社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－SLDM－127 (23)
2006／11／30

走行時パワーゲーティングを適用した低消費電力乗算器のアーキテクチャ設計

香嶋 俊裕† 武田 清大† 白井 利明‡ 大久保 直昭† 宇佐美 公良‡

†芝浦工業大学大学院 工学研究科 電気電子情報工学専攻

‡芝浦工業大学 工学部 情報工学科

〒 135-8548 東京都江東区豊洲 3-7-5

E-mail: †, ‡ {m106029, m106066, 103061, m105021, usami}@shibaura-it.ac.jp

あらまし 近年、集積回路の微細化により、リーク電力が急増している。消費電力を低減するためにはダイナミック電力、待機時リーク電力だけでなく、動作時リーク電力を低減することが重要になる。本稿では、32bit×32bit乗算器において、被演算データに対して上位 16bit のオールゼロ検出を行うことにより、細粒度でのパワーゲーティングを行う動作時スリープ方式を提案し、設計を行った。また、シミュレーションによる乗算命令におけるオペランドの値の解析を行った。解析を行った結果、32bit×32bit の全ての桁を使う場合は少なく、被演算データがともに上位 16bit がオールゼロ、または片方がオールゼロとなる頻度が多いことが明らかとなった。

キーワード パワーゲーティング, リーク電力, 乗算器, 低消費電力

Architecture Design for Low-Power Multiplier applying Run time Power Gating

Toshihiro KASHIMA† Seidai TAKEDA† Toshiaki SHIRAI‡

Naoaki OHKUBO† Kimiyoshi USAMI‡

†Graduate School of Engineering, Shibaura Institute of Technology

‡Department of Information Science and Engineering, Shibaura Institute of Technology

3-7-5 Toyosu, Koto-ku, Tokyo, 135-8548 Japan

E-mail: †, ‡ {m106029, m106066, 103061, m105021, usami}@shibaura-it.ac.jp

Abstract Leakage power of the integrated circuit is increasing rapidly with the transistor scaling. This paper describes a fine-grained runtime power gating technique to reduce active leakage power of a 32bitx32bit multiplier. By detecting all zeroes of higher 16-bit inputs, we dynamically turn off power switches for the circuit portions of the multiplier. We analyzed the value of operands in multiply instructions by simulation. We found that all zeros appear at the higher 16-bits of either or both operands at more than 80% probability in multiply instructions.

Keyword Power Gating, Leakage Power, Multiplier, Low Power

1. はじめに

集積回路はその誕生以来、微細化によって高性能化・多機能化してきた。しかし、近年、微細化によりリーク電流が急増し、リーク電力はダイナミック電力と比較して無視できない値となっている。今後もリーク電力は増大し、45nm プロセスでは高温状態においてリーク電力とダイナミック電力が同等程度にまで増加すると予測されている(図１)[1]。

リーク電力を低減する代表的な手法の一つに、高 Vth 且つ厚膜のトランジスタをパワースイッチとして、回路の論理部分とグランド間に挿入するパワーゲーティング技術がある[2]。論理回路の動作が不要なとき

（待機時）にパワースイッチをOFFにし、論理部分をグランドから切り離すことでリーク電力を低減する。パワースイッチがOFFしている期間、回路はスリープする。図1からも分かるように、今後、さらに微細化が進展するに従い、待機時のみならず動作時におけるリーク電力が顕著になる。従って、今後、LSIの消費電力を低減するためにはダイナミック電力、待機時リーク電力だけでなく、動作時リーク電力を低減することが重要になると考えられる。

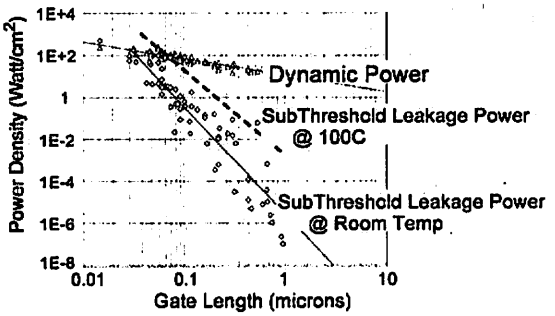

図1　微細化に伴う消費電力の推移

動作時リーク電力を低減する手法として走行時パワーゲーティング制御方式（Run-Time Power Gating：RTPG）がある。RTPGは動作時において使われていない回路部分とその期間を検出し、動的にパワースイッチを制御することで動作時リーク電力を低減する手法である。

文献[3]ではマイクロプロセッサにおいてアイドル状態を規定のサイクル数検出した場合、加えて分岐予測ミスが発生した場合に実行ユニットをスリープさせる。粗粒度で実行ユニットをスリープさせるアーキテクチャレベルでのRTPG手法が提案されている。

しかし、命令を実行する際には必ずしもプロセッサの実行ユニット全体を動作させる必要はなく、命令の実行に必要な一部の回路のみを動作させればよい。その他をスリープさせるにより、細粒度でリーク電力を低減することができる。

RTPGにおいては、リーク電力の低減効果（スリープ可能な期間と頻度）とパワースイッチの駆動によるダイナミック電力の増分を含めた消費電力の削減効果の定量的な評価が重要になる。さらに、パワースイッチの挿入によるクリティカルパス遅延への影響と、スリープ状態から復帰にようする時間（Wakeup-Time）は、RTPG適用時の回路性能を議論する上で重要なファクタとなる。ところが、このようなデータは、シミュレーションベースでの結果はあるが、実チップによる評価は行われていない。

本稿では、これらの評価を実チップを試作して行うことを目的とし、乗算器における被演算データに着目したアーキテクチャレベルからの細粒度RTPG適用設計について述べる。さらに、MIPSアーキテクチャCPUでプログラムを動作させた場合の乗算命令の頻度と乗算命令時の被演算データの分析結果について述べる。

2. 被演算データに着目した細粒度RTPG手法

2.1. 動作原理

32bit×32bitの乗算回路において、一方の被演算データに上位16bitオールゼロが入っていると、必ずしも全てのbitで演算を行う必要はない。その為、被演算データに0が入っているか検出を行い、0の乗算を行う論理回路部分をスリープさせることが可能である。

本手法では、被演算データの上位16bitに着目し、オールゼロであれば乗算を行う論理回路部分をスリープさせ、また乗算を行う必要のない場合は、乗算器全体をスリープさせる。

被演算データにおける上位16bitがオールゼロとなるスリープ可能な場合分けとして、以下の3つのCaseに分けることが出来る。

Case1:上位16bitが共にオールゼロの時
　0000_FFFF × 0000_FFFF = 0000_0000_FFFE_0001
　　⇒演算結果の上位32bitは常に0となる。
　　上位32bitの演算を行う論理回路部分がスリープ可能。

Case2:一方の上位16bitのみオールゼロの時
　0000_FFFF × FFFF_FFFF = 0000_FFFE_FFFF_0001
　　⇒演算結果の上位16bitは常に0となる。
　　上位16bitの演算を行う論理回路部分がスリープ可能。

Case3:すべての桁を使うとき
　FFFF_FFFF × FFFF_FFFF = FFFF_FFFE_0000_0001
　　⇒乗算器全体を動作させる必要があるため、スリープ可能な部分はない。

この3つの場合分けをもとにスリープ制御を行うために、スリープ可能な論理部分をDomainとして以下のように分類を行う必要がある。

- 63bit～48bitの演算を行う論理回路部分：
  Domain High
- 47bit～32bitの演算を行う論理回路部分：
  Domain Middle
- 31bit～0bitの演算を行う論理回路部分：
  Domain Low

Case1 において、スリープさせる Domain は Domain High, Domain Middle である。Case2 において、スリープさせる Domain は Domain High である。乗算をする必要のない場合は、Domain High, Domain Middle, Domain Low をスリープさせる。

2.2. パワースイッチの制御方法

パワースイッチを制御する信号の生成法として、大きく分けて二つの方法がある。設計者により明示的にＲＴＬ記述段階から信号を生成する方法と、論理合成段階においてゲーテッドクロック化を行い生成したゲーテッドクロックイネーブル信号を活用する方法[4]である。

本手法では、アーキテクチャレベルからの適用設計であるため、前者の方法により信号生成を行った。被演算データに対して 16bit ごとにオールゼロ検出回路を設計し、追加している。オールゼロ検出信号をもとに制御回路によって、パワースイッチの制御信号を生成している(図２)。

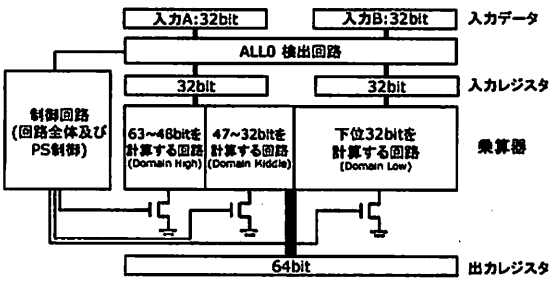

図２　RTPG 手法の構成図

2.3. 動作時・スリープ時における回路動作

パワースイッチを ON した動作時においては、論理回路を高速動作させる。スリープ時においては、パワースイッチを OFF にすることにより、論理回路部分をグランドから切り離す。

前述したように、被演算データが上位 16bit 同士の演算を必要としない場合、上位 16bit 同士の演算を行う回路部分をスリープさせ、下位 16bit を動作させる。本来の演算結果では、上位 16bit 同士の演算結果として０が出力されなければならない。その為、スリープした回路部分の出力は０に固定する構造とした。

3. アーキテクチャレベルからの適用設計

本手法を適用した低消費電力乗算器の設計を行った。設計を行った設計フローを図３に示す。

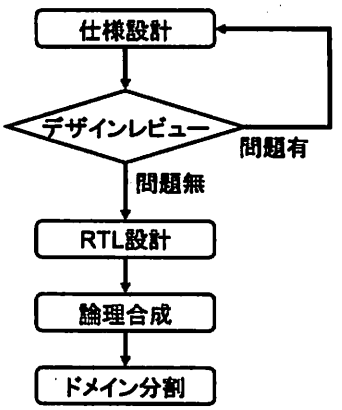

図３　アーキテクチャレベルにおける設計フロー

3.1. 仕様設計

仕様は、32bit×32bit の乗算器とし、被演算データ依存による細粒度でのスリープ制御を行う。

２．２において述べた Case ごとに Domain の制御を行うために、制御回路内で乗算器の出力を格納するレジスタのイネーブル信号とオールゼロ検出信号を使用し、各 Domain のパワースイッチ制御信号を生成する（図４）。乗算器の出力をレジスタに格納しないのであれば演算を行う必要はなく、乗算器全体をスリープさせることが出来る。そのため、乗算器の出力を格納するレジスタのイネーブル信号を使用している。

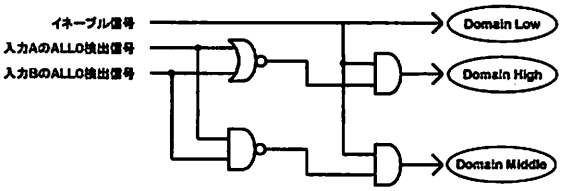

図４　パワースイッチの制御論理図

また、この段階において複数人で検証を行い、設計精度を高めるデザインレビューを行った。

3.2. RTL 設計

決定した仕様と制御方法をもとに、乗算器、制御回路、オールゼロ検出回路等を、ハードウェア記述言語 Verilog-HDL を使用して設計を行った。

3.3. 論理合成

ドメイン分割を行うために、RTL 記述の論理合成を行った。論理合成には、シノプシス社の Design Compiler を使用した。

3.4. ドメイン分割

論理合成後のネットリストをもと Fan-in 探索を行う。演算結果を格納するレジスタの Fan-in を被演算データを格納しているレジスタまで探索を行う。探索を行った結果を、各 Domain に属する論理ゲートを求めるために、以下の Group1, Group2, Group3 に分けた。

31bit～0bit の論理演算を行う論理ゲート群：Group1
47bit～32bit の論理演算を行う論理ゲート群：Group2
63bit～48bit の論理演算を行う論理ゲート群：Group3

上位の論理演算を行う論理ゲートの中には、下位の論理演算にも共通して使うゲートが含まれる。すなわち、上記の Group には共通して含まれるゲートが存在する。その為、得られた論理ゲート群をもとに下記の方法で、独立にスリープ制御を行える Domain High, Domain Middle, Domain Low にドメイン分割する。

Domain Low = Group1
Domain Middle = Group2 - Group1
Domain High = Group3 - Group1 - Group2

4. チップ評価のための回路の設計

4.1. 試作するチップの構成

今回の手法を適用した乗算器の試作を行うにあたり、実チップにしたときに評価を行えるように、チップ外部から計測を行えるようにデータの入出力用のモジュール、およびテストを行うための制御回路の設計を行った。

試作を行うにあたり、チップで使用できる入出力パッドの数には制限がある。このため、乗算器への被演算データの入力と演算結果の出力をチップ外部とするために必要となるパッド数が足りず、接続することが出来なかった。その為、入力、出力それぞれにおいて使用するパッドを削減する必要がある。入力におけるパッドの削減方法は、チップ外部から被演算データをパラレル入力で与えるのではなく、シリアル入力でシフトレジスタに一度格納し、そこからデータを読み出し使うことによって乗算器へ入力与えることで削減できる。出力におけるパッドの削減方法は、演算結果 64bit をそのまま出力するのではなく MUX により 8bit ごとに切り替えながら出力することで削減できる。図5は、入力用モジュール、出力用モジュールともに乗算器周辺回路として含めた試作におけるモジュール構成図である。

SHIFT_FF ： 外部からの入力データを保持
MUX4 ： 入力データを切り替え
MIR ： 入力データ用レジスタ
ALL0 ： オールゼロ検出回路
CTR ： 制御回路
MOR ： 出力データ用レジスタ
MUX ： 出力の切り替え

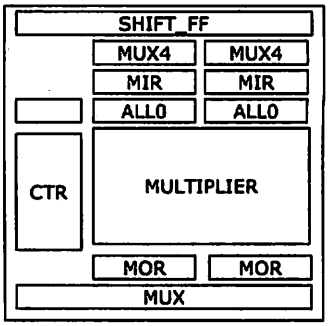

図5　試作用周辺回路を含めたモジュール構成図

また、チップによる測定を行うために、スリープ期間や被演算データを切り替える測定用のモードを定義し、ハードワイヤード方式による制御モードとして制御回路に実装した。

実装したモードは全部で 26 種類であるが、Manual mode と Auto mode に大別することが出来る。

4.2. 計測における制御モード

4.2.1. Manual mode

チップ外部からシリアル入力でシフトレジスタにデータを格納し、そのデータをもとに被演算データの切り替えとスリープ期間を制御するモードである。外部からシフトレジスタに格納するデータを変えることによって、自由度の高い計測を行うことが出来る。パラレルで入力するのに対し、シリアルで入力するためシフトレジスタに格納するまでに最低でも 64 サイクルの時間を要してしまう。そのため、被演算データを高速に切り替えながら計測を行えるように2パターンのデータを最初にシフトレジスタに格納し、図6で示しているように、MUX4 の切り替えを行うことによって計測を行う。

4.2.2. Auto mode

MUX4 において、オールイチのデータとオールゼロデータを選択できるようになっている（図6）。何らかの原因で、チップ外部からレジスタに格納できない場合、チップ内に仕込んであるオールイチデータまたはオールゼロデータを使い、最低限の計測を行えるモードである。

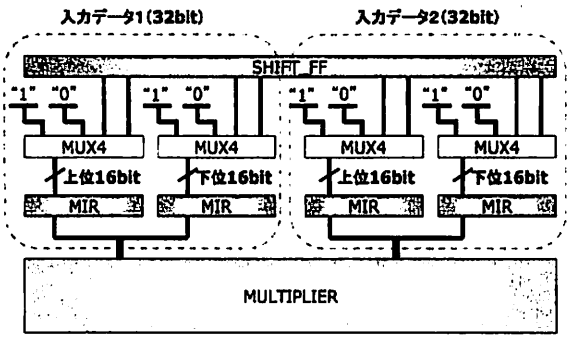

図6　乗算器への被演算データの入力図

5. シミュレーション評価と結果

本稿で被演算データによる細粒度 RTPG を適用した乗算器を提案した。提案した乗算を MIPS アーキテクチャ CPU に適用した場合、プログラム実行時の命令総ステップ数における各ドメインのスリープ期間の評価を行った。

MIPS アーキテクチャ CPU である R3081 プロセッサを評価用 CPU ターゲットとした。評価方法は、評価用プログラム実行時における MULT 命令の頻度と MULT 命令実行時におけるオペランドの値を分析することにより、各ドメインのスリープ期間を得る。

5.1. シミュレーション環境

R3081 プロセッサにおけるアセンブラの命令レベルシミュレータとして、ISIS を使用した。ISIS とは、慶応大学天野研究室が開発し、提供している並列計算機をターゲットとした、計算機シミュレータのための C++言語によるクラスライブラリである[5]。ISIS にはサンプルシミュレータとして、単一の R3081 プロセッサからなる計算機シミュレータを実装している。クラスライブラリなので必要に応じて変更可能である。今回、慶応大学天野研の協力を頂き、命令の総ステップ数、MULT 命令の頻度、MULT 命令実行時におけるオペランドの値を確認できるようにカスタマイズし、シミュレータとして使用した。

シミュレータを動作させるためには、評価用プログラムを、R3081 プロセッサ向けバイナリファイルにクロスコンパイルを行う必要がある。クロスコンパイラとして、ISIS とともに提供されている OSIRIS を使用した。

5.2. 評価用プログラム

C 言語ソースで公開されている、CPU ベンチマークプログラムの Dhrystone と、組み込み用途向け CPU ベンチマークプログラムである Mibench[6]を評価用プログラムとした。

MiBench の構造は、35 種類の組み込みアプリケーション・ソフトウエアを 6 種類の応用分野に分類して構成されている。自動車・産業機器，民生機器，OA 機器，ネットワーク機器，セキュリティ機器，通信機器，という分類である。

Mibench のベンチマークセットの中から、現行の ISIS と OSIRIS で対応できるプログラムを抽出し、シミュレーションを行った。

本稿で、評価に使用した Dhrystone ならびに Mibench の組み込みアプリケーション・ソフトウェアは以下のプログラムである（図 7）。また、scanf で格納する変数に対してはプログラム中に変数へ直接数値を入れる記述に変更を行った。

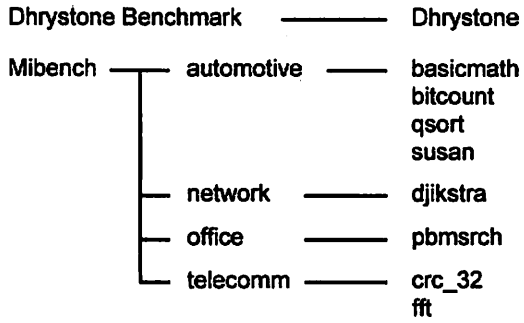

図7　評価用プログラム一覧

5.3. オペランドのオールゼロに着目した頻度解析

評価用プログラムを実行した際の MULT 命令実行時におけるオペランド値の解析を行った。また解析を行った結果、プログラム中で MULT 命令が使用されていないプログラムがあった。bitcount, qsort, susan, pbmsrch, crc_32, fft プログラムにおいて MULT 命令が使用されていなかった。

図 8 のグラフは、本手法において着目している上位 16bit がオールゼロになる頻度を解析した結果を表している。今回の解析によって、一方または両方の上位 16bit がオールゼロとなるオペランドの値が MULT 命令全体の 80%以上を占めることが明らかとなった。その結果、本手法を適用した場合の低消費電力化効果が高いことが分かる。

また、MULT 命令におけるオペランドの値はプログラムに依存する。シミュレータやクロスコンパイルにおいて実行することが出来ず解析を行うことが出来なかったが、JPEG 圧縮を行うプログラムなどは乗算を行う回数が多いと思われ、より本手法の適用効果が高いと

予測される。今後、Mibench に含まれる JPEG 圧縮のプログラムなどにおいても解析できる環境を整え、評価を行うことが課題である。

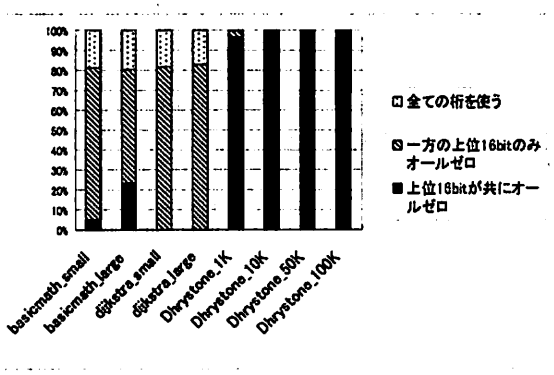

図8　オペランドのオールゼロに着目した頻度解析結果のグラフ

6. 結論

32bit×32bit 乗算器においてオールゼロ×オールゼロの演算の行う回路部分を被演算データそれぞれに対して上位 16bit のオールゼロ検出を行うことにより、細粒度でのパワーゲーティングを行う、RTPG 手法を提案し、設計を行った。

また、シミュレーションによる MULT 命令におけるオペランドの値の解析を行った。32bit×32bit の全ての桁を使う場合はすくなく、被演算データがともに上位 16bit がオールゼロ、または片方がオールゼロとなる頻度が多いことが明らかとなった。

謝辞

ISIS, OSIRIS を使用した評価において、多大なる協力を頂きました慶応大学天野研究室　天野英晴教授，関直臣氏に心より感謝致します。

本研究は東京大学大規模集積システム設計教育研究センターを通し、シノプシス株式会社、日本ケイデンス株式会社、メンター株式会社、京都大学の協力で行われたものである。

本チップ試作は東京大学大規模集積システム設計教育研究センターを通し、株式会社半導体理工学研究センター、富士通株式会社、松下電器産業株式会社、NEC エレクトロニクス株式会社、株式会社ルネサステクノロジ、株式会社東芝の協力で行われたものである。

文献

[1] David E. Lackey, Paul Rs. Zuchowski, Juergen Koehl "Designing Mega-ASICs in Nanotechnology", DAC, no 46.1, pp 770-775, June 2003

[2] S. Mutoh et al, " 1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold Voltage CMOS", IEEE J. Solid-State Circuits, vol. 30, no. 8, pp. 847-854, Aug. 1995.

[3] Z. Hu, et al, "Microarchitectual Techniques for Power Gating of Execution Units", Proc. ISLPED' 04, pp. 32-37, 2004.

[4] K. Usami , N. Ohkubo "A Design Approach for Fine-grained Run-Time Power Gating using Locally Extracted Sleep Signals" ICCD , Oct 2006

[5] "ISIS Homepage"
URL : http://www.am.ics.keio.ac.jp/isis/

[6] "Mibench"
URL : http://www.eecs.umich.edu/mibench/