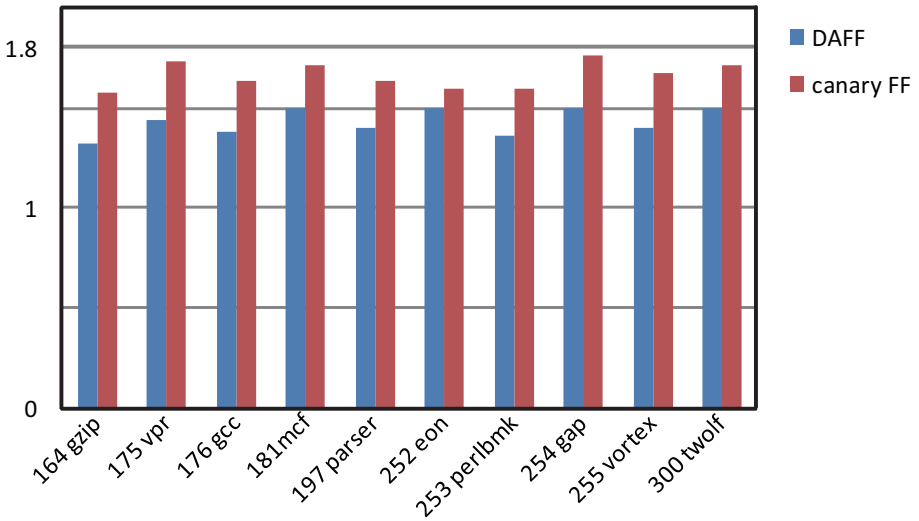


図 4.5: 各ベンチマークの整数系ＡＬＵでの限界動作電源電圧の比較

路でも評価フェーズにおける放電時間が大きくなる傾向がある。また、なるべく早い段階でタイミングエラーを検出したい場合は、データ遷移を示すパルス幅を広くすればよいが、このときも入力データが Ctrl 信号の立ち上がりを待っていると出力に反映されるのがおそくなるため、このことからもスレーブのラッチにはクロックを供給することがよいことがいえる。

また、1.34V において、入力の遅延に対して、遅延補償フリップフロップが追随できなくなっている。今回のシミュレーションで用いた入力では、信号の変化は図中の青線で示したタイミングで起こる一回のみである。この信号変化の開始のタイミングがクロックの立ち上がりよりもある閾値以上おそくなると、データの変化の前にフリップフロップコントロール信号が立ち上がってしまうため、入力がフリップフロップ前段のラッチに反映されない。この場合、フリップフロップコントロール信号の立ち上がりとほぼと同時、あるいは直後に入力データ信号が変化することになるため、最短パス遅延エラーが生じたのと同様の状態となり、当該エラー信号が出力されることになる。第 7 章で述べるようなシステマチックなエラー復旧機構を用いない場合は、このような状況に陥らないように、過大遅延エラーが検出され始めた段階で、電源電圧を昇圧するなどの対策をとることが必要となる。

また、図中にはクロックサイクルの 10%の遅延を挿入した場合のカナリアフリップフロップの動作範囲を示している。カナリアフリップフロップがエラーを予測し始める電圧と、遅延補償フリップフロップが、遅延を後段に受け流すことができなくなる理論的な限界電圧の間には、およそ 0.4V という大きな開きがある。

4.4　関連研究

このほかにも動作時のエラー検出の関連研究の種類は多い。Razor などでは Flip-Flop を冗長化していたが、前段の論理回路を冗長化し、その結果を比較するといった方法も存在する。しかしながら、回路全体を単純に冗長化するのは非常にハードウェア量や消費電力のオーバーヘッドが大きくなるので、approximation cirucuits[11] [12]、algorithmic noise tolerance 方式 [13]、TEAtime 方式 [14] などの工夫された手法が提案されている。る。これらの方法では、基本的に演算を行う論理回路の部分を一部あるいは全部冗長化し、その結果を比較するという方法がとられている、空間的な冗長化を施したものであるといえる。

DIVA [36] [37] では、図 4.6 に示すように通常の動作をするメインのプロセッサのほかに、チェッカープロセッサを用意する。このチェッカープロセッサは、投機実行の結果はメインのプロセッサが行った処理を利用できるなど、必要な機能が少なくて済むため、エラーが起きにくいような設計とすることが比較的容易であり、メインプロセッサの実行結果が誤っていないかをチェックする働きを持つ。なお、メインのプロセッサから大きく遅れ足りすることのないように、全体としての処理スループットはメインのフリップフロップと同程度になるように設計されている。

AR-SMT [38] では Simultaneous Malti-Threading (SMT) によって、実現される手法である。まず、A-Stream というスレッドが通常通り処理を行い、処理結果をいったんバッファに蓄えておく。次に R-Stream と呼ばれるスレッドで同じ処理を再び行い、保持されている A-Stream の結果と比較してその結果が正しいかどうかを確認する。

このように、多くのエラー検出手法では、ハードウェアを複数用いて空間的に冗長化するにしろ、時間をずらして再実行することで時間的に冗長化するにしろ、同じ処理を 2 回以上行う、あるいは、処理結果を 2 回以上サンプリングするなど、冗長化を基本概念とするものが主流である。この点で、遅延補償フリップフロップの冗長化を全く用いない方法というのは独特であることがわかる。

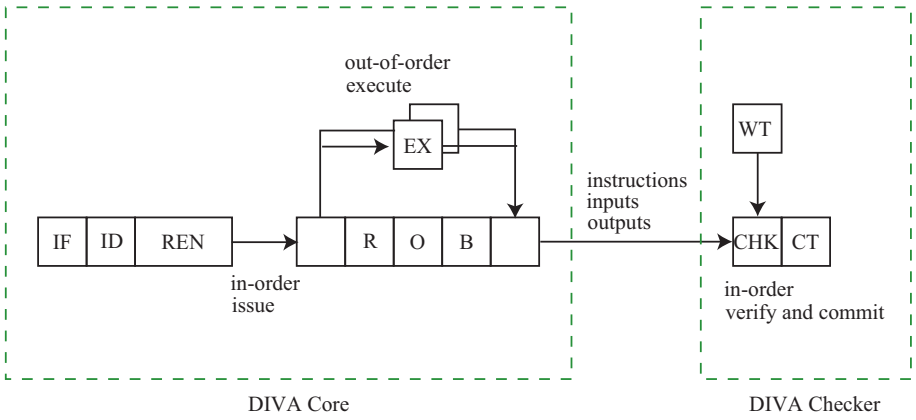


図 4.6: DIVA の構成 [36]。なお、図中の各略語は IF: instraction fetch, ID: instruction Decode, REN: rename, ROB: reorder buffer, EX: execute, WT: watchdog timer, CHK check, CT: commit を示している。

第5章

省電力応用

本章では、遅延補償フリップフロップを DVFS に応用した場合の省電力特性の評価をおこなう。また、追加回路の消費電力についての検討を行うなど、消費電力の側面から遅延補償フリップフロップの特性について考察する。

5.1 Dynamic Voltage and Frequency Scaling と動作時エラー監視技術

前章まではおもに、電源電圧を調整したり、動作周波数を調整したりすることによって、回復不可能なタイミングエラーが起こる前に対処し、最適な動作条件を用いるために動作時エラー監視とともにＤＶＦＳを検討してきた。DVFS は一般的には、処理能力に余裕があるときは電源電圧や動作周波数を低くすることによって消費電力を節約したり、逆に、一度に大量の処理を行う必要がある場合などには電源電圧と動作周波数を高めることによって処理能力を高めるといったように用いられている技術である。本章ではＤＶＦＳと遅延補償フリップフロップを用いた省電力効果について検討を加える。

5.2 追加された回路要素による電力消費

遅延補償フリップフロップ、カナリアフリップフロップ、Razor ともに、単純なフリップフロップになんらかの追加回路を付加することで実装されている。このため、追加回路によって消費される電力のオーバーヘッドが生じる。実際にはこの消費電力オーバーヘッドのほうが、動作時エラー監視を併用した DVFS による省電力効果よりも十分に小さくなければ、全体としての消費電力の削減効果は期待できない。

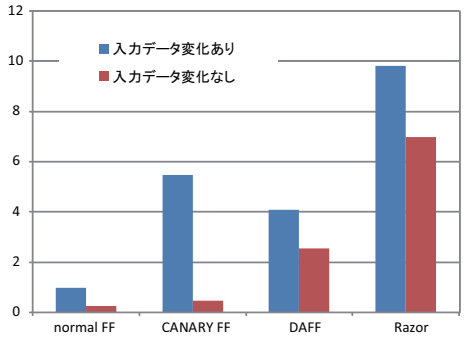


図 5.1: 各フリップフロップの 1 サイクル分の消費電力比較。ただし、クロックサイクルないでデータが変動し、保持するデータが変わった場合に通常のフリップフロップで消費された電力を 1 として正規化している。

図 5.1 に 1 クロックサイクルの間に各フリップフロップが消費する電力を、比較したものである。なお、シミュレーション対象とした回路の構成は、カナリアフリップフロップでは、2 個のフリップフロップと、ＸＯＲゲート 3ns の遅延素子までを、遅延補償フリップフロップでは、図 3.5 に含まれている部分までを、Ｒａｚｏｒフリップフロップでは 2 個のフリップフロップと、ＸＯＲゲート 3ns の遅延素子までを対象としている。このため、最終的なエラーの検出・予測などの信号を生成するためのビット間論理和などの部分は評価の対象に入っていない。なお、遅延素子はインバーターチェーンによって 3ns の遅延を実現している。この条件では、遅延素子が消費する消費電力が比較的大きくなっていることは注意すべき点である。

入力データが変化し、保持している状態が変化する場合に、それぞれのフリップフロップが 1 サイクルで消費する電力をシミュレーションにより求めたところ、遅延補償フリップフロップは 400%、カナリアフリップフロップは 510%、 Razor は 98% であった。このように、Ｒａｚｏｒが大きな値をとっていることには理由があり、今回のシミュレーションでは、カナリアフリップフロップで利用していた遅延素子を用いていることによる。今回のシミュレーションでは入力データの変化回数は 1 回としたが、クロックは 1 サイクルのうちに 2 回の変動が起こる。このため、Ｒａｚｏｒは入力データの変化がある、なしにかかわらず、消費電力が大きくなってしまっている。実際には、インバータを用いて反転クロックを用意したり、クロックは 1 サイクルで 2 度しか反転しないことが保証されているため、1 つあたりの素子の遅延を大きくした、より省電力な遅延素子を用いることが可能である。また、カナリアフリップフロップでは、データの入力に変化が起こらないときは、単純に 2 つのフリップフロップのクロックの十放電がおこる程度の挙動しか示さない。このため、入力データに変化が起こらないときのカナリアフリップフロップは非常に消費電力を小さく抑えることが可能である。これに対し、データ入力の変化の有無にかかわらず、遅延補償フリップフロップでは、クロックの信号遷移監視回路がクロックの立ち上がりごとに動作するため、データ入力の変化がない場合でもカナリアフリップフロップの場合のように大きく消費電力を抑えることができないが、データの信号遷移監視回路は動作しないため、当然ながら入力データが変化する場合に比べれば消費電力は少なくて済む。また、クロックが変化したときに消費される電力がＲａｚｏｒよりも少なくなっているのは、遅延補償フリップフロップのほうは、クロックの立ち上がりのみで追加回路が動作しているのに比べて、Ｒａｚｏｒではクロックの立ち上がり、立ち下がりの両方で動作が行われていることが原因として考えられる。

以上、どのような場合でも、追加回路が存在する分消費電力のオーバーヘッドがあることは避けられない。このため、省電力用途に利用することを考えると、これらのフリップフロップを適用する前段の組み合わせ論理回路は十分大きくなければ、これらのオーバーヘッドを隠蔽することは難しい。

5.3　クロックゲーティングとの協調

実際の回路中では、フリップフロップのデータがすべて毎サイクル更新されるということはなく、ある値を数サイクル保持し続けることのほうがむしろ多い。さらに、近年のハイエンドなプロセッサでは、整数系か浮動小数点系か、論理算術演算系かメモリアクセス系かなどによって、機能ユニットが細かく分かれていることが多いため、各機能ユニットに付随しているフリップフロップには必ずしも毎サイクルフリップフロップに新しいデータが入力されてくるわけではない [32]。また、データを構成するビットのうち、すべてのビットが使われているということは少ない [39]。以上のことから、フリップフロップに入力されるデータ信号とクロック信号では、圧倒的にクロックの遷移のほうが回数が多いことが予想される。しかしながら、遅延補償フリップフロップの追加回路は、クロックの入力から、クロック信号遷移検出回路・フリップフロップコントロール信号生成回路・フリップフロップコントロール信号遷移検出回路とつながっており、これらはデータ取り込みの有無にかかわらず毎サイクル動作することになる。これは、カナリアフリップフロップのように追加回路が主にデータ入力側とつながっている手法では起こらない現象であり、データ入力の変化が少ない場合は、提案手法のほうが既存手法に比べて多くの電力を消費することになるため、図 5.1 にも示すように遅延補償フリップフロップにとって不利な条件となる。

実際にＳＰＥＣ 2000 に含まれる整数系のベンチマークを実行した場合の電力を比較した図を図 5.2 に示す。この図は通常のフリップフロップを 1.8 Ｖ　の電源電圧で用いた場合の消費電力で正規化を行っている。

この図からもわかるように、単体で遅延補償フリップフロップを用いた場合には、電源電圧を低く動作させてもなお、通常の電圧で動作するフリップフロップよりもかなり消費電力が多い。このため、必要の

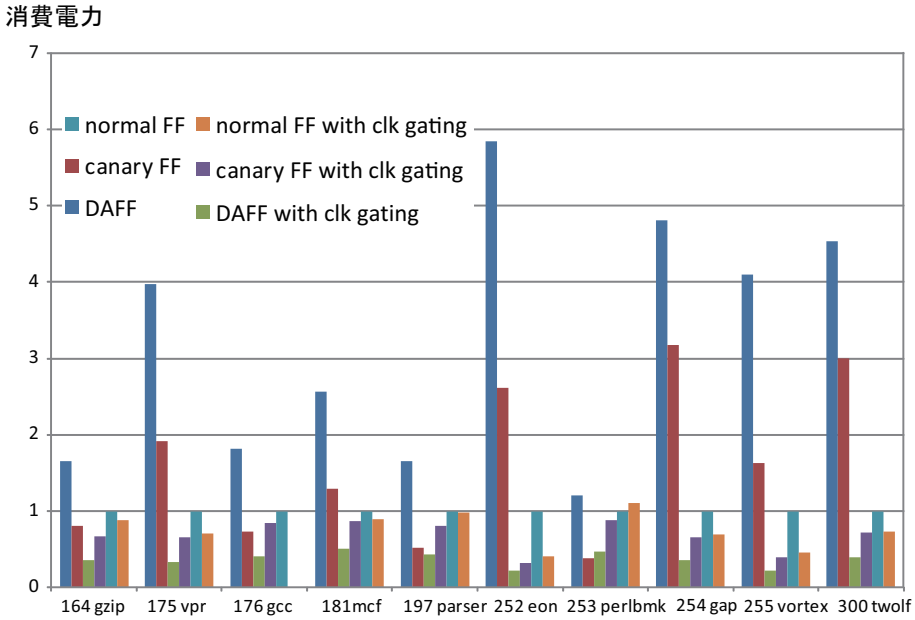


図 5.2: 各ベンチマークの整数系ＡＬＵでの消費電力の比較

ないときにはフリップフロップに供給されるクロックを遮断するクロックゲーティングなどによる省電力化とともに用いられることが望ましい。クロックゲーティングを用いた場合、シミュレーションを行ったすべてのベンチマークに対して省電力化に成功している。なお、本シミュレーションにおいては、図 4.5 に挙げた限界電圧で常に動作する場合の非常に理想化された条件でのシミュレーションである。また、今回用いたクロックゲーティングの手法としては、命令・データが ALU に入力されたときにのみクロックイネーブル信号を生成し、このクロックイネーブル信号と、クロック信号の論理積を実際にフリップフロップに入力される信号としてもちいたが、このイネーブル信号を生成するために必要な回路によるオーバーヘッドは消費電力として計算していない。単純に命令が有効かどうかだけを判断すればよいため、演算気においてこのようなクロックゲーティングを行うことは非常に容易であるとされている。なお、図 5.3 に ALU の使用率と、クロックゲーティングを用いなかった場合の遅延補償フリップフロップの消費電力の相関を示す。このように、ALU の使用率と、遅延補償フリップフロップの通常のフリップフロップに対する電力オーバーヘッドの間には強い相関が成立していることがわかるが、この結果も本節の考察を裏付けるものである考えられる。

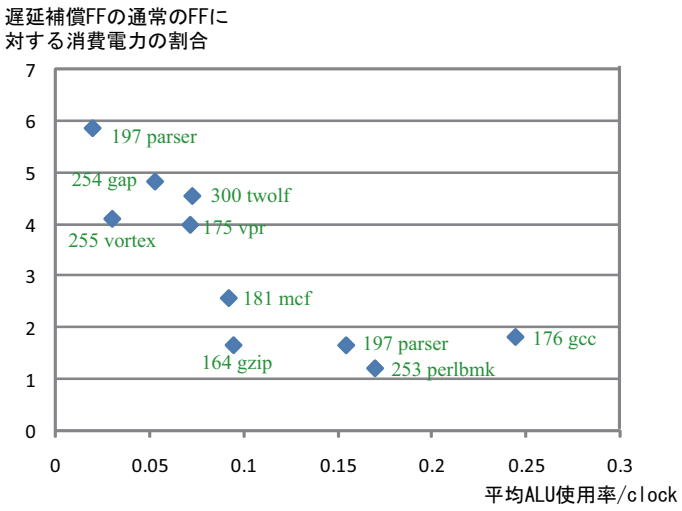


図 5.3: 各ベンチマークの整数系ＡＬＵでの消費電力の比較

5.4　省電力に関する関連研究

5.4.1　ダイナミック電力削減技術

クロックゲーティングの既存技術や DVFS に関しては、すでに簡単に取り上げているので、そのほかの関連低電力化技術と、その遅延補償フリップフロップの協調性などについて、以下に簡単に考察をくわえる。

- 並列化と多電源電圧割り当て
  回路要素を並列化し、高速で消費電力が大きいものと、低速だが低消費電なものを用意する。処理がクリティカルでない処理は低電力でゆっくり行っても、その定義上からもいえることであるが、全体としての処理性能に及ぼす悪影響は小さいため、すべての処理を高速で消費電力が大きいカイロで実行するよりも、性能の低下を極力抑えつつ、低消費電力化を達成することが可能となる。このようなシステムでは、低速処理部は十分に大きな設計マージンを取ることが容易であるため、遅延補償フリップフロップは、高速実行部にのみ適用するばよいと考えられる。
- コード圧縮頻繁に実行されるコードを圧縮することにより、回路の動作部分を削減することが可能となる。回路の動作部分が少なくなるということは、入力値が変化するフリップフロップも減少することが考えられる。このようなばあい、どのフリップフロップが動作するべきかをあらかじめ予測してクロックを供給するフリップフロップを決定する方法では追加回路が必要になるため、遅延補償フリップフロップの自己クロックゲーティングを利用しやすいことが予想できる。

5.4.2　スタティック電力削減技術

スタティック電力を削減する技術に関しては、おもに式位置電圧の変更によるもの、電源電圧の変更によるものの 2 つの種類に分けられる。閾値電圧の変更によるものの代表的なものには以下の 3 つがある。

- Dual Vth
  閾値電圧が低いトランジスタは高速にスイッチング動作をするがリーク電流が大きくなり、逆に閾値電圧が高いトランジスタはリーク電流は小さいがスイッチング動作は遅くなる。このような性質を利用して、高速高リーク電流の低閾値トランジスタと低速低リーク電流の高閾値トランジスタを製造段階で 2 種類作成し、回路中で必要に応じ使い分ける。これによって、高速処理が必要な部分に高速トランジスタを用いることで処理性能の低下を抑えながら、低速動作でよい部分の諸費電力を抑えることが可能であるため、効率的にリーク電流の削減を行うことが可能となる。しかしながら、2 種類のトランジスタを作成することにより、製造工程が増加することから、製造期間の長期化、製造コストの上昇、歩留まりの低下などが問題となる。
- VTCMOS（Variable Threshold-voltage CMOS）
  CMOS のバックゲートバイアスを調整することで、閾値を変化させる [21]。閾値を高くするほどリーク電力は削減できるが、その分動作速度は低下する。また、バックゲートバイアスには最適値があるため、研究の背景でも述べたように、今後さらにスケーリングが進むとその効果が低下してくることが懸念されている。
- DTVS(Dynamic Threshold Voltage Scaling)
  VTCMOS などを用いて、回路の閾値を動的に制御する技術。

閾値電圧を変更するものに関しては、リーク電力を削減するためには高閾値にする必要があるが、閾値の高いトランジスタは動作速度が遅くなるというトレードオフの関係がある。。このため、タイミングエラーの発生確率が大きくなるため、十分なマージンを取った上での、設計・運用が必要となる。このため、

遅延補償フリップフロップによる動的なタイミング監視は、このような閾値の最適化や設計段階での無駄なマージンの削減などの効果があり、非常にこれらの省電力化技術と相性のいい技術であると考えられる。

電源電圧の変更

- MTCMOS（Multi Threshold-voltage CMOS）
  高閾値電圧（低速低リーク）のトランジスタをシャットダウントランジスタとして利用する。回路が動作していないときはこのトランジスタを off にすればリークによる電力消費は削減できるが、直列に高閾値のトランジスタが挿入されるため、回路動作が遅くなるという欠点がある。
- MSV(Multiple Supply Voltage)
  高速動作が必要な場所には高電源電圧を、そうでない部分には低電源電圧を供給する。

入力データ設定

- IVC(Input Vector Control)
  リーク電流が最小となるような入力値を設定する [26]。たとえば、off のトランジスタが、なるべく直列に並ぶように入力をコントロールするなど。

電源電圧を変更するものに関しては、ここまで述べてきたように、低電源電圧において動作速度が遅くなるというトレードオフの関係がある。。このため、タイミングエラーの発生確率が大きくなるため、十分なマージンを取った上での、設計・運用が必要となる。このため、遅延補償フリップフロップによる動的なタイミング監視は、このような閾値の最適化や設計段階での無駄なマージンの削減などの効果があり、これらの省電力化技術との協調による有用性は高いと考えられる。

なお、クロックゲーティングは主にダイナミック電力削減技術に、DVFS はダイナミック電力・スタティック電力をともに削減する技術に分類される。

第6章

ソフトエラー耐性

本章では、遅延補償フリップフロップがもつソフトエラー耐性について考察を行う。遅延補償フリップフロップは、前段の組み合わせ論理回路のソフトエラー耐性を高める効果も有する。従来からソフトエラーによるフリップフロップの値の反転を検出するのに用いられている技術の多くは、遅延補償フリップフロップにも適用可能であるため、フリップフロップと組み合わせ論理回路の双方において非常にソフトエラー耐性の高いシステムが実現できる可能性があることを示す。

6.1　ソフトエラーに関する問題

宇宙線などに起因する中性子や　α 粒子が、プロセッサに入射することで引き起こされるエラーをソフトエラーという (figure 6.1)。このようなソフトエラーの影響は、航空宇宙などの分野で従来から問題となってきたものであるが、近年の VLSI の微細化、低電圧化、高集積化に伴い、一般的なプロセッサでもその影響が顕在化してきている。これは、微細化、低電圧化により、許容できるノイズ電荷量が小さくなったこと、高集積化により単位面積当たりの素子数が増えて、ソフトエラー発生確率（SER：Soft Error Rate）が大きくなっていることによる。このようなソフトエラーは、メモリ、ラッチ、フリップフロップ等の記憶回路に射した際に、記憶されているデータが反転してしまう Single Event Upset (SEU) と、組み合わせ回路に微粒子が入射して過渡パルスが発生し、フリップフロップにこの過渡パルスが取り込まれてエラーとなる Single Event Transient (SET) の 2 つに大別できる。

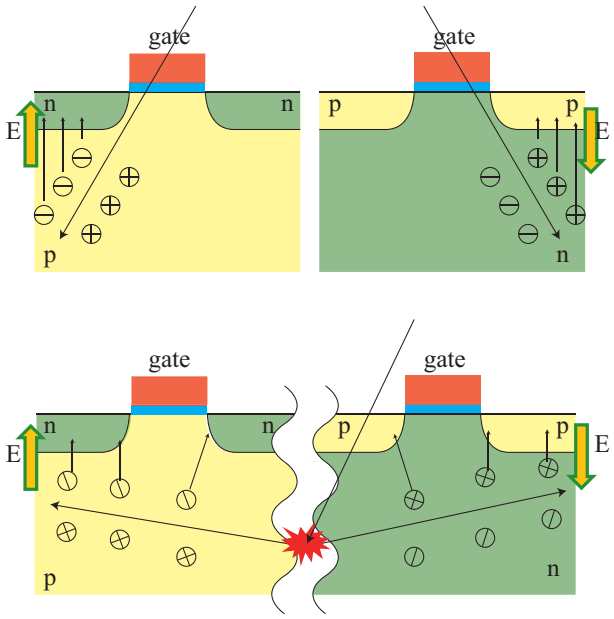


図 6.1: ソフトエラーの発生メカニズム

6.1.1　Single Event Upset (SEU)

メモリセルを構成するラッチや SRAM、DRAM などでこのようなエラーが生じた場合には、記憶されている値が反転することがある。DRAM においては、キャパシタンスに蓄積されている電荷が保持されているビットを表すため、ソフトエラーによって電荷が発生・または消失することは即保持されているデータの意図しない反転につながる。また、本稿で取り扱っているフリップフロップや、SRAM セルのようなインバータペアによってデータを保存しているタイプでは図 6.2 のように、あるノードの値が反転すると、その反転を維持するようなフィードバックがかかってしまうので、そのまま反転した値がラッチや SRAM

に保持されてしまうことになる。

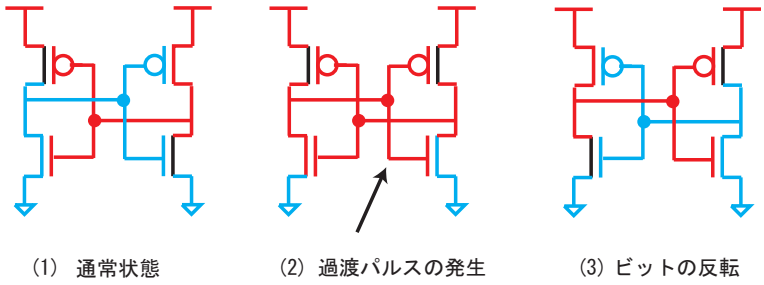


図 6.2: SEU の発生メカニズム

6.1.2　Single Event Transietn (SET)

組み合わせ論理回路でも、中性子や α 粒子が入射した際には過渡パルスが生じる。ただし組み合わせ論理回路で過渡パルスが生じたとしても直ちにエラーにはなるわけではなく、そのパルスが組み合わせ論理を伝搬してフリップフロップに取り込まれたときに初めてエラーとなる。しかし、このような組み合わせ論理回路上の過渡パルスの多くは以下にあげる要因などにより、フリップフロップに取り込まれず消滅することになる。

- 論理マスキング
  過渡パルスが論理的に出力に影響をおよぼさない場合は、その過渡パルスはそれ以上後方に伝搬しない。たとえば、2 入力 AND 論理の 1 入力が 0 であるならば、他方の入力にどのようなパルスが生じようとも出力は 0 で一定であり、後段に過渡パルスの影響が伝搬することはない。
- 電気的マスキング
  過渡パルスの持続時間が短いため、CMOS ゲートが十分に追従できず、振幅が減衰する。
- 時間的マスキング
  過渡パルスがフリップフロップまで伝搬しても、クロックエッジ前後のラッチウィンドウのタイミング中に入力されなければ、記憶されるデータに影響を与えることはない。

これらの効果のため、論理回路の SER は従来メモリの SER に比べて非常に小さく、従来その影響は非常に小さいものであった [34] [35]。

しかしながら、近年になってトランジスタの高速化により、急峻な過渡パルスにも素子動作が追従することができるようになってきたため、電気的マスキングの効果が小さくなった。さらに、動作周波数の増加により、クロックサイクルに占めるラッチウィンドウの長さが相対的に増加し、ラッチウィンドウマスキングの効果が小さくなってきており、SET の効果も無視できないものとなってきている [34]。また、過渡パルスが発生する場所によっては、後段のフリップフロップの値がマルチビットエラーになる可能性も高いことも SET の特徴であるといえる。

6.2　遅延補償フリップフロップの持つソフトエラー耐性

6.2.1　組み合わせ論理回路上の SET 耐性

前述の 3 種類のマスキング効果より、SET が問題となるのは過渡パルスが、ちょうどフリップフロップのクロックの立ち上がりのタイミングに入力に達する場合のみ、フリップフロップの値を書き換えてしまって問題となることがわかる。しかしながら、遅延補償フリップフロップにおいては、ラッチウィンドウの

タイミングで入力が変化すると、データの取り込みを遅らせることになる。また、SET の特徴として、間違ったデータになるのは過渡パルスが通過する一瞬であり、直ちに正しい値に戻ることになる。このため、連続して 2 度の入力値の変化が起きるため、フリップフロップのデータ取り込みのタイミングでパルスが発生した場合、遅延補償フリップフロップにおけるデータの取り込みは、入力データの値が正しい値になった後になり、組み合わせ論理回路で発生する SET に対して、遅延補償フリップフロップは高いエラー耐性を持つということがいえる。図 6.3 に、遅延補償フリップフロップの SET 回避動作の概念図を示す。

6.2.2　遅延補償フリップフロップ自体の組み合わせ論理部で起こるソフトエラーに対する耐性

遅延補償フリップフロップの実装の際に各フリップフロップに追加した要素のソフトエラーによる影響を考える。この追加部分が外部に出力している信号は、フリップフロップをコントロールしている Ctrl 信号, 最大遅延エラー、最小遅延エラーの 3 種類である。このうち最大遅延エラーはエラーの発生率のモニタリングに用いていること考えられるが、特に発生率の統計データが多少変動しても問題は発生しない。また、システムによっては過小遅延エラー信号は処理の再実行を誘発するため、処理のオーバーヘッドを生じる可能性があるが回復不能な状況に陥ってしまうことはない。

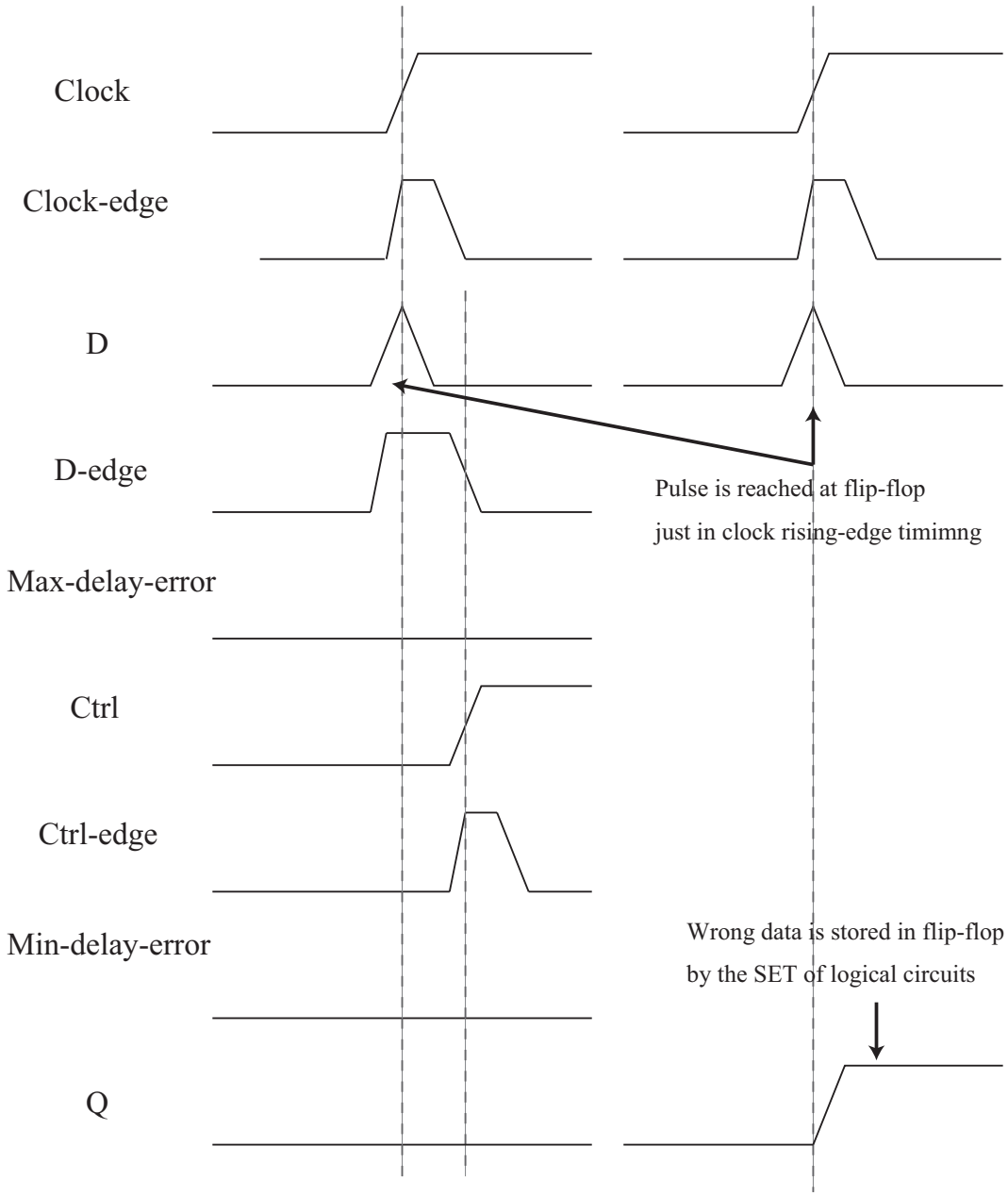
これにたいして、Ctrl 信号に過渡パルスが生じてしまうと、当該フリップフロップはその時の入力データの値を取り込んでしまうため、保持されているデータが反転してしまう可能性がある。また、Ctrl タイミング調整部に入力される、D-edge 信号は過渡パルスが発生した場合、Data が安定していないのに、D-edge の値が 0 となるようなパルスが発生した場合に Ctrl が立ち上がってしまい誤ったデータを取り込んでしまう可能性がある。しかしながら、これらのエラーは当該フリップフロップ外に直接影響を及ぼすことはないため、前段の論理回路中でソフトエラーが発生した場合のようにマルチビットエラーになることはない。結果的に、遅延補償フリップフロップの通常フリップフロップからの追加回路におけるエラーは、シングルビットの反転という症状を引き起こすため、これは従来から利用されている ECC などの SEU 耐性技術を使用することによって、対応することが可能である。これらの関連既存技術に関してはいくつかを章末にまとめて掲載する。また、これらソフトエラーが問題となる部分で使用されているトランジスタ数は少なく、面積的にこの部分でソフトエラーが発生する可能性は低い。

6.2.3　フリップフロップ上の SEU 耐性

遅延補償フリップフロップにおいてデータを保持している部分は通常のフリップフロップと基本的には差異はない。このため、誤り訂正符号など後の節で述べるような既存の SEU 耐性技術手法が遅延補償フリップフロップにも適応できる。

6.3　SET 耐性のシミュレーションによる検討

シミュレーションでは、図 6.4 に示すように入力として完全な矩形波を用いて、それを現実的な波形にするために Fanout-of-4 inverter (図 6.5) の前 2 段のインバータ（Shape Input）を通して成形したパルス（図 6.6）を通常のフリップフロップと DCFF に対して入力し、SET が発生するかどうかを調べた。このとき、さまざまな幅のパルスを入力パルスをクロックに対して様々なタイミングで入力し、フリップフロップの値が誤っているかどうかをシミュレーションを用いて調べた。これにより、実際にソフトエラーで起こるパルスが、完全にランダムなタイミングでフリップフロップに到着した場合に、そのパルスがソフトエラーを引き起こす確率（パルス到達時エラー発生率 (ERAP: error ratio by arrived pulses) とする）は



(a) Delay-compensation flip-flop　（b）通常のフリップフロップ

図 6.3: 遅延補償フリップフロップの SET エラー耐性

$$ERAP = \frac{range\ of\ pulse\ arrival\ timing\ which\ causes\ a\ error}{clock\ cycle} \tag{6.1}$$

として見積もることが可能である。この値はいわば、時間的マスキングが効果を持たない確率であるということができる。実際に SET によってエラーが発生する際の確率は、ソフトエラー発生確率 (SER: Soft error ratio) に、電気的マスキングや論理的マスキングによる影響、さらにこの値を掛け合わせたものであるといえる。したがって、ＥＲＡＰが 0 であれば、結果的に前段の論理回路がいかなものであるかとは無関係にエラーは生じない。

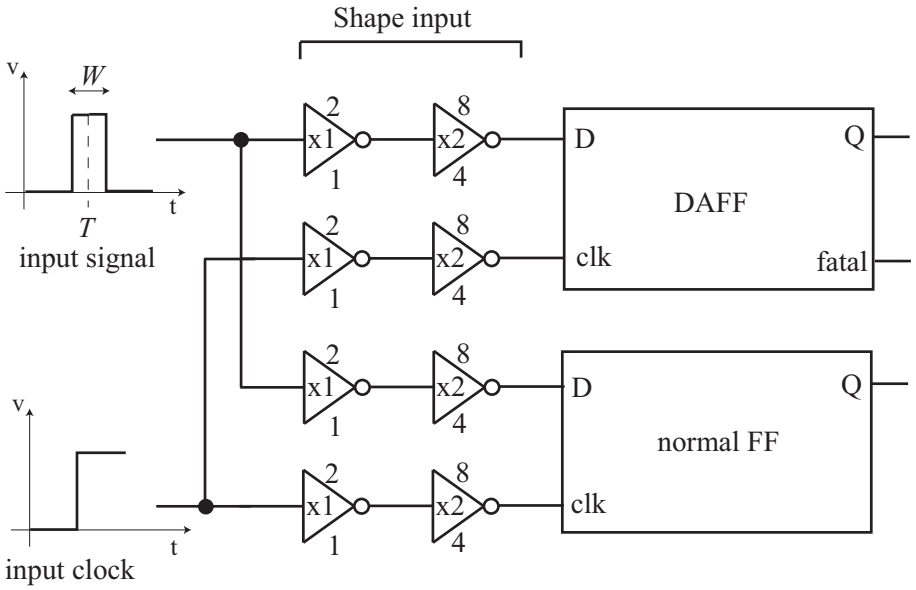


図 6.4: パルス到達時エラー発生率の評価を行った回路

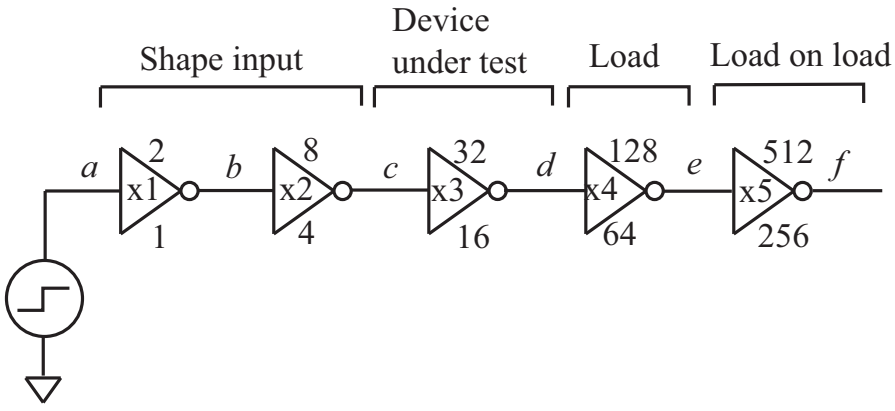


図 6.5: Fanout of 4 inverter

パルス到達時エラー発生率の算出する基準としたクロックの周期は Fanout-of-4 インバータ遅延の 30 倍とした。なお、シミュレーションによる Fanout-of-4 インバータ遅延は 180nm で約 80ps、45nm で約 15ps という結果になった（図 6.5 6.6、表 6.1）。

また、この手法の効果は到達するパルスのパルス幅の影響を大きく受けるため、どの程度のパルス幅が実際に発生すると見込まれるかが重要になる。そこで文献 [40] [41] を参考にソフトエラーによるパルス幅が図 6.7 のように変遷しているとし、シミュレーションを行うパルス幅の範囲を定めることとした。

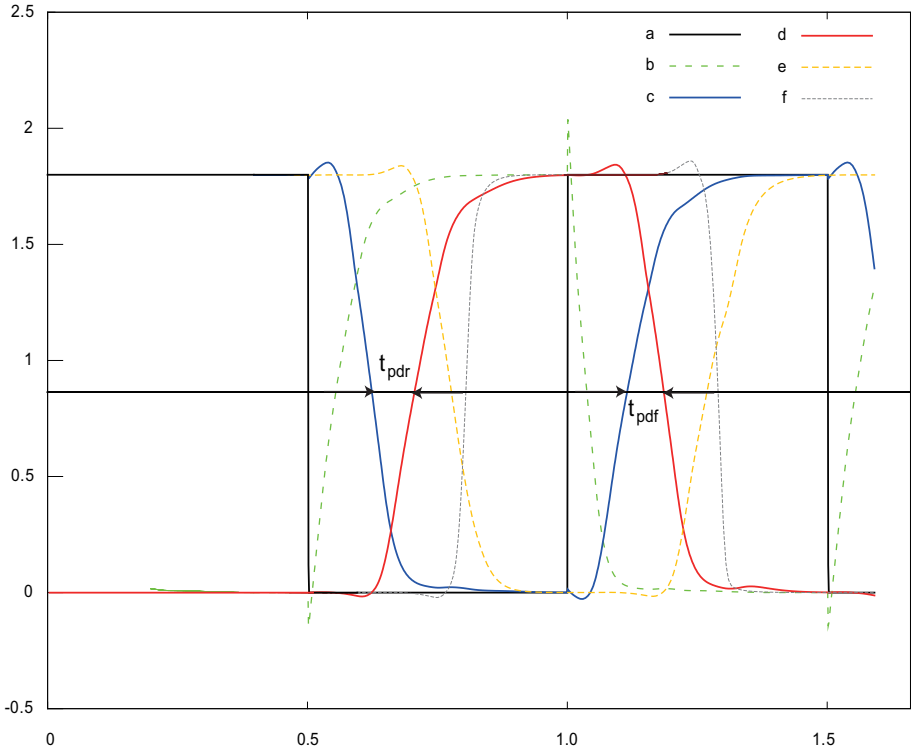


図 6.6: Fanout of 4 inverter delay of 180nm technology

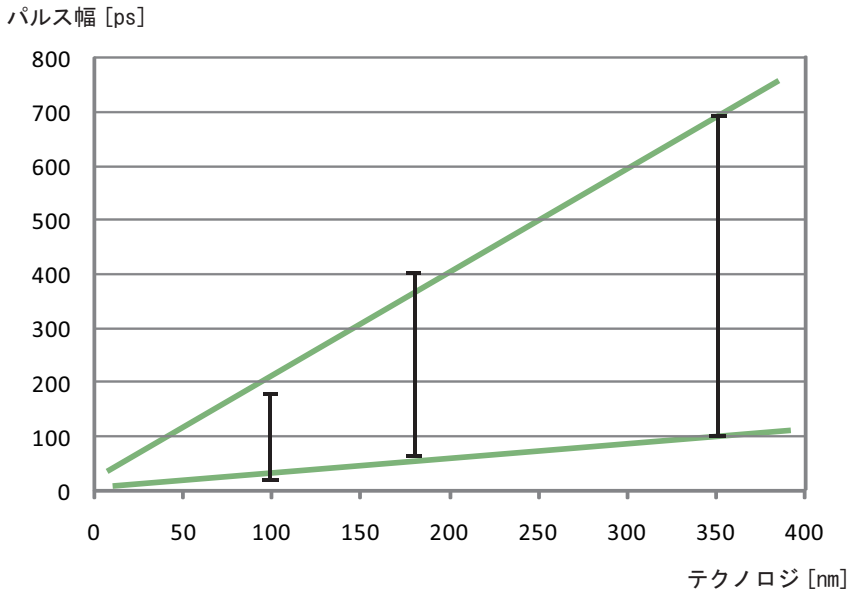


図 6.7: テクノロジによるＳＥＴのパルス幅の変遷

表 6.1: FO4 インバーター遅延.

Technology	Fanout of 4 inverter delay	Suply voltage
180nm	78ps (t_{pdf}: 67ps, t_{pdr}: 89ps)	1.8 V
90nm	22ps (t_{pdf}: 20ps, t_{pdr}: 24ps)	1.2 V
45nm	15.5ps (t_{pdf}: 14ps, t_{pdr}: 17ps)	1.0 V

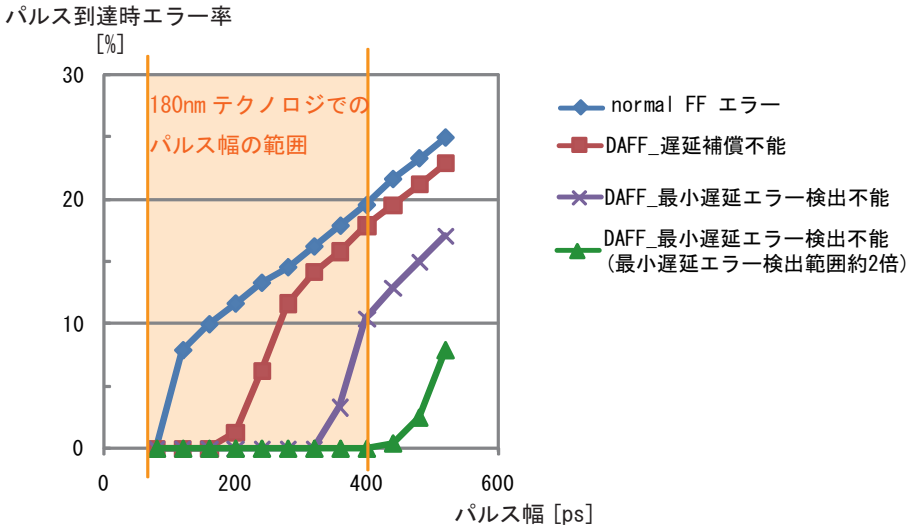


図 6.8: 180nm テクノロジにおける、パルス到達時エラー率

次に図 6.8 に遅延補償フリップフロップが、いかにパルス到達時エラー発生率を低く抑えるかについて述べる。シミュレーション結果によると、Fanout-of-4 inverter 遅延よりもパルス間隔が短いときにはどのようなタイミングで値が入ってきてもソフトエラーにはなっていない。これは、そもそも整形用にフリップフロップ前段に入れているインバーターの動作が追随しきれていないことによる、電気的マスキングの効果が表れているためであるということがいえる。図 6.7 に示すように、180nm のテクノロジにおいては、SET によって生じるエラーパルスはおよそ 50 から 400ps とされているので、多くの場合で SET のパルスがなくなった後に正しい値を取り込むことができる。また、汎用 CPU のようにプロセッサステートのフラッシュ・再実行などの機構を備えるものであれば、ショートパスエラー発生シグナルをトリガとしてプロセッサ状態の回復・再実行機構を同時に用いることで、ほぼ 100 %の SET 耐性を獲得することが可能となる。

さらに、図 6.9 6.10 に、それぞれ 90nm テクノロジ、45nm テクノロジにおけるパルス到達時エラー発生率を示す。ＳＥＴは微細化が進むほどその問題かが顕在化してくることが予想されているが、今後微細化が進行した場合、遅延補償フリップフロップの動作速度も、パルス幅の減少と同程度の高速化が見込まれるため、特に多くの追加回路などを用いなくても、同様のソフトエラー耐性を維持できると考えられる。

より高いＳＥＴ耐性を持たせたい場合は、データ信号遷移パルスの幅を広げることで遅延補償可能なパルス幅をより広い幅のパルスまで対応したり、Ctrl 信号遷移パルス幅を広げることでより広い幅のパルスに対してもＳＥＴを検出することが可能となる。しかしながら、データ信号遷移パルス幅を広げた場合には過大遅延エラー検出が、Ctrl 信号遷移パルス幅を広げた場合には過小遅延エラー検出が、それぞれ実際にはまだ余裕のある状況から検出され始め、回路のタイミング設計の自由度が下がることになる。つまり、信頼性と設計自由度の間には設計上のトレードオフが存在するといえる。

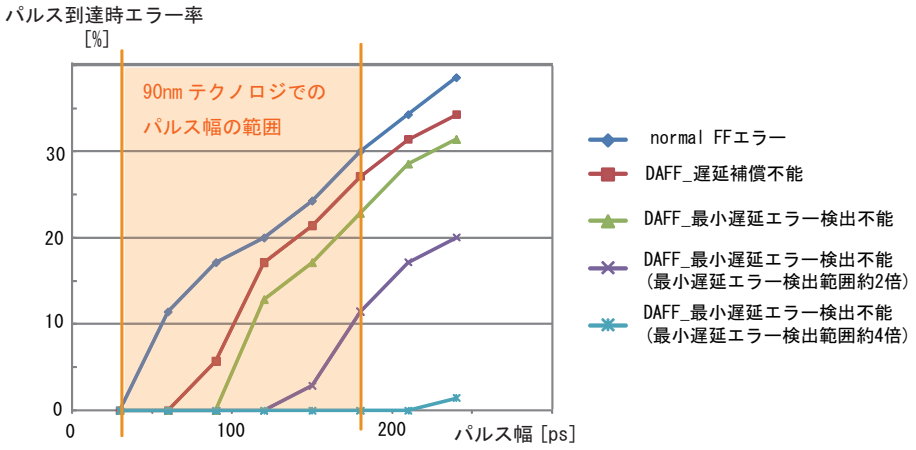


図 6.9: 90nm テクノロジにおける、パルス到達時エラー率

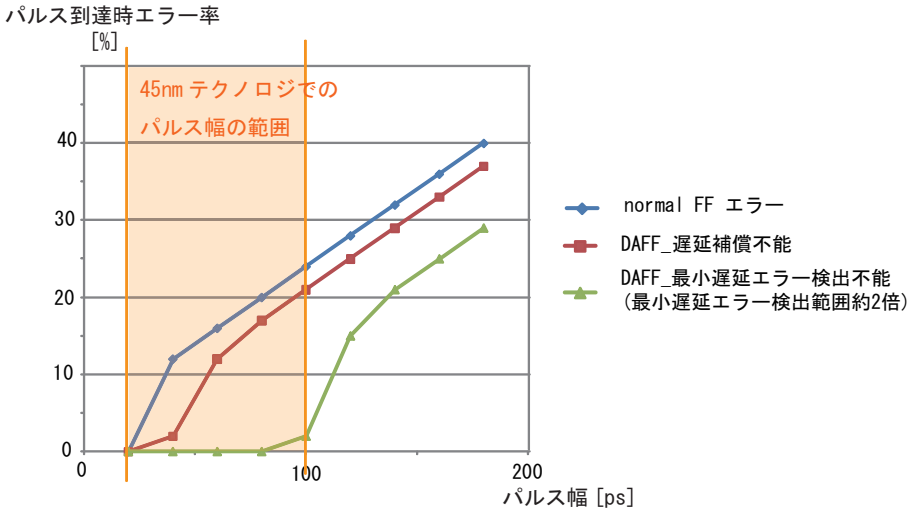


図 6.10: 45nm テクノロジにおける、パルス到達時エラー率

6.4　既存のソフトエラー対策技術

ソフトエラー耐性における技術は近年では商用のプロセッサにも搭載されるようになってきている。たとえば IBM 社の POWER6^{TM} では、高い信頼性をかくほするためにすべてのオンチップキャッシュに ECC を適用し、ほぼすべてのレジスタアレイとロジック部の 7 割にパリティチェックと剰余チェックが行われており、実際にエラーを検出した場合には Recovery Unit と呼ばれる部分に格納されている正常動作をしていた時点での状態を復元して再実行するようにしている [42]。このようにソフトエラーに対する信頼性の確保は近年より現実的な問題となってきており、その関連研究は多い。以下に代表的なものを幾つか述べ、遅延補償フリップフロップによるソフトエラー耐性との利害得失や協調の有効性に関する検討を行う。

6.4.1　SEU 対策技術

回路構成的に記憶素子の SEU 耐性を高める方法

- Dual Interlocked Storage Cell (DICE)

  DICE [43] は SEU 耐性を持つラッチとしてもともとは宇宙環境での使用における信頼性向上を目指して提案されたものである（図 6.11）。通常のラッチでは 2 つのインバーターをペアで駆動することで記憶を保持しているため、あるノードが反転すると直ちに他方のノードの値が反転し誤ったフィードバックがかかってしまうが、DICE では入力値と、入力の反転値を保持するノードがそれぞれ 2 つ存在し、それぞれのノードのプルアップ pMOS とプルダウン nMOS は異なるノードと接続されている。このため、放射性粒子によってあるノードの値が反転した場合でも、影響は隣接するノードのどちらか一方だけであり、ほかの 2 つのノードは正しい値を保持する。この手法はじっさいに、Los Alamos National Laboratory, Nerutron Science Center (LANSCE) において中性子ホワイトビームの照射下におけるソフトエラー耐性の実験的な評価が行われており、SER を 1/29 から 1/103 にまで削減することに成功している [44] 。なお、本手法では今後さらに素子が微細化した場合の SER に対する影響の予測も近年発表されており [45]、今後 SER は指数関数的に増加してしまうという見積もりが示されている。その理由は、素子が微細化すると 1 つの高エネルギー粒子が複数の拡散領域に電荷をもたらすことがあり、DICE では 2 つのノードが同時に反転してしまった場合にエラーを修復できないためである。電荷分割による複数ノードの反転の確率は拡散領域間の距離に関係するため、プロセスが微細化すると DICE のエラー耐性効果が小さくなると考えられているのである。

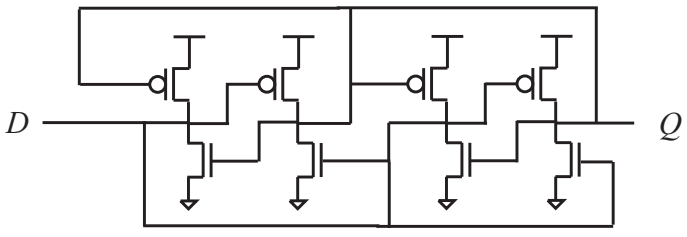


図 6.11: Dual Interlocked Storage Cell

- Soft error hardened latch

  上述のとおり、DICE は P 拡散、N 拡散のそれぞれに接続されたノードが 4 つあるため、エネルギー粒子に弱い面積が大きい。これに対して、荷電粒子の衝突に弱い面積を小さくすることでソフトエラー耐性を低くしようとするアプローチがある [46]。図 6.12 に示す回路は 3 つのノードから構成され、図中 PDH と NDH のノードは入力値 D を、DH のノードは入力の反転地を保持する。PDH、NDH のノードによって DH はプルダウンまたはプルアップされる。本回路で特徴的なのは、PDH,NDH の両

ノードがそれぞれ、P 拡散領域、N 拡散領域とのみ接続されていることである。荷電粒子入社によって発生する電荷が作るパルスの極性は、N 拡散領域、P 拡散領域によって清く性が決まっているため、PDH、NDH の両ノードが同時に反転することはない。アルファ線の照射事件を行った結果から、標準的なラッチと比較して 1/197 という大幅な SER の低下を達成している。

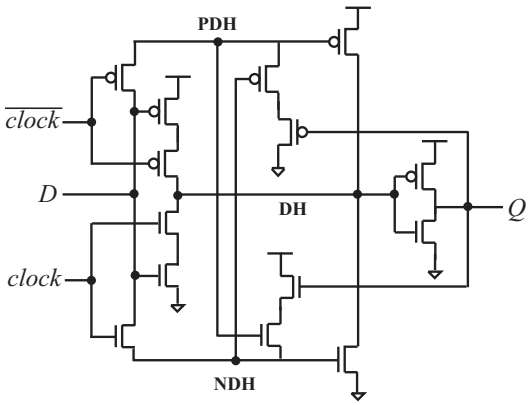


図 6.12: ソフトエラーを生じる面積を小さくしたラッチ

遅延補償フリップフロップに関しては、データの保持を行っている部分、つまり SEU の対象となる部分は通常のラッチが使用されている。このため、上記の 2 つの回路的にラッチの SEU 耐性を高める方法は遅延補償フリップフロップに対しても適用可能である。しかしながら、これらの SEU エラー耐性技術だけでは、フリップフロップコントロール信号生成部などの遅延補償フリップフロップ内の追加回路でソフトエラーが発生した場合には対応ができない。

記憶素子のデータ監視に基づく方法

- パリティ符号
  エラー検出によく用いられる非常に一般的な方法で、データの各ビットの排他的論理和を取る方法である。この排他的論理和をパリティビットとし、データ読み出しのたびに保持されたパリティビットと読みだされたデータの排他的論理和を比較しビットの反転を検知する。本手法では 1 ビットのエラーを検出し、2 ビット以上の偶数個のビットが同時に反転する場合には検出できない。また、エラーの訂正機構がないため、いったんエラーが発生した場合はリセットや再実行などシステムレベルでのエラー回復機構が必要となる。
- 誤り訂正符号 (ECC)
  誤り訂正符号も、データパス上のエラーを検出修正する方法として非常に有名な方法であり、SRAM の信頼性向上のために用いられるなど、その利用実績も豊富である。この手法では 2 ビットまでのえらを検出できるだけでなく、1 ビットのエラーを訂正することが可能となる。ECC にはいくつかの種類が、例としてハミングコードをとってみると、n ビットのデータに適用するのに必要なコードのビット数は

$$k = log_2(n + ki + 1) \tag{6.2}$$

  となり、面積、消費電力オーバーヘッドは大きい。
- Triple Module Redundancy (TMR)
  名前の通り、回路を 3 重化して、その出力を助けつ回路により決定するものである。一つの回路にエラーが生じても訂正することができるため、非常に堅牢で高信頼プロセッサ用とで用いられている技

術である。しかしながら、回路を 3 重化することによる、面積と消費電力のオーバーヘッドは巨大なものであることや、多数決回路が単一故障点となりうるなどの問題もある。

これらの手法は、フリップフロップの保持データが反転した場合のエラーを検出（ものによっては訂正まで）することができるため、遅延補償フリップフロップとの協調が非常に有効な技術である。なぜなら、上述のとおり遅延補償フリップフロップは、フリップフロップのデータ保持部と追加されている組み合わせ回路部のどちらでソフトエラーが生じても、1 ビットのエラーを生じる。したがって、1 ビットの反転が検出あるいは訂正できれば遅延補償フリップフロップ本体で発生するすべてのソフトエラーを検出することが可能であるといえる。このため、非常に短い周期で 2 つのソフトエラーが同時に起こるような状況を除けば、遅延補償フリップフロップと、記憶素子のデータの情報冗長性に基づくデータ監視を行うこれらの既存手法との組み合わせで、非常に対ソフトエラー耐性の高い回路を設計することが可能となる。

6.4.2　SET 対策技術

情報冗長性に基づくもの

- パリティ符号
  パリティ符号を組み合わせ論理回路に用いる場合、論理回路で起こったソフトエラーは複数の出力に影響を及ぼすことが多く、単純にビット間の偶奇をチェックするだけでは不十分な可能性がある。このため、組み合わせ論理回路では、パリティ符号を得る回路を別に用意したり、論理回路を共有し、単一の過渡パルスの影響を受けうる出力はグループ化し、グループごとにパリティチェックを行うなどの工夫が必要となる。さらに、加算器、乗算器などといった、出力に対する組み合わせ回路の重複が多い回路には適用が難しいという問題もある。
- 演算器に対応した検出法を用いる方法
  パリティ符号での対応が難しい加算器に対して、桁上げのチェックとパリティの予測を行う方法が提案されており [47]、これらの情報をソフトエラー耐性を高めるために用いることお可能である。この手法で提案されているパリティの予測方法はメモリユニットなどのほかの部分にも好影響があるとされているが、その用途は限定的である。

このほかにも、剰余チェックを用いるもの、算術符号を用いるもの、非順序符号を用いるもの、バーガー符号を用いるものなど、情報冗長性に基づくものとして分類されると考えられるものは多いが、適用可能な対象回路が限定的であったり、ロジック部に大きな追加回路が必要となったりすることが多い。これに対して、遅延補償フリップフロップでは、前段の組み合わせ論理回路から最終的にフリップフロップに入力されてくる信号の遷移タイミングのみをりようしているので、前段の回路の種類にとらわれない。また、通常通り回路を設計し、フリップフロップを遅延補償フリップフロップに変更するだけでソフトエラー耐性を向上させることができるため、回路設計も容易である、などの利点がある。当然遅延補償フリップフロップ自体にもオーバーヘッドはあるため、面積オーバーヘッド・消費電力オーバーヘッドなどの点での比較も必要となるが、本校においては今後の課題とし、詳細な比較は行わない。

空間冗長性に基づくもの　上述のＴＭＲは組み合わせ論理回路にも適用することができる。当然、組み合わせ回路の面積や消費電力のオーバーヘッドが大きくなるなどといった、記憶素子に適用する場合と同様の問題を含む。遅延補償フリップフロップでの回路がオーバーヘッドは通常のフリップフロップの 3 倍程度であるため、TMR によって 3 重化する元々の回路が、フリップフロップよりも大きい場合は、遅延補償フリップフロップを用いたほうが面積オーバーヘッドが少なくて済むことになる。また、SEU に用いる場合と同様に TMR を利用すると、正しい結果を選別するための多数決回路が単一故障点になることは避けられないという問題がある。

時間冗長性に基づくもの

- Built-In Soft Error Resilience (BISER)

  DFT（design for textability）のために用いられているスキャンフリップフロップをソフトエラー訂正に用いる方法が考案されている [15] [48]。図 6.13 中の信号 SCA, SCB, UPDATE, TEST を 0 とし、CAPTURE を 1 とすると 2 つのフリップフロップは並列な 2 つのフリップフロップとして動作する。これらのフリップフロップは C-element と呼ばれるエラー訂正回路に接続されており、この部分は 2 つの入力の両方が一致すると入力の反転を出力し、2 つの入力が異なると前の値を保持する。この手法ではもともとハードウェアに実装されているスキャンフリップフロップを有効利用しているため、ＳＥＴ検出のオーバーヘッドは少ない。また、この手法で構成されるフリップフロップは 2 つの独立したフリップフロップから構成されるため、値を保持するノードが組み合わさっているＤＩＣＥよりも、複数の拡散領域に同時にノイズ殿下が分割された時のＳＥＲは低いとされている [45]。

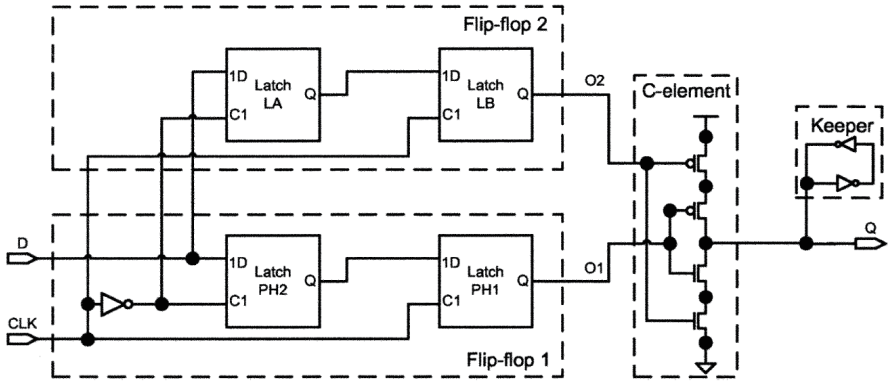


図 6.13: Built-In Soft Error Resilience [15] [48]

- Razor を利用する方法

  Razor とまったく同様の構成をＳＥＴ検出機構として動作させた手法もある [40]。つまり、ＳＥＴがおこったとしても、Ｒａｚｏｒが持つフリップフロップのどちらか一方にはラッチうウィンドウマスキングの効果がおこり、両フリップフロップに蓄えられるデータが不整合となるため、ＳＥＴのエラーが検出される。たとえば、メインフリップフロップでＳＥＴが発生したとしても、このときシャドーラッチではクロックが動作しておらず誤った値が取り込まれることはない。ただし、タイミングエラー検出のときとことなり、ＳＥＴの場合にはどちらのフリップフロップでエラーが発生したかを判別することができない。したがって、検出はできるが、メインフリップフロップの値とシャドーラッチに取り込まれている値のどちらを正しい値とうして利用するべきかを決められないため、ＳＥＴからの回復はじゅうらいのＲａｚｏｒの手法だけでは行うことができない。また、通常の一クロックサイクルでエラーを訂正する方法も、ソフトエラーの可能性があると使用できない。なぜなら、シャドーラッチに常に正しい値が保存されていることが保証されなくなるため、ただシャドーラッチの値を用いればよいというわけではなくなってしまうため、処理の再実行が必要となるためである。このため、ＥＣＣなどの既存手法と一緒に用いて、シャドー側でエラーが発生したのか、メインフリップフロップ側でエラーが発生したのかを判別するか、演算の再実行をするなどといった方法とらざるをえない。

- Canary FF を利用する方法

  今までに検討されている報告はないが、Canary FF を利用してもＲａｚｏｒと同様にラッチウィンドウマスキングの効果によるＳＥＴの検出が可能であることが期待できる。遅延素子より上流で荷電粒子入射によるパルスが生じた場合、1 対のフリップフロップにそのパルスが到着するタイミングは必

ず遅延素子の影響でずれるため、ちょうどクロック立ち上がりのタイミングで両フリップフロップの入力にＳＥＴのパルスが到着するということはありえない。また、遅延素子中でＳＥＴが発生した場合、メインのフリップフロップにそのような信号が伝搬するような経路が存在しないので、メインフリップフロップには正しい値が取り込まれる。このように、ＳＥＴによって両者の値が同時に誤るということは起こらないため、Ｃａｎary ＦＦによるエラーの検出は可能である。しかし、Ｃａｎａｒｙ　ＦＦでもタイミングエラー検出とちがい、どちらのフリップフロップでエラーが起こったかの判断はできないため、　Ｒａｚｏｒを用いてＳＥＴ対策を施す場合と同様の追加機構が必要となる。カナリアフリップフロップの最大の利点は、エラーの回復機構を必要としないこととされていたが、ソフトエラーの場合はメインのフリップフロップにエラーが生じる可能性があり、この利点を生かす場合には SET 耐性は発揮できない。

Razor やカナリアフリップフロップでは基本的にはエラーがどちらのフリップフロップで起こったかは判断できないため、処理の再実行などの処理オーバーヘッドと追加回路が必要となる。また、これらの冗長化されたフリップフロップでは当然SEU に対するぜい弱性が約 2 倍となる。また、SEU 耐性を高めるための手法を適用しようとするにしても、多くの手法では回路オーバーヘッドなどが約 2 倍となってしまう。

第7章

汎用プロセッサシステムへのDAFFの適用に関する検討

ここまでは、遅延補償フリップフロップによる遅延の後段への伝搬は高確率後段のタイミングマージンによって補償されるとしてきた。本章では、タイミングクリティカルな処理が連続するような部分における遅延の伝搬が後続のステージにどのような影響を及ぼすかについてのより詳細な考察を加える。また、ここまでは、遅延補償フリップフロップを主にパイプラインフリップフロップのような組み合わせ回路の後段で適用することを前提として議論してきた。しかしながら、近年の集積回路システムにおいては、メモリのキャッシュなど、何らかのメモリセルを有することが極めて多いので、実際にシステムを構築する上でメモリセルアレイに対して遅延が伝搬した場合に考慮すべき事柄を本章で述べることとする。

なお、本章の後半でＦＰＧＡを用いて実現した汎用ＣＰＵに対して遅延補償フリップフロップを適用する実験を行っているが、ベースとなったＣＰＵを実装するためのＶｅｒｉｌｏｇソースコードは、東京大学大学院情報理工学系研究科電子情報学専攻坂井研究室修士課程一年の杉本健氏が、本人の研究の一環として記述・作成していたソースコードを利用させていただいた [49]。

7.1　ステージ間にまたがる遅延補償

遅延補償フリップフロップではデータが遅れて届いても、それをそのまま後段に伝搬させることで、高確率で遅延が後段のタイミングマージンによって保障されることは上述したとおりである。しかしながら、2 段にわたって、タイミングクリティカルなパスが活性化する可能性は 0 ではなく、このような場合もエラーを起こさないように考慮する必要がある。以下に、クリティカルパスが 2 段連続で活性化する場合において、回路動作が何らかの外乱によって徐々に遅くなっていく場合について考察する。

遅延補償フリップフロップでは、クロックの立ち上がりに対してデータが遅れてきた場合には、Ctrl 信号が立ち上がっておらずクロックが立ち上がっているため、データはフリップフロップを素通りすることになる。このとき、後段のデータはこの前段の遅れとほぼ同じだけ遅れてフリップフロップを出発することになる。後段のクリティカパスの長さが同程度だと仮定する（許容されている遅延量の制約からこの過程は妥当である）と、後段に到達するデータの最大遅延はクロック遷移パルス幅 2 個分程度ということになる。なお、回路全体が同一周波数のクロックに同期して動作していると仮定すると、許されるタイミング制約の条件から、2 つのクリティカルパスの長さは同程度になり、動作速度の揺らぎの影響も同程度になると考えられる。このため典型的には、連続する処理のステージにおいては、遅延が 2 倍程度になる可能性がある。実際には、遅延補償が可能な最大遅延量の 2 倍程度の長さのショートパス検出が望まれるが、これは Razor よりもショートパスに対する設計規則が大きくなる可能性がある。

しかしながら、これら図 7.1,7.2,7.3,7.4, の挙動が十分ゆっくりと連続的に変化するという仮定の下では、大きな過小遅延エラー監視マージンは必要ないとすることができる。

7.2　汎用プロセッサ上での遅延補償不能な過大遅延に対する処理

ここまで見てきたように、遅延補償不能なほど大きな遅延を許すかどうかということは設計上の問題となる。つまり、過大遅延の検出が起こり始めたら早い段階で、供給電圧の昇圧などの対策をとり、遅延補償不能なほど大きな遅延の発生を許さないような設計を行う場合、過小遅延予測機構はショートパスの予測を行うだけでよく、特別な回復機構は必要ない。ただしこのような場合、許される最大の遅延量が遅延補償不能な過大遅延を許すような方式と比較するとエラーの検出範囲が狭くなるなどの欠点もある。

一方、過大遅延の検出が起こり始めてもすぐには対処を行わず、遅延補償できないような遅延が起きた場合には過小遅延予測機構によって検出をおこなうとすると、過小遅延予測機構がエラーを検知した場合には、ショートパスエラーを予測したのか、遅延補償不能な過大遅延エラーが起こったのかを判別できない場合がある。このような場合には、その時取り込まれたデータが誤っている可能性があるため、命令の

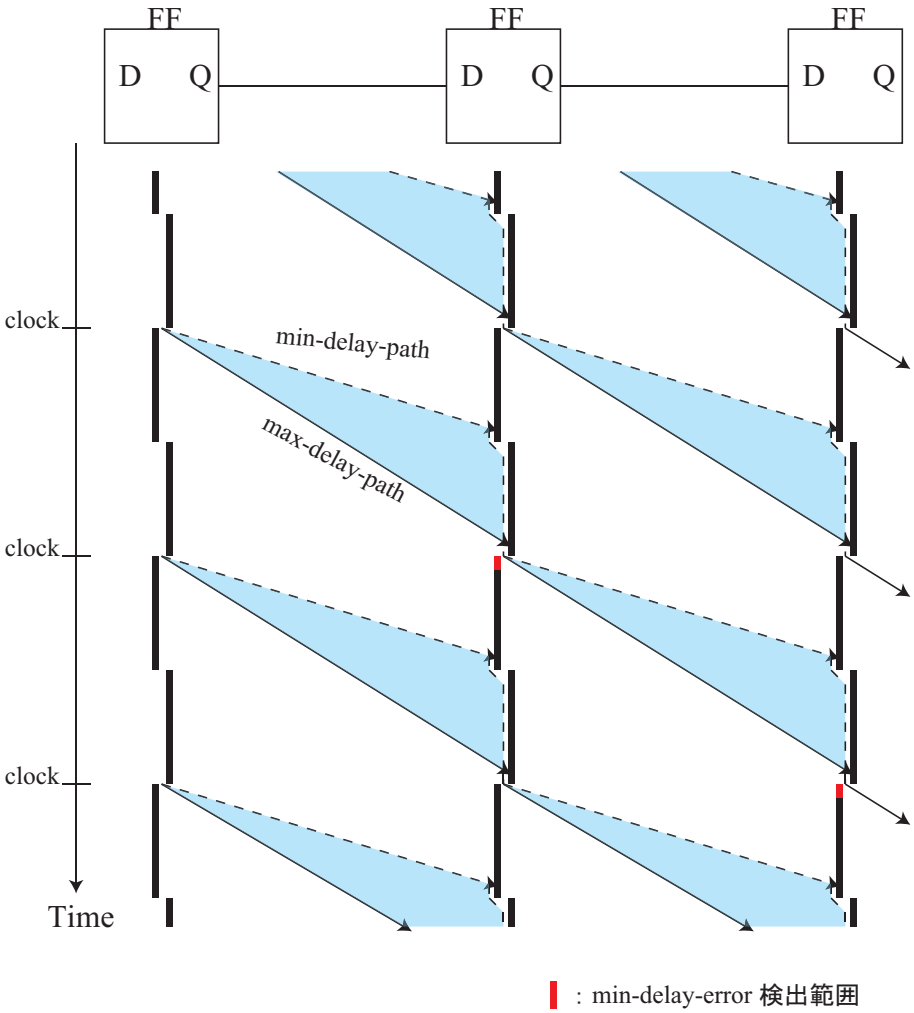


図 7.1: 通常動作時。セットアップタイムが守られているため動作は基本的には通常のクロックを用いている場合と同様。

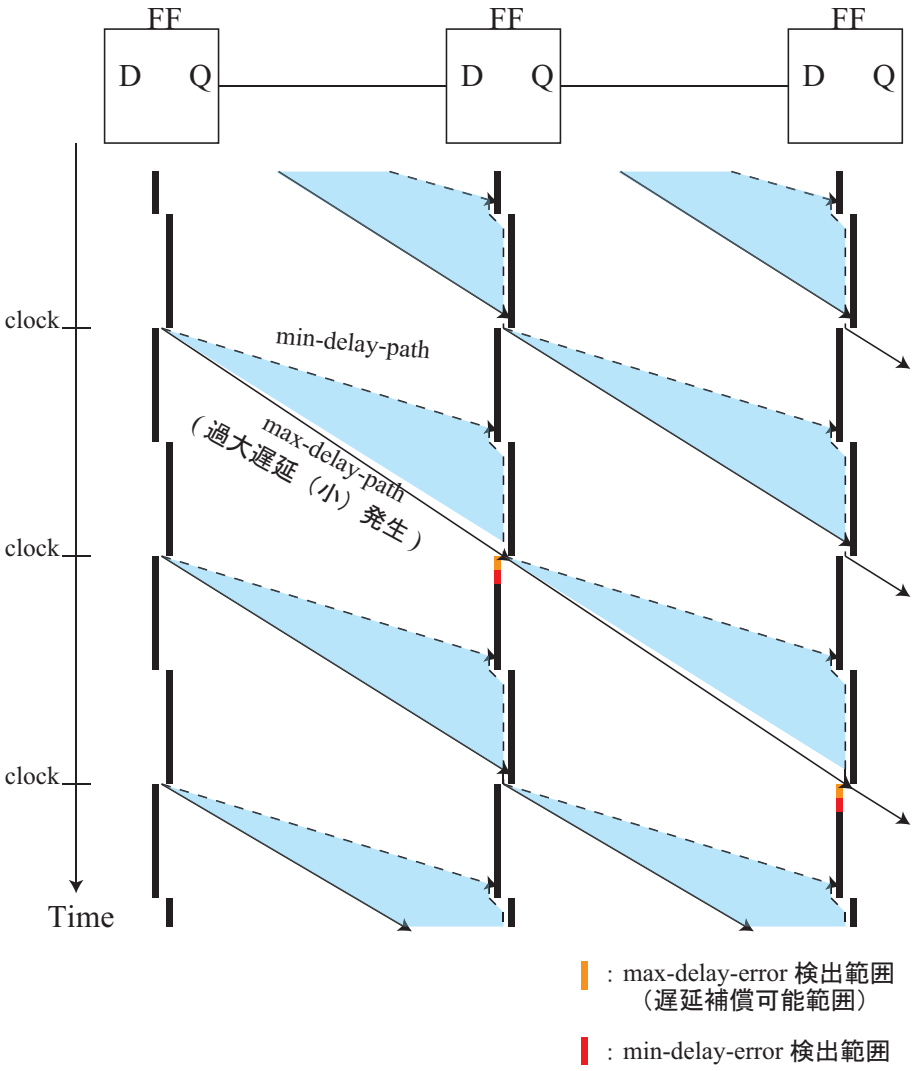


図 7.2: 遅延（小）発生時。遅延補償フリップフロップのマスタラッチとスレーブラッチにそれぞれタイミング調整された信号と、クロック信号が別に供給されている効果によって、D-to-Q 遅延が短縮されるため、大きな遅延の増大は発生しない。

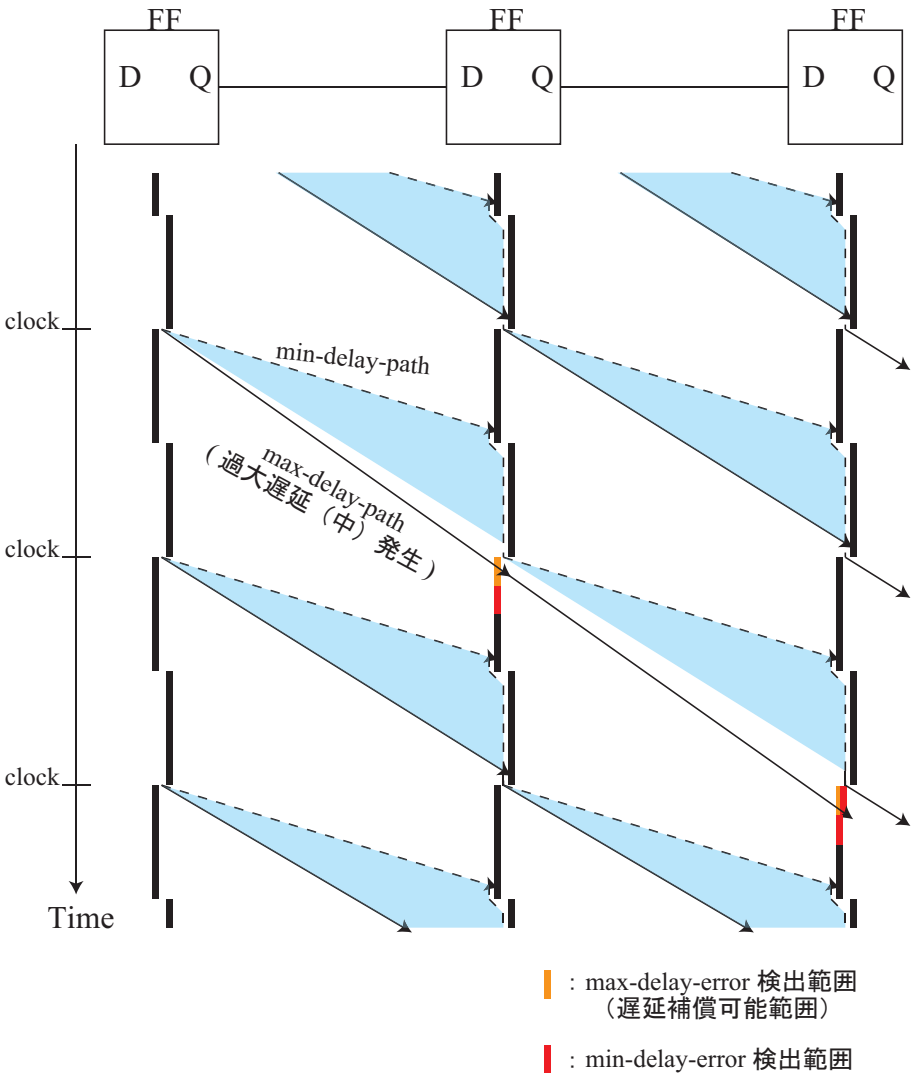


図 7.3: 遅延（中）発生時。遅延が雪だるま式に増える可能性がある。このため、連続して遅延が発生すると後段で遅延が大きくなりすぎたことが検出されるように、遅延補償可能な量と同程度の過小遅延制約監視のためのマージンを設ける必要がある。

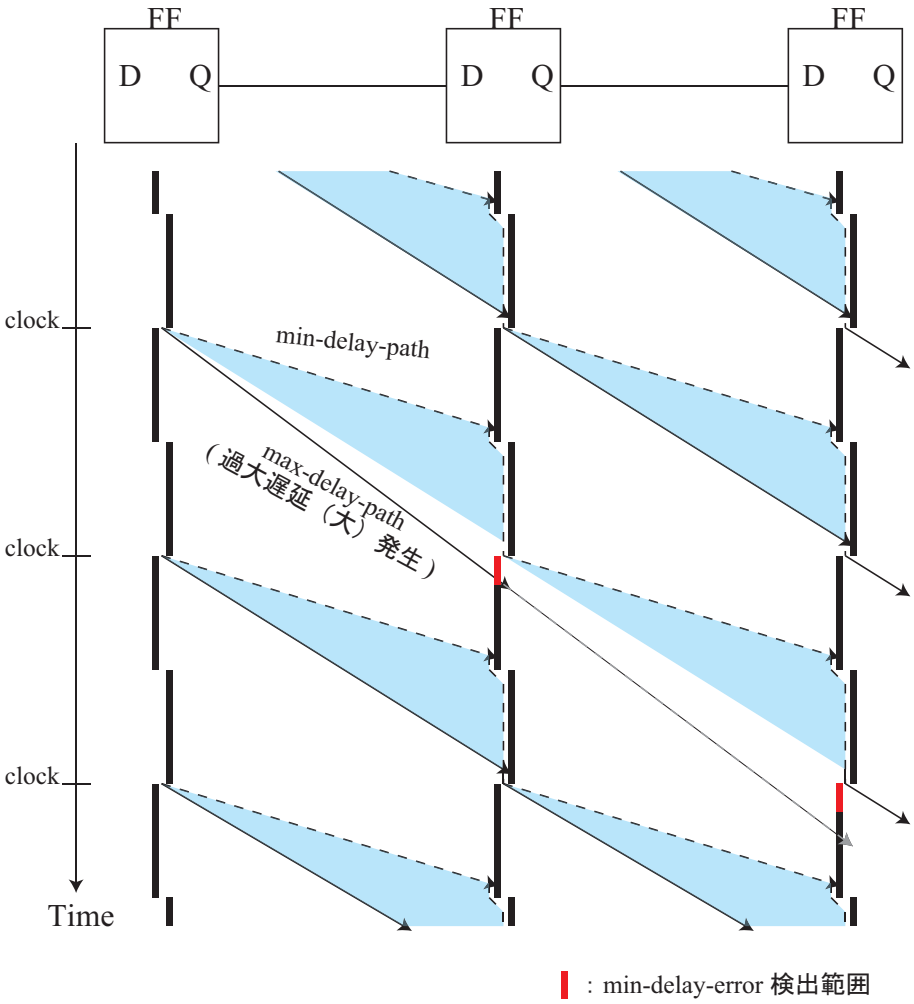


図 7.4: 遅延（大）発生時。この場合、一段目の遅延がすでに補償不可能な大きさになっているため、遅延は後段に伝搬しない。このような時連続して遅延が生じるとすると非常に大きくなり、必要な過小遅延マージンが大きくなりすぎることになるが、このような場合は考慮する必要はない。

再実行などの処理が必要となる。ここでは、効率的な再実行を行う手法として、近年の高性能プロセッサで多く用いられている out-of-oder プロセッサの例外処理に対する適応性を、リザベーション・ステーション・リオーダ・バッファ方式のプロセッサ中のデータパスを例に検討する。

7.2.1　例外処理による遅延補償不能な過大遅延に対する処理

Out-of-Oder プロセッサでは、プログラム順を無視して実行可能なデータから処理を開始する。しかしながら、プログラムの意味が変わってしまうようなことを行ってはいけないため、命令のフェッチ、命令の実行完了・プロセッサ状態の更新の確定などの部分はインオーダーで実行することになる。また、プログラム中に非常に多数存在する分岐命令を処理するために、分岐予測などの投機的な処理を行っている。これは、分岐命令の実行結果をあらかじめ予測して、より高確率で次に実行されるであろう命令を投機的に実行する手法であり、予測精度を高めるための研究は非常に多く近年では非常に高い分岐予測ヒット率を達成している。このような投機的な処理は分岐予測だけではなく、データアクセスがキャッシュにヒットすることを前提とした命令のスケジューリングなどでも利用されており、近年の高性能プロセッサでその考え方が使われている場所は非常に多い。しかしながら、常に 100%の予測が成立するわけではなく、投機的な実行が失敗した場合には速やかに必要な処理をやり直す方法が必要となる。プログラムの意味を変えず、無駄のない投機処理失敗後の再実行を行う場合には、ミスを起こした命令よりもプログラムオーダーで前の命令は実行の完了を確定し、当該命令以降の処理をフラッシュしてやるのが基本的な考え方となる。

このような手法はそのままタイミングエラーに対しても適用できる。つまり、ある命令がタイミングエラーを起こした時に、当該命令よりもプログラムオーダーで前にある処理は影響を受けないはずであり実行の完了を確定させ、当該命令以降の実行をやり直せば、安全で効率的な再実行が可能となる。リザベーション・ステーション・リオーダ・バッファ方式のプロセッサを例にとると、リオーダ・バッファが命令をプログラムオーダーでリタイアする役目を負っており、投機ミスの検出などはリオーダバッファからの命令のリタイア時に処理していることになる。このため、タイミングエラーが起こった命令には、命令に投機ミスを示すのと同様のエラーを示すビットを付加し、それをリオーダバッファまで当該命令と共に伝搬させてやればよい。なお、本章の冒頭も述べたが、本節における以上の検討と、これらの概念の FPGA を用いた検証は、本論文執筆時点では未発表であるが、東京大学大学院情報理工学系研究科電子情報学専攻坂井研究室の杉本氏が本人の研究として行ったものである。このときのタイミングエラー検出はカナリアフリップフロップを用いた遅延の予測を用いて行われていた。

なお、再実行を行う方式としても、ただリセットをするだけなど、他にもさまざまなものが考えられるが、　プロセッサ上のありとあらゆる部分に適用できるというわけではない。このため、遅延補償不能な過大遅延エラーを許すような設計を利用することが可能な部分は限定的であるということもできる。

7.2.2　FPGA による concept-proof 実験

そこで、本研究では杉本氏の構築した環境を元にして、必要な部分に手をくわえ、さらにタイミングエラー検出機構をカナリアフリップフロップから遅延補償フリップフロップに変更したもの実装し、両者の比較などを行った。図 7.5 のような DVFS 機能付きの FPGA のコンフィギュレーション用ボードを利用して実験を行った。このボードには 2 つの FPGA が搭載されており、各 FPGA の役目は表 7.1 のようになっている。また、このボードの結線の概要は図 7.6 のようになっており、0．6V から 1.75V まで 0.05V 刻み（高電圧域）または 0.025V 刻み（低電圧域）で、周波数は 1MHz から 100MHz ずつ 1MHz 刻みでの設定が可能となっている。また、DVFS 機能を実装している側の FPGA もユーザーがコンフィギュレーションすることが可能である。なお、Verilog のコンパイル・FPGA 書き込み用データの準備などは ALTERA® 社の Quartus™II version 7.2 を用いて行った。

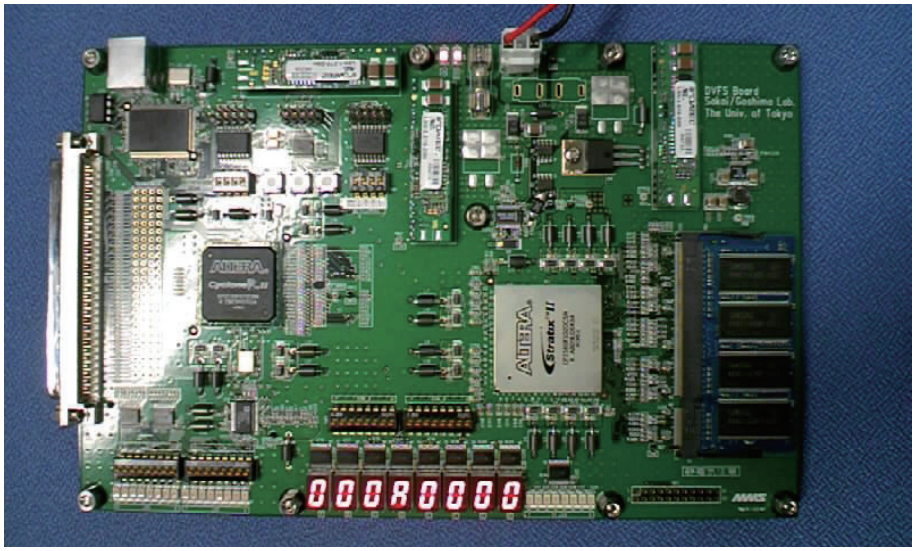


図 7.5: DVFS 機能付き FPGA ボード

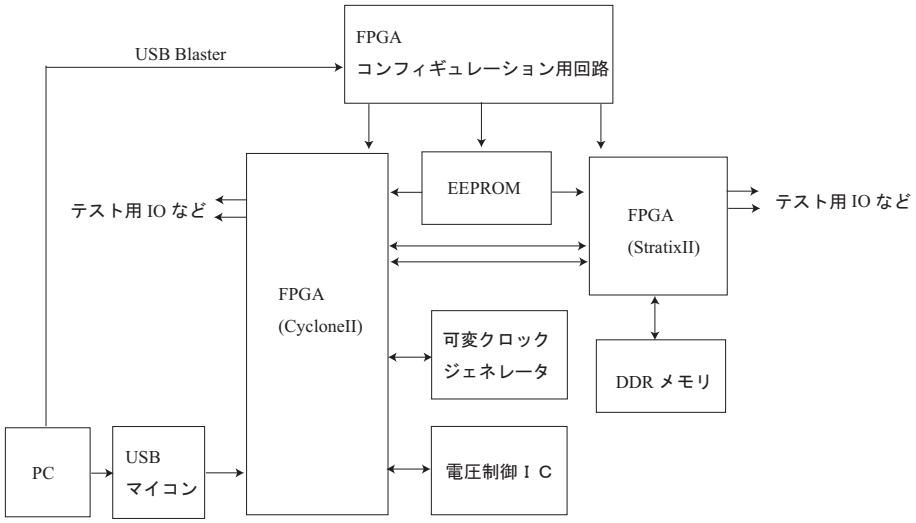


図 7.6: DVFS 機能付き FPGA ボードの結線図の概略

表 7.1: DVFS 機能付き FPGA コンフィギュレーションボードで使用されている FPGA と役割.

FPGA のタイプ	ALTERA®Stratix^{TM}II EP2S60F1020C5N	ALTERA®Cyclone^{TM}II EP2C35F672C6N
役割	検証対象ロジックの実装	DVFS 機能の実装

7.2.3　ＦＰＧＡによる汎用プロセッサシステムと遅延補償フリップフロップの実装

ベースとなるプロセッサは、図 7.7 のような構成を有する 2Way のアウトオブオーダプロセッサである。なお、図 7.7 からわかるとおり、ロード・ストアユニットっと整数演算ユニットのみしか持たない。また、プロセッサは 64 ビット RISC の Alpha AXP アーキテクチャに準拠しており、そのうちの整数系操作形式命令と、メモリ形式命令・分岐形式命令の一部に対応している。コンパイル時に生成されるタイミングレポートから、特にタイミングクリティカルなパスが存在する部分のフリップフロップを Canary Flip-Flop と遅延補償フリップフロップに変更して実装を行った。なお、ここまでの信号遷移検出回路で用いていたような、トランジスタのダイオード接続を利用した記述には、Quartus^{TM}II が対応していなかったため、遅延素子の前後の信号を NAND ゲートに入れるという、図 7.8 のような一般的な構成の信号遷移タイミング検出回路を実装した。

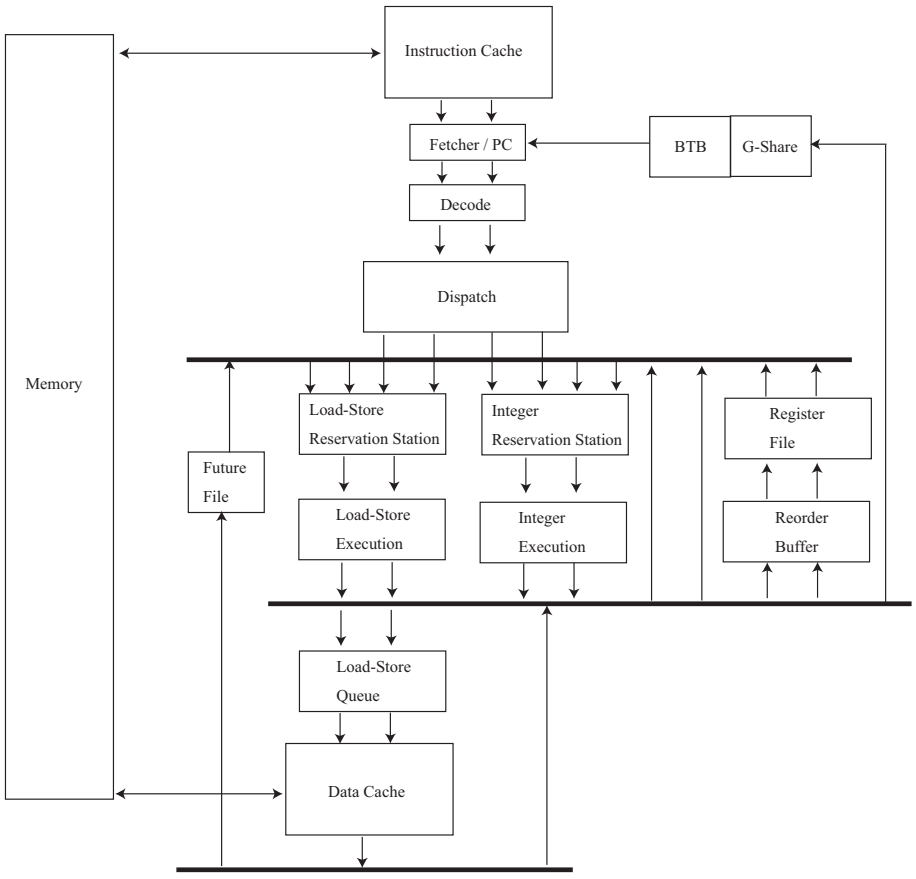


図 7.7: FPGA for measurement detailed waveform

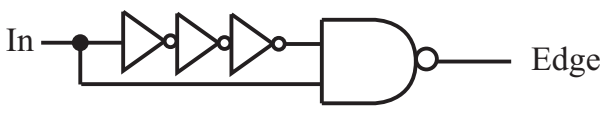


図 7.8: FPGA 上に実装した立ち上がり信号遷移タイミング検出回路

表 7.2: FPGA 上にカナリアフリップフロップ付き CPU を実装し DVFS を行いながらカウントアッププログラムを走らせた際の挙動.

Voltage[V]/Freq.[MHz]	15	16	17	18	19	20	21	22	23	24	25
1.5	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
1.45	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	Flush
1.4	OK	OK	OK	OK	OK	OK	OK	OK	OK	Flush	STOP
1.35	OK	OK	OK	OK	OK	OK	OK	OK	Flush	STOP	STOP
1.3	OK	OK	OK	OK	OK	OK	OK	Flush	STOP	STOP	STOP
1.25	OK	OK	OK	OK	OK	OK	Flush	STOP	STOP	STOP	STOP
1.2	OK	OK	OK	OK	OK	Flush	STOP	STOP	STOP	STOP	STOP
1.15	OK	OK	OK	OK	Flush	STOP	STOP	STOP	STOP	STOP	STOP
1.1	OK	OK	OK	Flush	STOP	STOP	STOP	STOP	STOP	STOP	STOP
1.05	OK	OK	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP
1	Flush	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP

表 7.3: FPGA 上に遅延補償フリップフロップ付き CPU を実装し DVFS を行いながらカウントアッププログラムを走らせた際の挙動.

Voltage[V]/Freq.[MHz]	15	16	17	18	19	20	21	22	23	24	25
1.5	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	DETECT
1.45	OK	OK	OK	OK	OK	OK	OK	OK	OK	DETECT	Flush
1.4	OK	OK	OK	OK	OK	OK	OK	OK	DETECT	Flush	STOP
1.35	OK	OK	OK	OK	OK	OK	OK	DETECT	Flush	STOP	STOP
1.3	OK	OK	OK	OK	OK	OK	DETECT	Flush	STOP	STOP	STOP
1.25	OK	OK	OK	OK	OK	DETECT	Flush	STOP	STOP	STOP	STOP
1.2	OK	OK	OK	OK	DETECT	Flush	STOP	STOP	STOP	STOP	STOP
1.15	OK	OK	OK	*	Flush	STOP	STOP	STOP	STOP	STOP	STOP
1.1	OK	OK	*	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP
1.05	OK	Flush	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP
1	Flush	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP

ただし、“*” は DETECT のうちで時々フラッシュが必要になったことを表す。

7.2.4　実験結果と考察

一般的なベンチマークでは、対応している命令数が足りず完全な動作ができないため、今回はデータを 1 ずつ加算していく簡単なプログラムを実行した結果について示す。正常動作している電圧から徐々に電源電圧を低下させていき、データのフラッシュが必要となる電圧を見つける。なお、カナリアフリップフロップに関しては、エラーの検知後に、遅延補償フリップフロップは過小遅延エラー予測が出力された直後に、フラッシュを行うようにしている。本来は、フラッシュの直後に電源電圧の昇圧を行うべきであるが、今回は検証対象 FPGA から、DVFS の制御用 FPGA にアクセスして自動的に昇圧を行うような実装は行っていない。この点に関しては今後の課題とする。

表 7.2 7.3 は、単純なカウントアッププログラムを走らせながら、電源電圧や動作周波数を調整した結果である。実験結果より、遅延補償フリップフロップもカナリアフリップフロップ同様にエラーの検出を正しく行っていることが確認できた。しかしながら、電圧変更の最小単位が大きいため、正確な特性の評価はできていない。

また、このようにエラーを検出を示す情報を、たとえばリオーダバッファなど、後段の特定のユニットまで伝搬させる場合は、若干カナリア方式のほうが遅延補償フリップフロップや Razor よりも有利である可能性がある。なぜならば、カナリアフリップフロップがタイミングエラーを検知する瞬間は、原因となるデータが取り込まれるクロックの立ち上がりタイミングと同時である。これに対して、遅延補償フリップフロップで、過小遅延エラー信号が生成されるのは、クロックが立ち上がり、一瞬遅れてフリップフロップコントロール信号が立ち上がった後に、データの変動があった場合に生成されるため、カナリアフリップフロップに対して少し遅れる。また、Razor では、エラーが判明するのは遅延クロックの立ち上がりの瞬間である。本節で取り上げているような方法では、エラービットを該当データと一緒に後段に伝搬する必

要があるため、後段のフリップフロップのデータの取り込みタイミングにエラー信号が間に合う必要があるので、エラーの検出が遅くなる 2 つの手法では、エラービット自体がタイミングエラーを起こさないように注意を払わなければならない。このため、前段のエラー信号が比較的早く生成されるカナリアフリップフロップは、タイミングエラー伝搬に利用することが比較的容易であるということがいえる。

7.2.5　要素回路の動作の詳細な評価

遅延補償フリップフロップで肝となる、信号遷移タイミング監視回路が実際に FPGA 中でどのような挙動を示しているかをより詳細に解析した。図 7.9 は信号波形の検証用に設計・作製した FPGA のコンフィギュレーション用ボードであり、Active Serial Configuration と JTAGConfiguration に対応し、FPGA への供給電圧・I/O ピンの入出力電圧・クロックの周波数を連続的に変更することができる仕様としている。信号線検出回路の遅延を変更しながら立ちあがり遷移検出回路の出力波形を測定したところ、確かに遅延が大きくなるほど信号遷移監視回路の出力パルス幅が長くなることがわかった。このうちのいくつかを図 7.10 に示す。

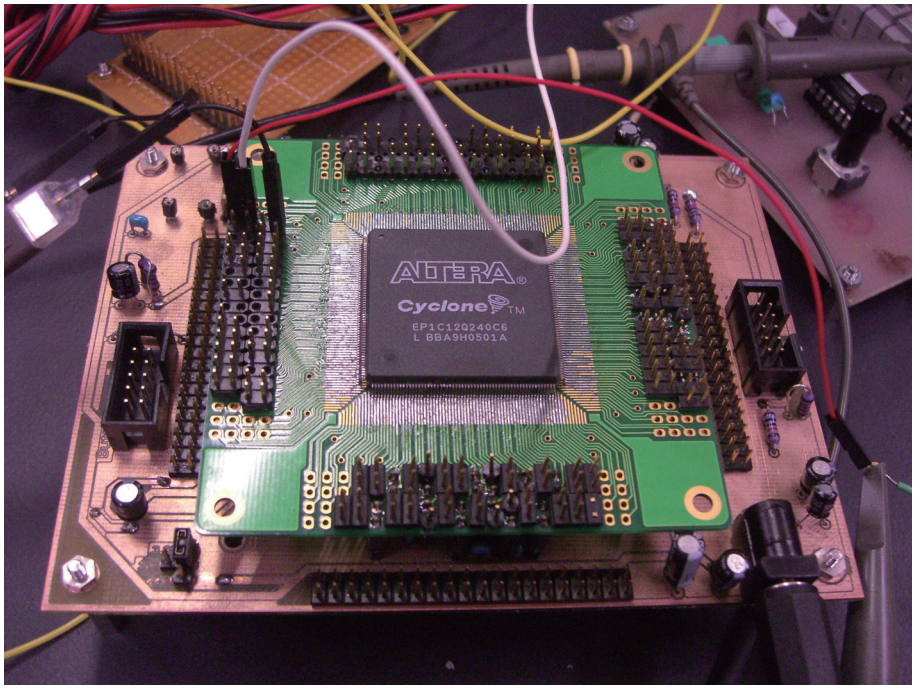


図 7.9: FPGA for measurement detailed waveform

本実験の結果より、信号遷移タイミング検出回路を遅延素子と通常の NAND ゲートを利用して実装した場合でも、前節で使用した DVFS 機能付きの FPGA ボードに実装したプロセッサの遅い動作周波数にくらべて、十分高速な応答を示していることがわかる。

7.3　メモリセルの保護

近年の集積回路システムのうち、とくにこのようなクロック同期やタイミングエラー等を考慮しなければいけないような規模のものは、特殊な用途のものを除いてキャッシュやレジスタファイルといったようなセルアレイ状のメモリユニットを持つことが多い。当然アプリケーションに応じてメモリセルの規模の大きさにはさまざまな種類のものがあると考えられるが、遅延補償フリップフロップを何らかのシステムに導入する際にはこのようなメモリセルの保護も当然ながら必要となってくる。なぜならば、他のロジッ

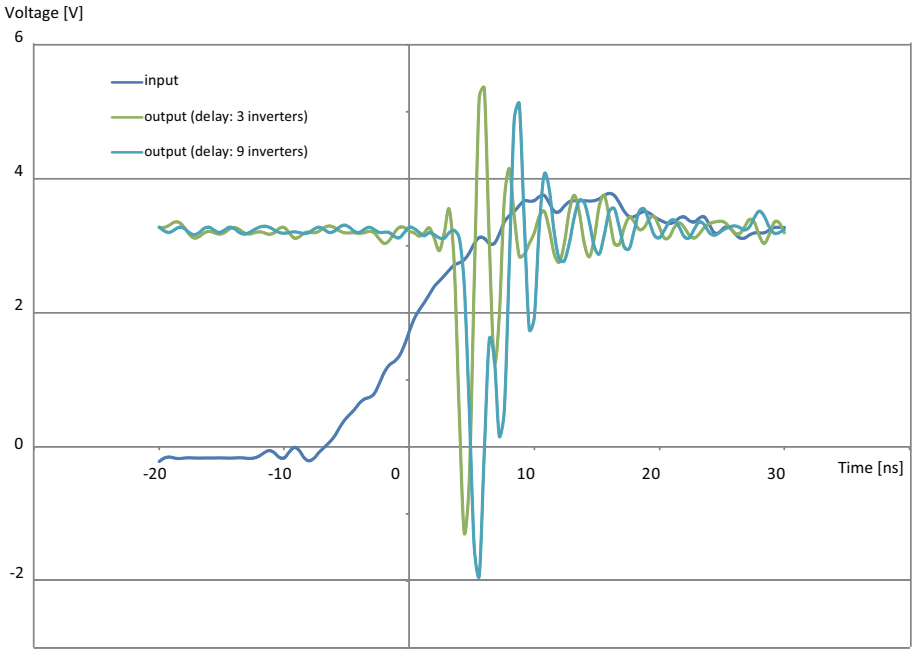


図 7.10: 波形測定用 FPGA に実装した、立ち上がり信号遷移検出回路の動作の様子。実際に動作させ、入出力信号を FPGA と直結しているピッチ変換基盤のピンからオシロスコープをもちいて測定を行った。

ク回路で演算された結果をデータやアドレスとしてメモリセルにアクセスするのが一般的であるが、そのアクセスとなる起点のデータが遅延補償フリップフロップによって遅れてくることが考えられるためである。メモリセル前段には遅延補償フリップフロップを用いずにシステムを構築することも考えられるが、メモリセル中のアクセスにかかる時間も物理的な場所によって異なるため、前段から伝搬された遅延がメモリセルアクセスで補償される可能性は十分にある。このため、省電力特性などを考えた場合、メモリユニット前段に入れても遅延補償フリップフロップが効果を発揮する可能性がある。また、メモリセル中のトランジスタ速度も電源揺らぎなどの影響を受けるため、メモリセルのエラー監視機構を設けること自体はディペンダブル VLSI の実現のための要素技術として不可欠なコンポーネントであるといえる。本節では、提案手法の主たる考え方である、動作時のタイミングエラー信号遷移監視に基づくような手法でエラーの監視を行う手法を提案する。以下で、ワードラインでコード時、データ読み出し時、データ書き込み時の各ステージごとに、検討を加える。

7.3.1　ワードラインのデコード時

ワードラインのデコードが遅れると、非常に危険な状態を引き起こす可能性がある。多くの場合、ワードライン・デコードの最終段階、あるいは比較的後段の処理にはダイナミック論理が使用される。これは、最終的なステージまで通常の CMOS のようなロジックで実装してしまうと、演算中に本来アクセスされるべきでないワードラインにグリッチが発生してしまうためである。書き込み時にこのようなグリッチが発生してしまうと、本来書き込まれるはずのないワードラインに接続されているセルに対して書き込みが行われてしまい、正しいデータが失われてしまう可能性がある。また、読み出し時にグリッチが発生した場合、本来読みだすべき値と反対方向にビットラインの電位が大きく傾いてしまい、正しいワードラインがアサートされた後に、本来の正しい値にビットラインが反転しない恐れがある。

したがって、このダイナミック論理が評価フェーズに入る前に、デコードの処理は終わっていなければいけない。あるいは、デコードが完全に終わってからダイナミック論理が評価フェーズに移行するように調整する必要がある。

ここでは、Source coupled logic (SCL) を利用したデコーダ [50] に動作時の信号遷移監視を適用し、グリッチの発生や、それにともなう間違ったワードラインのアサートが起きないようなデコーダとして図 7.11 のような回路を提案する。SRAM ではクロックから遅延素子を通してタイミング信号を生成するのが一般的であるが、まず初段のアドレスバッファで遅延補償フリップフロップの Ctrl 信号を SCL のイネーブル信号として用いることで、確実にデータの処理を行う。次に、これらのデータが全て出そろうのを確認するために、初段のアドレスバッファの SCL がドライブする 1 対の配線の排他的論理和をとる（実装上は両方 '0' か、どちらか一方が '1' であることから XOR ではなく OR でもよい）。この時点で、入力されるデータの到着タイミングのばらつきをそろえ、タイミングエラーを監視することで、後段の回路でグリッチが発生するのを防いだり、データ信号の安定よりも後に SCL のイネーブル信号がアサートされることを保証する。Pre-Decode のステージの SCL 実装された OR がタイミングエラーを起こさなければ、正しいワードライン一本のみがアサートされることになる。なお、ワードラインのアサートからディアサートまでの時間が非常に短く、書き込みに十分ではなかったり、完全にワードラインの信号が立ち上がらなかったりする可能性は、本手法を適用しても起こりうる。そのような場合は、次節以降に説明する手法で検出を行い、エラーが発生した場合はメモリアクセスのやり直しなどの措置を行うことになる。

7.3.2　書き込み時

読み出し時、書き込み時のタイミングエラーを監視するため図7.12 のような機構を検討する。書き込み時にはドライバによってビットラインに書き込むべきデータをセットした上で、ワードラインをアサートし書き込みたいラッチペアに値を書き込む。このため、以下の 3 点

- ワードラインがきちんとアサートされたか
- ビットラインが確かに書き込みたい値に変化したか
- セルに対する十分な長さのアクセス時間があったか

について監視すればよいということになる。そのため、以下のような手法を用いる。まず、ワードラインはロー・デコーダでどのラインがアクセスされるか決められたのち、当該ラインがドライブされる、このときデコーダー側から順に電位は伝わっていくため、デコーダと反対側の終端に、ワードラインがアサートされたかどうかをチェックするような機構を設ける。通常、アクセスされるワードラインは一本であるため、どれか一本がアクセスされたことを検知するための OR 回路を用意しておく。

同様に、ビットラインが確実に書き込みたい値に変化したかどうかに関しては、ビットラインのドライバと反対側の終端に排他的論理和を設けて監視する。ビットラインはデータ書き込み時に、対になる 2 つのラインが必ず逆方向にドライブされていなければならないため、これら排他的論理和の AND を取ることで、ビットラインが正しくドライブされていることを確認する。

さらに、ワードラインデコーダーと反対側にチェック用のセルを 1 column 用意する（以降この column を check column と呼ぶ）。この check column ではビットラインのドライブがほかの通常の Column よりも遅く開始されるようになっている。これに対して、プリチャージはビットラインのドライブの終了後ほかのセルとほぼ同様にのタイミングで行われる。このため、全 Column 中で最もワードラインがドライブされている期間が短く、この Column で十分にビットラインのドライブ時間が長ければ、通常の Column では十分のアクセス時間があったとしてよい。また、プリチャージの開始、ワードラインのディアサートはクロックに同期するため、ビットライン・ワードラインともに十分な長さのドライブ時間が以上の手法で確認できたならば、セルに対するアクセス時間（ワードライン・ビットラインがともにドライブされていた時間）は十分にあったとすることができる。check column は常に '0' が書き込まれるようにしてあるものであるから、ビットラインが '0' 担っていた時間が十分にあったかどうかを遅延素子と NAND ゲートなどを用いて監視してやればよい。

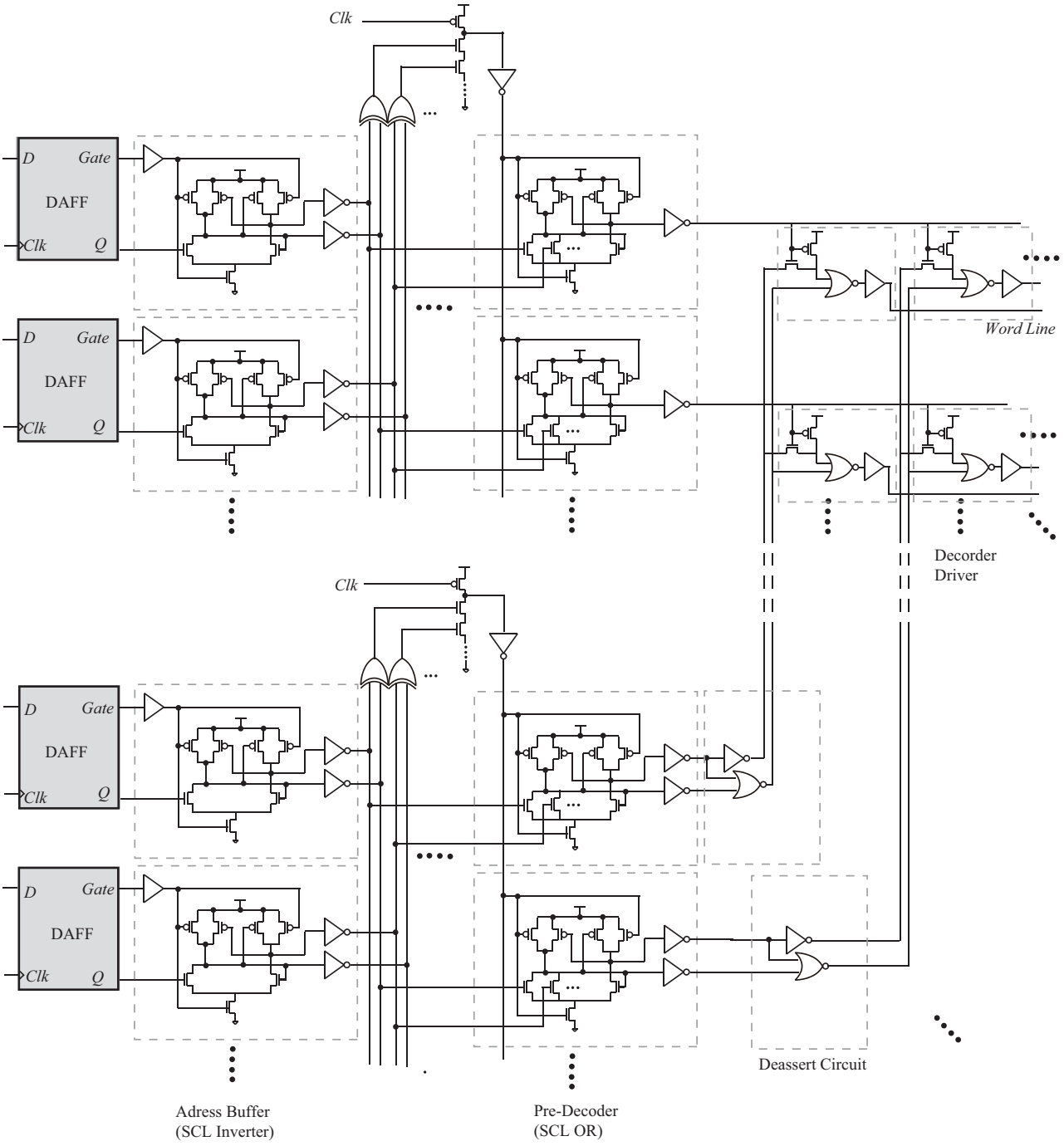


図 7.11: 遅延補償フリップフロップ後段に挿入するワードラインデコーダー

本来は、ワードラインのドライブが完全に終了するまで、ビットラインは書き込まれるべき値になっていることが望ましいが、仮にワードラインのドライブが完全に終了する前にプリチャージが開始されても重大なエラーに結び付く可能性は少ない。なぜなら、両方のビットが '1' となっても値は反転しないからである（もしこれで反転するならば、読み出しの時にデータを破壊してしまうことになる）。

なお、非常に大きいサイズのメモリに本手法を適用する場合には、AND、OR の各ラインにセンスアンプを設けるなどの追加の対策が必要となる。逆に、非常にビット数が少ないような場合には、ワードライン終端に遅延を挿入するなどして、必要とするマージンの幅をコントロールする必要がある。

7.3.3　読み出し時

読み出し時には、当該ワードラインをドライブし、そのあとにセンスアンプをイネーブルし、ビットラインの値を読み取る。このため、以下の 3 点

- ワードラインがきちんとアサートされたか
- ワードラインアクセス後十分にビットラインの値が変化してからセンスアンプが作動したか
- サイクル内で遅滞なくデータが読みだせたか

について監視を行えばよい。

まず、ワードラインに関する監視は上記の書き込みの保護と同様の方法で行う。SRAM の読み出し機構ではセンスアンプが用いられていることが多いが、この部分のセンスアンプイネーブル信号が速すぎず、遅すぎないタイミングで到達しているかどうかを監視する。センスアンプ信号が遅く到達しすぎると、当然ながらセンスアンプが動作しないため、NAND ペアにつながっているノードはどちらもプリチャージされている状態であり、データがは SR ラッチにとり込まれない。また、信号はドライバと逆の終端にあるセンスアンプに最も遅く届くため、センスアンプイネーブル信号のドライバと逆の終端に位置する Check Column の読みだした値を確認すればよい。この値が十分早く 0 になっていなければ、他のセルでもアクセスが間に合わなかった可能性がある。そこで、Check Column の読み出しデータを監視し、この信号がデータ読み出し時に適切なタイミングで '0' になっているかを監視すればよい。次に、信号が早く到達しすぎる場合を考える。とはいえセンスアンプイネーブル信号のアサートもクロックなどの制御信号に同期しているので、クロックより早く到達することは考えられない。信号が速く到達しすぎるというのは、ワードラインが当該ビットをアクセスしてからの相対的な遅れとして考えることができる。仮に十分にビットラインの電位差ができていないときにセンスアンプイネーブルがアサートされたと考えると、このようなフリップフロップでは正帰還がかかるため、本来読みだべきすデータとは関係なく素子の閾値バラツキの影響など増幅して、値を読みだしてしまうことになる。このため、どのワードラインがアクセスされても常に値が '0' である check column において、センスアンプの比較対象となる電位をわざと VDD よりも小さくすることで、その値が正しく "0" と読み込まれているかを監視する。なお、通常の SRAM セルの設計においては配線の太さが支配的であり、当該セルに必要な配線の面積内で素子領域を可能な限り大きく設計するような方針がとられることが多い。このような場合には、ワード線は非常に細長くなりがちであり、その分 RC ディレイは大きくなる。一方センスアンプのほうはそのような制約は SRAM のセルアレイの中身よりも小さいためセンスアンプイネーブル信号は比較的太くしやすい。したがって接続されているゲート容量や配線のドライブ力を調整することにより、ワードラインに対してセンスアンプイネーブル信号のほうが高速で信号が伝搬されるような構成となることが多い。このような場合は図示するとおり最もドライバから遠い側が、もっともワードラインアクセスと信号線アクセスの時間差が短いので、こちら側に検出用の列を持ってくればよく、ここまで仮定してきた check column の位置とも整合する。また、逆にセンスアンプイネーブルのほうが信号伝搬が遅いような構成になる場合には Check column を最もドライバに近い側に用意する必要があるといえる。

7.3.4　Check Column のセルの構成

SRAM は非常に画一的なセルが並んでいるため、密度、パターンが最適化しやすく、DFM に適した設計がされている。このため、製造時に SRAM 部分のみほかの場所よりも微細なプロセスを用いて作成されることがあるほどであるため、プロセスバラツキの影響を受けやすい。また、SRAM は各トランジスタの形状・不純物濃度などが非常にうまく設計されていなければ、読み出しができなかったり、書き込みのできないセルとなってしまうため、プロセスバラツキの影響は非常に甚大である。以上のことから、製造プロセス時のローディング効果などの影響の不均衡を抑制するためにレイアウト的にも配慮する必要がある。近年の一般的なメモリセルのレイアウトではゲートの方向を均一にするなどして、できる限りプロセスばらつきの影響を小さくしている（[57] 図 7.13）。また、大規模なメモリではワードラインの分割などが行われることもあり、このモニタリングコラムがセルアレイの端以外の部分に挿入されることも考えられる。このため、ゲートや不純物拡散領域などデバイスの活性層に関しては可能な限り通常の SRAM メモリセルを踏襲できるような構成とした。

7.3.5　関連研究

読み出し時のタイミングエラー監視　読み出しの方法としてビットラインのセンスアンプをシングルエンドのセンスアンプ 2 個として、その排他的論理和をとる手法がある。この方法ではビットラインの値が定まるまで排他的論理和は動作しないため、正しく値が取り込まれたかどうかがわかる。本提案手法で考慮した点をすべて満たしているかどうかを統一的に監視でき、各ビットを独立してすべて調べることが可能である多め余分なマージンを必要としない、などの点で非常に優れた方法であるといえる。この関連研究では、書き込み時のタイミングエラーについては考慮されていない。なお、本研究で仮定したようなラッチコンパレータ型の作動増幅回路 [51] [52] [54] [53] を用いる場合にはダイナミック回路的な正帰還がかかっているため、センスアンプが動作すると必ずでどちらかの値に固定されてしまうため、このような手法は適用できない。

書き込み時のタイミングエラー監視　SRAM のセンスアンプを二重化し、遅れて動作をするシャドウセンスアンプを追加することでレジスタ読み出し時のタイミングエラーを検出する方法がある [55]。本手法は、ここまで提案手法の類似関連研究として取り上げてきた Razor のシャドーラッチと同様の考え方に基づくものであるといえる。また、汎用プロセッサなどが有するレジスタファイルなどへの書き込みの補償として書き込み補償バッファと呼ばれるものが提案されている [56]。これは、本来書き込みたい対象となる記憶回路と並列して、書き込み保証バッファと呼ばれるより小容量なメモリを用意しておき、両者から書き込まれたデータを読みだして比較を行う手法である。この手法では、各アクセスごとに比較が行われるため、カナリア方式を適用した場合のようにメインとなる記憶回路側に余分なタイミングマージンを設定する必要がない。しかしながら、性能の低下を避けるためには書き込み保証バッファの容量はある程度大きく設定される必要があるため、ハードウェアオーバーヘッドの大きい手法であるといえる。

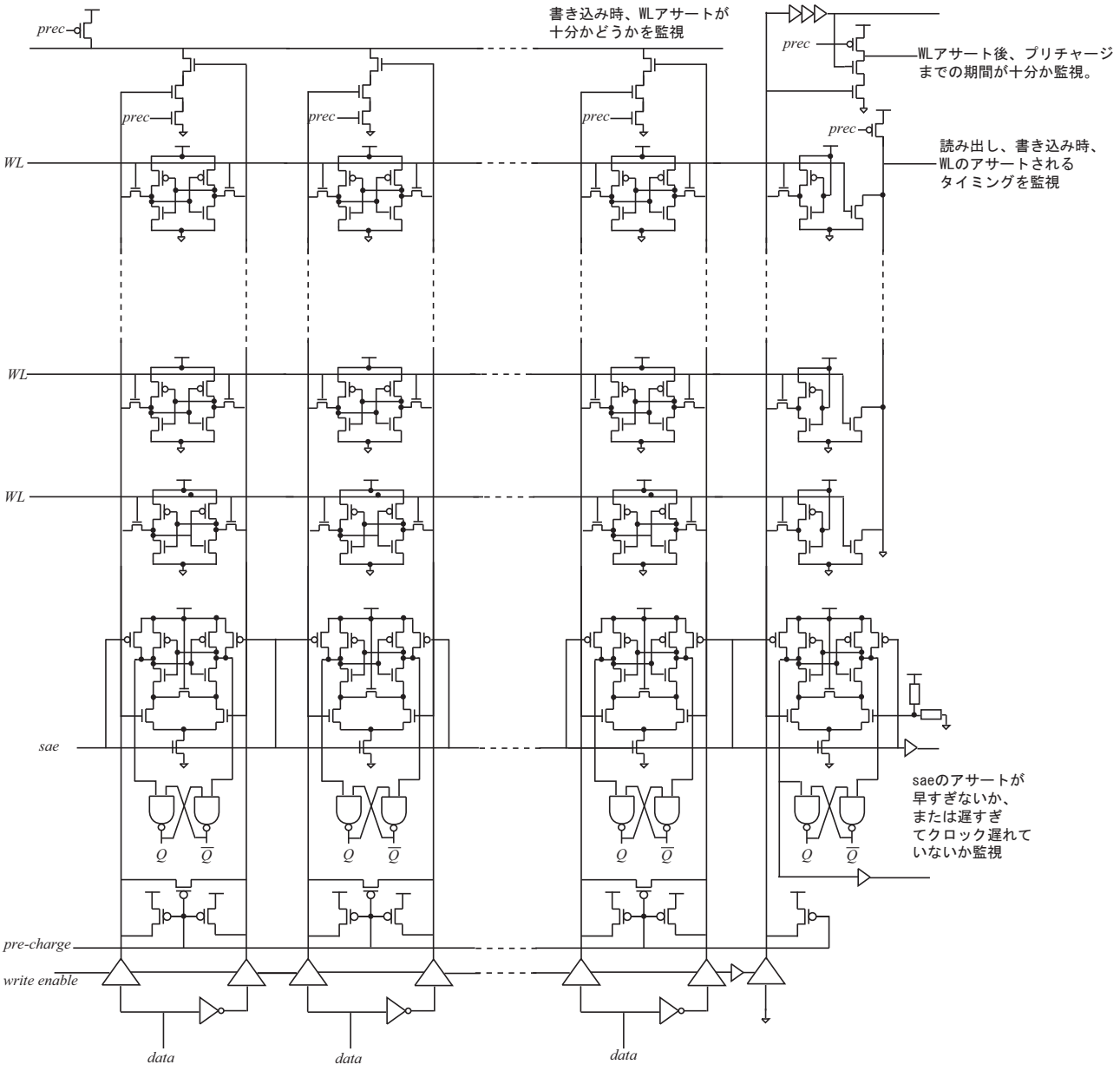


図 7.12: メモリセルアレイの読み出し書き込み保護機構

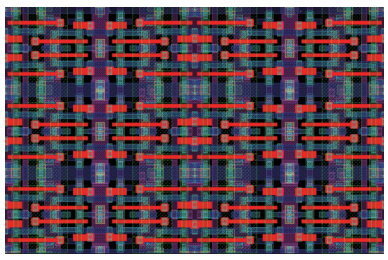


図 7.13: SRAM セルレイアウトの例。図中の赤い部分がゲートパタン。プロセスばらつきを抑えるために、非常に画一的に配置されていることが読み取れる。

第III部

考察・結言など

第8章

考察・展望・結言

8.1　全体を通しての考察

遅延補償フリップフロップでは、フリップフロップのデータの取り込みタイミングを、データ信号の遷移を直接監視しながら適切なタイミングで行うようにしている。この点で、主に冗長化した構成要素を実装し、その実行結果を比較することで誤りを見つける、従来一般的であった動的なタイミングエラー監視技術とは一線を画しているといえる。また、このような冗長化した実装を用いてエラーを見つけることの本質としては、冗長化された 2 つのユニットのタイミングマージンが互いに異なっており、より脆弱なほうが先にエラーを起こすため実行結果に違いが起こることになる。このため、冗長化した両者が同時にエラーを発生させないようにするために、十分なだけの脆弱性を持たせる必要があるということにもなる。ここでいう脆弱性とは、たとえばカナリアフリップフロップの最長遅延パス制約の狭小化であったり、Razor の最短遅延パス制約の狭小化であったりのことをさす。したがって、設計制約はある部分を堅牢にするために導入された脆弱性にかえって足を引っ張られることになりかねない。これに対して、遅延補償フリップフロップではこの脆弱な部分が動的に変更されるため、このような問題の一定のレベルを緩和しているといえる。

省電力特性については、フリップフロップ単体での動作に必要となる消費電力は、同電圧で動作させた場合カナリアフリップフロップや通常のフリップフロップの中で最大となった。追加回路があるため、通常のフリップフロップよりも消費電力が大きくなってしまうことに関しては当然と言えるが、カナリアフリップフロップよりも消費電力大きい理由としては追加回路のうちクロックに直結した部分が多いことがあげられる。たとえば、フリップフロップはビットごとに挿入されるため、基本的には 64 ビットのパスには 64 個、32 ビットのパスなら 32 個というように挿入される。しかしながら、毎サイクル必ずすべてのビットが演算に使用されるわけではなく、また必ずしもマイサイクルそのパスのデータが処理され変化するとは限らない。このため、フリップフロップに入力されている信号線のうち、クロックは通常毎サイクル動作するが、データはそれに比べると低い頻度でしか変化しない。したがって、クロックに追加回路が直結している遅延補償フリップフロップでは毎サイクルその部分でダイナミック電力が消費され、他のものよりも電力の消費量が大きくなってしまったことが考えられる。なお、クロックゲーティングを適用した場合の遅延補償フリップフロップは、消費電力を大幅に抑えることに成功したことも、この考察の裏付けであるといえる。

ソフトエラー耐性に関しては、組み合わせ回路の SET に対して非常に高い信頼性を持つことが確認できた。本手法をタイミングエラー監視として用いる場合には、タイミングクリティカルな部分だけに用いればよいが、ソフトエラー耐性を持たせるためには回路のありとあらゆる組み合わせ論理回路の後段のフリップフロップに用いなければいけないため、通常のフリップフロップと比べてハードウェア量や消費電力のオーバーヘッドが大きい本手法ではその影響が顕著に表れる可能性がある。それでも、TMR のような方法に比べれば一般的にそのオーバーヘッドは少なく、また、他の部分の設計に対してほとんど影響がないという利点を有すると考えられる。

8.2　今後の展望

消費電力に関する考察の部分で、遅延補償フリップフロップの素子数が大きいということに触れたが、これはすなわち面積のオーバーヘッドも大きいということになる。フリップフロップは VLSI 中で非常に大量に用いられる素子であるため、ほんの少しのオーバーヘッドでもチップ全体で見れば非常に大きなものになる。Razor に関する文献 [8] によると、適用するべきであるタイミングクリティカルなパスの後方にあるフリップフロップは全体の 9.4% 程度であるということが言われている。遅延補償フリップフロップでも、それ自体の電力・面積オーバーヘッドを考慮すると Razor 同様実装する場所はなるべく少ないほうが望ましいといえる。ただし、Razor やカナリアフリップフロップが最大パス遅延のみを監視しているのに対して、遅延補償フリップフロップでは最小パス遅延の監視も同時に行っている。クロックは H-tree やクロックメッシュなどのさまざまな手法を用いてなるべく回路全体に同時に供給できるように設計されているが、それでも実際にはスキューが生じるため、最短パス遅延制約を間違いなく満たせるように、マージンとして多くの遅延素子が挿入されているのが現実である。Razor などが最大パス遅延設計に対するマージンを削減するように、遅延補償フリップフロップでは、最小パス遅延設計に対するマージンも削減することができる可能性がある。このような設計時のハードウェアオーバーヘッドに関する問題では、より詳細な評価が必要となるため、最小パス遅延設計に対する定量的な評価は今後の課題としたい。

本論文では、遅延補償の考え方をメモリセルアレイに適応する方法について考察を加えた。また、実際に小規模なものではあるが提案手法を回路設計を通してシミュレーションを行い、その基本的な挙動を確認した。実際にどの程度の遅延まで対応できるか、追加回路の消費電力や面積オーバーヘッドなどはどの程度かなどの詳細で定量的な評価や、実際に多段の回路と組み合わせての特性評価などは今後の課題としたい。本論文では主に回路要素のみを個別に考察しているため、実際の本手法の有用性をより正確に判断するためにはプロセッサ全体の設計や、VLSI の施策を通じてより包括的な検証を行う必要があると考える。

このような動作時タイミングエラー監視技術が、設計時に入り込む電源揺らぎ・温度揺らぎなどのマージンを削減することができる可能性があることは、関連研究などからも明らかである。しかし、近年の集積回路技術における設計に関する問題はこのようなマージンの増大によるものだけではない。半導体電子回路の劇的な集積度の向上は利用可能な素子数を急速に増加させ、現実的な設計期間、費用、人的労力をもちいた場合にそのすべてのキャパシティを使い切ることは非常に難しい。そのため、設計時には CAD による設計支援がおこなわれ、その支援技術のさかんな研究のかいもあり設計生産性は急速に向上している。それでも実際には、利用可能なトランジスタ数に CAD の設計支援能力はどんどんと差をあけられているという現状がある。このような、CAD を利用した設計において問題となるものの一つにフォールスパス問題と呼ばれるものもある [58] [59]。フォールスパスとは回路上は確かに存在するが、絶対に活性化しないパスのことである。当然活性化しないのであるから、このパスをタイミング設計の際に考慮する必要はない。しかしながら、あるパスがフォールスパスであるかどうかを回路上のすべてのパスについて判断することは充足性判定問題（SAT）であり、その計算量は NP 完全であるとされる。SAT ソルバの性能は指数関数的に向上してはいるものの、現状では大規模な集積回路のフォールスパス問題を現実的な時間で解くには至っていない。したがって、現実的な時間内に判断ができなかった場合は、フォールスパスである可能性が高いとしても、当該パスは活性化する可能性があるとしてタイミング設計を行わなければならない。この、判定できなかったパスが実はフォールスパスで、かつ活性化すると判断された場合にタイミングクリティカルなパスであった場合には、無駄に長いパスに合わせたタイミング設計が行われることになり、動作周波数の低下など性能にも大きくかかわってくる。動作時タイミングエラー監視はこのようなフォールスパス問題を緩和できる可能性がある。つまり、フォールスかどうか判定できなかったパスが存在した場合に、投機的にフォールスパスであるとし、仮にそのパスが活性化するような場合があったとしても、その時に生じたタイミングエラーを検出し回復を行うことができる。当然ながら、投機的にフォールスパスとみなすことが可能なパスの長さには限りがあるが、フォールスパス問題を緩和する効果は大き

いと考えられる。なお、提案手法と主に比較してきた二つの手法のうち、カナリアフリップフロップでは通常よりも長いかもしれないパスが存在すると、2 つのフリップフロップが同時に誤ってしまう可能性があるため、フォールスパス問題には対応できない。一方、Razor はフォールスと仮定したパスが活性化した場合でもシャドーラッチが正しい値を保持できる範囲であれば問題ないため、フォールスパス問題の緩和に効果があると考えられる。このような、回路設計にかかわる問題に関するより詳細な考察も今後の課題としたい。

8.3　結言

半導体集積回路の指数関数的な集積度の向上と素子のスケーリングに関連して近年顕在化してきている問題として、電力密度上昇の問題、ノイズ耐性の低下の問題、プロセスばらつきの増大、あるいはそれらと関係した設計難化の問題などをとりあげ、これらの問題に統一的に対応する技術として回路動作時信号遷移監視に基づく遅延補償フリップフロップを提案した。これらの問題の中から特に、演算時の消費電力、タイミングエラー耐性、ソフトエラー耐性などの具体的なトピックについてシミュレーションなどを通じた効果の検証を行った。

提案手法はフリップフロップが正しい値を取り込むように、入力データ信号の変動を監視、データ取り込みタイミングを調整することによりタイミングエラーを回避することを基本動作とした。究極的には、フリップフロップ自体が入力されてくるデータのつなぎ目を認識し、最小遅延パスエラー・最大遅延パスエラーともに起こらないようにデータを取り込むことが理想的であるが、フリップフロップからみるとただの電圧である入力データにおいて、連続するデータ間の区切り目を正確に認識するのは難しい。本研究の手法では、データのつなぎ目が安定期間を持つなど一定の前提の下で、それを疑似的に行うことを目指したものであるということもできる。

この遅延補償フリップフロップでは、タイミング設計における自由度を既存手法よりも高く保ちつつ、タイミングエラー耐性を高めるだけでなく、クロックゲーティングや DVFS との協調によって省電力化に対して寄与することも確認した。また、論理回路のソフトエラー耐性を向上させる効果があることを確認し、非常に高い SET 耐性を実現することができることを示した。さらに、集積回路システムに実際に利用する場合に検討すべき事項に関していくつかの手法を提案し、定性的であるがその挙動について考察を行った。フォールスパス問題への適応、プロセッサ全体の中で実際に用いた場合の挙動等、今後引き続き検討していかなければならないことも多く残っているが、本論文を通じて本提案手法の利点のいくつか示すことができたと考える。

集積回路の極端な微細化は、物理的な限界を間近に控えているとされる現在でも絶え間なく研究開発が続いており、それに伴って様々な問題も表面化してくることが予想される。いわゆるハイパフォーマンスの分野では相変わらず不断の高性能化が追及されるであろうが、民生機器に搭載されるプロセッサの性能としては、多くの人がすでに大きな不満を感じなくなってきているのではないだろうか。ありとあらゆる電化製品にマイコンが組み込まれ、多くの人が当たり前に存在するある種のインフラストラクチャーとして集積回路を使用している現在、信頼性・省電力性などが第一に要求されてくる用途は確実に増えてきている。ユーザーや社会が真に依存できる “ディペンダブル” なプロセッサとはどのようなものかという問いに対する答えは、人によってさまざまなものであると思われるが、ディペンダブル VLSI の一つの要素技術の提案として行われた本研究が、その一助とまではならずとも、せめて何らかの知見を与えることに役立てば幸いに思う。

謝辞

無知浅学怠惰無能非才不忠な私に、研究者としての範をしめし、懇切丁寧なご指導を賜り、さまざまなことに挑戦させてくださいました、東京大学大学院情報理工学系研究科電子情報学専攻坂井修一教授に深淵なる謝意を表し、心よりお礼申しげます。東京大学大学院情報理工学系研究科電子情報学専攻五島正裕准教授には、動作時エラー検出による集積回路動作の動的調整という、本研究を行うきっかけとなる考えをご紹介いただき、また、実践的なご指導をいただきましたことに関しまして、深く感謝し心よりお礼申し上あげます。また、同じ部屋で研究生活を送らせていただいた科学技術振興機構研究員の入江英嗣博士には、研究生活において種々のアドバイスをいただきました。心より感謝しお礼申し上げます。

東京大学大学院情報理工学系研究科電子情報学専攻坂井・五島研究室の学生の皆様には、研究生活において様々な面でお世話になりました。特に、一林宏憲氏、塩谷亮太氏、渡辺憲一氏、亘理靖展氏、喜多貴信氏には、プロセッサシミュレータの使用について、杉本健氏、金大雄氏には、プロセッサコアのハードウェア記述に関して、多大なるご助力をいただきました。また、新領域創成科学研究科荻野健氏にはパブリックドメインのプロセッサシミュレーションツールについて技術的なアドバイスをいただきました。また、勝沼聡氏は研究室内で同じ仕事を分担していたため、自分が忙しい時などは、いつも仕事を代わりに行ってくださいました。ここに謝意を表し心より感謝申し上げます。

東京大学大学院新領域創成科学研究科基盤情報学専攻柴田直教授、同伊藤潔人博士（現、日立中央研究所）、同鈴木康文博士（現、日立中央研究所）、東京大学大学院工学系研究科電子情報学専攻山崎英男博士（現、東芝）、大阪大学大学院歯学研究科顎顔面強制学教室助手八木雅和博士には、集積回路の設計・理論・測定に関する様々な事柄に関しまして、多くのご指導をいただきました。また、仁木祐介氏、藤田和英氏、グェンタンリム氏、萬澤康雄氏には、自身の研究でご多忙の折にも関わらず、多くの集積回路設計・シミュレーションに関する実践的なアドバイスをいただきました。このような多くの具体的なアドバイスがなければ、私の研究は評価環境の構築からして頓挫していた可能性すらありました。心より感謝しお礼申しあげます。特に仁木氏、藤田氏には、もと同じ研究室の同期ということもあり、公私を問わずさまざまな面でお世話になったばかりか、自分でも時間をかければ調べられるような素人質問を遠慮もなくぶつけたにもかかわらず、それに対していつも温かく的確な情報を与えていただきました。お礼の言葉もありません。

また、本学で研究生活を送る上で以下の多くの方のご指導をいただき、多岐にわたるご助力・アドバイスをいただきました。特に、東京大学大学院工学系研究科電気工学専攻、藤田博之教授、三田吉郎准教授、高橋一浩氏、中田宗樹氏、伊藤浩太氏（現トヨタ中央研究所）、濱口洋平氏、今井義明氏、東京大学大学院工学系研究科電子工学専攻大津元一教授、浅田邦博教授、保立和夫教授、中野義明教授、川添忠准教授、種村拓夫講師、中根了昌助教、肥後昭男助教、杉浦邦晃博士（現東芝）、原田智之氏、清水俊匡氏、出浦桃子氏、新山太郎氏、東京大学大学院工学系研究科総合研究機構杉山正和准教授、久保田雅則助教、中村義雄技術員、東京大学大学院新領域創成科学研究科基盤情報学専攻修士課程卒業生シーサムラーヌサックダー氏、トンプラシットベンジャマース氏、東京大学工学部電気系工作室技術専門職員渋谷武夫氏、東京大学工学部電気系通信実験室岸眞人助手、東京大学生産技術研究所試作工場技術専門職員滑川敏夫氏、東京大学工学部電子工学科白石文高氏（現富士通）には、研究室の垣根を越えて、親身に接していただきました。とくに、三田准教授・原田氏は元同じ研究チームのメンバーということで、公私ともに多大なるご厚情を賜りました。さらに、文部科学省ナノテクノロジーネットワーク産学官連携研究員澤村智紀氏、東京大学大

学院理学系研究科物理学専攻山本智教授、芝祥一氏、東洋大学工学部バイオ・ナノエレクトロニクス研究センター花尻達郎准教授、株式会社キャノンマーケティングジャパン佐藤隆吾氏、株式会社 ADVANTEST 森村氏、日笠氏、株式会社バンパートナーズ植木武美氏、光洋サーモシステム株式会社保坂弘樹氏、株式会社 MARUWA 高橋浩道氏、大成理科工業株式会社軍司哲也氏、メルク株式会社川俣康弥博士、横川電機株式会社蒲原敦彦氏、スタンレー電気株式会社谷雅直氏駿河精機株式会社北和門氏、藤田元喜氏、林良子氏、株式会社ルネサステクノロジ、サンゴバン株式会社、ＪＳＲ株式会社の皆様には、広い意味でのシリコン半導体集積回路研究に関する幅広い知見を与えていただきました。ここに謝意を表し心よりお礼申し上げます。

また、八木原晴水氏、月村美和氏、内田杏氏、伊世知代氏、稲垣素子氏、森公子氏、大野栄氏、池谷幸絵氏には、研究に関する出張の際や、備品購入などの事務処理に関しまして大変お世話になりました。秘書、事務員の方々のおかげで非常に快適かつ迅速な研究生活が遅れましたこと、お礼申し上げます。

本研究に関する発表の際には、株式会社東京大学ＴＬＯ小森啓安氏、アイテック国際特許事務所伊神広行弁理士、小屋迫利恵弁理士に大変お世話になりました。ここに感謝申し上げます。

さらに、田中所長はじめ株式会社ヒタチ町田営業所の皆様、セールス課長谷川係長、岡本様はじめ、株式会社山崎製パン杉並工場の皆様、町田社長、石井マネージャはじめ有限会社フォンターナ・ラフォンテ森下の皆様にはよりプライベートな面から、大学生活・研究生活を送る上で多大なるご支援をいただきました。

なお、有意義な大学生活を送り、本研究に従事するにあたって、親類・友人・知人から、言い尽くせないほどのご支援を賜りました。公私を問わず私の大学生活・研究生活を支え、有益なものにしてくださいました多くの皆様に心より感謝し、お礼申し上げます。

なお、本研究におけるチップ試作は、東京大学大規模集積システム設計教育研究センターを通じ、ローム株式会社、凸版印刷株式会社、日本ケイデンス・デザイン・システムズ株式会社、シノプシス株式会社及びメンターグラフィックス株式会社の協力で行われたものである。また、本研究の一部は科学技術振興機構戦略的創造研究推進事業 CREST プロジェクト “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” の支援によるものである。

参考文献

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits," Electronics, vol. 38, no. 8, Apr. 19 1965, pp. 82-85.

[2] R. H. Dennard, F. H. Gaensslen, H. N. Yu, V. L. Rideout, E. Bassous, and A. R. Leblanc, "Design of ion-implanted MOSFETs with very small physical dimensions," IEEE Journal of Solid-State Circuits SC-9, 1974, pp. 256-268.

[3] G. E. Moore, "No Exponential is Forever: But "Forever" Can Be Delayed!," International Solid-State Circuits Conf., Vol. 1, Feb. 2003, pp. 20-23.

[4] S. Borkar, "Circuit techniques for subthreshold leakage avoidance, control and tolerance," International Electron Dvices Meeting 2004 (IEDM 2004), Dec. 2004, pp. 421-424.

[5] X. Tang, V. K. De, and J. D. Meindl, "Intrinsic MOSFET Parameter Fluctuations Due to Random Dopant Placement," IEEE Transactions on VLSI Systems, Vol. 5, Issue 4, Dec. 2004, pp. 369-376.

[6] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter Variations and Impact on Circuits and Microarchitecture," Design Automation Conference (DAC 2003), Jun. 2003, pp 338-342.

[7] T. Karnik, S. Borkar, and V. De, "Sub-90nm Technologies – Challenges and Opportunities for CAD," International Confference on Computer Aided Design 2002, pp 203-206.

[8] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," International Symposium on Microarchitecture, 2003. pp 7-18.

[9] S. Das, S. Pant, D. Roberts, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "A Self-Tuning DVS Processor Using Delay-Error Detection and Correction," Symposium on VLSI Circuits, 2005 pp. 258-261.

[10] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A Self-Tuning DVS Processor Using Delay-Error Detection and Correction," IEEE Journal of Solid-State Circuits, Vol. 41, Issue 4, Apr. 2006 pp. 792-804.

[11] T. Liu and S. L. Lu, "Performance Improvement with Circuit-Level Speculation," International Symposium on Microarchitecture, 2000. pp. 348-355.

[12] S. L. Lu, "Speeding up Processing with Approximation Circuits," IEEE Computer, Vol. 37, No. 3, Mar. 2004, pp. 67-73.

[13] N. R. Shanbhag, "Reliable and Efficient System-on-Chip Design," IEEE Computer, Vol. 37, No. 3, Mar. 2004, pp. 42-50.

[14] A. K. Uht, "Going beyond Worst-Case Specs with TEAtime," IEEE Computer, Vol. 37, No. 3, Mar. 2004, pp. 51-56.

[15] S. Mtra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust System Design with Built-in Soft-Error Resilience," IEEE Computer, Vol. 38, No. 2, Feb. 2005, pp. 43-52.

[16] M. Miller, K. Janik, and S. L. Lu, "Non-Stalling Counterflow Microarchitecture," International Symposium on High Performance Computer Architecture(HPCA-4), Feb. 1998, pp. 334-341.

[17] R. F. Sproull, I. E. Sutherland, and C. E. Molnar, "Counterflow Pipeline Processor Architecture,," Sun Microsystems Laboratories Inc. Technical Report, SMLI-TR-94-25, Apr. 1994.

[18] T. Sato and Y. Kunitake, "A Simple Flip-Flop Circuit for Typical-Case Designs for DFM," International Symposium on Quality Electronic Design, pp. 539-544.

[19] Troutman. R. R, "VLSI Limitations from Drain-Induced Barrier Lowering," IEEE Transactions on Electron Devices, Vol. 26, Issue 4, Apr. 1979, pp. 461-469.

[20] Troutman. R. R, "VLSI Limitations from Drain-Induced Barrier Lowering," IEEE Transactions on Electron Devices, Vol. 14, Issue 2, Apr. 1979, pp. 383-391.

[21] T. Kuroda, T. Fujita, S. Mita, T. Nagamatu, S. Yohioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9V 150MHz 10mW 4mm^2 2-D Discrete Cosine Transform Core Processor with Variable-Threshold Voltage Scheme," IEEE Journal of Solid-State Circuits, Vol.31, No.11, Nov. 1996, pp. 1770-1779.

[22] N. H. E. Weste and D. Harris, "CMOS VLSI Design -A Circuits and Systems Perspective-," Addkison Wesley, 2005.

[23] 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 Journal of Solid-State Circuits, Vol.33, No.3, Mar. 1998, pp. 463-472.

[24] A. Tiwari, S. R. Sarangi, and F. Torrellas, "ReCycle: Pipeline Adaptation to Tolerate Process Variation," International Symposium on Computer Architecture (ISCA), Jun. 2007, pp. 323-334.

[25] X. Liang and D. Brooks, "Mitigating the Impact of Process Variations on Processor Register Files and Execution Units," International Symposium on Microarchitecture, Dec. 2006, pp. 504-514.

[26] Y. Ye, S. Borkar, and V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," International Synposium of VLSI Circuits 1998, pp. 40-41.

[27] G. Gerosa, S. Gary, C. Dietz, D. Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, T. Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kahle, "A 2.2 W, 80 MHz Superscalar RISC Microprocessor," IEEE Jounal of Solid-State Circuits, Vol. 29, Issue 12, Dec. 1994, pp. 1440-1454.

[28] B. B. M. W. Badalawa and M. Fujishima, "60 GHz CMOS Pulse Generator," Electronics Letters, vol. 43, no. 2, Jan. 2007, pp. 100-102.

[29] K. Watanabe, H. Ichibayashi, M. Goshima, and S. Sakai, "Design of Processor Simulator "Onikiri" ," Synposium on Advanced Computing Systems and Infrastructures (SACSIS), May 2007, pp. 194-195.

[30] T. Austin, E.Larson, and D. Ernst, "SimpleScalar: An Infrastracture for Computer System Modeling," IEEE Computer, Vol.35, No.2, Feb. 2002, pp. 59-67.

[31] D. Brooks, V. Tiwari, and M.Martonosi, "wattch: A Framework for Architectural-Level Power Analysis and Optimizations," International Symposium on Computer Architecture, Jun. 2000, pp. 83-94.

[32] R. E. Kessler, "The Alpha 21264 Microprocessor," IEEE Micro, Vol. 19, Issue 2, Mar. 1999, pp. 24-36

[33] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Early Design Exploration," IEEE Transactions on Electron Devices, Vol. 53, No. 11, Nov. 2006, pp. 2816-2823.

[34] V. Joshi, R. R. Rao, D. Blaauw, and D. Sylvester, "Logic SER Reduction through Flipflop Redesign," International Symposium on Quality Electronic Design, Mar. 2006.

[35] N. Seifert and N. Tam, "Timing Vulnerability Factors of Sequentials," IEEE Transactions on Device and Materials Reliability, Vol. 4, No. 3, Sep. 2004, pp. 516-522.

[36] T. M. Austin, "DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design," International Symposium on Microarchitecture, Dec. 1999, pp. 196-207

[37] Weaver. C and T. Austin, "A Fault Tolerant Approach to Microprocessor Design," International Conference on Dependable Systemas and Networks (DSN), Jun. 2001, pp. 411-420.

[38] E. Rotenberg, "AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessor," Symposium on Fault-Tolerant Computing 1991, pp. 84-91.

[39] H. Yamashita, H. Yasuura, F. N. Eko, and C. Yun, "Variable Size Analysis and Validation of Computational Quality," International High-Level Design Validation and Test Workshop, Nov. 2000, pp. 95-100.

[40] L. D. Hung, M. Takada, Y. Ge, and S. Sakai, "A Cost-effective Technique to Mitigate Soft Error in Logic Circuits," IEICE Technical Report, Vol. 104, No. 477, Dec. 2004, pp. 31-36.

[41] P. Hazucha and C. Svensson, "Impact of CMOS Technology Scaling on the Atmospheric Nertron Soft Error Rate," IEEE Transactions on Nuclear Science, Vol. 47, No. 6, De. 2000, pp. 2586-2594.

[42] J. W. Kellington, R. Mcbeth, P. Sanda, and R. N. Kalla, "IBM®POWER6™Processor Soft Error Tolerance Analysis Using Proton Irradiation," Wrokshop on Silicon Errors in Logic (SELSE 3), Apr. 2007,

[43] T. Calin, M. Nicolaidis, and R. Velazco, "Upset hardened memory design for submicron CMOS technology," Transactions on Nuclear Science, Vol. 43, Issue 6, Dec., 1996, pp. 2874-2878.

[44] P. Hazucha, T. Karnik, S. Walstra, B. A. Bloechel, J. W. Tschanz, J. Maiz, K. Soumyanath, G. E. Dermer, S. Narendra, V. De, and S. Borkar, "Measurements and Analysis of SER-Tolerant Latch in a 90-nm Dual-V_T CMOS Process," Custom Integrated Circuits Conference, Sep. 2003, pp. 617-620.

[45] N. Seifert, B. Gill, V. Zia, M. Zhang, and V. Ambrose, "On the Scalability of Redundancy based SER Mitigation Schemes," International Conference on Integrated Circuit Design and Technology, May. 2007, pp. 1-9.

[46] Y. Arima, T. Yamashita, Y. Komatsu, T. Fujimoto, and K. Ishibashi, "Cosmic-Ray Immune Latch Circuit for 90nm Technology and Beyond," International Solid-State Circuits Conference, Feb. 2004, pp. 492-493.

[47] M. Nicolaidis, "Carry Checking / Parity Prediction Adders and ALUs," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 11, No. 1, Feb. 2003, pp. 121-128.

[48] M. Zhang, S. Mitra, T. M. Mak, N. Seifert, N. J. Wang, Q. Shi, K. S. Kim, N.R. Shanbhag, and S. J. Patel, "Sequential Element Design With Built-In Soft Error Resilience," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14. 2006, pp. 1368-1378.

[49] K. Sugimoto, H. Irie, M. Goshima, and S. Sakai, "Implementation of An Out-of-Order Superscalar Processor on FPGAs," (in Japanese) Symposium on Advanced Computing Systems and Infrastructures (SACSIS), May 2007, pp. 196-197.

[50] H. Nambu, K. Kanetani, K. Yamasaki, K. Higeta, M. Usami, Y. Fujimura, K. Ando, T. Kusunoki, K. Yamaguchi, and N. Homma, "A 1.8-ns Access, 550-MHz, 4.5-Mb CMOS SRAM," IEEE Journal of Solid-State Circuits, Vol. 33, No. 11, Nov. 1998, pp. 1650-1658.

[51] M. Matsui, H. Hara, Y. Uetani, L. S. Kim, K. Matsuda, and T. Sakurai, "A 200 MHz 13 mm^2 2-D DCT Macrocell Using Sense-Ampligying Pipeline Flip-Flop Scheme" IEEE Journal of Solid-State Circuits, vol. 29, No. 12, Dec. 1994, pp. 1482-1490.

[52] P. Gronowski, W. Bowhill, R. Preston, M. Gowan, and R. Allmon, "High-performance microprocessor design," IEEE journal of Solid-State Circuits, Vol. 33, No. 5, May 1998, pp. 676-686.

[53] F. Klass, C. Amir, A. Das, K. Aingaran, C. Truong, R. Wang, A. Mehta, R. Heald, and G. Yee, "A new family of semidynamic and dynamic flip-flops with embedded logic for high-performance processors," IEEE Journal of Solid-State Circuits, Vol. 34, No. 5, May 1999, pp. 712-716.

[54] J. Montanaro, R. T. Witek, K.Anne, A. J. Black, E. M. Cooper, D. W. Sobberpuhi, P. M. Donahue, J. Eno, G. W. Hoeppuner, D. Kruckemyuer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stephany, and S. C. Thierauf, "A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor," IEEE Journal of Solid-State Circuits, Vol. 38, No. 11, Nov. 1996, pp. 1703-1714.

[55] E. Karl, D. Sylvester, and D. Blaauw, "Timing Error Correction Techniques for Voltage-Scalable On-Chip Memories," International Symposium on Circuits and Systems, Vol. 4, May 2005, pp. 3563-3566.

[56] H. Irie, K. Sugimoto, M. Goshima, and S. Sakai, "Preventing Timing Errors on Register Writes: Mechanisms of Detections and Recoveries," International Workshop on Advanced Low Power Systems (ALPS), Jun. 2007, pp. 31-38.

[57] K. J. Kim, J. M. Youn, S. B. Kim. J. H. Kim, S. H. Hwang, K. T. Kim, and Y. S. Shin, "A Novel 6.4μm^2 Full-CMOS SRAM Cell with Aspect Ratio of 0.63 in a High-Performance 0.25μm-Generation CMOS Technology," Symposium on VLSI Technology, 1998, pp. 68-69.

[58] D. H. C. Du, S. H. C. Yen, and S. Ghanta, "On The General False Path Problem in Timing Analysis," International Conference on Design Automation, Jun. 1989, pp. 555-560.

[59] H. Lee, S. Heo, and J. Kim, "Global False Path-Aware Hierachical Timing Analysis," Annual Conference of IEEE Industrial Electronics Society, Vol. 2, Nov. 2004, pp. 1963-1965.

著者発表文献

主著論文

▲ 査読付論文誌

- "Simultaneous Vertical and Horizontal Self-Patterning Mehotd on Deep Three-Dimensional Micro Structures,"
  Kenichiro Hirose, Fumitaka Shiraishi, and Yoshio Mita,
  IoP Journal of Micromechanics and Microengineering, Vol. 17, 2007, pp. S68-S76.
- "Delay-Compensation Flip-Flop with In-Situ Error Monitoring for Low-Power and Timing-Error-Tolerant Circuit Design,"
  Kenichiro Hirose, Yasuo Manzawa, Masahiro Goshima, and Shuichi Sakai,
  IPAP Japanese Journal of Applied Physics,
  (to be published in Apr. 2008).
- "Polarization-Transmissive Photovoltaic Film Device Consisting of An Si Photodiode Wire-Grid"
  Kenichiro Hirose, Yoshio Mita, Yoshiaki Imai, Freédéric Marty, Tarik Bourouina, Kunihiro Asada, Shuichi Sakai, Tadashi Kawazoe, and Motoichi Ohtsu
  IoP Journal of Optics A: Pure and Applied Optics,
  (to be publishe in 2008 summer)

▲ 査読付国際学会

- "Deep-Trench Vertical Si Photodiode towards Active-Device Integrated OMEMS,"
  Kenichiro Hirose, Yoshio Mita, Masanori Kubota, and Tadashi Shibata,
  IEEE/LEOS International Conference on Optical MEMS (OMEMS 2006), at Big Sky, Montana, USA, Aug. 2006, pp. 191–192.
- "Self-Patterning Metal Deposition on Deep Theree-Dimensional Micro Structures for Vertically Buried Inductors,"
  Kenichiro Hirose, Yoshio Mita, and Tadashi Shibata,
  MicroMechanics Europe (MME 2006), at Southampton U.K., Sep. 2006, pp. 85–88.
- "Polarization-Transmissive Thin-Film Solar Cell with Photodiode Nanowires,"
  Kenichiro Hirose, Yoshio Mita, and Shuichi Sakai,
  IEEE/LEOS International Conference on Optical MEMS and Nanophotonics (OMEMS 2007), at 花蓮, 台湾, Aug. 2007, pp. 29–30.

共著論文

▲ 査読付論文誌

- “Deep-Trench Vertical Si Photodiodes for Improved Dfficiency and Crosstalk,”
  Yoshio Mita, Kenichiro Hirose, Masanori Kubota, and Tadashi Shibata,
  IEEE Jounal of Selected Topics in Quantum Electronics, Vol. 3, Issue 2, 2007, pp. 386–391.

▲ 査読付国際学会

- “Surface Corrugated p-n Junction on Deep Submicron Trenches for Polarization Detection with Improved Efficiency”
  Yoshiaki Imai, Yoshio Mita, Kenichiro Hirose, Masanori Kubota, and Tadashi Shibata, Asia-Pacific Conference on Transducers and Micro-Nano Technology (APCOT2008),
  (submitted in Jan. 2008).

▲ シンポジウム・その他

- “Towards Intelligent 3-D Micro Electro Mechanical Devices by Deep RIE Technology,”
  Yoshio Mita, Kenichiro Hirose, Masanori Kubota, Tomoyuki Harada, and Tadashi Shibata,
  COE Symposium on Advanced Electronics for Future Generations –”Secure-Life Electronics” for Quality Life and Society–, at Tokyo, pp.159–165, Jan. 2007.
- “Intégration d’un dispositif électronique actif dans une micro-structure tri-dimensionnel,”
  Yoshio Mita, Kenichiro Hirose, Masanori Kubota, and Tadashi Shibata,
  Journée scientifique francophone 2006, at Tokyo, pp. 31, Dec. 2006.
- “MEMS Integrated Nano and Flexible Electron Dvices by Deep Reactive Ion Etching Technology,”
  Yoshio Mita, Sakda Srisomrun, Yohei Hamaguchi, Yoshiaki Imai, Kenichiro Hirose, Masanori Kubota, Tomoki Sawamura, Jean-Bernard Pourciel, Freédéric Marty, Tarik Bourouina, Shuichi Sakai, and Tadashi Shibata,
  International Symposium on Secure-Life Electronics -Advanced Electronics for Quality life and Society-, Mar. 2008.
  (to appear in Mar. 2008)

特許

- 集積回路装置
  坂井修一、廣瀬健一郎、萬澤康雄、五島正裕
  （出願中）

受賞・その他

- MicroMechanics Europe (MME 2006), Best Poster Aword.
  “Self-Patterning Metal Deposition on Deep Theree-Dimensional Micro Structures for Vertically Buried Inductors,”
- 財団法人　電気・電子情報学術振興財団
  平成 19 年度猪瀬学術奨励賞 (2007)

- 第 3 回東京大学電気系学科優秀卒業論文賞、2006 年 3 月
  “トレンチ型不純物領域を有する高効率垂直フォトダイオード”

、

付録 A

遅延補償フリップフロップの作製

本研究の一環として、実際に遅延補償フリップフロップの作製を試みた。1 つは、セミカスタムのゲートアレイを用いて、エッジ検出回路を作製・測定を行った。また、ローム株式会社提供の 0.18μm テクノロジのカスタム LSI 施策サービスを用いて、遅延補償フリップフロップや、その関連回路を作製した。

A.1　0.18 μm カスタム LSI の試作

遅延補償フリップフロップの実際の挙動を確認する目的で、カスタム LSI チップの試作を行った。使用したプロセスは、株式会社ローム提供の 0.18 μm 5-metal CMOS テクノロジを利用し、試作チップの面積は 2.5 $\times$ 2.5 mm^2 である (図 A.3 A.4)。

作製した回路レイアウトを用いて比較を行ったところ、通常のフリップフロップでは約 60μm^2 程度なのに対し、遅延補償フリップフロップの専有面積は約 180μm^2 とおよそ 3 倍の面積オーバーヘッドがあることがわかった。なお、このときに用いた回路レイアウトを図、A.1 A.2 に示す。これらのレイアウトは、設計した回路を単純にレイアウトに起こしただけのものであり、面積の最適化、性能の最適化のために、回路レイアウトの際に考慮すべき事項は、ごく基本的な事柄を除き考慮していないことを付記する。

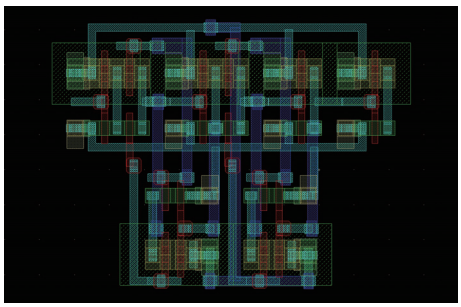


図 A.1: 通常のフリップフロップ一個分のレイアウト

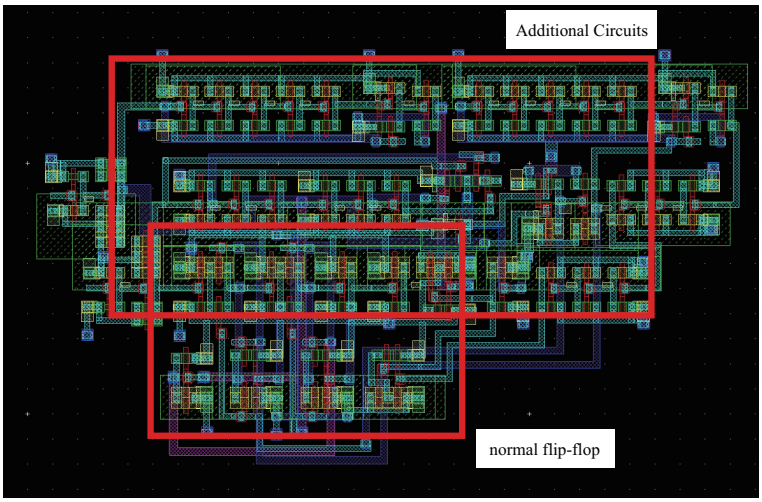


図 A.2: 遅延補償フリップフロップ一個分のレイアウト

A.2　3 μm セミカスタムゲートアレイによる、fast prottyping

3 μm のレガシープロセスを用いて、セミカスタムのゲートアレイチップを実際に作製しその評価を行った。実験室で製造できるレベルの非常に古いデザインを利用したため、ゲート遅延などは非常に大きいも

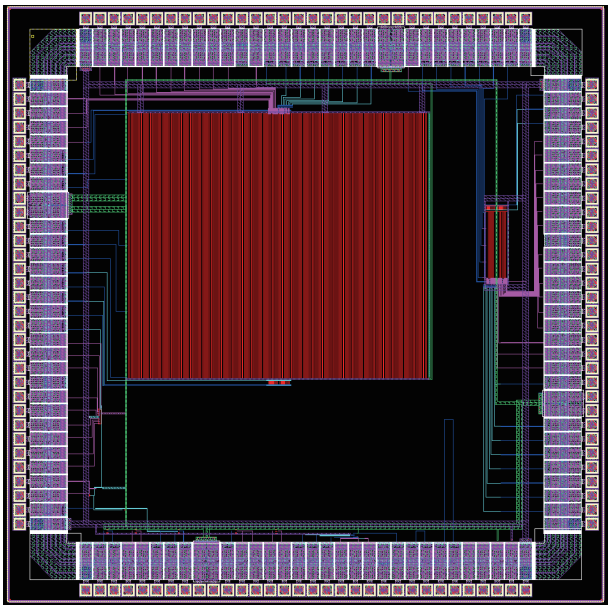


図 A.3: 試作ップの全体レイアウト。なお、遅延補償フリップ単体のテスト回路は、下辺の入出力パッドの直近に見える非常に小さい部分である。

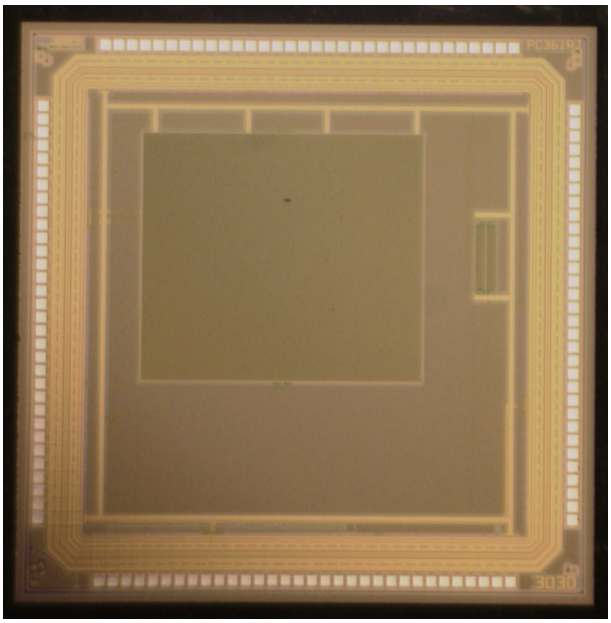


図 A.4: 製造後ベアチップのチップ写真

のとなっているが、同じくゲートアレイのチップ上に作製していた Fanout-of-4 インバータ遅延で正規化を行った場合には、最先端プロセスと同様の挙動していることが確かめられた。

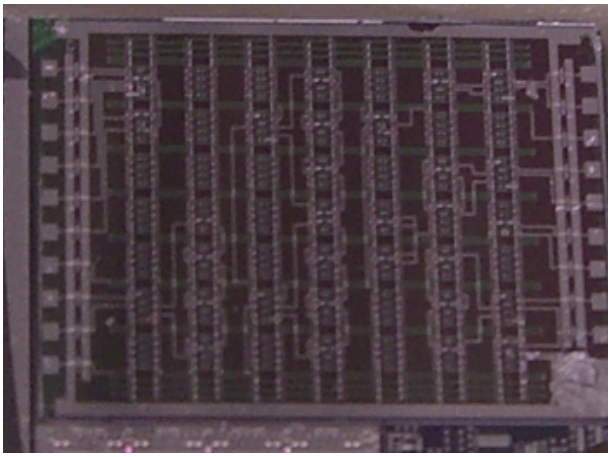


図 A.5: 実際に作成したゲートアレイのチップ写真

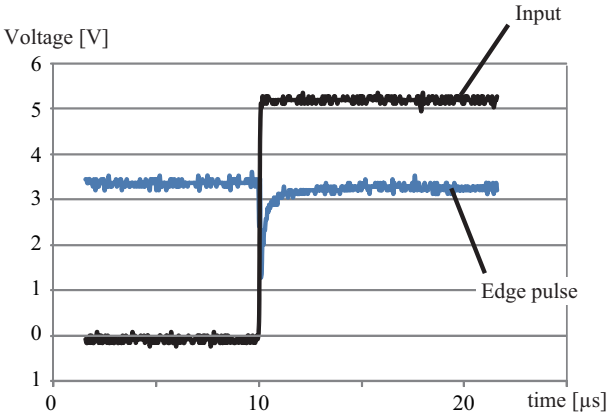


図 A.6: Gate Array チップで作成した信号遷移検出回路の実測波形

付録B

遅延補償フリップフロップの変形例

B.1　クロックの立ち上がりよりもデータの変動の開始が遅い場合に関して

本文中では、遅延補償フリップフロップを構成する論理回路が動く範囲で最も短い幅のパルスを信号遷移検出回路が発生するように設計したものでシミュレーションを行っているが、信号遷移検出回路の生成パルス幅をより大きなものを用いたり、

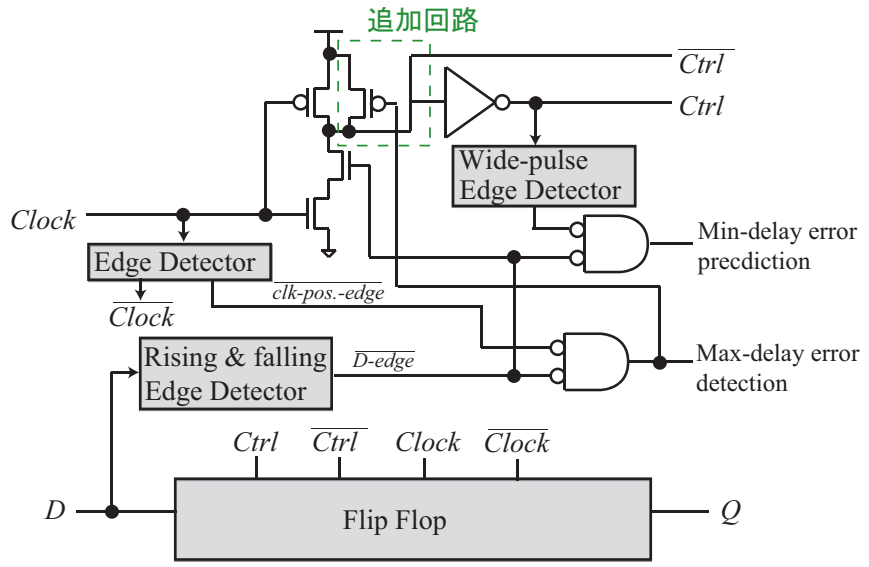


図 B.1: ゲートが立ち上がっても、一定期間内であればデータを取り直すことができる機構。この場合は、およそクロックのエッジパルスの幅文の遅れを救済する。しかしながら、この方法ではそのまま最小遅延パスが到達してもよいタイミングを後ろにずらさなければならないため、Clock Borrowing などによる手法と、タイミング設計的には基本的に同様になる。

図 B.1 のような追加回路をもつような遅延補償フリップフロップを利用することで、より広範囲なタイミングでの信号入力に対するエラー耐性を持たせることも可能となる。しかし、このように救済できる最大の遅延を後ろにずらすことは、当然ながら、最短パス遅延設計制約を狭小化するものである。このため、このような追加は、何らかの方法で Clock Borrowing を用いた場合と、タイミング設計的には大きな差異はないということになる。

B.2　遅延補償フリップフロップのデータ監視を利用したクロックゲーティング

また、遅延補償フリップフロップではデータの遷移を常にモニターしているため、それを利用して自らが細粒度なクロックゲーティングを行うことが可能となる。。つまり、各々のフリップフロップが各々のクロックでデータを更新する必要があるかどうかをわずかな追加回路で判断し、必要なときのみフリップフロップにクロックを供給できるようになる。また、汎用プロセッサのデータパスのような場所は、クロックの信号遷移を容易に予測できるが、集積回路全体として見たときには、制御信号線上のフリップフロップであったりといった容易にデータ入力の変更の有無が予測できないような箇所もある。このような場所

においても遅延補償フリップフロップは自ら信号を監視してゲーティングを行うため、フリップフロップにつながっているトランジスタの充放電による消費電力を抑えることができる。

なお、この方法では遅延補償ができないような範囲の過大遅延が発生した際に、metastable な状態が発生する可能性があることに注意しなければならない。このため、従来のようなエラー復旧手法を用いることは難しいため、遅延が大きくなり始めたら速やかに電源を昇圧するなどの対策が必要となる。しかしながら、データパス以外の部分においては、一度エラーが起こってしまうと命令の再実行などによって簡単に復旧処理を行うことは難しい場合も多く、このような部分ではショートパスエラー信号が立つまで遅延が大きくなりすぎることを許すような設計はそもそも非現実的であるため、この点は大きな問題とはならないと考えられる。

したがって、システマティックなクロックゲーティングや、最小遅延エラー発生時の処理の再実行などが容易な部分では、ここまで取り上げてきた遅延補償フリップフロップをもちい、それ以外の場合には、図 B.2 のような回路をもちいて自らクロックゲーティングを行うのが効率的であると考えられる。なお、これらのクロックゲーティングは、十分に入力データの変化頻度が低いときにより効果を発揮することになる。

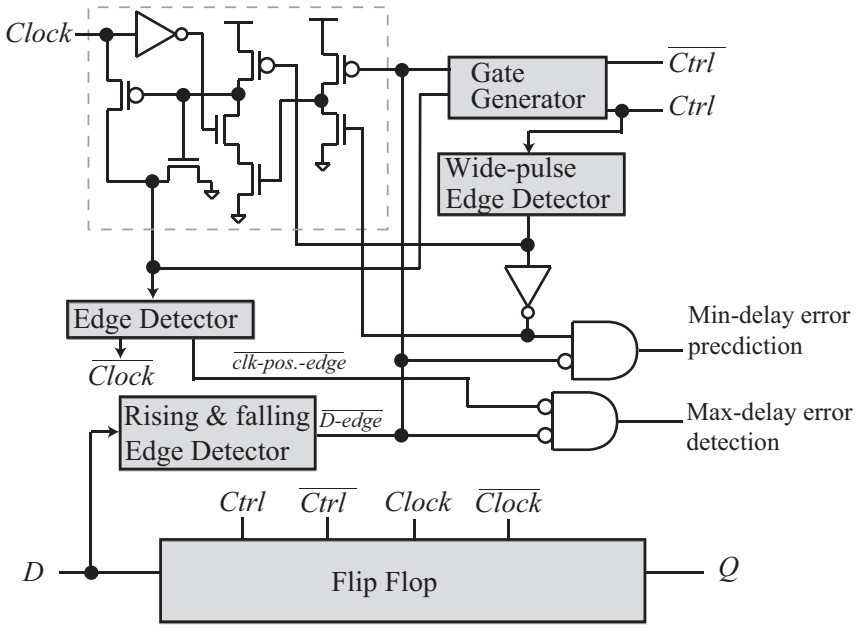


図 B.2: 遅延補償フリップフロップの細粒度クロックゲーティングへの適用