Vol.2013-ARC-206 No.6
2013/7/31

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評価

吉田 宗史[1,a)]　広畑 壮一郎[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．我々は以前，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した．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, この方式の適用を自動で行うツールの構成法やアルゴリズムについて述べる．

1. はじめに

半導体プロセスの微細化に伴い，素子のランダムな**ばらつき**の問題が顕在化しつつある [1], [2]．微細化によって，素子性能の平均 (typical) 値は向上するものの，ばらつきの増大によってワースト値の向上は減殺される．従来の LSI 設計は，このワースト値に基づくワースト・ケース設計である．したがって，このまま微細化を進めても従前のような性能向上は期待できなくなる．

動的タイミング・フォールト検出・回復

タイミング・フォールト (Timing Fault: TF) とは，回路遅延の動的な変動によって生じる過渡故障である．この TF を検出し，回復することにより，ワースト・ケース設計からの脱却を図る技術が提案されている．このような技術として，**Razor** [3], [4], [5] が有名であり，**DVFS** - Dynamic Voltage and Frequency Scaling [6] と組み合わせて用いられる．

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電圧 (V) と周波数 (F) の組を設定する．したがって，実際に TF が起こるのは，サーモ・センサの故障による熱暴走などの想定外の状況においてのみである．一方で，ワースト・ケースにおいても TF が発生しないような見積もりは，ばらつきが増大したプロセスでは悲観的になり過ぎる．

これに対し Razor では，ワースト・ケースよりも低い電圧 (V)，高い周波数 (F) で動作させ，その結果生じる TF を検出/回復する．回復にはペナルティが付随するから，その影響が十分に小さくなるように，TF の発生確率が十分に低い V/F の組を見つける．このようにすれば，ワースト・ケース設計で必要であったマージンを削減することができ，悲観的過ぎる見積もりから脱却することができる．

以前までの提案

我々は以前，入力ばらつきに着目することにより TF 検出技術を更に推し進め，Razor では成し得なかった，クリティカル・パスよりはるかに短い**実効遅延**に基づくサイクル・タイムを規定できるクロッキング方式を提案している [7]．実効遅延はロジックの出力が最終的に確定した遅延時間を表す．入力ばらつきと実効遅延の詳しい定義は 2.1 節で述べる．

提案方式は，端的に言えば，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．3.1 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，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遅延を融通するものである．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．

また，動的タイム・ボローイングを行っても，サイクル・

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
a) s-yoshida@mtl.t.u-tokyo.ac.jp

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．提案手法では更に，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．

本稿の内容

文献 [7] では，比較的簡単な回路に対して手作業で提案方式を適用し，FPGA 上で提案方式が実際に動作周波数 2 倍で動作することを確認している．しかし，プロセッサのように回路規模が大きいものに対する適用を手作業で行うのは，不可能に近い．

そこで現在，汎用の回路をグラフ構造に落とし込み，提案方式の適用を自動で行うツールの開発を進めている．本稿ではこの適用自動化ツールを構成する際に満たすべき要件を列挙し，実際のアルゴリズムを述べる．

以下，2 章ではまず，以前までの提案を支える入力ばらつきと実効遅延の正確な定義について，タイミング・ダイアグラムと呼ぶ図を基に説明し，さらに様々な既存のクロッキング方式について述べる．続く 3 章で，以前までの提案の概要を述べ，4 章で，適用自動化ツールの仕様と構成について述べる．

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．

クロッキング方式を理解する上では，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 1（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 1（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 $t = t_x$，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 1 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**したと言う．

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

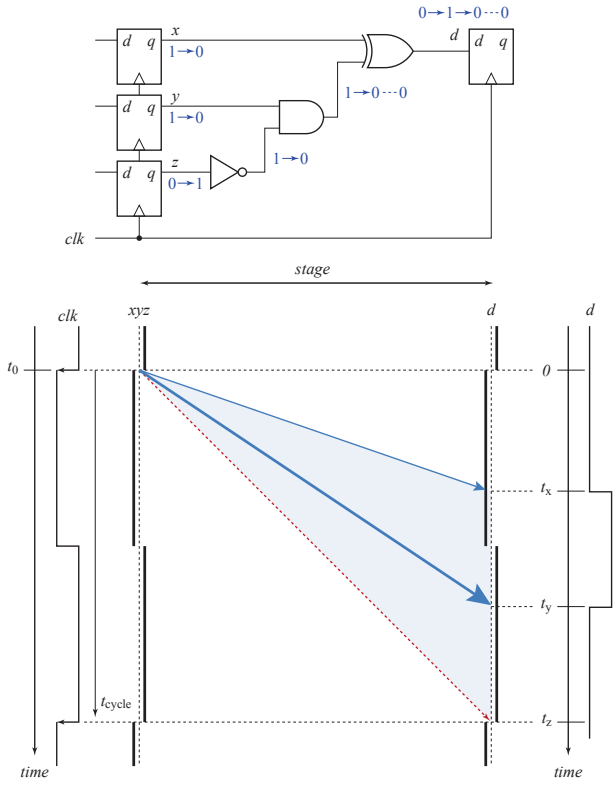

図 1 タイミング・ダイアグラム (t-diagram) と実効遅延

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，**??** 節でタイム・ボローイングの説明をする上で重要となる．

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 1 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 が変化した時には実効遅延が 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° の赤矢印を更に太くして表す．

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 [8]．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高々数十 % 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言える．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．

2.2 単相 FF 方式

図 2 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信号が伝播する様子で表すことができる．

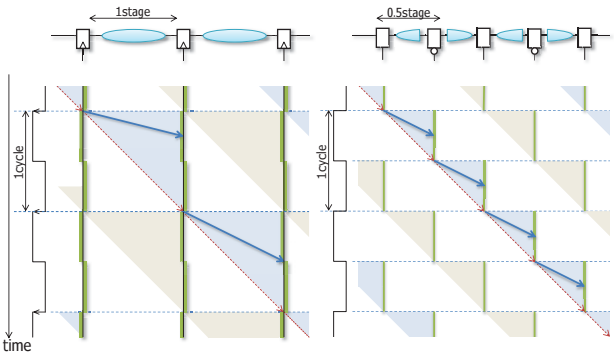

図 2 単相 FF(左) と二相ラッチ（右）の t-diagram

2.3 二相ラッチ方式

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．

二相ラッチと言っても，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 2 右が，二相ラッチ方式の t-diagram である．単相 FF ではラッチが常に閉じているが，二相ラッチ方式ではラッチの開いている区間が存在していることを特徴とする．この開いている区間を利用することにより，以下に述べるステージ間のタイム・ボローイングが可能になる．

静的タイム・ボローイング

図 3 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信号が到達していても，信号が次のステージに伝播するタイミングがクロックの立ち上がる瞬間に限定されているため，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，

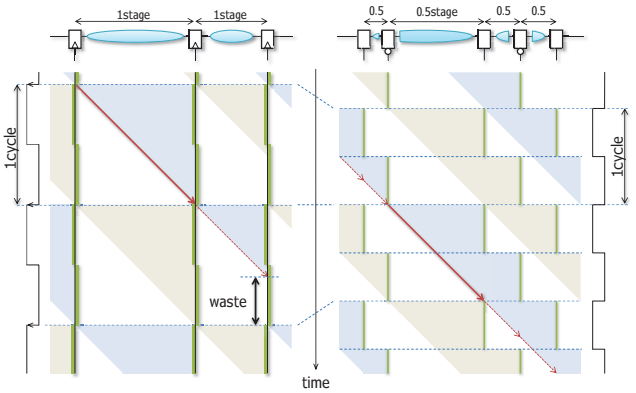

図 3 静的タイム・ボローイング

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．

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．

2.4 Razor

図 5（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 4 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．

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

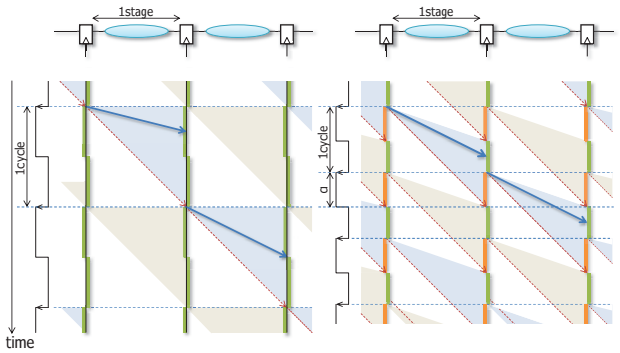

図 4 単相 FF（左）と Razor（右）の t-diagram

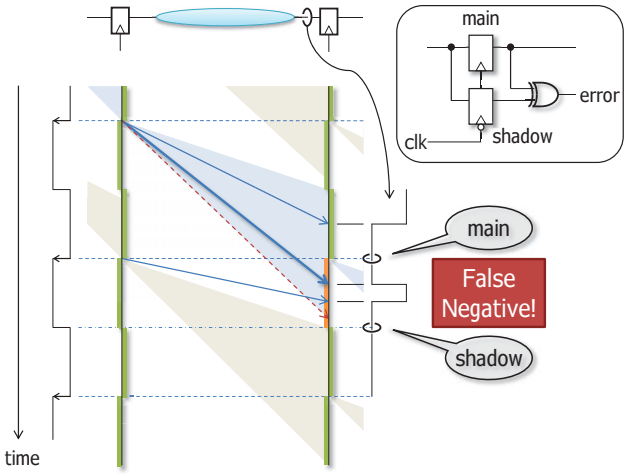

図 5 Razor の回路構成とショート・パス問題

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．

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 5 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．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，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．

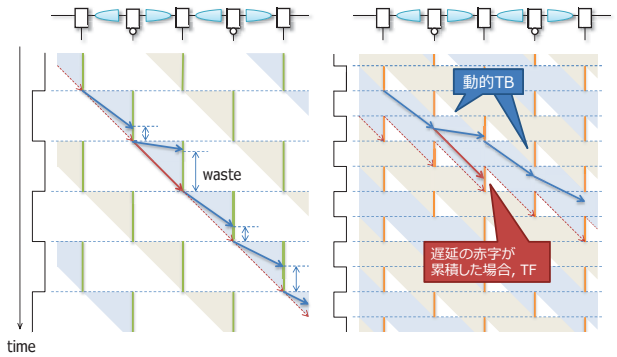

図 6 二相ラッチ方式（左）と提案手法（右）の t-diagram

図 5 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 節で動的タイム・ボローイングについて詳しく取り上げる．3.2 節で既存の TF 検出手法である Razor と比較して，提案手法の特徴や優位性を示す．3.3 節で提案手法の回路構成を述べる．

3.1 動的タイム・ボローイング

図 6 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 7** において説明する．図 7 は提案手法の t-diagram を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**，FF やラッチにより信号の変化が次ステージに伝搬しない時間 $I(i)$ を遅延の**収入**と定義する．単相 FF 方式や Razor の場合，1 ステージ毎に $I(i) = \tau$，二相ラッチ方式や提案手法の場合，0.5 ステージ毎に $I(i) = 1/2\tau$ 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．この時，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．この時，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．この時，$D(2) = 1/2\tau - 1/8\tau = 3/8\tau$ 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 6 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

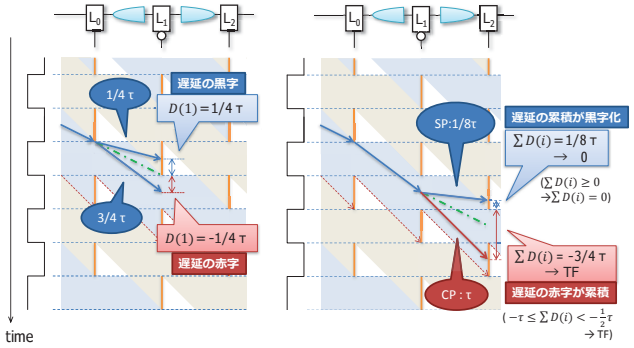

図 7 動的タイム・ボローイング

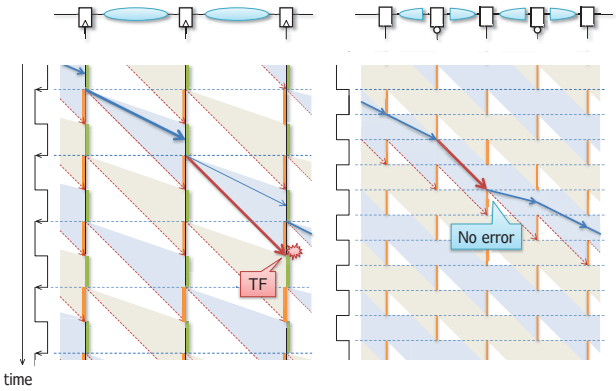

図 8 Razor（左）と提案手法（右）の t-diagram

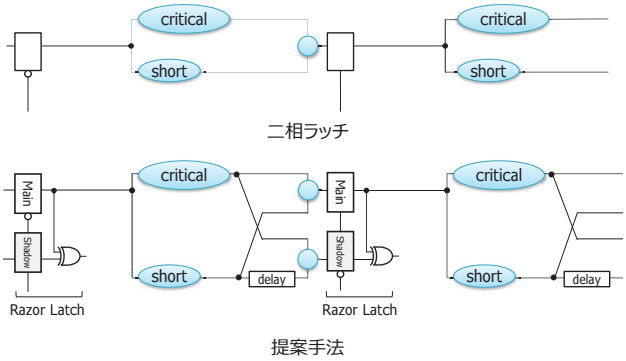

図 9 提案手法の回路構成

定める（図 6 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．

3.2 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 8** 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 ステージ毎に $I(i) = \tau$ の遅延の収入がある．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，その回復オーバー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，複数ステージ間に渡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．

3.3 回路構成

図 9 は提案手法の回路構成である．図 9 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 9 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．

ゲートの二重化

2.4 節で述べたように，TF 検出により最大遅延制約を緩和するためには，検出期間分の最小遅延制約を満たすように，ロジックのショート・パスに遅延を挿入し，検出期間を確保しなければならない．

このことは提案の根幹である動的タイム・ボローイングの効果を薄めてしまう．**図 10**(左) は，単純にロジックのショート・パスに遅延を挿入した場合の t-diagram を表し

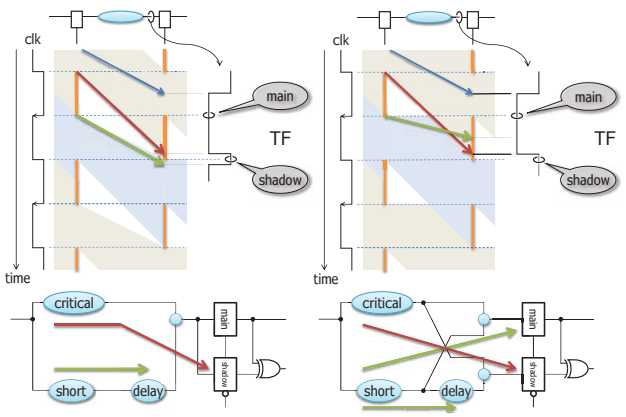

図 10　二重化の必要性

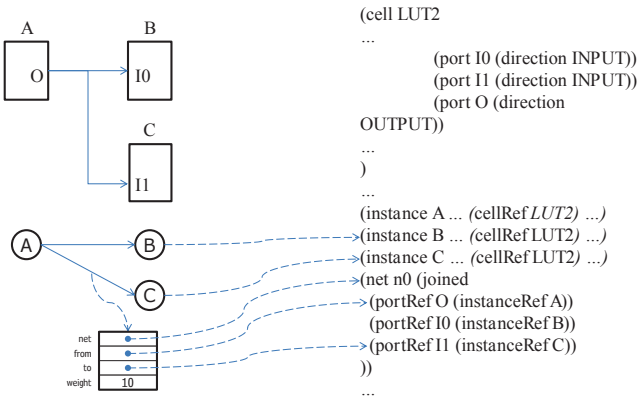

図 11　グラフへの変換

ている．図中青色の領域を見ると，ロジックの最小遅延が検出ウィンドウの分だけ伸びてしまっている．動的タイム・ボローイングの効果とは，実効遅延の値が小さいことで遅延の黒字を生み，遅延の赤字の累積を解消することで，実効遅延の平均に基づく動作を実現することにある．これでは，遅延の黒字を生むパスが存在しないことになり，動的タイム・ボローイングの効果を期待できなくなる．

ここで，TF 検出は，検出期間中に変化した Shadow Latch の値と，検出期間前に保持された Main Latch の値を比較することで行われることに着目する．すなわち，検出期間中は，Shadow Latch が開いており，Main Latch は閉じている．このことから，Razor のショート・パス問題の本質は，Shadow Latch の値がショート・パスの活性化によって変化してしまうこととわかる．

この点に着目し，提案の回路は，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，Main Latch と Shadow Latch に至るパスの経路を分離する構成を取る．Shadow Latch のショート・パスにのみ遅延を挿入することで，検出区間を確保する．Main Latch のショート・パスには遅延が挿入されてないので，ロジックの最小遅延は保たれる．

図 10(右) には提案の回路構成でショート・パス問題の対策を行った場合の t-diagram である．t-diagram においてはフェーズが「混ざって」いるように見えるが，各フェーズの信号の経路の違いにより，実際にはフェーズの信号が「混ざる」ことはない．

4.　適用自動化ツール

我々は以前，リプルキャリー・アダーやキャリールックアヘッド・アダーを用いたアップ・カウンタのような比較的簡単な回路に対する提案手法の適用を，手作業により行い，評価を行った [7]．しかし，プロセッサのように回路規模が大きいものに対する適用を手作業で行うのは，不可能に近い．

そこで現在，汎用の回路をグラフに落とし込み，提案方式の適用を自動で行うツールの開発を進めている．本章では，このツールの構成法について述べる．

このツールはグラフへの変換，ステージ分割，二相ラッチ化，TF 検出機構付与という 4 つのプロセスから構成される．以下，各プロセスの仕様，それを満たすための要件を列挙し，動作を実現するアルゴリズムを述べる．

4.1　グラフへの変換

適用ツールではまず，EDIF（Electronic Design Interchange Format）形式で書かれた汎用の回路を読み込み，FF や入力/出力ポート，ゲートといった回路中の素子をノード，素子間の配線をエッジとするグラフへと変換する．このグラフを元に，提案方式の適用を行っていく．

適用後は，グラフから EDIF への書き戻しを行う．そのために，グラフと EDIF との対応関係を明確にする必要がある．**図 11** にその様子を示す．EDIF は S 式のリスト構造で記述されている．これをリストのデータ構造に入れ，ノードやエッジに持たせる．これにより，提案方式を適用した後のグラフから，EDIF に書き戻すことも可能になる．

4.2　ステージ分割

先に述べた二相ラッチ化や Razor の挿入といった提案方式の適用は，各ステージごとに行なわれる．そのために，**図 12** に示すように，回路全体のグラフを，FF や入力/出力ポートに挟まれたロジック部分で切り出し，ステージ単位のグラフに分割する必要がある．

以下，FF や入力/出力ポートのノードを**境界ノード**，ロジックを構成する素子のノードを**ロジックノード**と呼ぶことにする．

満たすべき要件

回路全体のグラフを正しくステージ単位のグラフに分割するためには，バックエッジやフォワードエッジが存在し，どのようにステージ分割すべきかわかりにくいロジックに対しても正しく分割を行う必要がある．**図 13** にその例を示す．

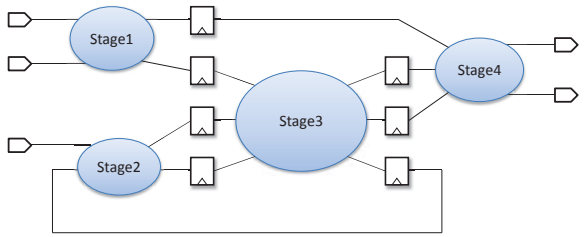

図 12 ステージに分割された回路

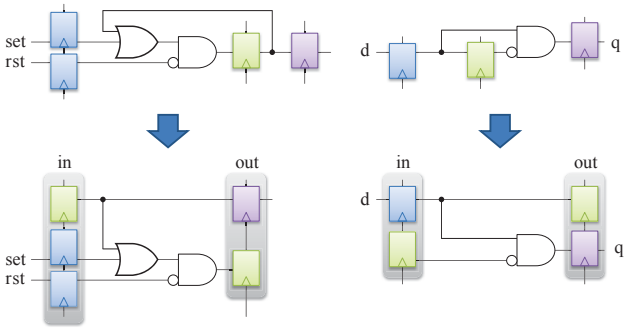

図 13 バックエッジ・フォワードエッジへの対応

図 13 上に示す二つの回路では，青色の FF と緑色の FF に挟まれたステージ，緑色の FF と紫色の FF に挟まれたステージの 2 ステージが存在し，バックエッジやフォワードエッジによって 2 つのステージが混ざっているかのように見える．

しかし，実際には，これは 1 つのステージであり，緑色の FF がこのステージの入力であり，出力でもある．緑色の FF を同ステージの入力側と出力側の境界ノード両方に振り分けることで，回路全体を 1 ステージとして，正しくステージ分割を行うことができる．

アルゴリズム

以下に，ステージ分割のアルゴリズムを示す．

(1) 任意の境界ノードを 1 つ選ぶ．
(2) その境界ノードからパスを辿り，同ステージに含まれるノードを探索する．辿ったノードがロジックノードであった場合は，エッジの方向に関わらず，パスを探索する．
(3) 境界ノードに到達した場合，境界ノードが入力側にあるか，出力側にあるかを見て，どちらかのリストに加える．入力側にあった場合は，(1) の行程でその境界ノードが選ばれないようフラグを付ける．出力側にあった場合は，フラグはつけない．その後，境界ノードへ辿った方向とは逆の方向へ戻り，探索を続ける．
(4) 全てのパスの探索が終わったら，今までに辿ったロジックノードを同ステージ内に含まれるロジックノードとしてまとめる．
(5) (1) に戻り，フラグのついていない境界ノードから別のステージを再度探索する．
(6) 回路中全てのノードについてステージ分割が行われたら，処理を終了する．

適用例

上記で述べたアルゴリズムを基に，**図 14** を例に，実際にステージ分割を行う．図中○印はロジックノードを，□印は境界ノードを表している．図 14(0) に，境界ノード d からバックエッジが，境界ノード f から e にかけて，フォワードエッジがある場合を示し，これをステージに分割する．

(1) ［図 14(1)］任意の境界ノードとして，a を選んだとする．境界ノード a に繋がるエッジを辿り，次のノードを探索する．正順，逆順の両方から次のノードを辿ると，境界ノード b と c にぶつかる．そして，b を入力側，c を出力側のリストに含め，探索される境界ノードから b を除外する．
(2) ［図 14(2)］境界ノード c から新たなステージを探索する．c から，(2) と同様に探索すると，境界ノード d が入力側，出力側双方からぶつかる．このため，d を入出力両方のリストに加え，探索される境界ノードから d を除外する．
(3) ［図 14(3)］d から次のノードに移り，(2) と同様に境界ノードを探索する．f, g を入力側に，e を出力側のリストに含め，探索される境界ノードから f, g を除外する．同様にして，g, h を出力側に含める．ステージ分割が行われたら，処理を終了する．

以上のようなステージ分割により作成したリストから生成したグラフが図 14(4) である．図 14(0) の回路と等価な回路であり，実際にステージに分割できることが分かる．

4.3 二相ラッチ化

FF のステージ単位に分割したグラフに対し，二相ラッチ化を行う．二相ラッチ化は，FF のステージ内のロジックに逆相のラッチを挿入し，FF のステージ単位に分割したグラフを二分することと言える．

満たすべき要件

二相ラッチ化を行う際に満たすべき要件は，以下の 2 つにより表される：

(1) **どのパス上を通ってもラッチの挿入位置が必ず「1 つだけ」存在する**

二相ラッチ化に伴うラッチの挿入位置の探索に際し，グラフの有効なカットを求めるアルゴリズムがそのまま適用できるかに思える．しかし，有効なカットが正しいラッチの挿入位置になるとは限らない．**図 15** にその例を示す．青太線はカットの位置を表している．

図 15 上に示す例はカットとしては有効であるが，二相ラッチ化は正しく行われていない．入力側の境界ノード i_1 から，出力側の境界ノード o_2 に至るパス上に 3 つのカットが存在する．このカット上に逆相ラッチを挿入すると，同パス上に逆相ラッチが複数配置され，回路が正しく動作

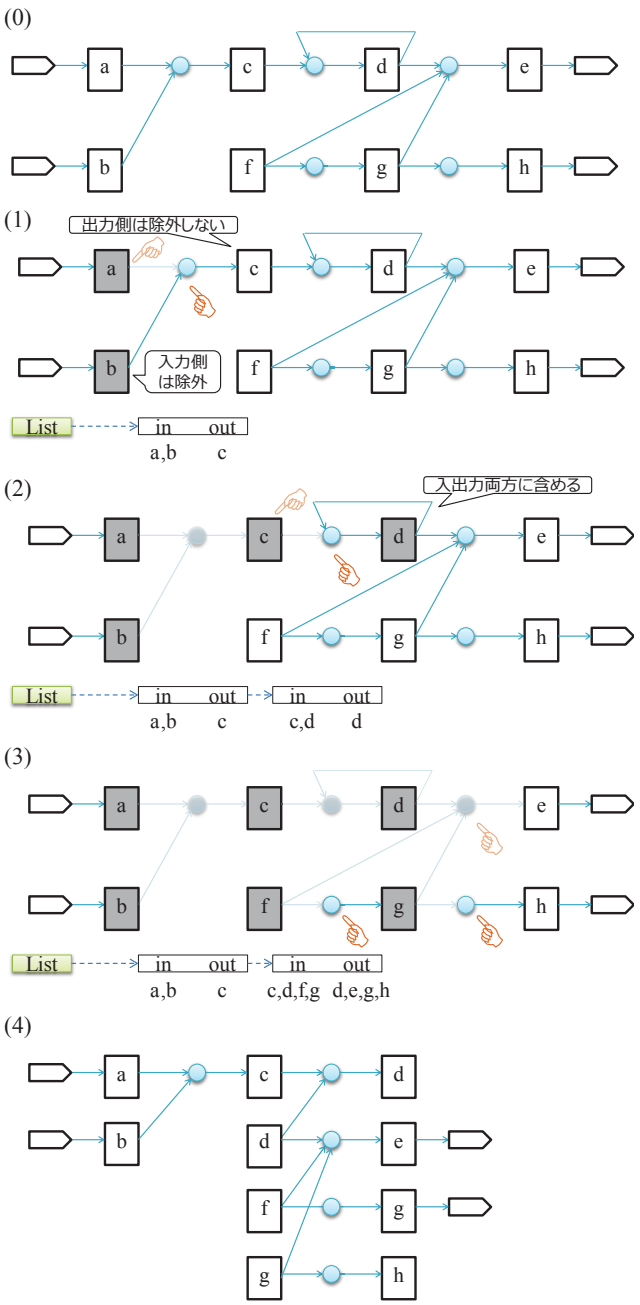

図 14 ステージ分割の流れ

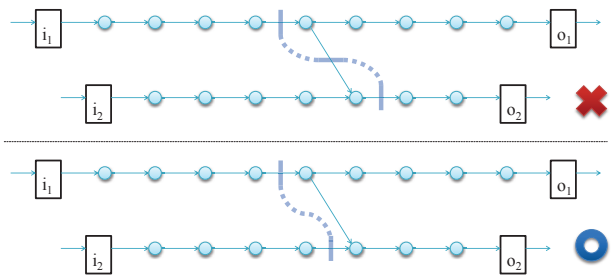

図 15 二相ラッチ化の実際

しない．

回路が正しく動作するためには，図 15 下に示す例のように，どのパスを通っても，カットが一つだけ存在するように，挿入位置を定めなければならない．

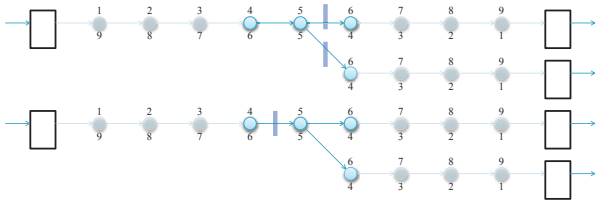

図 16 挿入箇所によるラッチの削減

(2) **二分したロジックのクリティカル・パス遅延が均等である**

提案方式では，ラッチの挿入により二分されたステージのクリティカル・パス遅延によって，サイクル・タイムが定まる．クリティカル・パス遅延がインバランスになるようにラッチを挿入すると，遅延の大きい方のステージのクリティカル・パス遅延により，サイクル・タイムが規定されてしまい，サイクル・タイムが短縮できない．

そのため，二分したロジックのパス遅延が均等になるような適切な位置にラッチを挿入しなければならない．

適用ツールでは，ロジックのパス遅延を Bellman-ford 法 [11] を用いて計算し，適切なラッチの挿入位置を探索している．二相ラッチ化には，クリティカル・パス遅延の値を元に閾値を定めるため，最長距離の計算を行う．Bellman-ford 法は最短距離を求めるアルゴリズムであるが，パラメータに負の値を取ることで最長距離も求めることができる．最短距離の計算は後の TF 検出機構付与のプロセスで使用する．

(3) **挿入するラッチの数が少なくなるようにする**

ラッチの挿入位置によって，挿入するラッチの数が少なくなる場合がある．**図 16** の二つの例は，共に正しく二相ラッチ化が行われている．しかし，図 16（下）のように，パスの分岐の手前にラッチを置くことにより，ラッチの数を 2 から 1 に減らすことができる．

このため，探索した挿入位置の候補から，挿入するラッチの数が少ないものを選ぶ．

探索空間の削減

先に述べた条件が満たされていさえすれば，二相ラッチ化は正しく行われるが，全ての挿入位置を探索するとなると，探索空間が膨大になる．

本ツールでは，パス遅延による挿入位置の自由度の違いに着目し，探索空間を抑える．以下では**図 17** を用いて，このことを説明する．ロジックノードの上に付随する数字は，入力側の境界ノードからの最長距離を示している．

先に述べた制約により，クリティカル・パス上のラッチの挿入位置（図 17 上）は，自由度が少ない．これに対し，ショート・パス上のラッチの挿入位置（図 17 下）は，二分したステージのクリティカル・パス遅延を超えないように分割すればよく，自由度は大きい．クリティカル・パス遅延よりショート・パス遅延が小さいほど，自由度は大きくなる．

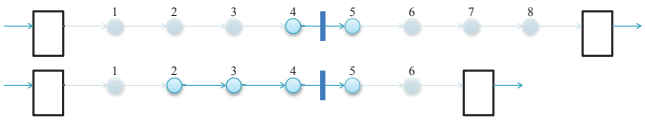

図 17 パス遅延による挿入箇所の自由度の違い

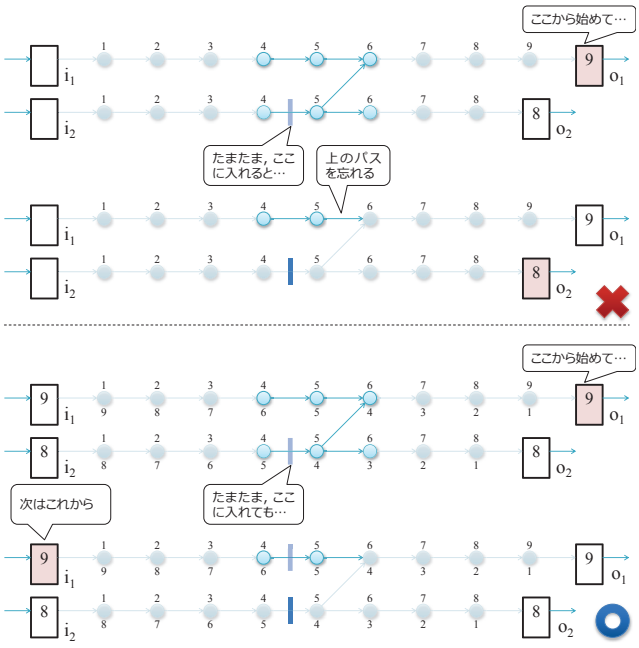

図 18 探索漏れの防止

このため，クリティカル・パスといった，遅延の大きいパスから順にラッチの挿入位置を定め，ラッチの挿入位置をある程度限定した後，ショート・パス上のラッチの挿入位置を定めることで，探索空間を抑えることが可能となる．

探索漏れの防止

挿入位置を決定する際に，出力側の境界ノードから挿入位置を探索しただけでは，ラッチの挿入位置に漏れが発生する場合がある．以下では**図 18** を用いて，このことを説明する．

図 18 上は，出力側の境界ノードからのみ探索した場合を表している．境界ノード o_1 に至るパスを探索し，青太線の位置にあるエッジを挿入位置として選んだとする．そのエッジからこれ以上ラッチを挿入する必要のないパスを探索し，候補から除外する．次に，境界ノード o_2 に至るパスを探索すると，パス上の全てのエッジがラッチを入れる必要がないことが分かり，探索が終了してしまう．しかし，挿入すべき個所は残存しており，このままでは正しく二相ラッチ化を行えない．

これを防ぐために，出力側の境界ノードからの最長距離も求め，入力側の境界ノードからもパスを探索する．図 18 下にこの様子を示す．ロジックノード下に付随する数字は，出力側の境界ノードからの最長距離を示している．境界ノード o_1 に至るパスを探索し，候補を除外した次に，境界ノード i_1 から出るパスを探索する．このようにすることで，挿入漏れを回避することができる．

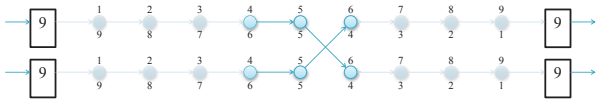

図 19 同時に選んだ場合

なお，探索の際に入力側と出力側の境界ノードを同時に選ぶのは得策ではない．**図 19** のように，距離が最大の入力側，出力側の境界ノードを選んでも，その間にパスがあるとは限らないからである．

アルゴリズム

これまでに述べた論点を踏まえ，以下に示すアルゴリズムにより，二相ラッチ化を行う：

(1) 境界ノードが FF である場合に，正相ラッチへと変換する．

(2) 入力側の境界ノードから出力側の境界ノードへと探索を行い，入力側の境界ノードから各ロジックノードに至るパスの最長距離を計算する．同様に，出力側の境界ノードから入力側の境界ノードへと逆順に最長距離を計算する．

(3) 境界ノードの持つ最長距離の最大値を，ステージのクリティカル・パス遅延 d_{max} とし，閾値 $d_{th} = 1/2d_{max}+1$ を求める．そして，d_{th} 以上の最長距離を持つノードから出るエッジをラッチの挿入位置候補から除外する．

以上までの行程を済ませたグラフを**図 20** に示す．以下，図 20 を基に挿入位置を決定するアルゴリズムを説明する．

(4) ［図 20(a)］入力側の境界ノードと出力側の境界ノードを，最長距離の大きい順にオーダリングする．最長距離の大きな境界ノードから順に 1 つ選ぶ．パス上の挿入位置を探索し，対応する箇所のエッジを 1 つ選ぶ．

(5) ［図 20(b)］そのエッジから入力側，出力側双方に対して探索を行い，ラッチを挿入する必要のないエッジを除外する．この探索時にぶつかる境界ノードは既にラッチが挿入されているパスを持つ．そのため，探索候補から取り除く．

(6) ［図 20(c)］探索候補に残っている境界ノードで遅延の大きいものを選択し，(4)-(5) を繰り返す．全てのエッジにラッチを挿入する必要がなくなったら，探索を終了する．

(7) (4)-(6) を繰り返し，全ての挿入候補を上げたのち，ラッチの一番少ないものを選ぶ．その挿入箇所のエッジに逆相のラッチを表す境界ノードを挿入し，グラフを 2 分する．

以上の行程で，二相ラッチ化を終了する．以降の行程では (7) で作成したグラフを用いる．

4.4 TF 検出機構の付与

二相ラッチ化を行ったグラフに対し，TF 検出機構を付与する．そのために，Razor の挿入や，遅延素子の挿入，

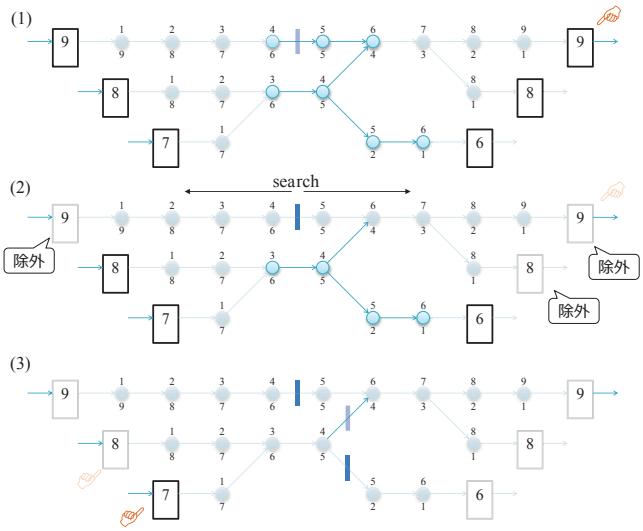

図 20 二相ラッチ化の流れ

ゲートの二重化といった措置を行う必要がある．

満たすべき要件

TF 検出機構を付与するために満たすべき要件は，以下の 3 つである：

(1) **TF を引き起こす可能性のあるパスが存在する出力側のラッチを Razor Latch に変更する**

一般に，TF 検出機構は，クリティカル・パスの遅延を検出限界として定める．検出ウィンドウ幅を W，ステージのクリティカル・パス遅延を d_{cp} とすると，Razor Latch の変更をするかを決める閾値 d_{razor} は，$d_{cp}-W$ で表される．出力側のラッチに至るパスの最大遅延を d_{max} $(0 < d_{max} \leq d_{cp})$ とすると，Razor の挿入条件は，$d_{max} \geq d_{razor}$ で表される．

(2) **検出ウィンドウ幅を超える遅延になるように，ショート・パスに遅延を挿入する**

検出ウィンドウを確保するために，ショート・パスの遅延を検出ウィンドウ幅 W 以上にする必要がある．このため，遅延を入れるかを決める閾値 d_{th} は，$d_{th} \geq W$ で表される．ショート・パスの遅延を d_{sp} とすると，検出ウィンドウを確保するために，$d_{sp} \leq d_{th}$ のパスに対し，最低 $d_{th} - d_{sp}$ の遅延を挿入する必要がある．

(3) **二重化ゲートの数ができるだけ少なくなるようにする**

回路面積のオーバーヘッドを抑えるために，できるだけ二重化するゲートを少なくしたい．入力側に遅延素子を挿入した場合，遅延が挿入されているパスとされていないパスを分けるためには，遅延素子以降に存在するロジックノード全てを二重化しなければならない．このため，ステージの出力側にできるだけ近いところに遅延素子を挿入することで回路面積の増加を軽減する．

アルゴリズム

以下，**図 21**，**図 22** のグラフを基に，TF 検出機構を付与するアルゴリズムを説明する．

(1) 入力側の境界ノードから，各ノードの最短経路 d_{min} と最長経路 d_{max} を Bellman-ford 法により求める．
［図 21(1)］各ノードの上部左側が d_{min}，右側が d_{max} を表す．

(2) 出力側の境界ノードの中で最長な距離をクリティカル・パス遅延 d_{cp} とし，これを基に閾値を求める．
［図 21(1)］この例では，$d_{cp} = 4$ であるから，$d_{razor} = 2$ 以上のラッチを Razor に変え，$d_{th} = 3$ 以下のショート・パスに遅延を挿入することとする．

(3) 出力側の境界ノードを任意に一つ選ぶ．この境界ノードの最長径路 d_{max} を見ることで，Razor に変更するかを決める．Razor に変更する必要があった場合，Shadow Latch を付与し，手前のノードからエッジをつなぐ．そして，入力側の境界ノードに向かって，深さ D を計算する．
［図 21(2)］選ばれた境界ノードを橙色で示す．ここでは，$d_{max} > d_{razor}$ であるから，Razor に変更が必要であるため，Shadow Latch を付与する．

(4) 出力側の境界ノードから入力側の境界ノードへ向かって，深さ優先でロジックノードを探索し一つ選ぶ．そのロジックノードがショート・パスを含むかどうかを判断する．ロジックノードの最小遅延 d_{min} と深さ D_{self} の和から 1 を引いたものが，d_{th} よりも小さければそのロジックノードはショート・パスを含んでいる．ショート・パスを含んでいた場合，そのロジックノードを二重化する．
［図 21(3)］ここでは，ロジックノード a が選ばれる．今，ロジックノード a では，$d_{min}+D_{self}-1 = 2 < d_{th}$ であるから，ショート・パスを含んでいる．このため，ロジックノード a を二重化する．

(5) 二重化したロジックノードに至るパスの中から，ショート・パスを探索し，遅延を挿入する．入力側に向かってノードを 1 つ分辿る．そのノードの最大遅延 d_{max} と，二重化したロジックノードの深さ D_{prev} の和が，d_{th} より小さければ，そのエッジはショート・パスであることが分かる．ショート・パスであることが分かったら，d_{th} から，辿ったノードの深さ D_{self} を引いた数だけ，二重化したエッジに遅延を挿入する．
［図 21(4)］ロジックノード a から入力側に向かってノードを 1 つ分辿ると，ロジックノード b，c が見つかる．このそれぞれに対し，$d_{max} + D_{prev}$ を計算すると，ロジックノード $a-b$ 間のエッジがショート・パスであり，ロジックノード $a-c$ 間のエッジがクリティカル・パスであることが分かる．ロジックノード b の深さ D_{self} を d_{th} から引いた値は 1 なので，ロジックノード $a-b$ 間の複製したエッジに遅延を 1 つ挿入する．

(6) 遅延挿入を行ったパス上のノードは探索から除外して，

次のロジックノードを探索し，(4)-(6) を繰り返す．そして，このロジックノードが二重化の必要があり，前に二重化されたロジックノードとのエッジが存在する場合は，そのエッジを二重化したロジックノードに繋ぎかえる．

［図 22(5)］先程ロジックノード b は遅延挿入を行ったので，ロジックノード c へ移動する．そして，ロジックノード c に対し，二重化と遅延挿入を行う．ロジックノード c はロジックノード a の二重化した方と繋がっているので，c の二重化の際に，そのエッジをロジックノード c の二重化した側に繋ぎかえる．

(7) (4) で述べたショート・パスの式を満たさないノードに到達した場合，これ以降のパスの探索をやめ，(3) で選んだ境界ノードからのパスに対する処理を終了する．

［図 22(6)］ロジックノード d へと移動する．ロジックノード d においては，$d_{min} + D_{self} - 1 = 3 \geq d_{th}$ であるから，このノードにはショート・パスが存在しないことが分かる．このため，ロジックノード e への探索をやめる．

(8) ［図 22(7)］別の境界ノードを選び，(3) の行程から再度やり直す．前の行程で二重化や遅延挿入を行ったノードに対しても，再度同様の処理を行う．

以上のアルゴリズムで，全てのパスに対し，TF 検出機構を付与する．図 21(1) に対し，TF 検出機構を付与し終わったグラフは図 22(8) である．図からわかる通り，Main Latch に至るパスには遅延は挿入されておらず，Shadow Latch に至るパスには全て遅延が挿入され，最小遅延制約を満たしていることが分かる．

5. おわりに

近年問題となりつつある LSI のばらつきへの対策として，タイミング・フォールトの検出/回復技術がある．我々は以前，この技術をさらに推し進めた手法として，動的タイム・ボローイングを可能にするクロッキング方式を提案している．これは入力ばらつきに着目することにより，既存の手法では成し得なかった，クリティカル・パスよりはるかに短い実効遅延に基づくサイクル・タイムを規定できるクロッキング方式であり，二相ラッチ方式にタイミング・フォールト検出技術を組み合わせることにより実現される．本稿では，この手法を単相 FF 方式の汎用な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述べた．

適用自動化ツールは，EDIF 形式の回路をグラフに変換し，そのグラフをステージごとに切り出す．その後，切り出したグラフに基づいて，二相ラッチ方式への変換と，タイミング・フォールト検出機構の付与を行うといったプロセスを経る．各プロセスの仕様，それを満たすための要件を列挙し，動作を実現するアルゴリズムを述べた．

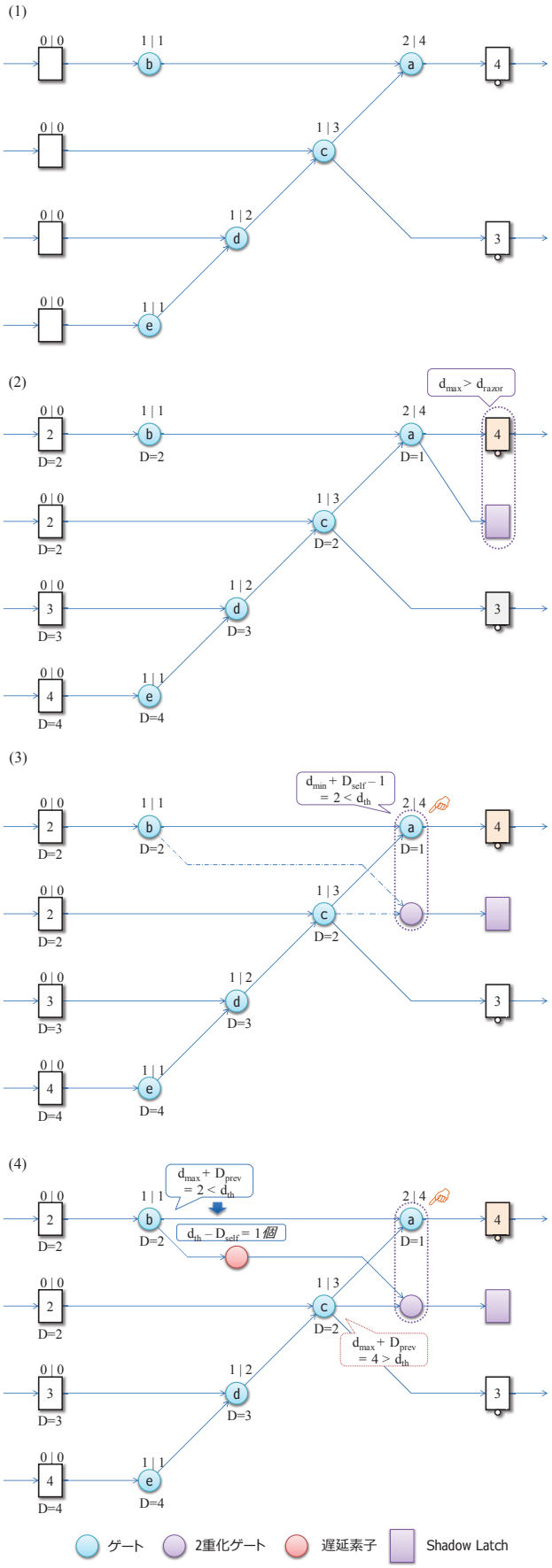

図 21 TF 検出付与の流れ (1-4)

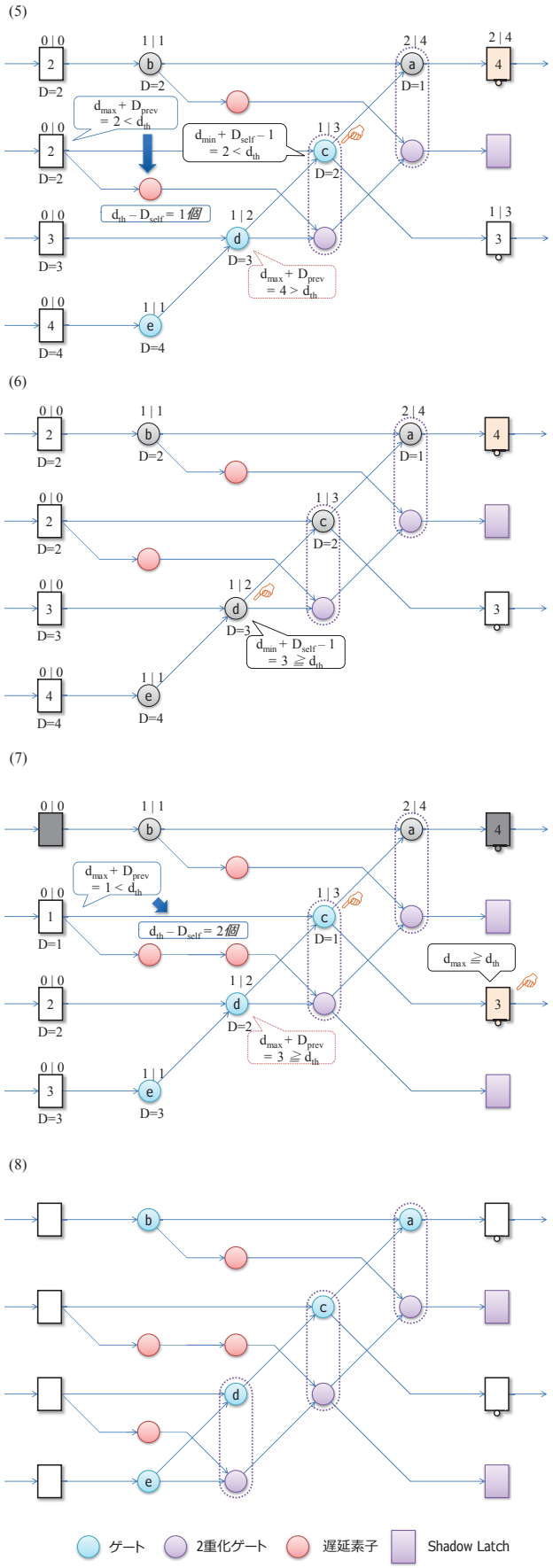

図 22 TF 検出付与の流れ (5-8)

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 本研究の一部は，文部科学省科学研究費補助金 No. 23300013 による．

参考文献

[1] 岡田健一：集積回路における性能ばらつき解析に関する研究，博士論文，京都大学 (2003).

[2] 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).

[3] 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).

[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 (online), DOI: 10.1109/ISSCC.2010.5433919 (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] 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).

[8] 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).

[9] Harris, D.: Skew-tolerant Circuit Design, *Morgan Kaufmann Publishers*, pp. 12–14 (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 (ACS)，Vol. 6, No. 1, pp. 17–30 (2013).

[11] Kleinberg, J. and Tardos, E.: *Algorithm Design*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2005).