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

山原幹雄[†1] 美馬和大[†2]
千代延昭宏[†3] 佐藤寿倫[†4]

半導体技術がディープサブミクロンの領域に突入するに従って，近い将来，最悪ケースを考慮していたのでは LSI の設計が不可能になると予測されている．ディープサブミクロン領域ではノイズや製造ばらつきが増大し，電源電圧の低下が要求される．このような状況では最悪ケースを考慮するための設計マージンを十分に確保できない．したがって，最悪ケースではなく典型的なケースを考慮した設計手法への転換が必要である．建設的タイミング違反（constructive timing violation: CTV）方式は，そのような転換を実現する設計手法の 1 つである．設計者は稀にしか現れない最悪ケースに悩まされることなく，典型的なケースでの最適化に注力できる．我々はこれまで，CTV の考えに基づいた ALU を 2 種類提案した．残念ながらいずれの ALU も，回路規模や動作速度の点で改善が必要だった．本稿ではタイミング違反の検出回路に加算比較器を応用することで従来の問題点を解決する．Verilog-HDL を用いて桁上げ選択加算器を設計し，論理合成と論理シミュレーションによる評価の結果，エネルギー利用効率をおおいに改善できることが確認できている．

A Fast Fault Detection Circuit for Low-power Adders with Timing Error Tolerance

Mikio Yamahara,[†1] Kazuhiro Mima,[†2] Akihiro Chiyonobu[†3] and Toshinori Sato[†4]

In the deep submicron (DSM) semiconductor technologies, a conservative approach called “worst-case design” will not work very soon. The DSM increases noise and process variations and requires supply voltage reduction, and thus reduces design margins that worst-case design methodologies require. We have to design microprocessors by considering typical case rather than worst case. The Constructive Timing Violation (CTV) paradigm is such a design methodology, where designers are focusing on typical cases rather than worrying about very rare worst cases. We have designed two types of ALUs that utilize the CTV, each of which has its own problems in size and in speed. In order to solve the problems, in this paper, we propose to utilize an adder-comparator as the fault detection circuit. Using Verilog-HDL, we implement a carry select adder that utilizes the CTV, and evaluate it on logic simulations after logic synthesis with delay information. It is observed that substaintial improvement in energy efficiency is achieved.

1. はじめに

半導体の微細化技術は依然として発展しており，LSI の高性能化・省電力化・高機能化・小型化に大きく貢献している．しかしその進展にともない，あらゆる最悪条件を考慮した保守的な設計が困難になってきた．ノイズやプロセスばらつきが増大し，また電源電圧は低下の傾向を示しているため，最悪条件を考慮するための設計マージンが小さくなっているためである．

一方で，汎用計算機から携帯情報端末に至るまで高性能かつ低消費電力なものが求められている．汎用計算機用のマイクロプロセッサの性能を向上させるためには，クロック周波数を上げることが最も単純な方法であるが，同時に消費電力の増大による発熱でプロセッサ自体を破壊するおそれがある．携帯情報端末では Java や動画像処理アプリケーションを動作させることがあたりまえのようになっており，それらを処理

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
†2 日立超 LSI システムズ
Hitachi ULSI Systems Co., Ltd.
†3 九州工業大学
Kyushu Institute of Technology
†4 九州大学
Kyushu University

するマイクロプロセッサは高性能なものが要求されている．同時にバッテリなどの制約により低消費電力化も必要である．しかし，性能向上にともない消費される電力も増加するという背反関係が問題となる．

以上の背景をふまえて我々は，最悪ケースを考慮するのではなく典型的なケースに最適化を施す設計手法を提案している．建設的タイミング違反（constructive timing violation: CTV）方式[1)~3)]と呼ぶその手法では，設計者は稀にしか生じない最悪ケースに頭を悩ませるのではなく，典型的なケースに注目して設計を行う．CTV では以下のような LSI の特徴を利用している．

まず，LSI におけるパスの遅延はすべてが等しいわけではない．LSI の動作周波数を決定する最長パスであるクリティカルパスは少数であり，残りの多くのパスはクリティカルパスの遅延よりも比較的小さな遅延で分布する[4)]．たとえば，約 80%のパスの遅延はクリティカルパスの遅延の半分である[5)]．設計時にはパス遅延の分布がクリティカルパス遅延付近に集中するように最適化を施すことが通常行われるが，それでもなおクリティカルパスは少数である[4)]．

次に，クリティカルパスが活性化されるような入力信号はきわめて少数である．つまり，静的遅延解析時とは異なり，動作時にはタイミング違反は稀にしか生じない．たとえば，加算器のクリティカルパスは最上位ビットでのキャリーとなるが，プログラム中で使用されるオペランドの約半分は 16 ビット以下であり[6)]，32 ビットあるいは 64 ビット長の加算器でクリティカルパスが活性化されるケースは少ない．この特徴を利用して LSI の高速化を図る方式として，上野ら[7)] は可変レイテンシパイプライン（variable latency pipeline: VLP）を提案し，整数 ALU に実装している．各演算に対して最長パスが短くなるように 2 つの回路を用意することで，高い周波数で動作しながら実効レイテンシがほぼ 1 となる ALU を実現している．我々は VLP を参考にして CTV を考案した．

さらに，前述のように LSI 設計では歩留まりを上げるために最悪ケースを考慮してマージンを導入しているが，その結果，LSI はその潜在能力よりも非常に低速に動作している[8)]．マニアの PC ユーザはこのマージンを利用してオーバクロック動作を試みている[9)]．

つまり LSI には，設計時に考慮された最悪ケースは稀にしか発現しないという特徴がある．CTV はこの特徴を利用して動作周波数や電源電圧といった LSI を正常に動作させるための設計制約を寛容に見積もり，それらの改善を図っている．

CTV の考え方は松尾ら[10)] により拡張され，適応性を付与されている．彼らはプロセッサのすべてのステージで CTV を採用し，デペンダブルパイプライニングと名付けた．デペンダブルパイプライニングでは，プロセッサ性能が効果的に改善されるようにクロック周波数を動的に変化させている．タイミング故障率が高くなるとクロック周波数を小さくし，そうでなければ逆に大きくする．彼らは Verilog-HDL を用いて桁上げ先見加算器を設計し，クロック周波数と違反発生率の関係を調査している．我々は桁上げ選択加算器を用いて同様の調査を行った[2)]．本稿では適応性を考慮の範囲外とする．

最悪ケースではなく典型的なケースに最適な設計を行うという点で CTV と共通したコンセプトを持つ方式に，Razor[11),12)]，近似回路（approximation circuits）方式[13)]，アルゴリズム的ノイズ耐性（algorithmic noise tolerance）方式[14)]，そして TEAtime[8)] が提案されている．たとえば Razor[11)] では，エネルギー利用効率改善のためにタイミング制約に違反することを許し，クリティカルパス遅延で決定される周波数よりも高いクロックで動作させる．タイミング違反を検出するために，Razor フリップフロップ（flip-flop: FF）と名付けられたシャドウ FF 付きの特殊な FF を用い，タイミング違反を犯しそうなすべての FF を Razor-FF に置き換える．シャドウ FF には逆相のクロックが供給されている[12)] ので，本来の FF に保持された値とシャドウ FF に保持された値が異なるとタイミング違反が検出できたことになる．

寛容に見積もることで起こる故障状態は，CTV ではフォールトトレランス機構を備えることで動作を保証している．CTV は回路レベルでの投機的実行に相当するので，タイミング違反を検出し，違反検出時にはプロセッサ状態を回復させる機構が必要になる．後者については，現行のプロセッサに備わっている機構を利用できる．前者については，これまで 2 つの方式を検討してきた[2),3)]．回路を多重化する 1 つ目の方式では，元の回路と比較して 2 倍以上の回路規模を必要とする問題があった[2)]．回路規模を削減しパイプライン化した 2 つ目の方法では，違反検出回路の遅延時間が期待したほど改善されなかった[3)]．そこで本稿では，新たな違反検出方式として，Cortadella ら[15)] の加算比較器（adder-comparator）を応用した高速化手法を提案する．加算比較器は，$A+B=K$ の条件が満足されるかどうかを高速に判定する．Lynch ら[16)] はキャッシュのタグ比較を高速化する目的で，また文献 17) では予測アドレスと実際のアドレスを高速に比較する目

的で，それぞれ加算比較器を利用している．本稿では，加算比較器が高速に判定できる $A+B=K$ の条件を利用して，違反検出回路の高速化を図る．

本稿の構成は以下のとおりである．まずタイミング違反検出方式の理解を助けるための準備として次章でタイミング制約と CMOS 回路の消費電力について説明する．3 章でタイミング違反方式を説明し，これまでの研究経過を紹介する．続く 4 章が本稿の主題であり，新たに提案するタイミング違反検出方式を述べる．5 章で提案方式の評価を行い，6 章で結論を述べる．

2. タイミング制約と消費電力

2.1 タイミング制約

ディジタル回路では，記憶素子（典型的には FF）に正しくデータを保持するためには，セットアップタイムとホールドタイムというタイミング制約を満足する必要がある．セットアップタイムとは，タイミング信号（典型的にはクロック）の到着よりも早くデータを確定しなければならない最小の時間である．一方ホールドタイムは，タイミング信号の到着後にデータを保持しなければならない最小の時間である．

回路内の，記憶素子からの信号を入力とし記憶素子を出力先とするパスを考える．これらのすべてのパスの中で，遅延時間が最も大きいパスをクリティカルパスと呼ぶ．クリティカルパスを通る信号の遅延時間が回路全体の最小クロック周期，つまり最大動作周波数を決定する要因となる．いいかえれば，クリティカルパスは動作周波数の向上の妨げとなる部分である．動作周波数はセットアップタイムとクリティカルパスの遅延時間の合計で決まる．この合計時間よりも短い周期で動作させると，正常な動作が保証されない．これがタイミング違反である．

2.2 CMOS 回路の消費電力

CMOS 回路の消費電力は，充放電電力，リーク電流電力，短絡電流電力に分けられる．近年その問題が注視されているのはリーク電流電力であるが，本稿では依然として消費電力全体の大きな部分を占める充放電電力に注目する．

充放電電力 P_{active} とゲート遅延 t_{pd} は以下の式により与えられる．

$$P_{active} \propto fC_{load}V_{dd}^2 \tag{1}$$

$$t_{pd} \propto \frac{V_{dd}}{(V_{dd}-V_{th})^{\alpha}} \tag{2}$$

f はクロック周波数，C_{load} は負荷容量，V_{dd} は電源電圧で，V_{th} はデバイスの閾値電圧である．α はキャリアの速度飽和を与えるパラメータで，近年の MOSFET では 1.3–1.5 の値をとる．式 (1) から分かるように，電源電圧を下げることが電力消費を小さくする最も有効な方法である．しかし，式 (2) から分かるように同時にゲート遅延が増加してしまう．すなわち，動作周波数が低下してしまうことが分かる．結局，消費電力を削減できてもマイクロプロセッサの性能を減じてしまうことになる．プロセッサの性能を維持するためには，電源電圧を下げてもクロック周波数を変えないことが必要となる．しかしそれではクリティカルパスが活性化したときにはタイミング違反を発生させ，最悪ケースにおいても正常に動作することを保証できない．

3. 建設的タイミング違反方式

現在の LSI 設計においては，依然として高速性と省電力が求められている一方で，プロセスばらつきの増大などの要因で設計マージンが少なくなっているという問題に直面している．この問題に対する解の 1 つとして，我々は典型的ケースを指向した設計手法を提案している．この手法では，従来は同時に満足されるように配慮される性能と機能を，独立に考慮して設計する．通常では機能と性能は仕様として与えられるものであるが，ここでは性能に，正しく動作しているときの，という制限を設ける．一方で機能には，いかなる条件下でも正しく動作すること，という条件を付加する．以上のように機能と性能を定義し直したうえで，本提案手法では，従来は 1 つの回路として実現される機能と性能を，2 つの回路で実現する．1 つは性能指向の回路であり，もう 1 つは機能指向の回路である．前者の設計では典型的ケースを想定して性能の最適化を行う．最悪ケースに配慮しなくてよいので設計の容易性が高まる．後者は最悪ケースにおいても機能が満足されるように設計される．ただし仕様で求められた性能を達成できないことを許容し，設計の困難度を低下させる．すなわち，動作時の大部分を占める典型的ケースでは性能指向の回路により性能が保証され，性能指向の回路が動作異常を生じる例外的な最悪ケースでも機能指向の回路により機能が保証される．通常は 1 つの回路として実現される仕様で与えられる性能と機能を，性能指向の回路と機能指向の回路の 2 つで実現するわけである．

CTV は，上記の典型的ケースを指向した設計手法の一例である．性能指向の回路と機能指向の回路を用いることで，省電力化を図る方式である．以下の節では，まず CTV の概要を述べ，続いてこれまで検討してきたタイミング違反の検出方式と違反からの回復方式を紹介し，CTV の潜在的効果を評価した後に，そ

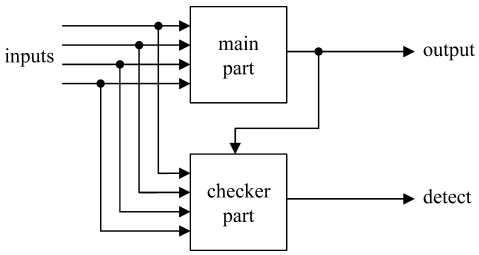

図 1 建設的タイミング違反方式
Fig. 1 Constructive timing violation technique.

れらの違反検出方式の問題点を説明する．

3.1 提案方式の概要

CTV の基本的な考え方を**図 1** を用いて説明する．CTV では，設計時に検出されたタイミング違反が実行時には発生しないと仮定している．回路レベルで投機的実行☆を行っており，タイミング違反を検出する機構と違反検出時にプロセッサの状態を正常に回復させる機構とが必要となる．図では回復機構は省略されている．図に示されているとおり，CTV で設計される回路は性能指向のメイン部（図の `main part`）と機能指向のチェック部（図の `checker part`）から構成される．

メイン部の設計では典型的ケースにおける性能を考慮するだけでよく，最悪ケースにおいては機能が満足されなくてよい．最悪ケース時の機能はチェック部が保証する．たとえばプロセッサで用いられる演算器の設計では，低レイテンシかつ高スループットという高性能を維持できるように設計しなければならないが，最悪ケースでタイミング違反を発生することを許容する．

一方チェック部の役割は，メイン部が動作不良を生じたときでも回路全体として機能を保証することである．つまり，メイン部で生じるタイミング違反を検出するために用意される．チェック部はタイミング違反を生じないように設計されなければならないが，例外的な最悪ケースでのみ利用されるので高性能である必要はない．つまりチェック部の設計では機能を考慮するだけでよく，性能に対する要求は低い．

1 章で述べたように CTV は VLP [7] に大きく影響されて考案されたものであるが，以下の点で異なる．VLP ではつねに正しい機能を保証するために，演算のレイテンシを可変にしている．1 サイクル演算ではタイミング制約を満足できないと検出されると，演算レイテンシを 2 サイクルに増やす．例外事象として除外しているわけではない．一方 CTV ではタイミング違反を例外事象として扱っているので，典型的には演算レイテンシは不変である．現行のプロセッサには，演算のレイテンシが変化する VLP は採用が困難である．なぜなら現在のプロセッサは，高速なクロック周波数を実現するために，各演算のレイテンシが一定であるという仮定の下で命令スケジューリングを行っているからである．この仮定が満足されないと，複雑な命令スケジューリング機構を要し，高速動作を実現できない．

本稿では CTV の適用範囲を組合せ回路に限定する．記憶素子に保持される入力から記憶素子への出力に至るパスを対象とする．ところで，CTV ではタイミング違反を生じたときにそれ以前の状態に回復可能なことが要求される．そのために，CTV の対象となる組合せ回路の外部にある記憶を利用することもありうる．対象となる回路が複雑になればなるほど，図 1 で省略されている回復機構には相応の高機能が要求される．

以下ではプロセッサにおける ALU を例に用いる．まずタイミング違反を検出する方式について，続いてプロセッサ状態の回復方式について説明する．

3.2 タイミング違反検出方式

これまでに回路を多重化する方式[1),2)] と，パイプライン処理を利用する方式[3)] を検討した．それぞれを説明する．

3.2.1 回路を多重化する方法

ある回路（ここでは対象回路と呼ぶことにする）のタイミング違反を動作時に検出する最も簡単な方法は，対象回路の出力を，対象回路と同じ機能を持つがタイミング違反を生じないことが保証されている回路（ここでは保証回路と呼ぶことにする）の出力と比較することである．一致すれば違反を生じていないことが確認できる．したがって保証回路の実現方法が問題となる．最も簡単な方法は，対象回路と同じ回路を利用することである．タイミング違反を生じないように十分なマージンを確保した低速動作をさせればよい．十分なマージンを確保できるのは，保証回路は機能のみ満足されていればよく，性能を要求されないためである．

CTV ではメイン部が対象回路であり，保証回路に比較器を追加すればチェック部を実現できる．このようにメイン部とチェック部では同じ回路を利用できるが，メイン部とチェック部で異なる回路方式を使用すれば，消費電力をさらに改善することなども可能である．たとえば，メイン部では桁上げ先見加算器を用い，チェック部では桁上げ伝搬加算器を用いる，などの選択も可能である．

図 2 に示す CTV を適用した省電力 ALU [1)] は以上

☆ 投機的実行方式については 3.3 節で詳細に説明する．

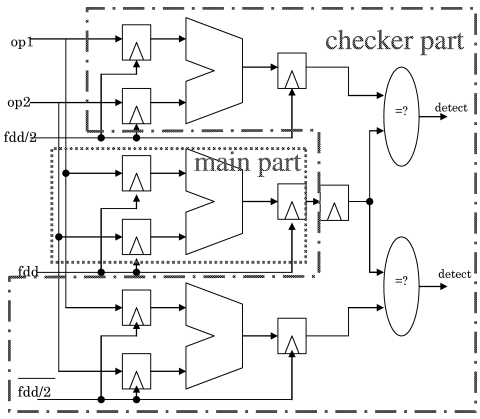

図 2 回路を多重化する方法を適用した ALU
Fig. 2 Duplication-based ALU.

説明した考え方に基づいており，同じ回路を複製して実現されている．`main part` と示されている部分がメイン部である．`checker part` と示されている部分は，メイン部の ALU で生じるタイミング違反に対するフォールトトレランス機構であり，チェック部である．メイン部の出力とチェック部の出力を比較すればタイミング違反を検出できる．ただし，メイン部の動作時にそのタイミング違反を検出するためには，チェック部はメイン部と同じスループットで動作しなければならない．そこで，チェック部はメイン部との動作速度比に応じて，同じ回路を複数持たなければならない．図 2 では，チェック部はメイン部の $\frac{1}{2}$ の速度で動作するので，ALU を 2 つ持つ必要がある．連続するサイクルでメイン部の ALU が演算しなければならない場合でも，チェック部の 2 つの ALU が交互に演算することで，メイン部におけるタイミング違反を調べることができる．

本方式を採用すると，消費電力は以下のように改善できる．式 (2) より，メイン部の最大動作周波数 f_{dd} は以下のように決定される．

$$f_{dd} \propto \frac{(V_{dd} - V_{th})^{\alpha}}{V_{dd}} \tag{3}$$

ここで説明を容易にするために，式 (3) を

$$f_{dd} \propto V_{dd} \tag{4}$$

に単純化する．これにより V_{dd} と f_{dd} との関係の傾向に違いが現れることはない☆．電力効率を改善するには電源電圧を V_{dd} よりも低い V_L（ここで，$V_{dd} > V_L$）に設定する必要がある．この結果，通常は動作周波数は V_L で決まる f_L（ここで，$f_{dd} > f_L$）に低下する．しかし，図 2 のメイン部には f_{dd} を供給する．なぜなら，タイミング違反を生じるのは稀であると期待できるからである．チェック部はメイン部のタイミング違反を検出するためタイミング違反を起してはならないので，電源電圧 V_L かつ動作周波数 f_L で動作させる．

この ALU は電源電圧 V_{dd} かつ動作周波数 f_{dd} で動作しているとする．ここで $V_L = kV_{dd}$，$f_L = kf_{dd}$ とする（ただし $0 < k < 1$）．従来の ALU は動作周波数 f_{dd}，電源電圧 V_{dd} で動作するので，ALU で消費される電力 P_{conv} は式 (1) より，式 (5) となる．

$$P_{conv} = Nf_{dd}C_{load}V_{dd}^2 \tag{5}$$

N は ALU を構成する MOSFET の数である．一方，タイミング違反を利用した低電力 ALU では，動作周波数 f_{dd} かつ電源電圧 V_L で動作する ALU を 1 つと，動作周波数 f_L かつ電源電圧 V_L で動作する ALU を 2 つとで構成されている．よってこの ALU の消費電力は式 (6) となる．

$$\begin{aligned} P_{dup} &= Nf_{dd}C_{load}V_L^2 + 2Nf_LC_{load}V_L^2 \\ &= Nf_{dd}C_{load}(kV_{dd})^2 \\ &\quad + 2Nkf_{dd}C_{load}(kV_{dd})^2 \\ &= (k^2 + 2k^3)Nf_{dd}C_{load}V_{dd}^2 \end{aligned} \tag{6}$$

たとえば $k = \frac{1}{2}$ の場合，回路規模は大きくなるが，全体として消費する電力は適用前の $\frac{1}{2}$ となる．

Chandrakasan ら[18] は，回路を二重化し低速動作させることで消費電力の削減を図っている．これは多重化方式におけるチェック部と同等である．つまり消費電力削減だけが目的であれば，多重化方式におけるメイン部は必要ない．CTV でメイン部を用意する理由は，できるだけプロセッサ性能を維持したいという動機による．Chandrakasan らの方法ではスループットは維持されるがレイテンシは延長される．SPECint に代表される非数値演算系のプログラムでは，演算レイテンシの延長は性能に深刻な影響を及ぼす．したがって多重化方式では，低レイテンシかつ高スループットで演算を行うメイン部が必要である．

3.2.2 パイプライン処理を利用する方法

3.2.1 項の方法は空間的冗長性を利用した方式である．よって多少の回路規模増加はやむをえない．しかし図 2 を見ても分かるように，回路規模が適応前の 3 倍以上になる．また，複数のクロックが必要になる．そこでこれらの問題を解決するために，**図 3** に示すようにパイプライン処理を利用することを検討した[3]．

メイン部は下位ビット ALU と上位ビット ALU か

☆ 現実の先端テクノロジにおいては電源電圧を単純に動作周波数に比例させることは困難である．たとえば，周波数が $\frac{1}{2}$ となったとしても，電源電圧を $\frac{1}{2}$ にすることは容易ではない．ここでは説明を容易にするために，電源電圧を動作周波数に比例させることが可能であるとしている．したがって，本節で説明する電力削減効果が小さくなりうることに注意されたい．以下の説明も同様である．

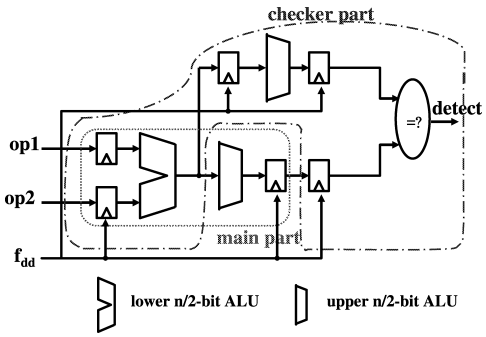

図 3 パイプライン化手法を適用した ALU
Fig. 3 Pipelining-based ALU.

ら構成され，周波数 f_{dd} のクロックを供給するがタイミング違反を生じる可能性がある．チェック部は上記の上位，下位ビット ALU と 1 個の比較器から構成される．チェック部は上位ビットと下位ビット ALU がパイプライン化され，周波数 f_{dd} で動作する．つまり，メイン部を 2 段のパイプライン回路に分けた場合と同じ動作をする．このときタイミング違反を発生しないように設計される．

チェック部はメイン部の処理の流れを停止させることはない．すなわちメイン部とチェック部のスループットは等しい．したがって，チェック部の出力と，1 クロック遅延させたメイン部の出力とを比較することで，タイミング違反の発生を調べることができる．

回路量を抑える目的で，下位ビット ALU はメイン部とチェック部で共有される．図 3 を見て分かるようにタイミング違反検出のための追加回路は上位ビット ALU，1 個の比較器そしてパイプライン化に必要なフリップフロップのみである．また，クロックも 1 系統のみで十分である．

ALU の上位と下位で回路規模と活性化率が同等だとすると，チェック部の回路規模をメイン部の半分だと仮定して，消費電力は以下のように見積もられる．

$$\begin{aligned} P_{pipe} &= Nf_{dd}C_{load}V_L^2 + \frac{1}{2}Nf_{dd}C_{load}V_L^2 \\ &= Nf_{dd}C_{load}(kV_{dd})^2 \\ &\quad + \frac{1}{2}Nf_{dd}C_{load}(kV_{dd})^2 \\ &= \frac{3}{2}k^2Nf_{dd}C_{load}V_{dd}^2 \end{aligned} \tag{7}$$

パイプライン方式は，多重化方式での回路規模問題を解決し，同時に消費電力を削減できる．たとえば $k = \frac{1}{2}$ であれば，元の回路の $\frac{3}{8}$ にできる．

3.3 タイミング違反からの回復方式

タイミング違反が検出されると，何らかの方法でプロセッサの状態を正常に回復する必要がある．そのためには，現在のマイクロプロセッサにすでに備わっている投機的実行方式のための機構を利用できる．

投機的実行方式には，命令間の制御依存関係を投機的に解消する方式と，命令間のデータ依存関係を投機的に解消する方式の 2 種類がある．前者は従来からよく知られているもので，プログラムの実行中に分岐命令が現れると，分岐の方向が決定される前に予測に基づいて将来実行されると期待できる命令を実行する方式である．分岐予測に失敗した場合には，プロセッサの状態を投機開始前の状態に回復させなければならない．この場合，プロセッサは予測に失敗した命令以降をすべて破棄し，命令フェッチからやり直す．

後者は近年注目されているもので，命令の演算結果を予測し，その命令とそれに依存する命令を同時に実行する方式である．演算結果の予測に失敗した場合には，やはりプロセッサの状態を投機開始前の状態に回復させなければならない．この場合は前者と異なり，プロセッサは予測に失敗した命令以降をすべてを破棄するわけではなく，その命令に依存する命令のみを選択的に選んで演算のみをやり直す[17]．

容易に分かるように，これらはいま必要としているタイミング違反からの回復と非常に類似している．つまり，タイミング違反を生じた命令を投機に失敗した命令であると読み替えることで，回復機構を実現できる．タイミング違反を起こした命令は再実行されないことに注意されたい．正しい結果はチェック部で求められており，それを用いて以降の処理を進める．したがって処理がデッドロックに陥る心配はない．

この方法を用いれば，タイミング違反からの回復を実現するハードウェアを改めて用意する必要はなく，ハードウェアのオーバヘッドはきわめて小さい．いずれの方法も利用可能であるが，本稿では後者のデータ依存の投機的実行失敗からの回復機構[17]をとる．

3.4 CTV の効果

本節の目的は，CTV のエネルギー利用効率改善における潜在的な効果を調べることであり，現実性を論じるわけではない．まず評価環境を説明し，その後にシミュレーション結果を紹介する．

SimpleScalar ツールセット[19]を用いて，タイミングシミュレーション環境を構築した．モデルはアウトオブオーダ実行のスーパスカラプロセッサであり，その構成を**表 1** にまとめる．

評価に用いるプログラムには SPEC2000 CINT ベンチンマークから，`164.gzip`，`175.vpr`，`176.gcc`，`197.parser`，`255.vortex`，そして `256.bzip2` の 6 本を使用した．どのプログラムにおいても，最初の 10

表 1 プロセッサ構成
Table 1 Processor configuration.

フェッチ幅	4 命令
分岐予測	512 セット，4 ウエイセットアソシアティブ BTB, 2,048 エントリ bimodal 予測器， 8 エントリリターンアドレススタック，3 サイクルミスペナルティ
命令ウインドウ	16 エントリ命令キュー，8 エントリロードストアキュー
発火幅	4 命令
コミット幅	4 命令
演算器	4 iALU's，1 iMUL/DIV，2 Ld/St's，4 fALU's，1 fMUL/DIV
レイテンシ（全体/間隔）	iALU 1/1，iMUL 3/1，iDIV 20/19，Ld/St 2/1，fADD 2/1，fMUL 4/1，fDIV 12/12
レジスタファイル	32 本の 32 ビット整数レジスタ，32 本の 32 ビット浮動小数点レジスタ
命令キャッシュ	16 K，ダイレクトマップ，32 バイトブロック，6 サイクルミスペナルティ
データキャッシュ	16 K，4 ウエイセットアソシアティブ，32 バイトブロック，2 ポート，ライトバック，ノンブロッキング， 6 サイクルミスペナルティ
2 次キャッシュ	共用，256 K，4 ウエイセットアソシアティブ，64 バイトブロック，48 サイクルミスペナルティ

億命令はウォームアップの目的でのみ実行し，続く 1 億命令をタイミングシミュレーションの対象とした．ただし NOP 命令は数えていない．

プロセッサ性能については，タイミング違反の検出方式に関係なく，CTV の影響は同じである．一方，エネルギー利用効率については，方式によって回路規模に大きな違いがあるため，CTV の効果も異なる．

評価で用いる電源電圧とクロック周波数は以下のように決める．まず多重化方式ではクロック周波数は $f_H = f_{dd}$ であり，チェック部が二重化されていることから $f_L = \frac{f_{dd}}{2}$ が可能である．つまり $k = \frac{1}{2}$ と定める．一方電源電圧は，チェック部でタイミング違反を生じない条件 $\frac{V_{dd}}{2} \leq V_L < V_{dd}$ を満足するように決めればよい．ここでは最も効果の大きな $V_L = \frac{V_{dd}}{2}$ を選択する．このときの消費電力は式 (6) より $\frac{1}{2}$ となる．

パイプライン方式ではクロックは一系統で $f_H = f_{dd}$ である．電源電圧はただちには定まらないので $V_L = \frac{V_{dd}}{2}$ と仮定する．このときの消費電力は式 (7) より $\frac{3}{8}$ となる．

タイミング違反が与える影響を見積もるために，タイミング違反がランダムに発生すると仮定する．その発生率（error rate）を 0–50%に変化させ，そのときのプロセッサ性能とエネルギー利用効率を求める．シミュレーションでは，3.3 節で述べた回復機構を用いて，タイミング違反からの回復に要する影響が考慮されている．違反を発生した同一命令が冗長に実行されると様々な資源の利用効率が低下するが，その影響も考慮されている．

プロセッサ性能の評価指標は時間あたりの命令数（instructions per second: IPS）である．コミットされた命令数のみをカウントする．つまり，分岐予測失敗時に破棄される命令はカウントせず，タイミング違反時に再実行される命令は違反時の 1 度のみカウントする．一方エネルギー利用効率の評価指標には，エネルギー遅延積（energy-delay product: EDP）を用いる．

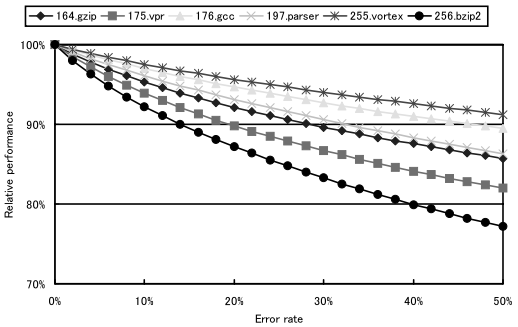

図 4 プロセッサ性能
Fig. 4 Processor performance.

図 4 は CTV がプロセッサ性能に及ぼす影響を示している．横軸はタイミング違反の発生率であり，縦軸は CTV を採用しないモデルと比較したときの性能である．値が大きいほど性能が高い．容易に想像されるとおり，違反発生率が上昇するにつれて，プロセッサ性能は単調に低下する．しかし，いくつかのプログラムでは，発生率が 50%に達しても性能低下率は 10%にすぎない．残りのほとんどのプログラムでもおおむね 20%以下の性能低下である．

図 5 に実行された命令の内訳を示す．各プログラムは 6 本のグラフから構成されており，それぞれタイミング違反の発生率が 0%，10%，20%，30%，40%，そして 50%の場合に相当する．各グラフは 3 つの部分に分けられる．下から順にそれぞれ，コミットされた命令の割合（committed），分岐予測失敗により破棄された命令の割合（misspeculated），そしてタイミング違反発生により再実行された命令の割合（reissued）

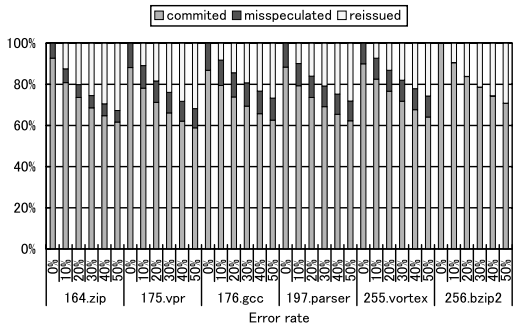

図 5 命令の内訳
Fig. 5 Distribution of instructions.

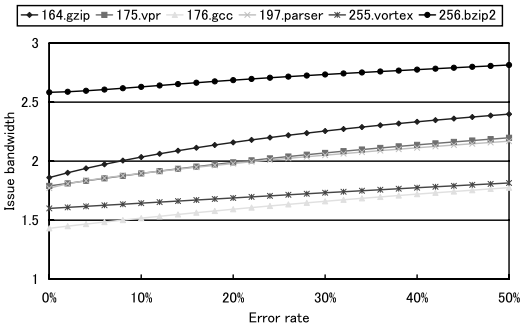

図 7 命令発行バンド幅
Fig. 7 Issue bandwidth.

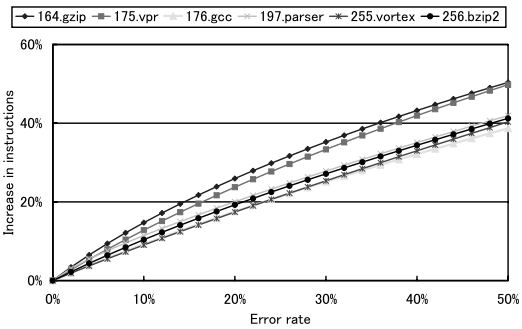

図 6 命令数の増加
Fig. 6 Increase in instructions.

を示している．当然であるが，発生率が上昇するにつれて再実行される命令の割合が増加している．

図 6 は実行された命令数の増加を示している．こちらも当然ながら，タイミング違反の発生率が上昇するにつれて実行される命令が増加している．発生率が 50%の場合で 40〜50%増加している．タイミング違反の発生率よりも実行される命令の増加率が小さいのは，タイミング違反を発生した命令が分岐予測失敗により破棄されるときには再実行されない場合があるためである．たとえば，ある命令の分岐先を投機実行中に，投機実行されている命令でタイミング違反が発生したと仮定する．この違反命令はただちに再実行されなければならないが，同じタイミングで先の分岐命令が実行されて予測失敗が判明すると，この違反命令は再実行されず，正しい分岐先の命令の実行が開始される．違反発生率が上昇するにつれてそのような場合が増加するので，命令の増加率の傾きが徐々に小さくなる．

図 7 に各サイクルに演算器に発行される命令数の平均値を示す．命令発行のバンド幅に相当する．容易に分かるように，タイミング違反を生じない場合には 1.5〜2.5 命令のバンド幅しかない．この理由は命令間には様々な依存関係が存在するためで，このことが命令レベルの並列実行を妨げている．タイミング違反発生により再実行される命令は余っているバンド幅を有効に利用しているため，プロセッサの性能低下が抑えられている．図 4 と図 7 を見比べると，元のバンド幅が小さなプログラムやタイミング違反発生によるバンド幅の上昇が小さなプログラムほど，タイミング違反の性能への影響が小さいことが分かる．

このような余っている命令発行バンド幅は，フォールトトレラントなプロセッサ[20),21)] においても有効利用されている．信頼性を獲得するために，フォールトトレラントなプロセッサは空間的あるいは時間的な冗長性を利用する．複数の演算結果を比較することで，正しく動作しているのかどうかを診断する．各命令を複製して異なる演算器で実行する方式[20)] や，各命令を 2 度実行する方式[21)] が提案されている．後者で用いられる機構は，基本的には 3.2 節で説明した選択的な再実行機構と同様に動作する．これらのプロセッサではすべての命令が二重に実行されるが，プロセッサ性能の低下はそれぞれ 15%と 4%にすぎない[21)]．プロセッサの構成やアウトオブオーダ処理の機構が異なるため単純には比較できないが，これらの報告を考慮すれば，タイミング違反発生のプロセッサ性能への影響が小さいことが理解されよう．

図 8 と**図 9** に，多重化方式とパイプライン方式での EDP をまとめる．横軸は違反発生率であり，縦軸は CTV を採用しないモデルと比較したときの EDP である．値が小さいほどエネルギー利用効率が高い．具体的には，値が 100%よりも小さいとエネルギー利用効率が改善されることを示している．タイミング違反の発生率が上がると，当然エネルギー利用効率は悪くなる．しかし CTV を採用していないモデルと比較すると，どちらの方式でも発生率 0〜50%の全範囲ですべてのプログラムでエネルギー利用効率が改善されている．特にパイプライン方式では，発生率が 50%に

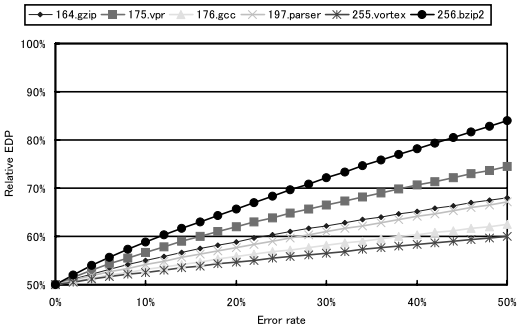

図 8 エネルギー遅延積（多重化方式）
Fig. 8 Energy-delay product (duplication-based).

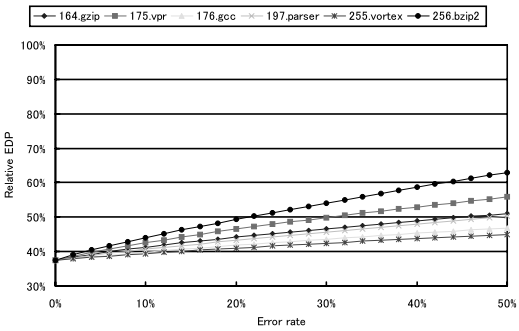

図 9 エネルギー遅延積（パイプライン方式）
Fig. 9 Energy-delay product (pipelining-based).

達してもエネルギー効率が 35%以上改善されている．

以上により，CTV の潜在的な効果が確認できた．

3.5 従来方式の問題点

前節で CTV の考え方を説明し，あわせてその潜在的な効果を紹介した．プロセッサ性能の低下を 10%に抑えなければならない場合は約 10%の発生率でタイミング違反を生じてよく，そのときには 40%以上のエネルギー利用効率改善が期待できる．しかし，実際に回路設計を行って評価したところ，必ずしも理想的な効果が得られないことが分かった[3]．本節では CTV を実現する際の問題点をまとめる．

まず，多重化方式の問題を説明する．第 1 に回路規模が元の回路の 3 倍以上になる．3 重化することで $f_L = \frac{f_H}{2}$ で決まるクロック周波数を提供できるが，回路面積にシビアな応用には適用が困難である．大幅なトランジスタ数の増加はリーク電流の点でも好ましくない．加えて，本方式は複数のクロックを必要とし，実現に困難をともなう．

加えて，タイミング違反の発生率を抑えるために電源電圧の削減を小さくすると，エネルギー利用効率の改善効果が著しく低下する．**図 10** は，電源電圧を $V_L = \frac{V_{dd}}{1.5}$ とし，$f_L = \frac{f_H}{2}$ で動作させたときの EDP

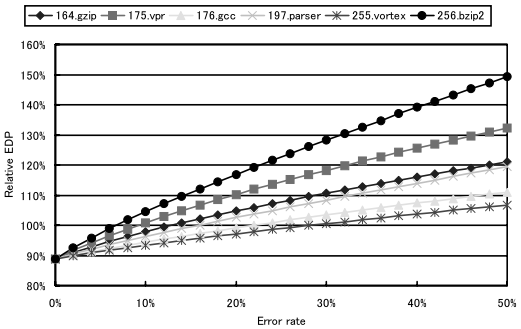

図 10 エネルギー遅延積（$V_L = \frac{V_{dd}}{1.5}$）
Fig. 10 Energy-delay product ($V_L = \frac{V_{dd}}{1.5}$).

である．電源電圧を半分にした場合の図 8 と比較して，大幅にエネルギー利用効率が悪化していることが分かる．タイミング違反の発生率が 5%を超えるあたりで相対的な EDP が元の回路よりも大きくなるプログラムが現れ，違反発生率が 30%を超えるとすべてのプログラムで相対的な EDP が 100%を上回る．

多重化方式における回路規模の問題を解決するために，パイプライン方式が考案された．チェック部をパイプライン動作させることで，違反検出回路の一部をメイン部とチェック部とで共有することを実現している．回路の共有により回路規模を削減できると同時に，パイプライン化することで最大遅延時間を短くする効果も期待された．ところが，実際に桁上げ選択加算器（carry select adder: CSLA）[22] を設計したところ，チェック部の最大遅延時間は約 10%短縮されるだけであった[3]．式 (4) が満足されると仮定しても，電源電圧は $\frac{1}{1.1}$ 倍までしか低下させることができない．この場合には，タイミング故障がまったく発生しなくても，消費電力は元の回路と比べて 1.24 倍に増加してしまう．上位ビットと下位ビットに分けるパイプライン動作が CSLA には適していないことは確かである．しかしそれを考慮しても，パイプライン方式は必ずしも問題の解決にはならないことも事実である．

4. 加算比較器を応用した違反検出方式

本章では，前章で述べた CTV を実現する際の問題点を解決できる方式を検討する．

4.1 加算比較器を応用する方式

回路規模の削減と動作の高速化の 2 つを実現するために，加算比較器（adder-comparator）[15] を応用することを提案する．規模の増大は，同じ回路を複製するためである．つまり，回路規模の小さな別の回路でタイミング違反を検出できれば，この問題は解決できる．加えて，回路規模が小さければ高速動作も期待で

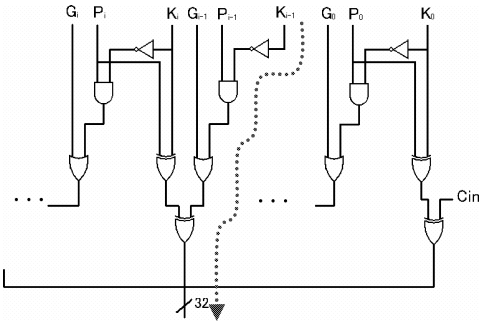

図 11 加算比較器
Fig. 11 Adder-comparator.

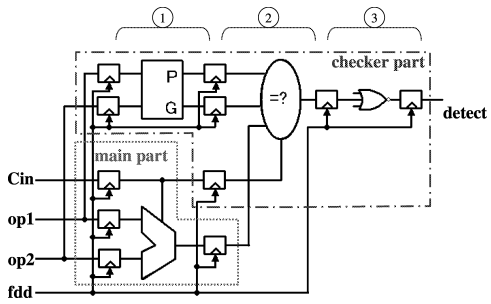

図 12 加算比較器を応用した ALU
Fig. 12 Adder-comparator-based ALU.

きる．

Cortadella ら[15] の加算比較器は，$A + B = K$ の条件が満足されるかどうかを高速に判定する．CTV では，入力オペランド A，B と，メイン部から提供される加算結果 K が利用できるので，加算比較器でチェック部を実現できる．加算比較器は，キャッシュにおけるタグ比較の高速化[16] や，アドレス予測における検証の高速化[17] ですでに使われているが，CTV にこれを用いるのは初めてである．

加算比較器の動作は以下のとおりである．n ビットの 2 つのオペランド $OP1 = (op1_{n-1}, \ldots op1_0)$ と $OP2 = (op2_{n-1}, \ldots op2_0)$ が与えられたとき，桁上げ発生関数（carry generation fuction）$G = (g_{n-1}, \ldots g_0)$ と桁上げ伝搬関数（carry propagation function）$P = (p_{n-1}, \ldots p_0)$ を以下のように定義する．

$$g_i = op1_i \wedge op2_i \tag{8}$$

$$p_i = op1_i \oplus op2_i \tag{9}$$

このとき，演算結果 $K = (k_{n-1}, \ldots k_0)$ との比較結果 $Z = (z_{n-1}, \ldots z_0)$ は，

$$z_0 = p_0 \oplus k_0 \oplus C_{in} \tag{10}$$

$$z_n = ((p_{n-1} \wedge \overline{k_{n-1}}) \vee g_{n-1}) \oplus (p_n \oplus k_n) \qquad (n \geq 1) \tag{11}$$

によって求められる．ここで K はメイン部から得られるので既知であることに注意されたい．

図 11 に式 (10)，(11) の回路図を示す．この図から分かるように，回路は単純なため回路規模は小さく，またゲート数はたかだか 4 段であるので，加算器よりも最大遅延時間は短くなることが期待できる．

図 12 に提案手法を適用した ALU のブロック図を示す．メイン部には周波数 f_{dd} のクロックが供給され，タイミング違反を起こす可能性がある．チェック部は以下に述べる 3 段のステージで構成される．まず入力から桁上げ生成信号 G と桁上げ伝搬信号 P を求める（図中 1）．式 (8)，(9) からも分かるようにゲート段数は 1 である．次に図 11 の比較回路により比較結果を求める（図中 2）．最終段では，比較論理によって得られた n ビットの結果を NOR で 1 ビットに収束させる（図中 3）．

加算比較器がタイミング違反を発見すると，プロセッサ状態を回復させなければならないが，そこで必要とされる正しい演算結果は容易に獲得できる．図 12 には示されていないが，式 (10)，(11) から分かるように，各ビットごとの誤りを求めているので，誤りのあるビットを反転させるだけでメイン部の結果を訂正でき，正しい演算結果を得ることが可能である．

5. 評　　価

前章で提案した，加算比較器を応用したタイミング違反検出方式の有効性を検証するために，CSLA[22] を設計し，回路規模，タイミング違反発生率，そして電力削減効果を評価する．ただし，設計時のマージンは考えず，設計時と動作時のクリティカルパス遅延の違いだけを考慮してタイミング違反を評価する．まず評価環境を説明し，続いて結果を紹介する．

5.1 評 価 環 境

Verilog-HDL を用いて**図 13** に示す CSLA を設計する．論理合成時に遅延情報を持つ論理素子ライブラリを使用することにより，回路に遅延を付加させる．今回使用するライブラリは VDEC から提供されている日立製 0.18 μm プロセス ASIC ライブラリである．論理合成にはシノプシス社の DesignCompiler を用い，回路解析および遅延情報を含むネットリストを出力する．得られた遅延情報から最大遅延時間を求め，動作周波数を決定する．

この周波数を基準とし，周波数を上昇させたときのメイン部でのタイミング違反発生率を論理シミュレーションにより測定する．シミュレーションにはケイデンス社の Verilog-XL を使用する．シミュレーション

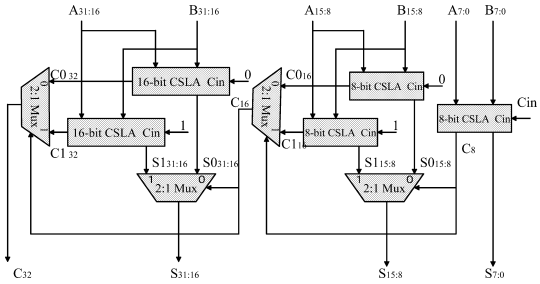

図 13 32 ビット桁上げ選択加算器
Fig. 13 32b carry select adder.

表 2 ベンチマークプログラム
Table 2 Benchmark programs.

Program	input set
164.gzip	ref/input.random
175.vpr	ref/net.in,ref/arch.in
176.gcc	ref/integrate.i
197.parser	ref/ref.in
255.vortex	ref/bendian1.raw
256.bzip	ref/inoput.graphic

に用いる入力データは以下のとおりに作成する．SimpleScalar ツールセット[19] で提供されている命令セットシミュレータ上で 3.4 節と同様に SPEC CPU2000 CINT ベンチマークプログラム実行する．**表 2** にベンチマークと入力セットを示す．実行される演算の中から，加減算命令とロード/ストア命令で行われる加減算を抽出する．各プログラムとも，最初の 10 億命令をスキップし，その後の 5 万命令を対象としている．3.4 節で行ったタイミングシミュレーションと比較すると，論理シミュレーションは著しく低速であるため，シミュレーション対象の命令数を制限する．

メイン部におけるタイミング違反を検出するために，**図 14** に示す検証回路を Verilog-HDL で設計した．遅延情報を含む回路と遅延情報を含まない回路とを比較することで，タイミング違反を検出する．破線で囲まれた部分が提案手法を適用した加算器であり，遅延情報を含んでいる．これとは別に，同じ回路ではあるが遅延情報を含まない加算器を用意する（図では省略されている）．これら 2 つの加算器に同じ入力を与え，結果を比較することでタイミング違反が検出できる．メイン部とチェック部で，それぞれ別にタイミング違反を調査する必要がある．チェック部はさらに桁上げ生成・伝搬信号生成回路と比較器とを別々に調査する．以上の目的で，図の破線外に示す 3 つの比較器を用意した．これらは評価用の回路であるので遅延情報を含んでいない．上から順にそれぞれ，桁上げ生成・伝搬信号生成回路，比較器，メイン部の加算器における故障検出のためのものである．図中 `ideal result` と書

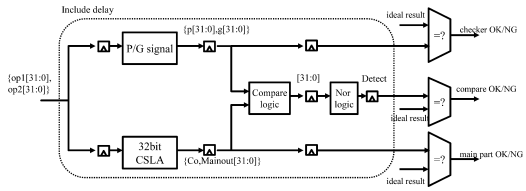

図 14 検証回路
Fig. 14 Verification circuit.

表 3 回路規模
Table 3 Area.

	面積（mm^2）	増加率（%）
適用前	0.094	—
パイプライン化方式	0.131	39.4
加算比較器方式	0.118	22.5

かれた結果が，図では省略されている遅延情報を含まない加算器の結果である．

電力は Verilog-XL とシノプシス社の PowerCompiler を用いて測定する．タイミング違反発生率の測定と同様にベンチマークプログラムを実行し，そのときのスイッチング情報を抽出する．そのスイッチング情報をもとに電力解析を行う．

5.2 回路規模

表 3 に，元の CSLA，パイプライン方式 CTV を適用した CSLA，そして加算比較器方式 CTV を適用した CSLA の，回路面積を示す．元の CSLA と比べたときの，加算比較器方式における回路規模の増加率は 22.5%にすぎない．パイプライン方式と比較しても，回路規模増加率は約半分である．多重化方式における回路規模の問題を大きく軽減できている．

文献 23) では，0.18 μm プロセスで R10000 プロセッサ[24] を実装するときの面積が見積もられている．L2 キャッシュを除くコア部は 48.2 mm^2 で，そのうちの 2.1 mm^2 を整数演算器が占めている．プロセッサ構成や ALU の回路方式が異なるため厳密ではないが，これらの値を用いて多重化方式 CTV と加算比較器方式 CTV を用いるプロセッサ全体の面積を見積もる．CTV は整数演算器にのみ適用されていると仮定する．多重化方式では整数演算器の領域が 3 倍になるので，$48.2 + 2.1 \times 2 = 52.4\,mm^2$ と見積もることができる．一方加算比較器方式では，表 3 に見られるように整数演算器の領域が 22.5%増になるので，$48.2 + 2.1 \times 0.225 = 48.6\,mm^2$ と求められる．すなわち，加算比較器方式は多重化方式に比して約 8%の面積削減を達成できる．多重化方式では 2 系統のクロックを供給する必要があり，その分配がレイアウトに与える影響は大きいと予想されるので，実際には 8%以

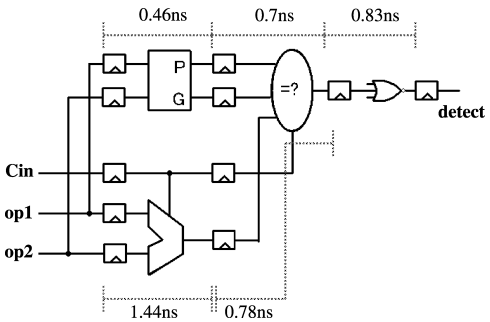

図 15 タイミング解析
Fig. 15 Timing analysis.

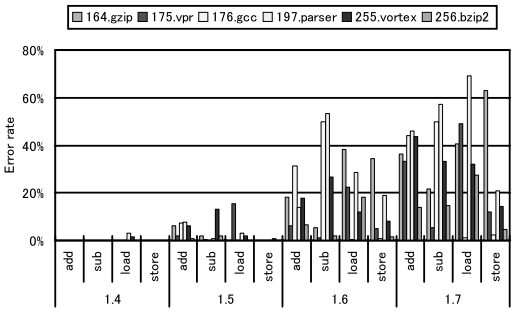

図 16 タイミング違反発生率
Fig. 16 Timing error rate.

上の面積削減効果があると考えられる．以上のようにプロセッサ全体の面積を考慮すると加算比較器を応用する方式の効果は小さいが，2 系統のクロックを必要としないことが加算比較器方式の大きな利点であることに変わりない．

5.3 動作周波数を変化させる影響

論理合成後のタイミング解析結果を**図 15** に示す．最大遅延時間は以下のとおりである．メイン部の CSLA では 1.44 ns となった．チェック部では，入力オペランドから桁上げ生成・伝搬信号の生成までが 0.46 ns，桁上げ生成・伝搬信号からビットごとの条件判定までが 0.7 ns，メイン部からの演算結果からビットごとの条件判定までが 0.78 ns，そして条件判定結果の全ビットを 1 ビットにまとめる NOR の通過に 0.83 ns となった．したがって，メイン部とチェック部の最大遅延時間はそれぞれ，1.44 ns と 0.83 ns である．このことから，チェック部でタイミング違反を発生することなく，メイン部の動作周波数を $\frac{1.44}{0.83} = 1.7$ 倍までに向上させることができる．

以上のタイミング解析結果をもとに，メイン部でのタイミング違反発生率を測定する．動作周波数を 1.1–1.7 倍までの間で変化させて論理シミュレーションを行い，メイン部のタイミング違反発生件数を測定した．**図 16** に測定結果をまとめる．横軸は，下段が周波数の倍率，上段が命令の種類である．周波数の倍率が 1.3 倍以下の場合については，タイミング違反がまったく発生していないため省略している．縦軸は違反の発生率である．

多くのプログラムでは周波数の倍率が 1.5 倍以下ではタイミング違反発生率が 10%以下である．それに対し 1.6 倍以上の場合では，タイミング違反発生率が 50%を超えるプログラムが存在する．すでに図で見たように，3.4 節で行った評価からタイミング違反発生率が 10%程度であればプロセッサ性能の低下を 10%までに抑えることができることが分かっている．そこで，動作周波数を 1.5 倍まで向上させることにして，以下の評価を続ける．

5.4 電力削減効果

電力の評価は以下のとおり行う．元の CSLA の電力を式 (5) で求められる電力 P_{conv} とする．一方，加算比較器方式 CTV を適用し，電源電圧を下げた CSLA の電力を P_{a-c} とする．どちらもクロック周波数は f_{dd} である．

P_{a-c} は以下のように求める．加算比較器方式の CSLA に電源電圧 V_{dd} を与えた場合の電力 $P_{a-c(dd)}$ は，

$$P_{a-c(dd)} = N_{a-c} f_{dd} C_{load} V_{dd}^2 \tag{12}$$

となる．N_{a-c} はトランジスタ数である．

さて，3.2.1 項では式 (4) に示す関係が満足されると仮定したが，脚注で説明したように周波数と同じ比率で電源電圧を削減することは困難である．そこでより現実的な削減率としてインテルのペンティアム M[25] での動作周波数と電源電圧の関係を用いることにする．ペンティアム M の 2.1 GHz@1.34 V と 1.4 GHz@1.18 V の値を用いると，動作周波数を $\frac{1}{1.5}$ にするときに電源電圧を $\frac{1}{1.35}$ にできることになる．電源電圧が V_{dd} で動作周波数を $1.5 \times f_{dd}$ にした場合と，電源電圧が $\frac{V_{dd}}{1.35}$ で動作周波数を f_{dd} にした場合で，メイン部のタイミング違反発生率は同様になると考えられる．電源電圧 $V_L = \frac{V_{dd}}{1.35}$ を与えた場合の電力を $P_{a-c(L)}$ は，

$$\begin{aligned} P_{a-c(L)} &= N_{a-c} f_{dd} C_{load} V_L^2 \\ &= N_{a-c} f_{dd} C_{load} (\frac{V_{dd}}{1.35})^2 \\ &= \frac{P_{a-c(dd)}}{1.85} \end{aligned} \tag{13}$$

であり，

$$P_{a-c} = P_{a-c(L)} = \frac{P_{a-c(dd)}}{1.85} \quad (14)$$

となる．したがって，まず電源電圧にライブラリで指定される 1.8 V を与えて $P_{a-c(dd)}$ を測定し，その結果を式 (14) に代入して P_{a-c} を求める．

以上の手順で求められた P_{conv} から P_{a-c} への電力削減率を**図 17** にまとめる．縦軸が電力削減率である．横軸は各ベンチマークプログラムの名前である．各プログラムには 4 本の棒グラフがあり，それぞれ，加算命令での加算，減算命令での減算，ロード命令での加減算，そしてストア命令での加減算での結果である．各命令ごとの平均で 19〜25%，特に 176.gcc の加算命令では 31%の電力を削減できている．

プロセッサ全体で論理シミュレーションをしてはいないので，プロセッサ性能への影響は調査できていない．そこで，図 16 にまとめられたタイミング違反発生率をもとに，図 4 を利用してプロセッサ性能への影響を見積もる．そして図 17 から知ることのできる消費電力を勘案すると，EDP を計算できる．結果を**図 18** に示す．エネルギー利用効率が，平均して 21%改善している．この改善率の有意性について考察する．アーキテクチャレベルで電力見積りを行うタイミングシミュレータとして，Wattch [26] が有名である．Brooks らによると，レイアウト情報を抽出した論理シミュレーションと比較して，タイミングシミュレーションにおける誤差は約 10%である[26]．Wattch はメモリのモデリングを詳細に行っているのに対して，3.4 節で用いたシミュレータではそのようなモデリングを行っていない．しかし，今回の評価ではメモリは重要な役割を演じておらず，また加算器で消費される電力は論理シミュレーションで得られた値を用いている．以上から，CTV により十分なエネルギー利用効率改善を見込めると考えられ，本稿の目的は達成された．

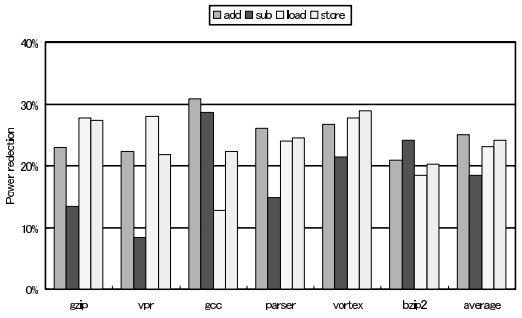

図 17 電力削減率
Fig. 17 Power reduction.

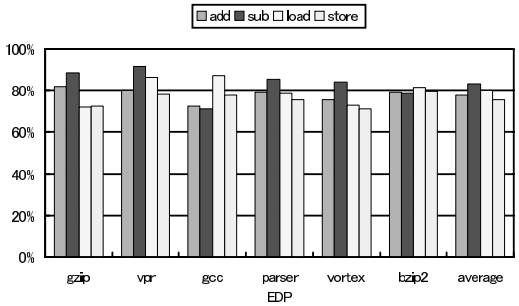

図 18 エネルギー遅延積
Fig. 18 Energy-delay product.

6. ま と め

本研究ではタイミング違反を利用した省電力 ALU における違反検出回路の高速化方式を提案し，その評価を行った．本方式はチェック部に加算比較器を利用することで，回路規模の問題を解決すると同時に高速化を実現している．Verilog-HDL を用いて設計した結果，回路規模の増加は 2 割増し程度であることが確認された．また，元の回路と比較して 1.7 倍までの高速化が可能であることも確認できた．

今回評価に用いたベンチマークプログラムでは，動作周波数を 1.6 倍以上に設定した場合エラー率が 30%を超える場合があり，タイミング違反からの回復に要するペナルティによりプロセッサ性能の低下をまねく可能性がある．本稿ではエラー率を低く抑えるために，動作周波数の倍率を 1.5 倍とした．この条件で動作周波数ではなく電源電圧を元々設定されている電圧から $\frac{1}{1.35}$ 倍に下げた場合，約 30%の消費電力削減と約 20%の EDP 改善が可能であることが確認できた．

今後の課題であるが，まず実際のタイミング違反情報を用いた性能と消費電力の評価が重要である．現在，論理シミュレーションで得られたタイミング違反情報を，タイミングシミュレーションにフィードバックさせることを構築し[27]，性能評価は可能になっている．近い将来に電力評価も可能にし，結果を報告したい．

別の課題として，本稿の回路は加算器のみにしか利用できないので，他の回路も検討する必要がある．加算器は現在ではチップ中に占める割合が小さいので，大きな面積を要する乗算器，除算器，そして浮動小数点演算器で応用できることが望ましい．幸いなことに，これらの回路では構成要素としての加算器の占める割合が大きい．十分有効利用できる可能性があるといえる．ただし，多くの加算比較器で検出されるタイミング違反を集約するための回路上の工夫が必要になると思われる．一方，加算比較器に関しては，さらなる回路規模削減の可能性がある．本稿の回路では n ビットす

べてで比較を行いタイミング違反の検出を行った．しかし，加算器でクリティカルになりうるのは上位ビットおよびキャリーアウトであることに着目すれば，下位ビットの比較を省略したタイミング違反検出回路を設計できる可能性がある．これにより回路規模が削減されると期待される．

本稿では典型的な遅延に配慮して省電力化を図るCTV を扱った．しかし本来，典型ケース指向の設計手法は省電力だけをターゲットにしているわけではなく，設計マージンが少なくなってきている中で，LSI 設計において現在我々が直面しつつある諸問題を統一的に解決することを目的としている．それらは，動作条件変動やシグナルインテグリティの問題を考慮したうえでの，信頼性確保や製造ばらつきに対する歩留まり向上などである．今後は，この考え方を省電力応用以外にも様々な形で発展させていくつもりである．

謝辞 本研究の一部は科学研究費補助金（#15650010，#16300019，#176549）の援助によるものです．なお，東京大学 VDEC を通じて提供いただいた株式会社日立製作所製の LSI 設計用ライブラリを使用しています．

参考文献

1) 佐藤寿倫，有田五次郎：遅延故障を考慮したフォールトトレランス技術に基づく低消費電力方式，情報処理学会研究報告，2001-SLDM-101, Vol.2001, No.042 (2001).
2) 谷野亜沙美，佐藤寿倫，有田五次郎：建設的タイミング違反方式に基づく ALU の HDL 設計とその評価，電子情報通信学会技術報告，VLD2002-147, Vol.102, No.683 (2003).
3) 美馬和大，佐藤寿倫：建設的タイミング違反方式を適用した ALU の改良とその評価，情報処理学会研究報告，2004-ARC-159, Vol.2004, No.80 (2004).
4) 黒田忠広：低電力 CMOS 設計の秘訣とテクニック，IEEE SSCS Kansai Chapter Technical Seminar (June 2003).
5) Usami, K., Igarashi, M., Minami, F., Ishikawa, T., Kanazawa, M., Ichida, M. and Nogami, K.: Automated Low-power Technique Exploiting Multiple Supply Voltage Applied to a Media Processor, *IEEE Journal of Solid-state Circuits*, Vol.33, No.3 (1998).
6) Brooks, D. and Martonosi, M.: Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance, *5th International Symposium on High Performance Computer Architecture* (Jan. 1999).
7) 上野喜代治，幾見宣之，近藤勝久，森 順治，平野勝士：Variable Latency Pipeline（VLP）を用いた 1 GHz ALU データパス，電子情報通信学会技術報告，ICD97-1，Vol.90, No.24 (1997).
8) Uht, A.K.: Going beyond Worst-case Specs with TEAtime, *IEEE Computer*, Vol.37, No.3 (2004).
9) 笠原 光：デュアルコアで空冷 4 GHz 超え，MYCOM ジャーナル (Apr. 2006). http://journal.mycom.co.jp/
10) 松尾 烈，藤川卓也，目次勝彦，村上和彰：Dependable Pipelining—マルチ GHz 時代のマイクロアーキテクチャ，電子情報通信学会技術報告，DC2002-20, Vol.102, No.262 (2003).
11) Ernst, D., Kim, N.S., Das, S., Pant, S., Pham, T., Rao, R., Ziesler, C., Blaauw, D., Austin, T., Mudge, T. and Flautner, K.: Razor: A Low-power Pipeline Based on Circuit-level Timing Speculation, *36th International Symposium on Microarchitecture* (Dec. 2003).
12) Robert, D., Austin, T., Blauww, D, Mudge T. and Flautner, K.: Error analysis for the support of robust voltage scaling, *6th International Symposium on Quality Electronic Design* (Mar. 2005)
13) Lu, S-L.: Speeding up Processing with Approximation Circuits, *IEEE Computer*, Vol.37, No.3 (2004).
14) Shanbhag, N.R.: Reliable and Efficient System-on-chip Design, *IEEE Computer*, Vol.37, No.3 (2004).
15) Cortadella, J. and Llaberea, J.M.: Evaluation of A + B = K Conditions without Carry Propagation, *IEEE Trans. Comput.*, Vol.41, No.11 (Nov. 1992).
16) Lynch, W.L., Lauterbach, G. and Chamdani, J.I.: Low Load Latency through Sum-addressed Memory (SAM), *25th International Symposium on Computer Architecture* (June 1998).
17) 佐藤寿倫：命令再発行機構によるデータアドレス予測に基づく投機実行の効果改善，情報処理学会論文誌，Vol.40, No.5 (1999).
18) Chandrakasan, A.P. and Brodersen, R.W.: Minimizing power consumption in digital CMOS circuits, *Proc. IEEE*, Vol.83, No.4 (1995).
19) Austin, T., Larson, E. and Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling, *IEEE Computer*, Vol.35, No.2 (2002).
20) Ray, J., Hoe, J.C. and Falsafi, B.: Dual use of superscalar datapath for transient-fault detection and recovery, *34th International Symposium on Microarchitecture* (Dec. 2001).
21) Smolens, J.C., Kim, J., Hoe, J.C. and Falsafi,

B.: Efficient resource sharing in concurrent error detecting superscalar microarchitectures, *37th International Symposium on Microarchitecture* (Nov. 2004).
22) Oklobdzija, V.G.: High-speed VLSI Arithmetic Units: Adders and Multipliers, *Design of High-Performance Microprocessor Circuits*, Chandrakasan, A., Bowhill, W.J. and Fox, F., IEEE Press (2001).
23) Burns, J. and Gaudiot, J.-L.: SMT layout overhead and scalability, *IEEE Trans. Parallel and Distributed Systems*, Vol.13, No.2 (2002).
24) Yeager, K.C.: The MIPS R10000 superscalar microprocessor, *IEEE Micro*, Vol.16, No.2 (Apr. 1996).
25) Intel Pentium M processor on 90 nm process with 2-MB L2 cache, Datasheet, Intel Corporation (Jan. 2006).
26) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A framework for architectural-level power analysis and optimizations, *27th International Symposium on Computer Architecture* (June 2000).
27) 国武勇次，千代延昭宏，田中康一郎，佐藤寿倫：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 (July 2006).

(平成 18 年 5 月 2 日受付)
(平成 18 年 9 月 8 日採録)

山原　幹雄

平成 17 年九州工業大学情報工学部知能情報工学科卒業．現在，奈良先端科学技術大学院大学情報科学研究科情報処理学専攻博士前期課程在学．LSI の設計とテストの研究に従事．情報処理学会九州支部平成 16 年度奨励賞を受賞．

美馬　和大（正会員）

平成 14 年九州工業大学情報工学部知能情報工学科卒業．平成 17 年同大学大学院情報工学研究科情報科学専攻博士前期課程修了．同年株式会社日立超 LSI システムズ入社．特定用途向け LSI の設計と開発に従事．

千代延昭宏（学生会員）

平成 14 年九州工業大学情報工学部知能情報工学科卒業．平成 16 年同大学大学院情報工学研究科情報科学専攻博士前期課程修了．同年九州産業大学情報科学部実習助手．平成 17 年より日本学術振興会特別研究員．現在，九州工業大学大学院情報科学研究科博士後期課程在籍．プロセッサアーキテクチャの研究に従事．IEEE 会員．

佐藤　寿倫（正会員）

平成元年京都大学工学部電子工学科卒業．平成 3 年同大学大学院工学研究科電子工学専攻修士課程修了．同年株式会社東芝入社．ULSI 研究所においてマルチプロセッサアーキテクチャ，および消費電力見積り手法の研究に従事．平成 8 年より同社マイクロエレクトロニクス技術研究所に所属し，組み込み用途向けマイクロプロセッサの開発に従事．九州工業大学情報工学部知能情報工学科助教授を経て，現在，九州大学システム LSI 研究センター教授．マイクロプロセッサアーキテクチャ，LSI 設計手法に興味を持つ．博士（工学）．情報処理学会平成 11 年度論文賞，同学会平成 15 年度山下記念研究賞を受賞．IEEE，ACM，電子情報通信学会各会員．