システムＬＳＩ設計技術　94－17
(2000. 1. 12)

FPAccA model 2.0チップの設計
—再構成可能な浮動小数点演算器アレイ—

河野 陽一　越智 裕之　津田 孝夫

広島市立大学 情報科学部 情報工学科

〒731-3194 広島市 安佐南区 大塚東 3-4-1
Tel:082-830-1550
E-mail: {yoichi, ochi, tsuda}@arch.ce.hiroshima-cu.ac.jp

あらまし　従来のFPGAにおける配置配線時間の増大などの問題を解消するため、より基本セルの粒度を粗くしたアーキテクチャとしてField Programmable Accumulator Array (FPAccA)が提案されている。本稿では、演算器としてIEEE規格754に準拠する単精度浮動小数点演算器を実装したFPAccA model 2.0チップを設計、評価した。設計したFPAccA model 2.0チップは、0.35μmプロセスで4.9mm角のチップ上に加減算器9個、乗算器3個を実装し、動作周波数100MHzにて300MFLOPSの理論最大性能が見込まれる。

キーワード　FPAccA、FPGAアーキテクチャ、再構成可能システム、浮動小数点演算器、チップ設計事例

Design of FPAccA model 2.0 Chip
—Reconfigurable Floating-Point-Unit Array—

Yoichi KAWANO, Hiroyuki OCHI, and Takao TSUDA

Department of Computer Engineering
Faculty of Information Sciences, Hiroshima City University

3-4-1, Ozukahigashi, Asaminami-Ku, Hiroshima, 731-3194, Japan
Tel:+81-82-830-1550
E-mail: {yoichi, ochi, tsuda}@arch.ce.hiroshima-cu.ac.jp

Abstract　Field Programmable Accumulator Array (FPAccA), a coarse-grain FPGA architecture, has been proposed to solve problems with conventional FPGAs such as increasing time taken for placement and routing. In this paper, FPAccA model 2.0 chip is designed and evaluated, which has a single-presision floating-point unit of IEEE 754 standard in each cell. Nine adders and three multipliers are implemented with 4.9 mm × 4.9 mm die size using 0.35 μm triple-metal CMOS process. Expected peak performace is 300 MFLOPS at 100 MHz clock flequency.

key words　FPAccA, FPGA Architecture, Reconfigurable System, Floating-Point Unit, Chip Design Experiments

1 はじめに

近年の集積回路技術の進歩に伴い、1チップ上にきわめて高機能かつ高性能なシステムが実現できるようになりつつある。このような中で、小量多品種、短TATなどの要請に応えるべく、Programmable Logic Device (PLD)やField Programmable Gate Array (FPGA)、さらにはLaser Programmable Gate Array (LPGA)など、様々なプログラマブルデバイスが開発され、実用化に至っており、今後ますますその可能性が広がっていくものと期待される。中でもSRAMベースのFPGAは、構成情報をSRAMに格納し、繰り返し設計変更が出来る柔軟性があることが特徴であり、ラピッドプロトタイピング、ハードウェアエンジンなどの分野でも利用されつつある。しかし近年、1チップのFPGAに実装可能な回路が大規模化するに伴い、構成情報の自動生成に要する時間が増大しつつあることが深刻な問題になっている。

この問題を解決する手段の一つとして、FPGA基本セル中のLUT (Look-Up Table)をALUに置き換え、粒度を粗くすることで構成情報の減少と構成情報生成時間の短縮を狙ったField Programmable Accumulator Array (FPAccA)アーキテクチャ[1]が提案されている。また、このFPAccAアーキテクチャに基づくFPAccA model 1.0チップが試作され構成情報の大幅な減少と、情報生成時間の大幅な短縮が確認されている。FPAccA model 1.0チップは演算器として8bit整数演算器のみを実装している。しかし、実際の応用では浮動小数点演算の需要も高まりつつあり、浮動小数点演算器を実装することで応用分野を広げることができるものと期待できる。

そこで本稿では、IEEE規格754[2]に準拠した単精度浮動小数点加減算器9個、同乗算器3個を有したFPAccA model 2.0チップを0.35μmプロセスルール、メタル3層CMOSテクノロジ上に実装した。周波数は100MHz、理論最大性能で300MFLOPSが見込まれる。

以下、2章では準備としてFPAccAとFPGAのアーキテクチャの相違、3章ではアーキテクチャの基本設計、4章ではFPAccA model 2.0チップの詳細設計と評価、5章で考察を述べる。

2 準備

FPGAの基本セルを図1(a)に示す。FPGAの基本セルは、数個のプログラマブルな組合せ回路と、オプショナルな出力フリップフロップからなるモジュールである。FPGAはこの基本セル、および、これらが相互接続するためのプログラマブルな配線リソースから構成されている。これに対し図1(b)に示すFPAccAは、プログラマブルなALUと、その間を接続するプログラマブルなバスリソースのみからなる。つまり、従来のFPGAよりもかなり粒度の粗いプログラマブルアレイである。

これらFPGAとFPAccAのアーキテクチャ上の違いによりFPAccAでは以下のような効果が期待される。

(1) 構成情報の自動生成がきわめて短時間で可能

FPGAで特に問題となっているのは配置配線の所要時間である。FPAccAはFPGAよりもはるかに少ないセル数で複雑な数値計算アルゴリズムを実現できる。このため、自動配置配線に要する時間は大幅に短縮することができる。

(2) 算術演算中心の応用では速度、実装密度で有利

FPGAでは算術演算機能を実現するために多数のLUTと膨大な配線リソースを割いて算術演算モジュールを構成しなければならない。しかし、FPAccAでは複雑な機能を持った高性能な演算器があらかじめ作り込まれておりFPGAに対して速度、実装密度の点で有利である。

(3) 高級言語との親和性が高い

C言語やFORTRANなどの高級言語で書かれたプログラムは固定長のデータに対して四則演算などの限られた演算を繰り返すことに終始している。そのため、高級言語で書かれるようなアルゴリズムはFPGAよりはむしろFPAccAを用いた方が容易にマッピングできる。

(4) 構成情報がコンパクト

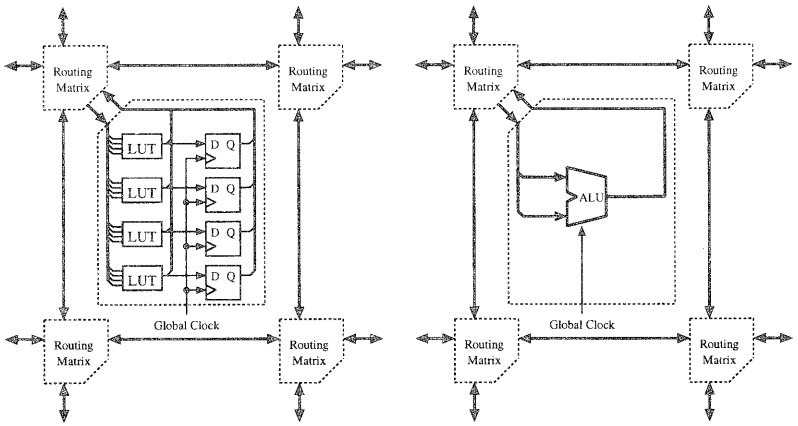
FPGAで演算器のような複雑な回路を構成する場合、多くの構成情報を必要とする。それに対して、FPAccAではALU部分については演算の選択のみを行ない、バス部分については配線リソースを1本ずつではなく基本語長単位で設定する。そのため、FPGAに比べ必要な構成情報は桁違いにコンパクトとなる。

文献[1]で試作、評価されたFPAccA model 1.0チップは図1(b)のFPAccAアーキテクチャに基づいており、各基本セルは8bit整数演算のみをサポートしている。このFPAccA model 1.0チップでは、上記(1)および(4)の優位性が実証されている。

3 アーキテクチャの基本設計

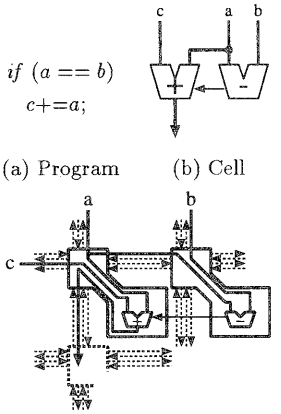
本稿では、基本セルに単精度浮動小数点演算機能をもたせたFPAccA model 2.0の設計を扱う。この章では、その基本的なアーキテクチャについて検討する。

2章で述べたように、FPAccAは高級言語との親和性を狙いの一つとしている。そのためには算術演算だけではなく、条件分岐処理や、繰り返し処理をマッピングできる必要がある。これらの処理を実現する方法として、あるセルで算術演算の結果に基づいた条件フラグを生成し、これを用いて他のセルの動作を制御できるようにし



(a) FPGA　　(b) FPAccA

図 1: FPGA と FPAccA のアーキテクチャの模式図



(a) Program　　(b) Cell

(c) Cell Map

図 2: else 節を伴わない条件分岐処理例

ておくことが考えられる。ここでは、条件フラグとしてプラス、マイナス、ゼロの 3 つのフラグを考え、条件フラグによる制御機能として「演算前のデータの通過」および「演算結果の廃棄」の 2 つを考える。

3.1 条件分岐処理

3.1.1 else 節を伴わない if 文

最も単純な条件分岐処理の例として、図 2(a) の C 言語で書かれたプログラムについて考える。対応する論理的なセル間の接続を図 2(b)、FPAccA model 2.0 チップ上での物理的なバスのトラック割り当て例を図 2(c) に示す。

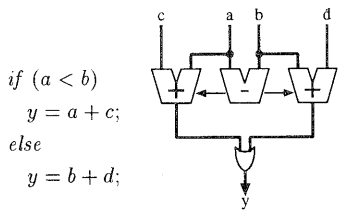
図 2(a) のように else 節を伴わない if 文をハードウェアで実現する場合には、条件外であれば演算をしない処理が必要である。FPAccA model 2.0 チップでは、条件フラグの制御機能の 1 つである「演算前のデータの通過」を行なうことでこれを実現している。

図 2(b) 右のセルは、図 2(a) のプログラムの $a == b$ を実現している。a と b の値を入力し、$a-b$ が行なわれることで、演算の結果に依存したフラグが生成され、左のセルに送られる。図 2(b) 左のセルは、右のセルからのフラグが入力されるまで処理を保留し、そのフラグがゼロであった場合 ($a = b$ であった場合) に加算結果 $(c+a)$ を出力し、それ以外のフラグの場合 ($a < b$ または $a > b$ であった場合) は演算前のデータ c をそのまま出力する。

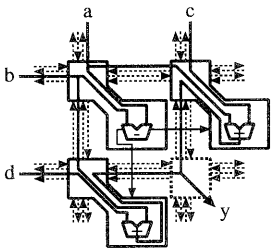
3.1.2 else 節を伴う if 文

else 節を伴う if 文の例として、図 3(a) の C 言語で書かれたプログラムについて考える。対応する論理的なセル間の接続を図 3(b)、FPAccA model 2.0 チップ上での物理的なバスのトラック割り当て例を図 3(c) に示す。

図 3(b) 中央のセルは、図 3(a) のプログラムの $a < b$ を実現している。a と b の値を入力し、$a-b$ が行なわれることで、演算の結果に依存したフラグが生成され、左右のセルに送られる。図 3(b) 左のセルは図 3(a) のプログラムの $if(a < b)\ \ y = a + c;$ の場合の処理を実現している。このセルは中央のセルからのフラグが入力されるまで処理を保留し、そのフラグがマイナスであった場合 ($a < b$ であった場合) に演算結果を出力し、それ以外のフラグの場合 ($a \geq b$ であった場合) は演算結果を廃棄する。図 3(b) 右のセルは図 3(a) のプログラムの $else\ \ y = b + d;$ の処理を実現している。このセルに入力されたフラグがプラス、ゼロであった場合 ($a \geq b$ であった場合) に演算結果を出力し、それ以外の場合 ($a < b$ であった場合) は演算結果を廃棄する。

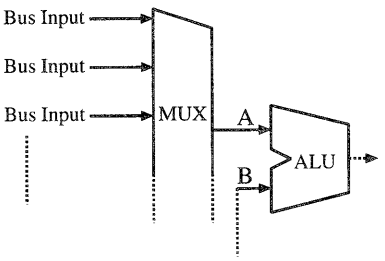


(a) Program (b) Cell

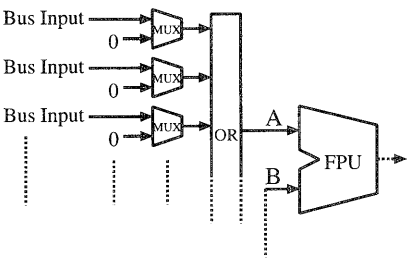


(c) Cell Map

図 3: else 節を伴う条件分岐処理例



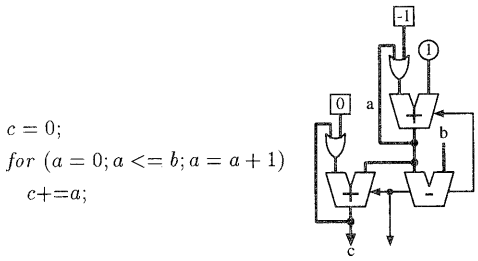
(a) FPAccA model 1.0



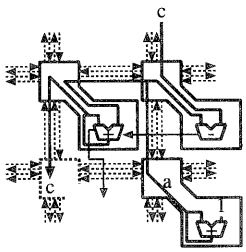
(b) FPAccA model 2.0

図 4: 演算器入力部の選択回路

演算結果はこれら 2 つのセルのいずれか一方から出力されるので、最後にこれらを 1 つにまとめる必要がある。つまり、図 3(b) の OR ゲートに相当するものが必要となる。FPAccA model 2.0 チップでは、これを次段の演算器の入力部で実現することとした。図 4 に、FPAccA model 1.0 チップおよび model 2.0 チップの演算器の入力部のアーキテクチャを比較して示す。FPAccA model



(a) Program (b) Cell



(c) Cell Map

図 5: 繰り返し処理

1.0 チップでは、図 4(a) に示すように複数の入力バスからあらかじめ設定されたただ 1 つを選択するためのマルチプレクサがあった。それに対し、FPAccA model 2.0 チップでは図 4(b) のように、演算器に入力してくるバスそれぞれにあらかじめ個別に設定可能なマルチプレクサが存在し、それら全てのバスの論理和をとったものを演算器に与えている。これにより図 3(b) の OR ゲートを実現することが可能となっている。

3.2 繰り返し処理

次に繰り返し処理の例として図 5(a) の C 言語で書かれたプログラムについて考える。対応する論理的なセル間の接続を図 5(b)、FPAccA model 2.0 チップ上での物理的なバスのトラック割り当てを図 5(c) に示す。

図 5(b) 上段のセルは図 5(a) のプログラムの $a = a + 1$ を実現しており、a の値に 1 を足し込む処理を繰り返し行なっている。その下のセルは繰り返し処理の終了条件である a が b を越えているかどうかについて $a - b$ を演算することで判別し、フラグを出力している。下段左のセルでは、入力されたフラグがマイナスかゼロの場合 ($a \leq b$ であった場合) $c+ = a$; が実行され、それ以外のフラグの場合 ($a > b$ であった場合) 演算結果の値を廃棄する。このフラグは図 5(b) に示すように演算結果の c を受けとるセルに対しても出力され、最終結果が出力されたタイミングの判別に使われる。つまり、フラグがマイナスの場合は、まだ演算が終了しておらず、フラグがゼロであれば出力先のセルは現在受けとっている値が有効な値で

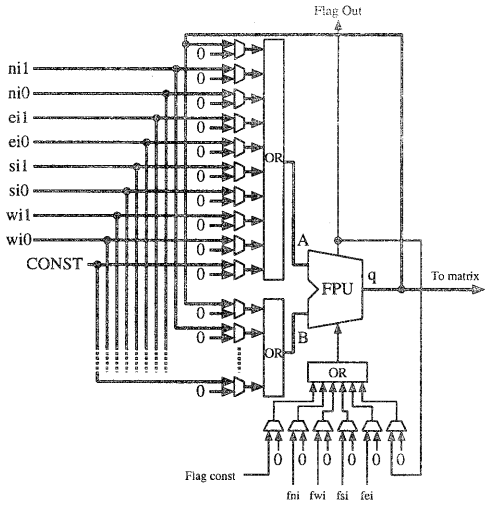


図 6: アキュムレータ

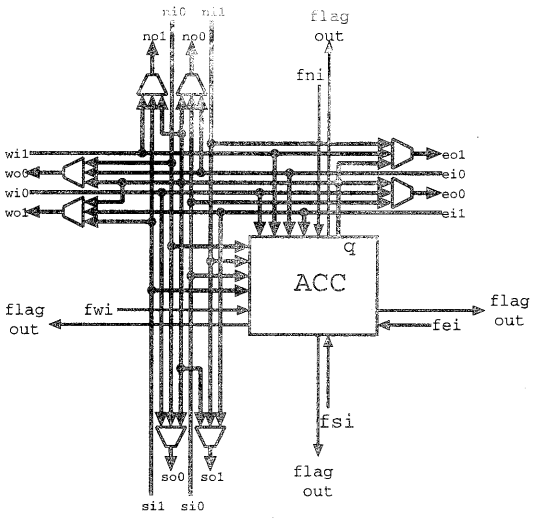


図 7: セル

あると分かる。

図 5(a) のプログラムの 1 行目にある $c = 0$ は、演算開始時に 1 度だけ実行されなければならない。このように、1 度だけ必要とされる定数はしばしば必要となる。他方、2 行目の $a = a + 1$ の右辺に出現する定数 1 は、実行中に繰り返し必要とされる。そこで FPAccA model 2.0 チップでは、コンフィグレーション時に構成情報によって設定可能な定数をセル毎に持たせることとした。この定数は演算開始時に 1 度だけ出力させたり、任意の間隔 (クロック) で繰り返し出力させたりすることができるようにしてある。図 5(b) 中、四角で囲まれた数字は 1 度だけ出力される定数を表し、丸で囲まれた数字は繰り返し出力される定数を表している。

4 FPAccA model 2.0 チップの詳細設計と評価

4.1 詳細設計

4.1.1 アキュムレータ

本稿では演算器 (FPU) および、FPU の入力選択回路からなるモジュールをアキュムレータと呼ぶ。図 6 にアキュムレータのアーキテクチャを示す。

FPU は、A と B の 2 つの入力を持ち、A、B それぞれには、「8 本のバス (図 6 中「ni1」から「wi0」の 8 本)」、「構成情報により設定される定数 (図 6 中「CONST」)」、「アキュムレータ自身の演算結果 (図 6 中「q」)」の 3 種類 (A、B 入力各 10 本) を与えることが可能である。FPU へのフラグ入力については、「東西南北に隣接する別のアキュムレータが出力したフラグ (図 6 中「fei」、「fwi」、「fsi」、「fni」)」、「構成情報により設定される定数 (図 6 中「Flag const」)」、「FPU のフラグ出力を同じ FPU のフラグ入力にする」の 3 つが使用できる。それぞれを使用するかしないかはマルチプレクサに与えられた構成情報によって選択される。

なお、加減算器、乗算器ともに IEEE 規格 754 に準拠しており、丸め処理は最近似値への丸め (round to nearest even) のみをサポートしている。

4.1.2 バスのアーキテクチャ

バスのアーキテクチャを図 7 に示す。このアキュムレータとバスを含んだ範囲をセルと呼ぶ。

東西南北の隣接する各セルとはそれぞれ往復各 2 本のバスで結ばれる。図 7 のバス ni0、no0、ei0、eo0、si0、so0、wi0、wo0 から構成される配線リソースをトラック 0、バス ni1、no1、ei1、eo1、si1、so1、wi1、wo1 から構成される配線リソースをトラック 1 と呼ぶ。トラック 0 は「直進または左折」が可能な配線リソースであり、トラック 1 は「直進または右折」が可能な配線リソースである。図 7 中のマルチプレクサには入力として「直進してくるバス」、「左折 (または、右折) してくるバス」、「アキュムレータからの演算結果 (図 7 中 q)」の 3 つがあり、これらは与えられた構成情報により選択される。

4.1.3 外部入出力

図 8 にチップ全体のアーキテクチャを示す。FPAccA model 2.0 チップは加減算セル 9 個、乗算セル 3 個の 12 セルで構成されている。なお、外部入出力バスは試作チップのピン数の制約から図 8 中の各辺に 1 組 (入出力バス各 9bit、入出力フラグ各 3bit) ずつしかとれなかった。これ以外には演算用のクロック (1bit)、リセット (1bit)、構成情報転送用のクロック (1bit)、構成情報入力 (1bit) などがある。

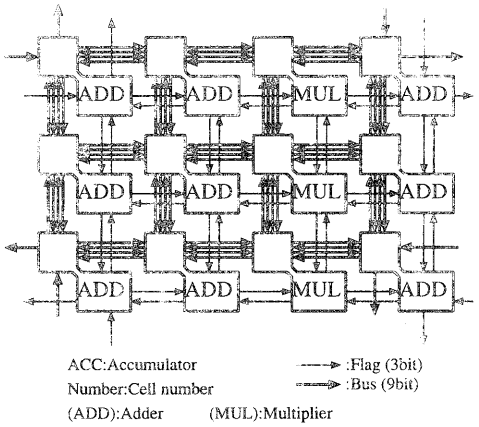


図 8: チップ

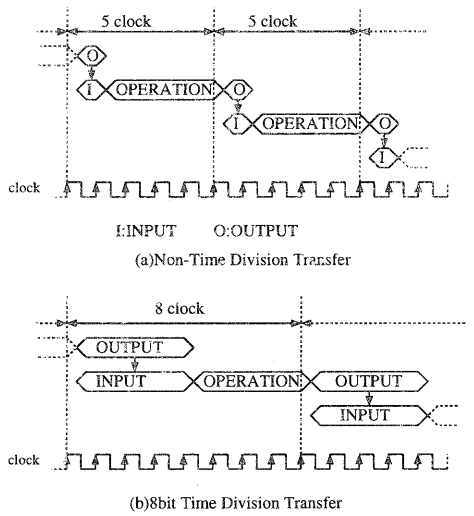


図 9: 実際の応用でのクロックサイクル

FPAccA model 2.0 チップはアレイ状に並べることにより入出力バスを他の FPAccA model 2.0 チップと結合することができる。これにより利用可能なセルを増やすことができ、多くのセルを必要とするアルゴリズムも実現することができる。

4.1.4 時分割データ転送

FPAccA model 2.0 チップは単精度浮動小数点演算を扱うため、本来であれば 32bit 幅のバスを必要とする。しかし 4.1.3 節で述べたように、試作チップのピン数の制約から、各辺に入力バス、出力バスを 32bit とることができなかった。そのため、今回の設計では時分割データ転送を採用している。時分割データ転送とはある長さのデータを一定の長さに分け、時分割で転送する方式である。この時分割データ転送を採用することで外部入出力ピンの使用数を抑えることができる。また、本来必要なバス幅よりも実際のバス幅を小さくすることができ、時分割データ転送を行なわない場合に対し実装面積の面でも若干有利であると考えられる。

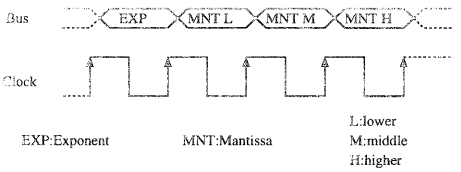


図 10: 時分割データ転送のタイミングチャート

一方、時分割データ転送は速度面で不利である。動作周波数を 100MHz と仮定し、時分割を行なわない 32bit 幅のバスを持つセルと、8bit の時分割データ転送を行なう FPAccA model 2.0 のセルを考えた場合、前者の理論最大性能は 100MFLOPS、後者の理論最大性能は 25MFLOPS であり 4 倍の差がある。

ここで、図 5(b) の a に 1 を足し込んでいるセルを例として、そのセルが時分割を行なわない 32bit 幅のバスを持つセルである場合と、8bit の時分割データ転送を行なう FPAccA model 2.0 のセルである場合を考える。時分割データ転送を行なわない場合は図 9(a) のように繰り返し周期が 5 クロックとなるため 20MFLOPS、時分割データ転送を行なった場合は図 9(b) のように繰り返し周期が 8 クロックとなるため 12.5MFLOPS である。つまり、両者の理論最大性能の差が 4 倍あるにも関わらず、この例ではその差は 1.6 倍にとどまっている。図 5 のような例は高級言語で書かれたアルゴリズムを FPAccA model 2.0 にマッピングする際にしばしば生じると考えられるため、実際の応用では時分割データ転送の使用による速度の影響は大きくないと考えることができる。

なお、FPAccA model 2.0 では時分割データ転送を行なう際に指数部の値を最初に転送し、その後で仮数部の値を転送している (図 10)。これにより、加減算の前処理におけるバレルシフトや、乗算の前処理における非正規化数の正規化などを全てのデータがそろう前から始めることが可能となり、動作周波数や面積の改善に寄与している。

4.2 評価

FPAccA model 2.0 チップの設計は、ローム社の 0.35μm メタル 3 層 CMOS テクノロジで約 4.9mm 角のチップ (コアのサイズは約 3.7mm 角) をターゲットとし、スタンダードセル方式にて行なった。設計記述は Verilog-HDL で約 2,000 行程度であり、回路規模はトランジスタ数約 62 万

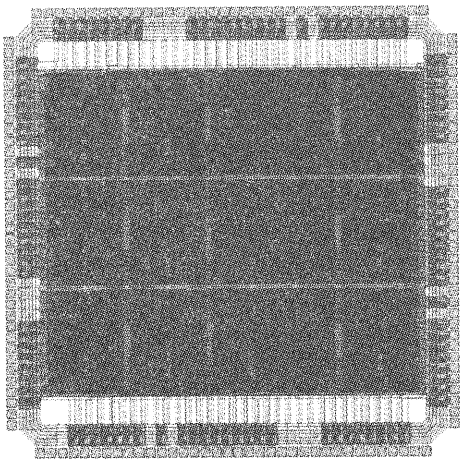


図 11: FPAccA model 2.0 チップのレイアウト図 (4.93mm×4.93mm)

個となった。図 11 にレイアウトを示す。加減算 1 セルの大きさは 801μm × 1,080μm、乗算 1 セルの大きさは 1,284μm × 1,080μm である。

FPAccA model 2.0 チップの動作周波数は 100MHz である。セル 1 個あたりの理論最大性能は、加算セル、乗算セルともにパイプラインピッチが 4 クロックであることから 25MFLOPS であり、チップ全体 (12 セル) では、300MFLOPS となる。

5 考察

FPAccA に期待される利点として 2 章で以下の 4 項目をあげた。

(1) 構成情報の自動生成がきわめて短時間で可能
(2) 算術演算中心の応用では速度、実装密度で有利
(3) 高級言語との親和性が高い
(4) 構成情報がコンパクト

以下、これら 4 項目について考察を行なう。

(1)、(3) に関しては、まだ FPAccA model 2.0 チップ用の自動配置配線ツールやコンパイラ等が未開発のため、現時点では評価できなかった。しかし、(1) に関しては従来の FPGA に対してセルの数が大幅に少ないため構成情報の自動生成時間の大幅な短縮が見込まれる。(3) に関しても浮動小数点演算を行なうことができ、3 章で述べた通り、条件分岐や繰り返し処理を容易にマッピングすることができると考えられる。

(2) の面積については、文献 [4] によれば、FPGA で浮動小数点加減算器を実装しようとした場合 FPGA(Xilinx 社 XC4010) が 2 個、乗算器を実装しようとした場合も 2 個必要としている。これに対して FPAccA model 2.0 チップでは 1 チップ上に単精度浮動小数点加減算器 9 個、同乗算器 3 個の実装に成功している。これだけの数の演算器の実装は最近の FPGA においても複数チップへの分割実装は免れ得ないものであると考えられ、従来の FPGA に対して実装面積において明らかに有利であると言える。

速度については、今回は 100MHz 動作、パイプラインピッチが 4 クロックのものを開発した。最先端のプロセステクノロジ上でフルカスタム設計すれば、クロック周波数をさらにこの 3 倍程度にすることは可能であると考えられる。また、入出力ピン数を増やすか、オンチップキャッシュを搭載すれば時分割データ転送の必要がなくなり、パイプラインピッチを 1 クロックにすることが可能である。このようなパフォーマンスを従来の FPGA に期待することはほぼ不可能であり、商用の汎用プロセッサなどにかなり近いレベルになったと言える。

(4) の構成情報の大きさについては、文献 [4] によれば FPGA(Xilinx 社 XC4010) 上に浮動小数点加減算器、浮動小数点乗算器を実装するために構成情報をそれぞれ約 360kbit 必要としている[1]。これに対して FPAccA model 2.0 チップでは浮動小数点加減算器、浮動小数点乗算器があらかじめ作り込まれているため、アキュムレータ 1 つを構成するために (加減算器、乗算器ともに)110bit、1 セル分のバスを構成するために 16bit 使用し、1 つのセルで 126bit、チップ全体ではたかだか 1.5kbit の構成情報で 12 個の演算器をプログラムできる。これは、FPGA の構成情報に対して格段に小さく、狙い通りの効果が出ていることが分かる。

6 おわりに

FPAccA は、基本セルの粒度を粗くすることで構成情報の自動生成時間の短縮などを狙いとしているアーキテクチャである。本稿では、この FPAccA アーキテクチャに基づき IEEE 規格 754 に準拠する単精度浮動小数点演算器を実装した FPAccA model 2.0 チップを設計した。

同チップは加減算セルを 9 個、乗算セルを 3 個有し、チップ全体の理論最大性能は 300MFLOPS が見込まれる。またこのチップは、高級言語との親和性を考慮し、条件分岐処理や、繰り返し処理のマッピングを容易にするための条件フラグ機能を備えている。既存の FPGA を使用した場合、1 チップ上に 12 個もの浮動小数点演算器を実装したり、1 チップで 300MFLOPS の性能を達成することは困難と考えられる。チップのコンフィグレーションに必要な構成情報量に関しても、従来の FPGA に対して格段に小さくすることができた。

今回は、構成情報の自動生成時間、高級言語との親和性、スケーラビリティの評価を行なうことができなかった。今後開発を行なう自動配線ツールや、コンパイラを用いてこれらの評価をおこなうことが今後の課題である。

[1] この値は XC4010 1 個あたりの構成情報サイズ [2] に所要 FPGA 数を乗じた値である。

謝辞 FPAccA model 2.0 チップの試作は東京大学大規模集積システム設計教育センター (VDEC) を通しローム (株) および凸版印刷 (株) の協力で行なわれたものであり、このような機会が与えられたことに心より感謝いたします。

参考文献

[1] 越智裕之: 「FPAccA:フィールドプログラマブルアキュムレータアレイ—FPAccA model 1.0 チップの設計と評価」、情報処理学会論文誌、vol.40、no.4、pp.1717-1725、1999.

[2] IEEE: IEEE Standard for Binary Floating-Point Arithmetic, 1985. ANSI/IEEE Std 754-1985.

[3] Xilinx Inc.: Programmable Logic Data Book 1996, San Jose, California (1996).

[4] 井上 弘士、中垣 憲一、大内 正英、柊山 太一朗、久我 守弘、末吉 敏則: 「DLX-FPGA マイクロプロセッサにおける浮動小数点パイプラインの実現」、情報処理学会研究報告会、95-ARC-110-19、1995。