情報処理学会論文誌　コンピューティングシステム　Vol.10　No.2　1–12 (Sep. 2017)

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

神保 潮[1,a)]　山田 淳二[2]　五島 正裕[3]

受付日 2017年2月24日, 採録日 2017年6月8日

概要：ワースト・ケースより平均的ケースにおける遅延に基づいた動作を実現する手法の 1 つとして，我々は動的タイム・ボローイングを可能にするクロッキング方式を提案している．このクロッキング方式は，動的なばらつき対策手法である動的タイミング・フォールト検出と二相ラッチによるクロッキング方式の組合せにより実現され，動作時にステージ間で実効的な回路遅延を融通することで，ティピカル・ケースに基づく速度で回路を動作させることが可能になる．本論文では，FPGA 上の 34 bit のリプル・キャリー・アダーを用いたカウンタにこの方式を適用し，通常の単相 FF 方式に対して 1.6 倍の周波数で動作することを確認した．

キーワード：ばらつき，タイミング・エラー，タイム・ボローイング，Razor FF，クロック

Application of Clocking Scheme That Enables Dynamic Time Borrowing

Ushio Jimbo[1,a)]　Junji Yamada[2]　Masahiro Goshima[3]

Received: February 24, 2017, Accepted: June 8, 2017

Abstract: As one of techniques for operation based not on worst but on actual delays, we have proposed a clocking scheme enabling dynamic time borrowing. This clocking scheme is by means of applying dynamic timing-fault detection in a conventional scheme using two-phase latches. This technique realizes dynamic time borrowing between neighboring stages on a circuit for more aggressive operation based on typical delays. In this paper, we applied this scheme to a counter of a 34-bit ripple carry adder on an FPGA. Evaluation results show that this scheme remarks the 1.6 times higher frequency than a conventional one with single-phase FFs.

Keywords: variation, timing error, time borrowing, Razor FF, clock

1. はじめに

チップ内のランダムなばらつきの増大 [1] により，従来のワースト・ケースに基づいた設計ではチップの性能の向上が見込めなくなりつつある．微細化により，遅延の典型値は短縮されている一方で，ばらつきの増大によって分散は大きくなっている．そのため，歩留まりを一定とすると，最悪値は，典型値ほどには短縮されなくなる．こうした傾向が続けば，微細化が進むにつれてティピカル遅延とワースト遅延の差は広がっていき，将来的には，ワースト遅延が短縮され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統計的静的タイミング解析（Statistic Static Timing Analysis: SSTA）[2], [3] もその一例である．SSTA によれば，ワースト・ケースほど

[1] 総合研究大学院大学複合科学研究科
School of Multidisciplinary Sciences, SOKENDAI, Chiyoda, Tokyo 101–8430, Japan
[2] 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
[3] 国立情報学研究所アーキテクチャ科学研究系
Systems Architecture Research Division, NII, Chiyoda, Tokyo 101–8430, Japan
[a)] ushio@nii.ac.jp

悲観的ではない遅延見積りを行うことができる．

タイミング・フォールト検出

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．

回路遅延の動的な変動によって生じる過渡故障を**タイミング・フォールト**（Timing Fault: **TF**）と呼ぶ．ワースト・ケース設計では，ワースト・ケースにおいてもこの TF が発生しないように，十分なマージンをとった電圧やクロック周波数を設定する．TF が生じるのは，サーマル・センサの故障による熱暴走など，想定外の場合に限られる．

ワースト・ケースではなく，ティピカル･ケースの遅延に基づいた動作を実現するため，TF を検出し，そこから回復する手法が提案されている [4], [5], [6], [7]．こうした手法では，ワースト･ケース設計で定められる限界を超えて回路を高い周波数，または低電圧で動作させることができる．DVFS と併用されることで，周波数や電圧は TF 発生による IPC の低下との釣り合いがとれる最適な状態に動的に決定でき，回路の動作環境に応じた周波数や電源電圧の適切な決定が可能になる．

本論文の内容

我々は，より効果的な周波数向上や電圧削減を可能にする手法として，動的タイム･ボローイング（Dynamic Time Borrowing: **DTB**）を可能にするクロッキング方式を提案してきた [8]．本論文では，カウンタに対してこの手法を適用し，FPGA 上に実装し，動作の確認を行い，動作周波数を計測することによって，この手法の基礎的な性能を確認する．

そのためには，通常の単相フリップ･フロップ（**FF**）によるカウンタ回路に対して，(1) 回路の変換と，(2) 回復機構の付与を行う必要がある．以前にも，カウンタにこの手法を適用し FPGA 上に実装し動作周波数を計測したことがあった [8] が，以下の点で不十分であった：

(1) 回路変換は手動であった．
　手動による適用は，カウンタ程度に対してならば可能ではあるが，実用的な回路に対してはほぼ不可能である．

(2) 回復機構を含んでいなかった．
　そのため，TF が検出された瞬間に回路は正常に動作しなくなる．したがって，TF が検出される直前の動作周波数を計測するにとどまっていた．

そこで，本論文では以下を行う：

(1) 開発した自動適用ツール [9] を用いて回路変換を行う．

(2) 回復機構を付加する．
　TF 検出時には，自動的に回復を行い，動作を継続する．したがって，TF をときどき起こしながらも動作可能な周波数を計測することになる．

以下，本論文は次のように構成される．2 章では，まず TF を検出・回復する手法も含めて，既存のクロッキング方式についてまとめる．3 章で提案方式について述べる．4 章では，提案方式のカウンタへの適用について詳細を述べ，5 章で，評価結果を示す．

2. 既存のクロッキング方式

本章では，次章で述べる提案方式をよりよく理解するために，まず既存のクロッキング方式を説明する．2.1 節では，クロッキング方式の理解に便利な**タイミング･ダイアグラム**を導入する．2.2 節以降で，単相 FF，二相ラッチ，そして，Razor [4] について説明する．

2.1 タイミング･ダイアグラムの基礎

図 1 に示すグラフを，我々はタイミング･ダイアグラムと呼んでいる．通常のタイミング･チャートが論理値–時間の関係を表すのに対して，タイミング･ダイアグラムは時間–空間の関係を表す．同図中，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．

実際のロジックには，それぞれ遅延が異なるパスが数多く存在する．ダイアグラムでは，入力の変化によって出力が変化したとき，その信号伝達を，入力が変化した点から出力が変化した点までを（右下がりの）直線矢印で結んで表す．

実効遅延

ロジック中の信号の伝達の仕方は，ロジックの入力の変化の仕方によって異なる．一部の信号の遷移はマスクされるため，一般にすべてのパスが出力の変化に関与するわけではない．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，そのパスは活性化されたという．

ダイアグラムでは，あるサイクルにおいて最後の出力の変化をもたらした信号の伝達を実線矢印で表す．この実線矢印の遅延（図上で縦方向の距離）を，そのサイクルの**実効遅延**と呼ぶ．

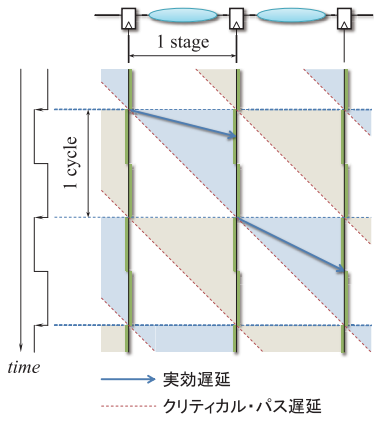

図 1　単相 FF のタイミング･ダイアグラム
Fig. 1　Timing diagram for single-phase FF.

ダイアグラム上で実線矢印が存在可能な範囲は，ロジック内の最小遅延とクリティカル・パス遅延を表す直線に挟まれた三角形の領域となる．ダイアグラムではこの領域を網掛けにより示す．図中の網掛けの 2 色については後述する．

なおダイアグラムでは，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，ダイアグラム上のステージの横幅によって表現することができる．

入力ばらつき

実効遅延という言葉を用いるなら，**入力ばらつき**は，ロジックの入力の変化の仕方に応じて生じる実効遅延のばらつきと定義することができる．

ロジックの出力が一度も変化しなかったとき，実効遅延は 0 と考えられる．すなわち入力ばらつきによって，ロジックの実効遅延は 0 からクリティカル・パス遅延まで変化することになる．ほかの要因によってはロジックの（クリティカル・パス）遅延は数割程度しかばらつかないことを考えると，入力ばらつきは非常に大きいといえる．

2.2　クロッキング方式の表現

次に，図 1 でのクロッキング方式の表現を説明する．

エッジ・トリガ動作

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 の下にある縦実線はラッチが閉じている状態を，縦実線と次の縦実線の間の空白は，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・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．

フェーズ

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．

パイプライン動作においては，一連の処理——典型的には，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．

ダイアグラムでは，あるフェーズの処理と次のフェーズの処理を，矢印が存在しうる領域の網掛けの色を分けることで区別している．

2.3　クロッキング方式の要諦

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．

ダイアグラム上では，以下の 2 つの条件が満たされていればよい：

(1) 実線矢印をたどって，次のサイクルに次のステージへと至ることができる．
(2) 矢印が存在しうる範囲を表す網掛けの領域が，前後のフェーズの，すなわち，色の異なる網掛けの領域と重ならない．

クロッキング方式のタイミング制約は，この 2 条件から導かれる．

次章からは，ダイアグラムを用いてそれぞれのクロッキング方式について説明する．

2.4　単相 FF 方式

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 1（および，**図 2** (a)）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．なお，簡単のため，FF やラッチのセットアップ/ホールド時間やスキューなどは省略しているが，これらを議論に組み込むことは容易である．

通常，クリティカル・パスが活性化される確率は高くない．図 1 のように，実効遅延とクリティカル・パス遅延の差の分だけ，無駄な待ち時間が生じることになる．

2.5　二相ラッチ方式

図 2 (b) に，二相ラッチ方式のダイアグラムを示す．二相ラッチ方式は，単相 FF 方式を基にすると，FF を構成するマスタ，スレーブの 2 つのラッチのうちの 1 つをロジックの中間へと移したものと理解することができる．移されたラッチによって分割されたステージの前半/後半をそれぞれ**半ステージ**と呼ぶことにする．

単にラッチの位置を動かしただけなので，二相ラッチ方式の最大遅延制約は，基本的には，$0.5\tau/1$ 半ステージ となり，単相 FF の $1\tau/1$ ステージ と変わらない[*1]．

[*1] ただし，二相ラッチ方式はクロック・スキューへ耐性を持つ．先述したように，議論の簡単化のためにクロック・スキュー耐性は考慮しない．

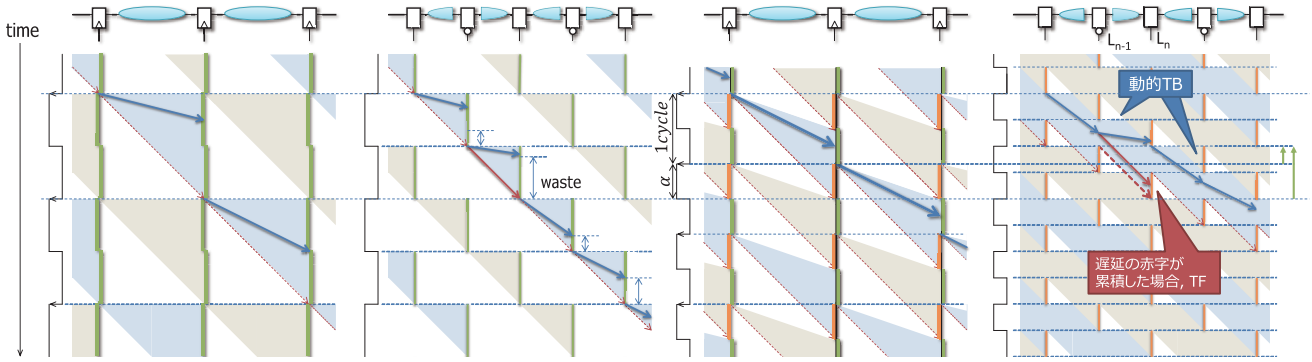

図 2　各クロッキング方式のタイミング・ダイアグラム：(a) 単相 FF，(b) 二相ラッチ，(c) Razor FF，(d) 提案方式

Fig. 2　Timing diagrams for clocking schemes: (a) Single-phase FF, (b) Two-phase latch, (c) Razor FF, and (d) Proposal.

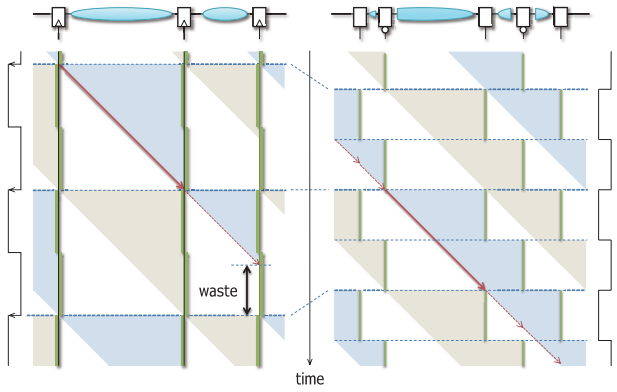

図 3　静的タイム・ボローイング（STB）

Fig. 3　Static time Borrowing (STB).

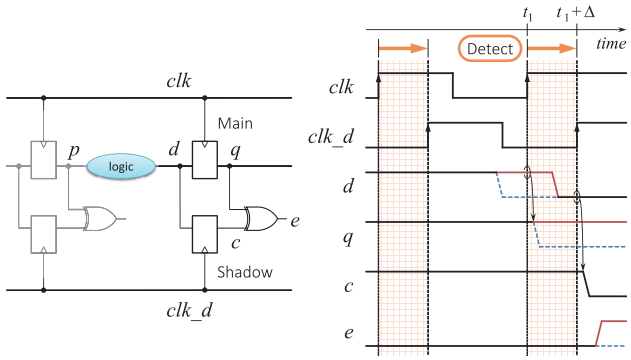

図 4　Razor FF の回路と動作

Fig. 4　Circuit and operation of Razor FF.

静的タイム・ボローイング

ただし二相ラッチ方式では，この制約を部分的に緩和できることがある．単相 FF 方式では，エッジ・トリガ動作により，信号が次のステージへと伝搬するタイミングがクロック・エッジに限定される．一方，二相ラッチ方式では，ラッチが開いている期間を活用することによって，遅延をステージ間で融通できる場合がある．

このことは一般に，タイム・ボローイングと呼ばれる．本論文では，提案手法で実現される動的タイム・ボローイング（Dynamic Time Borrowing: **DTB**）と区別するため，二相ラッチのそれを**静的タイム・ボローイング**（Static Time Borrowing: **STB**）と呼ぶことにする．

図 3 に，静的タイム・ボローイングの様子を示す．同図のように半ステージ間の遅延がバランスされていない場合に，STB は効果がある．単相 FF 方式では，サイクル・タイムは最も長いステージのクリティカル・パス遅延によって決まるため，短いステージでは無駄な時間が生じる．一方，二相ラッチ方式では，同図のように，クリティカル・パス遅延を表す直線が 1 本に結べれば，前述したクロッキング方式の 2 条件が満たされる．同図中，最も長い半ステージには 1τ が割り当てられている．すなわち，二相ラッチの最大遅延制約は，1 つの 0.5 ステージに限れば，$1\tau/0.5$ ステージ と，単相 FF 方式の 2 倍となる．ただし全体では，遅延の累積で $0.5\tau/0.5$ ステージ と，単相 FF 方式のそれと変わらない．

逆に，半ステージ間で遅延がバランスしている場合には，STB の恩恵は生じない．この場合，図 2 (b) に示すように，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，開いている期間は使われない．開いている期間を活用すべくそれ以上にサイクル・タイムを短縮した場合には，クリティカル・パスが連続で活性化するといずれサンプリング期間に間に合わず，TF となってしまう．

回路設計においては，まずステージ間で遅延をバランスさせることが肝要であり，STB を積極的に活用することは勧められてはいない．

2.6　Razor

本節では，TF 検出技術の代表として Razor FF [4] について述べる．

回路構成と動作

図 4 左に，Razor FF の回路構成を示す．1 つの Razor FF は，**メイン FF** と**シャドウ・ラッチ**によって構成され

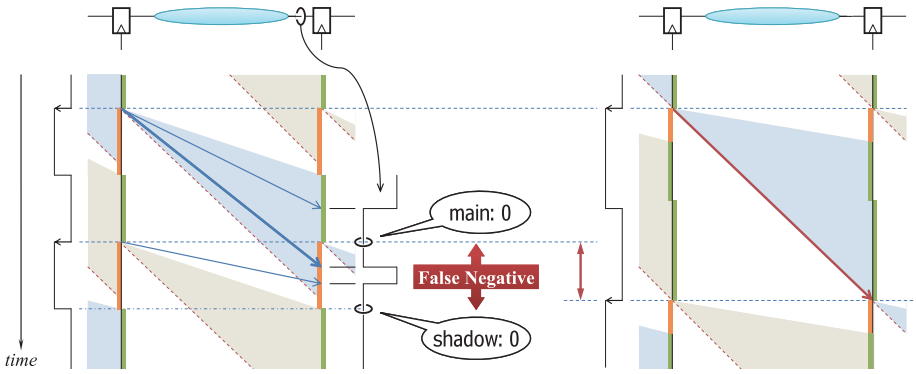

図 5　Razor 特有のショート・パス問題（左）と Razor の実際（右）
Fig. 5　Short-path problem specific to Razor (left) and Actual use of Razor (right).

る．シャドウ・ラッチには，メイン FF へのクロック clk より Δ だけ位相の遅れたクロック clk_d が供給されている．その結果，メイン FF とシャドウ・ラッチで 2 回，入力 d のサンプリングを行うことになる．それらの値が異なっていれば，TF が検出され，エラー e がアサートされる．

同図右は，d の遷移がメイン FF のクロック・エッジよりも遅れてしまった場合のタイミング・チャートである．メイン FF は t_1 で 1 をサンプリングするが，シャドウ・ラッチは $t_1+\Delta$ で 0 をサンプリングする．両者は異なっているため，e は 1 となる．t_1 から $t_1+\Delta$ の期間を，本論文では TF 検出ウィンドウと呼び，図中では網掛けで示す．

なお，メイン FF がメタステーブルとなった場合，ここで説明したダブル・サンプリングによる方法では対応できない．一方，遷移検出を用いる方式の Razor FF では，メタステーブルを TF として検出することができる [5], [10]．ただし後者は，ダイナミック・ロジックを利用するため，FPGA 上に実現することはできない．そこで以下では前者を前提として説明を行うが，同様の議論は後者についても成り立つ．

タイミング制約

図 2 (c) に，Razor FF のダイアグラムを示す．同図では，$\Delta = 0.5\tau$，すなわち，半周期遅れたクロックをシャドウ・ラッチに供給している．ダイアグラムでは，FF の下の濃さの異なる縦実線（橙色）が，TF 検出ウィンドウを表している．

クリティカル・パスの遅延に対応する 45° の破線が検出ウィンドウの下端までに到着するなら，TF が発生したとしても検出し，回復することができる．そのため，45° の破線矢印はジグザグとなる．TF 検出を行わない単相 FF や二相ラッチでは，45° の破線は一直線になっている（同図 (a)，(b)）．

TF 検出を行う方式では，このジグザグの分だけ，クリティカル・パス遅延を超えてサイクル・タイムを短縮することができる．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（図では $\alpha = 0.5$），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．

2.7　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，Razor 特有のショート・パス問題がある．

図 5（左）のダイアグラムを用いて，Razor のショート・パス問題を説明する．シャドウ・ラッチ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シャドウ・ラッチのサンプリング・タイミングよりも後に到達しなければならない．さもないと，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，シャドウ・ラッチ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 5 では，シャドウ・ラッチのサンプリングを 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．

2.8　Razor の限界

Razor FF は，遅延が τ より長いパスが，チップのどこか 1 カ所でも活性化されると，TF となって回復のペナルティを被ることになる．

そのため実際には，**図 5**（右）のようにして，TF の発生確率が十分に小さくなるようにする必要がある．すなわち，個々の個体の動作状況に合わせた実際のクリティカル・

パス遅延にほぼ一致するようにサイクル・タイムを制御する．この場合，α を大きくする意味はないので，たとえば 0.1 程度に設定する．

このことは，設計時に見積もったのではない，実際のクリティカル・パス遅延を基にサイクル・タイムを設定することを意味する．結局，実際の Razor の効果は，設計時に課せられるタイミング・マージンを削減するにとどまる．

これに対して，次章で詳述する DTB を可能にするクロッキング方式は，TF の発生確率自体を下げる効果を持つ．

3. DTB を可能にするクロッキング方式

我々は入力ばらつきにおける平均遅延に基づいた動作を可能にする手法として，DTB を可能にするクロッキング方式を提案してきた [8], [9]．

3.1 回路構成と動作

図 6 に，提案方式の回路構成を模式的に示す．提案方式は，基本的には，二相ラッチと TF 検出との組合せである．すなわち，同図上に示すような二相ラッチの回路のラッチ部分を，Razor の TF 検出回路に置き換えたものと考えてよい．なお，2.6 節で述べたように，本論文では TF 検出にダブル・サンプリングを用いた場合の説明を行うが，実用的な設計では遷移検出を想定する．

2.7 節で述べた Razor 特有のショート・パス問題を回避するため，ショート・パスに遅延を挿入する必要があるが，以下の工夫を行う：同図上の二相ラッチの回路では，ロジックのショート・パスとクリティカル・パスとが，図中○印で示すゲートで合流した後，ラッチに接続されている．この場合，合流するゲート○を二重化し，それぞれをメインとシャドウに接続する．そのうえで，シャドウに至るショート・パスにのみ遅延を挿入する．これにより，以下の 2 つを両立することができる：

- Razor 特有のショート・パス問題は，ショート・パスによりシャドウが正しい値をサンプリングできない問題であるから，シャドウに至るショート・パスに遅延を挿入すれば解消される．逆に，
- メインに至るパスに遅延を挿入しないことによって，ショート・パスが活性化した場合の実効遅延が伸びることが避けられる．3.2 節で詳述するように，これにより DTB の効果が最大化される．

実際の回路は，同図のようにショート・パスとクリティカル・パスがきれいに二分されているわけではない．実際の遅延の挿入方法は文献 [9] に詳しい．

3.2 動的タイム・ボローイング

2.5 節で述べたように，二相ラッチ方式においてはラッチの開いている期間を利用することは原則不可能であった．開いている期間を利用すべく，クリティカル・パス遅延よりサイクル・タイムを短くすると，クリティカル・パスが連続で活性化した場合に TF が発生するためである．提案方式では，TF 検出・回復を組み合わせることにより，ラッチの開いている期間を積極的に利用することが可能となる．

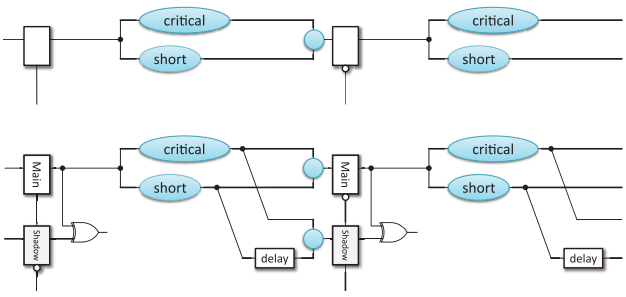

図 6　二相ラッチ（上）と提案方式（下）の回路の模式図
Fig. 6　Abstract diagrams for two-phase latch (upper) and proposed (lower) schemes.

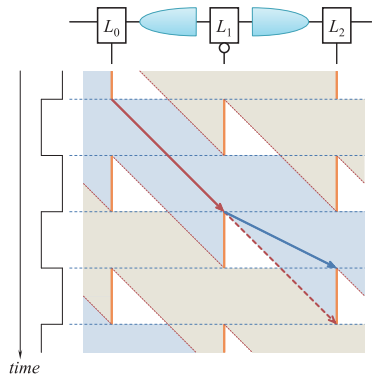

図 7　動的タイム・ボローイング（DTB）
Fig. 7　Dynamic time borrowing (DTB).

そしてこの結果，動作時に各ステージ間での**実効遅延の融通**が可能になる．**図 7** に，提案方式のダイアグラムを示す．同図では，最初の半ステージでクリティカル・パスが活性化しているが，直後の半ステージで実効遅延が 0.5τ のパスが活性化したため，ぎりぎり TF を起こすことなく動作した場合を表している．逆に，直後の半ステージで再びクリティカル・パスが活性化した場合には，TF として検出されることになる．

遅延の「借金」

このように提案方式では，ラッチの開いている期間を利用することによって，遅延の累積を解消することができる．ダイアグラム上における，直線矢印がつながってステージ間を伝播する様子は DTB の効果を表している．

このように，遅延の累積を解消するためには実効遅延が短いことが望ましい．3.1 節で述べたように，ショート・パス問題のための遅延の挿入はメインに至るパスには行わないが，それは実効遅延をできる限り短縮するためである．

遅延の「貯金」

同図では，網掛けの領域が上下にオーバラップしているが，これは図 6 に示す二重化されたパスの上で起こってい

る．すなわち，前のフェーズのシャドウに至るクリティカル・パスと，次のフェーズのメインに至るショート・パスにおける信号の伝達が同時に起こりうるため，ダイアグラム上でオーバラップして見えるのである．したがって，別のフェーズが「混ざる」ことはない．

ショート・パスが連続で活性化した場合には，（同図ではオーバラップの裏で）信号はラッチの閉じている期間に到着する．そこで，ラッチが開くまで待たされることになる．

したがって提案方式では，遅延の「借金」を持ち越して解消することができるが，遅延の「貯金」を持ち越すことは残念ながらできない．

タイミング制約

提案方式の最大遅延制約は，Razor と同様，TF 検出の検出限界によって決まる．図 7 のように，クリティカル・パスの遅延に対応する 45° の破線が検出ウィンドウの下端までに到着するなら，TF を検出することができる．

ただし提案方式では，前述したオーバラップによって，サイクル・タイムをさらに短縮することが可能となる．提案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．

大数の法則と入力ばらつき

開いている期間においては，ラッチはバッファとして機能する．すなわち，開いている期間を信号が通過する限りにおいては，各半ステージのロジックは，長大な 1 つの組合せ回路として動作することになる．このため，大数の法則により，入力ばらつきの平均値に基づく動作が可能となるのである．

3.3　クロッキング方式ごとの最小サイクル・タイムの比較

本節の最後に，各クロッキング方式における 1 ステージのクリティカル・パス遅延 c と，シャドウ FF/シャドウ・ラッチへのショート・パス遅延 s に対する最小・最大サイクル・タイムについてまとめる．各クロッキング方式の最小/最大遅延制約を満たすように最小/最大サイクル・タイム τ は，**表 1** のようにまとめられる．Razor は，提案方式と同じく，$\alpha = 0.5$ とした．

TF 検出を行う方式では，最大のサイクル・タイムがシャドウに至るショート・パスの遅延に応じて決まる．提案においては $1/2 \times c$ から s までのサイクル・タイムを取りうるため，c を所与とすると，s は $1/2 \times c$ 以上である必要がある．

表 1　クロッキング方式の最小/最大サイクル・タイム
Table 1　Minimum/maximum cycle times of schemes.

方式	最小	最大
単相 FF	c	N/A
二相ラッチ	c	N/A
Razor	$2/3 \times c$	$2 \times s$
提案方式	$1/2 \times c$	s

4.　DTB を可能にするクロッキング方式の適用

本章では，34-bit の RCA を用いたカウンタへの提案方式の適用に関して詳述する．提案方式の適用は，単相 FF 方式で構成された回路を対象として，二相ラッチ化と TF 検出のための回路変換と，TF からの回復のための機構の付与によって行う．

なお，以降はパスの遅延はパス上の論理ゲートの個数によって計算する．また，二相ラッチ化における自由度を増やすため，FPGA のキャリー・チェーンは使用しない．また，遅延素子には LUT を用いる．

4.1　二相ラッチ化と TF 検出機構の付与

図 8 に，7-bit の RCA を用いたカウンタに対して，回路変換を行う例を示す．

まず，文献 [9] で提案されたアルゴリズムを用いて，二相ラッチ方式への変換を行う．クリティカル・パスがラッチを境に二分されるようにラッチ挿入が行われる．

次に，ラッチを Razor latch へ置き換える．3.2 節で述べ

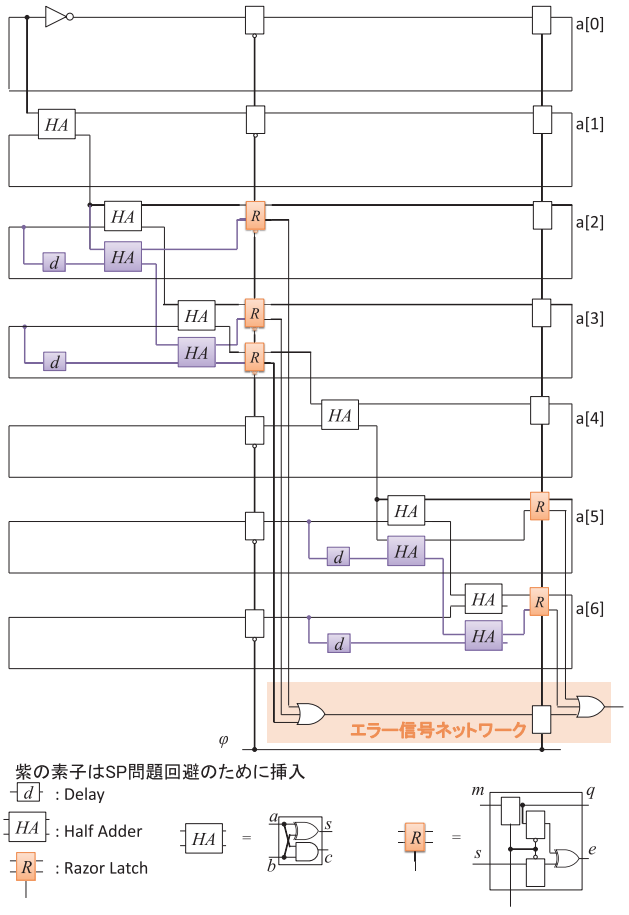

図 8　7-bit の RCA カウンタへの提案方式のための回路変換の例
Fig. 8　Conversion of 7-bit RCA counter for our proposed method.

たように，DTB を可能にするクロッキング方式では半ステージのクリティカル・パス遅延によって最小サイクル・タイムが決まり，クリティカル・パス遅延の 1/2 を超える遅延を持つパスが検出対象である．この回路の半ステージのクリティカル・パス遅延は 4 であるから，2 つ以上の LUT を通るパスの終端ラッチを Razor latch へ置き換える．

次に Razor latch に至るショート・パス遅延がショート・パス問題を起こさないように，Razor latch に至るショート・パスの一部の回路素子を複製し，遅延素子を挿入する．図中の紫色の素子はこうして挿入された素子である．ここで，3.3 節で述べたように，シャドウ・ラッチへのショート・パスはクリティカル・パスの 1/2 以上でなければならないため，本例ではショート・パスが 2 つ以上の LUT を通過するように遅延素子を挿入する．ただし，34 bit の RCA カウンタに対しては，配線遅延などのばらつきを考慮し，ショート・パスの遅延がクリティカル・パスの 2/3 以上となるように多めに遅延素子を挿入する．

最後に，Razor latch が出力するエラー信号を半ステージごとに OR ゲートによって集約し，コミット・ステージへ伝搬する．伝搬された回路全体のエラー信号は，次章で述べる回復のための制御回路に入力される．

4.2　TF からの回復機構の付加

回復手法の概要

TF を検出した後，回路は TF の回復を行う．

回復手法として，Razor の最初期の論文では回路レベルの手法が採用されていた．この手法は，TF 検出直後にシャドウ FF の正しい値をメイン FF にコピーすることでエラー訂正を行う．しかし，この方式では検出したサイクル内にパイプライン全体をストールさせる必要があり，大規模な回路では現実的ではない．

それゆえに，RazorII 以後の論文では，アーキテクチャレベルの手法が採用されている [10]．この手法では，TF による影響を受けた命令があたかも例外を起こしたかのように扱い，その命令と後続する命令をフラッシュ，つまりキャンセルして再実行することによって回復する．我々もまたスーパスカラ・プロセッサに適した回復手法を提案している [11], [12]．本論文ではアーキテクチャレベルの回復手法を前提とする．

この方式ではエラー回復のオーバヘッドは例外からの回復と同程度であり，数～数十サイクルとなる．TF の発生が数千サイクルに 1 回程度になるように電圧・サイクル・タイムを制御すれば，オーバヘッドは 1%程度に抑えることができる．

再実行

TF が発生しコミットされなかったフェーズの再実行においては，再び TF を起こさないようにする必要がある．そのために，再実行時に周波数を下げる方式と，各パイプライン・ステージでの 1 つのフェーズの実行を複数サイクル継続して行う方式が存在する [13]．本論文では後者の方式を採用する．

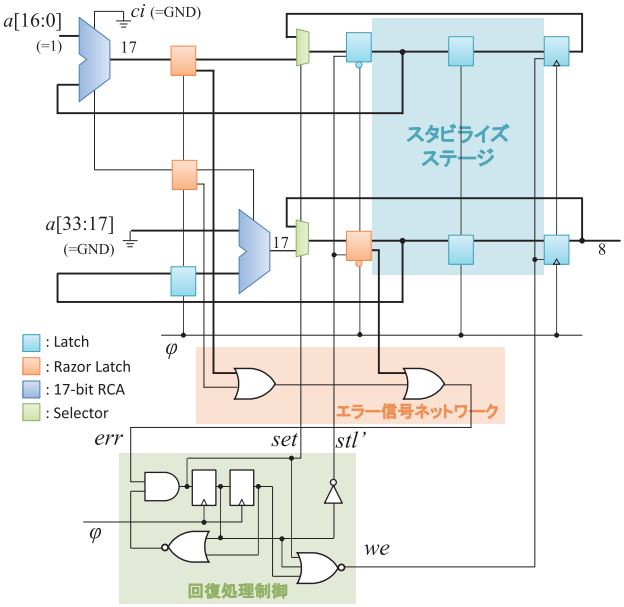

図 9　回復機構を含めたカウンタの回路構成
Fig. 9　A counter with a recovery mechanism.

この再実行方式では，サイクル・タイム τ がクリティカル・パス遅延 d の $1/n$ 倍以上であるようなとき，元の n 倍のサイクル数を再実行されるフェーズの実行にかける．こうすることで，TF を発生させることなく実行を完了できる．このために，後続のフェーズの実行開始を $(n-1)\times$ ステージ数だけ遅らせる．再実行されたフェーズの結果は $n\times$ ステージ数の間確定しないため，その間はコミットを行わない．

たとえば，3.2 節で述べたように，提案方式のサイクル・タイム τ の最小はクリティカル・パス遅延 d の 1/2 であるため，$n = 2$ の場合までを考慮すればよい．したがって，再実行されるフェーズの後続は 1 ステージ数分だけ後に実行される．また，コミット・ステージへの伝搬は 2 ステージ数分だけ停止した後に行う．

提案方式を適用したカウンタの回復動作

図 9 に回復機構を組み込んだカウンタのブロック図を示す．

カウンタをプロセッサにおけるプログラム・カウンタに見立て，コミット済みのプログラム・カウンタを記憶する FF を加えている．また，TF 検出の結果を得た後にコミットがなされる必要があるため，スタビライズ・ステージを設けている．

Razor FF から出力されたエラー信号は，エラー信号ネットワークに集約される．ネットワークの最終的な結果である *err* は回復処理の制御回路に入力される．回復処理の制御回路は，*err* を入力として，4.2 節で述べた再実行処理を

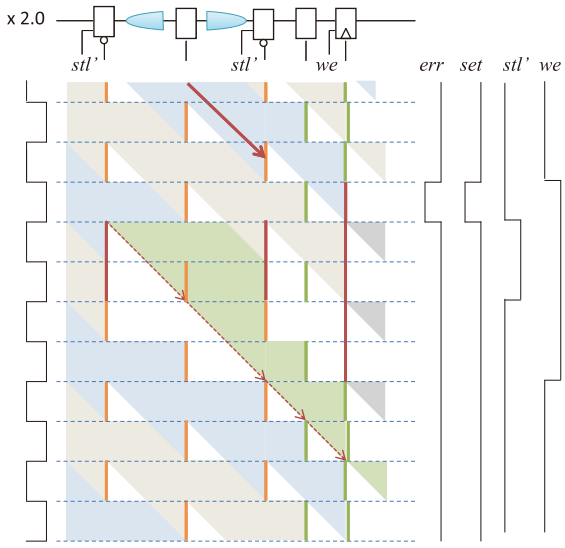

図 10　DTB を適用した回路における回復と再実行の様子
Fig. 10　Timing diagram for recovery of a circuit with clocking scheme that enables dynamic time borrowing.

制御する以下の信号を生成する：

set　再実行開始時のカウンタの値をセットするためのセレクト信号

stl　再実行中のカウンタの更新を止めるための信号

we　architecture state 更新のイネーブル

本回路は 1 ステージの回路であるから，再実行されるカウンタ計算の後続の計算は 1 サイクル待たされた後に実行される．また，コミット・ステージへの伝搬は 2 サイクルだけ停止した後に行う．したがって，回復のたびに必要なペナルティは 3 サイクルである．

図 10 に TF の検出の後に再実行が行われる様子を示す．同図の右側は制御回路が出力する信号群の波形を示したものである．まず，赤矢印で示されるパスの活性化によって TF が発生すると，そのサイクルの後半にエラー信号 *err* がアサートされる．同タイミングで，*we* がディスエーブルされ，architecture state の更新が止まる．また，同タイミングで *set* がアサートされ，カウンタに architecture state からの正しい値がロードされる．その次のサイクルから，*stl* がステージ数だけのサイクルの間アサートされることでカウンタの更新を止める．こうして，周波数が高くとも演算に十分な時間が確保され，再実行時は TF を起こすことなく演算が行われる．*we* のディスエーブルは 2×ステージ数のサイクル数の後に解除される．その後，architecture state が再実行された演算の結果によって更新される．

5.　評価：FPGA への実装

本章では，各クロッキング方式を適用したカウンタを FPGA に実装し，その動作可能な周波数を測定する．対象のカウンタは RCA を用いたもので，クロック・サイクルごとに 1 ずつカウントアップする．5 MHz ずつ周波数を上げていき，クロック・サイクルごとに 1 ずつカウントアップが行われなくなる（直前の）周波数を測定する．

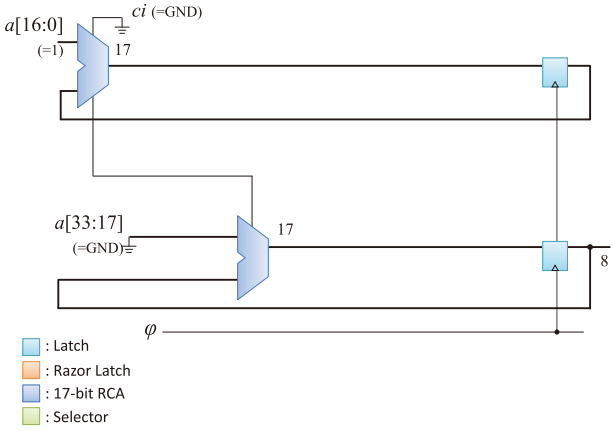

図 11　単相 FF によるカウンタ
Fig. 11　A counter with single-phase FF.

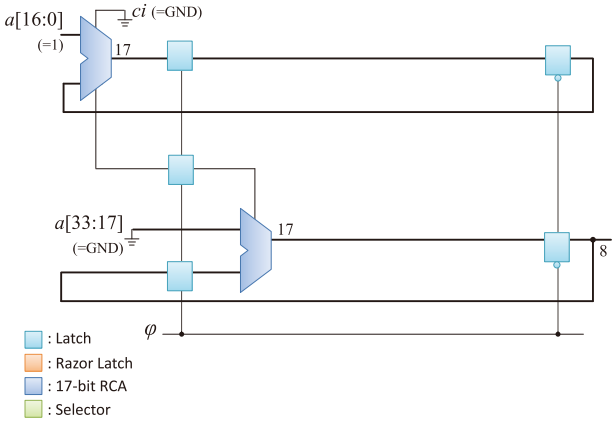

図 12　二相ラッチによるカウンタ
Fig. 12　A counter with two-phase latch.

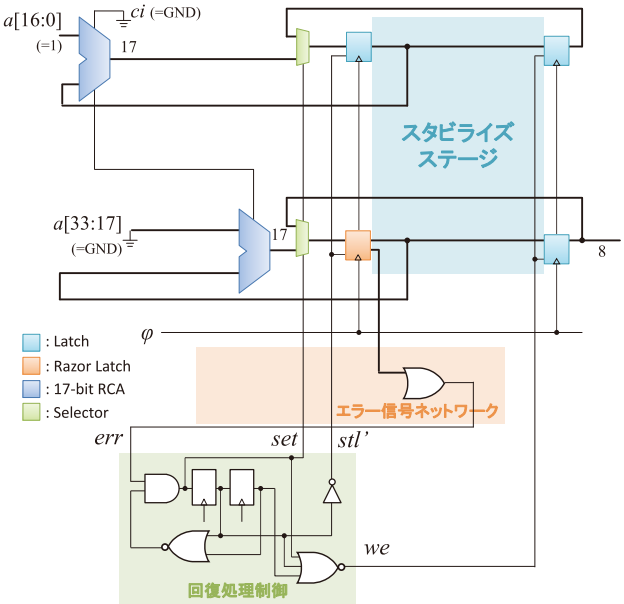

図 13　Razor によるカウンタ
Fig. 13　A counter with Razor.

5.1　評価方法

対象の回路構成

比較するクロッキング方式は，単相 FF，二相ラッチ，Razor，提案方式の 4 つである．

提案方式の回路構成は，図 9 に示した．それ以外に関しては，**図 11**，**図 12**，**図 13** にそれぞれ示す．

なお，Razor においては，3.3 節で述べたように，単相 FF 方式の 1.5 倍の周波数で動作することを想定し，クリティカル・パス遅延の 2/3 を超える遅延を持つパスを検出対象とし，シャドウ・ラッチに至るショート・パスはクリティカル・パス遅延の 1/2 を超えるように遅延挿入を行う．

評価環境

使用したボードは Digilent Nexys 4 DDR で，搭載 FPGA は Xilinx Artix-7 XC7A100T-1CSG324C である．

論理合成，配置配線には Vivado Design Suite 2016.3 を用いた．逆相ラッチと Razor FF の挿入は，4.1 節で述べたツールによって行った．

評価方法

34 bit のカウンタの上位 8 bit を FPGA ボードに備え付けられた 8-bit LED に出力する．配置配線に不要な制約を与えないため，カウンタの上位 8 bit はそれぞれ 2 つ FF を介した後に LED に出力しており，カウンタの値は 2 サイクルの遅れで常時 LED に表示される．各クロッキング方式が正しく動作したかどうかは，この 8-bit LED の表示が 1 ずつカウントアップするどうかを目視によって確認する．最高 300 MHz では，上位 8 bit のうちの最下位ビットは，$300\text{MHz} \div 2^{(34-8)} \simeq 4.5\,\text{Hz}$ で点滅するため，目視で確認できる．下位（$34-8=$）26 bit については LED などによって直接には観測していないが，パス遅延がより短いため，TF が起こる確率は無視してよい．

RCA を用いたカウンタのクリティカル・パスは，RCA のキャリー・チェーンである．動作周波数を上げていったときに最初に起こる TF は，最上位桁へのキャリーの伝搬が間に合わなくなるというものである．すなわち，8-bit LED の値が，本来 011…11 → 100…00 と変化するところ，TF 発生時には，011…11 → 000…00 と変化することになる．

TF 検出・回復

TF 検出・回復を行う方式では，回復は 3 サイクルで行われるため（4.2 節），上述した 011…11 → 000…00 の変化は目視では観測できない．TF の発生は，エラー信号がアサートされた後（しばらくの間）点灯するエラー LED によって確認する．その結果，TF が発生する周波数の領域では，目視で 8-bit LED は 1 ずつカウントアップし続けているように見えるが，このエラー LED が点滅することになる．

動作周波数が TF の検出限界を超えると，メイン FF に加えて，シャドウ・ラッチにも間に合わなくなるため，エラー LED も点灯しなくなる．

したがってこの評価では，TF 検出を行わない手法（単相 FF，二相ラッチ）については，TF が発生せずに動作する周波数を；TF 検出を行う手法（Razor，提案方式）については，TF が検出・回復可能な周波数を，それぞれ測定することになる．

一般には，4.2 節で述べたように，周波数向上による性能向上と TF からの回復のペナルティによる性能低下のトレードオフが存在するが，この評価では，回復のペナルティは考慮していない．後述するように，このようなカウンタでは TF がほとんど起こらない．したがって，周波数を上げても，回復のペナルティによる性能低下が，周波数向上による性能向上を上回ることはない．

5.2　カウンタにおける TF 発生率

本章では TF 検出を備えた方式に関して，カウンタにおける TF 発生率（Occurrence Rate of Timing Faults: ORTF）について記す．

カウンタの段数を $n = 34$ とおく．1 段の桁上げのゲート遅延を一律に d_c とすると，桁 i の変化するときのパスの遅延 d_i は $d_c i$ で与えられる．特に CP の遅延 c は $d_c n$ と計算できる．また，桁 i が変化する確率は $(1/2)^{i-1}$ で与えられる．

Razor においては，サイクル・タイム t に対して，$d_i > t$，すなわち $d_c i > t$ であるような桁 i の値が変化する際に TF が発生する．適用対象のカウンタでは $i+1$ 桁が反転するときは i 桁も反転することに注目すると，$d_c i > t$ である 1 番桁が小さい i の変化の確率が ORTF となることが分かる．

提案方式においては，$d_c i > 3/2 \times t$ であるような桁 i の変化が TF を起こす．Razor と同様にそうした i のうち最も桁が小さい i のパスの活性化確率が ORTF となる．

このサイクル・タイムと ORTF の関係を**図 14** に示す．ここでは c は 0.2 ns とした．ORTF の軸は対数表示である．Razor と提案方式はそれぞれの方式において，TF が起きる最大のサイクル・タイムから，TF が検出できる最小

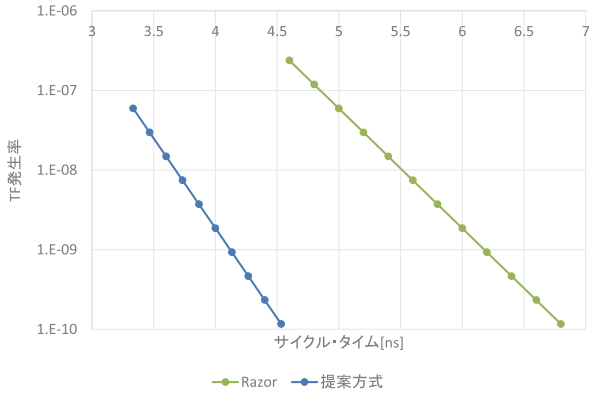

図 14　サイクル・タイムに対する TF 発生率
Fig. 14　Occurrence rate of TF corresponding to cycle time.

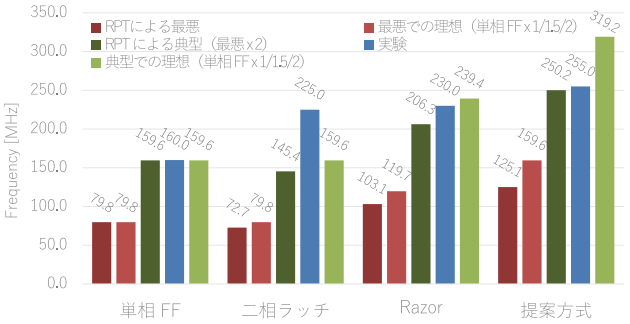

図 15　各クロッキング方式の理論値と実験結果

Fig. 15　Theoretical and experimental value for each clocking scheme.

のサイクル・タイムまでをプロットしている．

提案方式の ORTF は，同じサイクル・タイムのときの Razor に対して小さく，たとえば $t = 4.5$ ns 付近の点では 1/1000 ほど小さい．また，検出限界付近における ORTF を比べても提案方式の方が低く，提案方式は再実行のペナルティを比較的低く抑えたままサイクル・タイムを削減していくことができることが示されている．

しかし，いずれの手法でも TF の発生率は小さく，周波数を上げることによる性能向上を妨げるほどではない．そのため，本論文では動作可能な周波数のみを評価対象とする．

5.3　実験結果

図 15 はそれぞれの手法について実験結果を示す．また，以下の理論値を実験結果とともに示す：

RPT による最悪（濃赤）　Vivado のタイミング・レポート（RPT）から得られるクリティカル・パスのワースト・ケース遅延を基に，2 章と 3 章で示した式から計算した，すなわち，クリティカル・パス遅延 d に対して最小サイクル・タイムは，単相 FF：d，二相ラッチ：d，Razor：$2/3 \times d$，提案方式：$1/2 \times d$ となるとして計算した．実際の製品の定格周波数となる．

最悪での理想（赤）　3 章で述べた式でパスの遅延 d を一定とした場合，つまり単相 FF 方式の最悪（79.8 MHz）を，それぞれ，1，1，1.5，2 倍にしたもの（最小サイクル・タイムの 1，1，2/3，1/2 の逆数）．

RPT による最悪（濃赤）は，最悪での理想（赤）よりはいずれも若干小さくなっている．これは，単相 FF 以外の方式では，逆相ラッチやシャドウ FF，遅延素子などのオーバヘッドによりクリティカル・パス遅延 d の方が若干伸びるためである．

RPT による典型（濃緑）　ティピカル・ケースを想定し，RPT の結果を 2 倍にしたもの．単相 FF の実験結果とよく符合するため，実験環境（温度・電圧）ではワースト・ケースのおよそ 2 倍となっていると考えられる．

実験結果（青）　8-bit LED によって，カウンタが正しく動作していることを観測できた最大の周波数．提案方式は単相 FF 方式に対して 1.6 倍の周波数で動作することを確認できる．

典型での理想（緑）　単相 FF 方式の典型（159.6 MHz）を，それぞれ，1，1，1.5，2 倍にしたもの（最小サイクル・タイムの 1，1，2/3，1/2 の逆数）．

実験結果（青）が，RPT による典型（濃緑）と理想（緑）との間の値を示すことは，実験環境（温度・電圧）において回路が想定どおりの動作していることを示す．

実験結果（青）は，単相 FF，Razor，提案方式では，典型の場合とほぼ相違がない．一方，二相ラッチについては，典型の場合よりも実験結果が高い周波数を示した．

二相ラッチにおける DTB

これは，以下のように，カウンタでは二相ラッチでは実際には利用できない DTB が起きるからである．

カウンタの値が，$011\ldots10 \to_{-1} 011\ldots11 \to_{0} 100\ldots00 \to_{+1} 100\ldots01$ と変化するとき，$\to_0$ の遷移において最上位ビットへの桁上げが伝搬され，クリティカル・パスが活性化する．その直前・直後のサイクルにおいては，変化するのは，最下位のビットのみであり，実効遅延は非常に小さい．

したがって，$\to_0$ のサイクルにおいては，遅延の累積のない状態から始まり，ここで 0.5 cycle 分借金をしたとしても，次のサイクルには解消されるから，TF を起こすことなく動作することができる．

この場合の最小サイクル・タイムは $2/3 \times d$ となり，実験結果とよく符合する．

しかし，カウンタではない一般的な回路においては，クリティカル・パスが連続で活性化されることを考慮しなければならない．したがって，TF 検出を備えない二相ラッチ方式では，実験結果（青）のような高い周波数で動作させることはできない．

6.　今後の予定

我々はティピカル・ケースの遅延に基づいた動作を実現するための手法として，DTB を可能にするクロッキング方式 [8] を提案している．これまでは，手動による適用を行っており，TF からの回復機構を組み込んでいなかった．本論文では開発した自動適用ツール [9] を用いて回路変換を行うとともに，アーキテクチャレベルの TF からの回復手法を RCA によるカウンタに適用し，FPGA 上に実装し，その実効的な周波数の向上について評価を行った．

我々は現在，NORCS [14] など様々な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行っており，今後はこのプロセッサに DTB を可能にするクロッキング方式を適用し，より詳細な評価を行う予定である．

謝辞　本研究の一部は，文部科学省科学研究費補助金 No.16H02797 による．

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（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110007227367/⟩ (2009).

[2] Srivastava, A., Sylvester, D. and Blaauw, D.: *Statistical Analysis and Optimization for VLSI: Timing and Power*, Springer Science & Business Media (2006).

[3] Mukhopadhyay, S., Mahmoodi, H. and Roy, K.: Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol.24, No.12, pp.1859–1880 (online), DOI: 10.1109/TCAD.2005.852295 (2005).

[4] Ernst, D., Kim, N.S., Das, S., Pant, S., Rao, R., Pham, T., Ziesler, C., Blaauw, D., Austin, T., Flautner, K. and Mudge, T.: Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, *Proc. 36th Annual IEEE/ACM Int'l Symp. Microarchitecture*, pp.7–18 (online), DOI: 10.1109/MICRO.2003.1253179 (2003).

[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, *IEEE Int'l Solid-State Circuits Conf. Dig. Tech. Papers*, pp.284–285 (online), DOI: 10.1109/ISSCC.2010.5433919 (2010).

[6] 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, *IEEE J. Solid-State Circuits*, Vol.44, No.1, pp.49–63 (online), DOI: 10.1109/JSSC.2008.2007148 (2009).

[7] Choudhury, M., Chandra, V., Mohanram, K. and Aitken, R.: TIMBER: Time borrowing and error relaying for online timing error resilience, *Design, Automation and Test in Europe Conf. Exhibition* (*DATE*), pp.1554–1559 (2010).

[8] 吉田宗史，広畑壮一郎，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，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).

[9] 津坂章仁，谷川祐一，広畑壮一郎，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ための二相ラッチ生成アルゴリズム，研究報告計算機アーキテクチャ（ARC），Vol.2014, No.9, pp.1–10（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009808089/⟩ (2014).

[10] Das, S., Tokunaga, C., Pant, S., Ma, W.-H., Kalaiselvan, S., Lai, K., Bull, D.M. and Blaauw, D.T.: RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance, *IEEE J. Solid-State Circuits*, Vol.44, No.1, pp.32–48 (online), DOI: 10.1109/JSSC.2008.2007145 (2009).

[11] 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.6, No.1, pp.17–30（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009527308/⟩ (2013).

[12] 吉田宗史，倉田成己，塩谷亮太，五島正裕，坂井修一：タイミング・フォールト耐性を持つ Out-of-Order プロセッサの検出/回復方式，先進的計算基盤システムシンポジウム SACSIS，pp.10–19 (2013).

[13] Bowman, K.A., Tschanz, J.W., Lu, S.L.L., Aseron, P.A., Khellah, M.M., Raychowdhury, A., Geuskens, B.M., Tokunaga, C., Wilkerson, C.B., Karnik, T. and De, V.K.: A 45 nm Resilient Microprocessor Core for Dynamic Variation Tolerance, *IEEE Journal of Solid-State Circuits*, Vol.46, No.1, pp.194–208 (online), DOI: 10.1109/JSSC.2010.2089657 (2011).

[14] Shioya, R., Horio, K., Goshima, M. and Sakai, S.: Register Cache System not for Latency Reduction Purpose, *Proc. Int'l Symp. Microarchitecture* (*MICRO-43*), pp.301–312 (online), DOI: 10.1109/MICRO.2010.43 (2010).

神保 潮　（学生会員）

1990 年生．2013 年東京大学工学部電子情報工学科卒業．2015 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．同年総合研究大学院大学複合科学研究科情報学専攻進学．コンピューティング・システムの研究に従事．

山田 淳二　（正会員）

1980 年生．2010 年信州大学工学部情報工学科卒業．2012 年東京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．2017 年同博士課程修了．博士（情報理工学）．2004 年から 2015 年エルピーダメモリ株式会社（現マイクロンメモリジャパン株式会社）で DRAM 製品の開発に従事．2017 年より東芝メモリ株式会社で NAND Flash メモリの開発に従事．IEICE 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．2014 年国立情報学研究所教授．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．本会シニア会員．