ばらつき耐性を持つカナリアFFを利用したデザインマージン削減による省電力化

佐藤, 寿倫
九州大学 | 独立行政法人科学技術振興機構, CREST

国武, 勇次
九州工業大学

出版情報：情報処理学会論文誌. 49 (6), pp.2029-2042, 2008-06-15. 情報処理学会
バージョン：

Vol. 49 No. 6 情報処理学会論文誌 June 2008

ばらつき耐性を持つカナリアFFを利用したデザインマージン削減による省電力化

佐藤寿倫[†1,†2] 国武勇次[†3]

半導体技術の微細化が進展するにつれて，従来行われてきた最悪ケースを想定したLSI設計は困難になると予測されている．何故なら，そのために必要とされる設計マージンが大きくなり，性能や消費電力に与える悪影響が甚大となるからである．設計者が最悪ケースに煩わされること無く，典型的ケースに注力可能な設計手法が求められている．そのような典型的ケース指向設計を実現するために，我々はカナリア方式を検討している．本稿で我々は，カナリアFFがマイクロプロセッサの省電力化に有効であるかどうかを評価する．シミュレーションにより平均で約9%のエネルギー削減を確認している．

Canary: a Variation Resilient FF to Eliminate Design Margin for Energy Reduction

Toshinori Sato[†1,†2] and Yuji Kunitake[†3]

The deep submicron semiconductor technologies will make the worst-case design difficult, since they require huge design margins that have serious impact on performance and power consumption. Research directions should go to typical-case design methodologies, where designers are focusing on typical cases rather than worrying about very rare worst cases. We are investigating canary logic, which we proposed as a promising technique that enables the typical-case design. In this paper, we evaluate how canary FFs reduce power consumed by the total microprocessor and find the potential energy reduction of 9%.

1. はじめに

半導体製造プロセスの複雑度が増すにしたがって，素子ばらつきを制御することが困難になりつつある[4),11),22),24)]．省電力化に対する要求の高まりとともに電源電圧が降圧され，そのゆらぎが深刻になっている．クロック周波数の上昇に伴い，チップ内における温度ばらつきが増加している．これらの要因による素子特性のばらつきのために，ディープサブミクロン領域の半導体技術において，従来行われてきた最悪ケースに配慮したLSI設計は困難になりつつある．なぜなら，最悪ケースを考慮するために必要となる設計マージンが甚大になり，性能や消費電力に与える悪影響が深刻となるからである．ロバストな設計を実現するために，設計者には従来とは異なる設計手法が求められている．

ひとつの有望な解決策は，典型的ケースを指向する設計である．そこでは，最悪ケースではなく典型的ケースに最適な設計が実施される．近年，同様の思想に基づいて，様々な設計手法が研究されている．Razor[6),8)]，approximation circuits方式[15),16)]，algorithmic noise tolerance方式[20)]，TEAtime[23)]，そして建設的タイミング違反方式[26)]などである．我々はRazorに着目し，その改良版であるカナリア方式 を検討している．

本稿は以下の構成をとる．2節で典型的ケース指向設計を説明する．3節で関連研究を紹介し，特にRazorを詳しく説明する．4節でカナリア方式を提案し，5節ではカナリア方式を実装する上での解決すべき課題を説明する．6節で評価環境を紹介した後に，7節でシミュレーション結果を示す．8節でまとめとする．

2. 典型的ケース指向設計

半導体技術の微細化が一層進展するにつれ，素子ばらつきの問題が深刻化している．ばらつきが大きくなるに伴い，必要となる設計マージンが甚大になっている．素子特性の最悪ケースを想定してマージンを大きくすると，性能や消費電力に与える悪影響が深刻になる．例えば，素子特性のばらつきを考慮して，最悪条件下でも動作速度の使用を満足しようとすると，必要

†1 九州大学
Kyushu University
†2 独立行政法人科学技術振興機構, CREST
JST, CREST
†3 九州工業大学
Kyushu Institute of Technology

以上に電源電圧を高くすることになり，余計な電力を消費してしまう．それを避けるためにマージンを小さくすると，素子特性の最悪ケースを想定することは出来ず，従来の設計手法ではLSIの設計が困難な状況に陥ると危惧されている．このような背景に鑑み，設計容易さを向上するための手法が求められている．有望な設計手法のひとつとして，我々は典型的ケース指向設計を検討している[26]．この手法では，最悪ケースが現実に生じるのは極めて稀である，という観察結果を利用している．そこでは設計者は，最悪ケースに煩わされるのではなく，典型的ケースに集中することが出来る．最悪ケースに配慮する必要が無くなれば，設計制約は大きく緩和されるので，設計が容易になるとともにその期間も短縮されることが期待される．

典型的ケース指向設計において，設計者はひとつの機能に対して同時に二つの回路を実現する．ひとつは性能（例えば動作速度や消費電力）を指向した設計であり（性能指向設計），その際には典型的ケースのみが考慮される．最悪ケースを考慮する必要が無いので，制約が緩和され設計が容易になる．もうひとつは機能を保証する設計である（機能保証設計）．ここでは設計者は，最悪ケースを考慮しなければならないが，性能への配慮からは開放される．機能の保証のみを達成すれば良いので，シンプルな設計が選択可能になり，その検証も容易になる．

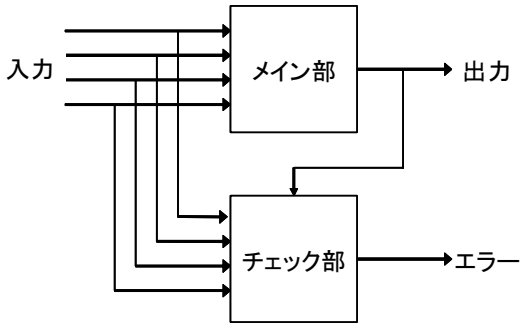

図 1 典型的ケース指向設計
Fig. 1 Typical-case Design

LSIチップを構成する機能ブロックの中で，(例えば動作速度の点で）クリティカルなブロックのみが上述の二つの設計を施される．典型的ケース指向設計の概念を**図 1** に示す．図において，設計対象は二つの回路として設計されている．それぞれをメイン部，チェック部と呼んでいる．それらの機能は同一であるが，役割と実装を異にしている．メイン部の設計において設計者は，性能を向上するような最適化を施す必要があるが，機能が常に正しいことを保証する必要は無い．つまり，メイン部はエラーを生じ得る．上述した性能指向設計に基づいている．チェック部は，信頼性に欠けるメイン部のセーフティネットとして用意される．メイン部で起こり得るあらゆるエラーを検出しなければならないので，設計対象のLSIチップにおける全ての設計制約を満足することが求められる．つまりチェック部の設計において設計者は最悪ケースを考慮しなければならないわけであるが，性能改善のための最適化を実施する必要は無い．上述の機能保証設計に基づいている．エラーが検出された際には，回路の内部状態を何らかの方法で正常な状態に回復させる．

3. 関 連 研 究

典型的ケース指向設計と同様の思想に基づいて，近年様々な研究が行われている．Razor[6),8)]，approximation circuits 方式[15),16)]，algorithmic noise tolerance 方式[20)]，TEAtime[23)]，そして建設的タイミング違反方式[26)] である．

Approximation circuits 方式[15),16)] では，所望の機能を実現するための完全な回路ではなく，それを模倣する簡易版の回路（近似回路）を用意する．近似回路は，完全な回路に比べて高速動作が可能な上，たいていの場合に正しい結果を出力する．したがって，最悪ケースに配慮した設計に比べると，より高速に動作する設計が可能になる．

Algorithmic noise tolerance 方式[20)] では，消費エネルギーと性能における限界を定めるために情報理論を応用する．回路レベルやアルゴリズムレベルでノイズに対する耐性を備えさせることで，上記の限界に近い領域での回路動作を実現している．

TEAtime[23)] では最悪ケースを模倣する回路（模倣回路）を用意し，その回路が正常動作する範囲で電源電圧を降圧したり動作周波数を上昇させたりする．模倣回路には，クリティカルパスのビット幅を圧縮させたコピーを用いる．その入力には 0/1 のトグルを与え，$0 \rightarrow 1$ と $1 \rightarrow 0$ のいずれの遷移においても正常動作することを保証する．

建設的タイミング違反方式[26)] は，回路遅延の入力ばらつきを利用する．クリティカルパスは常には活性化しないことに着目し，クリティカルパス遅延に違反する周波数あるいは電源電圧で，回路を動作させる．正常動作を保証するために，機能は同等であるが低速に動作させる回路を併せて用意する．

3.1 Razor

Razor[6),8)] では，エネルギー利用効率を改善する目

的で，設計におけるタイミング制約に違反することが許容される．つまり，クリティカルパス遅延で決まる周波数よりも速い速度で，Razor を採用する LSI は動作する．タイミング制約違反（タイミングエラー）を動作時に検出するために，**図 2** に示す Razor フリップフロップ (FF) が提案されている．タイミング制約においてクリティカルな FF（メイン FF）の全てにシャドウ FF を用意する．前段からシャドウ FF に至る回路のタイミング制約を満足できるように，シャドウ FF にはメイン FF に比べて遅いクロック（遅延クロック）が供給される．シャドウ FF は常に正しい値を保持することが期待されているわけである．もしメイン FF とシャドウ FF に保持されている値が異なると，前段からメイン FF に至る回路のタイミングエラーが検出されたことになる．マイクロプロセッサのパイプラインでタイミングエラーが検出された際には，カウンターフローパイプラインに基づく回復機構を利用して，プロセッサ状態を正常に復帰させる[8]．

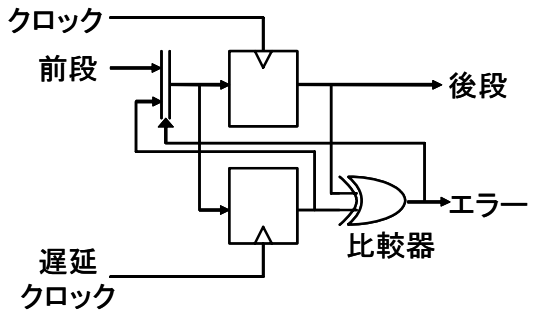

図 2 Razor フリップフロップ
Fig. 2 Razor Flip-flop

Razor では，消費電力を削減するために，電源電圧におけるマージンを取り除く．タイミングエラーの発生頻度に応じて電圧を制御する．Razor で用いられる動作時電圧制御システムを**図 3** に示す．エラー発生頻度が小さい間は，電源電圧を下げる余地がある．逆に頻度が高いと，電源電圧を上げる必要がある．閾値となるエラー発生頻度 (E_{ref}) が予め設定され，制御システムは E_{ref} を維持することに努める．決まったインターバルで，エラー発生頻度 (E_{sample}) と閾値との差 ($E_{diff} = E_{ref} - E_{sample}$) が計算される．$E_{diff}$ が正の値をとるときには，電圧を降圧する余地がある．逆の場合には昇圧しなければならない．

Razor を採用した設計も依然として困難を伴う．シャドウ FF が正しい値を常に保持することを如何に保証するかという点においてである．前段の遅延が小さいと，1 サイクル前の値をラッチすべき遅延クロックのタイミングで，現在のサイクルの値をラッチしてしまう．所謂ショートパス問題[6] である．

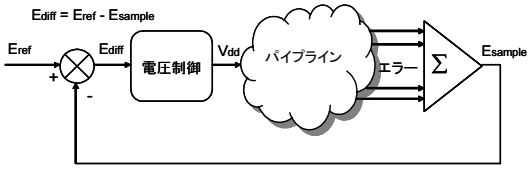

図 3 Razor おける電圧制御システム
Fig. 3 Razor's Voltage Controller

3.2 シャドウ FF を利用するロバスト性向上方式

Li ら[14] は Razor FF のロバスト性を向上させた回路を提案し，スーパースカラプロセッサを設計して評価している．

iRoC Technologies[7),19)] はシャドウ FF をソフトエラー検出の目的で利用する．二通りの実装法が検討されている．ひとつは Razor FF と非常に似ており，遅延クロックを必要とする．他方は遅延クロックを必要としないが，入力が遅延されてシャドウ FF に保持される．メイン FF とシャドウ FF に保持されている値が不一致となると，ソフトエラーが検出されたことになる．上述の遅延を調整することで，検出される過渡パルスの最大幅を変更可能である．

NEC[18] はシャドウ FF を経年劣化によるタイミング故障を予測する目的で利用する．全ての組合せ回路部は複製され，故障箇所はその複製に切り替えられる．故障を予測するために，故障予測 FF が提案されており，シャドウ FF を利用している．前段の組合せ回路部とシャドウ FF との間には遅延が挿入されており，メイン FF がタイミング制約違反に遭遇しない場合でも，シャドウ FF が違反に遭遇し得る．つまり，メイン FF とシャドウ FF に保持されている値を比較することで，経年劣化によるパス遅延の増大を検出可能である．

Agarwal ら[1] は NEC の故障予測 FF と似た方式を提案している．インテル[25] も同様の方式を提案しているが，それはソフトエラー対策が施された FF[17] をばらつきの検査が出来るように拡張したものである．

インテルの別の提案[2] では，シャドウ FF をパラメタばらつきに起因するタイミングエラーを検出する目的で利用する． シャドウ FF はメイン FF への入力を共有するのではなく，メイン FF のマスターラッチを通過した値を保持する．したがって，メイン FF よりもシャドウ FF の方がタイミング制約は厳しい．両 FF のマスターラッチが閉じたときにそれらの値が一致し

ないと，タイミングエラーが検出されたことになる．

我々の知る限り Kehl[12] の研究が，これまで述べてきた遅延と比較を利用する機構を，最初に提案した．ただし目的が異なっている．入力データを異なるレートで複数回サンプリングする．サンプルされたデータを全て入力データと比較し，比較結果に基づいてクロック周波数を調整する．

4. カナリア方式

Razor は設計マージンを取り除くことが出来る非常に優れた方式であるが，その回路実装には改善の余地が残されている．本節では Razor を改善するカナリア方式を述べる．

4.1 カナリア FF

我々は改善方式のひとつとして，**図 4** に示す回路の利用を提案している．この回路では，遅延クロックの代わりに遅延素子とカナリア FF が利用される．したがって iRoC Technologies[7),19)] の提案している FF と同様の回路構成となるが，回路構成の新規性を主張しているわけではなく，異なる用途での利用を提案している．カナリア FF は『炭鉱のカナリア』として働き，今まさに生じようとしているタイミングエラーを検出する目的で利用される．遅延素子が挿入されているので，カナリア FF はメイン FF よりもタイミング制約が厳しい．タイミング制約を徐々に厳しくしていくと，カナリア FF の方がメイン FF よりもが先に前段の回路におけるタイミングエラーに遭遇する可能性が高い．配線遅延の影響が無視出来ない現状では，カナリア FF へ至るパスよりもメイン FF に至るパスの遅延の方が大きくなる可能性が無いわけではないが，思慮深い設計が行われるという想定の下ではそのような状況が起こる可能性は極めて低い．メイン FF は元から存在する FF であり性能を決定するわけであるから，メイン FF に至る配線をカナリア FF に至る配線よりもわざわざ長くするとは考えにくい．

メイン FF とカナリア FF に保持されている値が異なることが，タイミングエラーの予報に相当する．エラーが予報されると，周波数を下げたり電源電圧を上げたり等の対処が施され，メイン FF でのエラーを予防する．カナリア FF の使用には以下のメリットがある．

遅延クロックの廃止

単一単相クロックの採用によって，クロック分配の設計が単純化される．加えて，Razor FF での設計を複雑化しているショートパス問題[8)] を取り除くことができる．

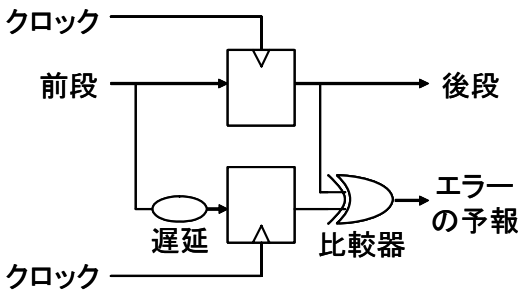

図 4 カナリア・フリップフロップ
Fig. 4 Canary Flip-flop

タイミングエラーの抑制

上述したように Razor FF と異なり，カナリア FF はメイン FF でのタイミングエラーを予防する．このため，いかなる回復機構も不要である．加えて，メイン FF の入力にあるマルチプレクサも不要なため，Razor FF と比較してタイミング制約が緩和される．

素子ばらつきに対する柔軟性

遅延素子が無いときの分岐点から各 FF への遅延を揃えることができると仮定すると，遅延素子にどのような素子ばらつきを生じても，メイン FF よりもカナリア FF が先にタイミングエラーに遭遇する．カナリア FF とメイン FF との間の相対的な動作遅延の差が大きくならないことが要求されるが，両者は空間的にも近い上に活性状態も一致するため，動作遅延の差は小さいと考えられる．つまり，大きな素子ばらつき下における動作保証が比較的容易となる．

4.2 スキャン回路を利用したカナリア FF の実装

量産時テストのために用意されているスキャン回路を利用することで，極めて小さなハードウエアコストでカナリア FF を実装可能である．

図 5 はマイクロプロセッサで用いられているスキャン回路[17)] である．システム FF とスキャン部から構成されている．図中の SI が別の FF の SO に接続され，プロセッサ中の全てのスキャン FF が接続されてシフトレジスタを構成する．テストモードでは，クロックとして SCA と SCB がそれぞれ LA と LB にテストパタンを入力する目的で供給される．続いてクロック UPDATE が与えられると，LB から PH1 に内容が移動し，テストパタンがシステム FF に書き込まれる．次にクロック CLK が与えられると，テストパタンを入力として得られた出力がラッチに保持される．更に PH1 から LA に結果を移動する目的でクロック CAPTURE が供給される．最後に再び SCA と SCB が，今度はチップ外に結果を出力する目的で供給される．

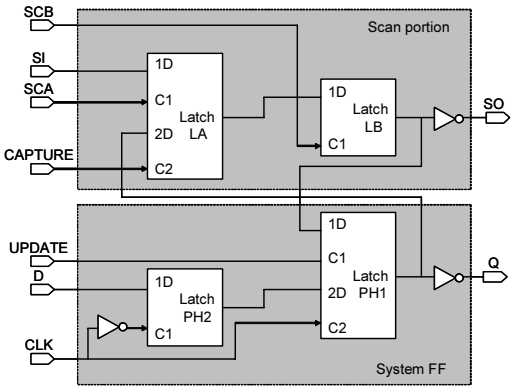

図 5 マイクロプロセッサのスキャン回路[17)]
Fig. 5 Scan FF in Microprocessors[17)]

スキャン部は全ての FF に必要だが，テスト時のみに利用されるので，通常動作時には LA と LB は使用されていない．これらを通常動作時にも使用することで，極めて小さなハードウエアコストの追加でカナリア FF を実装できる．スキャン回路を利用したカナリア FF を**図 6** に示す. テスト時の動作は上述と同様である．通常動作時には，LA と LB がそれぞれ PH2 と PH1 のコピーを保持する．タイミングエラーが生じていなければ ERROR が 0 であるので，LA には D を遅延した値が保持される．一旦 ERROR が 1 になると LA には D の反転が保持され，エラーの検出が維持できる．これを受けて動的電圧可変技術 (Dynamic Voltage Scaling：DVS) による昇圧を開始する．スキャン FF を利用出来るのはクロックが単一であるためであり，Razor FF に対するもうひとつの優位点である．

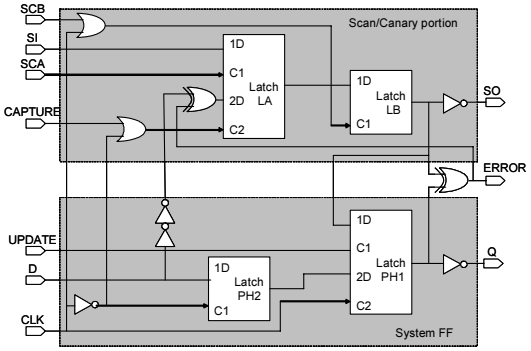

図 6 スキャン回路を利用したカナリア FF
Fig. 6 Canary Implementation via Scan Reuse

4.3 カナリア FF を利用する省電力方式

図 7 は，DVS がカナリア FF と協調動作する様子を説明している．横軸は時刻を表している．縦軸は電源電圧の変化を示している．DVS の動作を説明する．

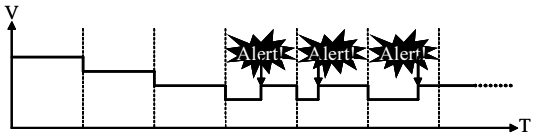

図 7 カナリア方式における DVS（ステップ法）
Fig. 7 Canary's DVS (STEP strategy)

パス遅延の動作時ばらつきを更に積極的に利用して電源電圧を降圧する．予め定められたインターバルの間にタイミングエラーが予報されなければ，ひとつ下のステップまで降圧する．タイミングエラーが予報された場合には，インターバルの途中であっても直ちに昇圧する．この昇圧ポリシーには，ひとつ上のステップへ移動するか，あるいは最も高い電圧へ移動するかの選択肢が存在する．前者を採用する場合をステップ法と呼び，後者を採用する場合をリセット法と呼ぶ．回路遅延は大幅に変動しないと予想されるのでステップ法で問題なく動作すると考えられるが，大幅な変動に対しても動作を保証するためにはリセット法の採用が必要である．

定常モードでは周波数が一定であることに注意されたい．クリティカルパス遅延で決定される電源電圧よりも低い状態になるため，大幅に消費電力を削減できる．カナリア方式は Razor と同等の電力削減[6),8)] が期待できる．

4.4 電源電圧における振動の抑制

タイミングエラーを生じる電源電圧に近づくと，しばしばエラー予報が発せられることになる．タイミングエラーが予報されると電源電圧を昇圧するが，インターバルの経過後に再び降圧される．その結果降圧と昇圧を繰り返すことになり，電源電圧に**図 8** に示すような振動現象が観測される．SPEC2000 に含まれる 164.gzip を実行中に選択された電源電圧を示している．電圧を変える度にオーバヘッドを被るので，この振動は性能の点でも消費電力の点でも好ましくない．

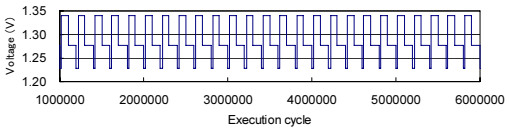

図 8 電源電圧の振動（164.gzip）
Fig. 8 Oscillation in Supply Voltage (164.gzip)

振動を抑制するためにカウンタを利用することを考

える．まず，タイミングエラーが予報された電源電圧を記憶する．次回タイミングエラーが予報された電圧が，記憶されている前回の電圧と同じであれば，カウンタをインクリメントする．異なる場合にはカウンタをリセットする．カウンタの値が予め決められた閾値を越えた場合には振動していると見做し，別に予め定められた期間降圧を停止する．降圧を停止する期間の決定には，電源電圧の変更に要するオーバヘッド時間を考慮する．この期間の経過後は，DVS 動作を再開する．

5. 実装上の課題

カナリア方式を実装するためには，多くの解決すべき課題がある．本節ではそれらをまとめ，可能性の高い解決策について述べる．

5.1 メタスタビリティの問題

シャドウ FF のタイミング制約は常に満足されるわけではないので，それはメタステーブルの状態に成り得る．メタスタビリティは，回路遅延を増大させるだけでなく，誤動作を引き起こしかねない．Razor では, メタスタビリティを解消して常に正しい値をする必要は無く，メタステーブル状態を検出し，タイミングエラーと同様に扱えば十分である．カナリア FF でも事情は同じである．Razor で用いられているメタスタビリティ検出回路[8] が使用可能である．ところで，Razor ではメイン FF がメタステーブル状態と成り得るのに対し，カナリア FF ではメイン FF は常に正しい値を保持することが期待されている．したがって，カナリア FF では時間を要するメタスタビリティ検出方法を選択することも可能である．

5.2 メイン FF でのタイミングエラーの問題

カナリア FF では，メイン FF が常に正しいことが期待されている．しかし，原理的にはメイン FF がタイミングエラーを生じないことを 100%は保証出来ない．クリティカルパス遅延が投機的に違反されると，二つの FF が同時にエラーとなる可能性があり，この場合には比較器の結果は一致しエラーを予報出来ない．しかしタイミング制約が急激に変化しないという条件下では，そのようなことは発生しないと期待できる．

5.3 遅延素子の問題

上述のようにカナリア FF において遅延素子は重要な役割を果たしている．何故なら，その遅延量がカナリア FF のロバスト性を規定するからである．問題となるのが，この遅延量が電源電圧に依存して変化することである．幸いなことに電源電圧が下がるにしたがって遅延量は増大するので，投機が積極的になればなるほどタイミングエラー予測は保守的になる．つまり，遅延量の電源電圧依存性には安全な方向の性質がある．

5.4 消費電力の問題

カナリア FF はスキャン FF を流用して実現可能なので回路規模のオーバヘッドは非常に小さいが，シャドウ FF の消費する電力は無視出来ないかも知れない．幸いなことに Razor の報告では，シャドウ FF と遅延バッファが消費する電力はチップ全体の 3%に過ぎない[6]．カナリア FF でも同様の傾向となると期待できる．

5.5 エラー予報収集の問題

カナリア FF を利用する DVS システムでは，各カナリア FF の発したエラー予報を収集する必要がある．ひとつでもエラー予報が発せられると，プロセッサ全体として電源電圧を変更する必要があるからである．そのためのネットワークはチップ面積を大いに消費する可能性がある．また，エラー収集に要するレイテンシがサイクルタイムに与える影響も懸念される．前者に対しては，スキャン回路を利用して面積を削減する方法を検討中である．後者に対しては，5.1 節で述べたようなレイテンシの大きな方法も利用可能であると考えている．

マルチ電源ドメイン（MVD: Multiple Voltage Domain）アーキテクチャの利用も検討するに値する．MVD はマルチクロックドメイン（MCD: Multiple Clock Domain)[21] のバリエーションのひとつであるが，クロック周波数は全ドメインで同一であるという点と，電源電圧が変化してもクロックは不変であるという点とで異なる．あるカナリア FF がエラーを予報すると，そのカナリア FF を含むドメインだけが電源電圧を昇圧する．MVD を採用すれば，エラー予報を収集するための大域的なオンチップ・ネットワークは不要になる．

6. 評 価 環 境

プロセッサ全体での効果を評価するためには全ての場所にカナリア方式を適用しなければならないが，本評価では整数加算器のみにカナリア方式を適用する．微細化技術が進むにつれ実行ステージの遅延と他のステージの遅延との間のギャップが拡大しており，プロセッサの歩留まりは主に実行ステージにおけるタイミングエラーで決定される[14]．つまり，実行ステージのみでタイミングエラーが生じる[14] という仮定の下で，初期評価を実施する．ただし，非常に深いパイプラインを持つプロセッサは各ステージの遅延が綺麗に

バランスされているため，上記の考察が正しくない可能性がある[14]．しかし Intel の方針変更からも裏づけされるように，微細化技術が進むにつれて深いパイプラインを持つプロセッサは最適な選択ではなくなっており[14]，本稿では評価の対象外とする．

つづいて，タイミングエラー率を求めるためのゲートレベル評価環境と，それを用いてプロセッサ全体への影響を見積もるためのアーキテクチャレベル評価環境を，それぞれ説明する．

6.1 ゲートレベル評価環境

電源電圧および動作周波数を変化させた時の加算器におけるタイミングエラー発生率を求めるために，Verilog-HDL で設計した桁上げ選択加算器（carry select adder：CSLA）を用い，ゲートレベルシミュレーションを実施する．論理合成には SYNOPSYS の DesignCompiler を用い，VDEC から提供されている日立 0.18μm スタンダードセルライブラリを使用する．合成結果は遅延情報を含むネットリストであり，これを用いて実施されたゲートレベルシミュレーション結果と期待値とを各演算で比較する．不一致となると，タイミングエラーを生じているわけである．

シミュレーションでは電源電圧を変更できないため，クロック周波数を変えてシミュレーションし，タイミングエラー率を求める．**表 1** に示すペンティアム M の電源電圧[10] を使用して，クロック周波数から電源電圧を換算する．合成後のネットリストは表 1 の周波数で動作しているわけではなく，そのクリティカルパスで決まる周波数を表 1 の最大周波数とし，残りの周波数は線形写像により決定する．こうして，電源電圧とタイミングエラーの関係を求める．

CSLA への入力は，後述するアーキテクチャレベル評価環境で採取される実行トレースから作成する．トレースから加算器への入力オペランドと制御信号を取得し，ゲートレベルシミュレーションにおけるスティミュラスとして与える．エラー率の分母は演算の総数であり，分子はそのうちのタイミングエラーを生じる演算数である．この入力には，プロセッサ状態を更新する演算だけではなく，失敗した投機中に実行される演算を含んでいる．

表 1 周波数 - 電源電圧
Table 1 Frequency - Voltage Specifications

F(GHz)	2.1	1.8	1.6	1.4
Vdd(V)	1.340	1.276	1.228	1.180
F(GHz)	1.2	1.0	0.8	0.6
Vdd(V)	1.132	1.084	1.036	0.988

ゲートレベルシミュレーションで求められるタイミングエラー率にはカナリア FF の遅延が含まれていないが，遅延素子における遅延を CSLA におけるクリティカルパス遅延の 5%と仮定し，この増加分を考慮して最終的なタイミングエラー率を求める．この値は電源電圧が変化しても変わらないと仮定している．適切な遅延量の決定とそれへの電源電圧の影響の考察は，将来の課題である．

6.2 アーキテクチャレベル評価環境

SimpleScalar ツールセット[3),5)] を用いたシミュレーションで評価する．命令セットには Alpha ISA を選択する．プロセッサ構成は**表 2** の通りである．ベンチマークには SPEC2000 の整数系プログラムから選ばれた 6 つである．なお，最初の 10 億命令をスキップし，続く 1 億命令を対象とする．

表 2 プロセッサ構成
Table 2 Processor Configurations

Clock frequency	2 GHz
Fetch width	8 instructions
L1 instruction cache	16K, 2 way, 1 cycle
Branch predictor	gshare + bimodal
Gshare predictor	4K entries, 12 histories
Bimodal predictor	4K entries
Branch target buffer	1K sets, 4 way
Dispatch width	4 instructions
Instruction window size	128 entries
Issue width	4 instructions
Integer ALUs	4 units
Integer multipliers	2 units
Floating ALUs	1 unit
Floating multipliers	1 unit
L1 data cache ports	2 ports
L1 data cache	16K, 4 way, 2 cycle
Unified L2 cache	8M, 8 way, 10 cycles
Memory	Infinite, 100 cycles
Commit width	8 instructions

電源電圧を変更するインターバルとして，100K，1M，10M サイクルを評価する．電源電圧の変更には 10μsec を要する[9)] と仮定した．その間にはプロセッサは停止状態にあると仮定している．周波数は 2GHz で一定とする．4.4 節で述べた降圧を停止する閾値には，2 と 8 を評価する．降圧を停止する期間は，電圧変更によるオーバヘッドが 0.1%となるように決定した．

アーキテクチャレベルシミュレーションでは，各電源電圧および動作周波数において，確率的にタイミングエラーが生じると仮定する．実際の入力に応じて CSLA のタイミングエラーを考慮するわけではないことに注意されたい．ゲートレベルシミュレーションで取得された確率で一様にタイミングエラーが発生する

と仮定する．例えば，電源電圧 V におけるタイミングエラー率が R であれば，各演算が R の確率でエラーを生じるとする．したがって，同時に N 個の演算が実行されているときには，タイミングエラーが生じる確率は 1-(1-R)^N となる．シミュレータ内部では確率計算をしているわけではなく，各演算でエラーが生じたかどうかを判定しているだけである．

したがって，実際のエラー発生状況とは挙動は一致しない．タイミングエラーの発生確率を一様乱数としてモデル化することで，エラー発生の偏りが失われる．エラー発生に偏りがあるということは，エラーを生じない期間が長くなる可能性が増すことを意味しており，4.3 節で提案した DVS にとって都合が良い．逆に言えば，一様乱数モデル下ではエネルギー削減効果は悲観的になる．加えて，性能評価への影響も小さい．13) で CSLA におけるタイミングエラーのみを考慮して建設的タイミング違反方式を評価した．その場合には，タイミングエラーの入力依存を考慮する場合と確率的な場合とで，結果に有意差は観察できなかった[13)]．本稿と 13) とでは評価対象となる機構が異なるが，タイミングエラー発生をモデル化しているメカニズムは同一であり，プロセッサの動作に与える影響は同じである．13) では確率モデルを採用することによるプロセッサ性能への影響は小さかったのであるから，本評価においても性能への影響は小さいはずである．以上の考察から，確率的にタイミングエラーを生じるという仮定を置いている．

7. 結　　果

まず CSLA 単体でゲートレベルシミュレーションを行い，エラー率と電力削減率を調査する．つづいて，このエラー率を用いてアーキテクチャレベルシミュレーションを実施し，選択された電源電圧，実行時間への影響，そしてエネルギー削減効果について調査する．

7.1 タイミングエラー率

図 9 にゲートレベルシミュレーション結果を示す．6.1 節で説明した方法で求めている．横軸は電源電圧であり，縦軸はタイミングエラー率である．電源電圧を 1.340V から 1.132V まで下げても，タイミングエラー率は 40%未満であることが判る．

タイミングエラーを発生しない最も低い電源電圧を常に選択できると仮定すると，**図 10** に示す割合で各電源電圧を選択できる．スティミュラスとして与えられる全ての演算に対して，ゲートレベルシミュレーションによりタイミングエラーを生じない最低の電源電圧を取得し，その分布から図 10 のグラフを求めている．もちろん，現実にはこのような電源電圧の選択は不可能であり，理想的な場合を表しているに過ぎない．1.132V 以下の電圧を 6 割以上の割合で選択していることがわかる．

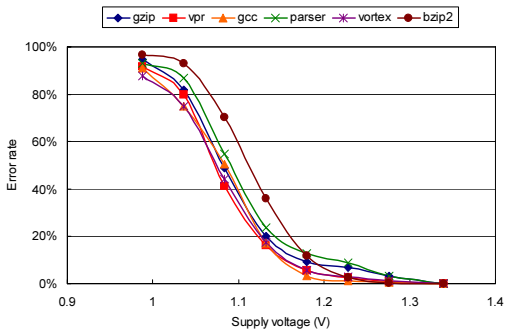

図 9 タイミングエラー率
Fig. 9 Timing Error Rate

図 10 に示した電源電圧を選択出来ると仮定したときの，電力削減率を**図 11** に示す．もちろん図 10 に示したとおりに電圧を選択することは現実には不可能であり，これは電力削減率の上限を与えているに過ぎない．図 11 より理想的な電圧選択が出来れば，おおよそ 30%の電力削減を達成できることがわかる．

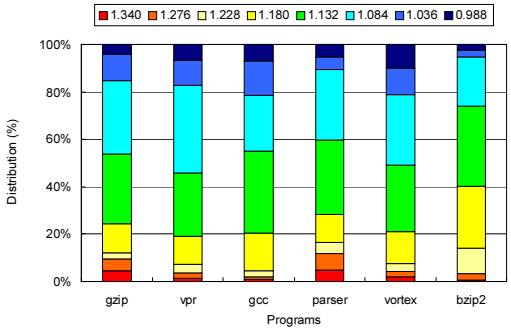

図 10 電源電圧の分布
Fig. 10 Breakdown of Supply Voltage

7.2 電 源 電 圧

つづいて，図 9 に示したタイミングエラー率を用い，アーキテクチャレベルシミュレーションを実施する．

図 12 に選択された電源電圧と，実行時間に対するそれらの割合を示す．アーキテクチャレベルシミュレータに 4.3 節で説明した方式を実装し，シミュレーション中に電源電圧が変更される回数を計測して，この結果を得ている．横軸は，各プログラムに対して，二つの DVS ポリシーとそのインターバルを示している．

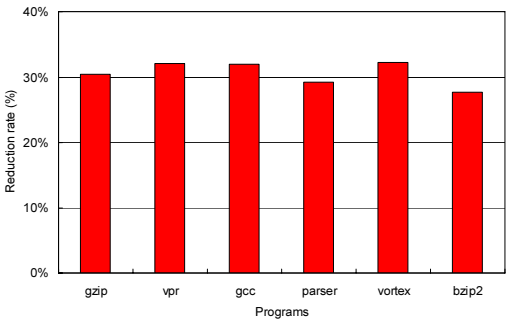

図 11 電力削減率
Fig. 11 Power Reduction

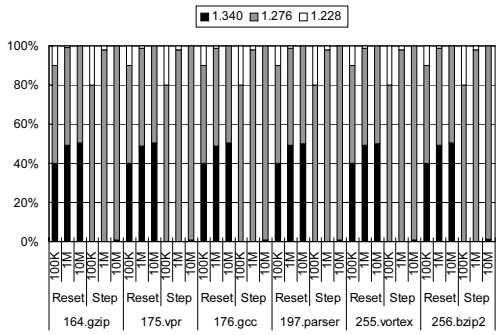

図 12 電源電圧の分布
Fig. 12 Breakdown of Supply Voltage

バヘッドの影響が大きくなり，実行時間への悪影響が懸念される．

また，選択される電圧に 4.4 節で説明した振動が観察された．低い電圧が選択されるとタイミングエラーを発生しやすく直ちに高い電圧に変更されるが，その高い電圧ではエラーを生じないので再び低い電圧へ移行する，という現象である．4.4 節で述べた振動抑制方法については 7.5 節で評価する．

7.3 実行時間

図 13 に実行時間に与える影響をまとめる．カナリア方式を採用しないプロセッサモデルでの実行時間に対する増加率で表している．

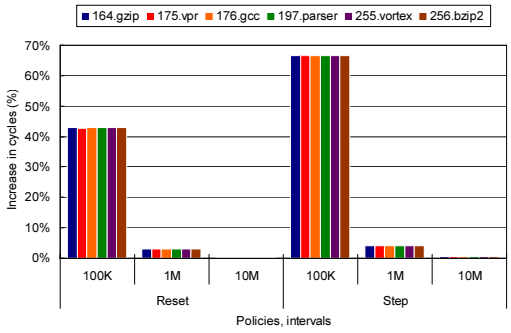

図 13 実行時間の増加割合
Fig. 13 Increase in Execution Cycles

100K サイクル，1M サイクル，10M サイクルのインターバルで，リセット法（図中 Reset）とステップ法（図中 Step）の両ポリシーについて表示している．縦軸は各電源電圧が選択された割合となっている．8 ステップの電圧から実際には 3 ステップしか選択されなかった．

まずリセット法とステップ法とを比較する．同じインターバルで両者を比較すると，ステップ法がより頻繁に低い電圧を選択している．リセット法はタイミングエラーを予報すると最も高い電圧に戻るので，これは容易に予想できる結果である．低い電圧をより頻繁に選択しているステップ法の方が，より大きな電力削減効果が得られると期待される．

続いてインターバルの影響を考察する．インターバルが短いほどより頻繁に低い電圧を選択している．インターバルが長いと次のステップに降圧されるまでの期間が長くなるので，これも容易に予想できる結果である．低い電圧を選択するほど，より大きな電力削減効果が得られると期待されるが，インターバルが短いと頻繁にステップ間を移動して電圧変更に要するオー

まずプログラム間の違いを観察する．容易にわかるように，プログラムの違いによる実行時間への影響の違いは見られない．図 9 よりプログラムの違いによるタイミングエラー発生率への影響は小さいことがわかるので，納得出来る結果である．

次に昇圧ポリシーの影響を考察する．プログラムやインターバルの違いに関係なく，ステップ法でリセット法よりも実行時間の増加割合が大きい．図 12 からステップ法の方が低い電圧を選択している割合が高いことがわかる．これはタイミングエラーに遭遇する確率も高いことを意味するので，ステップ法の方が頻繁に電圧を変更することになる．そのためのオーバヘッドが実行時間に大きな影響を与え，両方式間で実行時間に有意差が現れている．前節で観察した結果からステップ法の電力削減効果が期待されたが，実行時間の増大を考慮すると，期待されるほどのエネルギー削減はないことが危惧される．

最後にインターバルの影響を考察する．昇圧ポリシーやプログラムの違いに関係なく，インターバルが小さくなるほど実行時間への影響は大きい．インター

バルが小さくなればより頻繁に電源電圧を変更し，そのオーバヘッドを大きく被るので，容易に予想される結果である．前節での危惧が確認できたことになる．したがってどの程度のインターバルが適切であるかは，エネルギー消費量を評価しないことには判断できない．

7.4 エネルギー削減率

図 14 に消費エネルギー削減率をまとめる．カナリア方式を採用しないプロセッサモデルの消費エネルギーを基準にしている．負の値は消費エネルギーが増大したことを表している．動作させるスキャン部と遅延回路で消費される電力は評価出来ていない．Razor でこれらに相当するシャドウ FF 等による影響は 3%の電力増である[6]．

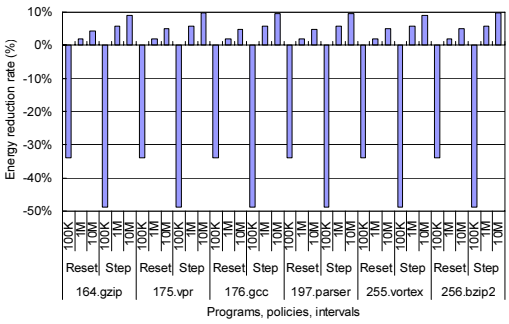

図 14 エネルギー削減率
Fig. 14 Energy Reduction

7.1 節での CSLA を用いた評価と比較して，プロセッサ全体の評価では，エネルギー削減効果が小さい．前者では細粒度な電圧変更が可能であるという仮定を採用していたのに対して，後者ではより現実的な粗粒度な電圧変更のみが可能であると仮定しているためである．細かく電圧を選択出来ないため，必ずしも最適な電圧で動作しているわけではないからである．

7.5 振動抑制方式の効果

次に，電源電圧における振動現象を抑える方式の効果を評価する．**図 15** に選択された電源電圧と，実行時間に対するそれらの割合を示す．今回も 8 ステップの電圧から実際には 3 ステップしか選択されず，1.276V が大部分を占める結果となった．リセット法とステップ法とを比較すると，後者がより頻繁に低い電圧を選択している．リセット法が降圧停止状態に入るまでには多くの時間を要するので，容易に予想できる結果である．インターバルの影響については，リセット法でのみインターバルが短いほどより頻繁に低い電圧を選択する現象が観察される．

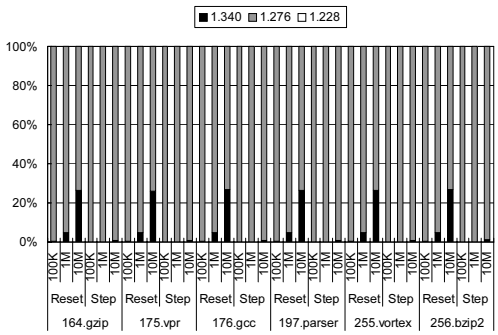

図 15 電源電圧の分布（T=2）
Fig. 15 Breakdown of Supply Voltage (T=2)

図 16 に実行サイクルへの影響を示す．不要な降圧と昇圧が取り除かれた結果，想定されたオーバヘッド（0.1%）よりは大きいが，性能への悪影響は無視できる程度にまで改善できている．

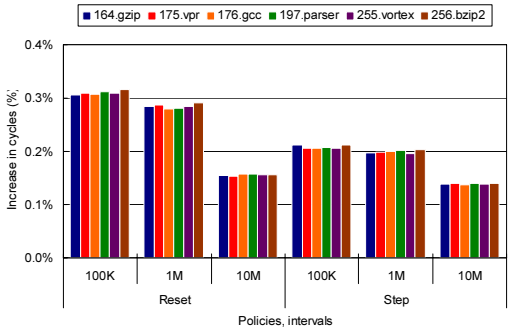

図 16 実行時間の増加割合（T=2）
Fig. 16 Increase in Execution Cycles (T=2)

図 17 に消費エネルギーの削減率を示す．リセット法では，選択された電源電圧の分布にしたがう様に，インターバルが小さいほどエネルギー削減効果が大きい．ステップ法ではインターバルの影響は観察されない．不要な降圧と昇圧を取り除いたにも関わらず，エネルギー削減効果は 7.4 節の結果と大差が無い．これは，電源電圧の選択が保守的となり，より低い電圧を選択出来なくなったためと考えられる．

図 18〜**図 20** に閾値が 8 の場合の結果を示す．閾値に至るまでのサイクル数が大きいため，閾値が 2 の場合よりも振動が長期に続き性能や消費エネルギーには好ましくないと予想されたが，実際には大きな違いは見られなかった．閾値に至るまでの実行時間が全体に占める割合が小さいためであると予想される．

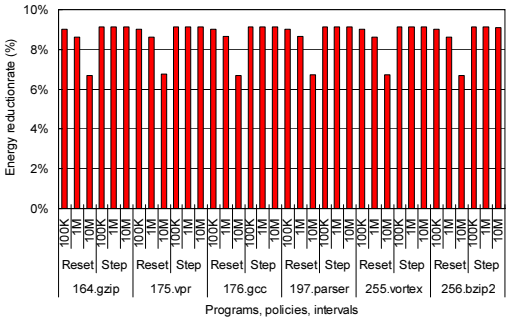

図 17 エネルギー削減率（T=2）
Fig. 17 Energy Reduction (T=2)

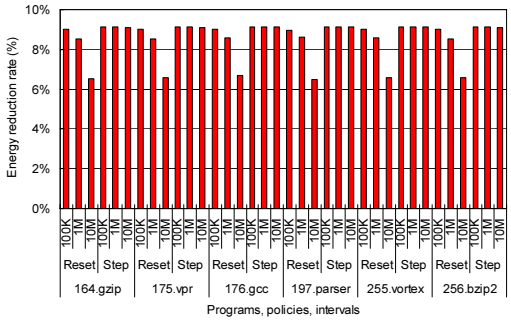

図 20 エネルギー削減率（T=8）
Fig. 20 Energy Reduction (T=8)

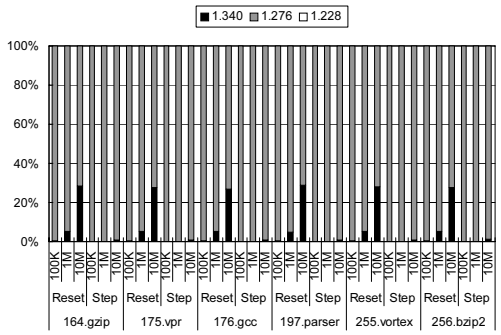

図 18 電源電圧の分布（T=8）
Fig. 18 Breakdown of Supply Voltage (T=8)

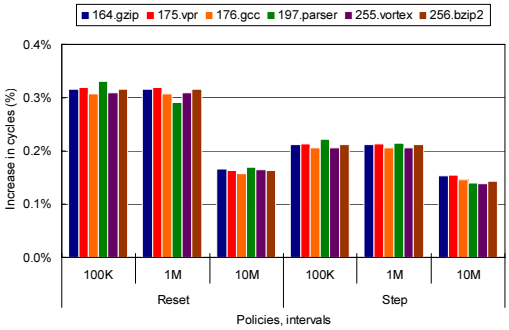

図 19 実行時間の増加割合（T=8）
Fig. 19 Increase in Execution Cycles (T=8)

7.6 エネルギー削減効果に関する考察

CSLA への入力値の多様性のみを利用してエネルギーを削減すると，これまで見てきたとおり 9%程度の削減効果しか得られていない．加えて Razor と同等のオーバヘッドを考慮すると，6%程度の削減効果に減じられてしまう．

Razor ではクロックのみを遅延すれば良いのに対して，カナリア方式ではデータを遅延するためビット幅分の遅延素子が必要になる．例えば 64 ビットデータを扱う場合には，Razor と比較して 64 倍の遅延素子を必要とし，その電力オーバヘッドは大きいと想像される．Razor FF の実装には不明な点があるが，素子ばらつきや配線遅延の影響をできるだけ小さくするためには，メイン FF とシャドウ FF をできるだけ近接して配置し，出来るだけシャドウ FF に近い場所で遅延クロックを生成すべきである．このように考えると，遅延クロックの生成はレジスタ単位ではなく FF 単位で行われていると想像できる．事実，Razor の論文[6),8)] では，全くシャドウレジスタに言及していない．つまりカナリア FF と同様にビット幅分の遅延素子が必要になると考えられ，Razor とカナリア方式とで消費電力に対するオーバヘッドに大きな差は無いと思われる．

加えて，本稿で主張したいことはマージンの削減による省電力化である．残念ながら実製品でどの程度の設計マージンを考慮しているかについてのデータを持っていないため，マージンの削減によるエネルギー削減効果を評価できていない．LSI 設計に従事されている複数の設計技術者にインタビューした限りでは，50%〜100%の設計マージンを確保しているとのことであり，カナリア FF によってこれらを削減できるなら，33%〜50%の省電力化が期待できることになる．本件については，エネルギー削減効果の潜在的な可能性を言及するにとどめる．

8. おわりに

ディープサブミクロン領域での LSI 設計を容易化するために，我々はカナリア方式を検討している．本稿ではマイクロプロセッサ全体での効果をシミュレーションにより評価した．電圧変更のポリシーとインターバ

ルを適切に選択すれば，平均で約 9%のエネルギー削減を達成出来ることが確認された．

将来の発展としては，まず，省電力効果の改善である．今回は入力依存の遅延ばらつきのみに着目した．1 節で述べたように，ばらつきには他にも電源のゆらぎや温度のばらつき等があり，それぞれに設計マージンが確保される．これらのマージンを取り除くことが出来れば，更なる省電力効果が期待できる．一方，電源電圧の振動を抑制する方式については，性能に対するオーバヘッドを大きく抑制することが出来たが，消費エネルギーの削減に関しては大きな改善は見られなかった．電源電圧の選択が保守的になることが理由と考えられるため，積極的な降圧が可能な方式の検討が必要である．

謝辞 本研究の一部は，文部科学省科学研究費補助金・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) M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra: Circuit Failure Prediction and Its Application to Transistor Aging, 25th VLSI Test Symposium (2007)
2) M. Annavaram, E. Grochowski, and P. Reed: Implications of Device Timing Variability on Full Chip Timing, 13th International Symposium on High-Performance Computer Architecture (2007)
3) T. Austin, E. Larson, and D. Ernst: SimpleScalar: an Infrastructure for Computer System Modeling, IEEE Computer, Vol.35, No.2 (2002)
4) S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De: Parameter Variations and Impact on Circuits and Microarchitecture, 40th Design Automation Conference (2003)
5) D. Burger and T. M. Austin: The SimpleScalar Tool Set, Version 2.0, ACM SIGARCH Computer Architecture News, Vol. 25, No. 3 (1997)
6) S. Das, P. Sanjay, D. Roberts, L. Seokwoo Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner: A Self-Tuning DVS Processor Using Delay-Error Detection and Correction, Symposium on VLSI Circuits (2005)
7) E. Dupont, M. Nicolaidas, and P. Rohr: Embedded Robustness IPs for Transient-Error-Free ICs, IEEE Design & Test of Computers, Vol. 19, No. 3 (2002)
8) D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge: Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation, 36th International Symposium on Microarchitecture (2003)
9) S. Gochman, R. Ronen, I. Anati, A. Berkovits, T. Kurts, A. Naveh, A. Saeed, Z. Sperber, and R. C. Valentine: The Intel Pentium M Processor: Microarchitecture and Performance, Intel Technology Journal, Vol.7, No.2 (2003)
10) Intel Corporation: Intel Pentium M Processor on 90nm Process with 2-MB L2 Cache, Datasheet (2006)
11) T. Karnik, S. Borkar, and V. De: Sub-90nm Technologies: Challenges and Opportunities for CAD, International Conference on Computer Aided Design (2002)
12) T. Kehl: Hardware Self-Tuning and Circuit Performance Monitoring, International Conference on Computer Design (1993)
13) 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)
14) H. Li, Y. Chen, K. Roy, and C.-K. Koh: SAVS: A Self-Adaptive Variable Supply-Voltage Technique for Process-Tolerant and Power-Efficient Multi-Issue Superscalar Processor Design, 11th Asia and South Pacific Design Automation Conference (2006)
15) T. Liu and S-L. Lu: Performance Improvement with Circuit-level Speculation, 33rd International Symposium on Microarchitecture (2000)
16) S-L. Lu: Speeding up Processing with Approximation Circuits, IEEE Computer, Vol.37, No.3 (2004)
17) S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim: Robust System Design with Built-In Soft-Error Resilience, IEEE Computer, Vol.38, No.2 (2005)
18) T. Nakura, K. Nose, M. Mizuno: Fine-Grain Redundant Logic Defect-Prediction Flip-Flops, International Solid-State Circuits Conference (2007)
19) M. Nicolaidis: Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies, 17th VLSI Test Symposium (1999)
20) N. R. Shanbhag: Reliable and Efficient System-on-chip Design, IEEE Computer, Vol.37,

No.3 (2004)
21) G. Semeraro, D.H. Albonesi, S.G. Dropsho, G. Magklis, S. Dwarkadas, and M.L. Scott: Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture, 35th International Symposium on Microarchitecture (2002)
22) X. Tang, V. K. De, and J. D. Meindl: Intrinsic MOSFET Parameter Fluctuations Due to Random Dopant Placement, IEEE Transactions on VLSI Systems, Vol.5, No.4 (1997)
23) A. K. Uht: Going beyond Worst-case Specs with TEAtime, IEEE Computer, Vol.37, No.3 (2004)
24) O. S. Unsal, J. W. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzalez, O. Ergin: Impact of Parameter Variations on Circuits and Microarchitecture, IEEE Micro, Vol. 26, No. 6 (2006)
25) M. Zhang, TM Mak, J. Tschanz, K. S. Kim, N. Seifert, and D. Lu: Design for Resilience to Soft Errors and Variations, 13th International On-Line Testing Symposium (2007)
26) 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)

(平成 年 月 日受付)
(平成 年 月 日採録)

佐藤 寿倫（正会員）

平成 1 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．マルチプロセッサアーキテクチャ，消費電力見積り手法などの研究，および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助教授，九州大学システム LSI 研究センター教授を経て，平成 20 年 4 月より福岡大学工学部教授．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，電子情報通信学会，各会員．ACM シニア会員．

国武 勇次

平成 18 年九州工業大学情報工学部知能情報工学科卒業．平成 20 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 4 月より九州大学大学院システム情報科学府博士後期課程在学．ディペンダブル VLSI の研究に従事．