

修士論文

ダイナミック・タイム・ボローイングを可能に
するクロッキング方式のSRAMへの適用

Application of Clocking Scheme Enabling Dynamic Timing Borrowing
on SRAM

平成27年02月05日提出

指導教員

坂井 修一 教授

東京大学大学院 情報理工学系研究科
電子情報学専攻

48-136449 神保 潮

概要

近年，半導体プロセスの微細化に伴ってチップ上の素子遅延のばらつきが増加しており，ワースト・ケース設計では性能が向上しなくなる恐れがある。そこで我々の研究室では，ワースト・ケースではなくティピカル・ケースに基づく動作を可能にするために，タイミング・フォールト検出と二相ラッチによるタイム・ボローリングを組み合わせた，動的タイム・ボローリングを可能にするクロッキング方式を提案している。本論文では，レジスタ・ファイルやキャッシュを構成するSRAMへの本手法の適用について検討・提案を行う。SRAMにおけるプリチャージ動作がタイミング・フォールトをマスクしてしまうため，タイミング・フォールト検出を単純には適用できない。本論文の提案手法は，SRAMの評価結果に応じて，プリチャージの有無を制御することによって適用を可能にする。また従来のSRAMのワード・ライン切り替えでは，タイム・ボローリングを許容した設計でTF検出の正しさを保証することは難しい。提案する手法では，TF検出期間に入った命令のワード・ラインを検出期間中は保持し，次サイクルの命令のワード・ラインと同時にアクセスすることで，TF検出の正しさを保証する。提案手法を適用したレジスタ・ファイルをトランジスタ・レベルで設計し，SPICEシミュレーション上でTFの発生率を評価した。

目次

1 はじめに	5
2 動的タイム・ボローイングを可能にするクロッキング方式	9
2.1 タイミング・ダイアグラム	10
2.2 TF 検出: Razor の構成とタイミング制約	11
2.3 二相ラッチによる静的タイム・ボローイング	13
2.4 動的タイム・ボローイングを可能にするクロッキング方式	15
3 SRAMへの適用の問題点	18
3.1 SRAM の構成と動作	18
3.2 動的タイム・ボローイング適用時の課題	20
4 提案手法	25
4.1 TF 検出適用のための提案	25
4.1.1 動作手順と完全性	25
4.1.2 回路構成	27
4.1.3 最大遅延制約の緩和	30
4.2 タイム・ボローイングのための提案	31
4.2.1 動作	31
5 評価	33
5.1 評価環境	33
5.2 結果	34
5.3 考察	36
6 おわりに	39
7 関連研究	41
7.1 ばらつき耐性を狙った回路/アーキテクチャレベル技術	41
7.1.1 ReCycle	41
7.1.2 TIMBER	41
7.1.3 Bubble Razor	43
7.2 SRAMへのRazor適用の既存手法	45

7.2.1	ダブルエンドビットラインを対象としたメモリのTF検出手法	45
7.2.2	シャドウセンスアンプ	46
参考文献		47

表 目 次

1	評価に用いたソフトウェア環境	33
---	----------------	----

図 目 次

1	LSI の典型値と最悪値の向上幅の乖離	6
2	タイミング・フォールト検出・回復と DVFS の組み合わせ	7
3	単相 FF のタイミング・ダイアグラム (t-diagram)	9
4	Razor の回路構成	12
5	単相 FF (左) と Razor (右) の t-diagram	12
6	単相 FF(左) と二相ラッチ (右) の t-diagram	13
7	静的タイム・ボローイング	14
8	動的タイム・ボローイングのための回路構成	15
9	二相ラッチ方式 (左) と動的タイム・ボローイングを可能にするクロッキング方式 (右) の t-diagram	16
10	SRAM の構成	19
11	Razor のダイナミック・ロジックへの適用の問題 : (左) SRAM のセンス・アンプ周辺回路, (右) 左図のタイミング・チャート	21
12	既存の Razor の SRAM への適用	22
13	TF 検出漏れ	23
14	(上) SRAM のセンス・アンプ周辺回路, (下) TF 検出機構の導入	27
15	提案手法の回路実装	29
16	SRAM の t-diagram : (左) 提案手法適用前, (右) 提案手法適用後	30
17	タイム・ボローイングのための提案回路構成	31
18	SPICE シミュレーション用回路	34
19	SPICE シミュレーションによる TF 検出動作の検証. サイクルタイム:250[ps], 温度:55[°C].	35

20	サイクル・タイムごとの TF, TF 検出誤検知, TF 検出漏れの発生率. 遅延累積あり	36
21	サイクル・タイムごとの TF, TF 検出誤検知, TF 検出漏れの発生率. 遅延累積なし	37
22	単相 FF (左) と ReCycle (右) の t-diagram	42
23	TIMBER の回路構成と t-diagram	43
24	TIMBER (左) と提案手法 (右) の t-diagram	44
25	Bubble Razor の t-diagram	45
26	Bubble Razor (左) と提案手法 (右) の t-diagram	46

1 はじめに

近年の LSI の性能向上は半導体プロセスの微細化によって支えられてきた。その一方で、トランジスタや配線の大きさが原子の大きさに近づくに従って、LSI 素子遅延のばらつきが大きな問題となりつつある [1][2]。

従来の LSI の設計と製造は、ワースト・ケースに基づいて行われていた。すなわち、回路遅延の変動要因である製造ばらつき (Process), 電源電圧 (Voltage), 温度 (Temperature) の 3 条件に関する許容範囲を設定し、範囲内での動作を保証するというものである。特に定められた条件のうち、ワースト・ケースにおける見積もられた遅延の下でも正しく動作するように設計を行う。これが満たされない場合、遅延の動的な変化により設計者の意図とは異なる動作が引き起こされる過渡故障であるタイミング・フォールト (Timing Fault : TF) が発生する。ワースト・ケース設計では、想定した動作条件内のワースト・ケースにおいて TF が発生しないように設計していることになる。

LSI 素子遅延のばらつきが増大していくと、従来の最悪値に基づいた設計手法は悲観的になりすぎる。この様子を図 1 に示す。図 1(上) は二つの異なるプロセス技術を元に製造されたトランジスタの遅延分布を表している。赤の線が 1 世代前のプロセス、青の線が微細化したプロセスに基づいたトランジスタの分布である。微細化に伴い、ばらつきが増大するため、青線の分布は赤線の分布に比べて裾野が広がっている。このとき、微細化が進むにつれて遅延の典型値が向上する一方、ばらつきの増大により最悪値は典型値ほど向上しないことが分かる。

したがって、最悪値に基づいた設計では LSI の動作速度が向上しなくなる恐れがある。図 1(下) はこのような傾向が続いた場合のプロセス技術の進化に基づく LSI 素子速度の典型値と最悪値を表している。プロセス技術が進むことで典型値の向上が見込めても、ばらつきの増大によって、最悪値と典型値の乖離が広がってしまい、性能向上が見込めなくなる恐れがある。

この問題に対処するために、ワースト・ケースの遅延ではなく実際の遅延に基づいた設計手法が数多く提案されている。

設計段階において遅延の見積もりを統計的に扱う SSTA (Statistic Static Timing Analysis : 統計的静的タイミング解析) [3][4] はその一例である。この手法では、あるステージの遅延をそれぞれの LSI 素子の遅延のワースト値の和として見積るのではなく、データパス上の LSI 素子が一定の確率分布にしたがって異なる遅延をとったときのデータパスの遅延の最悪値を統計的に見積る。パスを構成するト

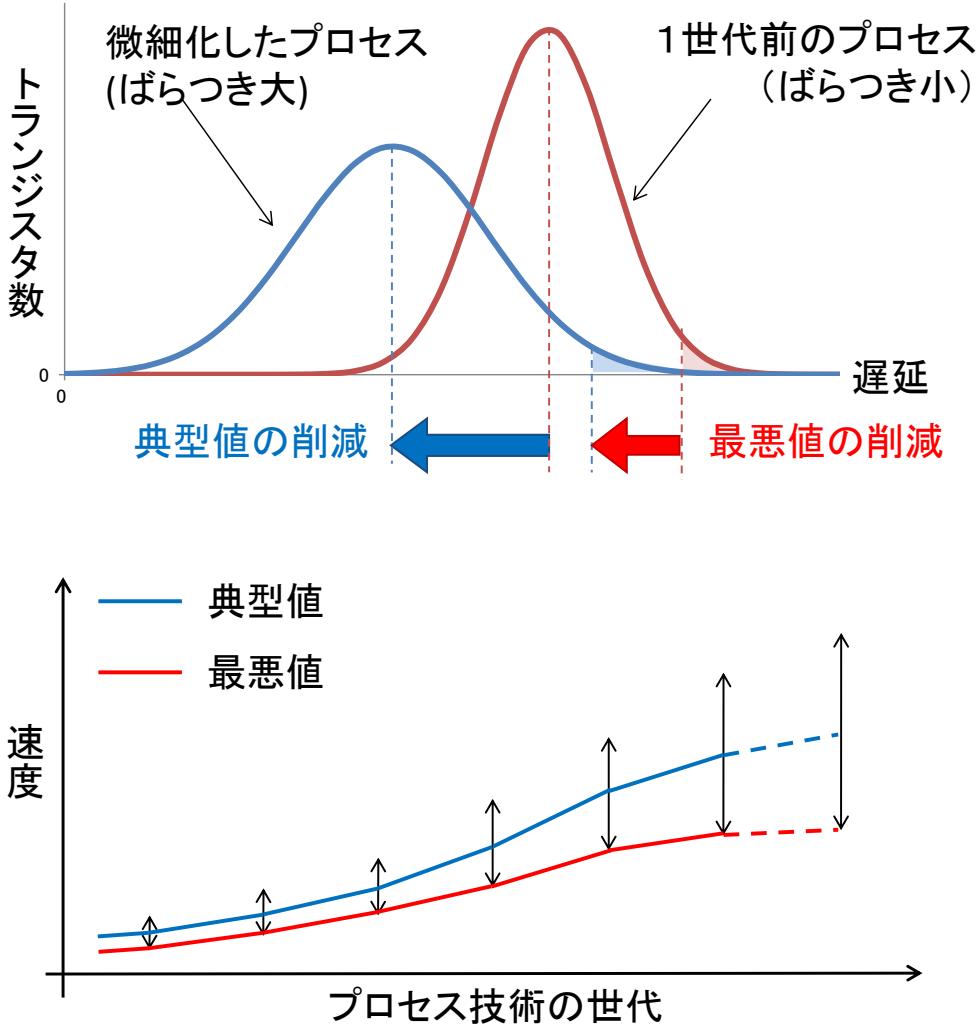


図 1: LSI の典型値と最悪値の向上幅の乖離

トランジスタ、配線の遅延がすべて最悪であるような確率は高くないため、ワースト・ケースほど悲観的ではない遅延見積もりを行うことができる。また、レイアウト後に、チップ内の温度分布や電源電圧分布を精密にシミュレートする手法や、製造後に実際の遅延を測定して、クロック・スキューレーを調整する手法も提案されている。これらの手法では、歩留りの向上が見込めるほか、見積もりではない、個々のチップの出来（実際のゲート幅やゲート長の長さ）に基づいた最高動作周波数や最低電圧での動作が可能となる。一方で動作条件の変動は考慮されていない上に、設計コストが大幅に増加するというデメリットが存在する。

チップごとの固有の環境に応じた動作を可能にするためには、チップ環境のモ

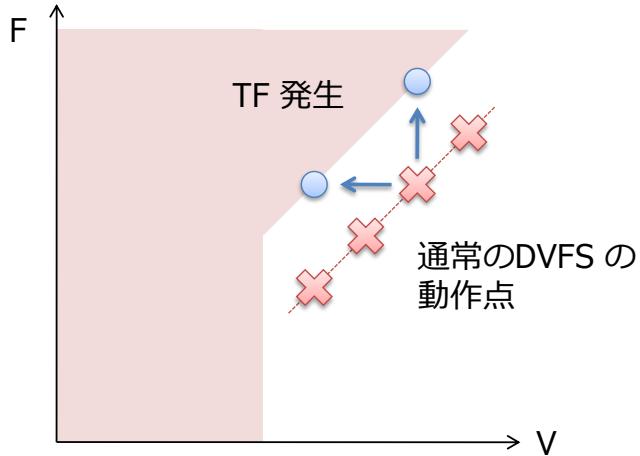


図 2: タイミング・フォールト検出・回復と DVFS の組み合わせ

ニタリングが不可欠である。チップ内温度やプロセッサコアの計算量に応じてクロック周波数を変化させるターボブーストテクノロジーはその一例である。

この類の手法の中で最も直接的であるのは TF の発生そのものをモニタリングする手法群 [5][6][7][8][9][10] である。

TF の発生を予期する手法として, Canary FF が提案されている [5][6]. この手法では予備的回路の TF を検出し, 主回路中の TF を起こさない条件下での動作を狙っている。Canary は, 予備的回路と主回路中の遅延が大きすぎると TF の予測の外れる可能性が高くなる一方で, 小さすぎると主回路中の TF も同時に起きてしまう可能性が増えるため, 遅延の設定に議論の余地があった。

そこで, TF の検出・回復機構を持つことで, 主回路中の TF の発生を許容する手法が提案された。2.2 章で述べる **Razor** [7][8][9] は, その代表例である。このような手法と DVFS (Dynamic Voltage and Frequency Scaling) [11] を組み合わせると, 以下のように, 見積もりではない, 実際の遅延に応じた動作を実現することができる [12].

図 2 にその様子を示す。図 2 中 \times 印はワースト・ケースで定められた DVFS の V (Voltage : 電源電圧) と F (Frequency : 動作周波数) の組を表している。ワースト・ケース設計では, このように, TF が発生しないよう十分なマージンを取つて V - F が設定される。TF 検出・回復を行う手法では, ここより V を下げる, または, F を上げることができる。図 2 中 \circ 印で表される検出直前の V - F が, 見積もりではない, そのチップのその時の動作環境における実際の遅延に応じた V - F である。このようにすれば, ワースト・ケース設計で必要であったマージンを劇的に

削減することができる。

我々は、より効果的なクロック周波数向上や電圧削減を可能にする手法として、動的にステージ間のタイム・ボローイングを可能にする手法を提案した[13]。この手法では、二相ラッチとTF検出を組み合わせることで、ステージ間の遅延の融通が行われる。これによって、たとえあるステージでサイクル・タイムより遅延の大きいパスが活性化されたとしても、その超過分を次のステージに持ち越す。次のステージにおいて遅延の小さいパスが活性化されれば、この超過分が相殺され、TFの発生を抑えることができる。

一方で、本クロッキング方式をSRAMへ適用する手法については考慮されていなかった。SRAMの遅延は配線遅延が多くを占めており、これはスケーリングによって減少することができないため、その相対的な遅延が増大している。したがって、SRAMで構成されるのアクセスが、回路全体におけるクリティカルな遅延をもつことは避けられない。SRAMはレジスタ・ファイルやキャッシュを構成に用いられるため、これへの本クロッキング方式の適用は不可欠であるといえる。

本論文では、動的タイム・ボローイングを可能にするクロッキング方式のSRAMへの適用について考察を行う。動的タイム・ボローイングを可能にするクロッキング方式の構成要素であるTF検出と二相ラッチについてそれぞれSRAMへの適用にあたって克服すべき課題をまとめ、適用手法を提案する。また、提案手法を適用したレジスタ・ファイルをトランジスタ・レベルで設計し、SPICEシミュレーション上でTFとその誤検知、検出漏れに関して評価を行った。

本論文における以降の構成は以下の通りである。第2章では、我々が提案した**タイミング・ダイアグラム**と呼ぶ図を導入し、動的タイム・ボローイングを可能にするクロッキング方式について説明する。第3章では、SRAMへの動的タイム・ボローイングを可能にするクロッキング方式の適用に関しての概観を述べ、TF検出の適用と二相ラッチ化への対応のそれぞれの課題点を述べる。第4章ではそれに関する提案手法を詳述する。第5章では提案手法の評価について述べ、第6章で本論文をまとめる。

2 動的タイム・ボローイングを可能にするクロッキング方式

本章では、我々の研究室で提案した動的タイム・ボローイングを可能にするクロッキング方式を紹介する。本方式はTF検出機構と二相ラッチ化から成り立っているので、それぞれについて説明を行った後に述べる。

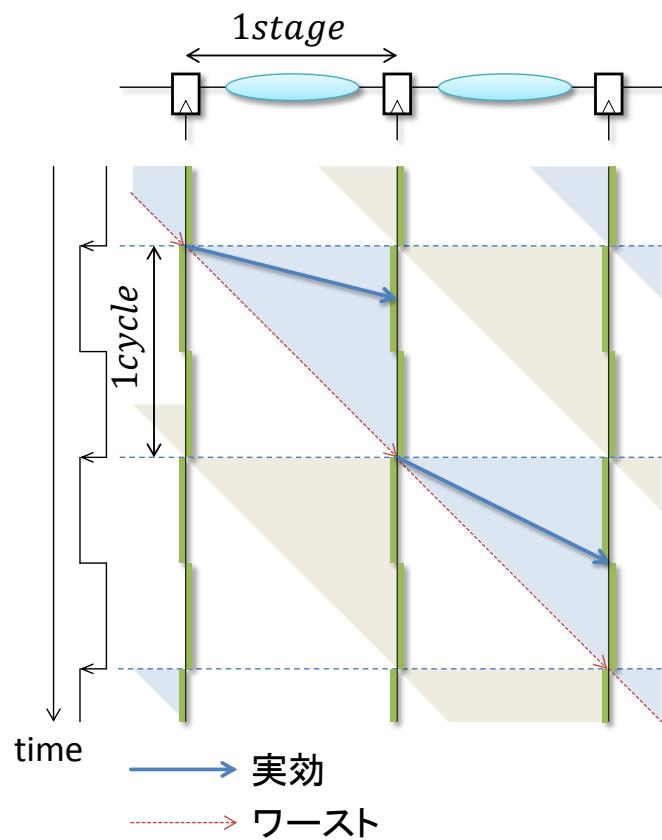


図 3: 単相 FF のタイミング・ダイアグラム (t-diagram)

2.1 タイミング・ダイアグラム

図3のような図を我々はタイミング・ダイアグラム (**t-diagram**) と呼んでいる。通常のタイミング・チャートが論理値-時間の次元を持つに対して、t-diagramは時間-空間の次元を持つ。

タイミング・チャートは、論理値の時間的変化を表現するが、1本の波形で表すことができるるのは回路の特定の1点の振る舞いに限られる。複数の点にまたがる動きを把握するためには、複数の波形を並べなければならない。

それに対して t-diagram は、下方向が時間を、右方向が回路中を信号が伝わって行く方向を表し、時間の経過につれて信号が伝わっていく様子を俯瞰することができる。

ロジック中のあるパスを通った信号によってロジックの出力が変化したとき、その信号によってそのパスが活性化したと言う。t-diagramでは、最後にパスを活性化した信号の伝達を実線矢印で表す。実際のロジックではパスが無数に存在するため、ロジック上の全遅延の存在領域は、ロジック内の最小遅延のパスとクリティカル・パスに囲まれた領域に網掛けすることにより示す。

クロッキング方式の表現 次に、図3でのクロッキング方式の表現を説明する。

同図はマスタースレーブ構造を持つFFを念頭に描かれている。同図において、FFの下にある実線はラッチが閉じている状態を、実線と実線の間の空白は、ラッチが開いている (transparent) 状態を、それぞれ表している。信号の矢印が実線にぶつかった場合、ラッチが開くまで信号は下流側に伝わらない。エッジ・トリガ動作は、マスタースレーブラッチを互い違いに記述することで生じる隙間から信号が「漏れる」様子で直感的に表すことができる。

パイプライン動作を行う際には、FFと次のFFに挟まれたロジックがパイプライン・ステージとなり、各クロック・サイクルごとに各ステージが並列に動作を行うことになる。

一連の処理（例えば、パイプライン型プロセッサにおける1つの命令の処理）は、あるサイクルにおいてあるステージで処理された後、次のサイクルにおいて次のステージの処理へと次々引き継がれていく。この一連の処理のことをあるフェーズの処理と呼ぶ。t-diagramでは、あるフェーズの処理と次のフェーズの処理を、矢印が存在し得る領域の網掛けの色を分けることで区別している。

なお t-diagram では、各ステージのクリティカル・パスに対応する直線矢印の角

度を 45° としている。こうすることによって、各ステージの遅延は、t-diagram 上のステージの横幅によって表現することができる。

クロッキング方式の要諦は、あるフェーズの信号が前後のフェーズの信号と「混ざる」ことがないように分離した上で、処理を次のサイクルに次のステージへと引き継いでいくことである。t-diagram 上では、以下の 2 つの条件が満たされていればよい。

1. 実践矢印をたどって、次のサイクルに次のステージへと至ることができる。
2. 矢印が存在し得る範囲を表す網掛けの領域が、前後のフェーズのそれと重ならない。

クロッキング方式のタイミング制約は、この条件から導かれる。

単相 FF 方式が上記の条件を満たして正しく動作するためには、各ステージにおいて、あるクロック・エッジで入力側の FF の出力が変化してから、次のクロック・エッジまでに出力側の FF の入力に必ず信号が到着しなければならない。すなわち、サイクル・タイムを τ とすると、各ステージのロジックのクリティカル・パスの遅延が τ 未満であればよいということになる。このことを、最大遅延制約は $1\tau/1$ ステージと表現することとする。図 3 では、クリティカル・パスの遅延を表す赤い 45° の線がちょうど次のクロック・エッジに到着しており、最大遅延制約の限界を達成した場合を表している。

2.2 TF 検出: Razor の構成とタイミング制約

図 4 (左) に、Razor FF の回路構成を示す [7]。RazorFF は、通常の FF (Main FF) と、Shadow Latch によって構成される。Shadow Latch には、Main FF へのそれより位相の遅れたクロックが供給されており、Main FF と Shadow Latch で 2 回、信号のサンプリングを行う。それらの値を比較して、異なっていれば *error* が出力され、TF と判定される。

同図 (右) は RazorFF への入力 *d* の遷移が Main FF へのクロック・エッジよりも遅れてしまった場合を表している。Main FF のサンプリングでは 0 を得るが、Shadow Latch のクロック・エッジには間に合うため、Shadow Latch のサンプリングでは 1 を得る。よって両者は異なっているから、*error* が出力されて TF が検出される。

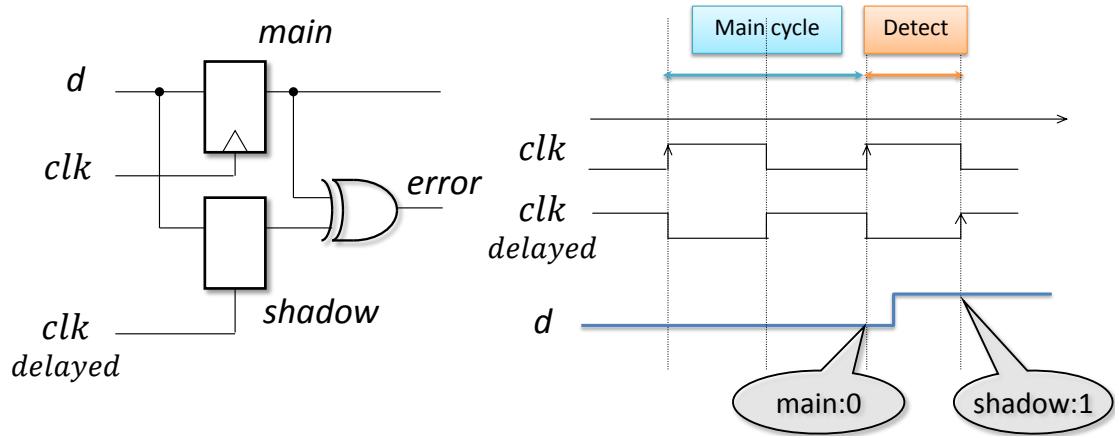


図 4: Razor の回路構成

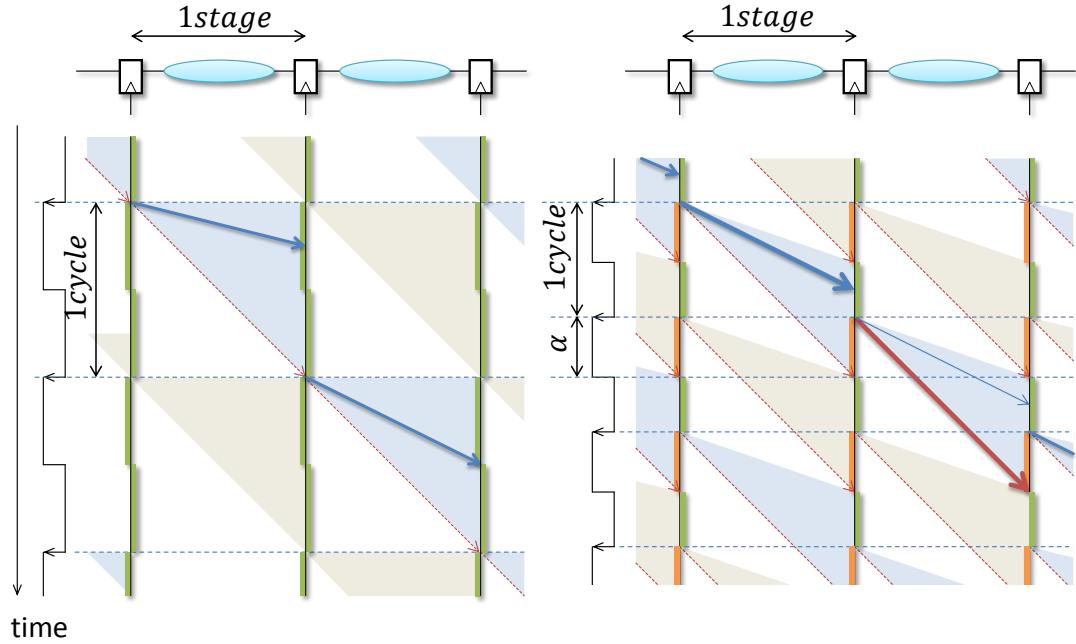


図 5: 単相 FF (左) と Razor (右) の t-diagram

図 5 は TF 検出機構を用いない単相 FF 方式と Razor の t-diagram を比較したものである。同図(右)における Razor では、Main FF から半周期遅れたクロックを Shadow Latch に供給している。t-diagram 上における FF の下の橙色の実線は、TF の検出ウィンドウを表している。つまり、検出ウィンドウの上端で Main FF が、下端で Shadow Latch がサンプリングを行い、その値を比較する。CP の遅延に対応

する 45° の赤線が検出ウィンドウの下端までに到着すれば、ワースト・ケースにおいても TF として処理することができる。したがって、サイクル・タイムに対する検出ウィンドウの割合を α とすると、最大遅延制約は $(1 + \alpha)\tau/1$ ステージとなり、単相 FF 方式より $\alpha\tau$ だけ改善される。

なお、Razor FF における Shadow Latch が、次サイクルの信号をサンプリングしてしまわないように、最小遅延制約を設けなければならない。

2.3 二相ラッチによる静的タイム・ボローイング

図6右が、二相ラッチの t-diagram である。二相ラッチは、FF を構成する 2 つのラッチ（マスター、スレーブ）のうちの 1 つを、ロジックのちょうど中間に移動したものと理解することができる。単相 FF 方式の 1 ステージに相当するロジックをラッチが二分する形になる。

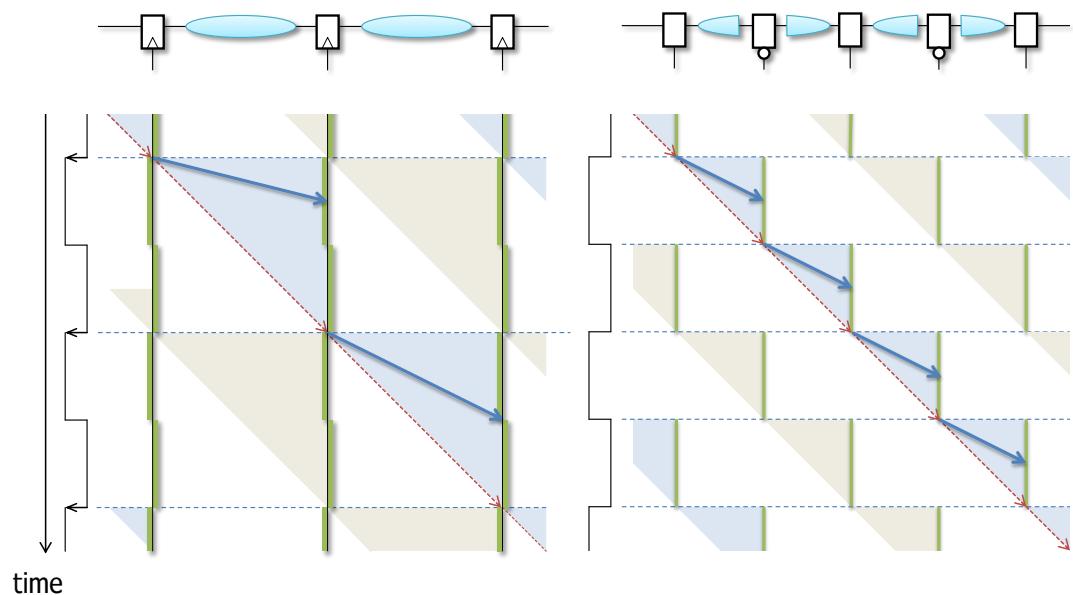


図 6: 単相 FF(左) と二相ラッチ (右) の t-diagram

静的タイム・ボローイング 図7はステージ間の遅延に偏りがある場合の单相 FF 方式 (左) と二相ラッチ方式 (右) の t-diagram である。

单相 FF 方式では常にラッチが閉じている状態のため、信号が次のステージに伝播するタイミングがクロックの立ち上がる瞬間に限定される。すなわち、仮にク

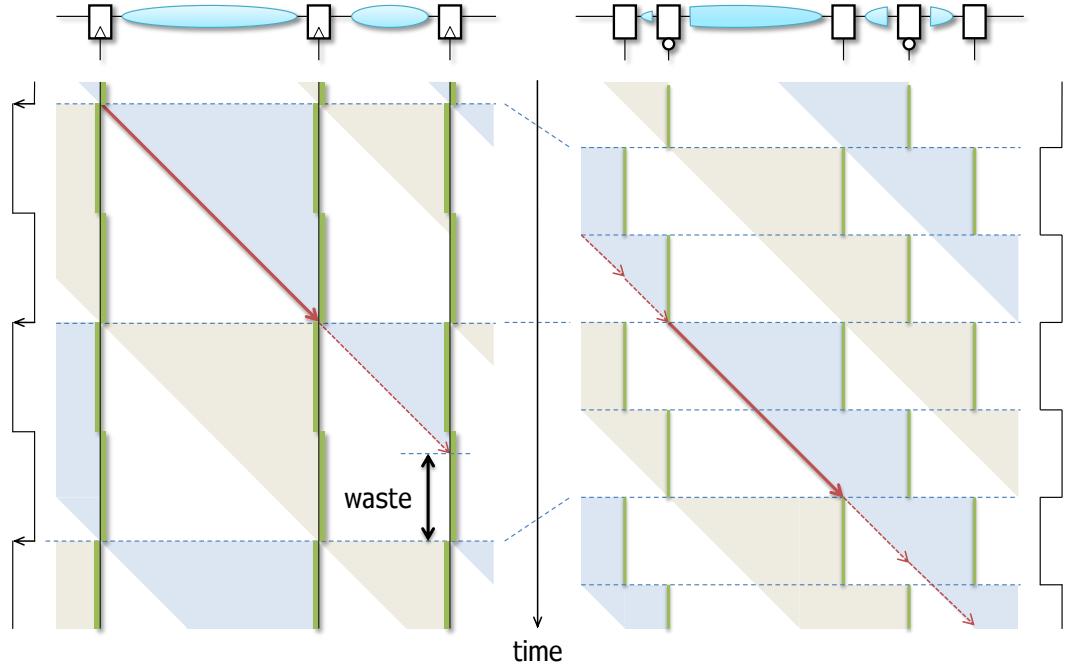


図 7: 静的タイム・ボローイング

ロックの立ち上がりより前に信号が到達していても次のステージに伝播されない。单相 FF 方式では遅延の最も大きいステージの最大遅延によってサイクル・タイムが定まるため、遅延の小さいステージではサイクル・タイムに無駄が生じてしまう。

二相ラッチ方式では、单相 FF 方式の 1 ステージに相当するロジックが 2 分されており、ロジックを通過する時間をステージ間で融通することができ、その結果サイクル・タイムが短縮できる。このように、前後のステージ間で時間を融通する手法を **タイム・ボローイング** と言う。後述する **動的タイム・ボローイング** と区別するため、この設計時におけるタイム・ボローイングを **静的タイム・ボローイング** と呼ぶ。

これにより、二相ラッチの遅延制約は累積で $0.5\text{cycle}/0.5$ ステージ、最大遅延制約は $1\text{cycle}/0.5$ ステージとなる。

なお、設計においてはまずステージ間の遅延をバランスさせることが肝要であり、タイム・ボローイングの効果を積極的に利用することは推奨されない。この性質は、クロック・スキーに対する耐性に効果があり [14]、実際にはスキー耐性のために採用されることが多いようである。

2.4 動的タイム・ボローイングを可能にするクロッキング方式

回路構成と動作 図8は動的タイム・ボローイングを可能にするクロッキング方式の回路構成である。図8上2つはそれぞれ単相FFと二相ラッチの回路を示す。単相FFにおける1ステージ分のロジックは二相ラッチにおいて2分されている。

図8下がTF検出と二相ラッチ化の組み合わせによる動的タイム・ボローイングを可能にするクロッキング方式の回路の概略図である。二相ラッチ化に対してさらにTF検出のために、各ラッチに逆相で動作するShadow Latchとサンプリングされた値を比較するXORゲートを追加する。これはRazorで用いられるRazor FFのMain FFをラッチに置き換えた構造となる。

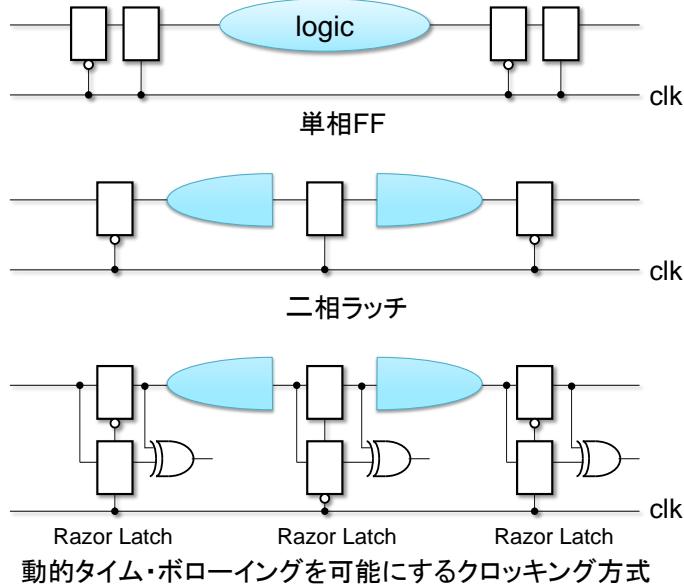


図8: 動的タイム・ボローイングのための回路構成

図9は、このクロッキング方式と二相ラッチ方式のt-diagramを比較したものである。2.3節で述べた制約上、二相ラッチ方式では信号は必ず次のラッチが閉じている期間に到着しなければならず、ラッチが開いている期間は原則使うことができない。各ステージでクリティカル・パスが活性化しなかったとしても、ラッチが開くまで信号の伝播は待たなければならない。

動的タイム・ボローイングを可能にするクロッキング方式では二相ラッチ方式において利用できなかったこのラッチの開いている期間を、TF検出を設けること

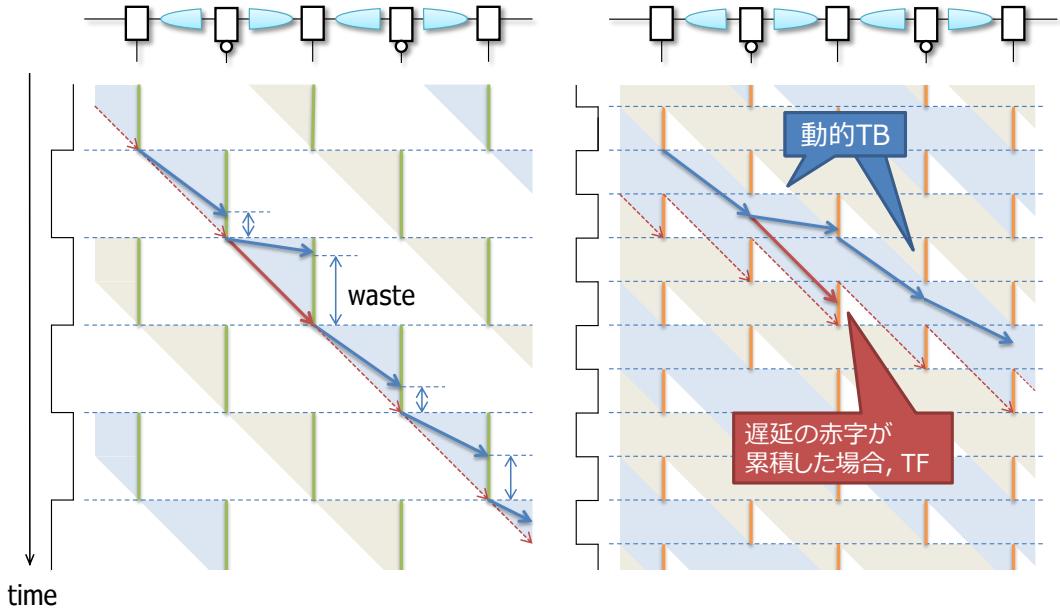


図 9: 二相ラッチ方式（左）と動的タイム・ボローイングを可能にするクロッキング方式（右）の t-diagram

により利用する。これにより、動作時に各ステージで実効遅延を融通することが可能となる。

最大遅延制約の緩和 再度、図 9 に着目する。動的タイム・ボローイングを可能にするクロッキング方式では、遅延が累積し、 $\sum D(i) = -\tau$ となった場合を TF 検出限界となるようサイクル・タイムを定める。各ステージにおいて生じうる、遅延の累積の最大値は $D(i) = -1/2\tau$ であるため、 $\sum D(i) = -1/2\tau$ の状態からクリティカル・パスが活性化し、 $\sum D(i) = -\tau$ になる場合をワースト遅延の境界と定める（図 9 右、赤点線）。すなわち、ラッチ L_{n-1} の閉じる上端からラッチ L_n の検出ウインドウの下端までがワースト遅延の境界となる。

このようにすると、t-diagram 上のクリティカル・パスの遅延によって定められるワースト遅延の境界が階段状となり、サイクル・タイムを詰めることができるとなる。これにより、ラッチの開いている区間を利用できるだけでなく、サイクル・タイムを 0.5 ステージ分のロジックのクリティカル・パスの遅延によって決定できる。

これにより、動的タイム・ボローイングを可能にするクロッキング方式の最大遅延制約は $1\tau/0.5$ ステージとなり、単相 FF 方式や二相ラッチ方式に比べ、最大

2倍の動作周波数の向上を見込むことができる。

回復機構 [15] 既存手法の回復手法はパイプライン・フラッシュによる回復であった。すなわち、TF検出は、例外と同様に扱われ、当該命令とその後続の命令がパイプラインから取り除かれる。しかし、パイプライン・フラッシュでは、FIFOやキューのポインタのような制御系で生じるTFに対処することができない。本手法ではフラッシュではなく初期化によってTFの除去を行う。初期化は全てのポインタ、カウンタの値を0にすることになるため、制御系を含めたTFの影響を除去することができる。

3 SRAMへの適用の問題点

SRAMへの動的タイム・ポローイングを可能にするクロッキング方式の適用にあたって、TF検出機構と二相ラッチ方式への適用を行う必要がある。本章では、SRAMのダイナミックな動作について述べた後に、適用する際の課題点について論じる。

3.1 SRAMの構成と動作

構成 図10に、SRAMの構成を示す。RAMでは、メモリ・セルアレイの各行にワード・ライン(WL)、各列にビット・ライン BL が通っている。メモリ・セルは BL とnMOSを介して接続されており、そのnMOSのゲートは WL で制御されている。以降このnMOSゲートをアクセス・トランジスタと呼ぶ。動作の詳細は3.1で述べるが、アドレス・デコーダによって選択された行の WL は $high$ になり、その行のすべてのアクセス・トランジスタを開くことで、メモリ・セルとビット・ラインとを接続する。なお、同図では書き込みポートは省略している。

図10は8トランジスタメモリ・セル[16]と呼ばれるメモリ・セルの構成をとっている。8トランジスタメモリ・セルでは、アクセス・トランジスタのドレインを、ドレイン側がグラウンドに接続されたnMOSのソース側に接続し、インバータのループの出力でそのnMOSを制御する。以降このnMOSをドライブトランジスタと呼ぶ。このようなメモリ・セル構成は、昨今のばらつきの増大によって十分なSNM(Static Noise Margin)を確保することが困難である状況で多く利用されている[17]。なお、図中ではトランジスタが6つしかないが、これは書き込みポートへのアクセス・トランジスタが省略されているためである。

読み出し動作 SRAMの読み出し動作は、プリチャージと評価が交互に行われることで実現されている。図10では信号 $Pchg$ がその切り替えを制御している。

まず $Pchg$ が low である間は、プリチャージpMOSがオンになることで、ビット・ライン BL が $high$ にプリチャージされる。この間がプリチャージ期間であり、評価の期間と区別される。

$Pchg$ が $high$ である間は評価の期間である。この期間ではアドレス・デコーダの結果選択された行の WL がアサートされて、その行のメモリ・セルのアクセス・トランジスタがすべてオンになる。値が $high$ であるメモリ・セルでは、ドライブ

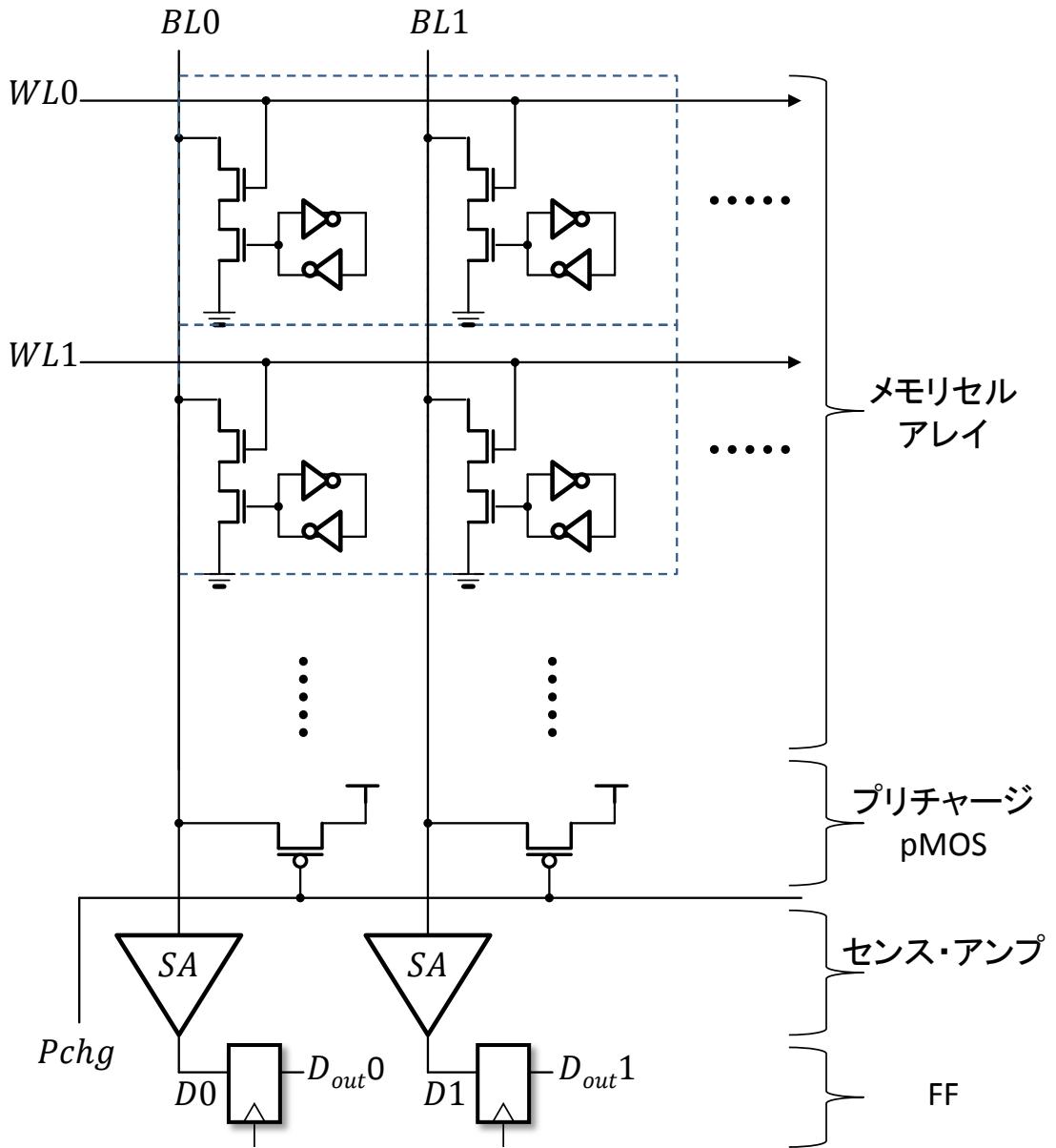


図 10: SRAM の構成

トランジスタがオンになるため、対応するビット・ラインがドライブされる。一方、値が *low* であるメモリ・セルでは、ドライブトランジスタがオフになるため、ビット・ラインは浮遊し、電位レベルは *high* に保たれる。

このように、SRAMにおいてはプリチャージされたノードの電荷がディスチャージされるか否かによって評価を行う。以降、ディスチャージされ *low* となる場合を **0** 読出し、ディスチャージされずに *high* に保たれる場合を **1** 読出しと呼ぶ。ビッ

ト・ラインの電位レベルが *low* の状態で正しい評価を行うことはできないから、評価を正常に行うためには、ビット・ラインのプリチャージが事前になされていなければならない。

3.2 動的タイム・ボローイング適用時の課題

動的タイム・ボローイングを可能にするクロッキング方式を適用するにあたって、TF検出機構への対応と、二相ラッチへの対応を行わなければならない。

TF検出の適用 Razorでは、その適用の対象としてスタティック・ロジックが暗黙のうちに想定されており、特にダイナミック・プリチャージ・ロジック (dynamic precharged logic) に対してそのまま適用することはできない。

このことは、特にSRAMで問題となる。SRAMの読み出しは通常、ダイナミック・プリチャージ・ロジックとして実装されるため、Razorをそのまま適用することができない。SRAMは、今日のLSIにおいて欠くべからざる要素であり、SRAMに適用できないことはRazorの重大な欠点であると言える。

2.2章で述べたように、RazorによるTF検出の正しさを保証するためには、Shadow Latchへの入力がサンプリング時点での正しくなくてはならない。しかしダイナミック・プリチャージ・ロジックにおいてこれを保証することは困難である。そのことを図11を用いて述べる。

図11(左)は対象とするSRAMのセンス・アンプとFF部分を表したものである。BLはビット・ラインを表している。

同図(右)の上下のタイミング・チャートは、どちらも同図(左)の回路のビット・ラインBLとそれを増幅した信号Dの遷移を表している。上下のタイミング・チャートはそれぞれクロック周波数が低い場合と高い場合の動作に対応している。

まず、同図(右)上側において、評価の期間中にBLがディスチャージされて電位がセンス・アンプの閾値を下回り、センス・アンプの出力Dがhighからlowに遷移している。一方、同図(右)下側においては、上側と同じ時間をかけてビット・ラインが遷移すると、Main FFのサンプリング時点では信号Dはhighである。しかし正しくはlowを伝搬しなければならず、TFが発生することが分かる。

このTFが検出されるためには、Shadow Latchがサンプリングする結果が正しい必要がある。しかし、Shadow Latchのサンプリング時点までにDが正しい結果で

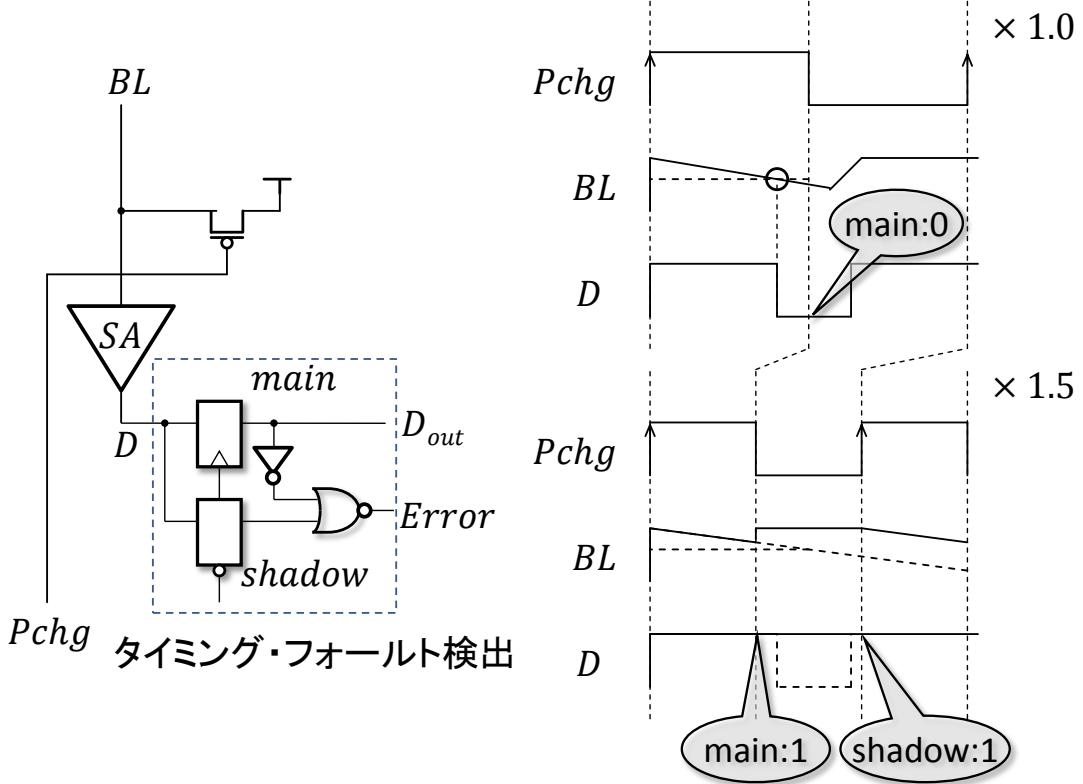


図 11: Razor のダイナミック・ロジックへの適用の問題: (左) SRAM のセンス・アンプ周辺回路, (右) 左図のタイミング・チャート

ある *low* に遷移することはない。なぜならば、Main FF のサンプリングの直後にプリチャージが行われ、ビット・ラインの電位が *high* に引き上げられるからである。

このようにプリチャージ動作のもとでは Shadow Latch がロジックの正しい結果をサンプリングすることが保証されない。すなわちプリチャージ動作が TF をマスクしてしまうため、Razor によっても TF を検出できないという問題がある。

既存適用手法 [18] は、Razor による TF 検出を SRAM において行った先行研究である。本手法は、プリチャージドライバを拡大しプリチャージにかかる時間を短縮した上で、プリチャージクロックをパイプラインのクロックに対して遅らせることで、検出期間を設ける手法である。

図 12 は本手法を用いた場合の回路動作を表す。図のように、プリチャージ期間が短縮され、評価期間がサイクル・タイムの半分を超える、出力側の FF のサンプ

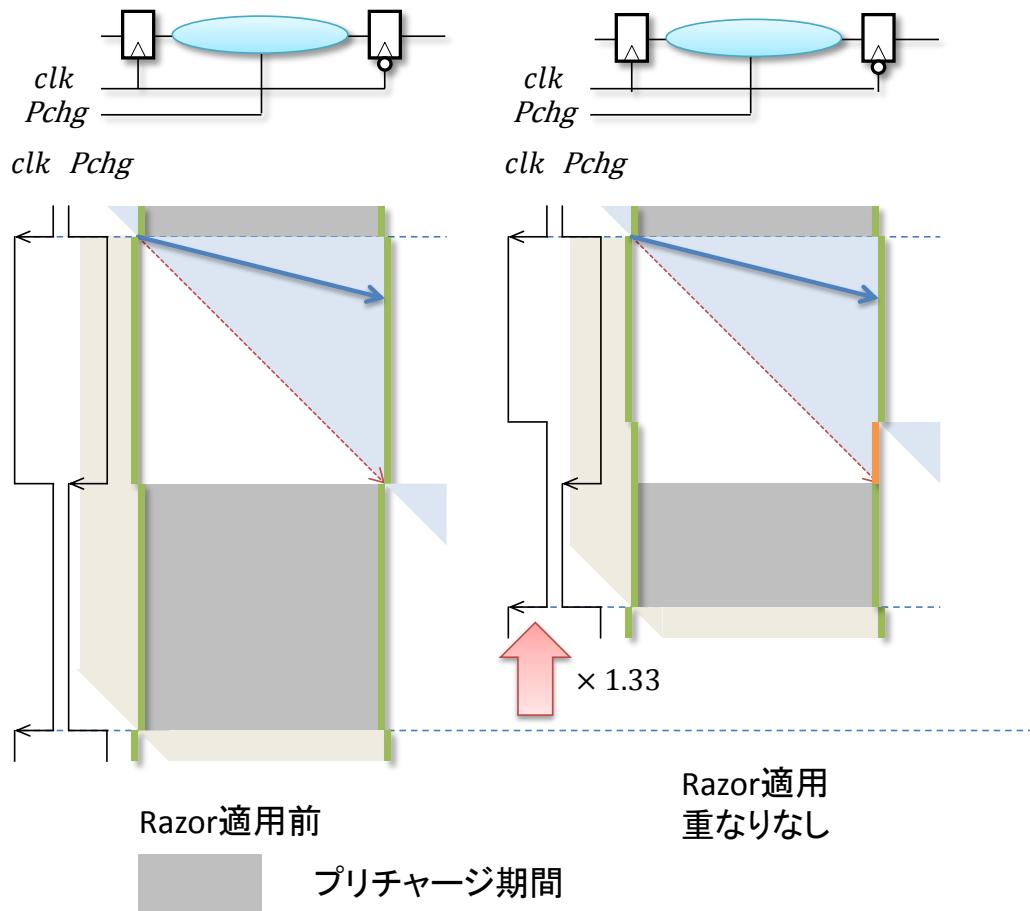


図 12: 既存の Razor の SRAM への適用

リングのクロックエッジに間に合わない場合が生じ得る場合も、出力側のクロックの立ち上がりから評価期間の終わりまでを検出期間として用いることで、最大遅延制約を緩和することができる。

本手法は、プリチャージクロックをパイプライン・クロックと別に設けなければならない点で、回路面積のコストが大きく、実用的ではないと考えられる。

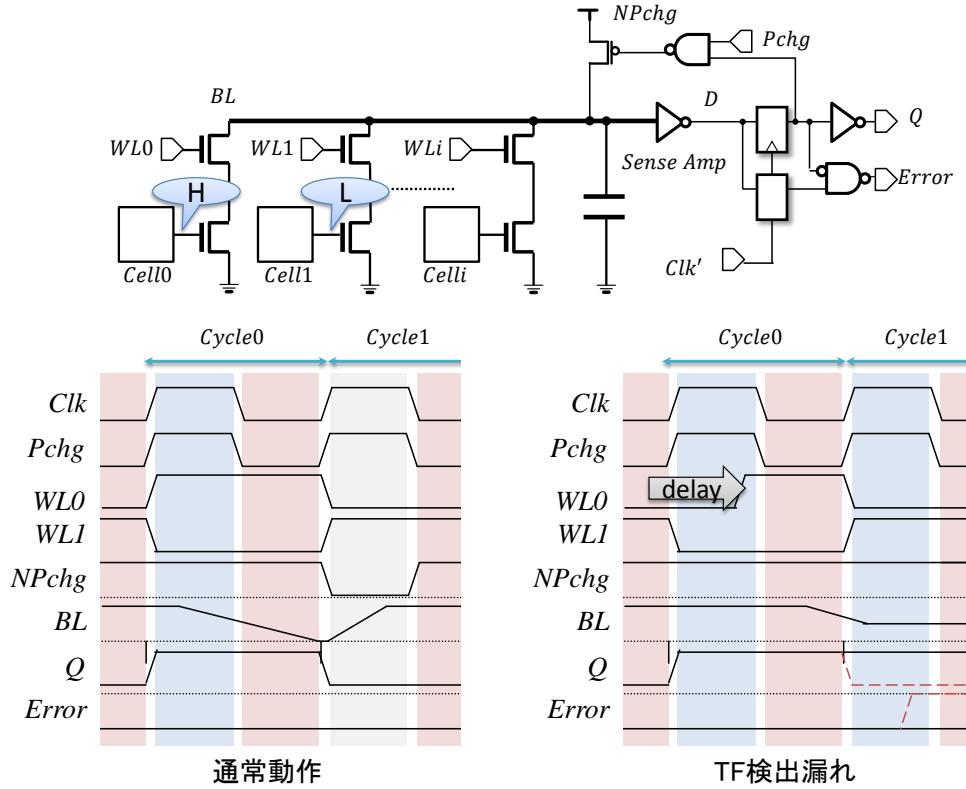


図 13: TF 検出漏れ

二相ラッチへの対応 二相ラッチに対応するにあたっては、SRAMへの入力までの遅延の累積がどの程度であっても回路が正しく動作するように保証する。

図 13 上部では、先述した TF 検出機構を持つ SRAM のビット・ライン 1 つの部分を示している。同図下側は本回路の動作を示したタイミング・チャートであり、左に累積遅延が十分小さく、回路が TF を起こさずに動作する場合を、右には累積遅延が大きく TF を起こしてしまう場合を示している。

本回路では、*Clk* が *high* の区間に入力が transparent に変化するものとする。図中の赤い領域は入力が閉じている区間であり、青い領域は TF 検出期間を表し、灰色の区間はプリチャージ期間を表す。

図 13 (左下) では、サイクル *Cycle0*において *WL0* が *high* となり、*Cell0* にアクセスする。*Cell0* の値は *high* であるから、*BL* はディスチャージされる。次のサイクル *Cycle1* では *WL1* が *high* となり、*Cell1* にアクセスする。*Cell0* の値は *low* であるから、*BL* は *high* に保たれる。

同図 (右下) のようにサイクル *Cycle0*において *WL0* が *high* になるのが遅れ、

さらにビット・ラインのディスチャージの完了も遅れた場合, TFが発生する. しかし, 図示するように検出が正しく行われない場合がある. これは $WL0$ が検出期間の初めの方で *low* になり, それ以降のビット・ラインのディスチャージが起こらなくなることが原因である. TF検出期間を大きく設けたとしても, ビット・ラインの遷移が途中で止まってしまうため, RazorにおけるShadow Latchに正しい値が届かない.

このことから, TF検出期間を十分に設けるためには, サイクル *Cycle1* の前半のTF検出期間中は $WL0$ が *high* に保たれていなければいけない. 単純にワード・ラインの切り替えを半サイクル遅らせることでこれを満たすことは可能だが, 遅延の累積が十分に小さい場合は無駄にレイテンシを増加させてしまう.

4 提案手法

本章では、SRAMに対して動的タイム・ボローイングを可能にするクロッキング方式の適用を可能にする手法を提案する。提案はTF検出の適用と、二相ラッチへの適用に分けられる。

4.1 TF 検出適用のための提案

まず、プリチャージ信号の制御によってSRAMに対しRazorによるTF検出の適用を可能にする手法を提案する。本提案によって、SRAMの読み出しにおける最大遅延制約を緩和することができる。

4.1.1 動作手順と完全性

前章で述べられたTF検出の適用に際しての問題点の克服のために、プリチャージとTF検出の要件について詳細に検討を行う。

SRAMの読み出しが適切に行われるための要件は、ビット・ラインが評価を行う前までに*high*に充電されていることであった。したがって、1-readの後ではプリチャージの必要はない。従来のプリチャージが評価の後に行われる仕組みは、プリチャージ期間の制御クロック、すなわち図中の信号

chbg

の生成を回路のグローバルなクロック信号と同一にすることで、回路の簡略化するために用いられている。

一方でTF検出はどういう場合に必要であるかを改めて検討しよう。SRAMの読み出しにおけるTFは0-readの場合にしか発生しない。なぜならば、1-readはビット・ラインが*high*のまま保持される読み出しであり、遅れが存在しないからである。したがって、SRAMの読み出し時に起こるTFは、0-readの遅れによって、Main Latchのサンプリングの際にビット・ラインがディスチャージされていない状態を観測したときに、それが1-readであると判断されてしまうことに限定できる。このことから、Main Latchのサンプリングの際にビット・ラインがディスチャージされている場合は0-readであると断定でき、TF検出を行う必要がないことが分かる。

このように読み出しの種類に応じて、プリチャージとTF検出はそれぞれ必要とされない場合を内包していることが分かる。Main Latchのサンプリング時点でビッ

ト・ラインがディスチャージされている場合には 0-read だと確定されるため、プリチャージを行い、TF 検出は行わないことで競合を回避できる。問題となるのは、Main Latch のサンプリング時点でビット・ラインが、この場合は 0-read であるか 1-read であるかを Main Latch の値からは判断できない。

提案手法は、Main Latch のサンプリング時点で、SRAM の各ビット・ラインがディスチャージされているか否かによって、プリチャージを以下のように制御するものである。

- ビット・ラインがディスチャージされている場合、プリチャージを行う。
- ビット・ラインがディスチャージされていない場合、プリチャージを行わない。

この提案手法が Razor の適用を可能にすることを、Main Latch のサンプリング時点でのビット・ラインの評価と、読み出し対象のメモリ・セルの値との組み合わせごとに説明する。

ビット・ラインがディスチャージされている場合 SRAM の読み出し動作においてビット・ラインの変化が起こる場合は、ディスチャージされていない状態から、ディスチャージされた状態への変化しかありえない。そのため、Main Latch のサンプリング時点でビット・ラインがディスチャージされている場合はそれ以降の変化は生じないため、それが正しい評価であると見なしてよい。そして、この場合には TF 検出の必要がないので、Shadow Latch のサンプリング時点まで待つ必要がなく、すぐにプリチャージを開始することができる。

ビット・ラインがディスチャージされていない場合 Main Latch のサンプリングの時点でビット・ラインがディスチャージされていない場合は、その周期においてビット・ラインが本来ディスチャージされるべきか否かという正しい結果は、Shadow Latch のサンプリングの時点までは判断できない。

しかし実際には、正しい結果がどちらであろうと、この場合はプリチャージを必要としないことが保証される。なぜならば、正しい結果がディスチャージされる状態である場合、ディスチャージされない状態である場合のそれぞれについて以下で述べるように、プリチャージをしなくとも問題がないからである。

- ビット・ラインがディスチャージされる場合、Razor によって Shadow Latch のサンプル時点において TF が検出され、アキテクチャ・レベルの手法に

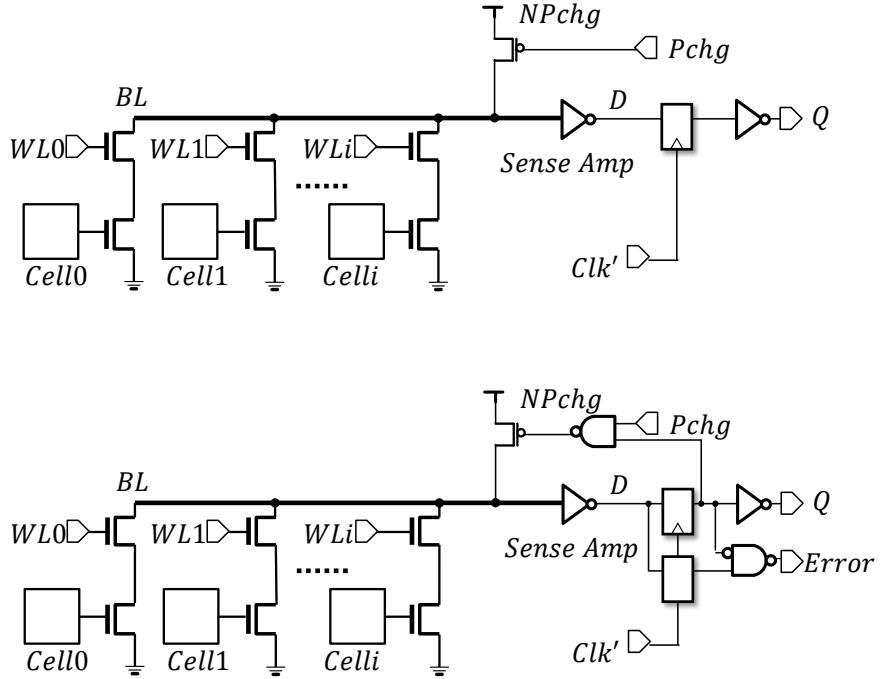


図 14: (上) SRAM のセンス・アンプ周辺回路, (下) TF 検出機構の導入

よって TF からの回復が行われる。したがってこの場合に次サイクルの評価のために即座にプリチャージを行う必要はない。

- ビット・ラインがディスチャージされない場合、Shadow Latch のサンプル時点で Main FF がサンプルした値が正しいことが保証される。一方ビット・ラインはディスチャージされていないため、プリチャージの必要がなく、そのまま次サイクルの評価を開始することができる。

以上から、Main FF のサンプル時点でビット・ラインがディスチャージされていれば即座にプリチャージし、そうでなければプリチャージをしないように制御することによって、Razor の適用が可能になる。この制御はビット・ライン値を制御信号としてプリチャージ信号のゲーティングを行えばよい。

4.1.2 回路構成

図 14 に、その構成を示す。図 14(上)は TF 検出機構導入前の SRAM の、1 つの

ビット・ラインだけを取り出したものである。便宜上、図 10 に対して 90° 傾けて表示している。同図（上）の回路に対して、同図（下）は Razor の TF 検出機構を付加し、読み出し値に応じたプリチャージ信号の制御のためのゲートが付加されている。

なお、同図は論理的な構成を表すものであり、回路実装においてはなるべく面積が少なくなるように実装する必要がある。FF もしくはラッチは、入力の遷移が限定されている場合、ダイナミック・ロジックによって代用することができ、トランジスタ数を大幅に削減することが可能である [14]。

このことを利用してトランジスタ数を抑えた回路実装を図 15 に示す。同図には図 14 の各機能との大まかな対応を示している。ただし Main FF や Shadow Latch の機能はそれに分散されているため、それらの対応は示していない。

例えば、同図のセンス・アンプ部分の最も入力に近いインバータは、出力が *high* であれば、それをプリチャージ期間の間入力から切り離して保持できる。これは、インバータの出力ノードと元々ある nMOS の間に付加された nMOS によるものである。この nMOS はプリチャージ制御信号である *NPchg* をゲート入力としており、*NPchg* が *low* の期間、すなわちプリチャージ期間におけるインバータの出力値がビット・ラインのプリチャージに伴って *low* になってしまふことを防いでいる。

また、ダイナミック・ロジックによる値保持の部分は、Main FF のサンプリング時点でのビット・ライン状態の記憶に用いられている。Main FF のサンプリング時点においてビット・ラインがディスチャージされていないとき、このダイナミック・ロジックの出力ノードはディスチャージされる。その後ビット・ラインの電位が下がったとしても再び出力ノードの電位が *high* に戻ることはないため、Main FF のサンプリング時点においてのビット・ラインの状態を保持できていると言える。

TF 検出部分では、ダイナミック・ロジックによるラッチの部分で記憶されている Main FF のサンプリング値と、センス・アンプが与えるビット・ライン値とを入力とし、*Error* 信号の評価を行っている。

また、同図に明示していないが、*D_{out}* と *Error* はともにプリチャージされる必要がある。ただしこれらは他のビット・ラインと共有するため、必ずしもビット・ラインごとに pMOS が必要となるものではない。

同図の実装においては、ダイナミック・ロジックによるラッチ機能の代用によって同様に省面積化されたセンス・アンプをベースとして、トランジスタ数を 3.5 倍の増加にまで抑えることができる。

提案手法は SRAM のメモリ・セル部には変更を加えず、センス・アンプ周辺の

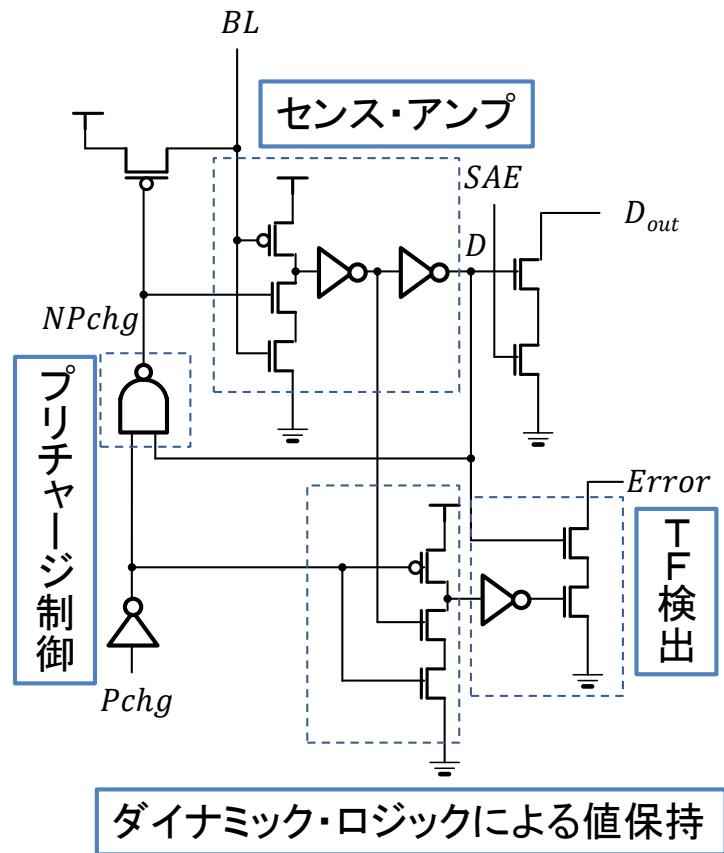


図 15: 提案手法の回路実装

増加に留めている。メモリ・セル部はセンス・アンプ部分に対して大きいため、本手法による回路面積増加は SRAM 自体の二重化手法をとる場合に比べて少ない。図 12 では、同様の主張により面積増加は 8% 程度であるとしており、後述の評価における 16 エントリのレジスタ・ファイルではセンス・アンプ部分とメモリ・セル部分との面積比は約 1:6 であり、トランジスタの増加数から換算すると本手法の SRAM 全体の面積増加は 15% 以下である。

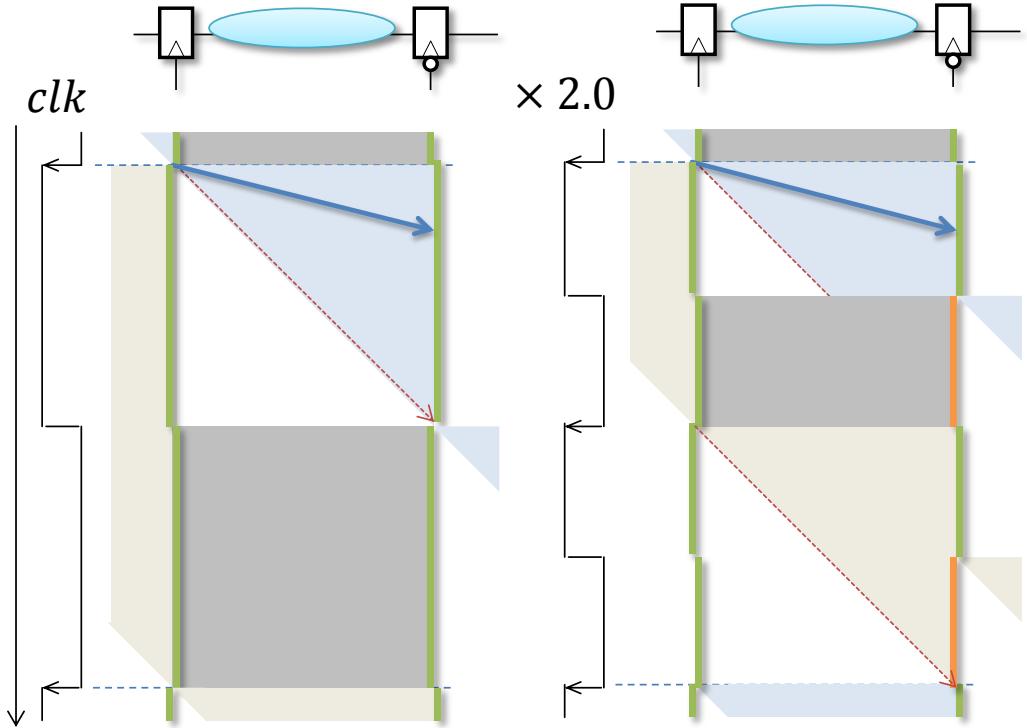


図 16: SRAM の t-diagram : (左) 提案手法適用前 , (右) 提案手法適用後

4.1.3 最大遅延制約の緩和

提案手法を適用した SRAM アクセスステージの t-diagram を図 16 に示す。ここで評価はワード・ラインがアサートされてから、ディスチャージが完了するまでとする。つまり信号の伝達を表す矢印の、開始点はデコーダによるワード・ラインのアサートの開始を表し、終着点はディスチャージがなされる場合はディスチャージの完了を表す。ディスチャージがなされない時には矢印は水平である。また、1 周期におけるプリチャージ期間の割合 β を 0.5 としている。

図 16(左) は、提案手法適用前の回路の t-diagram である。その最大遅延制約は $(1.0 - \beta)\tau/0.5$ ステージ = $0.5\tau/0.5$ ステージである。

一方、提案手法を適用した図 16(右) では、TF として処理できる限界まで最大遅延制約を緩和することができる。ステージの最大遅延に対応する 45° の赤線が検出ウィンドウの下端までに到着すれば TF として処理することができるため、サイクル・タイムに対する検出ウィンドウの割合を α とすると、最大遅延制約は $(0.5 + \alpha)\tau/0.5$ ステージとなる。仮に $\alpha = 0.5$ とすると、同図で示されるように、

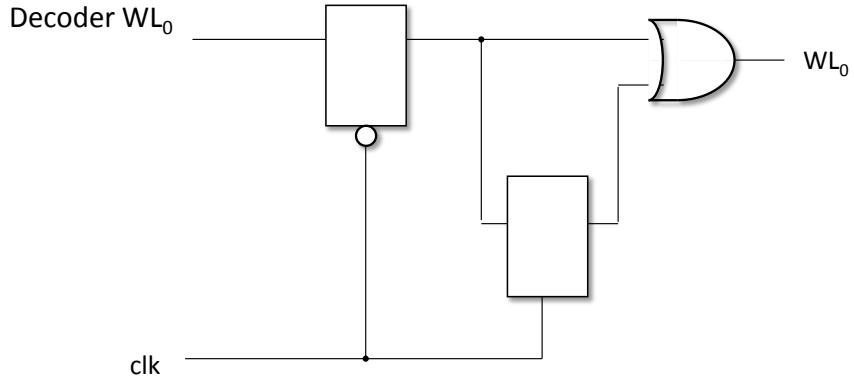


図 17: タイム・ポローイングのための提案回路構成

クロック周波数を最大 2.0 倍まで引き上げることが可能となる.

本章では、ワード・ラインを TF 検出期間の間保持することで、先述した検出限界の問題を解消する。本提案によって、検出幅をワード・ラインの遅延に関係なく設定することができるようになる。

4.2 タイム・ポローイングのための提案

図 17 に提案手法の回路構成を示す。図 17 では、アドレスデコーダの結果を半サイクルだけ保持するラッチを設け、それに記憶された前サイクルの値と今サイクルの値の OR をとって真のワード・ライン信号としている。

4.2.1 動作

前後の命令によってアクセスするエントリが異なる場合、このような回路では、TF 検出期間の間 2 本のワード・ラインの電位が *high* 状態になる。通常の SRAMにおいて、2 本のワード・ラインを *high* 状態にした場合の動作は想定されていない。したがって、本回路が正常に動作するために満たすべき条件について以下に述べる。

まず、SRAM のメモリ・セル構成は 8 トランジスタメモリ・セルを想定する。8 トランジスタメモリ・セルの SRAM では、ビット・ラインとメモリ・セル内のインバータループへの入力が電気的に隔離されているため、セル内容は保護される。

次に、動作が正しく行われるための条件について述べる、本提案方式では、次のような通常の読み出しあクセスでは起きない状態が生じる。

- プリチャージ中もワード・ラインが *high* 状態である
- TF 検出中のセル同士の衝突

プリチャージ中もワード・ラインが *high* 状態である この場合において、ビット・ラインのプリチャージが完了するための条件について考察する。

ビット・ラインのプリチャージが完了するためには、プリチャージ pMOS のドライブ能力がすべてのセルのドライブ能力を上回る必要がある。pMOS のドライブ能力の増加は、ゲート幅の増加によって実現できる。プリチャージ pMOS の面積増大は、RAM が十分大きい時回路面積への影響は小さい。

TF 検出中のセル同士の衝突 この場合において、TF 検出が正しく行われるための条件について考察する。

第 3 章で述べたように、Main Latch によって 1 読出しと判断された場合に TF 検出が行われる。したがって、セル同士が衝突するのは以下の 2 通りに分類できる。

1. 前サイクルが 0 読出しで TF が起きており、次サイクルが 1 読出し
2. 前サイクルが 1 読出しで TF は起きておらず、次サイクルが 0 読出し

(1) に関して、次サイクルに行われる 1 読出しへはビット・ラインに対して影響を与えないため、前サイクルの TF 検出は正しく行われる。

(2) に関して、次サイクルが 0 読出しであるため、ビット・ラインがディスチャージされる。もし TF 検出期間内にビット・ラインが *low* になると TF として扱われてしまい、誤検知となる。これは回路遅延が十分に小さいときに起きるため、動作環境の適切な制限が必要となる。

満たすべき条件のまとめ まず、プリチャージが完了するためにプリチャージ pMOS のドライブ能力の増加が必要となる。これはプリチャージ pMOS のゲート幅の調節によって可能である。また、TF 誤検知がないように動作環境に制約を与える必要がある。動作環境の制約に関しては、SRAM だけでなく、回路全体を考慮して決定しなければならないため、その詳細の議論については今後の課題とする。

表 1: 評価に用いたソフトウェア環境

回路・レイアウトエディタ	Virtuoso Version IC6.1.5_ISR15
RC 抽出	Calibre xACT3D Version 2012.3.31_26
シミュレーション	HSPICE Version H-2013.03
ライブラリ	FreePDK45nm [19]

5 評価

4 章で述べた提案手法に対して、 SPICE シミュレーションによって動作確認を行った。また、サイクル・タイムによる TF,TF 検出の誤検知、TF 検出漏れの発生率について測定し、提案手法の効果を評価した。

5.1 評価環境

表 1 に評価に用いたソフトウェアとテクノロジ・ライブラリを示す。

評価回路 評価には、図 18 に示す回路を用いた。評価回路のモデルは 16 エントリ、64 ビットの SRAM であり、評価回路はその 2 エントリ、1 ビット部分となっている。同図上側はそのビットラインから TF 検出機構周辺であり、同図下側はベース回路と評価回路のワード・ラインドライバ周辺回路を示している。

モデルと相違ない結果を得るために、ワード・ラインとビット・ラインの付加を与えている。ビットラインには動作に用いる他のエントリのアクセス・トランジスタ 14 つのソース部を接続し、配線容量を付加している。またワード・ラインには動作に用いるビット・ライン以外のアクセス・トランジスタ 63 つのゲート部を接続し、配線容量を付加している。それぞれの配線容量はモデルのレイアウトから抽出し、それぞれの平均を用いている。

また、プロセスばらつきとして、ITRS(International Technology Roadmap for Semiconductors)2009 の 47nm 時のばらつき予測値を参考とし、ゲート長ばらつきを $3\sigma = 3.18[\text{nm}]$ 、閾値電圧ばらつきを $3\sigma = 0.075[\text{V}]$ とした。

評価手順 TF 検出の動作検証と、TF の発生率の評価を行う。

TF 検出の動作検証ではメモリ・セル 0 と 1 にあらかじめ 0, 1 を書き込み、電源電圧を変動させつつ (1.00[V], 0.90[V], 0.85[V], 0.80[V])、交互に読み出しアクセスを行った。

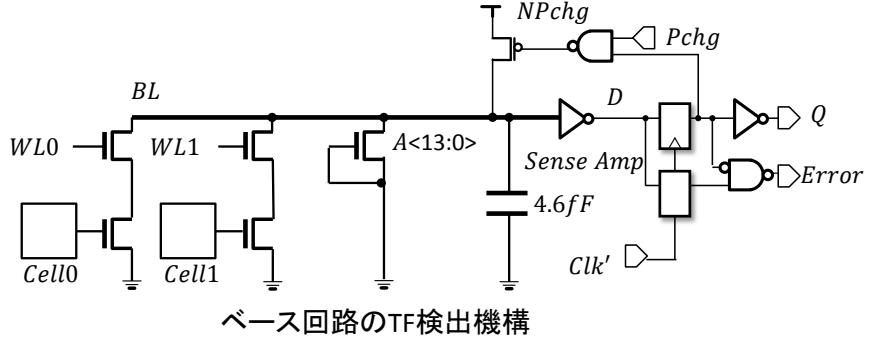


図 18: SPICE シミュレーション用回路

TF と検出漏れの発生率の評価は、次のように行う。あらかじめメモリ・セル 0 と 1 に 1, 0 を書き込む。交互に読み出しアクセスを行い、シミュレーションにおける読み出し結果と正しい値との比較によって真の TF の判定を行い、シミュレーションにおける TF 検出の結果と真の TF の比較によって検出漏れと誤検出の判定を行う。これをプロセスのばらつきを変えて 1000 回試行し、TF やその誤検知の発生率を算出する。さらにサイクル・タイムを 300[ps] から 1000[ps] まで変動させ、サイクル・タイムごとの TF, TF 検出の誤検知, TF 検出漏れの発生率を算出する。なお、温度は 90[°C], 電源電圧は 1.0[V] で一定とする。

5.2 結果

図 19 は SPICE シミュレーションによる動作波形である。ここで、 WL_0 , WL_1 はそれぞれメモリ・セル 0 と 1 のアクセス・トランジスタを制御するワードライン信号, P_{chg} はプリチャージ信号, BL はビット・ライン, NP_{chg} はゲーティング後のプリチャージ信号, D_{out} と $Error$ は図 18 で表されるようにそれぞれ読み出し

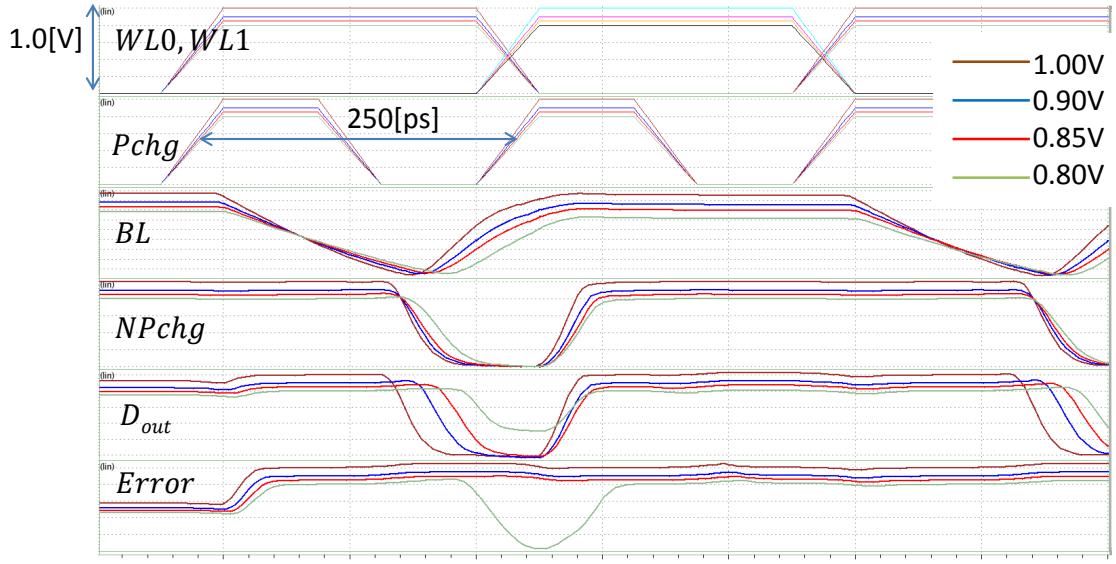


図 19: SPICE シミュレーションによる TF 検出動作の検証. サイクルタイム:250[ps], 温度:55[°C].

値の出力と TF 検出信号である. これらの信号波形が, 電源電圧の異なる場合について重ねて表示されている.

最初の周期では WL_0 がアサートされ, メモリ・セル 0 が読み出される. どの電源電圧条件においても, P_{chg} が *high* になってからビット・ラインが緩やかにディスチャージされはじめていることが見て取れる. 電源電圧が $0.85[V]$ より高い状態では, 読み出しは問題なく行われている. 一方電源電圧が $0.80[V]$ の状態では, 読み出しはうの信号 D_{out} は閾値近くにあり, これより低い電源電圧の環境下において TF が発生する確率は高いといえる. しかしこのときエラー信号 $Error$ がアサートされ, TF を検出している. 実際は TF が検出された場合は TF からの回復処理を行うことになる.

次の周期では, WL_1 がアサートされ, メモリ・セル 1 が読み出される. このときは BL のディスチャージは起こらない. P_{chg} が *low* になるタイミングで, NP_{chg} はゲーティングされて *high* のままであり, 評価が継続している. 次の周期の開始までに BL はディスチャージされなかったため, TF は発生していない. また, BL は電位が *high* レベルにあるので, 評価を行うことができる.

次に, 図 20 はサイクル・タイムごとの TF, TF 検出誤検知, TF 検出漏れの発生

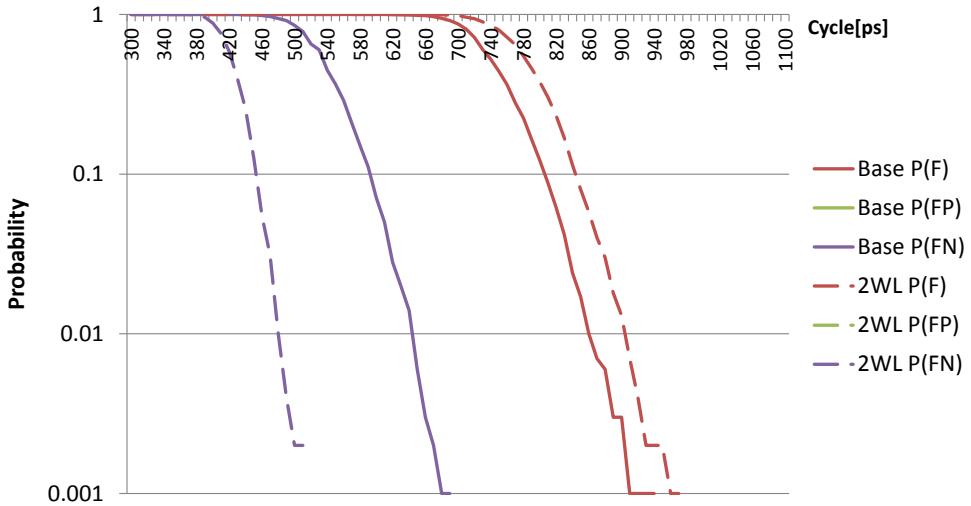


図 20: サイクル・タイムごとの TF, TF 検出誤検知, TF 検出漏れの発生率. 遅延累積あり

率を示す. 図中の実線はベース, 点線は提案手法の結果を表している. TF , FP , FN はそれぞれ TF, 誤検知, 検出漏れの発生率を表している.

TF 発生率が 0.001 を超えるサイクル・タイムの最大値は, ベース回路では 940[ps], 提案手法では 960[ps] となっている. これはワード・ラインに OR ゲートを挟んだことによって遅延が伸びたことによるもので, 本質的な問題とはならない. 誤検知の発生率が 0.001 を超えるサイクル・タイムの最小値は, ベース回路で 700[ps], 提案手法で 720[ps] となり, 20[ps] 改善している. これはワード・ラインに OR ゲートを挟んだことによって遅延が伸びたことによる. また, 検出漏れの発生率が 0.001 を超えるサイクル・タイムの最大値は, ベース回路で 660[ps], 提案手法で 500[ps] となり, 160[ps] 改善している.

5.3 考察

TF 検出が可能になったことで, 設計可能な最小サイクル・タイムを TF が発生し始める 940[ps] から, TF 検出漏れが発生し始める 660[ps] まで削減することができる. またワード・ラインに対する提案により, TF 検出漏れの発生し始めるサイ

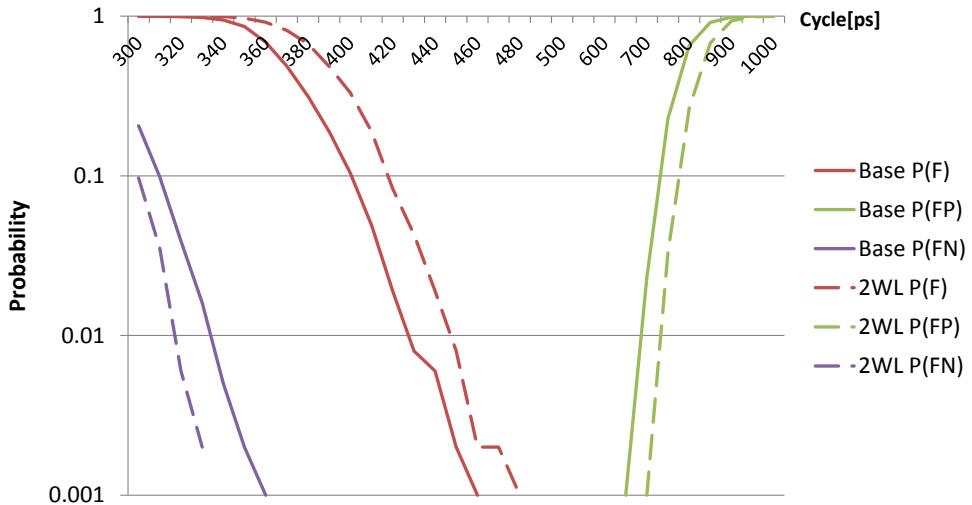


図 21: サイクル・タイムごとの TF, TF 検出誤検知, TF 検出漏れの発生率. 遅延累積なし

クル・タイムが小さくなり, 提案全体では最小サイクル・タイムの限界値を最大 47% 削減できることを示している.

これらの結果は, 回路の 1 ステージの遅延制約の緩和の程度を示すものであり, 回路全体の性能向上の程度を表すものではない. 実際の回路のサイクル・タイムは, TF 発生による回復のオーバーヘッドが性能向上を妨げないように決定される. また, タイム・ボローイングによる性能向上の寄与は, SRAM の前段のパイプライン・ステージの遅延累積を考慮しなければならないが, 遅延の累積がどのように分布するかは個々の回路実装に依存し正確には得られない. ただし, 遅延が累積することは稀であることがいくつかの文献から推察される [20][21]. したがって, TF 検出の検出限界に関する制約を, 累積最大ケースにおいても満たされるように設計した上で, 累積が小さい場合の動作を平均的ケースとして性能向上の寄与とすることは現時点の評価では妥当であるといえる.

TF 検出誤検知の発生は, 遅延累積がなく, かつ十分にサイクル・タイムが長い場合に増えている. DVFS を用いて回路の動作点を決定する場合, CPU 負荷率が大きくない場合などに消費電力を削減する目的で, クロック周波数を低下させる場合がある. そのため, サイクル・タイムが長い場合の運用も可能であることが

望ましいが、RazorによるTF検出には結果のようにサイクル・タイムの最大値に制約が存在する。これに関しては、サイクル・タイムが長い場合のTF発生が起こらないことが補償されていれば、検出ロジックをゲーティングするなどの対応を考えられる。しかし、遅延累積がある場合の結果から、TF検出誤検知の発生し始めるサイクル・タイムの最小値においては、遅延累積が最大の場合TFが発生し得ることが分かり、シミュレーションした回路においては上記の手法では対応できない。TF検出誤検知の抑制に関しては今後の課題とする。

6 おわりに

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

我々は、ステージ間の遅延の融通によって、より効果的なクロック周波数向上や電圧削減を可能にする手法である、動的にステージ間のタイム・ボローイングを可能にする手法を提案した。この手法は、二相ラッチとTF検出を組み合わせることで実現される。

本稿では、SRAMへの動的タイム・ボローイングを可能にするクロッキング方式の適用について、TF検出と、二相ラッチ化の適用手法の提案を行った。SRAMにおけるプリチャージ動作がタイミング・フォールトをマスクしてしまう問題に関して、SRAMの評価結果に応じて、プリチャージの有無を制御することによって適用を可能にする。また、タイム・ボローイングを許容するための設計ワード・ラインをTF検出期間の間保持する方式は、プリチャージ完了のためのpMOSの面積増と、検出幅の適切な設定によって正しく動作することが保証される。提案手法をSPICEシミュレーションによって検証し、検出幅の限界を拡張することが可能であることを示した。

今後の課題として、我々は現在これらの技術を取り入れた高効率なout-of-orderスーパスカラ・プロセッサの開発を目指している。EDIFフォーマットのネットリストを読み込んで、自動的に回路を解析し、提案手法の適用を行うツールを作成中であり、これを用いて適用を行う。開発の目的は以下の通りである。

- 実際の回路上でどのように適用可能なのかを調査する回路遅延が問題になる部分はどこか、回路面積の増大を抑えるバスの分離の仕方など
- 適用した回路のクロック周波数・電圧を変動させた場合の挙動の分析適用の仕方・適用した数と、変動可能範囲の関係など
- 面積増加がどの程度かの確認
- 上記の調査結果をフィードバックして最適な回路構成を検討する。

先に触れたが、TF誤検知や検出漏れが生じないように回路の動作環境を設定する。そのためにはステージ間の遅延累積の程度に関する理解が必要である。本論文の評価では累積遅延が十分に小さい場合TF誤検知が発生する可能性を含み、クロック周波数が小さい、あるいは電源電圧が高いといった比較的TF発生のないと考えられる状態において、TF誤検知の発生が増大することを観察した。また、累積の遅延が大きい場合のTF発生を起こすクロック周波数と、累積が小さい場合のTF誤検知発生のクロック周波数帯が重なっている。この場合、累積の大きさを考慮せずにタイム・ボローイングを前提とした設計をすることは困難である。

プロセッサを対象としてステージ間の遅延累積の程度を考慮したより実用的なタイミング制約の検討を行う。実際の回路では、動作点(周波数/電源電圧)の変動に応じて累積遅延の大きさの統計的傾向も変動するため、TF誤検知の発生について現実に即した対応を検討することが可能になる。

7 関連研究

本章では関連する研究について述べる。まず動的タイム・ボローイングを可能にするクロッキング方式に関する議論として、他の LSI ばらつき対応を試みたアーキテクチャ/回路技術について述べる。続いて、SRAM に対する Razor の適用手法について先行研究を述べる。

7.1 ばらつき耐性を狙った回路/アーキテクチャレベル技術

7.1.1 ReCycle

ReCycle[22] は、cycle time stealing [23] と呼ばれる技術をプロセッサ・パイプラインに適用したものである。図 22 はステージ間の遅延がバランスしていない場合の単相 FF 方式（左）と ReCycle（右）の t-diagram である。

2.3 節で述べたように、通常の単相 FF 方式では遅延の大きいステージのクリティカル・パスの遅延によってサイクル・タイムが定まり、遅延の小さいステージでは、サイクル・タイムに無駄が生じてしまう。二相ラッチ方式では、ラッチを置く位置を変更することで、ステージ間で時間を融通していた。

ReCycle は、遅延の大きいステージのロジックのパス遅延を予め解析する。これにより、そのステージの値をサンプリングする単相 FF のサンプリング・タイミングを、最短パスの遅延分まで遅らせることが可能となり、遅延の小さいステージのサイクル・タイムを分配し、遅延の大きいステージに 1 サイクル以上をかけることで、サイクル・タイムを短縮する手法ということができる。

7.1.2 TIMBER

[21] にはまず、チップ内におけるクリティカル・パスの分布が記されており、クリティカル・パス遅延のステージが連続する確率は極めて小さい、すなわち、クリティカル・パス遅延をもつステージの次のステージは遅延の小さいステージであることが多いということが示されている。このことを利用して、TIMBER は単相 FF 方式の検出ウィンドウにラッチの開いている区間を作ることによって TF 検出の発生回数を抑えることを提案している。

図 23 に TIMBER の t-diagram を示す。TIMBER はサンプリング・タイミングが異なるマスター・ラッチを複製し、セレクタによりスレーブ・ラッチへの出力を

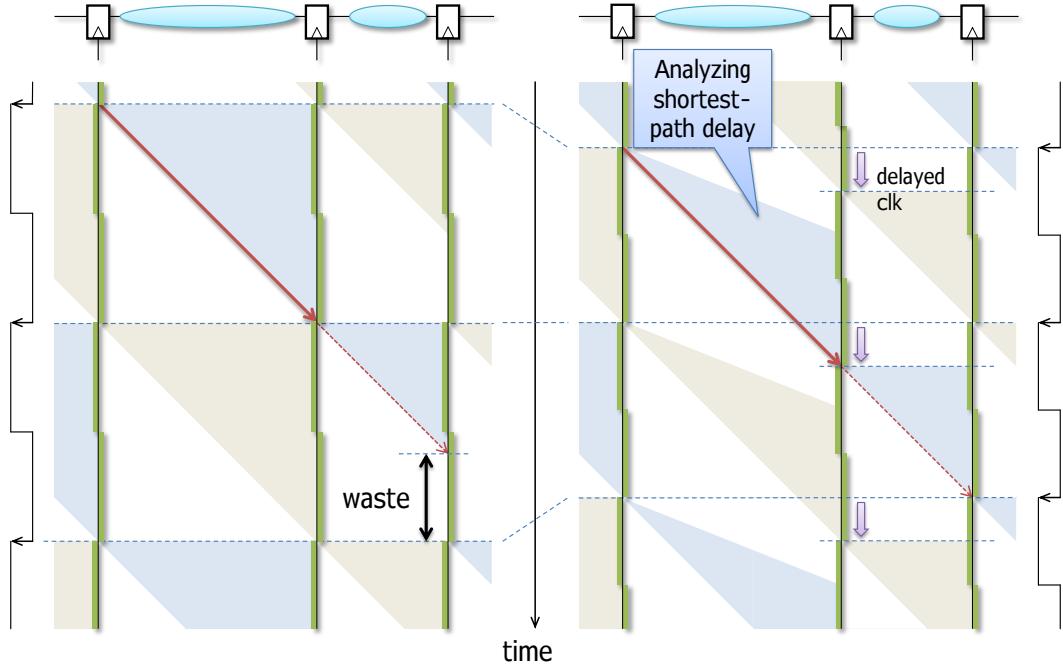


図 22: 単相 FF (左) と ReCycle (右) の t-diagram

切り替える。これにより、単相 FF 方式ながらラッチの開いている区間ができる、仮にクリティカル・パスが活性化しても次のステージでシグナルを早く伝搬させることで、次段のサンプリングに間に合わせることが可能である。

仮に 2 ステージ連続してクリティカル・パスが活性化した場合は、マスター・ラッチの値を比較して TF を検出する。検出時には、アーキテクチャ・レベルによる回復ではなく、セレクタの制御信号によりラッチの開いている区間を広げる回路レベルの処置が施される。

TIMBER の最大遅延制約は、ラッチの開いている区間の割合を β とすると、 $(1 + \beta)\tau/1$ ステージとなる。

図 24 は TIMBER と提案手法を比較した t-diagram である。TIMBER は単相 FF 方式の位相を遅らせた分、検出ウィンドウの割合が多くなり、提案手法や Razor に比べ、2.2 節で述べたように最小遅延制約が厳しくなる。さらに、動作中においても最小遅延制約が変動するため、回路中のショート・パスが満たすべき遅延制約はより厳しいものとなってしまう。

そもそも、TIMBER は半サイクルの間に、ラッチの開いている区間と検出する区間の二つを設けているため、ラッチの開いている区間の割合 β は、Razor などの検出ウィンドウの割合 α よりも小さい。そのため、Razor よりも理論上のサイク

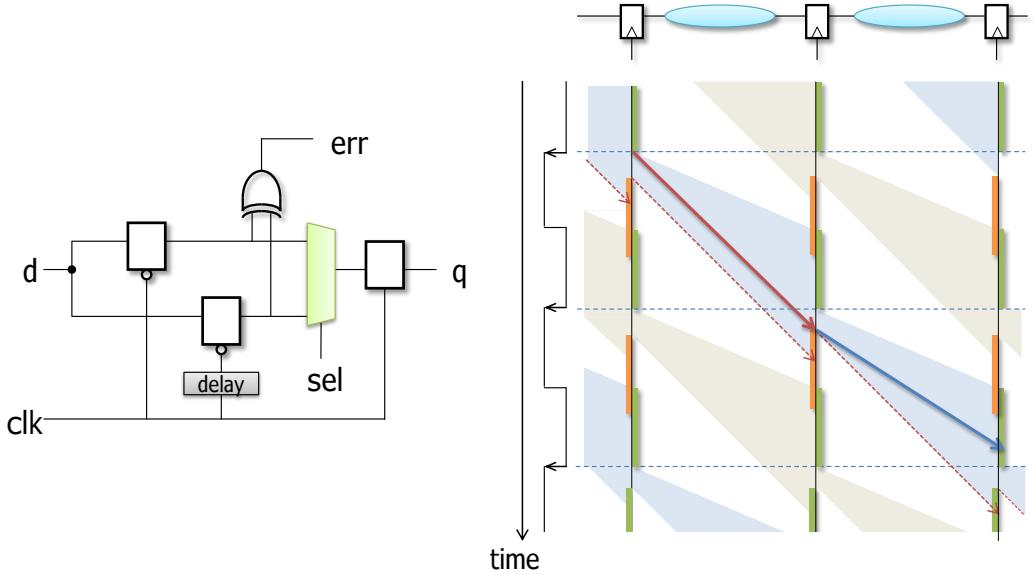


図 23: TIMBER の回路構成と t-diagram

ル・タイムが短縮されない。

さらに、TF 検出時のクロックを遅らせる処置が遅延素子をクロックに挿入することで行われており、現実的ではない。遅延素子における遅延が固定値となるため、DVFS などにより周波数を変動させると、期待通りのクロックが生成されないことが予想される。

7.1.3 Bubble Razor

Bubble Razor[24][25] は二相ラッチ方式に TF 検出を組み込むという、我々の提案手法と類似したアプローチをとっているが、TF の検出ウィンドウのとり方など、動作は大きく異なる。図 25 は Bubble Razor の動作を示した t-diagram である。

我々の提案する手法とは異なり、Bubble Razor ではラッチの開いている区間ににおいて値が変化した場合を TF と定め、検出ウィンドウを設けている。(図 25a)

検出ウィンドウにかかるパスが活性化した場合(図 25b-1)、クロック・ゲーティングにより、次段のラッチの開いている区間を閉じる。(図 25b-2) この閉じている区間を Bubble Razor では、Bubble と呼んでいる。これにより、TF の発生した次のステージで仮にクリティカル・パス遅延で信号が伝播したとしても、信号をサンプリングすることが可能になり、silent error を回避できる。

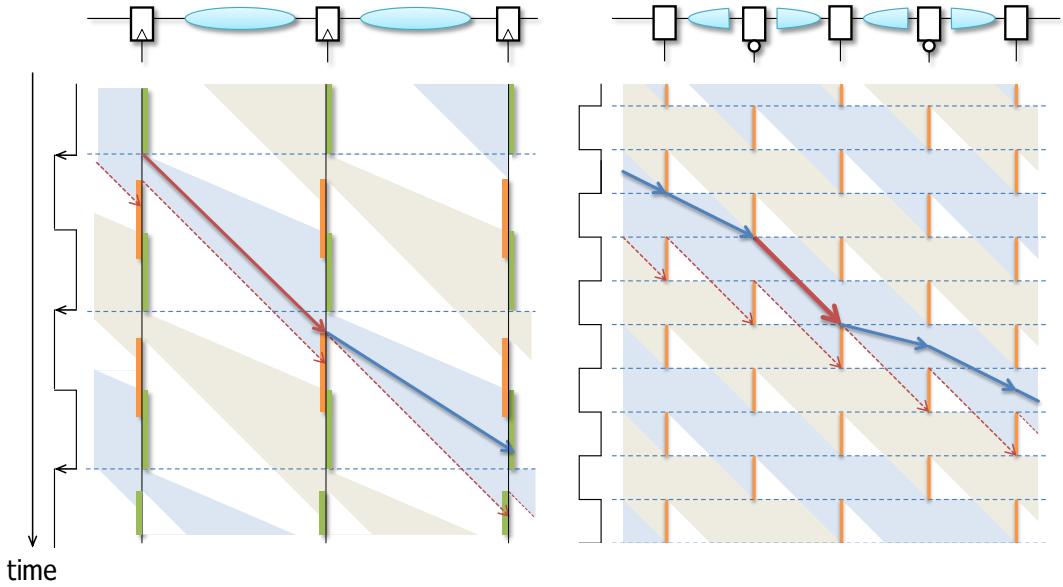


図 24: TIMBER (左) と提案手法 (右) の t-diagram

しかし、次段のラッチを閉じただけでは、次のフェーズの信号と混ざる問題や、前のフェーズの信号が伝播してしまう問題が生じる。(図 25b-3) そのため、Bubble Razor では Bubble が生じたラッチの前後のラッチへと 0.5cycle 遅れて Bubble を伝搬させ続けることにより対処する。(図 25b-4)

図 25b-5 のように、前のフェーズでも TF が発生した場合、Bubble が前後のラッチから同時に到達することがある。この場合は Bubble の伝搬を止め、必要以上にラッチを閉じないように設計されている。これにより、ループ回路やフォワーディングパスへの対応を行っている。

Bubble Razor の最大遅延制約は検出ウィンドウの割合を α とすると、 $(0.5 + \alpha)\tau/0.5$ ステージ、TF とならない最大遅延制約は $0.5\tau/0.5$ ステージとなる。

図 26 は Bubble Razor と提案手法を比較した t-diagram である。Bubble Razor と提案手法の最高動作周波数の理想値は共に同じであるが、Bubble Razor は二相ラッチ方式を採用しているにも関わらず、検出ウィンドウをラッチの開いている区間に設けたことにより、通常動作時において、タイム・ボローイングが出来なくなっている。これにより、2.2 節で述べた Razor と提案手法の比較同様、Bubble Razor はクリティカル・パスに近い遅延のパスが活性化した場合、必ず TF が検出され Bubble が発生する。結局のところ、Bubble Razor はクリティカル・パスの遅延以上にサイクル・タイムを短縮できず、提案手法と比べて、動作周波数は向上でき

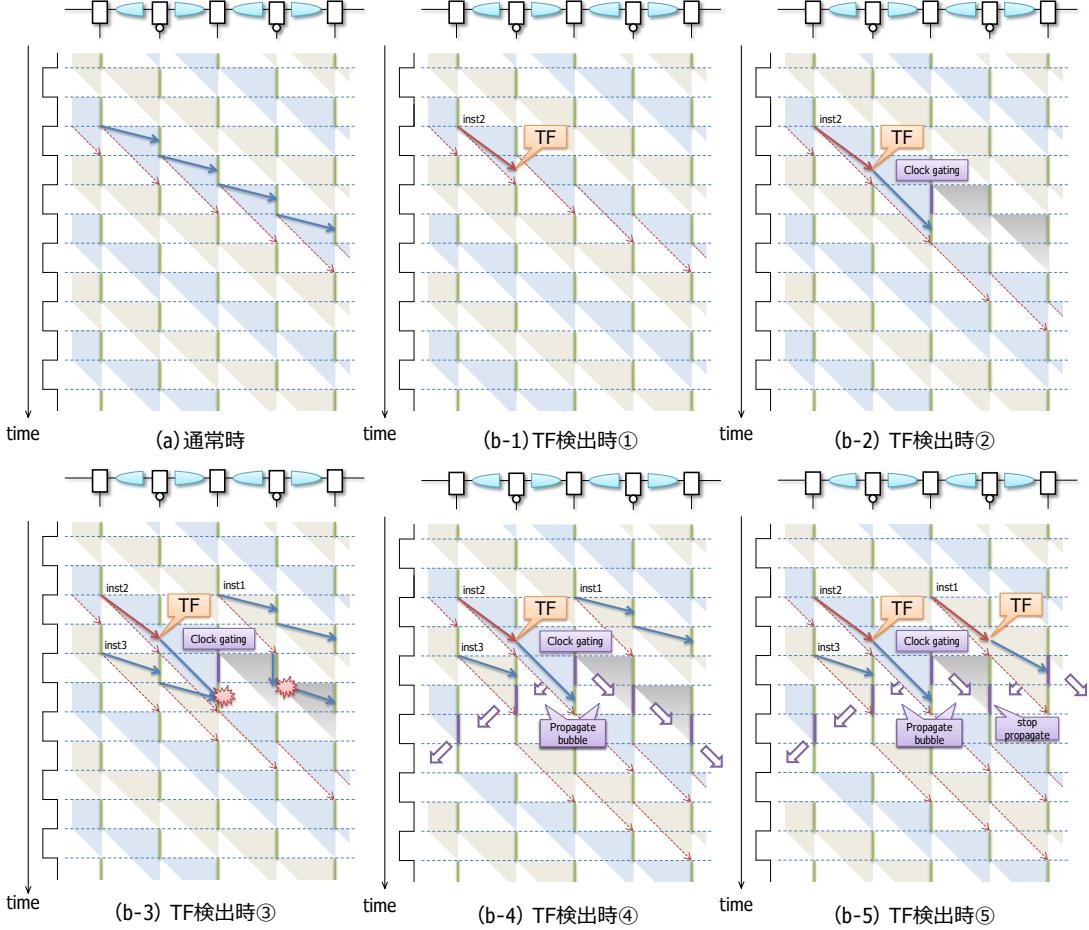


図 25: Bubble Razor の t-diagram

ないものと予想される。

7.2 SRAMへのRazor適用の既存手法

7.2.1 ダブルエンドビットラインを対象としたメモリのTF検出手法

入江らが提案した、ダブルエンドのビットラインを前提とした TFD の仕組み [26]。Razor のような時間的冗長化ではなく、二つのビットラインが正しく読み出しが行われた場合必ず相補的な関係になることを利用して TF を検出する。Main FF のサンプリングのタイミングで遅れを即座に検出することができる。配線長のばらつきから、ダブルエンドのセンスアンプは設計要件を満たすのが困難になってしまっており、かつ面積的制約から用途が限られている。本論文に含まれる提案はシン

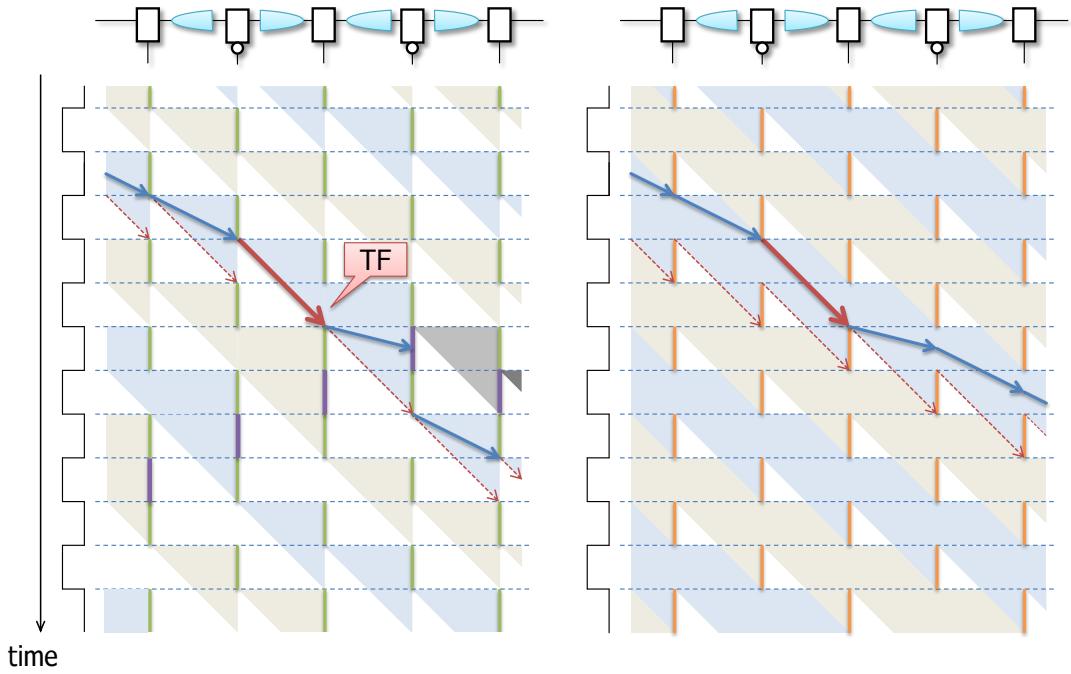


図 26: Bubble Razor (左) と提案手法 (右) の t-diagram

グルエンドの提案を前提としている。

7.2.2 シャドウセンスアンプ

プリチャージ期間を短くして、開いた時間を TF 検出に割り当てる手法 [18]. 対象としてはダブルエンドであるが、本質的な点でシングルエンドにも適用可能。プリチャージゲートを大きくする必要がある点は本論文の提案と同様だが、プリチャージ制御クロックをパイプラインクロックと別に生成する必要があり、回路面積コストとなる。また、最大遅延制約はプリチャージの短縮率を a として $0.5(1+a)cycle/0.5$ ステージであり、短縮率 a を限りなく 0 にしなければ本論文と同等の制約条件とならないが、短縮率 a を小さくするためにはよりプリチャージゲートを大きくしてプリチャージにかかる時間を削減しなければならないため、回路面積コストが本論文の提案に比べて大きいと考えられる。

参考文献

- [1] 平本俊郎, 竹内潔, 西田彰男: 1.MOS トランジスタのスケーリングに伴う特性ばらつき (小特集 CMOS デバイスの微細化に伴う特性ばらつきの増大とその対策), 電子情報通信学会誌, Vol. 92, No. 6, pp. 416–426 (2009).
- [2] 健一岡田: 集積回路における性能ばらつき解析に関する研究 (2003).
- [3] Ashish, S., Dennis, S. and David, B.: Statistical Analysis and Optimization for VLSI: Timing and Power, ISBN: 978-0-387-25738-9 (2005).
- [4] Mukhopadhyay, S., Mahmoodi, H. and Roy, K.: Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 24, No. 12, pp. 1859–1880 (2005). ID: 1.
- [5] 佐藤寿倫: カナリア・フリップフロップを利用する省電力マイクロプロセッサの評価, 先進的計算基盤シンポジウム SACSIS, pp. 227–234 (2007).
- [6] Kunitake, Y., Sato, T., Yasuura, H. and Hayashida, T.: A Selective replacement method for timing-error-predicting flip-flops, *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pp. 1 –4 (2011).
- [7] D.Ernst, N.Kim, S.Das, S.Pant, T.Pham, R.Rao, C.Ziesler, D.Blaauw, T.Austin and T.Mudge: Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 7–18 (2003).
- [8] Das, S., Tokunaga, C., Pant, S., Ma, W.-H., Kalaiselvan, S., Lai, K., Bull, D. and Blaauw, D.: RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance, *Solid-State Circuits, IEEE Journal of*, Vol. 44, No. 1, pp. 32–48 (2009).
- [9] Bull, D., Das, S., Shivshankar, K., Dasika, G., Flautner, K. and Blaauw, D.: A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation, *Solid-State Circuits Conference Digest of Technical Papers (ISSCC-57), 2010 IEEE International*, pp. 284 –285 (2010).

- [10] Bowman, K. A., Tschanz, J. W., Kim, N. S., Lee, J. C., Wilkerson, C. B., Lu, S. L., Karnik, T. and De, V. K.: Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance, *Solid-State Circuits, IEEE Journal of*, Vol. 44, No. 1, pp. 49–63 (2009). ID: 1.
- [11] Mallik, A., Cosgrove, J., Dick, R. P., Memik, G. and Dinda, P.: PICSEL: Measuring User-Perceived Performance to Control Dynamic Frequency Scaling, *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp. 70–79 (2008).
- [12] Das, S., Pant, S., Roberts, D., Lee, S., Blaauw, D., Austin, T., Mudge, T. and Flautner, K.: A self-tuning DVS processor using delay-error detection and correction, *VLSI Circuits, 2005. Digest of Technical Papers. 2005 Symposium on*, pp. 258–261 (2005). ID: 1.
- [13] 宗史吉田, 壮一郎広畑, 成己倉田, 亮太塩谷, 正裕五島, 修一坂井: 動的タイム・ボローイングを可能にするクロッキング方式, 情報処理学会論文誌コンピューティングシステム (ACS) , Vol. 6, No. 1, pp. 1–16 (2013).
- [14] Harris, D.: Skew-tolerant Circuit Design, pp. 12–14 (2001). Section 1.3 Skew Tolerant Circuit Design.
- [15] 正裕五島, 成己倉田, 亮太塩谷, 修一坂井: タイミング・フォールト耐性を持つ Out-of-Order プロセッサ, 情報処理学会論文誌. コンピューティングシステム, Vol. 6, No. 1, pp. 17–30 (2013).
- [16] Chang, L., Fried, D. M., Hergenrother, J., Sleight, J. W., Dennard, R. H., Montoye, R. K., Sekaric, L., McNab, S. J., Topol, A. W., Adams, C. D., Guarini, K. W. and Haensch, W.: Stable SRAM cell design for the 32 nm node and beyond, *VLSI Technology, 2005. Digest of Technical Papers. 2005 Symposium on*, pp. 128–129 (2005). ID: 1.
- [17] Kumar, R. and Hinton, G.: A family of 45nm IA processors, *Solid-State Circuits Conference - Digest of Technical Papers (ISSCC-56), 2009 IEEE International*, pp. 58–59 (2009). ID: 1.

- [18] Karl, E., Sylvester, D. and Blaauw, D.: Timing error correction techniques for voltage-scalable on-chip memories, *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, pp. 3563–3566 Vol. 4 (2005). ID: 1.
- [19] University, N. C. S.: NCSU EDA Wiki.
http://www.eda.ncsu.edu/wiki/NCSU_EDA_Wiki.
- [20] 喜多貴信, 塩谷亮太, 五島正裕, 坂井修一: タイミング制約を緩和するクロッキング方式, 先進的計算基盤シンポジウム SACSIS, pp. 347–354 (2010).
- [21] Choudhury, M., Chandra, V., Mohanram, K. and Aitken, R.: TIMBER: Time borrowing and error relaying for online timing error resilience, *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 1554 –1559 (2010).
- [22] Tiwari, A., Sarangi, S. R. and Torrellas, J.: ReCycle: pipeline adaptation to tolerate process variation, *ISCA*, pp. 323–334 (2007).
- [23] Bernstein, K., Carrig, K. M., Durham, C. M., Hansen, P. R., Hogenmiller, D., Nowak, E. J. and Rohrer, N. J.: *High speed CMOS design styles*, Kluwer Academic Publishers, Norwell, MA, USA (1998).
- [24] Fojtik, M., Fick, D., Kim, Y., Pinckney, N. R., Harris, D. M., Blaauw, D. and Sylvester, D.: Bubble Razor: An architecture-independent approach to timing-error detection and correction, *ISSCC*, pp. 488–490 (2012).
- [25] Kim, H., Kim, D., Kim, J.-J., Yoo, S. and Lee, S.: Coarse-grained Bubble Razor to exploit the potential of two-phase transparent latch designs, *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pp. 1–6 (2014). ID: 1.
- [26] 入江英嗣, 五島正裕, 坂井修一: メモリ装置およびメモリ読み出しエラー検出方法 (2008).
- [27] 神保潮, 山田淳二, 五島正裕, 坂井修一: ダイナミック・ロジックへのタイミング・フォールト検出手法の適用, 情報処理学会研究報告. システムソフトウェアとオペレーティング・システム, Vol. 2014, No. 18, pp. 1–8 (2014).

*研究業績

口頭発表（査読なし）

1. 神保 潮, 山田 淳二, 五島 正裕, 坂井 修一: ダイナミック・ロジックへのタイミング・フォールト検出手法の適用, 情報処理学会研究報告 2014-ARC-210 , No. 18, pp. 1-8 (2014).
2. 神保 潮, 五島 正裕, 坂井 修一: タイミング・フォールト検出手法の RAM への適用, 情報処理学会研究報告 2015-ARC-214 , No. 8, pp. 1-8 (2015).

本研究の一部は、JST CREST「ディペンダブルVLSIシステムの基盤技術」「アーキテクチャと形式的検証による超ディペンダブルVLSI」、および科学研究費補助金基盤研究(B)・26280012「レジリエンス指向コンピュータシステムに関する研究」の支援と、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し、シノプシス株式会社、日本ケイデンス株式会社、メンター株式会社の協力で行われたものです。

また、非常に多くの方々から多大なご指導、ご協力、励ましを頂き、本論文を完成させることができました。

坂井修一教授には、相談会等で有益かつ鋭いご指摘をいただきました。レジスター・ファイルの配線容量の抽出において、塩谷 亮太 助教(名古屋大学大学院・工学研究科)にはレイアウト・データをいただきました。

山田淳二氏には、回路設計における助言やシミュレータの使用法のレクチャーといった形でサポートしていただきました。

八木原晴水さんには、研究室における設備の導入や各種事務手続き・おいしいお菓子など、研究室で過ごすための様々なご支援を頂きました。

その他、愚痴を聞いてくださった研究室メンバーの同期や、後輩の方々、皆さまに感謝申し上げます。