Vol.2013-ARC-204 No.11
2013/3/27

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手法の実装

広畑 壮一郎[1]　神原 太郎[2]　吉田 宗史[1]　倉田 成己[1]　五島 正裕[1]　坂井 修一[1]

概要：半導体プロセスの微細化に伴う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. 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. そのため, ワースト・ケースより実際に近い遅延に基づいた動作を実現する手法が提案されている. 我々は動的なばらつき対策手法としてのタイミング・フォールト検出を, 二相ラッチのクロッキング方式に組み合わせることによって実現される, 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を提案する. 本手法によって, 動作時にステージ間で回路遅延を融通し, 実効遅延に近い速度で動作させることが可能になる. 本稿では, 通常の回路を提案手法を適用した回路に変換するツールを実装する.

キーワード：ばらつき, TF 検出, Razor, 2 相ラッチ, タイムボローイング

Applying method of A Clocking Scheme Enabling Dynamic Time Borrowing

Soichiro Hirohata[1]　Kanbara Taro[2]　Shuji Yoshida[1]　Naruki Kurata[1]　Masahiro Goshima[1]
Shuichi Sakai[1]

Abstract: Variations in circuit delay increase due to the miniaturization of semiconductor processes, is becoming a serious problem in the circuit design. Variation is going to increase design method based on the worst case the conventional too pessimistic. Therefore, a method to realize the operation based on the delay realistic than worst case has been proposed. We propose a clocking system that allows the dynamic time borrowing which is achieved by combining two phase latch and a timing fault detection measures as a means of dynamic variation. By this method, it is possible to interchange the circuit delay between the operation stage to operate at a speed close to the effective delay. In this paper, we have implemented a tool to convert the circuit applying the proposed method the normal circuit.

Keywords: variation, TF detection, Razor, two phase latch, time borrowing

1. はじめに

半導体プロセスの微細化に伴って，素子遅延の**ばらつき**が大きな問題となりつつある．ここで特に問題とされているのは，チップ間に跨るシステマティックなばらつきではなく，チップ内のランダムなばらつきである．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ずる本質的な問題であり，原理的に避けえない．

ばらつきが増大していくと，従来の**ワースト値**に基づいた設計手法は悲観的になりすぎる．微細化が進むにつれて，ばらつきの増大により，平均値とワースト値の差は広がっていく．その結果，LSI の設計上の動作速度が向上しなくなってしまうことも考えられる．

そのため，ワースト・ケースより実際に近い遅延に基づいた動作を実現する手法が提案されている．設計段階において遅延のばらつきを統計的に扱う **SSTA**（Statistic Static

[1] 東京大学 大学院 情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
[2] 東京大学 工学部 電子情報工学科
EEIC Engineering Department, The University of Tokyo

Timing Analysis : 統計的静的タイミング解析）などもその一例である．SSTA によれば，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．

動的タイミング・フォールト検出・回復

SSTA のように，設計時に用いられる手法は，静的な方法ということができる．それに対して，動作時にタイミング・フォールトを検出し回復する手法は，動的な方法ということができる．

タイミング・フォールト（以下，**TF** と略す）は，遅延の動的な変化によって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．想定した動作条件内のワースト・ケースでも動作するように設計するのがワースト・ケース設計であるので，そのように設計・製造された LSI では，原則 TF は発生しない．実際に TF が起こるのは，想定した動作条件を外れた場合，例えば，冷却ファンや温度センサの故障による熱暴走などに限られる．

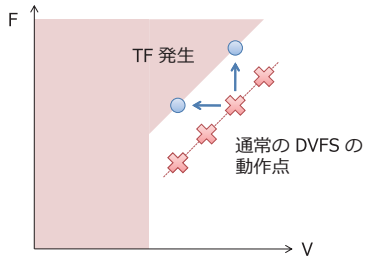

図 1 Razor と DVFS の組み合わせによる V/F の改善

Razor [1] は，TF を検出する機能を持つ．このような回路に **DVFS** (Dynamic Voltage and Frequency Scaling) を組み合わせると，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．**図 1** にその様子を示す．**V**（Voltage : 電源電圧）を下げる，または，**F**（Frequency : 動作周波数）を上げると，回路はいずれ TF を生じ，検出される．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．後は，TF が頻発しないように V–F を調整すればよい．

既存手法の限界

このような TF 検出手法はしかし，実際には，プロセスばらつきに対する直接的な解法にはなっていない．

クリティカル・パスが活性化される確率が 1/100 程度である [2] とすると，クリティカル・パスの遅延より V–F を改善することは現実的ではない．クリティカル・パスの遅延以上に V–F を改善すると，100 サイクルに 1 回は TF を生じ，回復のペナルティを被るからである．

ここで，クリティカル・パスの遅延にはプロセスばらつきの影響が含まれていることに注意されたい．チップ内の各クリティカル・パスの遅延はランダムばらつきにより増減するが，チップの V–F は最も増大した遅延によって決まるのである．

結局，TF 検出手法の効果とは，DVFS のマージンを削減することということができる．

本稿の提案

大数の法則が示すように，あるパスを構成するゲート段数が増加していくと，パスの遅延は構成する個々のゲートのティピカル遅延の総和に近づく．すなわち，**パスが十分に多段であれば，ばらつきの影響は無視できるようになる**のである．

本稿で提案するのは，端的に言えば，TF 検出と二相ラッチを組み合わせたクロッキングの方式である．このことにより，**動的タイム・ボローイング**が可能になる．後で詳しく述べるが，従来からある二相ラッチ方式で可能になるタイム・ボローイングは，言わば**静的タイム・ボローイング**と呼べるもので，設計時にステージ間で回路遅延を融通する．融通される回路遅延は，ワースト遅延である．それに対して，本手法で可能になる動的タイム・ボローイングは，動作時に，ステージ間で回路遅延を融通することができる．しかも，融通される遅延は，ワースト遅延ではなく，実効遅延である．

動的タイム・ボローイングの結果，複数ステージ間に渡る多段のパスが形成され，プロセスばらつきの影響が軽減される．さらに，ワースト遅延ではなく，ワースト遅延より大幅に短い実効遅延に基づく動作が可能となる．その最大動作周波数は，TF の検出限界によって決まり，それは従来のクロッキング方式の 2 倍になる．

以下，2 章では，今回特に考慮するばらつきである入力ばらつきについて取り上げ，実際の回路遅延は実効遅延で決定されることを示し，さらに様々な既存のクロッキング方式のタイミング制約を述べたうえで，そのばらつき耐性について議論を進める．3 章で提案手法の構成・動作を示す．4 章では，提案手法の適用を自動化する変換ツールの実装を説明する．

2. 入力ばらつきと既存のクロッキング方式

本章では，入力ばらつきと既存のクロッキング方式について述べる．

2.1 t-diagram と 入力ばらつき

図 2（上）の回路において，信号が伝わる様子を同図（下）に示す．このチャートを我々は，**タイミング・ダイアグラム**と呼んでいる．以下，t-diagram と表記する．通常のタイム・チャートが論理値-時間の 2 次元を持つに対して，t-diagram は時間-空間の 2 次元を持つ．

通常のタイム・チャートでは，右方向が時間を，上下方向が論理値を表す．タイム・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のは回

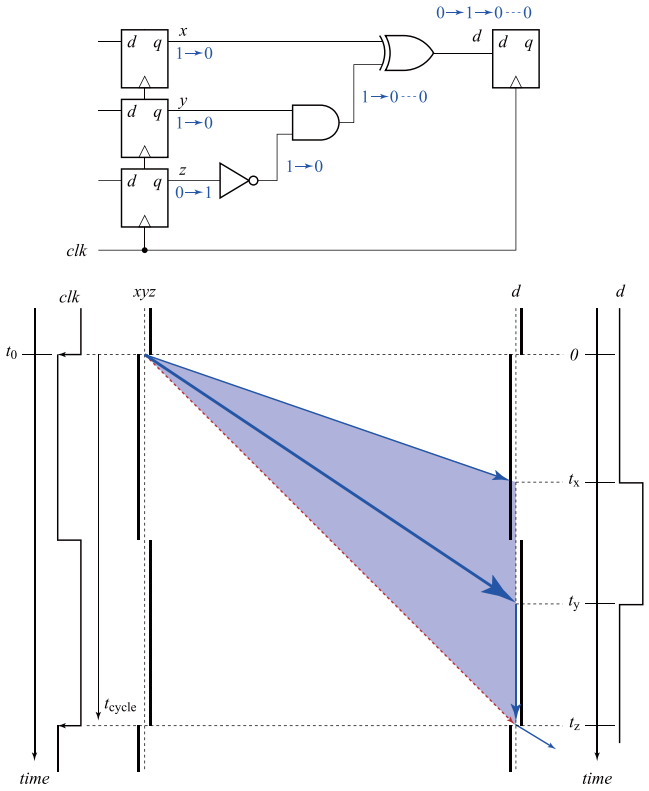

図 2　複数のパスを持つ回路

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．図 2（上）に示す回路で，時刻 $t = 0$ に 3 つの FF の出力 (x, y, z) が $(1, 1, 0)$ から $(0, 0, 1)$ に遷移したとする．x，y，z から d に至るパスの遅延をそれぞれ t_x，t_y，t_z とすると，ロジックの出力 d は，時刻 t_x，t_y において $0 \to 1 \to 0$ と遷移する．z から d に至るパスの信号は，y から d に至るパスの信号によって変化がマスクされるため，時刻 t_z には出力は変化しないことに注意されたい．

同図の右端にある波形が，d における通常のタイム・チャート（を右に 90° 回転したもの）である．同図のように t-diagram では，ロジックの入力において入力が変化した時刻から，出力において出力が変化した時刻までを直線矢印で結ぶことによって，信号の伝わる様子を表すことができる．

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．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．

実効遅延

前述したように，z から d に至るパスの信号は，出力 d に影響を与えない．実際にパスを通ったシグナルがロジックの出力に影響を与えたことを，そのパスが**活性化**したと言う．t-diagram では活性化されたパスを実線で表す．反対に，活性化パスにより，変化が**マスク**され，ロジックの出力に影響を及ぼさないパスは点線で表す．

あるステージにおいて最後に活性化されたパスの遅延を，このステージの**実効遅延**と呼ぶことにする．

ロジックのパスは無数に存在するが，すべてのパスを伝わる信号が出力に影響を及ぼすわけではない．t-diagram では実効遅延を決めるパスを太実線で表す．図 2 の場合，時刻 t_z においてクリティカル・パスを通った信号が到達しているが，活性化パスによって変化がマスクされているため，ロジックの出力 d は変化しない．この場合，実効遅延は t_y となる．

各ステージへの入力と 1 サイクル前の入力によって出力の変化の仕方は様々であり、どのパスが最後に活性化されるかは各サイクルごとに異なる．つまり実効遅延は入力によっても変化する．これを**入力ばらつき**と呼ぶ．特に，ロジックの出力が直前のサイクルと同じで，1 度も変化しなかった場合には，実効遅延は 0 となることに注意されたい．

2.2　既存のクロッキング方式

同期式順序回路を構成する方法を**クロッキング方式**という．また，ロジックのパスの遅延がどこまで許容できるかの制約を**タイミング制約**と呼び，これを満たすように設計しなければ，回路が正しく動作しない．本章ではさまざまなクロッキング方式のタイミング制約を示し，そのばらつき耐性について議論を進める．

2.2.1　単相 フリップ・フロップ

図 3 左が，単相 FF 方式の t-diagram である．マスタ-スレーブ型の FF は逆相で動くラッチを 2 つ組み合わせる構造をとる．

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表している．信号の線がこの実線に沿って伝う様子は，その間ラッチが値を保持していることを表す．エッジ・トリガ動作は，マスタ–スレーブを互い違いに記述することで生じる隙間からシグナルが伝播する様子で表すことができる．

クロックの立ち上がりまでに信号が間に合っていればよいので，最大遅延制約は 1cycle/1stage となる．

2.2.2　二相ラッチ

図 3 右が，二相ラッチ方式の 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．

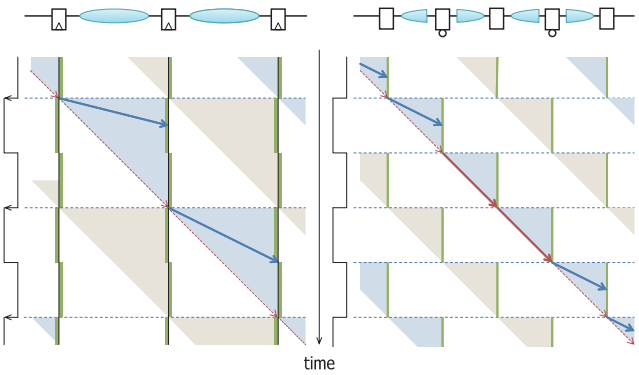

図 3 単相 FF(左) と二相ラッチ（右）の t-diagram

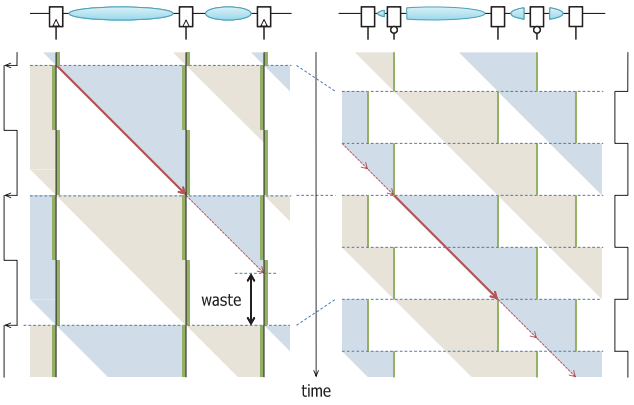

図 4 静的タイム・ボローイング

この形式はクロックスキューに対しても高い耐性もあることが知られている [3].

2.2.3 静的タイム・ボローイング

図 4 はステージ間の遅延に偏りがある場合の単相 FF 方式（左）と二相ラッチ方式（右）の t-diagram である.

単相 FF 方式では常にラッチが閉じている状態のため，仮にクロックの立ち上がりより前にシグナルが到達していても，シグナル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ているため，ステージごとに時間を融通できない．そのため，遅延が大きいステージによってワースト遅延が定まるため，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，単相 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．後述する提案手法の**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 0.5cycle/0.5stage, 最大遅延制約は 1cycle/0.5stage となる.

2.2.4 Razor FF

図 5 右は Razor FF の t-diagram である.

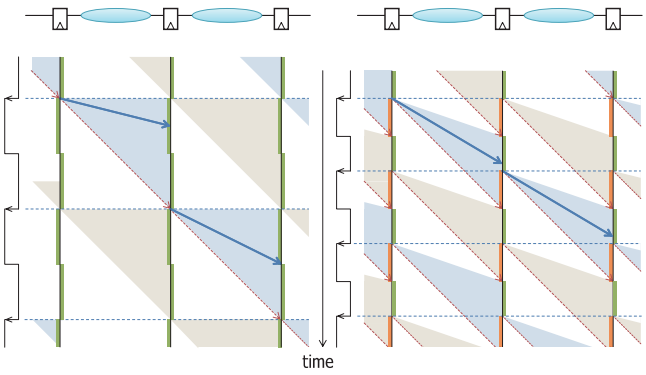

図 5 単相 FF（左） と Razor FF（右）の t-diagram

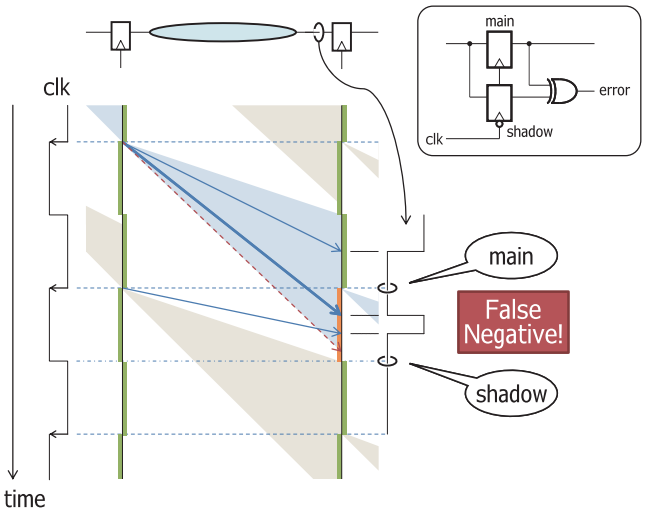

図 6 Razor FF の回路構成とショート・パス問題

Razor FF は通常の FF(Main FF) に，Shadow Latch が加えられている．Shadow Latch には，Main FF よりも遅れたクロックが供給されていて，サンプリング・タイミングが遅くなっている．図 5 では，0.5cycle 遅らせたクロックを Shadow Latch に供給している．このため，TF が発生して Main FF のサンプリング・タイミングまでにクリティカル・パスのシグナルが到達しなくても，Shadow Latch はクリティカル・パスの値をサンプリングすることができる．Main FF と Shadow Latch の値を比較することで，TF を検出する.

2.2.5 Razor FF のショート・パス問題

クリティカル・パスのおおむね半分以下の遅延を持つパスを**ショート・パス**と呼ぶ．セットアップ/ホールド・タイム違反など，ショート・パスの活性化が原因でロジックのタイミング制約が満たされない問題を**ショート・パス問題**と呼ぶ.

図 6 は Razor FF のショート・パス問題を図示した t-diagram である．Razor FF は，Main FF と Shadow Latch の値を比較することで TF を検出するが，正しい値をサンプリングするためには，ロジックのショート・パスを通ったシグナルが Shadow Latch のサンプリング・タイミング

よりも後に到達するように設計しなくてはならない．さもないと，次のサイクルでショート・パスを通ったシグナルが前サイクルの遅れたシグナルと混ざり，Shadow Latch のサンプリング・タイミングで Shadow Latch が正しい値をサンプリングできず，エラー信号が正しく出力できない．

図 6 の場合では，Shadow Latch のサンプリングは 0.5cycle 遅れて行われているので，ロジックの最小遅延が 0.5cycle 以上になるように，Shadow Latch に至るパス の遅延を 0.5cycle 以上にするなどの細工が必要である．このために，例えばショート・パスに遅延素子を挿入し，遅延を伸ばす方法がある．

Razor FF の遅延制約は，検出ウィンドウの割合を α とすると，最大遅延制約は $(1+\alpha)$cycles/1stage となり，TF 検出制約は最大 1cycle/1stage となる．

3. 提案手法

本章では，二相ラッチ方式と TF 検出を組み合わせたクロッキング方式を提案する．これにより，**動的タイム・ボローイング**が可能になる．

3.1 回路構成

図 7 は提案手法の回路構成である．図 7 上は二相ラッチの回路の概略図である．ロジックのショート・パスとクリティカル・パスがとあるゲート（図中○印）で合流した後，パイプライン・ラッチに接続されている．

図 7 下は提案手法の回路の概略図である．TF 検出のために，各パイプライン・ラッチに Razor-like な処置を施す．逆相で動作する Shadow Latch と比較器としての XOR ゲートを追加する．ここでは便宜上，Razor Latch と呼ぶことにする．

2.2.5 で述べた Razor のショート・パス問題が起きないよう，ロジックに遅延を挿入する．TF 検出時は Shadow Latch が開き，Main Latch が閉じている状態であり，Main Latch は前サイクルの値を保持しているため，次サイクルのショート・パスの活性化の影響を受けない．このことから，ショート・パスとクリティカル・パスの合流するゲートを二重化し，Shadow Latch に至るショート・パスにのみ遅延を挿入する．Main Latch に至るショート・パスには遅延が挿入されていないので，ロジックの遅延分布がクリティカル・パスの遅延の方に偏る心配もない．

3.2 動的タイム・ボローイング

図 8 は，二相ラッチ方式と提案手法の t-diagram を比較したものである．図中実線は各ステージにおいて遅延の同じパスが活性化していることを示している．

最近の商用のプロセッサでは，ステージ間のクリティカル・パスの遅延は均等になるように作られているため，2.2.3 で述べた静的タイム・ボローイングの効果は実際には

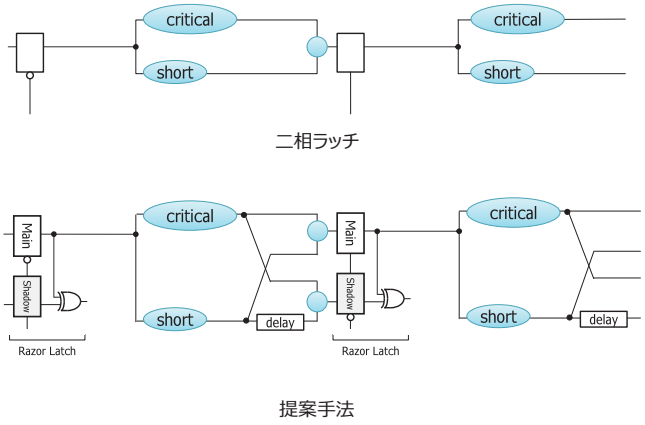

図 7 提案手法の回路構成

限定的なものであると言える．

通常の二相ラッチ方式は TF 検出が備わっていないために，ラッチの開く瞬間の部分でワーストを定めねばならない．そのため，ロジック上の全遅延の存在領域は図の網掛けした部分に限られ，実際には静的タイム・ボローイングを可能にしていたラッチの空いている領域をうまく利用できていない．結果，FF 方式と同様ステージ間の時間を融通できていないこととなる．

提案手法では TF 検出により，ラッチの開く瞬間ではなく，検出限界までワーストを定めることができ，ロジック上の全遅延の存在領域をラッチの空いている区間にも広げることが可能となる．

これにより，動作周波数が向上しているにもかかわらず，実効遅延を累積させて回路を動作させることが可能となる．実行遅延を融通させるこの時間の貸し借りのことを，動的タイム・ボローイングと呼ぶ．

t-diagram 上では実線がつながってステージ間を伝播する様子で動的タイム・ボローイングの効果を表すことができる．

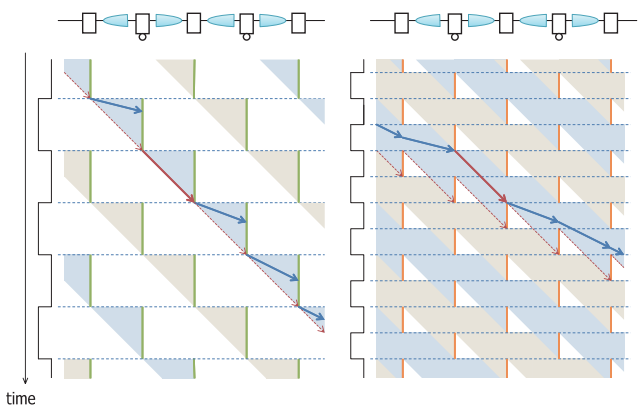

図 8 二相ラッチ（左）と提案手法（右）の t-diagram

3.3 Razor と提案手法の比較

動的タイム・ボローイングの効果は Razor と比較するこ

とで，さらに明確なものになる．**図 9** は Razor と提案手法の t-diagram である．

Razor は FF 方式のタイミング・フォールト検出回路であるため，タイム・ボローイングができず，検出ウィンドウにかかるパスが活性化した場合，その時点で必ずタイミング・フォールトを起こしてしまう．タイミング・フォールトが検出されるごとに命令再実行などの回復処理が行われるため，その回復オーバーヘッドは無視できないものとなる．

提案手法では，遅延の大きいパスが連続して活性化した時にのみ TF 検出となるように設計されている．動的タイム・ボローイングにより，あるステージでクリティカル・パスのような遅延の大きいパスが活性化したとしても，その前後のステージを遅延の小さいパスで通過させることでタイミング・フォールトの発生を抑えることができる．

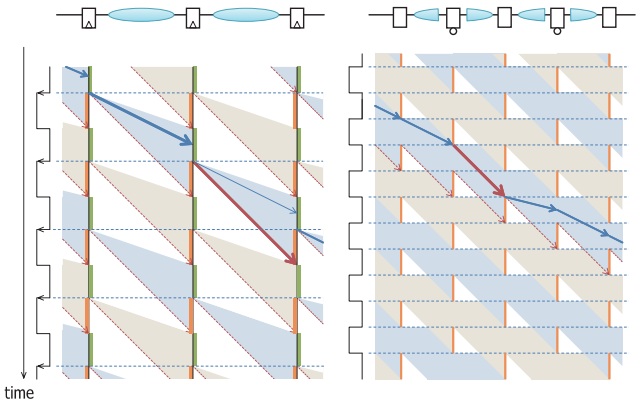

図 9 Razor（左）と提案手法（右）の t-diagram

3.4 既存のクロッキング方式との比較

表 1 は，単相 FF，二相ラッチ，Razor FF，そして提案手法のタイミング制約をまとめたものである．単相 FF 方式の動作周波数を 1τ とし，各ステージのクリティカル・パスの遅延は均等であるとする．

単相 FF 方式は「ワースト遅延のワースト」で動いているといえよう．

二相ラッチ方式は実際には実効遅延の累積で動作させることが可能である．しかし，クリティカル・パスが活性化した際の保障がないため，「ワースト遅延の累積」で設計せざるをえない．

Razor FF は，TF 検出によりワーストが検出限界ギリギリに到達することを許す方式であるから，「一番長いステージのクリティカル・パスの活性化を許すが，検出限界を超えない」ようにワースト値を設計できる．そのため実効遅延での動作が可能となり，Razor FF は「実効遅延のワースト」で動いていると言える．

提案手法は，その TF 検出を二相ラッチ形式に適用したものである．「実効遅延の累積」で回路を動かすことが可

表 1 既存のクロッキング方式との比較

	ワースト・ケース設計	動的 TF 検出・回復	
	最大遅延制約	最大遅延制約	TF 検出制約
単相 FF	ワースト遅延 各ステージ：1τ [普通]	ワースト遅延 各ステージ：$(1+\alpha)\,\tau$	実効遅延 各ステージ：1τ [Razor]
二相ラッチ	ワースト遅延 累積の平均：1τ [静的タイム・ボローイング]	ワースト遅延 各ステージ：2τ	実効遅延 累積の平均：1τ [動的タイム・ボローイング]

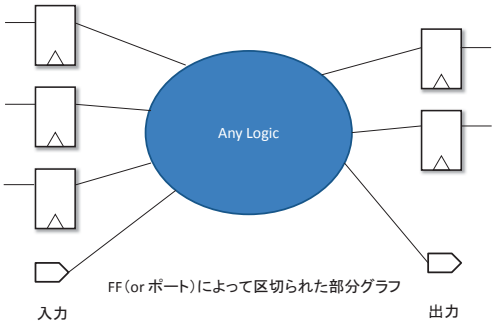

図 10 1 つのステージ

能になり，またステージが二分され，その半分のロジックのクリティカル・パスが検出限界を超えないように設計できるので，理論上は単相 FF の半ロジック分の動作周期，すなわち 2 倍の動作周波数を実現できるのである．

4. 変換ツール

以前我々は，リプルキャリー・アダーを用いたアップカウンタに対して提案手法を適用した [4]．このような手作業での提案手法の適用を行うためには，回路のすべてのパスの遅延を考慮する必要があり，膨大な作業を行わなければならず，より複雑な回路への適用は困難である．そこで我々は EDIF で記述された回路を提案手法を適用した回路に自動的に変換するツールを作成した．このプログラムは主に 3 つのステップから構成される．最初にステージ分割．次に 2 相ラッチ化．最後に Razor の挿入である．

4.1 ステージ分割

回路の 2 相ラッチ化を行うにあたって，まず回路をステージという単位に分割する必要がある．

4.1.1 ステージとは

我々がステージと呼んでいるものは，**図 10** のように回路の中でのフリップフロップからフリップフロップまでの間のロジックである．回路は**図 11** に示すように，ステージがいくつも繋がって構成されたものであると考えることができる．

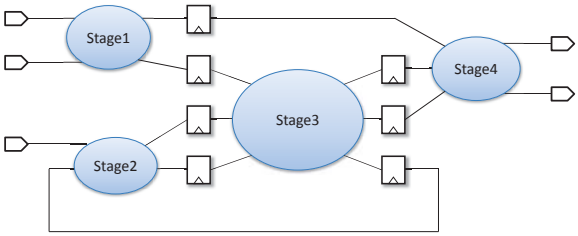

図 11 ステージに分割された回路

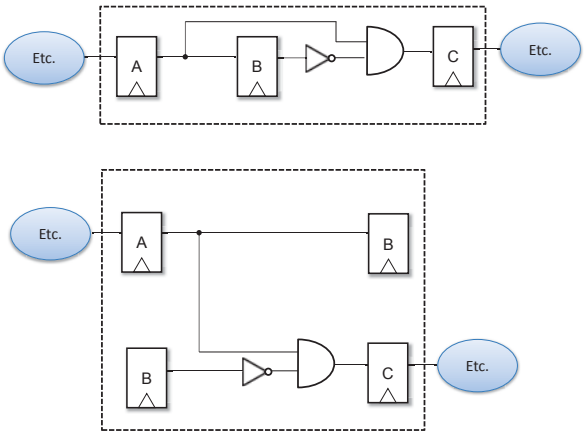

図 12 フォワードエッジのある回路のステージ分割

4.1.2 フォワードエッジ, バックエッジ

ステージを考えるにあたってたとえば図のような回路を考えることにする. ステージを分割する際に一見どう分割するべきかわかりにくいものもある. たとえば**図 12** の上のような回路や**図 13** の上のような回路では A-B 間のステージと B-C 間のステージがあり, それがステージをまたぐ配線によって混ざっているようにみえる. このような回路でも B の FF がステージの入口でも出口でもあると考えることで, A-B 間のステージと B-C 間のステージがあるのではなく, 各図の下のように AB-BC というステージが 1 つあるのだと考えることができる.

4.1.3 分割アルゴリズム

まず任意のゲートを 1 つ選ぶ. このゲートと同じステージにあるゲート全てを探索するためには単純にゲートにつながっている入力, 出力両方のエッジをたどっていき, たどり着いたゲートすべてを同じステージとしてマークする. もしたどり着いたゲートが FF であったときはそれ以上先の探索を打ち切る. ここで検出されなかったゲートは別のステージに属するので, 同様に探索を行い, すべてのゲートの属するステージが決まった所で探索終了となる.

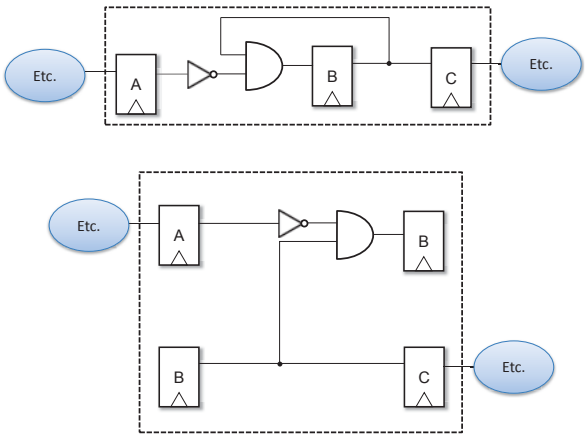

図 13 バックエッジのある回路のステージ分割

4.2 二相ラッチ化

ステージを二相ラッチ化するためには, ステージのロジックの間にラッチを挟むことで, ロジックを二分する必要がある.

まずステージのクリティカル・パスの遅延を求める. クリティカル・パスの計算は最長経路を求めることであり, これには最短経路アルゴリズムを応用できる. パラメータが負の値を取ることのできる最短経路探索アルゴリズムを用いることで各 LUT 素子の入力からのクリティカル・パス遅延を求めることができる.

遅延がクリティカル・パスの遅延の二分の一程度になる場所を探索し, そこにラッチを挟むことで, ロジックを二相ラッチ化する.

4.3 Razor の挿入

TF を検出するために, TF を引き起こす可能性のあるパスに Razor を挿入する必要がある. 提案手法では TF はクリティカル・パスの活性化によって遅延が蓄積することで発生する. したがって TF が発生する可能性のあるパスとは, 遅延が蓄積する可能性のあるパスのことである. 遅延の蓄積はサイクルタイムの半分より遅延の大きいパスが活性化することで起こる. 動作させる目標の動作周期の半分より長い遅延を持つパスを検出し, そこに Razor を挿入する.

4.3.1 2 重化と遅延の挿入

Razor の検出ウィンドウは, Razor の TF 検出が Shadow Latch によって行われるために, 動作クロックによって検出ウィンドウの長さが決まる. Razor の検出ウィンドウは Main Latch が閉じている期間である. ショート・パスが活性化することでこの検出ウィンドウに入るとショート・パス問題が起こる. したがってショート・パス問題が発生する遅延は, サイクルタイムの半分となる. Razor のショート・パス問題を回避するために, ショート・パスに遅延を挿入する必要がある.

しかし, 単純にショート・パスに遅延を挿入すると, ス

テージ間の動的タイムボローイングに於いて遅延の蓄積が大きくなり, TF が発生しやすくなってしまう. これを避けるために, 遅延素子からラッチまでの間にあるゲートを複製し, Main Latch につながるゲートと Shadow Latch につながるゲートを分ける. 我々はこれをゲートの 2 重化と呼んでいる. そして Shadow Latch 側のゲートにだけ遅延を挿入することで, Main Latch 側に遅延が入らなくなる.

しかしこのようにゲートを 2 重化すると, 素子数が増えてしまう. 回路面積のオーバーヘッドを抑えるために, できるだけ 2 重化するゲートを少なくしたい. Main Latch に遅延を入れないためには, ステージ内の遅延より出力側のすべてのゲートを 2 重化する必要がある. したがって遅延素子をステージの出力側にできるだけ近いところに挿入することで回路面積の増加を軽減する事ができる.

4.3.2 アルゴリズム

このような動作を実現するアルゴリズムを以下に示す.

ショート・パス問題を起こさないための最小遅延を D_{th} とする. あらかじめステージのゲートの入力からの最大遅延と最小遅延を求めておく.

ステージの出力側のラッチから入力を辿って探索を始める.

ゲートの最小遅延 + このゲートから探索開始ラッチまでの遅延 が D_{th} より小さいならばこのゲートはショート・パスを含んでいる.

このゲートの出力側に入れることのできる遅延は 探索開始ラッチの最大遅延 - (ゲートの最大遅延 + 探索開始ラッチまでの遅延) である. ショート・パス問題を解消するために必要な遅延がこの上限より小さい時はこのゲートとこのゲートの出力につながっている 2 重化されたゲートの間に遅延を挿入する. ショート・パス問題を解消するために必要な遅延がこの上限を超えているときは遅延は更に前のゲートに入れる. このゲートを 2 重化し, このゲートの入力側のゲートに再帰的にこの処理を繰り返す. この再帰を Razor のラッチすべてに対して行うことで回路の 2 重化と遅延の挿入が完了する.

4.3.3 適用例

それでは**図 14** の回路を例にしてこのアルゴリズムの動作を見ていこう. 簡単のため配線遅延を無視し, 1 つのゲートの遅延を 1 とする. ショート・パス問題を防ぐためには 3 以上の遅延が必要だとする. まず右上のラッチから探索を開始する. a のゲートを見ると, 最小遅延は 2 であり, ショート・パス問題を起こすことがわかる. したがって a より前のどこかに遅延を挿入する必要がある. a の入力を辿って, b を見ると最小遅延は 1 なので, b-a のパスはショート・パスであることがわかる. したがって a を 2 重化し, b と a の 2 重化ゲートの間に遅延を挿入する. 次に a の入力を辿って c のゲートを見ると, 最小遅延が 1 である. しかし c は b と違って最大遅延が 3 であり, a と c の間

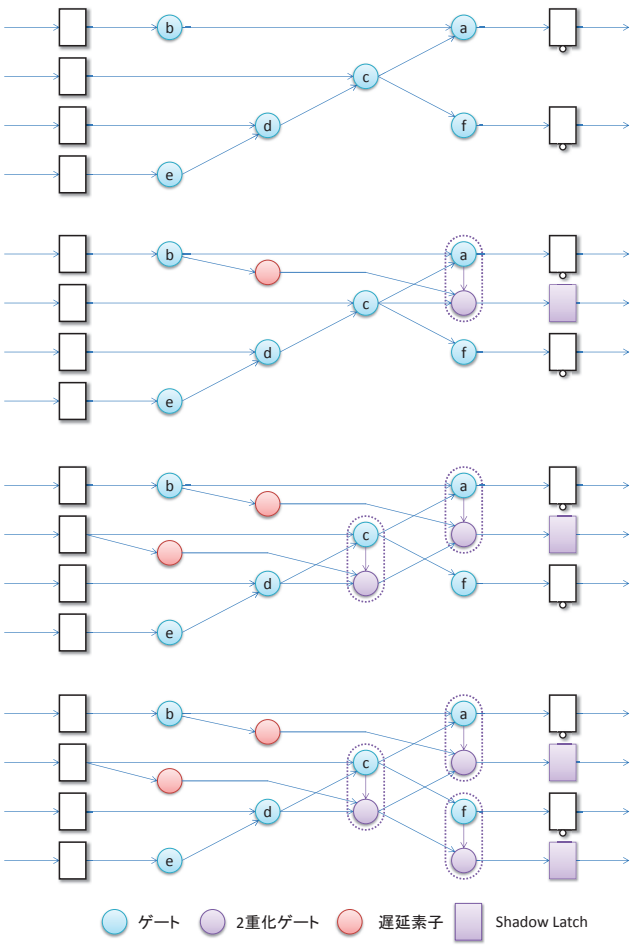

図 14 2 重化と遅延挿入

に遅延を挿入するのでは最大遅延が増えてしまう. そこで更に c の入力を辿っていくとまず入力側のラッチがあるので, これがショート・パスであることがわかる. ラッチと c の間に遅延を入れるのならば最大遅延は増えないので, c を 2 重化して遅延を入れ, c から a の 2 重化ゲートに繋がる配線を c の 2 重化ゲートから a の 2 重化ゲートに繋がる配線につなぎ替える. 次に c の入力を辿って d を見ると最小遅延 1 なので, このパスはショート・パス問題を起こさないため, これ以上先の e を見に行くことはしない. 右上のラッチについての探索はここで終わり, 右下のラッチを見ていく. f の最小遅延は 2 であり, f の入力の c の最小遅延は 1 である. よって c の前に遅延を入れる必要があるが, c はすでに 2 重化されている. c の 2 重化ゲートをもう一つ作るのは無駄なので, すでにある 2 重化ノードをそのまま f の 2 重化ノードとつなげる. 以上でこのステージの 2 重化と遅延の挿入が完了した.

5. おわりに

本稿では, 二相ラッチと Razor を組み合わせることにより動的タイム・ボローイングを可能にするクロッキング方式を提案した. ステージ間でワースト遅延ではなく実効遅

延を融通することができ, さらに Razor により, 半ロジックのクリティカル・パス遅延で動作周波数を決定できるため, 従来の単相 FF 方式と比べて最大で 2 倍の動作周波数で動作させることが可能となる. この提案手法を単相 FF 方式の回路に対して自動的に適用するためのツールの実装方法について説明した. 回路をステージという単位で考え, それぞれのステージに対して 2 相ラッチ化と Razor の挿入, SP への遅延挿入を行う手順を説明した.

今後はこのツールを用いて提案手法を適用した FPU やプロセッサなどを実装して評価していく予定である.

謝辞 本論文の研究は, 一部 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」による.

参考文献

[1] 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, *Int'l Symp. on Microarchitecture (MICRO)*, pp. 7–18 (2003).
[2] 喜多貴信：タイミング制約を緩和するクロッキング方式, 東京大学修士論文，pp. 19–24 (2010).
[3] Harris, D.: Skew-tolerant Circuit Design, *Morgan Kaufmann Publishers*, pp. 12–14 (2001).
[4] 吉田宗史， 有馬慧，倉田成己，塩谷亮太，五島正裕，坂井修一：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予備実験，信学技報，CPSY2011-11, Vol. 111, 鹿児島，pp. 13–18 (2011).