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　1–16 (Jan. 2013)

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

吉田 宗史[1,a)]　広畑 壮一郎[1]　倉田 成己[1]　塩谷 亮太[2]　五島 正裕[1]　坂井 修一[1]

受付日 2012年7月4日, 採録日 2012年10月30日

概要：半導体プロセスの微細化にともなう素子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．この問題に対処するため，ワースト・ケースより実際に近い，実効的な遅延に基づいた動作を実現する手法が数多く提案されている．我々はこの中でも，動作時にばらつきに対処する動的な手法としてのタイミング・フォールト検出に着目し，これ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，提案手法を適用した簡単な回路を FPGA に実装し，従来手法との評価も行う．従来の単相 FF 方式と比べて最大 2 倍の動作周波数の向上を達成できる．

キーワード：クロッキング方式，入力ばらつき，実効遅延，タイミング・フォールト，Razor，二相ラッチ，タイム・ボローイング

A Clocking Scheme Enabling Dynamic Time Borrowing

Shuji Yoshida[1,a)]　Soichiro Hirohata[1]　Naruki Kurata[1]　Ryota Shioya[2]
Masahiro Goshima[1]　Shuichi Sakai[1]

Received: July 4, 2012, Accepted: October 30, 2012

Abstract: Recently, increasing random variation of elements, which is caused by reducing the size of semiconductors, has more influences on circuit design. As the random variation increases, a conventional circuit design based on worst-case estimation becomes too pessimistic. In order to cope with this variation problem, some techniques realizing operation based on effective delay, which is much smaller than worst-case estimation, are proposed. We propose a clocking scheme enabling dynamic time borrowing that is realized by means of applying dynamic timing-fault detection in two-phase latch scheme. Our clocking scheme can borrow time in action from neighboring stage and allow the accumulation of delay, which means no faults are reported until the accumulation of delay violates the constraint of timing-fault detection. In this paper, we apply our clocking scheme to a simple circuit and implement this circuit on FPGA and do system evaluation. Comparing with a conventional scheme using 1-phase flip-flops, evaluation shows that our clocking scheme can double the frequency.

Keywords: clocking scheme, input variation, effective delay, timing fault, Razor, 2-phase latch, time borrowing

1. はじめに

半導体プロセスの微細化にともなって，素子遅延のばらつきが大きな問題となりつつある [1]．ここで特に問題とされているのは，チップ間にまたが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

[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, Bunkyo, Tokyo 113–8656, Japan
[2] 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University, Nagoya, Aichi 464–8603, Japan
[a)] s-yoshida@mtl.t.u-tokyo.ac.jp

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る恐れもある．

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 Timing Analysis：統計的静的タイミング解析）[2] 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静的な手法に対し，動作時にタイミング・フォールトを検出し回復する動的な手法がある．

タイミング・フォールト（Timing Fault: **TF**）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．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・ケースの遅延を見積もり，その場合でも TF が発生しないように設計する．したがって，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，たとえば，温度センサの故障による熱暴走を起こした場合などに限られる．

一方で，TF の発生自体は許容し，TF の検出・回復を行う手法が考えられる．2.5 節で詳しく述べる **Razor** [3], [4], [5] は，その代表例である．このような手法と **DVFS**（Dynamic Voltage and Frequency Scaling）[6] を組み合わせると，以下のように，見積りではない，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．図 1 中×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とって 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．そのようにすると，いずれ TF が発生し検出される．図 1 中○印で表される検出直前の V–F が，見積りではない，そのチップのそのとき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．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．

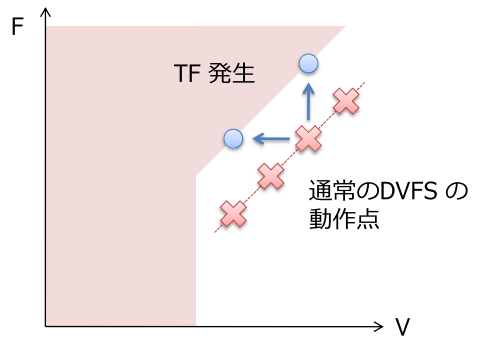

図 1　タイミング・フォールト検出・回復と DVFS の組合せ
Fig. 1　Combination of Timing-fault detection/recovery and DVFS.

実効遅延の平均（typical）に基づく動作

動的に TF を検出・回復する手法は，ロジックの実効的な遅延がサイクル・タイムより長いと TF として検出する．そこでこれらの手法では，さらに，クリティカル・パスの遅延ではなく，それよりはるかに短い**実効遅延**に基づく動作が可能になるのではないかと期待される．実効遅延の正確な定義は，2.1 節で述べる．

実際にロジックの実効遅延の分布を調べると，その平均はクリティカル・パスの遅延の半分程度以下であり，分布は遅延小に大きく偏っている [7]．特に，クリティカル・パスが活性化する確率は 1/100 程度である一方，実効遅延が 0 となる確率は 1/2 程度にもなる．なぜなら，ロジックの出力が変化する確率は 1/2 程度であり，出力が変化しなかった場合には実効遅延は実質 0 となるからである．

したがって，クリティカル・パスのワーストではなく，実効遅延の平均（typical）に基づく動作が可能であるなら，動作周波数 2 倍も夢ではない．

既存手法の限界

しかし実際には，Razor などの既存の手法では，実効遅延に基づいた動作を実現すること，つまり，クリティカル・パスで規定されるサイクル・タイムより短縮することは実際上できない．3.3 節において簡単な実験をもとに詳しく述べるが，チップ内に TF を起こしうるパスは無数に存在するため，サイクル・タイムを短縮すると，パスのうちどれか 1 つで必ず TF が発生しうる．TF が発生するたび，検出し，回復処理を行わなければならず，チップとしてそもそも機能しないのである．

すなわち，既存の TF 検出・回復手法でできることは，実質，DVFS のマージンを削減することであるといえる．

動的タイム・ボローイング

このことからまた，以下のことが示唆される：TF 検出・回復によってサイクル・タイムをクリティカル・パスの遅延より短縮するためには，実効遅延の平均（typical）が小さいだけでは不十分であり，実効遅延の分散が十分に小さい必要がある．

しかし現実には，実効遅延の分散を小さくする手法などおそらくない．そこで本稿で提案するのは，実効遅延の分散を緩和することによって，実効遅延の平均に近いサイ

クル・タイムでの動作を可能とするクロッキング方式である．これは，端的にいえば，TF 検出と二相ラッチを組み合わせたもので，このことにより**動的タイム・ボローイング**が可能になる．3.2 節で詳しく述べるが，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，いわば**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遅延を融通するものである．本提案で可能となる動的タイム・ボローイングとは，実行時に実効遅延がサイクル・タイム以上に伸びてしまった場合，この超過分を次のステージに持ち越すというものである．次のステージの実効遅延が短ければ，この超過分は相殺される．このことにより実効遅延の分散を吸収し，実効遅延の平均に近いサイクル・タイムでの動作が可能となるのである．

TF 検出限界の改善

動的タイム・ボローイングを行っても，サイクル・タイムは実効遅延の分布に従って無制限に短縮できるわけではない．TF を検出する手法には，これ以上削減すると TF が正しく検出できなくなる**検出限界**が存在する．Razor の検出限界は，クリティカル・パス遅延の 2/3 倍程度である．提案手法ではさらに，TF の検出方法を工夫することにより，この検出限界をクリティカル・パス遅延の 1/2 倍へと削減することができる．

提案手法は，主にこれら 2 点により，サイクル・タイム 1/2，すなわち，動作周波数 2 倍を達成することができる．

以下，2 章ではまず，我々が**タイミング・ダイアグラム**と呼ぶ図を提案する．そのうえで，実効遅延の正確な定義を与え，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について述べる．続く 3 章で，提案手法の構成・動作を示す．提案手法は，タイミング・ダイアグラム上で理論的に導き出されたものである．そこで 4 章では，比較的簡単なものではあるが，現実の回路に提案手法を適用した結果，実際に 2 倍の動作周波数を実現できることを示す．5 章では，提案手法の関連研究について述べる．

2. 入力ばらつきと既存のクロッキング方式

どのようなクロックを分配するか，フリップ・フロップ（FF）とラッチのどちらを用いるかといった，同期式順序回路の同期動作を規定する方式を**クロッキング方式**という．本章では，主に既存のクロッキング方式について述べる．

クロッキング方式が正しく動作するための**タイミング制約**は，クロッキング方式によって大きく異なる．本稿では特に，ロジックの遅延の最大値を与える**最大遅延制約**が，クロッキング方式によってどのように変わるかに興味がある．最大遅延制約は，一次近似的にはクロックのサイクル・タイムによって与えられるが，FF/ラッチのセットアップ/ホールド・タイム，クロック–データ遅延，および，クロック・スキューなどを考慮する必要がある [8]．しかし，説明が煩雑になりすぎるため，本稿では，サイクル・タイムのみに着目し，その他の要因については省略することにする．必要であれば，これらを議論に追加することは容易である．

クロッキング方式を理解するうえでは，我々が**タイミング・ダイアグラム**（**t-diagram**）と呼ぶ図を用いると都合がよい．また，特に TF 検出を行うクロッキング方式では，ロジックの**実効遅延**と呼ぶ概念が重要になる．以下，2.1 節で t-diagram と実効遅延について紹介した後，2.2 節以降で，既存のクロッキング方式とその最大遅延制約について述べる．

2.1 タイミング・ダイアグラムと入力ばらつき

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．同図（下）の図を，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路の特定の 1 点の振舞いに限られる．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いく方向を表し，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

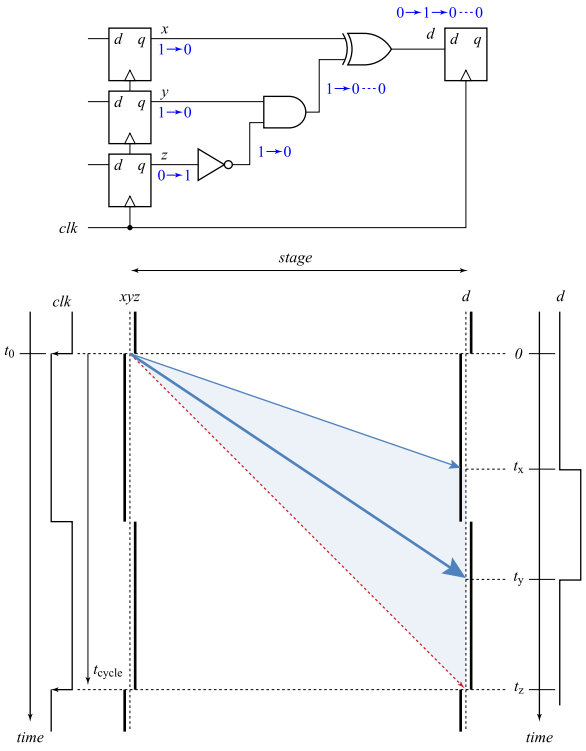

図 2　タイミング・ダイアグラム（t-diagram）と実効遅延
Fig. 2　Timing diagram (t-diagram) and effective delay.

(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 $t = t_x$，t_y において $0 \to 1 \to 0$ と遷移する．z から d に至るパス上を伝わる信号は，y から d に至るパスの信号によってマスクされるため，時刻 $t = t_z$ 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．

パスの活性化とタイミング・ダイアグラム

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，同ロジックの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．

図 2 に示した例では，前述したように，z から d に至るパスを通る信号は途中でマスクされるため，時刻 $t = t_z$ においては出力 d は変化しない．パスを通った信号によって実際に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**したという．

t-diagram では，パスを活性化した信号の伝達を実線矢印で表す．活性化しなかった場合には，途中でマスクされた段階で信号は物理的には消失しているが，仮想的に点線矢印で表すことにする．

入力ばらつきとタイミング・ダイアグラム

t-diagram では，ロジックのクリティカル・パスの遅延に対応する矢印を赤で描くこととし，その角度を 45° と決める．この約束により，あるロジックのクリティカル・パスの遅延は，t-diagram 上のロジックに対応する領域の横幅によって表現することができる．このことは特に，2.4 節でタイム・ボローイングの説明をするうえで重要となる．

実際のロジックでは，ばらつきのため，遅延は連続的に変化する．そのため，矢印の存在範囲は，ロジックの最小遅延の矢印とクリティカル・パスの遅延の赤矢印に上下を挟まれた領域となる．

t-diagram では，網掛けを施してこの領域を示す．

実効遅延

あるロジックにおいて最後に活性化されたパスの遅延を，このロジックの**実効遅延**と呼ぶことにする．図 2 の場合，時刻 $t = t_z$ においてクリティカル・パスを通った信号が到着するはずだが，マスクされたため，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．

時刻 $t = t_y$ において出力 d が変化したときには実効遅延が t_y であることは分からない．時刻 $t = t_z$ において d が変化しなかったことを見て初めて t_y であったことが分かる．このように，実効遅延は事後的に分かることに注意されたい．

t-diagram では実効遅延に対応する矢印を太実線で表す．クリティカル・パスが活性化した場合には，45° の赤矢印をさらに太くして表す．

入力ばらつきと実効遅延

ロジックへの入力の変化の仕方によって出力の変化の仕方も様々であり，どのパスが最後に活性化されるかは毎サイクル異なる．つまり実効遅延は，入力の変化の仕方によって大きくばらつく．このことを**入力ばらつき**と呼ぶ．

入力ばらつきは，他のばらつきに比べて非常に大きい [7]．出力が直前のサイクルから変化しなかった場合には，実効遅延は実質 0 となる．すなわち，入力ばらつきは，0 からクリティカル・パス遅延まで変化するのである．他のばらつきによる遅延の変化がたかだか数十%程度であることを考えると，この変化の度合いは非常に大きいといえる．また，ロジックの出力の変化率は 1/2 程度であることが知られている．すなわち，1/2 程度の高い確率で実効遅延は 0 となることにも注意する必要がある．

以下では，既存のクロッキング方式として，単相 FF，二相ラッチ，および，Razor の 3 方式について紹介し，主にその最大遅延制約を示す．

2.2　単相 FF 方式

図 3 左に，単相クロックとエッジ・トリガ型 FF を組み合わせた単相 FF 方式の t-diagram を表す．

同図は，マスタ–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–スレーブ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（たとえば，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

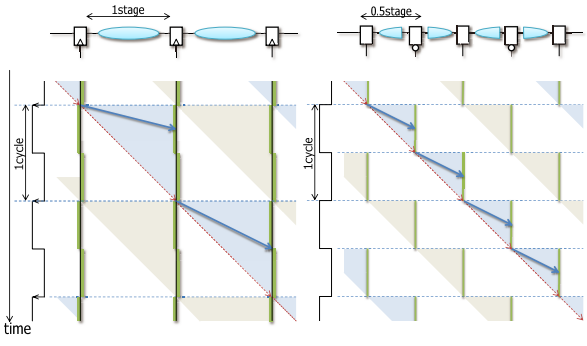

図 3　単相 FF（左）と二相ラッチ（右）の t-diagram

Fig. 3　T-diagram of 1-phase FF scheme (left) and 2-phase latch scheme (right).

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，矢印が存在しうる三角形の領域の網掛けの色を分けることで区別している．

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うえで，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) クリティカル・パスの遅延を表す 45° の赤線をたどって，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．

最大遅延制約

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・エッジで上流側の FF の出力が変化してから，次のクロック・エッジまでに下流側の FF の入力に信号が必ず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，ロジックのクリティカル・パスの遅延が各ステージあたり τ 未満であればよい．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にする．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，実際に最大遅延制約の限界を達成した場合を表している．

なお実際には，FF のクロック–データ遅延やセットアップ・タイム，クロック・スキューのため，ちょうど次のクロック・エッジに到着するようにはできない．また，クロック・スキューや FF のホールド・タイムのため，遅延の最小値に関する制約である**最小遅延制約**も生じる．ただし前述したように，本稿ではこれらについては考慮しない．

2.3　二相ラッチ方式

図 3 右が，二相のクロックとラッチを組み合わせた二相ラッチ方式の t-diagram である．

二相ラッチは，マスタ–スレーブ構造を持つ FF を構成する 2 つのラッチのうちの 1 つを，ロジックの中ほど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，このラッチが二分する形になる．

二相ラッチといっても，実際には，二相のクロックを用いる必要はない．正相と逆相の 2 種類ラッチを用いることで，クロックは（デューティ比 50%の）単相とすることができる．

図 3 右に示すように，t-diagram では，ラッチが開いた瞬間どうしをクリティカル・パスの遅延を表す 45° の赤線で結んでおり，赤線の上の三角形の網掛けの中を信号は通過する．信号は，ラッチが開いた瞬間から出発し，次のラッチが閉じている期間に到着し，このラッチが開くまで待つことになる．

このように，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．この理由については，後で詳しく述べる．

ダブル・パイプライニングとの比較

ここで注意すべきことは，単相 FF で組まれた回路をベースに二相ラッチ化したとしても，回路の処理能力，スループットは変わらないということである．

このことは，同じ単相 FF の回路をベースに，ラッチを移すのではなく，FF を挿入した場合を考えると分かりやすいであろう．各ロジックの中ほどに FF を挿入することは，パイプライン・ピッチを倍にすることで，ダブル・パイプライニングとも呼ばれることもある．この場合，ステージ数が 2 倍となって各ロジックの遅延が半分となった分，クロック周期を 2 倍にして，スループットを 2 倍にすることができる[*1]．

一方，同じ単相 FF の回路を二相ラッチ化した場合には，クロック周期は原則変わらず，スループットも変わらない．これは，図 3 の左右で，どちらも 1 サイクルあたり 1 フェーズしか処理されていないことによって確認できる．

同様に，パイプライン動作における並列実行の単位であるステージは，ベースとなった単相 FF のステージと変わらない[*2]．すなわち，二相ラッチ方式においては，正相のラッチから次の正相のラッチまでが 1 ステージ であり，正相のラッチから次の逆相のラッチまでは 0.5 ステージ と表現することができる．

最大遅延制約

二相ラッチ方式の最大遅延制約は $0.5\tau/0.5$ ステージ であり，単相 FF 方式の $1\tau/1$ ステージ と，率としては変わらない．前述したように単相 FF を二相ラッチ化しても回路のスループットは変わらないのだから，結論としては自明であろう．以下では，この制約を守らないとどのような不具合が起こるかを具体的に説明しよう．

これ以上にサイクル・タイムを短縮すると，クリティカル・パスが連続して活性化した場合に不具合が発生する．**図 4**（右）は，$(0.5\tau \times 1.33)/0.5$ ステージ となるまでサイ

*1　もちろん，ハザードの影響が増大するため，回路の実効的な処理能力が必ず 2 倍になるというわけではない．ステージ数は，アーキテクチャなど上位設計との協調によって決められるべきことで，回路レベルで自由に増減できるものではない．

*2　ダブル・パイプライニングの場合とは対照的に，単相 FF か二相ラッチかは上位設計に影響を及ぼさないので，上位設計とは独立に回路レベルで自由に選択できる．

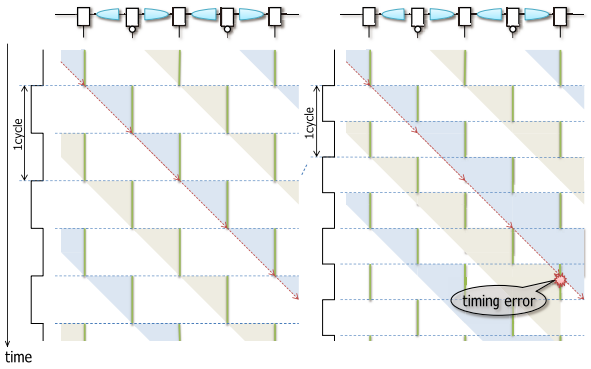

図 4　二相ラッチ方式の最大遅延制約
Fig. 4　Maximum delay constraint of 2-phase latch scheme.

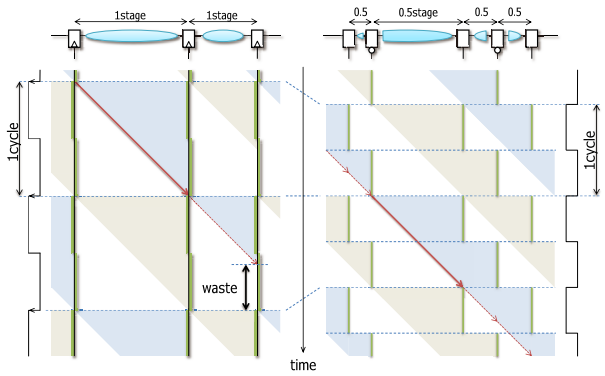

図 5　静的タイム・ボローイング
Fig. 5　Static time borrowing.

クル・タイムを短縮した場合を表している．同図中に爆発のマークで示した点において，以下のように，前述した条件を満たしていない：

(1) 45° の赤い直線が次のフェーズのラッチの閉じた期間にかかっており，次のサイクルに次のステージに処理が引き継がれない．
(2) 次のフェーズの三角形の領域との重なりが生じている．

実際に連続してクリティカル・パスが活性化すると，信号が間に合わないうえ，次のフェーズの信号と「混じっ」てしまう．

このようなことを防ぐためには，最大遅延制約は $0.5\tau/0.5$ ステージ でなければならない．その結果，前述したように，信号は必ず次のラッチが閉じている期間に到着することになり，矢印の存在領域は図 4（左）の網掛けした三角形の内部に限られる．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．

2.4　静的タイム・ボローイング

二相ラッチ方式は，単相 FF 方式に比べ設計がやや煩雑になるが，サイクル・タイムをステージ間で融通することにより各ステージごとの遅延制約を緩和することができる．この効果はタイム・ボローイングとして知られている [8]．本稿では，提案手法における動的タイム・ボローイングと区別して，この効果を**静的タイム・ボローイング**と呼ぶことにする．

図 5 に，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）と二相ラッチ方式（右）の t-diagram を示す．前述したように，各ステージのクリティカル・パスの遅延を表す赤線が 45° であることに注意されたい．同図のように，各ステージのクリティカル・パスの遅延の違いは，t-diagram 上ではステージの横幅の違いによって表される．

単相 FF 方式では，すべてのステージにおいて最大遅延制約 $1\tau/1$ ステージ を満たさなければならない．そのため，クリティカル・パスの遅延のワースト，すなわち，各ステージのクリティカル・パスの遅延の中で最も大きいものによってサイクル・タイムが定まる．その結果，クリティカル・パスの遅延の小さいステージでは，クリティカル・パスが活性化した場合であっても，次のクロック・エッジを待つことになる．この待ち時間を，図中では waste と示した．単相 FF 方式では，この待ち時間を有効に活用する手段はない．

一方，二相ラッチ方式では，ラッチの開いている期間を利用することで，この制約を緩和することができる．図 5（右）に示されているように，正しく動作する条件はラッチが開いている期間にクリティカル・パスの遅延に対応する 45° の赤い直線が通ることである．同図（右）において，前述したクロッキング方式が正しく動作するための 2 つの条件が満たされていることを確認されたい．

同図（右）は，ステージ間の遅延のインバランスが最も大きい場合のもので，最も遅延の大きい 0.5 ステージ に 1τ を費やしている．この 0.5 ステージ において，上流側のラッチが開くと同時に出発した 45° の赤い直線が，次のラッチが閉じる直前に到着している．したがって，これ以上この 0.5 ステージ に時間を割り当てることはできない．一方で，その前後の遅延の小さい 0.5 ステージ にはごく短い時間しか割り当てられていない．すなわち，遅延の長い 0.5 ステージ は短い 0.5 ステージ からサイクル・タイムの一部を借りている（borrow）というわけである．ただし，借りられた時間はけっして返されることがない．

このタイム・ボローイングの結果，ステージ間の遅延がバランスしていない場合に，単相 FF 方式に比べてサイクル・タイムを短縮することができる．二相ラッチ方式の最大遅延制約は，特定の 0.5 ステージ においては $1\tau/0.5$ ステージ となり，単相 FF 方式の $1\tau/1$ ステージ に対して 2 倍となる．ただし，すべてのステージにおける平均の遅延制約は，前述したように，$0.5\tau/0.5$ ステージ である．

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・スキューに対する耐性に効果があり [9]，実際にはスキュー耐性のために採用されることが多いようである．

2.5　Razor

図 6（上）に，Razor FF の回路構成を示す [3]．Razor FF 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 TF として検出する．なお，TF 検出後は，パイプライン・フラッシュなど，アーキテクチャ・レベルの手法によって TF からの回復が行われる [3], [10]．

図 7 は単相 FF 方式と Razor の t-diagram を比較したものである．同図では Main FF と逆相の，すなわち，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．検出ウィンドウの，上端で Main FF が，下端で Shadow Latch が信号のサンプリングを行い，その値を比較する．したがって，検出ウィンドウに実効遅延に対応する太矢印が到着していれば，TF となる可能性がある．ただし，偶数回変化して元に戻った場合には，TF とならない．

Razor のように TF 検出を行うクロッキング方式では，t-diagram 上で正しく動作する 2 つの条件の 1 つ目は，以下のように修正される．2 つ目は変わらない：

(1) クリティカル・パスの遅延を表す赤線でなくても，次のサイクルに次のステージへと至る矢印が描ける．
(2) 矢印が存在しうる三角形の領域が，前後のフェーズのそれと重ならない．

最大遅延制約

単相 FF 方式では，クリティカル・パスの遅延に対応する 45° の赤線が次のクロック・エッジに間に合う必要があるため，最大遅延制約は $1\tau/1$ ステージ となる．

一方，Razor では，クリティカル・パスの遅延に対応する 45° の赤線が検出ウィンドウの下端までに到着すれば，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．

実効遅延に対応する太矢印が検出ウィンドウの上端より先に到着していれば，TF とならない．すなわち，TF とならない最大遅延制約は $1\tau/1$ ステージ となる．

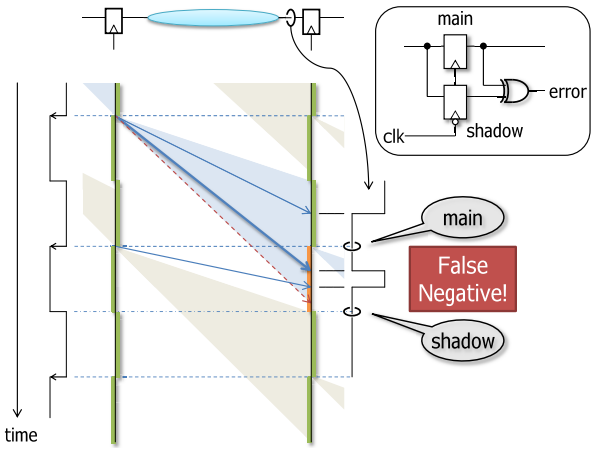

図 6　Razor の回路構成とショート・パス問題
Fig. 6　Circuit structure and short-path problem of Razor.

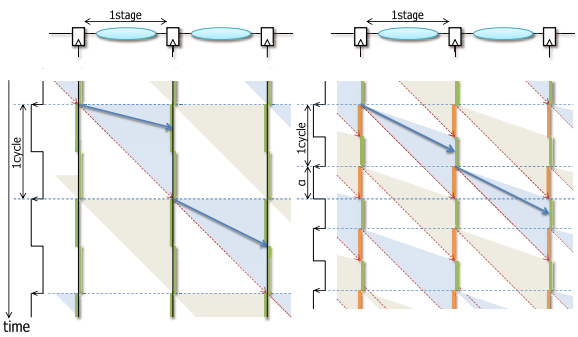

図 7　単相 FF（左）と Razor（右）の t-diagram
Fig. 7　T-diagram of 1-phase FF (left) and Razor (right).

2.6　Razor のショート・パス問題

クロック・スキューに起因するホールド・タイム違反など，ショート・パスが原因で遅延制約が満たされない問題を**ショート・パス問題**と呼ぶ．Razor には，特有のショート・パス問題がある．

図 6 を用いて，Razor のショート・パス問題を説明する．Main FF と Shadow Latch の値を比較することで TF を検出する．Shadow Latch が正しい値をサンプリングするためには，ロジックのショート・パスを通った信号が Shadow Latch のサンプリング・タイミングよりも後に到達しなければならない．そうでなければ，図に示されているように，あるフェーズにおいてショート・パスを通った信号が，前のフェーズの信号と「混ざる」．その結果，Shadow Latch が本来とは異なる値をサンプリングする可能性がある．その結果，誤検出（false positive）となれば問題ないが，検出漏れ（false negative）となると致命的である．

このため Razor は，Razor 特有の最小遅延制約を生じる．図 6 では，Shadow Latch のサンプリングを 0.5τ 遅らせているため，最小遅延制約は $0.5\tau/1$ ステージ となる．前節と同様に，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，最小遅延制約は $\alpha\tau/1$ ステージ となり，単相 FF 方式より $\alpha\tau$ だけ厳しくなる．ショート・パスに遅延素子を挿入するなどして，ロジックの最小遅延を $\alpha\tau$ 以上にする必要がある．

このように Razor には，最大と最小遅延制約の間に，サ

イクル・タイムに対する検出ウィンドウの割合 α を介して，直接的なトレードオフが存在する．

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．以下，3.1 節で提案手法の回路構成を述べ，3.2 節以降で，既存のクロッキング方式との比較を行い，提案手法の特徴や優位性を示す．

3.1 回路構成

図 8 は提案手法の回路構成である．図 8 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・パスとが，あるゲート（図中○印）で合流した後，ラッチに接続されている．

図 8 下は提案手法の回路の概略図である．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．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．ここでは便宜上，Razor Latch と呼ぶことにする．

そして，2.6 節で述べた 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．遅延を挿入するのは Shadow Latch に至るショート・パスにだけ入れればよい．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からである．

このため，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 9 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．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として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．

提案手法では二相ラッチ方式によって本来的には利用可能であ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．

実効遅延を融通するとはどのようなことかについて，**図 10** において説明する．図 10 は提案手法の t-diagram を

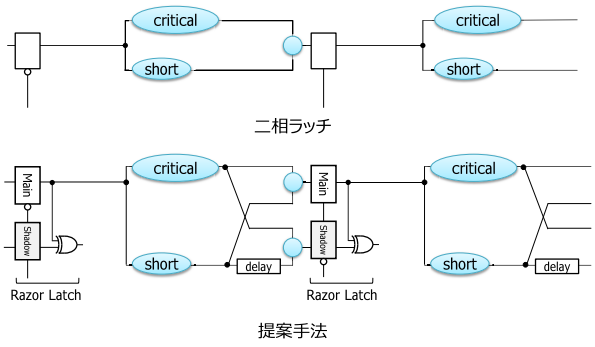

図 8　提案手法の回路構成
Fig. 8　Circuit structure of proposal.

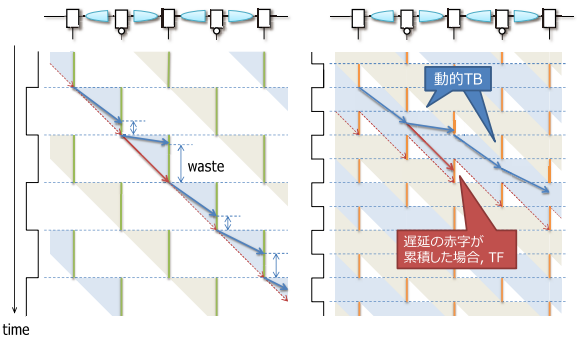

図 9　二相ラッチ方式（左）と提案手法（右）の t-diagram
Fig. 9　T-diagram of 2-phase latch scheme (left) and proposal scheme (right).

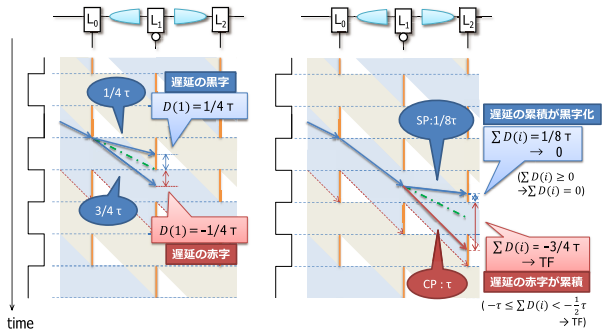

図 10　動的タイム・ボローイング
Fig. 10　Dynamic time borrowing.

拡大したものである．説明のため，各パイプライン・ラッチに L_0，L_1，L_2 と名前を付ける．

以降，ラッチ L_{i-1} と L_i の間のステージにおける実効遅延の値 $E(i)$ を遅延の**支出**，ラッチが閉じている時間 $I(i)$ を遅延の**収入**と定義する．単相 FF 方式や Razor の場合，1 ステージごとに $I(i) = \tau$，二相ラッチ方式や提案手法の場合，0.5 ステージごとに $I(i) = 1/2\tau$ の遅延の収入がある．そして，ステージにおける遅延の**収支**を $D(i) = I(i) - E(i)$ で表し，$D(i) \geq 0$ の場合を遅延の**黒字**，$D(i) < 0$ の場合を遅延の**赤字**と定義する．

今，L_0 と L_1 の間のステージで L_0 の開いた瞬間から信号が伝播し，実効遅延 $E(1) = 3/4\tau$ で L_1 の値が変化した

とする．このとき，このステージにおける遅延の収支 $D(1)$ は，$D(1) = 1/2\tau - 3/4\tau = -1/4\tau$ となり，遅延の赤字が生まれる．仮に，実効遅延 $E(1) = 1/4\tau$ で L_1 の値が変化したとすると，$D(1) = 1/2\tau - 1/4\tau = 1/4\tau$ となり，遅延の黒字が生まれる．同図緑色の点線は，損益分岐（$D(i) = 0$ となる境界）を表している．

$D(1) = -1/4\tau$ の状態で，L_1 と L_2 の間のステージでクリティカル・パス（遅延 τ）が活性化した場合，このステージにおける遅延の収支 $D(2)$ は，$D(2) = 1/2\tau - \tau = -1/2\tau$ となる．このとき，L_0 から L_2 までのステージにおける遅延の収支の**累積**は $\sum D(i) = D(1) + D(2) = -3/4\tau$ となる．提案手法では，このように遅延の赤字が累積し，$-\tau \leq \sum D(i) < -1/2\tau$ となった場合を TF として検出する．

次に，L_1 と L_2 の間のステージで遅延の短いショート・パス（遅延 $1/8\tau$）が活性化した場合を考える．このとき，$D(2) = 1/2\tau - 1/8\tau = 3/8\tau$ で，遅延の黒字が生まれ，遅延の収支の累積も $\sum D(i) = -1/4\tau + 3/8\tau = 1/8\tau$ の黒字となる．

このように提案手法ではラッチの開いている区間を利用することで，遅延の赤字の累積を解消することが可能である．入力ばらつきに着目した，実効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 上における，直線矢印がつながってステージ間を伝播する様子は動的タイム・ボローイングの効果を表しているといえる．

なお，遅延の収支の累積が黒字になった場合は，ラッチ L_n が閉じている状態で値が変化したことになるので，次のステージに信号が伝播するタイミングはラッチの開く瞬間となる．そのため，$\sum D(i) \geq 0$ の場合，次ステージにおいて累積した黒字の分を捨て，$\sum D(i) = 0$ として見る．

最大遅延制約

再度，図 9 に着目する．提案手法では，遅延の赤字が累積し，$\sum D(i) = -\tau$ 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，遅延の赤字の最大値は $D(i) = -1/2\tau$ である．そのため，$\sum D(i) = -1/2\tau$ 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 の検出ウィ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，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 t-diagram 上において階段状となり，サイクル・タイムを詰めることが可能となる．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．

このことから，提案手法の最大遅延制約は，$1\tau/0.5$ ステージと表すことができ，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．

なお，提案手法の t-diagram においてフェーズが「混ざって」いるように見えるが，前節で述べたショート・パスとクリティカル・パスの合流するゲートを二重化し，各フェーズの信号の経路を分けてあるため，実際にはフェーズの信号が「混ざる」ことはない．

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 11** は Razor と提案手法の t-diagram である．

Razor は FF を用いた TF 検出回路であるためタイム・ボローイングができない．前述したように，Razor は 1 ステージごとに $I(i) = \tau$ の遅延の収入ある．しかし，仮に実効遅延の値が $E(i) \leq \tau$ でステージにおける遅延の収支が $D(i) \geq 0$ で黒字であっても，次のクロック・エッジまで待たなければならず，次のステージに遅延の黒字を持ち越すことはできない．そのため，各ステージにおいて独立に遅延の収支 $D(i)$ を見なければならない．

実効遅延の値が $E(i) > \tau$ とサイクル・タイムを超え，遅延の収支が $D(i) < 0$ で赤字となった時点で，必ず TF として検出する．TF が検出されるごとに回復処理が行われるため，その回復オーバヘッドは無視できないものとなる．

一方，提案手法では，ロジック上の全遅延の存在領域をラッチの開いている区間にも広げることで，複数ステージ間にわたる多段のパスが形成される．これにより，各ステージ独立で遅延の収支 $D(i)$ を見るのではなく，全ステージにおける遅延の収支の累積 $\sum D(i)$ を見ることが可能となる．

遅延の赤字の累積が $\sum D(i) < -1/2\tau$ となった場合に TF を検出する．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，その後のステージで遅延の小さい

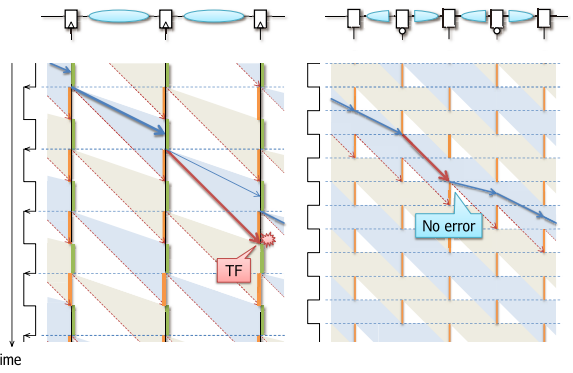

図 11　Razor（左）と提案手法（右）の t-diagram
Fig. 11　T-diagram of Razor (left) and proposal scheme (right).

パスが活性化することで，遅延の赤字の累積を減らし，TF の発生を抑えることができるのである．

Razor の限界と提案手法の効果

2.5 節において，Razor の最大遅延制約はサイクル・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 $\alpha\tau$ だけ改善されると述べた．ところが実際には，DVFS におけるマージンを削減する効果しかない．ここでは簡単な実験を例に，その理由を説明する．

サイコロを 2 個振り，出た目の積により実効遅延が決定するとしよう．このときのクリティカル・パスの遅延は，36（$= 6 \times 6$）である．サイコロの目の和ではなく積としたのは，入力ばらつきが遅延小に偏ること [7] を少しでも再現するためである．この場合，クリティカル・パスの活性化率は 1/36 と文献 [7] で示された値 1/100 よりやや大きく，目の積の期待値は 12.25 とクリティカル・パスの半分の 18（$= 36 \div 2$）よりやや小さい．

Razor では，サイクル・タイムを τ とすると，「目の積が τ を超えたとき」TF となる．TF を発生することなく連続何サイクル動作を継続できるかを測った．その結果を**図 12** の Razor 1，10，100 に示す．1，10，100 は，チップ内で TF を起こしうると目されたパスの数である．Razor では，TF を起こしうるパスには，通常の FF に替えて TF 検出を行う Razor FF を挿入する．したがって，1，10，100 は，通常の FF の代わりに挿入された Razor FF の数と考えてよい．

同図から分かるように，TF を起こしうるパス数が 1 の場合（Razor 1）には，サイクル・タイム $\tau = 30$ 程度まで削減しても，平均して連続 50 サイクル 弱は TF を起こさずに動作することができ，実効遅延に基づいた動作が実現されたといえないことはない．しかし，パスの数が 10（Razor 10），100（Razor 100）の場合，$\tau = 36$ からわずかでも削減すると，たちまち連続動作できるサイクル数はほぼ 0 となってしまう．これは，パスが 100 もあれば，1 つくらいは 6 のゾロ目を振るからである．この結果は，Razor では，クリティカル・パスで規定されるサイクル・タイムより短縮することは実際上できないことを示している．

提案手法の場合は，「出た目の積とサイクル・タイム（τ）をそれぞれ半分にし，その差（遅延の赤字）の累積が $1/2\tau$ を超えたとき」TF となる．仮に，クリティカル・パスの積の目が出たとしても，次のステージで積の値が小さければ，大幅に遅延の借金を解消でき，TF が発生しない．

提案手法に適用した結果が，図 12 の Proposal 1，10，100 である．このように，パスの数が 100 であっても $\tau = 26$ 程度までサイクル・タイムを削減することができ，はるかに Razor を上回る．実際の実効遅延の分布は，サイコロ 2 個の目の積よりも遅延小に偏っているので，さらなる削減が期待される．

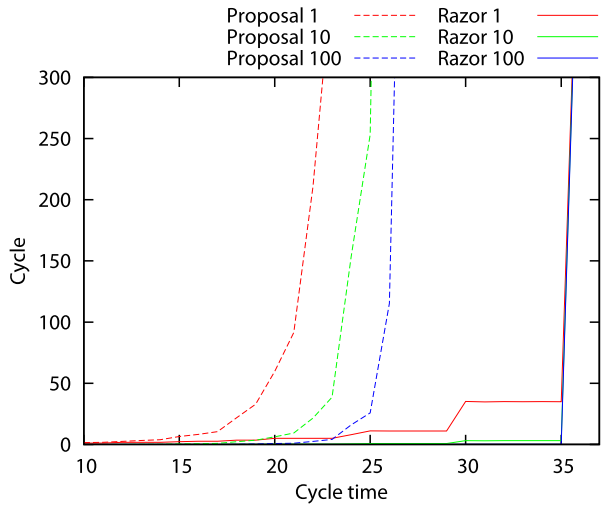

図 12　Razor の限界と提案手法の効果

Fig. 12　The limit of Razor and the effect of proposal scheme.

3.4　既存のクロッキング方式との比較

表 1 は，単相 FF 方式，二相ラッチ方式，Razor，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．各ステージのクリティカル・パスの遅延は均等であるとし，単相 FF 方式の動作周波数を f とする．

単相 FF 方式はステージ間で回路遅延を融通できない方式であり，TF も検出できないため，一番遅延の大きいステージのクリティカル・パスの遅延に合わせて動作周波数が決定する．いわば，「クリティカル・パス遅延のワースト」で動作周波数が決まる方式といえる．

二相ラッチ方式は静的タイム・ボローイングが可能になる方式であり，設計時にステージ間で回路遅延を融通できる．2.3 節で述べたような「クリティカル・パス遅延の累積」で動作周波数を決定できる．ところが各ステージのクリティカル・パスの遅延がバランスしている場合，ラッチの開いている区間を生かすことができないので，動作周波数は単相 FF 方式と変わらず f である．

Razor FF は，TF 検出により，一番遅延の大きいステー

表 1　既存のクロッキング方式との比較

Table 1　Comparison of conventional clocking schemes.

	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	通常 ×ステージ間で時間を融通できない ×動作時に赤字が出たら暴走	Razor ×ステージ間で時間を融通できない ○動作時に赤字が出たら破綻・再建	
	ワースト遅延 各ステージ：1τ	ワースト遅延 各ステージ：(1+α) τ	実効遅延 各ステージ：1τ
二相ラッチ	静的タイム・ボローイング ○設計時にステージ間で時間を融通 ×動作時に赤字が出たら暴走	動的タイム・ボローイング ◎動作時にステージ間で時間を融通 ◎赤字が累積したら破綻・再建	
	ワースト遅延 累積の平均：1τ	ワースト遅延 各ステージ：2τ	実効遅延 累積の平均：1τ

ジのクリティカル・パスの活性化を許すが，TF 検出限界を超えないように設計する方式である．そのため実効遅延での動作が可能となり，Razor は「実効遅延のワースト」で動作周波数が決定する．動作周波数は検出ウィンドウの割合を α とすると，$(1+\alpha)f$ となる．

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．ラッチの開いている区間を利用できるため，「実効遅延の累積」で回路を動かすことが可能になる．また単相 FF 方式の 1 ステージに相当するロジックが二分され，TF 検出により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可能となる．これにより，単相 FF 方式の半ロジック分の動作周期，すなわち $2f$ の動作周波数を実現できる．

4. 評価

3 章で述べた回路構成で正しく動作するか，実際に動作周波数が向上するかの理論の検証のため，提案手法を適用した 64 bit のリプルキャリー・アダ（Ripple Carry Adder: RCA）を用いたアップ・カウンタを FPGA に実装し，動作の確認と各種クロッキング方式との比較を行った．FPGA は Xilinx 社の Virtex-6 XC6VLX760-2ff1760 を用いている．**図 13** にアップ・カウンタのブロック図を示す．ここでのセレクタは初期化の役割を果たしている．

64 bit の RCA を下位 32 bit と上位 32 bit の 2 つに分ける．キャリー・ネットワークにラッチを挟むことにより，ロジックを均等に二分し，二相ラッチ方式を適用する．

そして，TF が起こりうるステージである，RCA の出力をサンプリングするラッチを Razor Latch に置き換える．部分和 s の出力をサンプリングした後のステージは，出力をそのまま次のラッチに流す遅延の小さいステージであるので，Razor Latch に変更する必要はない．

図 14 は 6 bit の RCA を例として Razor のショート・パス問題を回避するための遅延素子を挿入した図である．図中の白抜きの長方形はキャリーを出力する多数決回路を，網掛けした長方形は遅延素子を表す．

RCA では，ci から co に至るキャリー・ネットワークがロジックのクリティカル・パスとなり，入力 a，b からのパスがおおむねショート・パスである．すなわち，RCA のショート・パスとクリティカル・パスは部分和 s を出力する XOR ゲートで合流しているといえる．これを二重化し，Razor Latch の Main Latch と Shadow Latch に至るパスを分ける．

Virtex-6 では，LUT（Look Up Table：真理値表）を 1 つのモジュールとしてインスタンシエートすることができる [11]．XOR ゲートと多数決回路はゲートレベルでの段数に違いがあるが，LUT のパラメータを書き換えることにより，同じ遅延を持つ同一の LUT で表現することが可能である．これにより，パス上の LUT の数でパスの遅延を

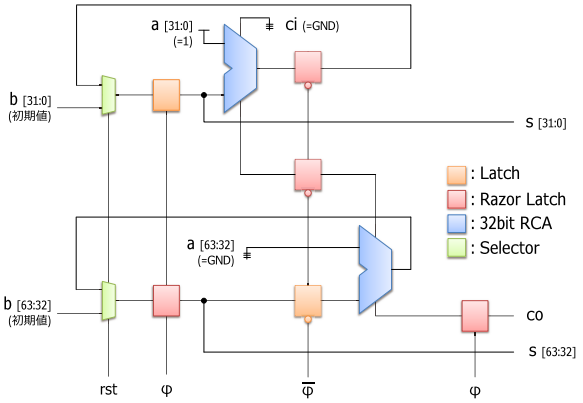

図 13　提案手法を適用した 64 bit アップ・カウンタ
Fig. 13　Applying proposal scheme to 64 bit up-counter.

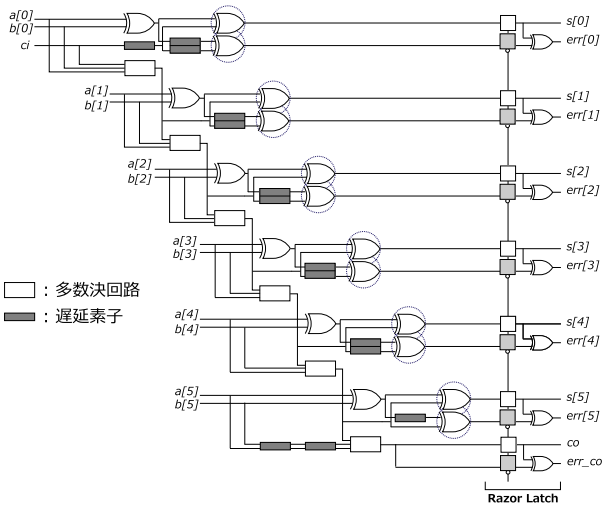

図 14　RCA への適用
Fig. 14　Applying proposal scheme to RCA.

見積もることができる．遅延素子はバッファ回路を LUT で表現したものである．

クリティカル・パス上の LUT の段数は 6 段である．そのため，Shadow Latch に至るショート・パス上の LUT の段数が 3 段以上になるように遅延を挿入する．通常出力側に遅延を挿入すればよいが，上位ビットに行くにつれ，クリティカル・パス自体の遅延が伸びてしまう可能性がある．その際は入力側にも挿入箇所を振り分けることで対処する．

同様の方法で図 13 の 2 つの 32 bit RCA に遅延を挿入し，提案手法の適用を行った．以下にその評価結果を示す．

4.1 動作周波数

64 bit の RCA を用いたアップ・カウンタに各クロッキング方式を適用し，ある特定のビットが変化しなくなる最高動作周波数を測定した．FPGA 基盤上の LED にアップ・カウンタの部分和の出力と TF 検出のエラー信号を表示させ，PLL 設定スイッチで周波数を調整して評価を行った．TF 検出を備えた Razor と提案手法においては，TF の検出限界を最高動作周波数として測定している．**図 15** はその

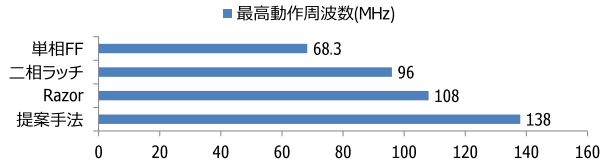

図 15　最高動作周波数の比較

Fig. 15　Comparison of maximum frequency.

結果である．提案手法の最高動作周波数は 138 MHz となり，単相 FF 方式の 2 倍，二相ラッチ方式の 1.4 倍，Razor の 1.3 倍の動作周波数を計測できた．

単相 FF 方式や Razor に対しては，理論値に近い性能向上が見られたが，二相ラッチ方式の動作周波数が予想よりも高めに出た．その理由の 1 つに，今回適用したアップ・カウンタという回路の特殊性が考えられる．

アップ・カウンタではキャリーが伝播するステージの前後では必ず最下位ビットが 0 から 1 に変化するだけで，小さい遅延でステージを通過することができる．そのためワースト遅延以上のパスが仮に活性化しても，ラッチの開いている区間に信号が通りさえすれば，次のラッチのサンプリングには間に合う．いわば，実効遅延の貸し借りが TF 検出という保障がない状況で行われていた結果であると推測される．

4.2　回路面積のオーバヘッド

二相ラッチの回路の総 LUT 数は 292 であるのに対し，提案手法の回路の総 LUT 数は 2,062 にも及ぶ．これは RCA における遅延素子が原因である．今回の回路は提案手法が実際に動作するかを確認するための回路であるため，遅延が $O(n)$ であるが，ロジックを二分しやすい RCA を用いて動作確認・評価を行った．実際には遅延が $O(logn)$ のキャリールックアヘッド・アダ（Carry Look-ahead Adder: CLA）が用いられるため，遅延素子の挿入における回路面積の増加は抑えられると考えられる．**図 16** は 8 bit の CLA に提案手法を適用した際の図である．

一般的に CLA はキャリールックアヘッド・ジェネレータのトゥリー接続によって実現される．ロジックを二分する点は最上位のキャリールックアヘッド・ジェネレータから折り返す部分である．そこで 2 bit のキャリールックアヘッド・ジェネレータを，各桁の g（generate），p（propagate）をまとめる LUT と，g，p から c（carry）を出力する LUT の 2 つに分割してトゥリー接続を開いた構造にする．

トゥリー接続から折り返す部分にラッチを挿入し，クリティカル・パスを通る信号をサンプリングするラッチを Razor Latch に変更する．ラッチの挿入位置によって前半部と後半部が均等に分割されていないのは，ロジックのクリティカル・パスをすべて 1 つのラッチにまとめ，余分な Razor Latch 化を抑えるためである．

そして，ショート・パス問題に対処するため，Razor

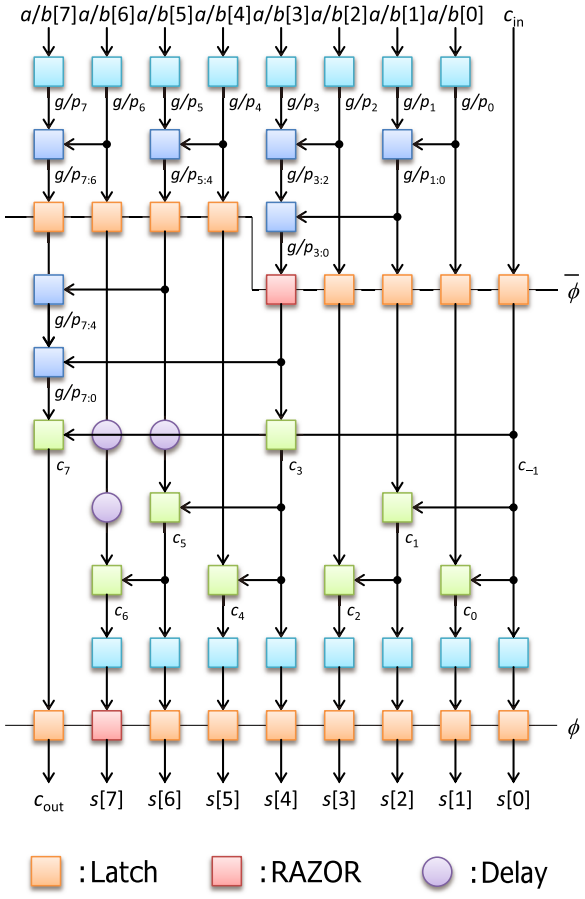

図 16　CLA への適用

Fig. 16　Applying proposal scheme to CLA.

Latch に至るショート・パスに遅延を挿入する．図 16 では，ショート・パスの遅延をクリティカル・パスの遅延に合わせるように挿入している．前半の部分においては，Razor Latch に至るパスがすべてクリティカル・パスと同じ遅延を持つため，遅延を挿入する必要はない．このようにして，遅延の挿入による回路面積の増加は抑えられる．

5.　関連研究

本章では提案手法の関連研究として，**ReCycle** [12]，**TIMBER** [13]，**Bubble Razor** [14] を紹介する．

5.1　ReCycle

ReCycle は，cycle time stealing [15] 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 17** 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．

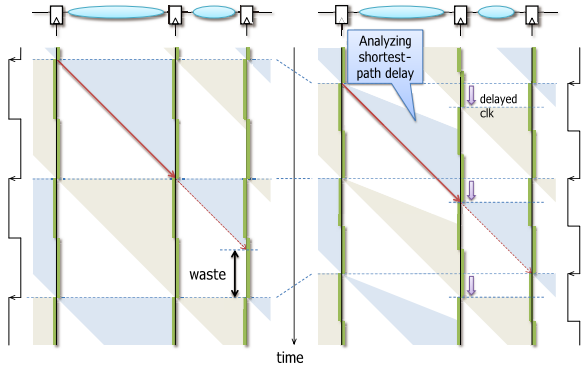

図 17　単相 FF（左）と ReCycle（右）の t-diagram
Fig. 17　T-diagram of 1-phase FF scheme (left) and ReCycle (right).

ReCycle は，遅延の大きいステージのロジックのパス遅延をあらかじめ解析する．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．

5.2　TIMBER

文献 [13] 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きわめて小さい，すなわち，クリティカル・パス遅延を持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 18 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・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，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・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 19 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，

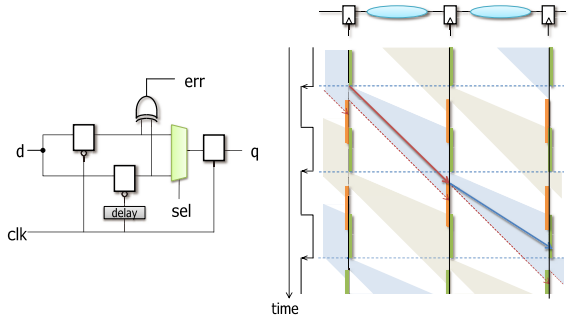

図 18　TIMBER の回路構成と t-diagram
Fig. 18　Circuit structure and t-diagram of TIMBER.

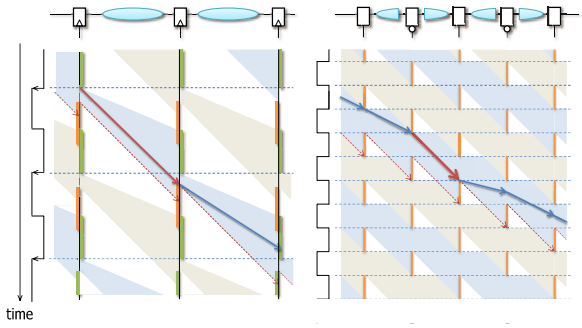

図 19　TIMBER（左）と提案手法（右）の t-diagram
Fig. 19　T-diagram of TIMBER (left) of proposal scheme (right).

2.6 節で述べたショート・パス問題が厳しくなる．動作中においても最小遅延制約が変動するため，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 2 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，期待どおりのクロックが生成されないことが予想される．

5.3　Bubble Razor

Bubble Razor 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 20** 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 20 a）．

検出ウィンドウにかかるパスが活性化した場合（図 20 b-

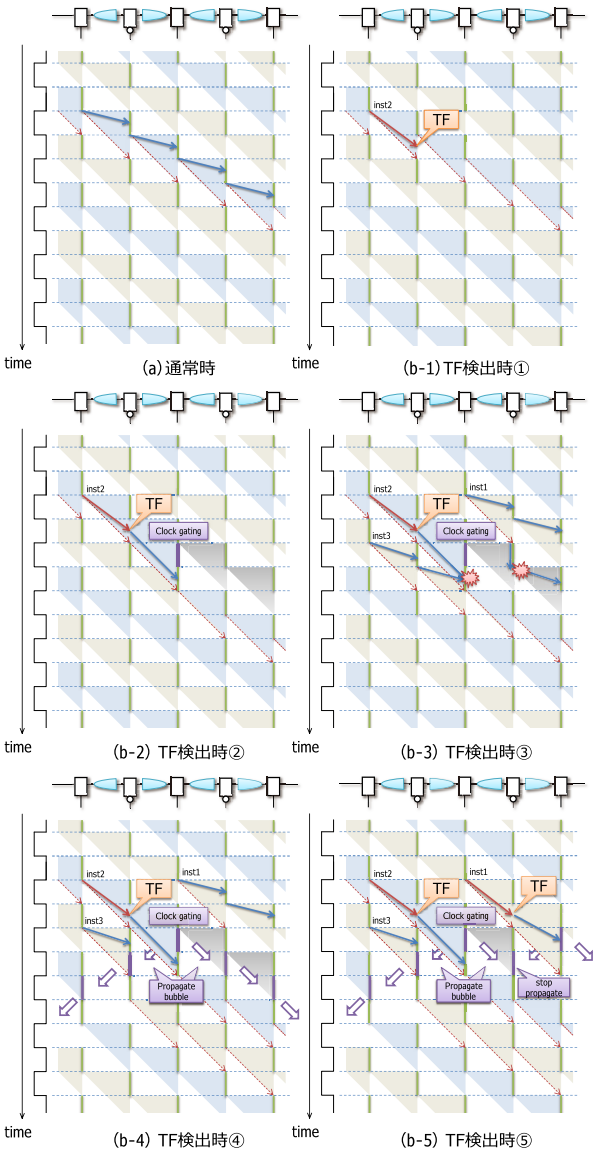

図 20　Bubble Razor の t-diagram
Fig. 20　T-diagram of Bubble Razor.

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 20 b-2）．この閉じている区間を文献 [14] 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．

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という問題が生じる（図 20 b-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 0.5 cycle 遅れて Bubble を伝搬させ続けることにより対処する（図 20 b-4）．

図 20 b-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．

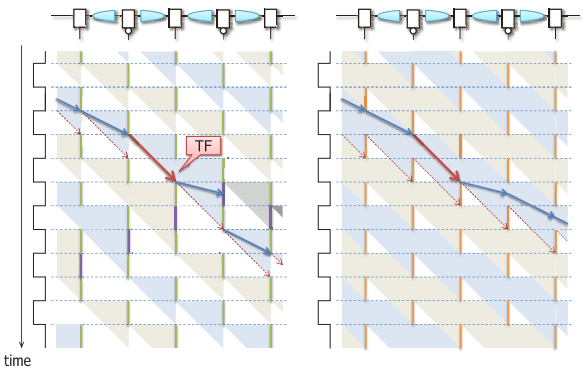

図 21　Bubble Razor（左）と提案手法（右）の t-diagram
Fig. 21　T-diagram of Bubble Razor (left) and proposal scheme (right).

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 21 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ともに同じであるが，Bubble Razor は二相ラッチ方式を採用しているにもかかわらず，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・ボローイングができなくなっている．これにより，3.3 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ないものと予想される．

6.　おわりに

本稿では，二相ラッチと Razor を組み合わせた動的タイム・ボローイングを可能にするクロッキング方式を提案し，簡単な回路に適用することで評価を行った．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 TF 検出により，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，通常の 2 倍の動作周波数で動作させることが可能となる．

現在，より一般的な CLA に提案手法を適用しており，その知見のもと，ロジックの全パスの遅延に基づいて，回路に提案手法を自動で適用するツールを作成中である．その後，この手法も含め，Out-of-Order プロセッサにおける TF からの回復手法 [10] や，NORCS [16] などに代表される，我々の研究室で提案している手法を適用したプロセッサを制作予定である．

謝辞　本稿の研究は，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] 平本俊郎，竹内　潔，西田彰男：MOS トランジスタのスケーリングに伴う特性ばらつき，電子情報通信学会誌，Vol.92, No.6 (2009).

[2] Ashish, S., Dennis, S. and David, B.: *Statistical Analysis and Optimization for VLSI: Timing and Power*, ISBN: 978-0-387-25738-9 (2005).

[3] Ernst, D., Kim, N., Das, S., Pant, S., Pham, T., Rao, R., Ziesler, C., Blaauw, D., Austin, T. and Mudge, T.: Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *Int'l Symp. on Microarchitecture* (*MICRO*), pp.7–18 (2003).

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

[5] 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*), *2010 IEEE International*, pp.284–285 (2010).

[6] 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).

[7] 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).

[8] Bakoglu, H.: *Circuits, interconnections, and packaging for VLSI*, VLSI systems series, Addison-Wesley Pub. Co. (1990).

[9] Harris, D.: *Skew-tolerant Circuit Design*, pp.12–14, Morgan Kaufmann Publishers (2001).

[10] 五島正裕，倉田成己，塩谷亮太，坂井修一：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.53, No.41 (2012).

[11] Xilinx Inc.：Virtex-6 ライブラリ ガイド（HDL 用），pp.190–193 (2009).

[12] Tiwari, A., Sarangi, S.R. and Torrellas, J.: ReCycle: Pipeline adaptation to tolerate process variation, *ISCA*, pp.323–334 (2007).

[13] 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).

[14] 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).

[15] 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).

[16] Shioya, R., Horio, K., Goshima, M. and Sakai, S.: Register Cache System not for Latency Reduction Purpose, *Int'l Symp. on Microarchitecture* (*MICRO-43*), pp.301–312 (2010).

吉田 宗史　（学生会員）

1987 年生．2011 年東京大学工学部電子情報工学科卒業．同年同大学大学院情報理工学系研究科電子情報学専攻修士課程に進学，現在に至る．コンピュータアーキテクチャの研究に従事．2011 年情報処理学会学生奨励賞受賞．

広畑 壮一郎　（学生会員）

1987 年生．2012 年東京大学工学部電子情報工学科卒業．同年同大学大学院情報理工学系研究科電子情報学専攻修士課程に進学，現在に至る．コンピュータアーキテクチャの研究に従事．

倉田 成己　（学生会員）

1987 年生．2010 年東京大学工学部電子情報工学科卒業．2012 年同大学大学院情報理工学系研究科電子情報学専攻修士課程修了．同年同専攻博士課程に進学，現在に至る．プロセッサアーキテクチャの研究に従事．2010 年情報処理学会推奨卒業論文に認定．

塩谷 亮太　（正会員）

1981 年生．2006 年東京大学工学部電子工学科卒業．2008 年同大学大学院情報理工学系研究科電子情報学専攻修士課程修了．2009 年より日本学術振興会特別研究員．2011 年東京大学大学院情報理工学系研究科電子情報学専攻博士課程修了，博士（情報理工学）．同年より名古屋大学大学院工学研究科助教，現在に至る．コンピュータアーキテクチャの研究に従事．2011 年 IEEE Computer Society Japan Chapter Young Author Award 受賞．電子情報通信学会，IEEE 各会員．

五島 正裕　（正会員）

1968 年生．1992 年京都大学工学部情報工学科卒業．1994 年同大学大学院工学研究科情報工学専攻修士課程修了．同年より日本学術振興会特別研究員．1996 年京都大学大学院工学研究科情報工学専攻博士後期課程退学，同年より同大学工学部助手．1998 年同大学大学院情報学研究科助手．博士（情報学）．2005 年東京大学情報理工学系研究科助教授，2007 年同大学同研究科准教授，現在に至る．コンピューティング・システムの研究に従事．著書に「ディジタル回路」．2001 年情報処理学会山下記念研究賞，2002 年同学会論文賞受賞．IEEE 会員．

坂井 修一　（フェロー）

1958 年生．1981 年東京大学理学部情報科学科卒業．1986 年同大学大学院工学系研究科修了，工学博士．電子技術総合研究所，MIT，RWC，筑波大学を経て，1998 年東京大学助教授．2001 年より東京大学大学院情報理工学系研究科教授．計算機システムおよびその応用の研究に従事．特に並列処理アーキテクチャ，相互結合網，最適化コンパイラ，省電力アーキテクチャ，ディペンダブルシステム等の研究を進めている．情報処理学会研究賞（1989），情報処理学会論文賞（1991），IBM 科学賞（1991），市村学術賞（1995），IEEE Outstanding Paper Award（1995），Sun Distinguished Speaker Award（1997），大川出版賞（2012）等受賞．情報処理学会フェロー．電子情報通信学会フェロー．人工知能学会，ACM，IEEE 各会員．日本学術会議連携会員．