タイミング制約違反を利用する設計手法とコ・シミュレーション環境による評価

国武, 勇次
九州工業大学情報工学研究科情報科学専攻

千代延, 昭宏
富士通研究所

田中, 康一郎
九州産業大学情報科学部知能情報学科

佐藤, 寿倫
九州大学システムLSI研究センター

出版情報：電子情報通信学会技術研究報告. DC, ディペンダブルコンピューティング. 107 (174), pp.31-36, 2007-08-02. 電子情報通信学会
バージョン：
権利関係：

タイミング制約違反を利用する設計手法と コ・シミュレーション環境による評価

国武勇次† 千代延昭宏†† 田中康一郎††† 佐藤寿倫††††

† 九州工業大学 情報工学研究科 情報科学専攻
†† 株式会社 富士通研究所
††† 九州産業大学 情報科学部 知能情報学科
†††† 九州大学 システム LSI 研究センター

あらまし 半導体技術の微細化に伴う素子のばらつき増大により，設計制約が非常に厳しくなっている．従来行われてきた保守的な最悪ケース指向設計が必要とする設計マージンが大きくなっているためである．我々は最悪ケースに囚われるのではなく，典型的ケースを配慮することで設計制約を緩和する手法として，建設的タイミング違反方式 (Constructive Timing Violation: CTV) を検討している．本手法を評価するにあたり，回路遅延を考慮した評価環境が必要になる．本稿では，ゲートレベル・シミュレーションとアーキテクチャレベル・シミュレーションのコ・シミュレーションによる評価環境を構築し，CTV とその性能改善手法の評価を行い，マイクロプロセッサに与える効果を明らかにする．

キーワード ばらつき，コ・シミュレーション，最悪ケース指向設計，典型的ケース指向設計，信頼性

A Typical-case Design Methodology Mitigating Timing Constraints and its Evaluation via Co-Simulations

Yuji KUNITAKE†, Akihiro CHIYONOBU††, Koichiro TANAKA†††, and Toshinori SATO††††

† Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
†† Fujitsu Laboratories Ltd.
††† Faculty of Information Science, Kyushu Sangyo University
†††† System LSI Research Center, Kyushu University

Abstract The deep submicron semiconductor technologies have increased process variations. They make worst–case designs impossible. This is because larger variations require larger design margins. In order to realize robust designs, we have to design LSIs by considering typical-cases rather than worst cases. We are investigating such a typical-case design methodology, which we call Constructive Timing Violation (CTV). In order to evaluate the CTV, we have to consider circuit delay. We build a co-simulation environment by combining gate level simulation with architectural level simulation. We evaluate the CTV and its enhanced techniques by the co-simulation environment.

Key words parameter variations, co-simulation, worst-case design, typical-case design methodology, reliability

1. まえがき

半導体製造技術が物理的な微細化の限界に近付くにつれ，製造プロセスのばらつきが無視できない程まで大きくなり，問題となっている．また，微細化による熱密度の増加は省電力化の要求を高め，電源電圧を降圧する傾向にあるため，電源電圧の揺らぎが深刻になっている．さらに，消費電力の分布の不均一がチップ内の温度ばらつきを増加させている．これらの要因による素子特性のばらつきによって設計制約が厳しくなり，これまで行われてきた保守的な最悪ケースを考慮した設計が困難になっている．

そこで我々は，典型的ケースを配慮して設計を行うことで最悪ケースに配慮する設計マージンを取り除き，設計容易さの向上を目指す．ばらつきを考慮するために必要なマージンを取り除く設計手法として，我々は建設的タイミング違反方式 (Constructive Timing Violation: CTV) [6], [7] を検討している．LSI を正常に動作させるための設計制約を緩和することで生じる故障状態に対し，フォールトトレラント機構を備えることで，その動作を保証する方式である．

CTV は遅延ばらつきに配慮する設計手法であるため，評価

を行うためには適用回路の遅延を考慮したタイミングシミュレーションが必要である．回路遅延を考慮したシミュレーションはゲートレベル・シミュレーションが一般的である．しかし実行時間が非常に長く，またアーキテクチャの評価を行う上でプロセッサ全体のタイミングシミュレーションは必要ない．そこで，遅延情報テーブルと呼ぶテーブルにより遅延情報を考慮したアーキテクチャレベルのシミュレーション環境の構築を目指した．しかし遅延情報テーブルでは，1 サイクル前の出力信号値に依存して変化する遅延をアーキテクチャレベル・シミュレーションで再現出来なかった [2], [5]．そこで本稿では，ゲートレベル・シミュレーションとアーキテクチャレベル・シミュレーションのコ・シミュレーションにより CTV の評価を行い，本設計手法とその性能改善手法の効果を明らかにする．

以下本稿の構成について説明する．2 章で典型的ケース指向設計とその例である CTV について述べる．3 章で CTV を評価するための評価環境について述べる．4 章でコ・シミュレーションによる CTV の評価を紹介する．5 章で今後の課題について述べ，6 章で本論文をまとめる．

2. 典型的ケース指向設計

半導体製造技術の微細化により素子特性のばらつき問題が無視できない程まで深刻化している．素子特性のばらつきが大きくなると，ばらつきの最悪ケースを予測することが難しくなり，従来行われてきた最悪ケース指向設計のための設計マージンの確保が困難となる．このような背景から，設計容易さを向上するような設計手法が求められている．

我々は，この設計容易さを向上させる設計手法として，典型的ケース指向設計を検討している．典型的ケース指向設計では，従来の最悪ケース指向設計のように最悪ケースに囚われることなく，典型的ケースに配慮する．これは，最悪ケースが実際に起こるのは稀であるためである．最悪ケースに配慮する必要がなくなれば，設計制約が大きく緩和され設計が容易になるとともに，その期間も短縮できると期待される．

典型的ケース指向設計において，設計者は一つの機能に対して二つの回路を実装する．一つは，性能を重視した設計 (性能指向設計) で典型的ケースのみが考慮される．つまり最悪ケースに配慮する必要がなく，設計制約を緩和することができるために設計が容易に行える．もう一つは，機能を保証する設計である (機能保証設計)．ここでは，最悪ケースに配慮しなければならないが，性能に配慮する必要がない．つまり性能向上のための最適化を行う必要がなくなる．このためにシンプルな設計が行え，その検証も容易に行える．

我々の検討している CTV は，この典型的ケース指向設計の概念に基づいた設計手法である．次節で CTV の詳細について述べる．

2.1 建設的タイミング違反方式

CTV で入力依存による遅延ばらつきを利用する方法を説明する．同期回路ではクロックに同期して回路の出力信号が FF に取り込まれるが，出力が確定するまでの遅延は一定ではない．これらのなかで最も大きい遅延を決定するパスをクリティカルパスと呼ぶ．従来の最悪ケース指向設計ではクリティカルパス遅延を考慮して周波数が決定される．しかし，クリティカルパスを活性化する入力は稀にしか発生しない．CTV はこの回路遅延の特徴を利用している．例えば以下のような応用が考えられる．電源電圧を下げるとパスの遅延は増加する．遅延が増加すると，クリティカルパス付近のパスはタイミング制約を満たすことが出来なくなりタイミングエラーを起こす．しかし，タイミングエラーが起きるのは稀であり，ほとんどのパスは正常な動作が期待できる．つまり，性能を維持しつつ消費電力を抑えることが出来る．発生するタイミングエラーに対してはプロセッサ状態を回復する機能が必要となる．図 1 に CTV の概要を示す．

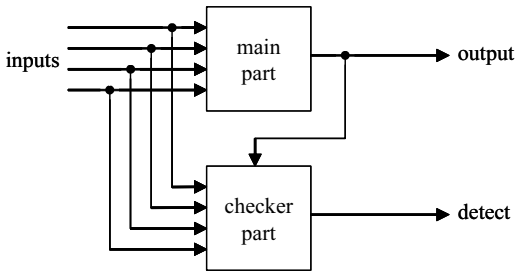

図 1　建設的タイミング違反方式

CTV 方式で設計された回路はメイン部 (図 1 の `main part`) とチェック部 (図 1 の `checker part`) から構成される．メイン部では省電力かつ高性能な設計が行われる．つまり，電源電圧が動作保証する範囲を越えた高い周波数が供給される．このため，クリティカルパスに近い回路遅延を持ったパスはタイミングエラーを起こす可能性がある．一方，チェック部はタイミングエラーを検出する機構として用意される．機能はメイン部と同じであるが，常に正しい値が求められるため周波数はクリティカルパスの遅延を満たすように設計される．しかし, 性能を考慮する必要がないので設計は容易になる．

チェック部によってタイミングエラーが検出された場合，回路の状態を回復させるための回復機構が必要となる．マイクロプロセッサの場合は容易である．ある命令でタイミングエラーが検出された場合，当該命令に依存する命令は正しい実行結果を得るために再実行されなければならない．このプロセスはマイクロプロセッサに既に備わっている投機実行失敗からの回復に類似する．したがって，タイミングエラーを起こした命令を投機実行に失敗した命令に読み替えることで，マイクロプロセッサを正しい状態に回復できる．

2.2 CTV を適用した ALU

CTV の例を紹介する．図 2 は CTV を適用した ALU である．checker part と書かれている点線内部は従来の ALU に対して CTV を適用したときの追加回路である．各回路はメイン部 (`main part`) とチェック部 (`checker part`) で構成される．チェック部には二つの ALU が用意され，それぞれにタイミング制約を満たした周波数 f_L が供給される．つまり，チェック部は動作保証されているためタイミングエラーを発生しない．一

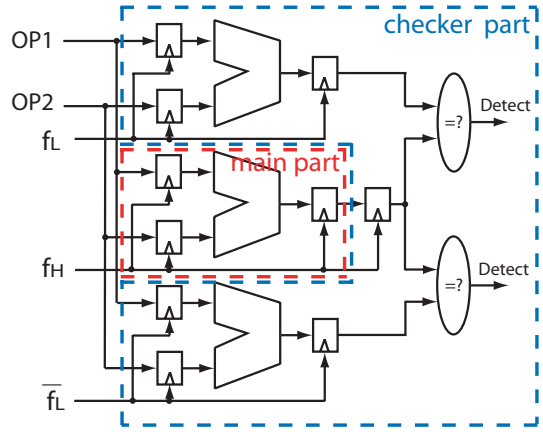

図 2　CTV を適用した ALU

方メイン部では，電源電圧が動作保証をする周波数より高い周波数である f_H が供給される．このため，クリティカルパスに近い回路遅延を持ったパスが活性化されるとタイミングエラーを起こしてしまう可能性を秘めている．このメイン部のタイミングエラーをチェック部の二つの ALU で検出する．チェック部の二つの ALU には互いに逆位相の f_L が供給される．これは，f_H で動作するメイン部のタイミングエラーの検出を補完して検出するためである．つまり，f_H は $f_L <= f_H < 2 * f_L$ の範囲で指定できる．

2.3　性能改善手法

以後は，CTV を適用した ALU を持つマイクロプロセッサについて考える．タイミングエラーが発生するとパイプラインをフラッシュしプロセッサ状態を回復する．タイミングエラーが検出されるとタイミングエラーを発生した命令の後続命令は再実行される．つまり，タイミングエラーが頻発するとプロセッサ状態を回復するためにかかるペナルティが増大する．そこで，タイミングエラーを起こさない対策をとり，回復のペナルティを抑制する性能改善手法が提案されている [6]．この手法は一度タイミングエラーを起こした命令や演算は再びタイミングエラーを起こす可能性が高いという仮説を利用する．

2.3.1　演算結果キャッシュ

演算結果キャッシュ [6] は過去にタイミングエラーを生じた演算の結果を保持する．図 3 に演算結果キャッシュの構成を示す．

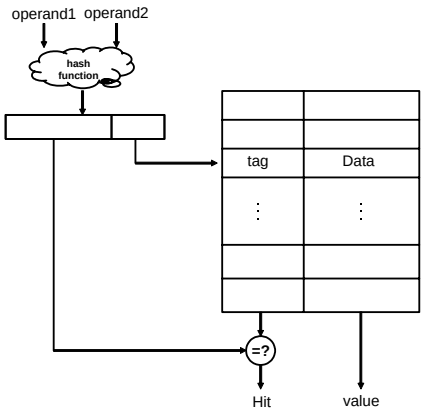

図 3　演算結果キャッシュ

キャッシュアクセスには演算オペランドをハッシュ関数にかけた値を用いる．ハッシュ値の下位ビットをインデックスとしキャッシュラインにアクセスする．演算結果キャッシュのタグにはハッシュ値の上位ビットが格納されている．したがって，ハッシュ値の上位ビットとタグを比較して等しければキャッシュヒットとなる．データフィールドには演算オペランドの演算結果が保持されており，キャッシュにヒットした場合，保持されている値を演算結果として利用する．このため，タイミングエラーを回避することができる．等しくなければキャッシュミスとなり，その演算オペランドはタイミングエラーを発生しないと予想される．

演算結果キャッシュへの挿入はチェック部によってタイミングエラーが検出された場合に行う．

2.3.2　故障履歴バッファ

故障履歴バッファ [6] は過去にタイミングエラーを起こした命令のプログラムカウンタ (PC) を保持するバッファである．図 4 に故障履歴バッファの構成を示す．バッファへのアクセスは PC の下位ビットをインデックスとしてバッファラインにアクセスする．故障履歴バッファのタグには PC の上位ビットが格納されている．したがって PC の上位ビットと比較して等しければキャッシュヒットとなり，等しくなければキャッシュミスとなる．ミスした場合は通常の実行を行いチェック部で検証を行う．ヒットした場合はエラーを起こす命令と判断して，初めからメイン部の ALU を用いないで，チェック部の ALU の結果を利用する．これにより演算レイテンシは 2 サイクルへ増加するが，回復にかかるペナルティに比べると非常に小さい．

故障履歴バッファへの挿入は回復機構でタイミングエラーが検出された場合に行う．

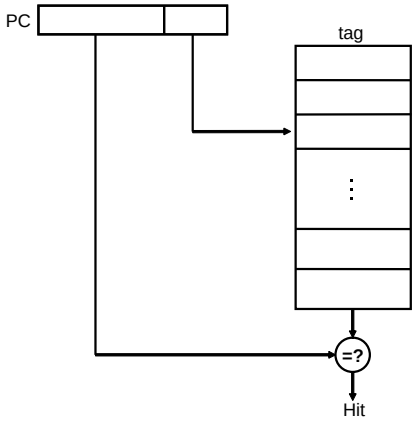

図 4　故障履歴バッファ

3.　コ・シミュレーションによる CTV の評価環境

実際の回路では入力値が活性化するパスによって回路遅延が異なる．このため，クリティカルパスに近いパスを活性化する入力値を持った命令に依存してタイミングエラーは発生すると考えられる．つまり CTV を正確に評価するためには回路遅延を考慮したシミュレーションが必要となる．それにはゲートレベル・シミュレーションを行うことが一般的である．しかし，

アーキテクチャの評価にはゲートレベルでの詳細な検証は必要ない上，マイクロプロセッサ全体を評価すると評価時間が膨大になってしまう．

そこで，マイクロプロセッサの評価に一般に用いられるアーキテクチャレベル・シミュレーションと，回路遅延を考慮したい CTV を適用した回路のタイミングシミュレーションを行うゲートレベル・シミュレーションを協調させる，コ・シミュレーションを行う．これによりゲートレベル・シミュレーションの対象となる回路規模を抑えつつ，回路遅延を反映したアーキテクチャレベル・シミュレーションが可能になる．

図 5 にコ・シミュレーションの概要を示す．

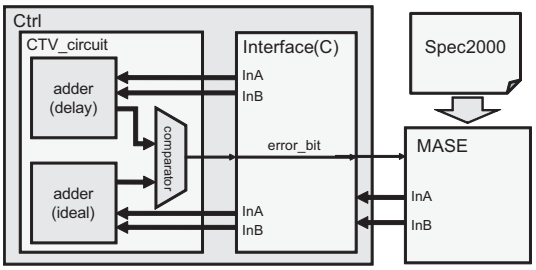

図 5　コ・シミュレーション

図 5 の右側のブロックがアーキテクチャレベル・シミュレータを示している．左側の Ctrl がゲートレベル・シミュレータを示しており，その中の CTV_circuit が CTV を適用した回路を示している．Interface(C) はゲートレベル・シミュレータとアーキテクチャレベル・シミュレータの通信のためのインターフェースである．ゲートレベル・シミュレータの回路部分とインターフェースのデータの送受信には SystemVerilog の機能である Direct Programing Interface(DPI) を使用する [4]．また，アーキテクチャレベル・シミュレータとインターフェースのデータの送受信にはプロセス間通信の共有メモリを使用する．以下にコ・シミュレーションの動作フローを示す．

（1） アーキテクチャレベル・シミュレーションでタイミング解析が必要な命令 (ADD 命令と SUB 命令) が実行されると，ゲートレベル・シミュレータにオペランドが渡される．

（2） オペランドを受け取ったゲートレベル・シミュレータはタイミングシミュレーションを行い，メイン部とチェック部の結果を比較する．そして，比較結果をアーキテクチャレベル・シミュレータに渡す．

（3） アーキテクチャレベル・シミュレータはタイミングシミュレーションの結果を受け取る．エラーが報告されるとパイプラインをフラッシュさせ，プロセッサの状態を正常な状態に回復させ再実行を行う．また，エラーが報告されなければシミュレーションを続行する．

1 から 3 の操作を繰り返すことでコ・シミュレーションを行う．

4. 評　　価

4.1 評価環境

プロセッサのモデルは命令発行幅が 4 命令でアウトオブオーダ実行のスーパースカラプロセッサとし，アーキテクチャレベル・シミュレータには MASE [3] を使用する．命令ウィンドウは 128 エントリとする．タイミングエラーの検出にかかるレイテンシは 2 サイクルとし，タイミングエラーが検出された場合，分岐予測失敗と同様にパイプラインをフラッシュさせている．

CTV の評価は整数 ALU を対象とし，32bit の桁上げ選択加算器 (Carry select adder: CSLA) を Verilog HDL で設計する．CSLA は 11 個のリップルキャリーアダーからなり，それぞれは，4 種類のビット幅のものに分けられる．論理合成には，SYNOPSIS 社の DesignCompiler を使い，遅延ライブラリには日立の 0.18 μ スタンダードセルライブラリを使用する．メイン部の ALU に供給する周波数は DesignCompire の合成結果を基に，クリティカルパス遅延を満たす周波数の 2 倍の周波数を適用する．シミュレーションには，Cadence 社の NC-Verilog を使用する．

命令セットには PISA を選択する．ベンチマークプログラムには SPEC2000CINT の整数系プログラムから 7 つを選択する．なお，各ベンチマークプログラムの実行命令は SimPoint [1] を使用し，10M 命令のシミュレーションポイントを抽出しシミュレーションを行っている．

4.2 回路遅延の考慮による影響評価

回路遅延を考慮することによって評価に与える影響を調べるために，タイミングエラーをランダムに発生させる評価環境とコ・シミュレーションによる評価環境を比較する．タイミングエラーをランダムに発生させる評価環境のエラー発生率には，コ・シミュレーションによる各ベンチマークプログラムのタイミングエラーの発生率を使用した．つまり 2 つの評価環境でタイミングエラー率は等しい．

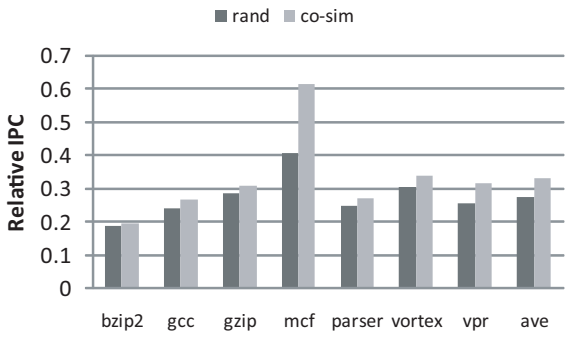

図 6　回路遅延を考慮することによる性能への影響

図 6 に二つの評価環境を用いて得られた IPC(Instruction Per Cycle) を比較したグラフを示す．各グラフは CTV を適用していないプロセッサモデルの IPC を 1 として正規化を行っている．左のグラフがタイミングエラーをランダムに発生させる評価環境による IPC を指示しており，右のグラフがコシミュレーションによる評価環境の IPC を示している．

全てのベンチマークプログラムにおいて結果に違いが見られる．最も特徴がある mcf で，コ・シミュレーションによる IPC が元の約 1.5 倍になっている．つまり，回路遅延を考慮して CTV を評価することで，ランダムにタイミングエラーを起こす評価環境では表れなかった特徴が観測されている．

4.3 性能改善手法の評価

4.3.1 演算結果キャッシュの評価

図 7 に演算結果キャッシュを CTV に適用した場合の性能評価の結果を示す．縦軸が IPC を正規化したもの，横軸が各ベンチマークを示している．また，グラフは最も左が CTV のみを適用した IPC を示しており，その他のグラフは左から順にエントリ数を 2 から 64 まで増やしたものとなっている．また，図 8 に演算結果キャッシュによるタイミングエラーの発生率を示す．縦軸がタイミングエラーの発生率を示しており，分母は総実行命令数，分子がタイミングエラーを起こした ADD と SUB 命令の総数となっている．横軸がエントリ数を示しており，各折れ線グラフはベンチマークプログラムを示している．

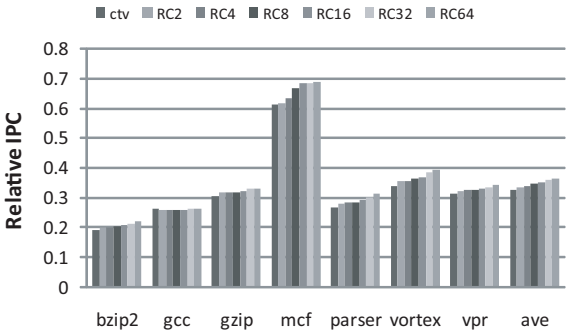

図 7　演算結果キャッシュの影響

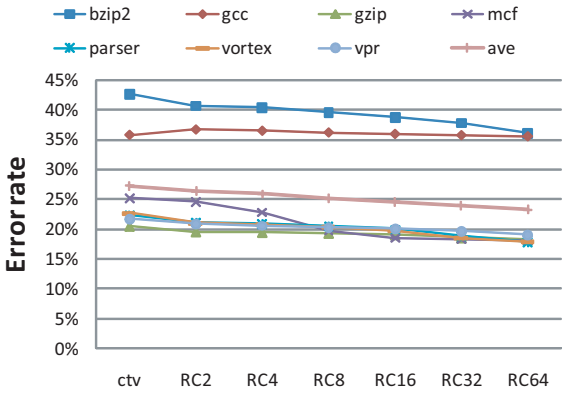

図 8　演算結果キャッシュによるタイミングエラー発生率

図 7 から分かるように，64 にエントリ数を増やしても IPC は平均で 5% ポイントしか改善されていない．また，図 8 からそのときのエラー率も平均で 5%ポイント程度しか削減されていない．この理由は 2 つ考えられる，一つはエラーを発生した同じオペランドを用いる演算が少ないことである．もう一つは演算結果キャッシュにエントリされた演算結果が，再利用される前に書き換わっていることが考えられる．そこで，演算結果キャッシュのエントリを 1024 エントリまで増やして評価した．その結果，最大で約 15%ポイント，平均で約 10%ポイントしか性能が改善されなかった．このことから，エラーを発生した全く同じオペランドを用いる演算が少ないことが分かる．

4.3.2 故障履歴バッファの評価

図 9 に故障履歴バッファを CTV に適用した場合の性能評価の結果を示す．縦軸が IPC を正規化したもの，横軸が各ベンチマークを示している．また，グラフは最も左が CTV のみを適用した IPC を示しており，その他のグラフは左から順にエントリ数を 2 から 1024 まで増やしたものとなっている．また，図 8 と同様に図 10 に故障履歴バッファによるタイミングエラーの発生率を示す．

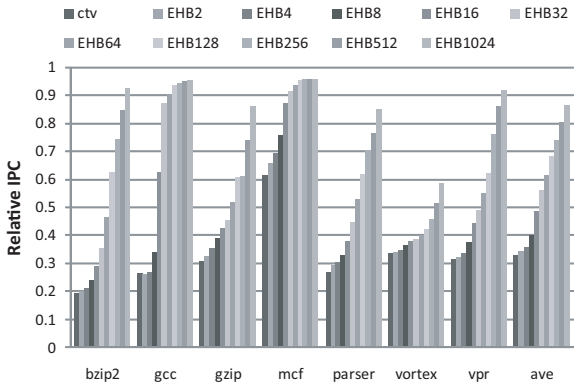

図 9　故障履歴バッファの影響

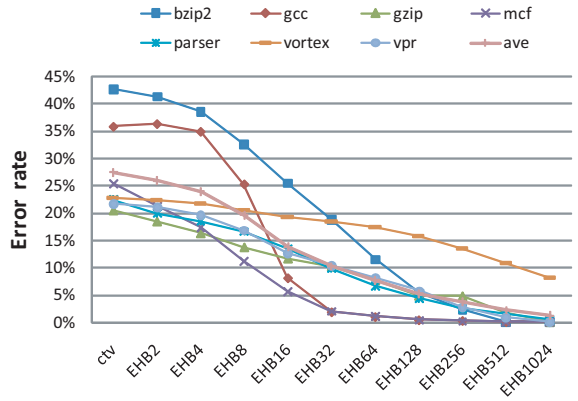

図 10　故障履歴バッファによるタイミングエラー発生率

図 9 から分かるように，CTV を適用していないプロセッサモデルのをベースとすると，故障履歴バッファを使用しなかった場合平均でベースの IPC の約 30%に対して，エントリ数が 1024 エントリの場合平均でベースの IPC の約 85% までに改善されている．故障履歴バッファは最大で 1024 エントリあるが，1 エントリに対してタグフィールドと有効ビットを 1 ビット保持するだけである．よって，ハードウェアのコストを考慮しても，タイミングエラーを回避する手法として有効であると言える．また gcc と mcf に着目すると，128 エントリから 1024 エントリの範囲ではほとんど IPC に変化が見られない．これは，128 エントリでほとんどのタイミングエラーを起こす命令をカバーできているためであると考えられる．実際，図 10 の gcc と mcf のグラフを見て分かるように，128 エントリ以上ではほとんどタイミングエラーは発生していないことが確認できる．

4.3.3 両手法を組み合わせた場合の評価

図 11 に 2 つの性能改善手法を組み合わせて CTV に適用した場合の性能評価を示す．演算結果キャッシュのエントリ数は 64 エントリとし，故障履歴バッファのエントリ数は 1024 エントリとした場合で評価を行った．縦軸が IPC を正規化したもの，横軸が各ベンチマークを示している．各グラフは左から順に CTV のみ適用したもの，演算結果キャッシュのエントリ数が 64 エントリのもの，故障履歴バッファのエントリ数が 1024 エントリのもの，両手法を組合せたものとなっている．

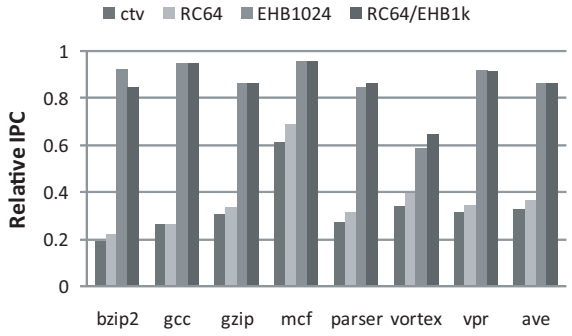

図 11　両手法を組み合わせた場合の影響

組合せの実行方法は，演算結果キャッシュは演算結果を再利用するため高速に実行できるが，故障履歴バッファがヒットした場合は演算が低速に実行されるため，演算結果キャッシュを優先する．

図 11 から分かるように，組合せによる改善はほとんど見られない．これは，演算結果キャッシュの評価結果からも分かるように，演算結果キャッシュの効果が低いため IPC の改善に至らなかったと考えられる．

5. 今後の課題

典型的ケース指向設計では，典型的ケースを配慮して設計を行うことで，従来の保守的な最悪ケースに配慮した設計マージンを取り除き設計の容易さを向上させる．しかし，CTV を ALU に適用した例を見て分かるように，CTV を適用することにより回路規模が非常に大きくなる．また，CTV を適用するためには回路に専用のチェック部を設計する必要がある．よって設計制約を緩和できても，面積と専用回路の点で設計の容易さが向上したとは言いがたい．今後は，より汎用的で単純な設計手法を検討する．

また，CTV では入力依存による遅延ばらつきによるマージンを削減することが出来た．しかし素子特性のばらつきによる設計マージンは，入力依存による遅延ばらつき以外に，プロセスばらつき，電源電圧の揺らぎ，温度ばらつきなどのばらつきに対して設定される．そこで，これらの素子特性のばらつきによる設計マージンを削減できるような設計手法を提案し検討する必要がある．

6. ま と め

本稿では，CTV とその性能改善手法の評価を行い結果を示した．評価には回路遅延を考慮した評価環境としてコ・シミュレーションによる評価環境を構築することによって，アーキテクチャレベル・シミュレーションで回路遅延を考慮したシミュレーションを実現した．評価の結果，CTV を適用した場合の IPC は通常の 3 割程度に性能が落ちてしまうことが分かった．演算結果キャッシュは性能改善手法としての効果がほとんど見られなかったが，故障履歴バッファの適用によって 8 割まで IPC を改善できることを確認した．

謝辞

本研究の一部は，科学研究費補助金・基盤 A(No.19200004)，および科学技術振興機構・CREST プロジェクトの支援によるものである．なお，東京大学 VDEC を通じて提供された日立製作所の 0.18 μ m ライブラリを使用している．

文　献

[1] G. Hamerly, E. Perelman, J. Lau, and B. Calder, “SimPoint 3.0: Faster and More Flexible Program Analysis,” Workshop on Modeling, Benchmarking and Simulation, 2005.

[2] Y. Kunitake, A. Chiyonobu, K. Tanaka, and T. Sato, “Challenges in Evaluations for a Typical-Case Design Methodology,” 8th International Symposium on Quality Electronic Design, 2007.

[3] E. Larson, S. Chatterjee, and T. Austin, “MASE: A Novel Infrastructure for Detailed Microarchitectural Modeling,” International Symposium on Performance Analysis of Systems and Software, 2001.

[4] S. Surtherland, S. Davidmann, and P. Flake, “SystemVerilog for Design,” Kluwer Academic Publishers, 2003.

[5] 国武, 千代延, 田中, 佐藤, “タイミング違反を積極的に利用するプロセッサの評価のための回路遅延を考慮するアーキテクチャレベル評価環境の構築,” DA シンポジウム, 2006.

[6] 千代延, 美馬, 佐藤, “タイミング違反を利用した省電力プロセッサにおける履歴を用いた性能低下抑制手法,” 情処研報 2005-ARC-167, 2006.

[7] 山原, 美馬, 千代延, 佐藤, “タイミング違反を許容する省電力加算器における違反検出回路の高速化,” 情報処理学会論文誌コンピューティングシステム, Vol.47, No.SIG18(ACS16), 2006.