Vol.2017-ARC-226 No.18
2017/5/24

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スカラ・プロセッサへの適用

神保 潮[1,a)]　五島 正裕[2]

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一つとして，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み合わせにより実現され，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．本稿では，FPGA をターゲットとし、スカラ・プロセッサである Rocket に対して本方式を適用する手順について述べる。また、本方式のための回路変換を行った際の回路オーバーヘッドについて示す。

Application of Clocking Scheme That Enables Dynamic Time Borrowing to a Scalar Processor.

JIMBO USHIO[1,a)]　GOSHIMA MASAHIRO[2]

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・ケースほど悲観的ではない遅延見積も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，十分なマージンを取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 [4–7]．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を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

[1] 総合研究大学院大学 複合科学研究科
School of Multidisciplinary Sciences, SOKENDAI (The Graduate University for Advanced Studies)
[2] 国立情報学研究所 アーキテクチャ科学研究系
National Institute of Informatics, Systems Architecture Research Division
[a)] ushio@nii.ac.jp

可能になる．

本稿の内容

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]．現在，本手法の回路への実装による評価を行っている．今までは，カウンタのような小規模な回路への適用のみが行われていた．本稿では，スカラ・プロセッサ Rocket [9] を対象としてこの手法を適用し，FPGA 上に実装する場合の回路オーバーヘッドについて確認する．

以下，本稿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 章では提案方式の適用手順について述べ，5 章で評価結果を示す．

2. 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時，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．

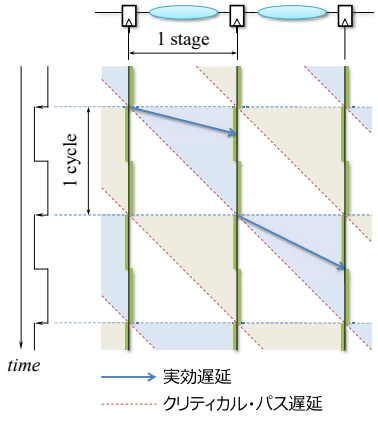

図 1: 単相 FF のタイミング・ダイアグラム

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訳ではない．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言う．

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．

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．図中の網掛けの二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時，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．他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言える．

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．

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，矢印が存在し得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．

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) 矢印が存在し得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．

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播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．本稿では，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一本に結べれば，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

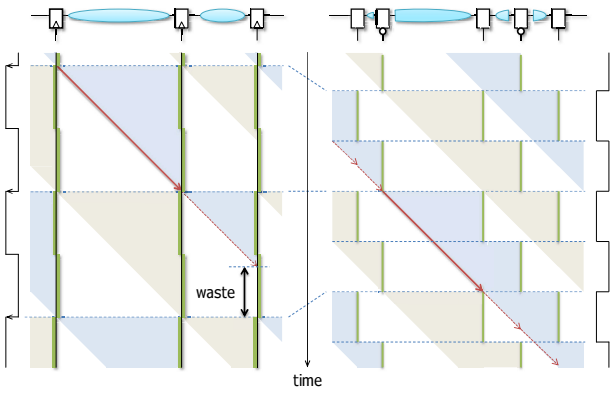

図 3: 静的タイム・ボローイング (STB)

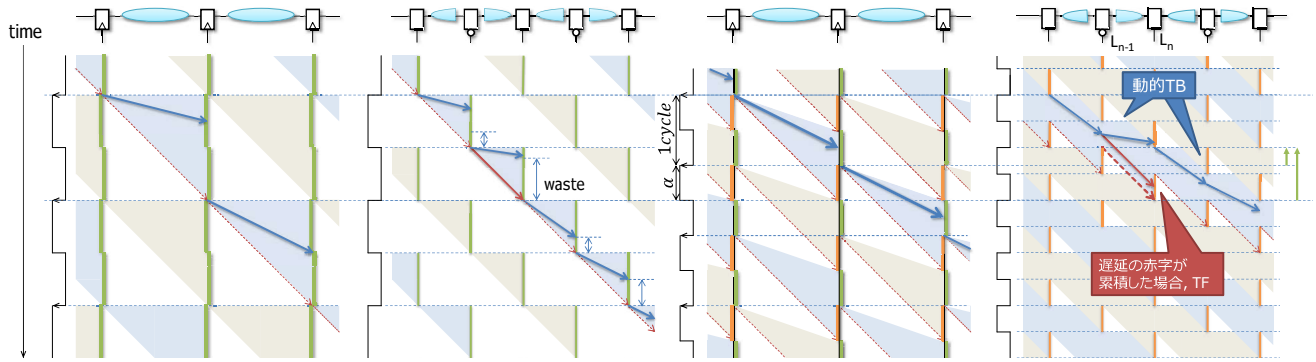

図 2: 各クロッキング方式のタイミング・ダイアグラム：
(a) 単相 FF，(b) 二相ラッチ，(c) Razor FF，(d) 提案方式

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，本稿では 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．

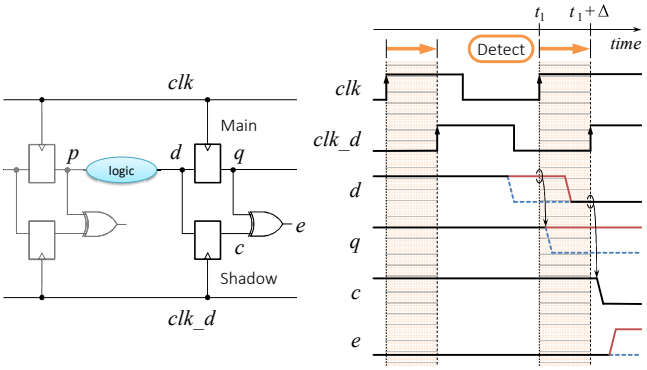

図 4: Razor FF の回路と動作

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

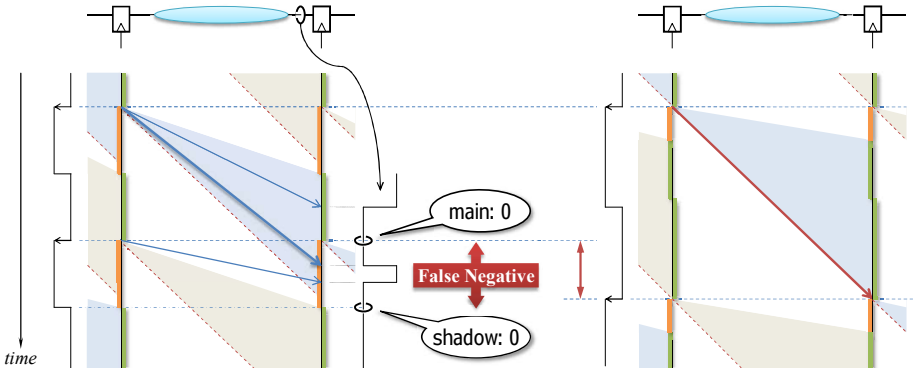

図 5: Razor 特有のショート・パス問題（左）と Razor の実際（右）

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 か所で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，例えば 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留まる．

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, 11].

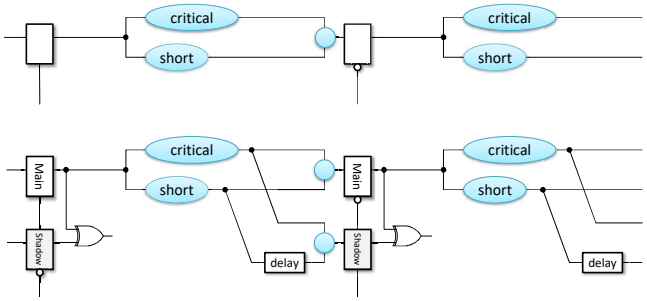

図 6: 二相ラッチ（上）と提案方式（下）の回路の模式図

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 検出との組み合わせである．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，本稿では 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．その上で，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訳ではない．実際の遅延の挿入方法は [11] に詳しい．

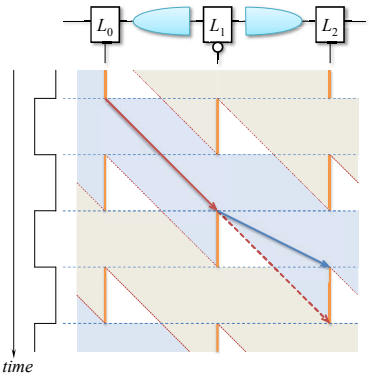

図 7: 動的タイム・ボローイング (DTB)

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．

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ーラップしているが，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得るため，ダイアグラム上でオーバー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，(同図ではオーバーラップの裏で) 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，前述したオーバーラップによって，サイクル・タイムを更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 つの組み合わ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

本章の最後に，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・タイムを取り得るため，c を所与とすると，s は $1/2 \times d$ 以上である必要がある．

4. DTB を可能にするクロッキング方式の適用

本章では，スカラ・プロセッサへの提案方式の適用に関して詳述する．

4.1 スカラ・プロセッサへの適用手順

本稿はイン・オーダの 5-ステージ・スカラ・プロセッサである Rocket [9] を適用の対象とする．ハードウェア記述

表 1: クロッキング方式の最小/最大サイクル・タイム

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

言語である Chisel 及び Verilog によって記述されており，ソースコードが公開されているため改変が容易である．また，メモリコントローラ等 FPGA ボード上の FPGA 外の機器とのインターフェースが利用しやすい [12]．オプションとして ISA は RV64G，FPU がないバージョンを採用する．

Rocket への提案方式の適用は以下のような手順で行う．

(1) TF の発生を想定する回路部分 (A) を明確にし，スタビライズ・ステージと回復機構を RTL 記述で付加する．
(2) A に対して，論理合成によって得たネットリストに対して，二相ラッチ方式化と TF 検出のための回路変換を行う．
(3) 変換後のネットリストを元の回路部分 A と置き換え，全体に対して通常通り論理合成，配置配線等を行う．

本稿では，この手順のうち，回路変換を行うステップの回路素子の増加について測定を行う．変換を行う対象としては，Rocket コアの命令デコード及びレジスタ読出しと演算ステージを担う部分回路とする．以下回路変換について詳細を述べる．

4.2 二相ラッチ化と TF 検出のための回路変換

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．なお，本小章で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，遅延素子には 1 入力の LUT を用いる．

まず，二相ラッチ方式への変換を行う．変換前のステージのクリティカル・パスがラッチを境に均等に二分されるようにラッチ挿入が行われる．このために以下のような手順で回路変換を行う．

(1) 変換対象のネットリスト中の FF すべてに対して，その出力側に FF を新たに挿入する．
(2) こうして得られたネットリストを論理合成ツールに入力し，リタイミングオプションを使用して論理合成を行う．この時，変換前からあった FF に対してはリタイミング時に位置を移動しないようにコメント等により指示を与える．すると，挿入された FF の位置は，変換前のステージのクリティカル・パスを 2 分するように移動される．
(3) 最後に，挿入された FF 群と，変換前にあった FF 群をそれぞれ正相，逆相のラッチに変換する．

次に，ラッチを Razor latch へ置き換える．3.2 節で述べたように，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 を超える遅延をも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のプロセスで付加された素子である．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．

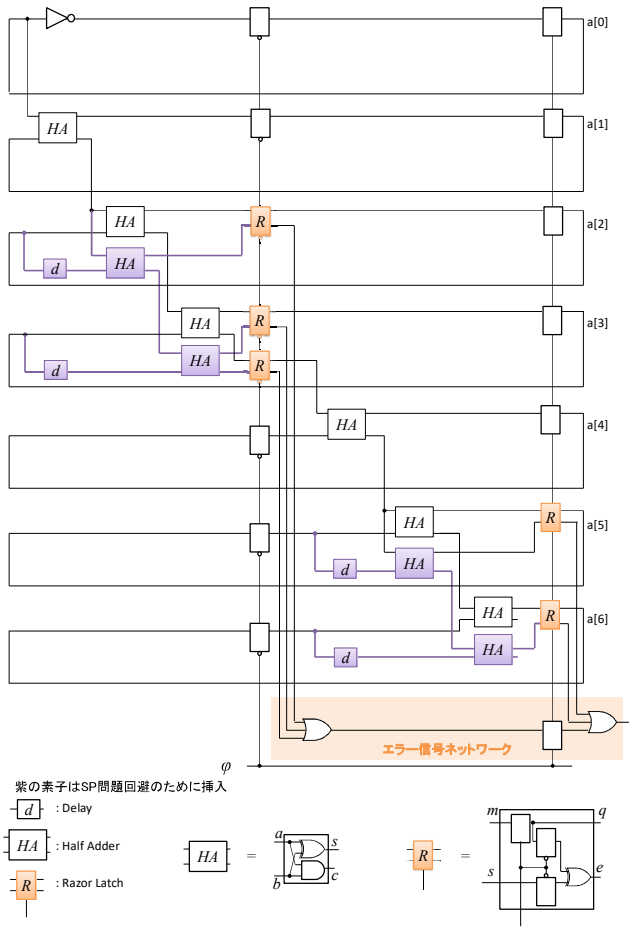

図 8: 回路変換の例:7-bit の RCA カウンタの変換

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・ステージまで命令のデータフローと共に伝搬する．伝搬されたエラー信号は回復機構への入力となり，回復処理を駆動する．

5. 評価

本章では，FPGA をターゲットとし、Rocket に対して提案方式のための二相ラッチ方式化と TF 検出機構の追加を行った際のオーバーヘッドについて記す。ターゲットとなる FPGA チップは Xilinx Artix-7 FPGA XC7A100T-1CSG324C である．論理合成ツールとして Vivado Design Suite 2016.3 を使用し，論理合成時は DSP を利用しないオプションを付加した．

表 2 に変換における各ステップの回路素子数と，二相方式化後から TF 検出機構付加後の追加分の内訳を示す．回路変換前と二相ラッチ方式化後を比べると，FF/ラッチ数は 2 倍弱になっている．これは概ね 1 つの FF が 2 つのラッ

表 2: 提案方式のための回路変換による回路オーバーヘッド

変換手順	LUT 数	FF/ラッチ数
回路変換前	4975	1719
二相ラッチ方式化後	5593	3298
TF 検出機構付加後	8005	3395
追加分の内訳		
遅延素子 (1 入力 LUT)	2254	N/A
比較器 (2 入力 LUT)	48	N/A
シャドウ・ラッチ等のラッチ	N/A	96
複製された素子等	110	1

チに置き換えられているためである．二相ラッチ方式化後と TF 検出機構付加後を比べると，回路オーバーヘッドの内訳のほとんどは遅延素子とシャドウへのパスを分離するために複製された素子であることが分かる．また，TF 検出対象となったラッチの数はラッチ全体に対して 3% 弱であることが分かった．

6. おわりに

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．これまでは小規模なカウンタに対する適用のみが行われていた．本稿では開発した自動適用ツール [11] を用いてスカラ・プロセッサへの一部に対して DTB を可能にするクロッキング方式のための回路変換を行い，その回路オーバーヘッドについて測定を行った．

今後は，このスカラ・プロセッサへの回復機構の実装によって，動作周波数向上を考慮した性能の評価を行う予定である．また，我々は現在，NORCS [13] 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，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．

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] Asanovi, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., Karandikar, S., Keller, B., Kim, D., Koenig, J., Lee, Y., Love, E., Maas, M., Magyar, A., Mao, H., Moreto, M., Ou, A., Patterson, D. A., Richards, B., Schmidt, C., Twigg, S., Vo, H. and Waterman, A.: The Rocket Chip Generator, Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley (2016).

[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] 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).

[12] Bradbury, A., Ferris, G. and Mullins, R.: Tagged memory and minion cores in the lowRISC SoC, *Memo, University of Cambridge* (2014).

[13] 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).