システムＬＳＩ設計技術　99－1
（2001. 1. 11）

仮想ハードウェア HOSMII のための DRAM 型マルチコンテキスト FPGA の設計

川上 大輔† 柴田 裕一郎† 天野 英晴†

†慶應義塾大学大学院 理工学研究科 計算機科学専攻
〒 223-8522 横浜市港北区日吉 3-14-1
E-mail: {kawakami, shibata, hunga}@am.ics.keio.ac.jp

あらまし

仮想ハードウェアシステム HOSMII は、複数の配線情報メモリをチップ内に持たせたマルチコンテキスト FPGA を用い、データ駆動的に制御し回路の仮想化を行なうシステムである。FPGA/CPLD に DRAM を混載させることで、大量の配線情報データをチップ内に格納することができ、従来の SRAM 型 FPGA で問題となっていた回路の再構成時間を大幅に短縮させることが可能である。しかし、DRAM を混載させた FPGA の研究、試作は行なわれているが、一般に利用できるものではない。そこで今回、DRAM 混載型マルチコンテキスト FPGA(HOSMII チップ) の試作、設計を行なった。HOSMII チップは電源電圧が 3.3V、Metal3-Poly2 層の 0.35μm プロセスであり、チップ面積は 4.93mm^2 である。またチップ内にコンテキストを 16 面格納することが可能である。簡単なシミュレーションから、論理ブロックを 100MHz で動作させることができ、回路の再構成時間も 5ns と高速に行なえることが確認できた。本稿では HOSMII チップのアーキテクチャについて述べ、性能評価を検討する。

キーワード　仮想ハードウェア, データ駆動, マルチコンテキスト FPGA, DRAM 混載

Design of DRAM type Multi-context FPGA for Virtual Hardware HOSMII

D. KAWAKAMI† Y. SHIBATA† H. AMANO†

†Dept. of Information and Computer Science, Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan
E-mail: {kawakami, shibata, hunga}@am.ics.keio.ac.jp

Abstract

HOSMII is a virtual hardware system with a data driven control mechanism. It uses a multi-context FPGA which provides multiple sets of configuration memory inside the chip. Using FPGA/CPLD embedded with DRAM enables us to keep a lot of configuration memory sets in a chip. Though research prototypes of such a chip have been implemented, these devices are not optimized for HOSMII mechanism. Here, a multicontext DRAM FPGA chip called HOSMII chip is implemented for HOSMII mechanism. HOSMII chip is a 0.35μm standard cell CMOS with metal-3 poly-2 layer. It provides 16 contexts in 4.93mm^2. Electric simulation shows the logic block works at 100MHz with 3.3V voltage supply. Also, it takes about 5ns to reconfigure the whole circuits.

key words　virtual hardware, data driven, multicontext FPGA, embedded DRAM

1 背景

近年LSIのプロセス技術の発展により、従来別々のプロセスで製造されていたDRAMとロジックを同一チップ上に混載することが可能となった。この技術は、マイクロプロセッサやマルチメディア用LSIのメモリと演算装置のバンド幅を飛躍的に大きくすると共に実装面積、消費電力を節約できるため、既に様々な分野で応用されている。一方、近年ユーザが自分の手で自由にディジタル回路をプログラムすることのできるPLD、FPGAが急速に発達および普及し、特にダイナミックに結線情報を変更することのできるSRAM型は、アルゴリズムを直接ハードウェア化して実行するReconfigurable MachineあるいはCustom Computing Machineへの応用が盛んである。さらに、Reconfigurable Computingで用いるために、従来のSRAM型のFPGAを拡張したマルチコンテキストFPGAと呼ばれるデバイスの研究も進んで行われている[1]、[2]。マルチコンテキストFPGAは、図1に示すように、複数セットの配線情報をチップ内に持たせ、一定の機構で回路を入れ替えることができるデバイスである。

このようなFPGAを用いてハードウェアを仮想化する研究についても現在盛んに行われている。我々もデータフロー機構を利用してハードウェアの仮想化を行うシステムWASMII[3]およびHOSMII[4]を提案し、研究を進めている。このうちHOSMIIはFPGA/PLDとDRAMを混載させることで、大量の配線データをチップ内に格納し、それらを動的に入れ換えることにより仮想ハードウェアを実現するシステムである。HOSMIIは、DRAMの容量の大きさを利用することにより従来のSRAMを用いたWASMIIに比べて場合によっては数十倍の性能を実現することができる。DRAMを混載させたFPGAはMITで研究されているDPGA[5]やNECで試作が行われたDRAM混載FPGA[6]などの例があるが、一般に利用できる状態ではない。また、仮想ハードウェアを実現するためにはチップ内に制御回路を組み込む必要がある。そこで小規模ではあるが、仮想ハードウェアで用いるためのDRAM混載型FPGAの試作、設計を行なった。本稿では2節で仮想ハードウェアシステムHOSMIIについての概要を述べる。3節で試作、設計したチップの概要とアーキテクチャについて説明し、4節でチップの性能評価について述べる。最後に5章で結論及び今後の課題について述べる。

2 仮想ハードウェアHOSMII

HOSMIIシステムはDRAM型マルチコンテキストFPGAを利用し、動的に回路を変更することでFPGAで実現できる回路を見かけ上大きくすることを可能にした仮想ハードウェアシステムである。一般にマルチコンテキスト

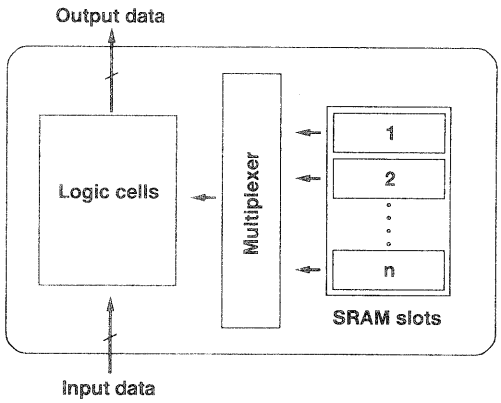

図 1: マルチコンテキストFPGA

FPGAだけでは仮想ハードウェアシステムを実現できない。これは任意のアルゴリズムをハードウェア化する手段が確立されていないことと一般的な論理回路ではコンテキストの切り替えのタイミング制御が困難であることに起因する。そこでページ切り替えを含む処理全体にデータ駆動制御の考えを採り入れた。まず対象とする問題をデータフローグラフに変換し、このグラフを結線情報ページひとつで実現可能なサイズのサブグラフに分割する。各サブグラフに対応する一連のページは、その全ての入力アークにトークンが到着し、実行可能になった時点でデバイス上に実現される。すなわち、それぞれのノードでは全ての入力アークにデータが揃うとその演算処理を行い、結果を次のノードの入力に転送する。

静的なデータフローグラフをそのままハードウェア化したならば、各演算ノード内の記憶素子は演算中の一時的なデータを記憶するだけでよく、演算終了後はその内容を保持する必要がない。これは静的なデータフローグラフが副作用を持たないことに対応する。したがって、コンテキストの切替え時に順序回路の状態が失われてしまうようなデバイスにも、システムの制御メカニズムを変更することなく対応可能である。また、状態ノードの導入によりデータフローグラフ内で局所的なループを構成するトークンを制御することができ、演算効率を高めることができる。

図2に、仮想ハードウェアHOSMIIの基本構成を示す。HOSMIIシステムは演算部と制御部の2つの部分に分けられる。演算部は分割されたサブグラフが実際に可変論理セル上に実現され演算処理が行われる部分である。各ページの結線情報は内蔵のDRAMに蓄えられており、次々と異なるサブグラフの結線情報を読み出していくことにより、動的なページの入れ替えを実現する。一方、制御部は、各ページへ入力されるトークンの待ち合わせを行う入力トークンレジスタや、次にアクティブにすべきページを定めるページ制御回路、各トークンの行き先を決定するトークンルー

タなどが構成される。制御部も演算部と同じようにFPGA上に構成されるため、トークンのルーティングテーブルなどアプリケーション固有の情報をハードウェア的に組み合わせ回路などで実現できる。また、アプリケーションに合わせてデータパスのビット幅などを最適に構成することができる。ただし、制御部の回路はアプリケーションを実行する前に一度だけ結線情報を読み出して回路を実現すればよく、アプリケーションの実行中に動的にその構成を変更する必要がない点が演算部と異なる。またDRAM領域は、制御回路の結線情報を保持するだけではなく、制御回路のデータメモリとしても使用され、例えば入力トークンレジスタなどがマッピングされる。

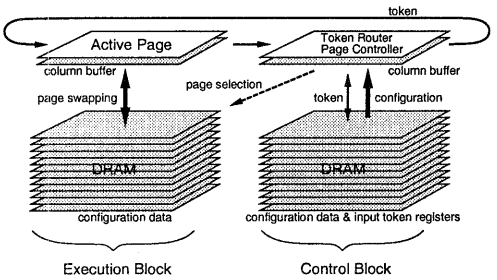

図 2: HOSMIIの基本構成

図3は制御部の構成を示している。制御部では、演算部のサブグラフから出力されたトークンが到着すると、まずルーティングテーブルによってそのトークンをどこに転送すべきかを調べる。一方、他のユニットから来たトークンは一度トークン用のFIFOに格納される。そして、クロスバスイッチ(SW)によって、各トークンはユニット内のトークンレジスタ、または他のユニットへと振り分けられる。もし、スイッチ上で2つのトークン間の出線競合が起きた場合は、演算部から到着したトークンが優先され、一方のトークンはトークンFIFO上で待たされることになる。また、ページ制御回路では常にトークンの到着状況を監視し、演算部に対してアクティベイトすべきページやタイミングについての指示を行う。

3 HOSMIIチップの特徴

図4に試作したHOSMIIチップの概念図を示す。HOSMIIチップは演算処理を行なうCore部と、回路を切替えたり、外部から回路構成データを書き込むように命令を送るController部からなる。次にこの2つの機能ブロックの詳細及び特徴を述べる。

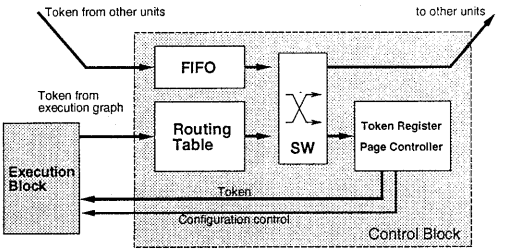

図 3: 制御部の構成

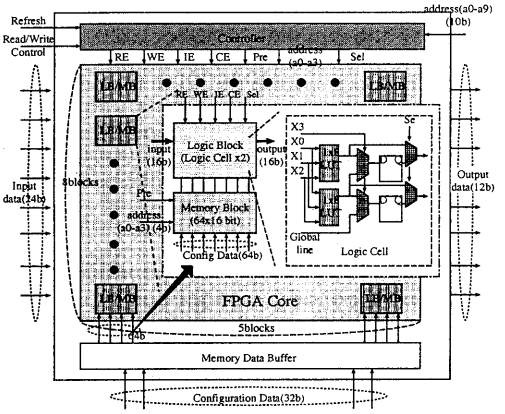

図 4: HOSMIIチップの概念図

3.1 FPGA Core

3.1.1 基本構成

FPGAの心臓部であり、演算処理を行なう部分であるCore部は論理演算を行なう論理ブロックを2次元アレイ状に配置した構成になっている。論理ブロックはさらに、FPGAの基本単位である論理セルより構成される。図5に論理セルの構造を示す。論理セルは、2個の1 × 8 *LUT*(*Look Up Table*)、2個のDFF及びプログラマブルなマルチプレクサからなり、入力は4ビット、出力は2ビットである。

またこの論理セルは、3入力2出力論理及び4入力1出力論理のいずれかを構成するできるようになっている。3入力2出力論理のセルを構成した場合、3ビットの入力($X0$、$X1$、$X2$)がセル内の2個の*LUT*をアドレッシングするために用いられる。一方、4入力1出力論理のセルを構成した場合、2個の1 × 8 *LUT*を1個の1 × 16 *LUT*として扱う事ができる。4ビットの入力のうち、1ビット($X3$)が2つの*LUT*のうちのどちらか一方を選択するために用いられる。またシーケンシャルなデータ処理も行えるようにDFFを持っており、*Se*によってDFFからの出力とLUTからの出力を選択できるようにした。

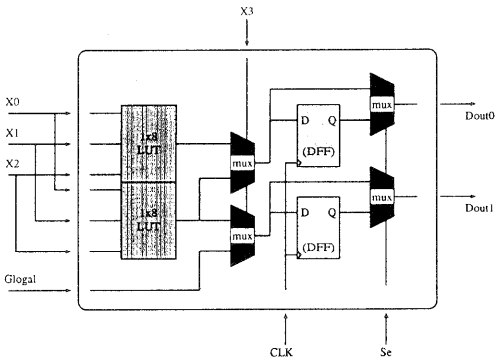

図 5: 論理セル

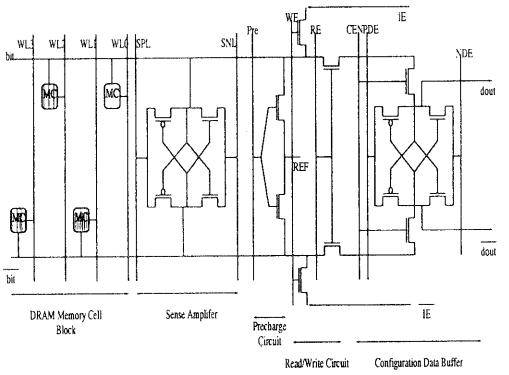

図 6: Memory/Logic インタフェース

論理ブロックをこのようなデザインにすることで、全加算器や乗算器のような演算モジュールを容易に構成することが可能であり、さらにALUなどの粒度の粗いマクロセルを機能ブロックとして用いたに比べ柔軟性を持ったデバイスとなる。

回路構成用DRAMと論理ブロック間のインタフェースを図6に示す。回路構成用DRAMから次の回路構成データを読み出すためにメモリのWord線がactivateされると、メモリセルからの微小電流を感知し、微小電位差を高速にVDDまで増幅するためのSense Amplifierがメモリブロックの上部に配置してある。このSense Amplifierの上部には、回路構成用データの読み出しと、外部から書き込みを行なうRead/Write回路と回路構成用DRAMのビット線対をVDD半分の電圧にプリチャージしておくための周辺回路が配置されている。Read回路の先に回路構成データを格納しておくためのConfiguration SRAMを置く。このSRAMを付加したことで、外部からコンフィギュレーションデータを書き込む要求が来ても、現在実行中の論理演算を中断することなく、コンフィギュレーションを回路構成用DRAMに書き込むことができる。SRAMへの書き込みは、コンフィギュレーションをDRAMセルから読み出す際に同時に行なわれる。

3.1.2 主な特徴

HOSMIIIがデータ駆動型制御機構をターゲットにしているので、論理ブロックから出力されるデータは、ある特定の方向のみに送られるように制限した。図7に各論理ブロック間の結合を示す。始めにFPGAのCore部を、論理ブロック単位に縦方向をステージ、横方向をラインに分けて考える。ある論理ブロックからの出力データは、基本的に次のステージの論理ブロックの真横か、その上下の論理ブロック以外にしか送ることができない。しかし、カウンタなどのようなループバック構造を持つ回路を構成する場合も考えられるので、各ラインごとにフィードバックパスを、さ

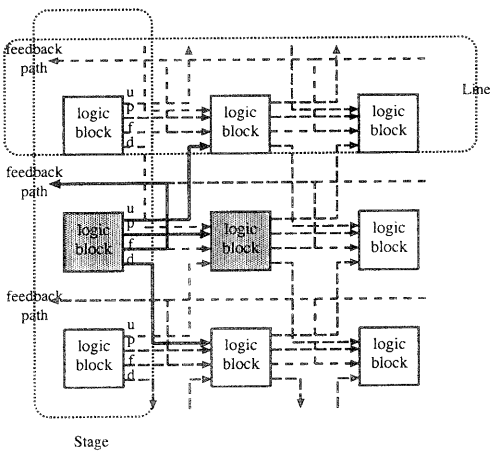

図 7: 論理ブロック間の接続

らに各ステージ及びラインごとにグローバルバスを持たせた。このようにデータの出力方向に制限を持たせたことにより、柔軟性は減るが、配線の複雑さを回避できる。また配線パターンをプログラムするためのコンフィギュレーションデータについても半分近く減らすことができた。

論理セルを構成するためのコンフィギュレーションデータの詳細な割り当てを図8に示す。コンフィギュレーションデータの下位10ビット($memory[9:0]$)は入力信号($X0$、$X1$、$X2$、$X3$)を選択するために用いられる。11ビット目から16ビット分($memory[25:10]$)はルックアップテーブルの内容として利用される。ルックアップテーブルの内容を格納した場所から上位2ビット($memory[27:26]$)は、LUTからの出力にするか、FFからの出力にするかを選択するために用いられ、残りの4ビット($memory[31:28]$)はデータの出力方向を決定するために利用される。論理ブロックで考えると、論理セルが2組あるので、このような割り当てが残りの上位32ビット($memory[63:32]$)に対しても同様に行なわれる。

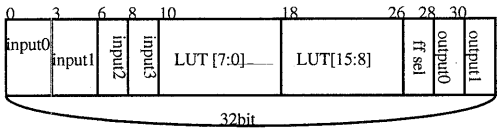

図 8: コンフィギュレーションの割り当て

3.2 FPGA Controller

FPGA Controller部はContextの切り替え時や、外部から回路構成用メモリに配線情報データを書き込む際にFPGA Core部を制御するモジュールである。FPGA Controller部は、Coreを制御すると同時に、回路構成用メモリのアドレスデコーダの機能も兼ね備えている。回路構成用メモリに対してデータを書き込む場合と読み出す場合及びリフレッシュのタイミングを図9に示す。またここに上げられている信号名は図6と対応している。

3.2.1 データ書き込み時の動作

始めにデータを書き込む場合について述べる。*Req*信号がEnableになり、*WR*信号が外部から入力されると、回路構成データがConfiguration Data Bufferに格納される。このConfiguration Data Bufferは、パッケージのピン数に限りがあったため、外部からデータを書き込む場合、32ビットデータを2回に分けて入力し、内部で64ビットデータにして、メモリセルに書き込むようにしている。同時にメモリにデータを書き込むためのアドレスデータもController内のAddress Bufferに格納する。

次にConfiguration Data Bufferに格納されたデータをメモリセルに書き込むためのメモリブロックとそのブロックのラインを選択しなくてはいけない。またその際にビット線対を電源電圧の半分になるようにプリチャージしておかなくてはいけない。ビット線をプリチャージするためには、*PRE*信号をassertする。*PRE*信号がassertされているかどうかは、チップ内部のプリチャージ完了フラグによって検出される。プリチャージが完了すると、フラグが立ち、*PRE*信号がLowになる。

プリチャージ完了フラグが立つとControllerは、データを書き込むためのメモリブロックとブロックのラインを選択するために、10ビットのアドレスをアドレスバッファから出力する。10ビットのアドレスのうち、上位6ビットはブロックを選択する際に利用され、下位4ビットはラインを選択するために用いられる。10 ビットのアドレスを出力すると、Configuration Data Bufferから書き込むデータが出力され、上位6ビットで選択されたメモリブロックの、下位4ビットで指定されたラインにデータが書き込まれる。書き込みの際にはこの動作を各ブロックに対してcontext数分だけ連続して行う。

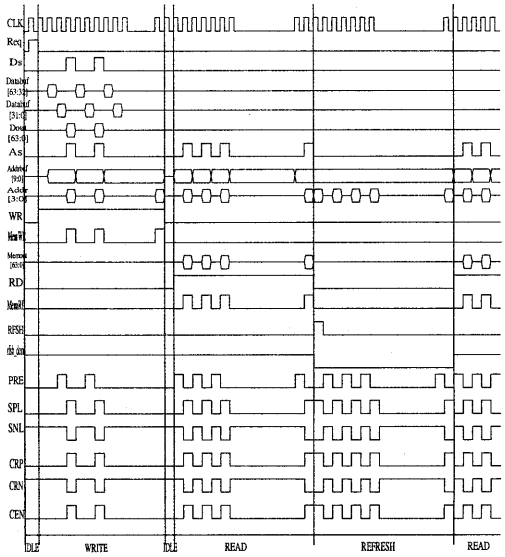

図 9: タイミングチャート

3.2.2 データ読み出し時の動作

読み出し動作は一度に全ブロックのラインを読み出すため、10ビットのアドレスのうち、ラインを選択する下位4ビットのみが必要になる。書き込み動作と同じようにビット線のプリチャージが完了し*PRE*信号がLowになると、10ビットのアドレスのうち下位4ビットを出力することによってControllerは全メモリブロックの、データを読み出すラインを選択する。下位4ビットがワード線選択回路に入力され、Word線が選択されると同時にSense Amplifier と Configuration SRAMをactiveにするために、*SPL*及び*CRP*をHighに、*SNL*及び*CRN*をLowにする。Sense AmplifierとConfiguration SRAMがactiveになると、セルから読み出されたデータがConfiguration SRAMへ書き込まれ、データの読み出しが完了する。

3.2.3 リフレッシュ時の動作

回路構成用メモリがDRAMであることから、リフレッシュ動作が必要である。リフレッシュ動作は外部から*RFSH*信号を入力することで行なう。*RFSH*が外部から入力されると、リフレッシュ完了フラグがリセットされる。リフレッシュ時の動作はリフレッシュアドレスを入力する以外は読み出し動作と同じである。リフレッシュアドレスはController内部のリフレッシュカウンタによって生成される。リフレッシュカウンタの値に対応したWord線が選択され、メモリのリフレッシュ動作に入る。1ラインのリフレッシュが完了すると、リフレッシュカウンタの値を1つインクリメントする。このカウンタの値を続けて入力することで、リフ

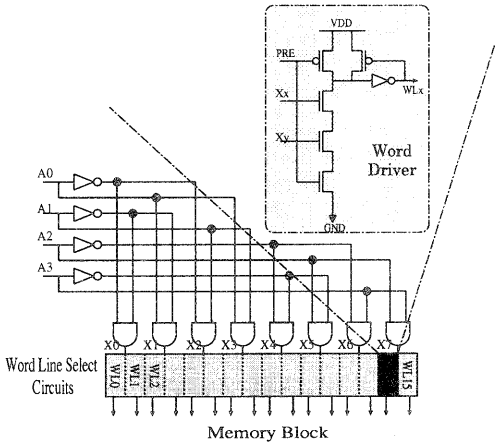

図 10: ワード線選択回路

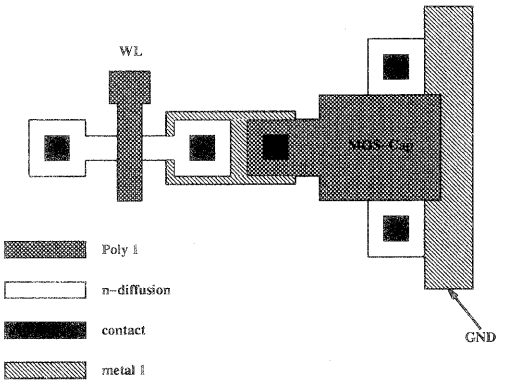

図 11: DRAM セルのレイアウト

レッシュを連続して行うことができる。全 Word 線が選択され、リフレッシュが終了するとリフレッシュ完了フラグが立ち、リフレッシュによって中断された動作モードに戻る。

リフレッシュ動作時においても、回路の書き換えが起こらない場合は、Configuration SRAM に現在実行中のコンフィギュレーションデータが格納されているので、演算をサスペンドすることない。

3.3 回路構成用 DRAM ブロック

次に回路構成用 DRAM ブロックについて説明する。回路構成用 DRAM ブロックは配線データの書き込みと読み出しで、動作が若干異なる。データの書き込みは通常の DRAM と同様にブロックとラインを選択し、外部からデータを書き込む。データの読み出しは、全ブロックに対してラインを選択し、データを読み出す。こうすることで Word 線の切り替えだけで、回路を再構成することができる。

図 10 に今回設計した DRAM ワード線選択回路を示す。ワード線は、ビット線対をプリチャージしている間、すべてオフになっていなくてはならない。そのためワード線選択回路は図 10 に示すような回路を構成した。この回路は PRE 信号が High の時は出力が Low になるように制御される。またワード線選択回路にアドレスを 4 ビットをそのまま入力せずに、上位 2 ビットと下位 2 ビットでプリデコードして入力している。こうすることでデコーダ回路の配線領域を削減することができ、かつ回路も容易になる。

また図 11 に今回用いた DRAM セルのレイアウトを示す。今回使用したのはロジック用のプロセスであったため、DRAM セルとなるキャパシタ用のプロセスが存在しない。そのため、nMOS のゲート容量を使用し、DRAM のメモリセルとした。

4 HOSMII チップの諸元及び性能評価

今回試作した HOSMII チップのレイアウト及び諸元を図 12 及び表 1 に示す。HOSMII チップは電源電圧が $3.3V$ で、$Rohm$ 社の Metal3- Poly2 層、$0.35\mu m$ プロセスを用いて試作を行なった。チップサイズは $4.93mm^2$ であり、全論理ブロック数は $40block$ となっている。チップ内にコンテキストを 16 構成面分格納できる。

アプリケーションを実装する予備評価として、回路構成用メモリ及び論理ブロックの性能をシミュレーション評価した。回路構成用メモリは、データの読みだし及び書き込み時間、ビット線のプリチャージ時間などの測定を行なった。また論理ブロックは、簡単な回路を実際に構成し、動作速度をシミュレーション評価した。

まず回路構成用メモリの Read/Write 時の動作速度を測定した。アドレスが Word 線選択回路に入力されてから、選択された Word 線が On 状態になるのにおよそ $1ns$ かかることがわかった。Word 線が On 状態になってからメモリセルからデータが読み出されるまでにおよそ $1ns$ であり、アドレスが入力されてからメモリセルのデータが出力されるまでに $2ns$ かかることがわかった。読み出されたデータが安定するまでの時間として $2ns$ 程度をマージンにとって考えると、およそ $5ns$ でデータ読み出しを行なえる。したがって回路の再構成時間は $5ns$ となる。ビット線をプリチャージする時間を考慮すると、プリチャージにかかる時間もおよそ $5ns$ なので、合わせて $10ns$ で行なえることが確認できた。これより回路構成用メモリは理論上 $100MHz$ で動作させることが可能であることがわかった。

次に 1 論理ブロックあたりの入力及び出力特性を調べるために、組み合わせ回路として 1 ビットの全加算器を、また順序回路として 1 ビットのカウンタをマッピングした。全加算器を構成した場合、論理ブロックの入力から出力まで

の遅延はおよそ$3ns$程度であることが確認できた。

またカウンタを構成した場合、LUTの入力からフリップフロップまでの遅延時間はおよそ$2ns$であり、フリップフロップの出力から論理ブロックの入力までの遅延時間は$2ns$であった。よって次の段のフリップフロップの入力まではおよそ$4ns$の遅延であることが確認できた。したがって論理ブロックは$100MHz$で十分に動作させることが可能である。

次に今回設計したDRAMとSRAMについてのトレードオフについて検討する。回路構成用DRAMブロックと論理ブロックの間に配置したConfiguration SRAMを配線情報メモリとして用いた場合、セルの面積がDRAMセルの約2倍になることより、チップサイズ固定の場合では、コンフィギュレーションデータはおよそ半分に減少する。またデータ読み出し時の速度についても測定した。DRAMを用いた場合、ビット線のプリチャージ時間等を含めると、データを読み出すためには、$10ns$近くかかる。SRAMを用いた場合、セル容量による負荷がないため、$2ns$程度でデータを読み出すことが可能である。

1回再構成するのに必要な配線情報を、外部メモリから読み込むのにかかるレイテンシを$100nsec$と仮定し、再構成時間を各メモリのリード時間と同じと考えてトレードオフを計算した。またDRAM型を用いた場合、ページ化されたアプリケーションがすべてチップ内に収まるものとする。さらにコンテキスト切り替え時において、過電流防止のために、各ステージの8ブロックずつ再構成するように仮定した。SRAM型のチップ内コンテキスト数をNc、DRAM型のチップ内コンテキスト数をNd、アプリケーションの全ページ数をP、チップ内論理ブロック数をLb、ステージ数およびライン数をSt、LiとしてSRAM型の全ページの再構成時間Ts及びDRAM型の全ページの再構成時間Tdを計算すると以下のようになる。

$$Ts = 100 \times Lb \times (\frac{P}{Nc} - 1) \times Nc + 2 \times P \quad (\text{SRAM型})$$
$$Td = (10 \times St \times \frac{Li}{8}) \times Nd \quad (\text{DRAM型})$$

ここで$Lb = St \times Li$であることより、$Td < Ts$を計算すると以下の式が得られる。

$$80 \times \{(1 + \frac{1}{40 \times Lb}) - Nc\} > Nd \qquad (1)$$

$Nd = P$であった場合、(1)式に代入し計算すると$Nc < 0.9875P$となる。ここでNc、Lb、Pは十分大きな値として近似を行った。この結果から、ページの再利用が困難なアプリケーションやチップ内に全てのページが入りきらなず、外部バックアップメモリにアクセスしなくては行けないような比較的大きなアプリケーションを実装する場合、数回に分けて再構成を行ったとしてもDRAM型を利用した方が有利である。次にDRAM型でどのくらいまでのページ数が保持できればSRAM型を用いるよりも性能がよいか調べた。$Nd = Nc \times 2$として(1)式に代入し計算すると

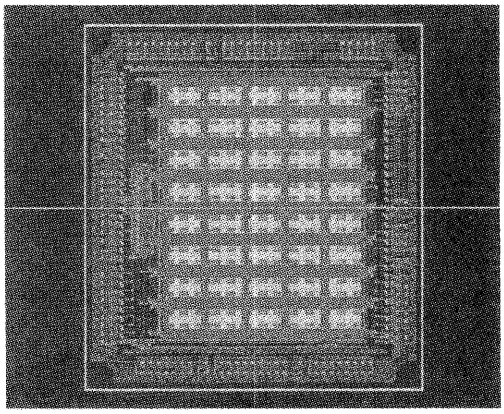

図 12: HOSMIIチップのレイアウト

表 1: HOSMIIチップの諸元

電源電圧	$3.3V$
プロセスルール	$0.35\mu m$ (Metal3-Poly2)
論理ブロック面積	$128(h) \times 479(w)um^2$
メモリブロック面積	$106(h) \times 479(w)um^2$
メモリ/ロジック面積比	0.83 : 1
メモリセル面積(cell only)	$1.95(h) \times 3.40(w)um^2$
メモリセル面積(with sel Tr)	$7.45(h) \times 4.50(w)um^2$
メモリセル数(論理ブロック当り)	64×16 (bits)
メモリセル容量	34.27(fF)
コンテキスト数	16 context
チップサイズ	$4.93mm^2$
論理ブロック数	$8 \times 5blocks$
再構成時間	$5ns$
論理ブロックの動作周波数	$100MHz$

$0.513Nd < P \leq Nd$となる。したがってアプリケーションの全ページ数がDRAM型のコンテキストの半分を上回るものであれば、DRAM型を利用する方が有利である。

また最大実現ゲート数の測定も行なった。1つの論理セルで実現可能なゲート規模を16〜20ゲートとして計算するとおよそ1300〜1600ゲート程度となる。

5 まとめ

最後に試作したHOSMIIチップについて簡単にまとめる。HOSMIIチップは、電源電圧が$3.3V$で、$Rohm$社のMetal3-Poly2層、$0.35\mu m$プロセスを用いて試作を行なった。チップサイズは$4.93mm^2$であり、論理ブロックは$40block$で、チップ内コンテキスト数は16面とした。

FPGAの基本構成単位となる論理セルは3入力2出力論理及び4入力1出力論理を選択することが可能なセルであり、容易に演算器モジュールを構成することができる。また、ターゲットとしているシステムがデータフロー制御を用

いていることより、論理ブロックの演算処理結果はある特定の方向にしか出力できないように制限した。こうすることで、配線パターンの柔軟性は欠けるものの、配線領域を大幅に削減可能であるとともに、配線をプログラムするためのコンフィギュレーションデータも半分近く削減することができた。さらにDRAMのRead回路の先にConfiguration SRAMを付加したことにより、論理ブロックでの演算処理を中断させることなく、外部からコンフィギュレーションを書き込むことが可能である。

チップ性能の予備評価を行なったところ、回路構成用メモリは$5ns$でデータを読み出すことが可能であった。これより回路の再構成を$5ns$で行えることが確認できた。また論理ブロックについても、論理演算を$100MHz$以上で動作させることが可能であると確認できた。

6 課題及び今後の予定

今回設計したHOSMIIチップは2001年1月末にパッケージ化される予定である。実チップが完成した後、DRAMセルのリテンション特性や、実チップ上で簡単なアプリケーションを実装する予定である。さらなる性能向上を考えた場合、回路構成について若干検討しなければいけない。今回は内部DRAMを回路構成用にのみ利用しているが、これをデータメモリとして演算結果を格納できるように拡張することで、さらなる性能向上が望めると考えられる。またチップサイズ及びピン数の制限により本チップで実装できるアプリケーションは、小規模なものに限定されてしまうが、このチップを多数接続したマルチチップ構成にすることで、より大きなアプリケーションにも対応できると考えられる。さらに各チップに並列に演算処理を行なわせることで、より性能向上が期待できる。また部分再構成の機能も採り入れることで、HOSMIIシステムを構成するための制御部を効率良く実装することができる。演算部においても部分再構成可能にすることによる性能向上を期待できる。

現在の大きな問題点として、コンフィギュレーションをマッピングする作業は、すべて手動で行なわなくてはいけない。そのためコンフィギュレーションをFPGAにマッピングするためのソフトウェアの開発を今後行なう必要がある。

アプリケーションの実装として現在考えているものは、パターンマッチングや高速フーリエ変換(FFT)などを検討している。これらのアプリケーションに対する評価結果を従来のFPGAを用いた場合やSRAM型マルチコンテキストFPGAを用いた場合などと比較検討して行く予定である。

参考文献

[1] T. Fujii et. al. "A Dynamically Reconfigurable Logic Engine with a Multi-Context/Multi-Mode Unified-Cell Architecture" Proc. International Solid State Circuit Conference 1999

[2] S.Scalera et. al. "The Design and Implementation of a Context Switching FPGA" Proc. FPGAs for Custom Computing Machines 1998

[3] X.-P. Ling and H. Amano "WASMII: A Data Driven Computer on a Virtual Hardware" Proc. FPGAs for Custom Computing Machines 1993

[4] Y. Shibata, H. Miyazaki, X.-P. Ling and H. Amano "HOSMII: A Virtual Hardware Integrated with DRAM" Proc. Parallel and Distributed Processing (LNCS 1338) 1998

[5] A. DeHon "Dynamically Programmable Gate Arrays: A Step Toward Increased Computational Density" Proc. Canadian Workshop on Field Programmable Devices 1996

[6] M.Motomura et. al. "An Embedded DRAM-FPGA Chip with Instantaneous Logic Reconfiguration" Proc. Symposimu on VLSI Circuits 1997