(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-127460
(P2006-127460A)
(43)公開日　平成18年5月18日(2006.5.18)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 15/80	(2006.01)	G06F 15/80		5B056
G06F 17/16	(2006.01)	G06F 17/16	M	

審査請求 未請求　請求項の数 40　ＯＬ　（全 125 頁）

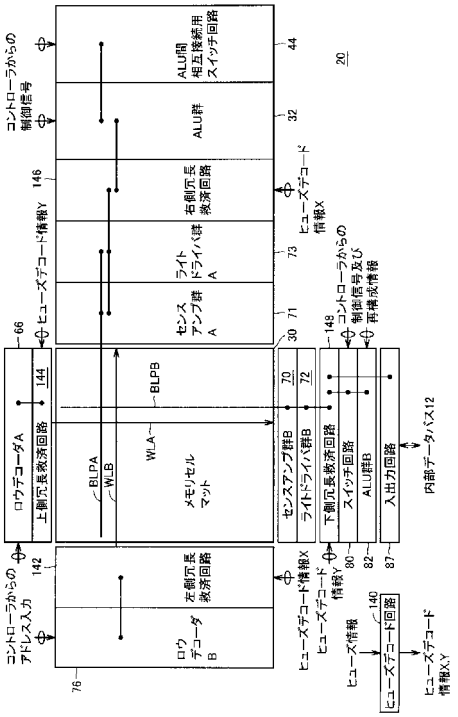
(21)出願番号　特願2005-143109 (P2005-143109)
(22)出願日　平成17年5月16日 (2005.5.16)
(31)優先権主張番号　特願2004-171658 (P2004-171658)
(32)優先日　平成16年6月9日 (2004.6.9)
(33)優先権主張国　日本国 (JP)
(31)優先権主張番号　特願2004-175193 (P2004-175193)
(32)優先日　平成16年6月14日 (2004.6.14)
(33)優先権主張国　日本国 (JP)
(31)優先権主張番号　特願2004-282449 (P2004-282449)
(32)優先日　平成16年9月28日 (2004.9.28)
(33)優先権主張国　日本国 (JP)

(71)出願人　503121103
株式会社ルネサステクノロジ
東京都千代田区丸の内二丁目４番１号
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平
(74)代理人　100096781
弁理士　堀井　豊
(74)代理人　100098316
弁理士　野田　久登
(74)代理人　100109162
弁理士　酒井　將行

最終頁に続く

(54)【発明の名称】半導体装置、半導体信号処理装置、およびクロスバースイッチ

(57)【要約】
【課題】　大量のデータを、その演算内容およびデータビット幅にかかわらず高速で演算処理する。
【解決手段】　メモリセルマット（３０）を複数のエントリ（ＥＲＹ）に分割し、各エントリ（ＥＲＹ）に対応して、演算処理ユニット（ＡＬＵ）を配置し、これらのエントリと対応の演算処理ユニットとの間で、ビットシリアルかつエントリパラレル態様で演算処理を実行する。並列演算性が低い場合には、このメモリセルマット（３０）下部に設けられた演算器群（８２）に対して、エントリシリアルかつビットパラレル態様でデータを転送して演算処理を実行する。
【選択図】　図３０



【特許請求の範囲】
【請求項１】
行列状に配列されかつ複数のエントリに分割される複数のメモリセルを有するメモリセルアレイ、
各前記エントリに対応して配置され、各々が指定された演算を対応のエントリのデータに対して行なう複数の第１の演算回路、
各前記エントリと対応の第１の演算回路との間でデータを転送する複数のデータ転送線、および
前記複数のデータ転送線それぞれに対応して配置され、対応のデータ転送線と対応の第１の演算回路との間でビット単位でかつエントリパラレル態様でデータを転送する複数のデータ転送回路を備え、
各前記エントリには多ビットデータが格納され、各前記第１の演算回路は対応のエントリの多ビットデータに対してビットシリアルな態様で演算を実行する、半導体装置。
【請求項２】
前記複数の第１の演算回路の間でデータを転送する接続切換転送回路をさらに備える、請求項１記載の半導体装置。
【請求項３】
前記複数のエントリは、前記メモリセルアレイの各列に対応して配置されるエントリを備え、
前記データ転送線は、各列に対応して配置される、請求項１記載の半導体装置。
【請求項４】
各前記メモリセルは、書込ポートと読出ポートとを有するマルチポートメモリセルであり、
前記複数のデータ転送線は、対応のエントリのメモリセルの書込ポートに接続される書込データ線と、対応のエントリのメモリセルの読出ポートに接続される読出データ線とを備える、請求項１記載の半導体装置。
【請求項５】
前記メモリセルアレイの各エントリにおいて同一位置のメモリセルの読出ポートを選択状態へ駆動する読出セル選択回路と、
前記読出セル選択回路と別に設けられかつ並行して動作可能であり、前記メモリセルアレイの各エントリの同一位置のメモリセルの書込ポートを選択状態へ駆動する書込セル選択回路とをさらに備える、請求項４記載の半導体装置。
【請求項６】
前記メモリセルアレイは、各前記エントリを分割するように第１のメモリアレイと第２のメモリアレイとに分割され、
前記複数の第１の演算回路は、前記第１および第２のメモリアレイの間に配置され、前記第１および第２のメモリアレイの対応するエントリからデータを並行して受けて該受けたデータに対して演算を実行する、請求項１記載の半導体装置。
【請求項７】
各前記第１の演算回路は、演算を実行する演算部と、対応のエントリから与えられたデータを格納するレジスタ回路と、前記演算部の演算結果を格納する結果レジスタと、前記演算部の演算処理を禁止するマスクデータを格納するマスクレジスタ回路を備える、請求項１記載の半導体装置。
【請求項８】
各前記メモリセルは、互いに独立に選択可能な第１および第２のポートを有するマルチポートメモリセルであり、かつ
前記複数のメモリセルは、前記行および列の一方に対応する第１の方向および前記行および列の他方に対応する第２の方向に整列して配置され、
前記半導体装置は、さらに、
各々が前記第１の方向に沿って整列して配置されるメモリセルに対応して配置され、各

々に対応のメモリセルの第１のポートが接続される複数の第１のワード線と、
各々が前記第２の方向に沿って整列して配置されるメモリセルに対応して配置され、かつ各々が対応のメモリセルの第１のポートに結合されて、かつさらに、前記データ転送線を構成する複数の第１のビット線と、
各々が前記第２の方向に沿って整列して配置されるメモリセルに対応して配置され、かつ各々が対応のメモリセルの第２のポートに接続する複数の第２のワード線と、
各々が前記第１の方向に沿って整列して配置されるメモリセルに対応して配置され、かつ各々が対応のメモリセルの第２のポートに接続する複数の第２のビット線と、
前記複数の第２のビット線と対向してかつデータを転送可能に配置される複数の第２の演算回路とをさらに備える、請求項１記載の半導体装置。

【請求項９】
前記複数の第２のビット線と前記複数の第２の演算回路とのデータ転送経路を変更する経路変更回路をさらに備える、請求項８記載の半導体装置。

【請求項１０】
各前記エントリには、多ビットデータワードが格納され、前記複数の第２の演算回路は、与えられた多ビットデータワードをワード単位で演算する、請求項８記載の半導体装置。

【請求項１１】
前記第２の演算回路は、複数段にわたって配置される演算器を備える、請求項８記載の半導体装置。

【請求項１２】
前記複数の第２のビット線と装置外部との間でデータをエントリ単位で転送する入出力回路をさらに備える、請求項８記載の半導体装置。

【請求項１３】
前記第１のワード線に対応して配置され、第１の不良アドレス情報に従って前記複数の第１のワード線とアドレスとの対応をシフトして切り替えて不良アドレスを救済する第１のシフト冗長救済回路、
前記第２のワード線に対応して配置され、第２の不良アドレス情報に従って前記複数の第２のワード線とアドレスとの対応をシフトして切り換えて不良アドレスを救済する第２のシフト冗長救済回路、
前記複数の第２のビット線に対して設けられ、前記第１の不良アドレス情報に従って、前記複数の第２のビット線と前記複数の第２の演算回路との対応をシフトして切り替える第３のシフト冗長救済回路、および
前記複数の第１のビット線に対して設けられ、前記第２の不良アドレス情報に従って前記複数の第１のビット線と前記複数の第１の演算回路との対応をシフトして切り替える第４のシフト冗長救済回路をさらに備える、請求項８記載の半導体装置。

【請求項１４】
前記複数の第２の演算回路に対応して配置され、前記複数の第２の演算回路間でデータを転送する、転送経路が変更可能なデータ転送回路をさらに備える、請求項８記載の半導体装置。

【請求項１５】
前記メモリセルアレイのメモリセル列それぞれに対応して配置され、各々に対応の列のメモリセルが接続し、かつ前記複数のデータ転送線を構成する複数のビット線、および
前記複数の第１の演算回路に対応して配置され、前記データ転送線と前記第１の演算回路との対応をシフトして変更して不良演算回路を救済する冗長回路をさらに備える、請求項１記載の半導体装置。

【請求項１６】
前記メモリセルアレイの各エントリに対して共通に配置され、各エントリにおいて有意のデータが格納される領域を指定するポインタ回路をさらに備え、前記ポインタ回路は、各前記エントリに複数の多ビットデータが格納されるとき、各多ビットデータの格納領域

を指定する、請求項１記載の半導体装置。
【請求項１７】
行列状に配列されるメモリセルを有し、各々が複数ビットを有する複数のエントリに分割されるメモリセルアレイ、および
各行に対応して配置される複数の演算回路を備え、前記演算回路は、演算器と、少なくとも第1および第2のレジスタと、1つのマスクレジスタとを含み、さらに、
前記第1のレジスタに第1の演算ビットを格納し、前記演算器に前記メモリアレイからのデータビットと前記第1の演算ビットの演算を行わせ、該演算結果を前記第1のレジスタに格納して、該第1のレジスタの格納値を前記メモリセルアレイの対応の位置に書き込む制御回路を備え
、前記制御回路は前記演算器の演算内容を設定する、半導体信号処理装置。
【請求項１８】
前記第２のレジスタは、加減算演算時のキャリを格納し、
前記制御回路は、演算最終時に前記第２のレジスタの格納データを対応のエントリの対応のビット位置に格納する、請求項１７記載の半導体信号処理装置。
【請求項１９】
乗算時には、前記マスクレジスタに被乗数ビットを格納し、
前記マスクレジスタの格納値にしたがって、前記対応のエントリの乗数ビットと乗算結果ビットとの加算を選択的に行い、該演算結果を前記第１のレジスタに格納して対応の乗算結果格納領域の元の位置に書込み、前記乗数のビット全てについて選択的加算を行った後、前記被乗数ビットの対応のエントリのビット位置を指定するビット位置アドレスを増分してかつ前記乗算結果ビットの位置のアドレスを１増分して、乗数について同様の動作を繰り返すように前記制御回路が動作する、請求項１７記載の半導体信号処理装置。
【請求項２０】
行列状に配列される複数のメモリセルを有し、かつ複数のエントリに分割されるメモリセルマット、および
各エントリに対応して配置される複数の演算回路を備え、
各前記演算回路は、
対応のエントリの第1の領域からのデータビットの組について2次のブースアルゴリズムに従ってデコードした結果を格納するブースレジスタ回路と、
前記対応のエントリの第2および第3の領域の対応にビット位置からのデータビットを受け、前記ブースレジスタ回路の格納データにしたがって受けたデータに対して演算処理を行う演算器と、
前記演算器の出力データを格納する結果レジスタとを備え、さらに、
前記メモリセルマットの各エントリから前記第1、第2および第3の領域のデータを対応の演算回路に転送し、かつ前記演算器の出力データを対応のエントリの第3の領域に転送して書き込み、かつさらに、前記演算回路の演算動作を制御する制御回路を備える、半導体信号処理装置。
【請求項２１】
前記演算器は、前記ブースレジスタ回路の格納値に従って、前記第２および第３の領域からのデータの組について、前記第２の領域からのデータの１ビットまたは２ビットシフトおよび反転または正転および無演算を選択的に行い、該演算結果と前記第３の領域からのデータとの加算を行う、請求項２０記載の半導体信号処理装置。
【請求項２２】
各前記エントリは、偶数ビットを格納する偶数エントリと奇数ビットを格納する奇数エントリとを含み、
各前記演算器は、対応のエントリの偶数エントリおよび奇数エントリの対応のビット位置のデータを並列に受けて処理を実行する、請求項２０記載の半導体信号処理装置。
【請求項２３】
行列状に配列される複数のメモリセルを有し、各々が同一ビット幅を有する複数のエン

トリに分割されるメモリセルマット、
各エントリに対応して配置され、各々が対応のエントリのデータに対して指定された処理を実行する複数の演算回路を含む演算ブロック、
前記演算ブロックの所定数の演算回路に対応して設けられ、対応の演算回路に対して演算内容を指定する制御信号を伝達する演算制御信号線を備える、半導体信号処理装置。
【請求項２４】
行列状に配列される複数のメモリセルを有し、かつ各々がワード単位のデータを格納するメモリセルを含む複数のエントリに分割されるメモリセルマットと、各エントリに対応して配置される演算器を含み、互いに演算処理を個々に実行することのできる複数の基本演算ブロック、
各基本演算ブロックに共通に配置される内部データバス、
前記内部データバスに結合され、行列状に配列されるメモリセル含みかつ各基本演算ブロックとデータ転送を行うことのできる大容量メモリ、および
前記大容量メモリと選択された基本演算ブロックとの間で、前記大容量メモリの１行のデータ単位でデータ転送を行う制御回路とを備える、半導体信号処理装置。
【請求項２５】
前記大容量メモリは、互いに独立にアクセスすることのできる複数のバンクを備え、各前記バンクは、前記メモリセルマットと同一構成を有する、請求項２４記載の半導体信号処理装置。
【請求項２６】
各々が、行列状に配列される複数のメモリセルを有し、かつ複数のエントリに分割されるメモリセルマットと、各エントリに対応して配置される演算回路とを備える複数の演算ブロック、
隣接演算ブロック間の対応の位置の演算回路を選択的に結合する隣接ブロック接続バス、および
各演算ブロックにおいて、演算回路を選択的に結合するビット転送回路を備える、半導体信号処理装置。
【請求項２７】
前記ビット転送回路は、
隣接演算回路に対応して配置される第１のバス線と、
所定数はなれた位置の演算回路に対して配置される第２のバス線と、
前記第１および第２のバス線に対して配置され、導通時対応のバス線と演算回路とを接続するスイッチ回路を備える、請求項２６記載の半導体信号処理装置。
【請求項２８】
前記スイッチ回路は、接続態様がスルー状態、対応のバス線と対応の演算回路とのデータ転送可能状態、および対応のバス線のデータ転送を禁止するダミー状態のいずれかに切り換えられるプログラマブルスイッチ回路を備える、請求項２７記載の半導体信号処理装置。
【請求項２９】
前記複数のスイッチ回路の接続態様を各前記演算ブロック個々に設定する接続制御回路をさらに備える、請求項２７記載の半導体信号処理装置。
【請求項３０】
各々が、行列状に配列される複数のメモリセルを有しかつ複数のエントリに分割されるメモリセルマットと、各メモリセルマットのエントリに対応して配置される複数の演算回路とを含む複数の演算回路ブロックと、
前記複数の演算回路ブロックに共通に配置されるグローバルデータバス、
外部処理装置に結合されるシステムデータバス、
前記システムデータバスと第１の内部転送バスとの間に接続され、前記システムバスと前記第１の内部転送バスそれぞれに転送されるデータの構成を変更する直交変換回路、
前記第１の内部転送バスと第２の内部転送バスとの間に接続され、前記第１および第２

の内部転送バスの接続経路を変更するクロスバースイッチ、および
前記第２の内部転送バスと前記グローバルデータバスとの間に接続され、前記第２の内部転送バスと前記グローバルデータバスのバス線を選択的に接続する選択回路を備える、半導体信号処理装置。
【請求項３１】
前記クロスバースイッチは、
演算回路ブロックからの接続情報をデコードするデコード回路と、
前記デコード回路の出力信号に従って前記第１および第２の内部転送バスの接続経路を確立するスイッチマトリクスを備える、請求項３０記載の半導体信号処理装置。
【請求項３２】
１列に配置された複数の第１機能ブロックと、それぞれ前記複数の第１機能ブロックに対向して設けられた複数の第２機能ブロックとの間に設けられ、前記複数の第１機能ブロックと前記複数の第２機能ブロックとを１対１で任意の組合せで接続するクロスバースイッチであって、
各第１機能ブロックに対応して設けられて対応の第１機能ブロックのデータ信号端子に接続され、前記複数の第１機能ブロックの配列方向と同じ方向に延在する第１データ信号線、および
各第１データ信号線に対応して設けられ、対応の第１機能ブロックからのセレクト信号に従って前記複数の第２機能ブロックのうちのいずれかの第２機能ブロックを選択し、選択した第２機能ブロックのデータ信号端子と対応の第１データ信号線とを接続する選択回路を備える、クロスバースイッチ。
【請求項３３】
各第１機能ブロックに対応して設けられ、前記複数の第１機能ブロックの配列方向と直交する方向に延在する第２データ信号線を備え、
前記第１データ信号線は、対応の第２データ信号線を介して対応の第１機能ブロックのデータ信号端子に接続されている、請求項３２に記載のクロスバースイッチ。
【請求項３４】
前記選択回路は、
それぞれ前記複数の第２機能ブロックに対応して設けられ、各々が対応の第２機能ブロックのデータ信号端子と対応の第１データ信号線との間に接続された複数のスイッチング素子、および
それぞれ前記複数のスイッチング素子に対応して設けられ、各々が予め割り当てられたセレクト信号が与えられたことに応じて対応のスイッチング素子を導通させる複数のデコード回路を含む、請求項３２に記載のクロスバースイッチ。
【請求項３５】
前記選択回路は、
対応の第１機能ブロックのセレクト信号端子と前記複数のデコード回路のうちのあるデコード回路との間に接続され、前記複数の第１機能ブロックの配列方向と直交する方向に延在する第１セレクト信号線、および
前記第１セレクト信号線に接続されて前記複数の第１機能ブロックの配列方向と同じ方向に延在し、前記セレクト信号を他のデコード回路に与える第２セレクト信号線を含む、請求項３４に記載のクロスバースイッチ。
【請求項３６】
前記選択回路は、各デコード回路に対応して設けられ、対応のデコード回路の出力信号をラッチするラッチ回路を含む、請求３４に記載のクロスバースイッチ。
【請求項３７】
前記セレクト信号は、グローバルセレクト信号とローカルセレクト信号を含み、
前記選択回路は、
それぞれ前記複数の第２機能ブロックに対応して設けられて予め複数のグループに分割され、各々が対応の第２機能ブロックのデータ信号端子と対応の第１データ信号線との間

に接続された複数のスイッチング素子、
それぞれ前記複数のグループに対応して設けられ、各々が予め割り当てられたグローバルセレクト信号が与えられたことに応じて活性化信号を出力する複数のグローバルデコード回路、および
各グループに対応して設けられて対応のグループの複数のスイッチング素子にそれぞれ対応して設けられ、各々が、対応のグローバルデコード回路から活性化信号が出力され、かつ予め割り当てられたローカルセレクト信号が与えられたことに応じて対応のスイッチング素子を導通させる複数のローカルデコード回路を含む、請求項３２に記載のクロスバースイッチ。
【請求項３８】
前記選択回路は、さらに、対応の第１機能ブロックに対向する第２機能ブロックからのセレクト信号に従って前記複数の第２機能ブロックのうちのいずれかの第２機能ブロックを選択し、選択した第２機能ブロックのデータ信号端子と対応の第１データ信号線とを接続する、請求項３２に記載のクロスバースイッチ。
【請求項３９】
前記複数の第２機能ブロックのうちの不良な第２機能ブロックと置換するための冗長第２機能ブロックが設けられ、
前記選択回路は、対応の第１機能ブロックからのセレクト信号によって前記不良な第２機能ブロックが指定された場合は、前記不良な第２機能ブロックの代わりに前記冗長第２機能ブロックを選択し、選択した冗長第２機能ブロックのデータ信号端子と対応の第１データ信号線とを接続する、請求項３２に記載のクロスバースイッチ。
【請求項４０】
前記複数の第１機能ブロックおよび前記複数の第２機能ブロックは複数組設けられ、
前記クロスバースイッチは、各組毎に前記複数の第１機能ブロックと前記複数の第２機能ブロックとを１対１で任意の組合せで接続し、
前記第１データ信号線および前記選択回路は各組毎に設けられ、
電源電圧は、各組毎に供給／遮断することが可能になっている、請求項３２に記載のクロスバースイッチ。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、半導体装置に関し、特に、高速に大量のデータの演算処理を行なう半導体メモリを用いた演算回路の構成に関する。
【背景技術】
【０００２】
近年、携帯端末機器の普及に伴い、音声および画像のような大量のデータを高速に処理するデジタル信号処理の重要性が高くなってきている。このデジタル信号処理には、一般に、専用の半導体装置として、ＤＳＰ（デジタル・シグナル・プロセサ）が用いられる。音声および画像に対するデジタル信号処理においては、フィルタ処理などのデータ処理が行なわれ、このような処理においては、積和演算を繰返す演算処理が多い。したがって、一般に、ＤＳＰの構成においては、乗算回路、加算回路および累算用のレジスタが設けられる。このような専用のＤＳＰを用いると、積和演算を１マシンサイクルで実行することが可能となり、高速演算処理が可能となる。
【０００３】
このような積和演算を行なう際に、レジスタファイルを利用する構成が、特許文献１（特開平６－３２４８６２号公報）に示されている。この特許文献１においては、レジスタファイルに格納された２項のオペランドデータを読出して、演算器で加算した後、再び書込データレジスタを介してレジスタファイルに書込む。この特許文献１に示される構成では、レジスタファイルに対して、書込アドレスおよび読出アドレスを同時に与えて、データの書込およびデータの読出を同時に行なうことにより、データの書込サイクルおよびデ

ータの読出サイクルを別々に設けて演算処理する構成に比べて、処理時間を短縮することを図る。

【０００４】

また、大量のデータを、高速で処理することを意図する構成が、特許文献２（特開平５－１９７５５０号公報）に示されている。この特許文献２に示される構成においては、複数の演算装置を並列に配置し、それぞれの演算装置にメモリを内蔵する。各演算装置において個々にメモリアドレスを生成することにより、並列演算を高速で行なうことを図る。

【０００５】

また、画像データのＤＣＴ変換（離散コサイン変換）などの処理を高速に行なうことを目的とする信号処理装置が、特許文献３（特開平１０－７４１４１号公報）に示されている。この特許文献３に示される構成においては、画像データがビットパラレルかつワードシリアルなシーケンスで、すなわちワード（画素データ）単位で入力されるため、直列／並列変換回路を用いてワードパラレルかつビットシリアルなデータに変換してメモリアレイに書込む。メモリアレイに対応して配置される演算器（ＡＬＵ）へデータを転送して並列処理を実行する。メモリアレイは、画像データブロックに応じてブロックに分割されており、各ブロックにおいては、対応の画像ブロックを構成する画素データが、行ごとにワード単位で格納される。

【０００６】

この特許文献３に示される構成においては、メモリブロックと対応の演算器との間でワード（１つの画素に対応するデータ）単位でデータを転送する。各ブロック個々に、対応の演算器において同一処理を転送されたワードに対して実行することにより、ＤＣＴ変換などのフィルタ処理を高速で実行することを図る。演算処理結果は、再びメモリアレイに書込み、再度並列／直列変換を行なってビットシリアルかつワードパラレルデータをビットパラレルかつワードシリアルなデータに変換して１ラインごとのデータを順次出力する。通常の処理においては、データのビット位置の変換は行なわず、演算器において通常の演算処理を、複数のデータに対して並列に実行する。

【０００７】

また、複数の異なる演算処理を並行して実行することを目的とするデータ処理装置が特許文献４（特開２００３－１１４７９７号公報）に示されている。この特許文献４においては、各々その機能が限定された複数の論理モジュールを、マルチポート構成のデータメモリに接続する。これらの論理モジュールとマルチポートデータメモリとの接続においては、論理モジュールが接続されるマルチポートメモリのポートおよびメモリが制限されており、従って、各論理モジュールが、マルチポートデータメモリへアクセスしてデータの読出および書込を行なうことのできるアドレス領域は制限される。各論理モジュールで演算を行なった結果を、アクセスが許可されたデータメモリに書き込み、これらのマルチポートデータメモリを介してデータを順次論理モジュールを介して転送することにより、パイプライン的に、データを処理することを図る。

【０００８】

また、複数の演算回路の接続を切り換える構成として、特許文献５（特開平１０－２５４８４３号公報）にクロスバースイッチが示されている。

【特許文献１】特開平６－３２４８６２号公報
【特許文献２】特開平５－１９７５５０号公報
【特許文献３】特開平１０－７４１４１号公報
【特許文献４】特開２００３－１１４７９７号公報
【特許文献５】特開平１０－２５４８４３号公報

【発明の開示】

【発明が解決しようとする課題】

【０００９】

処理対象のデータ量が非常に多い場合には、専用のＤＳＰを用いても、性能を飛躍的に向上させることは困難である。たとえば、演算対象のデータが１万組ある場合、１つ１つ

のデータに対する演算を１マシンサイクルで実行することができたとしても、最低でも１万サイクルが演算に必要となる。したがって、特許文献１に示されるような、レジスタファイルを用いて積和演算を行なうような構成の場合、１つ１つの処理は高速であるものの、データ処理が直列に行なわれるため、データ量が多くなるとそれに比例して処理時間が長くなり、高速処理を実現することができない。

【００１０】

また、このような専用のＤＳＰを利用する場合、処理性能は動作周波数に大きく依存することになるため、高速処理を優先した場合、消費電力が増大することになる。

【００１１】

また、この特許文献１に示されるようなレジスタファイルおよび演算器を利用する場合、ある用途に特化して設計されることが多く、演算ビット幅および演算回路の構成等が固定されることになり、他の用途に転用する場合には、そのビット幅および演算回路の構成等を設計し直す必要があり、複数の演算処理用途に、柔軟に対応することができなくなるという問題が生じる。

【００１２】

また、特許文献２に示される構成においては、演算装置個々にメモリが内蔵されており、各演算装置において異なるメモリアドレス領域をアクセスして処理を行なう。しかしながら、データメモリと演算装置とは、別々の領域に配置されており、論理モジュール内において演算装置とメモリとの間でアドレスを転送およびデータアクセスを行なう必要があり、データ転送に時間を要し、このため、マシンサイクルを短縮することができなくなり、高速処理を行なうことができなくなるという問題が生じる。

【００１３】

また、特許文献３に示される構成においては、画像データのＤＣＴ変換などの処理を高速化することを図っており、画面１ラインの画素データを１行のメモリセルに格納して、行方向に整列する画像ブロックに対して並列に処理を実行している。したがって、画像の高精細化のために１ラインの画素数が増大した場合、メモリアレイの構成が膨大なものとなる。たとえば、１画素のデータが８ビットで、１ラインの画素数が５１２個の場合、メモリアレイの１行においては、メモリセルの数が、８・５１２＝４Ｋビットとなり、１行のメモリセルが接続される行選択線（ワード線）の負荷が大きくなり、高速でメモリセルを選択して、データを演算部とメモリセルの間に転送することができなくなり、応じて高速処理を実現することができなくなるという問題が生じる。

【００１４】

また、この特許文献３においては、メモリセルアレイを、演算回路群両側に配置する構成は示されているものの、具体的なメモリセルアレイ構造は示されておらず、また、演算器において演算器をアレイ状に配置することは示されているものの、どのように演算器群を配置するかの詳細については何ら示されていない。

【００１５】

また、特許文献４に示される構成においては、複数のマルチポートデータメモリと、これらのマルチポートデータメモリに対してアクセス領域が制限される複数の低機能の演算器（ＡＬＵ）とが設けられている。しかしながら、この演算器（ＡＬＵ）とメモリとは別の領域に配置されており、配線容量などにより、高速でデータを転送することができず、パイプライン処理を実行しても、このパイプラインのマシンサイクルを短縮することができなくなるという問題が生じる。

【００１６】

また、これらの特許文献１から４においては、演算処理対象のデータの語構成が異なる場合、どのように対応するかについては何ら検討していない。

【００１７】

また、演算器が多数配置され、これらの演算器群においてデータの転送を行って並列演算処理を行う構成においては、データの転送経路を切り替えることにおり処理内容の変更に柔軟に対処することができる。このようなデータ転送経路の切り替えとしては、通信分

野の回線切り替えおよび並列計算機において特許文献５に示されるようにクロスバースイッチが用いられる。

【００１８】

この特許文献５に示されるクロスバースイッチの構成においては、機能ブロックの接続可能な経路にスイッチを配置し、経路指定情報に従ってスイッチを選択的に導通状態としてデータ転送経路を設定する。しかしながら、この様なスイッチマトリクスを利用する場合、接続対象の演算器（機能ブロック）の数が増大すると応じて接続可能経路も増大し、スイッチ回路のレイアウト面積が増大し、また、スイッチ制御信号線の配置が錯綜する。

【００１９】

それゆえに、この発明の目的は、高速で大量のデータを処理することのできる半導体装置を提供することである。

【００２０】

この発明の他の目的は、データの語構成および演算内容にかかわらず、高速で演算処理を実行することのできる半導体装置を提供することである。

【００２１】

この発明のさらに他の目的は、柔軟に処理内容を変更することのできる演算機能内蔵半導体装置を提供することである。

【００２２】

それゆえに、この発明さらに他の目的は、小占有積で演算器群間の接続経路を設定することのできるクロスバースイッチ回路を提供することである。

【課題を解決するための手段】

【００２３】

この発明に係る半導体装置は、行列状に配列されかつ複数のエントリに分割される複数のメモリセルを有するメモリセルアレイと、各エントリに対応して配置され、各々が指定された演算を対応のエントリのデータに対して行なう複数の第１の演算回路と、各エントリと対応の第１の演算回路との間でデータを転送する複数のデータ転送線と、これらのデータ転送線それぞれに対応して配置され、対応のデータ転送線との間でビット単位でかつエントリパラレルの態様でデータを転送する複数のデータ転送回路とを含む。

【００２４】

各エントリにおいては多ビットデータが格納され、各第１の演算回路は、対応のエントリの多ビットデータに対して、ビットシリアル態様で演算を実行する。

【００２５】

この発明に係る半導体信号処理装置は、複数のエントリに分割されるメモリセルアレイと、各エントリに対応して配置される複数の演算回路と、演算回路の演算を制御する制御回路とを備える。演算回路は、演算器と、第１および第２のレジスタと、マスクレジスタとを含む。制御回路は、第1のレジスタに対応のエントリからの演算ビットを格納し、この第1のレジスタの演算ビットとメモリセルアレイからの第２の演算ビットとの演算を実行し、該演算結果を第1のレジスタに格納して、この第1のレジスタの格納値をメモリセルアレイの対応の位置に格納するように制御する。

【００２６】

この発明の第２の観点に係る半導体信号処理装置は、複数のエントリに分割されるメモリセルマットと、エントリに対応して配置される複数の演算回路とを備える。この演算回路は、対応のエントリの第１の領域からのデータビットの組について２次のブースアルゴリズムに従ってデコードした結果を格納するブースレジスタ回路と、対応のエントリの第２および第３の領域の対応の位置からのデータビットを受け、ブースレジスタ回路の格納データに従って受けたデータに対して演算処理を行う演算器と、この演算器の出力データを格納する結果レジスタとを備える。

【００２７】

この第２の観点に係る半導体信号処理装置は、さらに、メモリセルマットの各エントリ

から第１、第２および第３の領域からのデータを対応の演算回路に転送しかつ演算器の出力データを対応のエントリの第３の領域に転送して書込み、かつ演算器の演算処理を制御する制御回路を備える。

【００２８】

この発明の第３の観点に係る半導体信号処理装置は、複数のエントリに分割されるメモリセルマットと、エントリに対応して配置される複数の演算回路と、これらの複数の演算回路の所定数の演算回路に対応して設けられ、対応の演算回路に動作制御信号を伝達する演算制御信号線を備える。

【００２９】

この発明の第４の観点に係る半導体信号処理装置は、各々が、複数のエントリに分解されるメモリセルマットと、エントリに対応して配置される複数の演算回路とを含み、個々に演算処理を実行することが可能な複数の基本演算ブロックと、これらの複数の基本演算ブロックに共通に配置される内部データバスと、この内部データバスに結合される大容量のメモリと、大容量メモリと、選択された基本演算ブロックとの間で大容量メモリの１行のデータ単位でデータ転送を行う制御回路とを備える。

【００３０】

この発明の第５の観点に係る半導体信号処理装置は、各々が、複数のエントリに分割されるメモリセルマットと、各エントリに対応して配置される演算回路とを備える複数の演算ブロックと、隣接演算ブロックの対応のエントリを相互接続する隣接ブロック接続バスと、演算ブロック内の演算器を相互接続するビット転送回路とを備える。

【００３１】

この発明の第６の観点に係る半導体信号処理装置は、各々が、複数のエントリに分割されるメモリセルマットと、エントリに対応して配置される演算器とを含む複数の演算回路ブロックと、これらの複数の演算回路ブロックに共通に配置されるグローバルデータバスと、外部処理装置に接続されるシステムバスと、このシステムバスと第１の内部伝送バスとの間に配置され、これらのバスに転送されるデータの構成を変更する直交変換回路と、第１の内部転送バスと第２の内部転送バスとの間に配置され、これらの第１および第２の内部転送バスの接続経路を変更するクロスバースイッチと、第２の内部転送バスとグローバルデータバスとの間に接続され、これらのバス線を選択的に接続する選択回路とを備える。

【００３２】

この発明に係るクロスバースイッチは、一列に配列される複数の第１機能ブロックと、複数の第１機能ブロックに対向して配置される複数の第２機能ブロックとの間に設けられ、第１および第２の機能ブロックを１対１で任意の組合せで接続するクロスバースイッチであって、各第１機能ブロックに対応して設けられ手対応の第１機能ブロックのデータ信号端子に接続され、複数の第１機能ブロックの配列方向と同じ方向に延在する第１データ信号線と、各第１データ信号線に対応して設けられ、対応の第１機能ブロックからのセレクト信号に従って複数の第２機能ブロックのうちのいずれかの第１機能ブロックを選択し、選択した第２機能ブロックのデータ信号端子と対応の第１データ信号線とを接続する選択回路とを備える。

【発明の効果】

【００３３】

メモリセルアレイを複数のエントリに分割し、各エントリに対して第１の演算回路を配置しており、複数のエントリのデータに対する演算を並列に行なうことができ、高速処理が実現される。

【００３４】

また、第１の演算回路とデータ転送線との間のデータ転送をビット単位で実行し、第１の演算回路においてビットシリアル態様で演算を実行することにより、演算対象のデータの語構成にかかわらず、対応のエントリの多ビットデータに対して指定された演算処理を実行することができる。すなわち、各エントリに有意データワードを格納し、ビットシリ

アル態様で各第１の演算回路で演算処理を行なう構成とすれば、データの語構成（ビット幅）の変更に対しても、大幅なハードウェアの変更を行なうことなく対応して演算処理を行なうことができ、種々のアプリケーションに対して柔軟に対応することができる。

【００３５】

演算回路内に演算器とレジスタとを配置することにより、演算対象データをレジスタに格納してビットシリアル態様で種々の演算処理を実行することができる。

【００３６】

また、演算回路内に複数のレジスタ回路を配置することにより、ビットシリアル態様で乗算を行う場合においても２次のブースアルゴリズムに従って乗算を行うことが可能となる。

【００３７】

また、演算ブロックに対して所定数ごとの演算ブロックに共通に制御信号を伝達することにより、所定数の演算ブロック単位で必要とされる演算を実行することができ、個々の演算ブロックを個別に制御する構成に比べて演算制御が容易となり、容易に単一命令で複数のデータの処理を実行することができる。

【００３８】

また、複数の演算ブロックに共通に大容量メモリを設けることにより、大容量メモリと演算ブロックとの間のデータ転送のバンド幅を大きくすることができ、個々の演算ブロックにおける演算処理に対してデータ転送がボトルネックとなるのを防止することができる。

【００３９】

また、演算ブロック間でおよび演算器間でデータを転送することができるように配置することにより、隣接画素間の演算処理などを容易に実行することができる。

【００４０】

また、入出力インターフェイス部分にデータ変換回路を配置することにより、容易にワードシリアルかつビットパラレルのデータ列とビットシリアルかつワードパラレルのデータ列の変換を行うことができ、演算器内においてビットシリアルにデータ処理を行い外部ではワード単位で処理を行うことができる。

【００４１】

この発明に係るクロスバースイッチでは、各第１機能ブロックに対応して第１データ信号線を設け、その第１データ信号線をセレクト信号によって指定された第２機能ブロックに接続する。したがって、構成の簡単化を図ることができ、レイアウト面積が小さくて済む。

【発明を実施するための最良の形態】

【００４２】

［実施の形態１］

図１は、この発明の実施の形態１に従う半導体演算装置を利用する処理システムの構成を概略的に示す図である。図１において、処理システムは、並列演算を実行する半導体演算装置１と、この半導体演算装置１における処理の制御、システム全体の制御およびデータ処理を行なうホストＣＰＵ２と、このシステムの主記憶として利用されて必要な種々のデータを格納するメモリ３と、メモリ３に対し、直接ホストＣＰＵ２を介することなくアクセスするＤＭＡ（ダイレクト・メモリ・アクセス）回路４とを含む。このＤＭＡ回路４の制御により、メモリ３と半導体演算装置１の間でデータ転送を行なうことができ、また、半導体演算装置１へ直接アクセスすることとができる。

【００４３】

ホストＣＰＵ２、メモリ３、ＤＭＡ回路４、および半導体演算装置１は、システムバス５を介して相互接続される。半導体演算装置１は、複数の並列に設けられる基本演算ブロックＦＢ１－ＦＢｎと、システムバス５とデータ／命令を転送する入出力回路１０と、この半導体演算装置１内部での動作処理を制御する集中制御ユニット１５を含む。

【００４４】

基本演算ブロックＦＢ１－ＦＢｎおよび入出力回路１０は、内部データバス１２に結合され、また集中制御ユニット１５、入出力回路１０および基本演算ブロックＦＢ１－ＦＢｎは、内部バス１４に結合される。基本演算ブロックＦＢ（ＦＢ１－ＦＢｎを総称的に示す）の間には、ブロック間データバス１６が設けられる（図１においては、基本演算ブロックＦＢ１およびＦＢ２の間に配置される隣接ブロック間データバス１６を代表的に示す。

【００４５】

基本演算ブロックＦＢ１－ＦＢｎを並列に設けて、半導体演算装置１内部で並列に同一または異なる演算処理を実行する。これらの基本演算ブロックＦＢ１－ＦＢｎは、同一構成を有するため、図１においては基本演算ブロックＦＢ１の構成を代表的に示す。

【００４６】

基本演算ブロックＦＢ１は、メモリセルアレイおよび演算器を含む主演算回路２０と、マイクロコード化された実行プログラムを格納するマイクロプログラム格納メモリ２３と、基本演算ブロックＦＢ１の内部動作を制御するコントローラ２１と、アドレスポインタ等として用いられるレジスタ群２２と、主演算回路２０における不良の救済を行なうためのヒューズプログラムを実行するためのヒューズ回路２４を含む。

【００４７】

コントローラ２１は、ホストＣＰＵ２から、システムバス５および入出力回路１０を介して与えられる制御命令により、制御が手渡されて、基本演算ブロックＦＢ１－ＦＢｎの動作を制御する。これらの基本演算ブロックＦＢ１－ＦＢｎにマイクロプログラム格納メモリ２３を設け、コントローラ２１が、このメモリ２３内に実行プログラムを格納することにより、基本演算ブロックＦＢ１－ＦＢｎそれぞれにおいて実行する処理内容を変更することができ、基本演算ブロックＦＢ１－ＦＢｎにおいて、それぞれ演算実行される処理内容を変更することができる。

【００４８】

隣接ブロック間データバス１６が、基本演算ブロックＦＢ１－ＦＢｎの間のデータ転送を行うために設けられる。この隣接ブロック間データバス１６は、内部データバス１２を占有することなく、基本演算ブロック間の高速データ転送を可能とし、たとえば、ある基本演算ブロックに内部データバス１２を介してデータ転送中に、別の基本演算ブロック間でデータ転送を行なうことができる。

【００４９】

集中制御ユニット１５は、制御用ＣＰＵ２５と、この制御用ＣＰＵが実行する命令を格納する命令メモリ２６と、制御用ＣＰＵ２５のワーキングレジスタまたはポインタ格納用のレジスタを含むレジスタ群２７と、マイクロプログラムのライブラリを格納するマイクロプログラムライブラリ格納メモリ２３を含む。集中制御ユニット１５は、内部バス１４を介してホストＣＰＵ２から制御権を手渡され、内部バス１４を介して基本演算ブロックＦＢ１－ＦＢｎの処理動作を制御する。

【００５０】

マイクロプログラムライブラリ格納メモリ２３に、各種シーケンス処理がコード化されたマイクロプログラムをライブラリとして格納することにより、集中制御ユニット１５から必要なマイクロプログラムを選択して、基本演算ブロックＦＢ１－ＦＢｎのマイクロプログラム格納メモリ２３に格納されるマイクロプログラムを変更することができ、処理内容の変更に柔軟に対応することができる。

【００５１】

また、ヒューズ回路２４を利用することにより、この基本演算ブロックＦＢ１－ＦＢｎそれぞれにおいて不良発生時、冗長置換を用いて不良救済を行なうことにより、歩留まりを改善する。

【００５２】

図２は、図１に示す基本演算ブロックＦＢ１－ＦＢｎそれぞれに含まれる主演算回路２０の要部の構成を概略的に示す図である。図２において、主演算回路２０は、メモリセル

ＭＣが行列状に配列されるメモリマット３０と、このメモリマット３０の一方端に配列される演算処理ユニット（ＡＬＵ）群３２を含む。

【００５３】

メモリマット３０においては、行列状に配列されるメモリセルＭＣが、ｍ個のエントリＥＲＹに分割される。エントリＥＲＹは、ｎビットのビット幅を有する。本実施の形態１においては、１つのエントリＥＲＹは、一列のメモリセルで構成される。

【００５４】

演算処理ユニット群３２は、このエントリＥＲＹそれぞれに対して設けられる演算処理ユニット（ＡＬＵ）３４を含む。演算処理ユニット３４は、加算、論理積、一致検出（ＥＸＯＲ）、および反転（ＮＯＴ）などの演算を実行することができる。

【００５５】

このエントリＥＲＹと対応の演算処理ユニット３４の間で、データのロードおよびストアを行なって演算処理を行なう。このエントリＥＲＹは、メモリマット３０の列方向に整列して配置されるメモリセルＭＣで構成され、演算処理ユニットＡＬＵ３４は、ビットシリアルな（データワードをビット単位で処理する）態様で演算処理を実行し、従って演算処理ユニット群３２において、ビットシリアルかつ複数のエントリが並行して処理されるエントリパラレルな態様でデータの演算処理が実行される。

【００５６】

演算処理ユニット（ＡＬＵ）３４において、ビットシリアル態様で演算処理を実行することにより、演算対象のデータのビット幅が異なる場合においても、単に演算サイクル数がデータワードのビット幅に応じて変更されるだけであり、その処理内容は変更されず、容易に語構成の異なるデータを処理することができる。

【００５７】

また、複数のエントリＥＲＹのデータを、演算処理ユニット群３２において同時に処理することができ、エントリ数ｍを多くすることにより、大量のデータを一括して演算処理することができる。

【００５８】

ここで、一例として、エントリ数ｍは、１０２４であり、１エントリのビット幅ｎは、５１２ビットである。

【００５９】

図３は、図２に示すメモリセルＭＣの構成の一例を示す図である。図３において、メモリセルＭＣは、電源ノードとストレージノードＳＮ１の間に接続されかつそのゲートがストレージノードＳＮ２に接続されるＰチャネルＭＯＳトランジスタＰＱ１と、電源ノードとストレージノードＳＮ２の間に接続されかつそのゲートがストレージノードＳＮ１に接続されるＰチャネルＭＯＳトランジスタＰＱ２と、ストレージノードＳＮ１と接地ノードの間に接続されかつそのゲートがストレージノードＳＮ２に接続されるＮチャネルＭＯＳトランジスタＮＱ１と、ストレージノードＳＮ２と接地ノードの間に接続されかつそのゲートがストレージノードＳＮ１に接続されるＮチャネルＭＯＳトランジスタＮＱ２と、ワード線ＷＬ上の電位に応答してストレージノードＳＮ１およびＳＮ２を、それぞれ、ビット線ＢＬおよび／ＢＬに接続するＮチャネルＭＯＳトランジスタＮＱ３およびＮＱ４を含む。

【００６０】

この図３に示すメモリセルＭＣは、フルＣＭＯＳ（相補ＭＯＳ）構成のＳＲＡＭ（スタティック・ランダム・アクセス・メモリ）セルであり、高速で、データの書込／読出を行なう。

【００６１】

メモリセルＭＣとしては、２つのＤＲＡＭセルがビット線ＢＬおよび／ＢＬの間に直列に接続されかつ共通のワード線ＷＬで選択状態とされる「ツインセル構造」のＤＲＡＭセルユニットが用いられてもよい。

【００６２】

主演算回路２０において演算を行なう場合には、まず各エントリＥＲＹに、演算対象データの格納を行なう。次いで、格納されたデータのある桁のビットを、すべてのエントリＥＲＹについて並列に読出して、対応の演算処理ユニット３４へ転送（ロード）する。２項演算の場合には、各エントリにおいて別のデータワードのビットに対しても同様の転送動作を行なった後、各演算処理ユニット３４で、２入力演算を行なう。この演算処理結果は、演算処理ユニット３４から対応のエントリ内の所定領域に再書込（ストア）される。

【００６３】

図４は、図２に示す主演算回路２０における演算操作を例示的に示す図である。この図４においては、２ビット幅のデータワードａおよびｂの加算を行なって、データワードｃを生成する。エントリＥＲＹには、演算対象の組をなすデータワードａおよびｂがともに格納される。

【００６４】

図４においては、第１行目のエントリＥＲＹに対する演算処理ユニットにおいては、１０Ｂ＋０１Ｂの加算が行なわれ、２行目のエントリに対する演算処理ユニットにおいては、００Ｂ＋１１Ｂの演算が行なわれる。ここで、“Ｂ”は、２進数を示す。３行目のエントリに対する演算処理ユニットにおいては１１Ｂ＋１０Ｂの演算が行なわれる。以下、同様に、各エントリに格納されたデータワードａおよびｂの加算演算が行なわれる。

【００６５】

演算は、下位側ビットから順にビットシリアル態様で行なわれる。まず、エントリＥＲＹにおいてデータワードａの下位ビットａ［０］を対応の演算処理ユニット（以下、ＡＬＵと称す）３４へ転送する。次にデータワードｂの下位ビットｂ［０］を対応のＡＬＵ３４へ転送する。ＡＬＵ３４においては、これらの与えられた２ビットデータを用いて加算演算を行なう。この加算演算結果ａ［０］＋ｂ［０］は、データワードｃの下位ビットｃ［０］の位置に書込まれる（ストアされる）。すなわち、１行目のエントリＥＲＹにおいては、“１”が、ｃ［０］の位置に書込まれる。

【００６６】

この加算処理を、次いで、上位ビットａ［１］およびｂ［１］に対しても行ない、その演算結果ａ［１］＋ｂ［１］が、ビットｃ［１］の位置に書込まれる。

【００６７】

加算演算においては、桁上がりが生じる可能性があり、この桁上がり（キャリ）値が、ビットｃ［２］の位置に書込まれる。これにより、データワードａおよびｂの加算が、すべてのエントリＥＲＹにおいて完了し、その結果がデータｃとして各エントリＥＲＹにおいて格納される。エントリ数ｍとして、たとえば１０２４を準備した場合、１０２４組のデータの加算を並列に実行することができる。

【００６８】

図５は、この加算演算処理時の内部タイミングを模式的に示す図である。以下、この図５を参照して、加算演算の内部タイミングについて説明する。ＡＬＵ３４において、２ビット加算器（ＡＤＤ）が利用される。

【００６９】

図５において、“Ｒｅａｄ”は、メモリマットから演算対象のデータビットを読出して対応のＡＬＵ３４に転送する動作（ロード）を示し、“Ｗｒｉｔｅ”は、ＡＬＵ３４の演算結果データを対応のエントリの対応のビット位置に書き込む動作（ストア）または動作を命令示す。

【００７０】

マシンサイクルｋにおいて、データビットａ［ｉ］がメモリマット３０から読出され、次のマシンサイクル（ｋ＋１）で、次の演算対象のデータビットｂ［ｉ］が読出され（Ｒｅａｄ）、ＡＬＵ３４の加算器（ＡＤＤ）にそれぞれ与えられる。

【００７１】

マシンサイクル（ｋ＋２）においては、ＡＬＵ３４の加算器（ＡＤＤ）において、与えられたデータビットａ［ｉ］およびｂ［ｉ］の加算処理が行なわれ、マシンサイクル（ｋ

＋３）で、加算結果ｃ［ｉ］が対応のエントリの対応の位置に書込まれる。

【００７２】

次のマシンサイクル（ｋ＋４）および（ｋ＋５）において、次の演算対象のデータビットａ［ｉ＋１］およびｂ［ｉ＋１］が読出されて、ＡＬＵ３４の加算器（ＡＤＤ）へ転送され、マシンサイクル（ｋ＋５）において、ＡＬＵ３４により加算処理が行なわれ、マシンサイクル（ｋ＋６）において加算結果がビット位置ｃ［ｉ＋１］へ格納される。

【００７３】

メモリマット３０とＡＬＵ３４の間でのデータビット転送に、それぞれ１サイクル必要とされ、ＡＬＵ３４において１マシンサイクルの演算サイクルが必要とされる。したがって、２ビットデータの加算および加算結果の格納を行なうために、４マシンサイクルが必要とされる。メモリマットを複数のエントリに分割し、各エントリに演算対象データの組をそれぞれ格納して、対応のＡＬＵ３４においてビットシリアル態様で演算処理を行なう方式の特徴は、１つ１つのデータの演算には、比較的多くのマシンサイクルが必要とされるものの、処理すべきデータ量が非常に多い場合には、演算の並列度を高くすることで、高速データ処理を実現することができる。また、ビットシリアル態様で演算処理を行なっており、処理されるデータのビット幅は固定されないため、種々のデータ構成を有する種々のアプリケーションに適用することができる。

【００７４】

演算対象のデータワードのビット幅がＮの場合、各エントリの演算には、４・Ｎマシンサイクルが必要となる。演算対象のデータワードのビット幅は、８ビットから６４ビット程度であり、エントリ数ｍをたとえば１０２４と大きくすることにより、並列演算処理時に、たとえば８ビットデータの場合、３２マシンサイクルで、１０２４個の演算結果を得ることができ、１０２４組のデータをシーケンシャルに処理する場合に比べて、大幅に処理時間を短縮することができる。

【００７５】

図６は、主演算回路２０の構成をより具体的に示す図である。メモリマット３０においては、メモリセルＭＣが行列状に配列され、各メモリセル行に対応してワード線ＷＬが配置され、メモリセル列それぞれに対応してビット線対ＢＬＰが配置される。メモリセルＭＣは、これらのビット線対ＢＬＰとワード線ＷＬの交差部に対応して配置される。ワード線ＷＬには、対応の行のメモリセルが接続され、また、ビット線対ＢＬＰには、対応の列のメモリセルが接続される。

【００７６】

エントリＥＲＹは、各ビット線対ＢＬＰに対応して設けられ、メモリマット３０においては、ビット線対ＢＬＰ０からＢＬＰ（ｍ－１）それぞれに対応して、エントリＥＲＹ０－ＥＲＹ（ｍ－１）が配置される。ビット線対ＢＬＰが対応のエントリＥＲＹと対応のＡＬＵ３４との間のデータ転送線として利用される。エントリＥＲＹを１列のメモリセルで構成することにより、１エントリに格納されるデータのビット幅が用途に応じてまたは処理内容に応じて変更される場合においても、ビットシリアル態様で対応のＡＬＵで演算処理を行うことができ、データビット幅の変更に容易に対応することができる。

【００７７】

メモリマット３０のワード線ＷＬに対して、コントローラ２１（図１参照）からのアドレス信号に従って、演算対象のデータビットが接続されるワード線ＷＬを選択状態へ駆動するロウデコーダ４６が設けられる。ワード線ＷＬには、エントリＥＲＹ０－ＥＲＹ（ｍ－１）の同一位置のメモリセルが接続されており、このロウデコーダ４６により、各エントリＥＲＹにおいて同一位置のデータビットを選択する。

【００７８】

演算処理ユニット群（ＡＬＵ群）３２においては、各ＡＬＵ３４がビット線対ＢＬＰ０－ＢＬＰ（ｍ－１）に対応して配置されるが、図６においては、明確には示していない。このＡＬＵ群３２とメモリマット３０との間に、データのロード／ストア（転送）を行なうためのセンスアンプ群４０およびライトドライバ群４２が設けられる。

【００７９】
センスアンプ群４０は、各ビット線対ＢＬＰに対応して設けられるセンスアンプを含み、対応のビット線対ＢＬＰ（ＢＬＰ０－ＢＬＰ（ｍ－１））に読出されるデータを増幅して、演算処理ユニット群３２の対応のＡＬＵ３４に伝達する。
【００８０】
ライトドライバ群４２も同様、ビット線対ＢＬＰ（ＢＬＰ０－ＢＬＰ（ｍ－１））それぞれに対応して配置されるライトドライバを含み、演算処理ユニット群３２の対応のＡＬＵ３４からのデータを増幅して対応のビット線対ＢＬＰへデータを転送する。
【００８１】
これらのセンスアンプ群４０およびライトドライバ群４２がビット線（データ転送線）とＡＬＵ３４との間の転送回路を構成し、メモリマットとＡＬＵとの間で双方向にデータを転送することができる。
【００８２】
これらのセンスアンプ群４０およびライトドライバ群４２に対し、入出力回路４８が設けられ、図１に示す内部データバス１２との間でデータの転送が行なわれる。この入出力回路４８のデータの入出力態様は、エントリ数およびデータビット幅に応じて適当に定められる。
【００８３】
演算処理ユニット群３２に対してさらに、スイッチ回路４４が設けられる。このスイッチ回路４４は、ＡＬＵ３４間の相互接続経路を、図１に示すコントローラ２１からの制御信号に基づいて設定する。これにより、バレルシフタ等と同様に、隣接ＡＬＵ間でのデータ転送のみならず、遠く物理的に離れたＡＬＵ間でのデータ転送を行なうことができる。このＡＬＵ間相互接続用スイッチ回路４４は、たとえば、ＦＰＧＡ（フィールド・プログラム・ゲート・アレイ）などを用いたクロスバースイッチで実現される。
【００８４】
また、このスイッチ回路４４としては、バレルシフタなどのように、１マシンサイクル内で複数ビット間のシフト動作を行なう構成が用いられてもよい。
【００８５】
なお、図６においては、図１に示す隣接ブロック間データバス１６は明確には示していない。この隣接ブロック間データバス１６は、ＡＬＵ間相互接続用スイッチ回路４４に接続されてもよく、また入出力回路４８とセンスアンプ群４０およびライトドライバ群４２との間の内部データ転送バスに接続されてもよい。
【００８６】
なお、演算処理ユニット群３２のＡＬＵ３４は、コントローラ２１からの制御信号に従ってその演算処理動作タイミングおよび演算操作内容が決定される。
【００８７】
図７は、１つのＡＬＵの構成の一例を示す図である。図７において、ＡＬＵ３４は、指定された演算処理を行なう算術演算論理回路５０と、対応のエントリから読出されたデータを一時的に格納するＡレジスタ５２と、対応のエントリから読出されたデータまたは算術演算論理回路５０の演算処理結果データまたはライトドライバへ転送するデータを一時的に格納するＸレジスタ５４と、加減算処理時のキャリまたはボローを格納するＣレジスタ５６と、この算術演算論理回路５０を演算処理の禁止を指定するマスクデータを格納するＭレジスタ５８を含む。
【００８８】
図６に示すセンスアンプ群４０およびライトドライバ群４２は、単位構成の基本回路として、対応のビット線対ＢＬＰに対応して設けられるセンスアンプ６２およびライトドライバ６０を含む。センスアンプ６２は、対応のエントリのメモリセルから読出されたデータを増幅して、Ａレジスタ５２またはＸレジスタ５４へその増幅データを内部データ転送線６３を介して転送する。ライトドライバ６０は、Ｘレジスタ５４に格納されたデータをバッファ処理して、対応のエントリのメモリセルへ対応のビット線対ＢＬＰを介して書

込む。

【００８９】

算術演算論理回路５０は、加算（ＡＤＤ）、論理積（ＡＮＤ）、論理和（ＯＲ）、排他的論理和（ＥＸＯＲ）、反転（ＮＯＴ）等の演算を実行することができ、その演算内容が、コントローラからの制御信号（図７には示さず）により設定される。Ｍレジスタ５８に格納されるマスクデータは、“０”のときに、このＡＬＵ３４の演算処理動作を停止させ、“１”のときに、このＡＬＵ３４の演算処理動作をイネーブルする。この演算マスク機能を利用することにより、仮に全エントリが利用されない場合においても有効エントリに対してのみ演算を実行することができ、正確な処理を行うことができ、また、不必要な演算を停止させることにより消費電流を低減することができる。

【００９０】

Ｘレジスタ５４は、また、スイッチ回路４４に含まれるＡＬＵ間接続回路６５を介して他のＡＬＵに接続される。このＡＬＵ間接続回路６５は、ＦＰＧＡセルなどのスイッチ回路で構成され、演算処理ユニット群３２に含まれる任意のＡＬＵ３４に対してデータを転送する際に用いられる。このＡＬＵ間接続回路６５の転送機能により、メモリマット内のさまざまな物理位置に格納されているデータとの演算を実現することができ、演算の自由度を高くすることができる。

【００９１】

ＡＬＵ間接続回路６５は、例えば、スイッチマトリクスで構成されればよく、また、その占有面積が問題となる場合には、転送可能なＡＬＵの経路が制限されてもよい。たとえば、ｍ個のエントリを複数のブロックにグループ化し、このグループ間でのデータ転送のみが行なわれるように、ＡＬＵ間接続回路６５の転送経路が制限されてもよい。

【００９２】

図８は、この図７に示すＡＬＵ３４の動作シーケンスを示す図である。図８においては、１ビット加算器を利用して、２項加算演算ａ＋ｂを実行する。

【００９３】

まず、マシンサイクル（ｋ－１）において、Ｍレジスタ５８に、ビット“１”をセットして演算処理実行を指定し、また、Ｃレジスタ５６を“０”にクリアして初期化する。

【００９４】

マシンサイクルｋにおいて、メモリマットから、データビットａ［ｉ］が読出され、センスアンプ６２を介してＸレジスタ５４に転送されて格納される。このＸレジスタ５４の格納値は、次のマシンサイクル（ｋ＋１）において確定する。

【００９５】

マシンサイクル（ｋ＋１）において、メモリマット３０から、データビットｂ［ｉ］が読出されてＡレジスタ５２に転送されて格納される。

【００９６】

マシンサイクル（ｋ＋２）においては、データビットａ［ｉ］およびｂ［ｉ］が確定状態にあるため、ＡＬＵ３４において演算が実行され、マシンサイクル（ｋ＋３）において、その演算結果（加算結果）ｃ［ｉ］の書込が、ライトドライバ６０を介して行なわれる。ＡＬＵ３４においては、マシンサイクル（ｋ＋２）において、加算結果ａ［ｉ］＋ｂ［ｉ］が確定しており、また、キャリＣ［ｉ］の有無も確定している。従って、マシンサイクル（ｋ＋３）において、ＡＬＵ３４のＸレジスタ５４からライトドライバ６０を介して、メモリマットのビットｃ［ｉ］に加算結果を書込むことができる。キャリＣ［ｉ］はＣレジスタ５６に格納され、その書込は行われない。

【００９７】

次のマシンサイクル（ｋ＋４）において、次の上位データビットａ［ｉ＋１］が読出され、ＡＬＵ３４に転送され、次のマシンサイクル（ｋ＋５）において、Ｘレジスタ５４の格納データビットが、ビットａ［ｉ＋１］に確定する。このマシンサイクル（ｋ＋５）において、メモリマット３０においてビットｂ［ｉ＋１］が読出される。このとき、メモリマットからＡＬＵ３４のＡレジスタ５２に対してビットの転送が行なわれており、Ａレジ

スタ５２（図７）においては、先のマシンサイクル（ｋ＋１）において読出されたデータビットｂ［ｉ］が格納されている（マシンサイクル（ｋ＋５）においてＡレジスタ５２の書き換えが行われ、その格納データが、マシンサイクル（ｋ＋６）においては確定状態にある。

【００９８】

マシンサイクル（ｋ＋６）において、Ａレジスタ５２およびＸレジスタ５４の格納データビットが確定状態にあり、これらのビットに対して演算（加算演算）が実行され、次のマシンサイクル（ｋ＋７）において、加算結果ａ［ｉ＋１］＋ｂ［ｉ＋１］が、ビットｃ［ｉ＋１］の位置に書込まれる。また、キャリＣ［ｉ＋１］が、Ｃレジスタに格納される。これらの一連の動作を、対応のエントリのデータワードａおよびｂの全ビットに対して繰返し実行することにより、データワードａおよびｂの加算演算が実現される。最終ビットの加算演算結果の書込後、Ｃレジスタの格納するキャリＣの書込が、データワードｃの格納領域の最上ビット位置に対して行なわれる。

【００９９】

メモリマット３０のワード線ＷＬの選択時、図６に示すロウデコーダ４６は、これらのデータワードａ、ｂおよびｃの各ビットの記憶領域の開始時点をレジスタ群のポインタ値として格納し、各マシンサイクルごとに、そのポインタ値を増分することにより、下位ビットから上位ビットへの加算および加算結果の格納を実現することができる。

【０１００】

図９に示すように、メモリマット３０において、データワードａを格納する領域＃Ａ、データワードｂを格納する領域＃Ｂおよび演算結果ワードｃを格納する領域＃Ｃにおいてそれぞれ、最下位ビット［０］の位置をポインタＰＡ、ＰＢおよびＰＣでそれぞれ指定し、各マシンサイクルごとに、これらのポインタを順次活性化するとともに、１ビットデータについての演算完了後、ポインタ値を増分する。この場合、メモリマット３０において、領域＃Ａおよび＃Ｂのビット幅が予め決定される場合、ポインタＰＢおよびＰＣとしては、ポインタＰＡに基づいてこのデータ領域＃Ａ、＃Ｂのビット幅に応じた加算値が用いられてもよい。これらのポインタＰＡ－ＰＣは、図１に示すレジスタ群２２に格納され、図６に示すロウデコーダ４６へ与えられる。

【０１０１】

このレジスタ群にポインタＰＡ－ＰＣを設定して、順次マシンサイクルごとにロウデコーダへ与えることにより、演算対象のデータワードのビット幅に応じて、メモリマット３０におけるデータワードの格納領域を設定することができる。

【０１０２】

ポインタＰＡ－ＰＣを発生する構成としては、カウント回路が用いられてもよく、また、コントローラ２１（図１参照）によりレジスタの格納値が更新されてもよい。

【０１０３】

以上のように、この発明の実施の形態１に従えば、メモリマットを複数のエントリに分割し、各エントリに対応して演算処理ユニットを設け、ビットシリアル態様で、各演算処理ユニットが並列に演算処理を行なっており、大量のデータを並列演算処理することができ、データビット幅に係らず高速演算処理を行なうことのできる演算装置を実現することができる。

【０１０４】

［実施の形態２］

図１０は、この発明の実施の形態２に従うメモリマットのメモリセルＭＣの構成を示す図である。この図１０において、メモリセルＭＣは、書込ポートと読出ポートが別々に設けられたデュアルポートメモリセルである。このメモリセルＭＣに対しては、読出ワード線ＲＷＬおよび書込ワード線ＷＷＬが設けられ、また読出ビット線ＲＢＬおよび／ＲＢＬと書込ビット線ＷＢＬおよび／ＷＢＬが設けられる。読出ポートは、この読出ワード線ＲＷＬの信号電位に応答して記憶ノードＳＮ１およびＳＮ２をそれぞれ読出ビット線ＲＢＬおよび／ＲＢＬに接続するＮチャネルＭＯＳトランジスタＮＱ５およびＮＱ６を含む。書

込ポートは、書込ワード線ＷＷＬ上の信号電位に応答してストレージノードＳＮ１およびＳＮ２を、それぞれ書込ビット線ＷＢＬおよび／ＷＢＬに接続するＮチャネルＭＯＳトランジスタＮＱ７およびＮＱ８を含む。

【０１０５】

このメモリセルＭＣのデータ記憶部は、負荷ＰチャネルＭＯＳトランジスタＰＱ１およびＰＱ２と、ドライブ用のＮチャネルＭＯＳトランジスタＮＱ１およびＮＱ２を含む。

【０１０６】

この図１０に示すデュアルポートメモリセル構造を利用することにより、ビットシリアル態様でデータの演算処理を行なう場合、書込および読出を同時に行なうことができる。この場合、演算結果が書込まれる領域は、演算対象のデータが格納される領域とは別に設けられており、これらのメモリセルにおいて、書込データおよび読出データの衝突は生じず、通常のマルチポートメモリにおけるアービトレーションの問題は生じない。

【０１０７】

図１１は、この発明の実施の形態２における演算処理動作時の内部タイミングを例示的に示す図である。以下、図１１を参照して、先の実施の形態１と同様、１ビット加算器を利用する演算処理操作について説明する。ＡＬＵの構成およびメモリマットのエントリの構成は、先の実施の形態１と同様である。従って、この実施の形態２においても、メモリマットは、各列に対応してエントリに分割されており、各エントリに対応して、ＡＬＵ（３４）が配置される。

【０１０８】

マシンサイクルｋにおいて、データビットａ［ｉ］が読出される（Ｒｅａｄ）。この読出動作時においては、データビットａ［ｉ］に対応する読出ワード線ＲＷＬが選択状態へ駆動され、メモリセルのストレージノードＳＮ１およびＳＮ２が、読出ビット線ＲＢＬおよび／ＲＢＬに結合されて、データビットａ［ｉ］の読出が行なわれる。

【０１０９】

次のマシンサイクル（ｋ＋１）において、次のデータビットｂ［ｉ］が読出され、対応のＡＬＵ３４の加算器（ＡＤＤ）へ与えられる。このＡＬＵ３４においては、マシンサイクル（ｋ＋２）において演算処理が行なわれ、その演算結果データｃ［ｉ］が、結果レジスタ、すなわちＸレジスタ５４に格納される。

【０１１０】

マシンサイクル（ｋ＋２）において、次のデータビットａ［ｉ＋１］がメモリマットから読出されて、ＡＬＵへ読出ビット線ＲＢＬおよび／ＲＢＬを介して転送される。

【０１１１】

マシンサイクル（ｋ＋３）において、マシンサイクル（ｋ＋２）で生成された演算処理結果（ａ［ｉ］＋ｂ［ｉ］）が、メモリマットのビットｃ［ｉ］の位置に書込まれる（Ｗｒｉｔｅ）。この書込動作時においては、ビットｃ［ｉ］に対応する書込ワード線ＷＷＬが選択状態へ駆動され、書込ポートのＭＯＳトランジスタＮＱ７およびＮＱ８が導通し、ストレージノードＳＮ１およびＳＮ２が、書込ビット線ＷＢＬおよび／ＷＢＬに接続され、対応のライトドライバからのデータビットが格納される。

【０１１２】

このマシンサイクル（ｋ＋３）において、並行して、データビットｂ［ｉ＋１］の読出が行なわれ、ＡＬＵ３４へ読出ビット線ＲＢＬおよび／ＲＢＬを介してこの読出されたデータビットｂ［ｉ＋１］が転送される。

【０１１３】

マシンサイクル（ｋ＋４）において、ＡＬＵにおいてビットａ［ｉ＋１］およびｂ［ｉ＋１］の加算が行なわれる。このマシンサイクル（ｋ＋４）において、加算（演算）操作と並行して、メモリマットにおいてビットａ［ｉ＋２］が読出されて、ＡＬＵへ転送される。

【０１１４】

マシンサイクル（ｋ＋５）において、マシンサイクル（ｋ＋４）において確定した演算

結果ａ［ｉ＋１］＋ｂ［ｉ＋１］が、ビットｃ［ｉ＋１］に書込ビット線を介して、対応のライトドライバから転送され、対応のメモリセルへ書込ポートを介して書込まれる。

【０１１５】

このマシンサイクル（ｋ＋５）において、また、書込と並行して、次の演算対象のビットｂ［ｉ＋２］が読出され、ＡＬＵ３４に転送される。マシンサイクル（ｋ＋５）において、ＡＬＵにおいて加算操作が行なわれ、その演算結果が、マシンサイクル（ｋ＋６）においてメモリマットのビットｃ［ｉ＋２］の位置に書込まれる。

【０１１６】

上述のように、デュアルポートメモリセルを利用する場合、書込動作および読出動作時においては、データビットは、それぞれ読出ビット線および書込ビット線と別々の経路を介して転送されるため、並行して書込データおよび読出データの転送を行なうことができる。データの書込は、２サイクルに１回であり、また、各データマシンサイクルにおいて演算対象のビットを読出すことができる。１ビット加算操作に必要なサイクルは、書込および読出が並行して行なわれるため、２サイクルに低減され、Ｎビットのデータ幅を有するデータワードの演算（加算）操作においては、２・Ｎサイクルで、加算処理を行なうことができ、４・Ｎサイクルが必要となる先の実施の形態１の処理性能と比べると、２倍の演算性能（処理速度）を実現することができる。従って、ビットシリアル態様で加算演算操作を行なっても、高速の演算処理を実現することができる。

【０１１７】

なお、加算演算実行と並行して、次の演算対象のデータビットが転送される。したがって、演算結果を格納するレジスタと、最初に転送される演算対象のデータビット（ａ［ｉ］）が格納するレジスタは、別々のレジスタ回路とするのがデータの衝突を避ける上で好ましい。例えば、先の図７に示すＡＬＵ３４の構成において、Ａレジスタ５２に最初に転送される演算対象のビットａ［ｉ］を格納し、Ｘレジスタ５４に加算演算結果を格納してライトドライバを介して転送する構成とすることにより、次の演算対象のデータビットａ［ｉ＋１］と演算結果ａ［ｉ］＋ｂ［ｉ］との衝突を防止することができる。

【０１１８】

図１２は、この発明の実施の形態２に従う主演算回路２０の構成を概略的に示す図である。この図１２に示す主演算回路２０の構成においても、メモリマット３０において、メモリセルＭＣは、デュアルポートＳＲＡＭセルで構成され、行列状に配列される。メモリセルＭＣの各行に対応して、書込ワード線ＷＷＬおよび読出ワード線ＲＷＬが配置される。メモリセルＭＣの各列に対応して、書込ビット線対ＷＢＬＰおよび読出ビット線対ＲＢＬＰが配置される。メモリセルＭＣの各列が、エントリＥＲＹとして利用される。ここで、書込ビット線対ＷＢＬＰは、書込ビット線ＷＢＬおよび／ＷＢＬで構成され、読出ビット線対ＲＢＬＰは、読出ビット線ＲＢＬおよび／ＲＢＬで構成される。

【０１１９】

周辺部のセンスアンプ群４０およびライトドライバ群４２、演算処理ユニット群３２およびＡＬＵ間相互接続用スイッチ回路４４および入出力回路４８は、先の実施の形態１と同様である。

【０１２０】

センスアンプ群４０は、各エントリＥＲＹ０－ＥＲＹ（ｍ－１）それぞれに対応して設けられるセンスアンプＳＡを含む。センスアンプＳＡは、対応のエントリの読出ビット線対ＲＢＬＰに接続され、かつ演算処理ユニット群３２の対応のＡＬＵに結合される。

【０１２１】

ライトドライバ群４２は、エントリＥＲＹ０－ＥＲＹ（ｍ－１）それぞれに対応して配置されるライトドライバＷＤを含む。このライトドライバＷＤは、対応のエントリの書込ビット線対ＷＢＬＰに接続される。ライトドライバＷＤは、対応のＡＬＵに結合され、演算処理結果データを対応の書込ビット線対ＷＢＬＰに転送する。

【０１２２】

読出ワード線ＲＷＬおよび書込ワード線ＷＷＬがそれぞれ別々に設けられており、従っ

て、ロウデコーダとして、書込ワード線ＷＷＬを選択するライト用ロウデコーダ３６ｗと、読出ワード線ＲＷＬを選択するリード用ロウデコーダ３６ｒが別々に設けられる。これらのロウデコーダ３６ｗおよび３６ｒは、図１に示すコントローラ２１から与えられたアドレス信号に従って、また制御信号に従って選択的に活性化され、指定されたワード線ＲＷＬおよびＷＷＬを選択状態へ駆動する。これらのロウデコーダ３６ｗおよび３６ｒに対するアドレスは、図１２においては明確には示していないが、先の実施の形態１と同様に、ポインタを利用して生成される。

【０１２３】

この実施の形態２における図１２に示す主演算回路２０の構成において、メモリマット３０において、メモリセルＭＣがデュアルポートメモリセルで構成され、かつ書込用および読出用にそれぞれ内部のデータ転送線としての読出ビット線対および書込ビット線対が設けられ、また読出用および書込用のワード線選択用のロウデコーダが別々に設けられる構成を除いては、実施の形態１と同じであり、ライト用ロウデコーダ３６ｗは、１ビット加算器を利用する加算演算操作の場合には２マシンサイクルに１回活性化され、また、リード用ロウデコーダ３６ｒは、各サイクルごとに活性化される。

【０１２４】

以上のように、この発明の実施の形態２に従えば、メモリセルをデュアルポートメモリセルで構成し、演算処理ユニットとメモリマットとの間で書込および読出データを並行して同時に転送するように構成しており、演算処理時間を短縮することができる。

【０１２５】

［実施の形態３］

図１３は、この発明の実施の形態３に従う主演算回路２０の要部の構成を概略的に示す図である。この図１３に示す主演算回路２０においては、演算処理ユニット群３２の両側に、メモリマット３０Ａおよび３０Ｂが配置される。これらのメモリマット３０Ａおよび３０Ｂは、同一構成を有し、データビット幅がｎビットのエントリＥＲＹが、それぞれｍ個配置される。このメモリマット３０Ａおよび３０Ｂの各エントリの間に、演算処理ユニット群３２のＡＬＵ３４が配置される。このＡＬＵ３４は、メモリマット３０Ａおよび３０Ｂの対応のエントリをデータについて、指定された演算処理を行なう。２項演算を各ＡＬＵ３４が行なう場合、メモリマット３０Ａおよび３０Ｂに、各項の演算対象データを格納し、その演算処理結果は、メモリマット３０Ａおよび３０Ｂの一方に格納する。従って、メモリマット３０Ａおよび３０Ｂにおいては、格納されるデータ量が１つのメモリマットを利用する構成に比べて少なくすることができる。メモリマット３０Ａおよび３０Ｂのエントリの合計サイズ（ビット幅）が、実施の形態１または２のメモリマット３０のエントリのサイズ（ビット幅）と同程度にされてもよい。メモリセルとしては、先の実施の形態２と同様、デュアルポートメモリセルが利用される。

【０１２６】

図１４は、この発明の実施の形態３における主演算回路２０の演算シーケンスの内部タイミングを示す図である。以下、図１４を参照して、この図１３に示す主演算回路２０の演算操作について説明する。

【０１２７】

メモリマット３０Ａおよび３０Ｂには、演算対象のデータワードａおよびｂの組が、それぞれ対応のエントリに格納される。マシンサイクルｋにおいて、メモリマット３０Ａおよび３０Ｂから、対応のデータビットａ［ｉ］およびｂ［ｉ］が読出される。

【０１２８】

マシンサイクル（ｋ＋１）において、ＡＬＵにおいて、ＡＤＤ演算処理（加算処理）がこれらのデータビットａ［ｉ］およびｂ［ｉ］に対して行なわれる。メモリマット３０Ａおよび３０Ｂは、メモリセルがデュアルポートメモリセルで構成されており、演算結果が、マシンサイクル（ｋ＋２）においてメモリマット３０Ａのビットｃ［ｉ］に書込まれる。一方、マシンサイクル（ｋ＋１）においては、次のデータビットａ［ｉ＋１］およびｂ［ｉ＋１］が読出され、対応のＡＬＵへ与えられ、マシンサイクル（ｋ＋２）において、

書込データ（加算演算結果データ）のビットｃ［ｉ］への転送と並行して次の演算対象のデータビットの組に対して加算演算操作が行なわれる。

【０１２９】

このマシンサイクル（ｋ＋２）においては、再び次の演算対象のデータビットａ［ｉ＋２］およびｂ［ｉ＋２］が読出され、ＡＬＵ３４に転送される。

【０１３０】

マシンサイクル（ｋ＋３）においては、マシンサイクル（ｋ＋２）においてＡＬＵ３４で行なった演算操作結果が確定するため、対応のメモリセルビットｃ［ｉ＋１］への演算結果データの書込が行なわれる。このマシンサイクル（ｋ＋３）においては、さらに、次のデータビットａ［ｉ＋３］およびｂ［ｉ＋３］の読出が行なわれ、ＡＬＵ３４への転送が行なわれる。

【０１３１】

したがって、このメモリマット３０Ａおよび３０Ｂに２項演算の各項のデータワードをそれぞれ対応するエントリに格納し、同一マシンサイクルで、これらのメモリマット３０Ａおよび３０Ｂから対応のデータビットを読出してＡＬＵへ転送することにより、各マシンサイクルにおいてデータの書込を行なうことができる。したがって、Ｎビットのデータ幅を有するデータワードの加算の場合、Ｎマシンサイクルで、演算操作を完了することができ、さらに、動作速度（処理速度）を高速化することができる。

【０１３２】

図１５は、この発明の実施の形態３に従う主演算回路２０の構成をより具体的に示す図である。メモリマット３０Ａおよび３０Ｂにおいては、メモリセルＭＣが、先の実施の形態２に示すメモリセルの構成と同様、デュアルポートメモリセルであり、書込ワード線ＷＷＬおよび読出ワード線ＲＷＬが、行方向に配列されるメモリセルに対応して設けられ、また列方向に整列するメモリセルに対して、書込ビット線対ＷＢＬＰおよび読出ビット線対ＲＢＬＰがそれぞれ配置される。これらのメモリマット３０Ａおよび３０Ｂは、それぞれエントリＥＲＹ０－ＥＲＹ（ｍ－１）のｍ個のエントリをそれぞれ有し、これらのエントリが対応して配置される。

【０１３３】

図１５においては明確に示していないが、メモリマット３０Ａおよび３０Ｂの間に、演算処理ユニット群３２が設けられる。この演算処理ユニット群３２に対しては、先の実施の形態１と同様、ＡＬＵ間相互接続用スイッチ回路が同様配置され、物理的に離れた位置のＡＬＵ間のデータ転送を可能にする。

【０１３４】

この演算処理ユニット群３２とメモリマット３０Ａの間に、センスアンプ群４０Ａおよびライトドライバ群４２Ａが配置され、演算処理ユニット群３２とメモリマット３０Ｂの間に、センスアンプ群４０Ｂおよびライトドラバ群４２Ｂが配置される。

【０１３５】

センスアンプ群４０Ａは、メモリマット３０Ａの読出ビット線対ＲＢＬ（ＲＢＬＰ０－ＲＢＬＰ（ｍ－１））それぞれに対応して配置されるセンスアンプＳＡを含み、ライトドライバ群４２Ａはメモリマット３０Ａの書込ビット線対ＷＥＬＰ（ＷＥＬＰ０－ＷＥＬＰ（ｍ－１））それぞれに対応して配置されるライトドライバＷＤを含む。

【０１３６】

センスアンプ群４０Ｂも、同様、メモリマット３０Ｂの読出ビット線対ＲＢＬＰ（ＲＢＬＰ０－ＲＢＬＰ（ｍ－１））それぞれに対応して設けられるセンスアンプＳＡを含み、ライトドライバ群４２Ｂは、このメモリマット３０Ｂの書込ビット線対ＷＢＬＰ（ＷＢＬＰ０－ＷＢＬＰ（ｍ－１））それぞれに対応して配置されるライトドライバＷＤを含む。

【０１３７】

メモリマット３０Ａに対しては、読出ワード線ＲＷＬを選択するリード用ロウデコーダ３６ｒＡおよび書込ワード線ＷＷＬを選択するライト用ロウデコーダ３６ｗＡが設けられ、メモリマット３０Ｂに対しても、同様、読出ワード線ＲＷＬを選択するためのリード用

ロウデコーダ３６ｒＢおよび書込ワード線ＷＷＬを選択するライト用ロウデコーダ３６ｗＢが設けられる。

【０１３８】

このセンスアンプ群４０Ａおよびライトドライバ群４２Ａとライトドライバ群４２Ｂおよびセンスアンプ群４０Ｂに対して、内部データバス（図１のバス１２）とデータの転送を行なう入出力回路４９が設けられる。

【０１３９】

この入出力回路４９は、実施の形態１と異なり、メモリマット３０Ａおよび３０Ｂそれぞれに転送されるデータを並列に受けて転送する。これらのメモリマット３０Ａおよび３０Ｂそれぞれに格納されるデータそれぞれがメモリマット単位で、ビット位置の並べ替えが行なわれてもよく、またメモリマット３０Ａおよび３０Ｂそれぞれに、直並列変換および並直列変換用のレジスタ回路が配置され、ワード線単位でのデータの書込および読出がこのレジスタ回路とメモリマットの間で行なわれて、外部とのデータの入出力が行なわれてもよい。また、他の構成が利用されてもよい。

【０１４０】

ライト用ロウデコーダ３６ｗＡおよび３６ｗＢおよびリード用ロウデコーダ３６ｒＡおよび３６ｒＢは、先の実施の形態２の構成と同様である。リード用ロウデコーダ３６ｒＡおよび３６ｒＢが、同一マシンサイクルで、同一ビット位置の読出ワード線を選択状態へ駆動する。演算操作結果が、メモリマット３０Ａに格納される場合には、ライト用ロウデコーダ３６ｗＡが活性化されて、対応の書込ワード線が選択状態へ駆動される。この場合、メモリマット３０Ｂにおけるライト用ロウデコーダ３６ｗＢは、非活性状態に維持される。

【０１４１】

以上のように、この発明の実施の形態３に従えば、メモリマットを２つ配置し、これらの間にＡＬＵ群を配置しており、各メモリマットに演算対象のデータの組をそれぞれ格納することにより、各マシンサイクルごとに演算、データの書込およびデータの読出を行なうことができ、高速演算処理が実現される。

【０１４２】

［実施の形態４］

図１６は、この発明の実施の形態４に従う主演算回路２０の構成を概略的に示す図である。この図１６において、メモリマット３０においてデュアルポートメモリセルＭＣが行列状に配列される。メモリマット３０においては、ワード線ＷＬＡおよびＷＬＢが互いに直交する方向に配列され、またビット線対ＢＬＰＡおよびＢＬＰＢが互いに直交する方向に配置される。すなわち、ワード線ＷＬＡおよびビット線対ＢＬＰＢが平行して配置され、ワード線ＷＬＢおよびビット線対ＢＬＰＡが平行に配置される。このメモリマット３０に対して、演算処理を行なうための、演算処理ユニット群３２、メモリマット３０と演算処理ユニット群３２の間でデータの転送を行なうためのセンスアンプ群Ａ７１およびライトドライバ群Ａ７３と、演算処理ユニット（ＡＬＵ）群３２のＡＬＵ間のデータ転送経路を切換えるＡＬＵ間相互接続用スイッチ回路４４が設けられる。

【０１４３】

センスアンプ群Ａ７１においては、ビット線対ＢＬＰＡに対してセンスアンプＳＡが設けられ、ライトドライバ群Ａ７３においては、ビット線対ＢＬＰＡに対してワードドライバＷＤが設けられる。演算処理ユニット（ＡＬＵ）群３２においては、したがって、このビット線対ＢＬＰＡが１つのエントリを構成し、１つのビット線対ＢＬＰＡに対して１つのＡＬＵが配置される。

【０１４４】

一方、ビット線対ＢＬＰＢに対して、センスアンプ群Ｂ７０とライトドライバ群Ｂ７２と内部データバス１２（図１参照）との間でデータ転送を行なう入出力回路７４が設けられる。これらのセンスアンプ群７０、ライトドライバ群７２および入出力回路７４は、内部データバス１２とメモリマット３０の間のビット線対ＢＬＰＢの間でデータ転送を行な

う。したがって、このセンスアンプ群Ｂ７０、ライトドライバ群７２および入出力回路７４は、そのビット幅が、メモリマット３０の１つのエントリのビット幅と等しくされる（コラムデコーダは設けられていない）。

【０１４５】

ワード線ＷＬＡに対してはロウデコーダＡ６６が設けられ、ワード線ＷＬＢに対してはロウデコーダＢ７６が設けられる。これらのロウデコーダＡ６６およびロウデコーダＢ７６は、図１のコントローラ２１から与えられるアドレスに従ってワード線の選択を行なう。

【０１４６】

図１６に示す主演算回路２０の構成において、メモリマット３０の列を選択することは要求されないため、コラムデコーダは設けられない。エントリ単位で内部データバス１２とめモリマット３０との間でデータの転送が行なわれ（エントリシリアルビットパラレルにデータの転送が行われ）、また、演算実行時には、各エントリに対して並行にビットシリアルにデータ転送が演算処理ユニット群３２との間で実行されて、演算処理が行なわれる。

【０１４７】

すなわち、データの内部データバス１２との転送時においては、ロウデコーダＢ７６によりワード線ＷＬＢを選択して、１つのエントリを選択状態へ駆動して、エントリシリアルかつビットパラレルで内部データバス１２との間でデータ転送を行なう。演算実行時においては、演算処理ユニット群３２に対しては、各エントリ内のデータをビットシリアル態様で転送してビットシリアルかつエントリパラレルの態様で、演算操作を実行する。

【０１４８】

したがって、内部データバスとのデータ転送時においては、エントリの数に等しいサイクルだけ時間がデータ転送に必要とされる。２項演算を行なう場合、演算結果を格納する領域には、データを格納する必要はない。この場合、単に、結果データ書込領域には、データ“０”が格納されればよい。

【０１４９】

また、入出力回路７４において、内部データバス１２からの演算対象のデータをワードシリアル態様で受ける場合、この入出力回路７４において、データ入力部に、ワードシリアルに入力されるデータワード（演算対象データ）を並列データに変換して、センスアンプ群７０を介して対応のエントリに書込む構成が利用されてもよい。また、演算結果データのみの内部データバス１２への転送が必要とされる場合、入出力回路７４においては、ライトドライバ群Ｂ７２の出力のうち、コントローラ２１の出力する制御信号の制御の下に、演算結果データ領域のデータのみを選択的に内部データバス１２に出力するように構成されてもよい。従って、内部データバス１２のバス幅は、エントリのビット幅に等しくする必要はない。

【０１５０】

図１７は、図１６に示すメモリセルＭＣの構成の一例を示す図である。この図１７において、メモリセルＭＣは、交差結合される負荷ＰチャネルＭＯＳトランジスタＰＱ１およびＰＱ２と、交差結合されるドライブＮチャネルＭＯＳトランジスタＮＱ１およびＮＱ２を記憶部として含む。このメモリセルＭＣは、さらに、ワード線ＷＬＡ上の信号に応答してストレージノードＳＮ１およびＳＮ２をビット線ＢＬＡおよび／ＢＬＡに接続するＮチャネルＭＯＳトランジスタＮＱＡ１およびＮＱＡ２と、ワード線ＷＬＢ上の信号電位に従ってストレージノードＳＮ１およびＳＮ２をビット線ＢＬＢおよび／ＢＬＢに接続するＮチャネルＭＯＳトランジスタＮＱＢ１およびＮＱＢ２を含む。

【０１５１】

ビット線ＢＬＡおよび／ＢＬＡがビット線対ＢＬＰＡを構成し、ビット線ＢＬＢおよび／ＢＬＢが、ビット線対ＢＬＰＢを構成する。ワード線ＷＬＢは、ビット線ＢＬＡおよび／ＢＬＡと平行に配設され、ワード線ＷＬＡが、ビット線ＢＬＢおよび／ＢＬＢと平行に配設される。これにより、メモリマット３０において、データ書込時と演算操作時に行列

方向を９０°回転させて、データの外部との転送および演算処理時のデータ転送を実現することができ、内部データバス１２との間のデータ転送に要する時間を短縮することができる。

【０１５２】

なお、ビット線対ＢＬＡおよび／ＢＬＡとビット線ＢＬＢおよび／ＢＬＢが直交し、またワード線ＷＬＡおよびＷＬＢが直交する。この場合、ワード線ＷＬＡおよびＷＬＢを異なる配線層で構成し、またビット線ＢＬＢおよび／ＢＬＢとビット線ＢＬＡおよび／ＢＬＡを異なる配線層で形成することにより、このビット線の直交配置およびワード線の直交配置を実現することができる。

【０１５３】

以上のように、この発明の実施の形態４に従えば、メモリマット３０において、ビット線を互いに直交する方向に配置するとともに、ワード線も互いに直交する方向に２組配置しており、データ転送を外部の内部データバスと行なう場合には、エントリシリアルビットパラレル態様で行なうことができ、内部データバスとメモリマットとの間のデータ転送に要する時間を短縮することができ、この結果、高速演算処理を実現することができる。

【０１５４】

［実施の形態５］

図１８は、この発明に従う演算装置における演算対象データの分布の一例を概略的に示す図である。この図１８においては、メモリマット３０においてエントリＥＲＹがｍ個配置され、また各エントリＥＲＹは、そのデータビット幅がｎビットである。エントリＥＲＹそれぞれに対応して、ＡＬＵ３４が配置される。

【０１５５】

演算対象データ領域ＯＰＲに格納される演算対象データが、エントリのデータビット幅方向において一部分であるものの、メモリマット３０のエントリＥＲＹ全体にわたって分布している場合、ビットシリアルでかつエントリパラレルの態様で、演算処理を実行することにより、高い演算性能を実現することができる。

【０１５６】

しかしながら、演算処理内容によっては、図１９に示すように、演算対象データが、少数のエントリにわたる領域ＯＰＢにしか存在しない場合もある。図１９においては、演算対象データ領域ＯＰＢが、３つのエントリＥＲＹに分布するだけであり、残りのエントリには、演算対象データが存在しない。このような場合、ビットシリアルかつエントリパラレルで演算処理を行なっても、データビット幅ｎが、演算対象データを格納する有効エントリ数ｍよりも大きいため、演算処理性能がかえって低下する。このようなエントリ間での並列度が低いデータに対しても、効率的に高速で演算処理を行なうのが、汎用性の高い演算装置を実現する上で重要である。

【０１５７】

図２０は、この発明の実施の形態５に従う演算装置の構成を概略的に示す図である。図２０において、メモリマット３０においては、先の実施の形態１から４と同様、メモリセルＭＣが行列状に配列される。図２０の水平方向に並列されるメモリセルＭＣにより、１つのエントリＥＲＹが形成され、各エントリＥＲＹに対して、演算処理ユニット群３２においてＡＬＵ３４が配置される。

【０１５８】

一方、この図の垂直方向のメモリセルに対して、スイッチ回路８０を介して、演算器群８２が配置される。この演算器群８２においては、複数のＡＬＵ８４が配置される。これらのＡＬＵ８４は、その演算内容は、個々に設定可能であり、いわゆるＭＩＭＤ（マルチ・インストラクション・データ・ストリーム）演算を実現することができる。

【０１５９】

スイッチ回路８０は、このメモリマット３０の列方向に整列されるメモリセル列を接続経路を切換えて、演算器群８２のＡＬＵ８４に接続する。このスイッチ回路８０は、たとえば、ＦＰＧＡなどのクロスバースイッチで構成される。また、これに代えて、スイッチ

マトリクスで、このスイッチ回路８０が構成されてもよい。スイッチ回路８０としては、メモリマット３０の列と演算器群８２のＡＬＵ８４との接続経路を選択的に切換えて確立する構成が利用されればよい。

【０１６０】

この図２０に示す構成の場合、エントリ間並列度の高いデータの演算処理を行なう場合には、メモリマット３０の右側に配置された演算処理ユニット群３２のＡＬＵ３４を用いてビットシリアルかつエントリパラレルで演算を行ない、並列演算処理による処理性能を発揮する。

【０１６１】

一方、図１９に示すように、エントリ間並列度が低く、演算対象データが少数のエントリ間に分布する場合、スイッチ回路８０を介して、メモリマット３０を演算を実行するＡＬＵ８４に接続する。この場合、エントリシリアルかつビットパラレルで、各エントリごとに、演算処理が実行される。したがって、エントリＥＲＹにおいて演算対象データａおよびｂが存在する場合、この演算処理対象データａおよびｂが並列に読出されて、スイッチ回路８０を介して対応のＡＬＵ８４に結合されて演算処理が行なわれまたその演算結果が、対応のエントリＥＲＹの演算結果格納領域（ｃ：図示せず）に格納される。

【０１６２】

このスイッチ回路８０を利用することにより、各エントリＥＲＹにおいて、演算対象データ格納領域および演算結果書込領域を自由に設定することができ、またエントリ数が少数であるため、エントリシリアルに演算処理を行なっても、その処理時間の増大は抑制される。特に、メモリマット３０と演算器群８２とが同一チップ上に形成される場合、これらの間の内部配線がチップ上配線であり、高速でデータの転送を行なうことができ、メモリマットと演算器ＡＬＵ（８４）とが別々のチップに設けられている構成に比べて、高速でデータを転送して処理を実行することができる。

【０１６３】

また、スイッチ回路８０により、メモリマット３０の列と演算器群８２のＡＬＵ８４との接続を設定することにより、演算処理対象のデータのビット幅が変更される場合においても、容易に対応することができる。たとえば、演算器群８２において、ＡＬＵ８４が８ビットの演算処理を行なう構成の場合、１６ビットデータの処理の場合には、隣接する２つのＡＬＵ８４を利用して、上位バイトおよび下位バイトをビットＡＬＵに与えて演算処理を行なうことにより、データビット幅が異なる場合にも容易に対応することができる。ただし、この場合、加算処理を行なう場合キャリー伝搬があるため、隣接ＡＬＵ間でキャリーの伝搬を行なう必要がある。この場合、図７に示すＣレジスタ５６において、シフト機能を持たせることにより、このような上位バイトおよび下位バイト並列演算処理時においてもキャリー伝搬を行なって加算を行なうことができる。

【０１６４】

実際の信号処理においては、アプリケーションによって演算対象データの形態が大きく異なる。また、ある１つのアプリケーションにおいても、処理のプロセスごとにデータの形態が異なる場合がある。したがって、この演算対象データの形態に応じて、メモリマット３０の右側に配置される演算処理ユニット３２のＡＬＵ３４を利用するかまたは、このメモリマット３０の下側に配置される演算器群８２のＡＬＵ８４を用いて処理を行なうかを、図１に示すコントローラ２１の制御の下にダイナミックに切換える。これにより、演算対象データの形態にかかわらず、高い演算処理能力を有する演算装置を実現することができる。

【０１６５】

特に、この演算処理ユニット群３２のＡＬＵ３４を利用してデータ処理を行なう状態と、このメモリマット３０の下辺に配置される演算器群８２のＡＬＵ８４を用いてデータ処理を行なう状態の切換は、図１に示すコントローラ２１からの演算内容に応じたリアルタイムで生成される制御信号によって行なわれる。この場合、メモリマット３０の右辺および下側に配置されるロウデコーダ、センスアンプ、ライトドライバおよびＡＬＵ群を動作

させるかが制御信号によって切換えられる（活性／非活性化される）だけであり、この演算形態切換に伴う時間的なオーバーヘッドは全くなく、時間的に連続的に、演算処理形態を切換えて、演算処理を実行することができる。

【０１６６】

図２１は、この発明の実施の形態５に従う主演算回路２０の構成をより具体的に示す図である。図２１において、メモリマット３０の右側に、エントリパラレルで演算処理を行なうための演算処理ユニット群３２が配置され、メモリマット３０の下側に、エントリシリアルかつビットパラレルで演算処理を行なう演算器群（ＡＬＵ群Ｂ）８２が配置される。演算処理ユニット群３２とメモリマット３０の間に、センスアンプ群Ａ７１とライトドライバ群Ａ７３が配置され、演算器群８２とメモリマット３０の間に、センスアンプ群Ｂ７０とライトドライバ群Ｂ７２が配置される。

【０１６７】

演算処理ユニット群３２に対しては、またＡＬＵ間の転送を行なうためのＡＬＵ間相互接続用スイッチ回路４４が設けられる。演算器群８２とライトドライバ群Ｂ７２およびセンスアンプ群Ｂ７０の間に、スイッチ回路８０が設けられる。このセンスアンプ群Ｂ７０およびライトドライバ群Ｂ７２は、入出力回路８７に結合され、内部データバス１２とデータ転送を行なう。したがって、この入出力回路８７は、先に実施の形態３における図１６に示す入出力回路７４と同様の構成であってもよい。

【０１６８】

メモリマット３０の右側および下側に、演算対象データを転送するために、メモリマット３０においては、図の水平方向に沿ってビット線対ＢＬＰＡが配置され、このビット線対ＢＬＰＡと直交する方向にビット線対ＢＬＰＢが配置される。ビット線対ＢＬＰＡと平行に、ワード線ＷＬＢが配置され、ビット線対ＢＬＰＢと平行に、ワード線ＷＬＡが配置される。ビット線対ＢＬＰＡは、センスアンプ群Ａ７１およびライトドライバ群Ａ７３に結合され、それぞれセンスアンプＳＡおよびライトドライバＷＤにビット線対ＢＬＰＡが結合される。

【０１６９】

ビット線対ＢＬＰＢは、センスアンプ群Ｂ７０のセンスアンプおよびライトドライバ群７２のライトドライバに結合される。これらのセンスアンプ群Ｂ７０およびライトドライバ群７２と演算器群８２のＡＬＵとの接続は、スイッチ回路８０によりその経路が決定される。スイッチ回路８０の接続経路は、図１に示すコントローラ２１からのスイッチング情報により設定され、また、演算器群８２のＡＬＵの実行する演算内容は、コントローラ２１からの再構成情報に基づいてその演算処理内容が決定される、または活性化されるＡＬＵが指定される。

【０１７０】

ワード線ＷＬＡに対してロウデコーダＡ６６が設けられ、ワード線ＷＬＢに対してロウデコーダＢ７６が設けられる。これらのロウデコーダＡ６６およびロウデコーダＢ７６に対しては、コントローラ２１からアドレス信号またはレジスタに格納されたポインタが与えられ、これらのロウデコーダＡ６６およびロウデコーダＢ７６の一方が、演算内容に応じて、コントローラ２１により選択的に活性化される。

【０１７１】

このコントローラ２１により、メモリマット３０の右側の演算処理ユニット群３２および下側の演算器群８２の一方を選択的に活性化することにより、演算処理内容をダイナミックに変更することができる。

【０１７２】

メモリセルＭＣが、このワード線ＷＬＡおよびＷＬＢとビット線対ＢＬＰＡおよびＢＬＰＢの交差部に対応して配置される。メモリセルＭＣの構成は、先の実施の形態４において図１７を参照して説明したメモリセルＭＣのデュアルポートメモリセル構造を利用することができる。これにより、データバス１２とメモリマット３０の間での効率的なデータ転送を実現するとともに、演算処理、処理プログラムにおいて更新することができる。

【０１７３】
以上のように、この発明の実施の形態５に従えば、メモリマットの直交する２辺にそれぞれＡＬＵ群を配置しており、演算対象データの形態に応じてビットパラレルかつエントリシリアルまたはエントリシリアルかつビットパラレルの演算処理を実現することができ、演算処理データの形態にかかわらず高速演算を実現することができる。
【０１７４】
また、このメモリマットの各列と演算器群との接続経路を切換えるスイッチ回路８０を設けることにより、１つのエントリ内に、演算対象の組のデータが配置される場合においても、確実に、演算対象のデータの組を対応のＡＬＵに転送することができる。また、このスイッチ回路を利用することにより、ＡＬＵの演算ビット幅をも変更することができ、また各ＡＬＵの演算内容を変更することにより、複数命令を並列に実行するＭＩＭＤ演算を実現することができる。
【０１７５】
［実施の形態６］
図２２は、この発明の実施の形態６に従う主演算回路の構成を概略的に示す図である。この図２２に示す主演算回路２０の構成においても、メモリマット３０が、複数のエントリＥＲＹ（ｍ個）に分割され、このメモリマット３０の図の右側に、演算処理ユニット群３２の各ＡＬＵ３４が各エントリＥＲＹに対応して配置される。一方、メモリマット３０の下側に、スイッチ回路９０を介して演算器群８２のＡＬＵ８４が配置される。この演算器群８２のＡＬＵ８４は、図２２に示すように、コントローラ２１からの再構成情報に従って、この演算ビット幅を変更することができる。たとえば、このＡＬＵが、加算器の場合、８ビット演算から１６ビット演算に変更される場合には、キャリーの伝搬経路を延長するために８ビット加算演算を行なう２つのＡＬＵにおいてキャリーの伝搬経路が接続される。これは、単にセレクタまたはスイッチ回路を選択的に導通状態とすることにより、８ビット加算器または１６ビット加算器を択一的に設定することができる。
【０１７６】
減算を行なう場合、例えば、２の補数表示のデータを用いて８ビット減算から１６ビット減算を行なう場合、上位バイトの最下位ビットでの“１”加算に代えて、下位バイトの最上位ビットからのキャリーが与えられる。この構成も、コントローラからの再構成情報に従ってセレクタを用いることにより、容易に実現することができる。
【０１７７】
したがって、たとえば図２２に示すように、演算器群８２の２つのＡＬＵ８４ａおよび８４ｂを用いて演算を行なう場合、これらのＡＬＵ８４ａおよび８４ｂのビット幅の合計のビット幅のＡＬＵ８８を等価的に実現して演算を実行することができる。
【０１７８】
スイッチ回路９０は、先の実施の形態５と同様、コントローラからの接続経路情報に従ってその接続経路が設定される。このデータビットの変更時、コントローラ２１からの演算器再構成情報に基づいて、演算器群８２のＡＬＵ８４の構成が再構成される。この再構成情報は、コントローラ２１から演算対象データのビット幅に応じてダイナミックに与えられるため、この演算器群８２の再構成に伴う時間的オーバーヘッドは存在しない。したがって、高速で、種々のデータビット幅のデータに対して演算処理を実行することができる。
【０１７９】
なお、スイッチ回路９０は、先の実施の形態５のスイッチ回路８０と同様の、ＦＰＧＡなどのクロスバースイッチ回路を利用することができ、また、単に、データ転送経路を切換えるスイッチマトリクスで構成されてもよい。
【０１８０】
以上のように、この発明の実施の形態６に従えば、エントリシリアルかつビットパラレルの演算処理時、演算器の処理演算ビット幅を変更可能に設定しており、種々のデータのビット幅に対応して高速で演算処理を実行することができる。

【０１８１】
［実施の形態７］
図２３は、この発明の実施の形態７に従う主演算回路の要部の構成を概略的に示す図である。この図２３においては、メモリマット３０の下部に配置される演算器群８２に対応する部分の構成を示す。この図２３に示す構成においては、演算器群として、複数段の演算器群ＯＧ１－ＯＧｋが配置される。メモリマット３０と各演算器群ＯＧ１－ＯＧｋの間に、スイッチ回路ＳＫ０，ＳＫ１，ＳＫ２…が配置される。演算器群ＯＧ１－ＯＧｋそれぞれにおいては、ＡＬＵ（演算器）１００が配置される。これらのＡＬＵ１００は、コントローラからの再構成情報に従ってその内部構成およびビット幅を変更することができるようにされてもよい。ＡＬＵ１００は全てその構成が同一とされてもよい。
【０１８２】
信号処理演算においては、積和演算を繰返すなど非常に複雑な演算処理が多い。したがって、１段のＡＬＵ群を用いた場合、十分な処理速度を得ることができない場合が考えられる。この図２３に示すように複数段の演算器群ＯＧ１－ＯＧｋを利用し、これらの間のスイッチ間ＳＫ０，ＳＫ１，ＳＫ２…で接続経路を設定する。これに、複数段の演算器群ＯＧ１－ＯＧｋ、たとえば乗算および加算などの異なる演算を順次実行することにより、パイプライン的に処理を実行することにより、高速処理が実現される。また、１つの乗算処理において、中間の部分積を生成する加算部、中間部分積を加算して最終積を生成する最終積回路を、各段のＡＬＵでそれぞれ実現することにより、高速の乗算装置を実現することができる。
【０１８３】
また、スイッチ回路ＳＫ０－ＳＫ２，…を用いて、各演算器群ＯＧ１－ＯＧｋのＡＬＵ１００の接続経路を設定しており、物理的に位置の離れたＡＬＵの演算結果同士で新たな演算を実行することができ、非常に複雑な演算も実現することができる。
【０１８４】
以上のように、この発明の実施の形態７に従えば、エントリシリアルに演算を行なう演算器群において複数段の演算器群を配置しており、高速に、複雑な演算処理を実現することができる。
【０１８５】
なお、このスイッチ回路ＳＫ０－ＳＫ２，…の接続経路は、先の実施の形態６と同様、コントローラ２１からの経路設定情報に続いてその経路が指定されて設定される。これらのスイッチ回路ＳＫ０－ＳＫ２，…は、ＦＰＧＡを利用するクロスバースイッチ回路またはスイッチマトリクスで構成されてもよい。
【０１８６】
［実施の形態８］
図２４は、この発明の実施の形態８に従う主演算回路２０の要部の構成を概略的に示す図である。この図２４に示す構成においても、メモリマット３０の下部に配置されるエントリシリアルかつビットパラレルで演算を行なう演算器群の構成を概略的に示す。この図２４に示す構成においては、複数段の演算器群ＯＧ１－ＯＧｋが配置され、これらのメモリマット３０および演算器群ＯＧ１－ＯＧｋの間に、スイッチ回路ＳＫ０－ＳＫ２，…が配置されて、データ転送経路が選択的に確立される。
【０１８７】
演算器群ＯＧ１－ＯＧｋにおいて、演算処理ユニットの処理データビット幅を再構成可能に設定する。図２４において、演算器群ＯＧ１において、ＡＬＵ１１０が配置され、演算器群ＯＧ２は、演算器群ＯＧ１のＡＬＵ１１０のビット幅の４倍のビット幅を有するＡＬＵ１１２が構成され、演算器群ＯＧｋにおいては、演算器群ＯＧ１のＡＬＵ１１０のビット幅の２倍のビット幅を有するＡＬＵ１１４が構成される。
【０１８８】
演算器群ＯＧ１－ＯＧｋそれぞれにおいて、ＡＬＵの処理ビット幅を変更可能に設定することにより、演算処理などを実行して、データの有効ビット幅が変化した場合において

も、容易に対応することができる。これらの演算器群ＯＧ１－ＯＧｋの各ＡＬＵのビット幅は、図１に示すコントローラ２１からの構成情報に基づいて設定され、またスイッチ回路ＳＫ０－ＳＫ２，…も、その接続経路が、コントローラからの接続情報に基づいて経路が設定される。

【０１８９】

図２５は、ビット幅が再構成可能なＡＬＵの構成の一例を概略的に示す図である。この図２５に示す構成においては、演算器としては、８ビットデータの処理、１６ビットデータの処理、および３２ビットのデータの加算を行なう２項加算回路の構成が一例として示される。

【０１９０】

図２５において、４つの８ビット加算器１２０ａ－１２０ｄが配置される。これらの８ビット加算器１２０ａ－１２０ｄは、それぞれ、キャリー入力Ｃｉ、および２項入力ＩＮ１およびＩＮ２と、サム出力Ｓおよびキャリー出力Ｃを含む。８ビット加算器１２０ａには、キャリー入力Ｃｉにビット“０”が与えられ、また入力ＩＮ１およびＩＮ２に、８ビットオペランドデータＯＰ１およびＯＰ２が与えられる。加算器１２０ｂへは、８ビットオペランドデータＯＰ３およびＯＰ４が与えられ、加算器１２０ｃには、８ビットオペランドデータＯＰ５およびＯＰ６が与えられ、加算器１２０ｄには、オペランドデータＯＰ７およびＯＰ８が入力ＩＮ１およびＩＮ２にそれぞれ与えられる。

【０１９１】

加算器１２０ｂのキャリー入力Ｃｉに対しては、加算器１２０ａのキャリー出力Ｃとビット“０”の一方を選択するセレクタ１２２ａが設けられ、加算器１２０ｃのキャリー入力Ｃｉに対しては、加算器１２０ｂのキャリー出力Ｃとビット“０”の一方を選択するセレクタ１２２ｂが設けられ、加算器１２０ｄのキャリー入力Ｃｉに対しては、加算器１２０ｃのキャリー出力とビット“０”の一方を選択するセレクタ１２２ｃが設けられる。

【０１９２】

セレクタ１２２ａは、ｘ８ビット構成指示信号Ｘ８の活性化時、固定ビット“０”を選択し、それ以外では、加算器１２０ａのキャリー出力Ｃを選択する。セレクタ１２２ｂは、３２ビットワード構成を指定する×３２ビット指示信号Ｘ３２の活性化時、加算器１２０ｂのキャリー出力を選択し、それ以外では、固定ビット“０”を選択する。セレクタ１２２ｃは、８ビットワード構成が指定されたとき、×８ビット指示信号Ｘ８に従って固定ビット“０”を選択し、それ以外では、加算器１２０ｃの出力信号を選択する。

【０１９３】

これらの加算器１２０ａ－１２０ｄに対し、信号Ｘ８、Ｘ１６およびＸ３２をそれぞれビット幅選択信号として受けるビット幅選択スイッチ回路１２４が設けられる。

【０１９４】

データ処理が、×８ビット構成で行なわれる場合には、セレクタ１２２ａ－１２２ｃは、それぞれ、固定ビット“０”を選択し、ビット幅選択スイッチ回路１２４は、それぞれ加算器１２０ａ－１２０ｄから出力される８ビットのサム出力Ｓおよび１ビットのキャリー出Ｃ力を選択して並列に出力する。加算器１２０ａ－１２０ｄは、したがって、キャリー入力Ｃｉには固定ビット“０”が与えられるため、それぞれ対応のオペランドデータに基づいて、２項加算処理を実行する。

【０１９５】

×１６ビット構成の場合、セレクタ１２２ａが、加算器１２０ａのキャリー出力Ｃを選択し、またセレクタ１２２ｃが、加算器１２０ｃのキャリー出力Ｃを選択する。セレクタ１２２ｂは、この状態においても、固定ビット“０”を選択する。したがって、加算器１２０ａおよび１２０ｂが、１６ビット加算器として動作し、また加算器１２０ｃおよび１２０ｄが、１６ビット加算回路として動作する。ビット幅選択スイッチ回路１２４は、この場合、１６ビットデータワード構成を指定する信号Ｘ１６に従って、加算器１２０ｂおよび１２０ｄのキャリー出力を選択し、かつサム出力Ｓとして、各加算器１２０ａ－１２０ｄのサム出力を選択する。この場合、ビット幅選択スイッチ回路１２４が以下のように

構成されてもよい。加算器１２０ａおよび１２０ｃのキャリー出力が、次段の演算器群のＡＬＵで利用されないため、そのスイッチ回路１２４の出力経路を、１ビット下位ビット方向にシフトして、加算器１２０ｂおよび１２０ｄの出力信号を生成し、それぞれ、加算器１２０ａおよび１２０ｂに対して、入力ビットデータと１ビットのキャリーの１７ビットデータを出力し、また加算器１２０ｃおよび１２０ｄに対しても、同様、１６ビットのサム出力および１ビットのキャリー出力とで構成される１７ビットデータを出力する。

【０１９６】

×３２ビット構成のデータの処理を行なう場合には、セレクタ１２２ａおよび１２２ｃが、それぞれ加算器１２０ａおよび１２０ｃのキャリー出力を選択する。セレクタ１２２ｂが、また、加算器１２０ｂのキャリー出力を選択する。したがって、これらの加算器１２０ａ－１２０ｄが接続されて、３２ビット加算回路が実現される。ビット幅選択スイッチ回路１２４は、×３２ビットデータ構造を指示する信号×３２に従って、加算器１２０ａ－１２０ｄのそれぞれの８ビットサム出力と、加算器１２０ｄのキャリー出力を選択し、３２ビットデータで構成されるサム出力Ｓおよび１ビットのキャリー出力Ｃを生成する。

【０１９７】

図２５に示すようなビット幅選択スイッチ回路１２４を利用することにより、基本単位として８ビットデータを演算する加算回路１２０ａ－１２０ｄを利用して、×１６ビットデータおよび３２ビットデータの加算を実行することができる。６４ビットデータの場合、この図２５に示す構成をさらに縦続接続する。

【０１９８】

なお、この図２５に示す加算器の構成においては、セレクタ１２２ａ－１２２ｃにより、キャリー出力が選択的に伝搬されており、リップルキャリー加算器が実現され、キャリー伝搬により加算時間が長くなることが考えられる。この場合、キャリールックアヘッド方式またはキャリーセーブ加算方式が利用されてもよい。

【０１９９】

以上のように、この発明の実施の形態８に従えば、エントリシリアルで演算を行なう場合、演算器を複数段配置し、かつ各段の演算器の処理データビット幅をリコンフィギアラブルに構成しており、データビット幅および演算処理内容にかかわらず、高速で、必要な演算処理を実現することができる。

【０２００】

［実施の形態９］

図２６は、この発明に従う半導体装置を利用する処理システムの構成の一例を示す図である。図２６において、図１に示す構成と同様、システムバス５に、ホストＣＰＵ（中央演算処理装置）２、ＤＭＡ回路４およびメモリ３が接続される。このシステムバス５に対し、さらに、この発明に従う半導体演算装置１が接続される。この半導体演算装置１内において、図１に示すように、制御ＣＰＵ（２５）を主要構成要素とする集中制御ユニット１５が設けられる。この半導体演算装置１は、システムバス５に対し並列に複数個設けられてもよい。この図２６に示す処理システムの構成の場合、ホストＣＰＵ２が、メモリ３に格納されるデータを利用して必要な処理を実行する。画像データ処理などの大量のデータに対する処理が必要な場合には、この発明に従う半導体演算装置１が、データの処理を担当する。すなわち、システム構成を、ホストＣＰＵ２および半導体演算装置１内の集中制御ユニット１５の階層ＣＰＵ構成とすることにより、高速に処理を実行することができる。

【０２０１】

［変更例１］

図２７は、この発明に従う半導体装置１を利用する処理システムの変更例１のシステム構築例を示す図である。この図２７に示す処理システムにおいては、図２６に示す処理システムと同様、システムバス５を介して、半導体演算装置１、ホストＣＰＵ２、メモリ３、およびＤＭＡ回路４が接続される。この半導体演算装置１内においては、図１に示すよ

うに基本演算ブロック（ＦＢ１－ＦＢｎ）が並列に配設され、各基本演算ブロックの主演算回路（２０）内においては、メモリマット３０が配置される。したがって、これらの基本演算ブロック内のメモリマット３０を、画像データを格納するフレームメモリとして利用させることにより、この半導体演算装置１を、メモリマクロとして動作させることができる。したがって、画像データ処理のワーキングメモリとしてこの半導体演算装置１を利用することができ、またフレームバッファとして、この半導体演算装置１を利用することができる。

【０２０２】

また、この半導体演算装置１内においては、このメモリマット３０がＳＲＡＭセルで構成されており、高速のメモリが実現される場合、メモリマット３０をキャッシュメモリとして利用し、メモリ３を主記憶として利用することにより、高速のデータ処理システムを構築することができる。

【０２０３】

［変更例２］

図２８は、主演算回路２０に含まれるビットシリアルかつエントリパラレルの演算を行なう演算器（ＡＬＵ）３４の構成の一例を概略的に示す図である。図２８において、ＡＬＵ３４は、ＡＮＤゲート１３２と、ＮＯＴゲート１３４と、およびＥＸＯＲゲート１３６と、演算処理内容を設定するデータを格納するレジスタ回路１３０と、レジスタ回路１３０の出力信号に従ってＡレジスタおよびＸレジスタとこれらのゲート１３２、１３４、…１３６との間の接続経路を設定する選択回路１３８と、レジスタ回路１３０の格納データに従ってこれらのゲート回路１３２－１３６の出力を、ＣレジスタおよびＸレジスタへ選択的に結合する選択回路１３９を含む。

【０２０４】

１ビット乗算を行なう場合には、ＡＮＤゲート１３２が利用され、加算動作を行なう場合には、ＡＮＤゲート１３２およびＥＸＯＲゲート１３６を利用する。比較演算操作を行なう場合には、ＥＸＯＲゲート１３６を利用する。減算を実行する場合、ＮＯＴゲート１３４を利用し、その後、２の補数表示の加算を実行する。

【０２０５】

このＡＬＵ３４の演算処理内容を、レジスタ回路１３０の格納データにより設定する。レジスタ回路１３０の格納データは、図１に示すマイクロプログラム格納メモリ２３に格納されるプログラム命令に従ってコントローラ２１が設定する。したがって、ＡＬＵ３４の個々の演算処理内容は、プログラマブルである。したがって、図１に示すように、基本演算ブロックＦＢ１－ＦＢｎが複数個設けられている場合、この半導体演算装置１を、種々の論理回路を実現するプログラマブルロジック回路として利用することができる。この場合、プログラムデータを、システム起動時または半導体演算装置１の動作時にロードすることにより、その演算処理内容を設定することができる。

【０２０６】

この演算器（ＡＬＵ３４）の演算処理内容が、レジスタ回路１３０に格納されるデータに応じて変更される構成は、エントリシリアルかつビットパラレルで演算処理を行なうＡＬＵ８４等においても同様、適用することができる。その場合には、加算回路、乗算回路、および比較回路等の演算回路が選択される。

【０２０７】

［変更例３］

図２９は、この発明の実施の形態９の変更例３に従う処理システムの構成を概略的に示す図である。この図２９に示す処理システムにおいても、システムバス５に、ＣＰＵ２、メモリ３、ＤＭＡ回路４および半導体演算装置１が接続される。この半導体演算装置１内においては、図１に示すように複数の基本演算ブロックＦＢ１－ＦＢｎが並列に設けられ、各基本演算ブロックＦＢ１－ＦＢｎ内に、メモリマットおよびＡＬＵ群が配置される主演算回路が設けられる。このＡＬＵ群の演算処理内容は、先の図２８に示すように、プログラマブルである。したがって、これらの基本演算ブロックＦＢ１－ＦＢｎにおいては、

互いに独立に、その内部に含まれるコントローラによりマイクロプログラム格納メモリ（２３）に格納されたマイクロプログラムに従って処理が実行され、それらの処理内容は互いに独立に設定することができる。したがって、これらの基本演算ブロックＦＢ１－ＦＢｎを、完全に同一機能を有する演算ブロックとして取扱うことができ、また、一部をメモリ、一部をプログラマブルロジック回路および一部を高速演算処理回路（並列演算処理実行による高速演算処理装置）として利用することができる。これにより、処理システムにおける演算処理内容に応じて、種々の演算処理を並列に実行する並列演算装置を実現することができ、高速かつ高性能の処理システムを構築することができる。

【０２０８】

以上のように、この発明の実施の形態９に従えば、この発明に従う半導体装置を用いて処理システムを構築することにより、演算データのビット幅の制限もなく、非常に柔軟に、データ処理形態をダイナミックに変化させて演算処理を行なうことができるとともに、種々の階層ＣＰＵシステム、階層メモリシステムおよびコプロセッサシステムを柔軟に構築することができる。

【０２０９】

［実施の形態１０］

一般に、ＲＡＭ（ランダム・アクセス・メモリ）においては、ウェハプロセスでメモリマット内に不良が発生した場合には、予め準備された冗長ビットと不良ビットとを置換することにより、不良ビットを等価的に救済して良品ＲＡＭとして用いる不良救済回路技術が一般的に用いられる。本発明においても、主演算回路は、大部分がメモリセルで構成されるため、この不良救済技術を用いて、製品歩留りを向上させることが可能となる。以下、この構成について説明する。

【０２１０】

図３０は、この発明の実施の形態１０に従う主演算回路２０の要部の構成を概略的に示す図である。この図３０に示す主演算回路２０は、図２１に示す主演算回路２０と同様の構成を備える。しかしながら、本実施の形態１０における主演算回路２０の構成としては、他の実施の形態における主演算回路の構成であっても同様適用することができる。

【０２１１】

図３０に示す主演算回路２０は、以下の点で、図２１に示す主演算回路２０とその構成が異なる。すなわち、ロウデコーダＢ７６とメモリマット３０のワード線ＷＬＢとの間に左側冗長救済回路１４２が設けられ、ロウデコーダＡ６６とワード線ＷＬＡとの間に上側冗長救済回路１４４が設けられる。ビット線対ＢＬＰＡと演算処理ユニット（ＡＬＵ）群３２の間に、右側冗長救済回路１４６が設けられ、ビット線対ＢＬＰＢと演算器群（ＡＬＵ群Ｂ）８２の間に、下側冗長救済回路が設けられる。

【０２１２】

これらの冗長救済回路１４２、１４４、１４６および１４８の救済態様を設定するために、図１に示すヒューズ２４からのヒューズ情報をデコードして、ヒューズデコード情報Ｘ，Ｙを生成するヒューズデコード回路１４０が設けられる。冗長救済回路１４２および１４６に対し同じヒューズデコード情報Ｘが与えられ、冗長救済回路１４４および１４８に対し共通のヒューズデコード情報Ｙが与えられる。ワード線ＷＬＡの不良救済時においては、このワード線ＷＬＡに接続されるメモリセルが接続するビット線対ＢＬＰＢについても、不良救済を行なう必要があるためであり、同様、ワード線ＷＬＢが不良救済を行なう必要がある場合、ビット線対ＢＬＰＡの不良救済を行なう必要があるためである。

【０２１３】

ワード線に対する冗長救済回路１４２および１４４は、テスト時において、各種のヒューズ素子を用いて予めプログラムされた不良アドレスを回避して、メモリマット３０にアクセスするように動作する。すなわち、これらの冗長救済回路１４２および１４４は、いわゆる「シフトリダンダンシ」方式に従って不良救済を行なう。

【０２１４】

ＡＬＵ群３２および８２に対しても、冗長救済回路１４６および１４８がそれぞれ配置

されるのは以下の理由による。ロウデコーダ７６および／または６６においてワード線の不良救済が行なわれる場合、同様、これらのビット線対ＢＬＰＡおよびＢＬＰＢにおいても連動して、不良救済を行なう必要がある。このビット線対に対する冗長救済回路１４６および１４８を配置することにより、演算処理ユニット群（ＡＬＵ群）３２および演算器群（ＡＬＵ群Ｂ）８２それぞれにおいて、メモリマット３０における不良置換の有無にかかわりなく、正常に、正常メモリセルに格納されたデータを用いて演算処理を実行することができる。

【０２１５】

不良アドレスのプログラムは、ウェハテスト時に、メモリマット３０に対するデータの読出および書込動作の試験を行なった後、ヒューズ用の溶断可能なメタル線を、レーザ等のエネルギー線を用いて切断することにより行なわれる。これらの不良アドレスプログラム用のヒューズは、図１に示す基本演算ブロックＦＢ１－ＦＢｎ内にそれぞれヒューズ２４として配置されている。このヒューズ情報は、図３０に示すヒューズデコード回路１４０を用いて、ヒューズデコード情報Ｘ，Ｙに変換される。このヒューズデコード情報Ｘ，Ｙが、たとえばチップ起動時などに各冗長救済回路へ転送され、不良救済処理を実現する。

【０２１６】

図３１は、ロウデコーダに対して設けられる冗長救済回路の構成の一例を示す図である。図３１においては、メモリマット３０におけるワード線ＷＬｎ－ＷＬ（ｎ＋３）を代表的に示す。この図３１に示すワード線ＷＬは、ワード線ＷＬＡまたはＷＬＢである。これらのワード線ＷＬｎ－ＷＬ（ｎ＋３）をアドレス入力に従って選択状態へ駆動するために、ワード線デコード回路１５０が設けられる。このワード線デコード回路１５０は、図３０に示すロウデコーダＡ６６またはロウデコーダＢ７６に対応する。

【０２１７】

ワード線デコード回路１５０の出力ＷＯｎ－ＷＯ（ｎ＋２）に対して、それぞれ、ヒューズデコード情報レジスタ１５５ｎ－１５５（ｎ＋２）が設けられる。これらのヒューズデコード情報レジスタ１５５ｎ－１５５（ｎ＋２）は、シフトレジスタ回路またスキャンパスを構成し、ヒューズデコード回路１４０（図３０参照）により生成されたヒューズデコード情報を、順次シフトして対応のワード線に対するヒューズデコード情報を格納する。

【０２１８】

また、ワード線デコード回路１５０の出力ＷＯｎ－ＷＯ（ｎ＋２）それぞれに対応して、ヒューズデコード情報レジスタ１５５ｎ－１５５（ｎ＋２）の格納データに従ってワード線デコード回路１５０の出力ＷＯｎ－ＷＯ（ｎ＋２）の転送経路を切換えるシフト切換用マルチプレクサ１６０ｎ－１６０（ｎ＋２）が配置される。これらのシフト切換用マルチプレクサ１６０ｎ－１６０（ｎ＋２）は、対応のヒューズデコード情報レジスタの格納データが“０”のときには、ワード線デコード回路の対応の出力信号を対応のワード線に伝達し、一方、対応のヒューズデコード情報レジスタの格納データが“１”のときには、図の上側方向（ワード線番号の大きい方）にシフトして、ワード線デコード回路の出力信号を伝達する。

【０２１９】

今、図３１に示すように、ヒューズデコード情報レジスタ１５５ｎに、ビット“０”が格納され、ヒューズデコード情報レジスタ１５５（ｎ＋１）および１５５（ｎ＋２）にビット“１”が格納されている状態を考える。この場合、シフト切換用マルチプレクサ１６０ｎは、ヒューズデコード情報レジスタ１５５ｎの格納ビット“０”に従って、ワード線デコード回路１５０の出力信号ＷＯｎを、対応のワード線ＷＬｎへ伝達する。このワード線ＷＬｎよりも番号の少ないワード線には、したがって、ワード線デコード回路１５０の出力信号がシフトされることなく転送される。

【０２２０】

一方、ヒューズデコード情報レジスタ１５５（ｎ＋１）および１５５（ｎ＋２）にはビ

ット“１”が選択格納されているため、シフト切換用マルチプレクサ１６０（ｎ＋１）および１６０（ｎ＋２）は、それぞれワード線デコード回路１５０の出力信号ＷＯ（ｎ＋１）およびＷＯ（ｎ＋２）をワード線ＷＬ（ｎ＋２）およびＷＬ（ｎ＋３）へ伝達する。したがって、ワード線ＷＬ（ｎ＋１）は、ワード線デコード回路１５０の出力から分離されており、このワード線ＷＬ（ｎ＋１）は、常時非活性状態に維持される。これにより、不良ワード線ＷＬ（ｎ＋１）を、常時非選択状態に維持することができ、不良アドレスを回避する不良救済を実現することができる。

【０２２１】

なお、いうまでもなく、メモリマット３０においては、シフトリダンダンシ方式に従って不良救済が行なわれるため、このメモリマット３０のアドレス空間（エントリ数）よりも多い数のワード線を設けることが要求される。

【０２２２】

上述のように、ワード線ＷＬｎまでは、順次ワード線デコード回路１５０の出力信号に従って選択状態へ駆動される。不良ワード線に対するレジスタ回路およびそれより上位のレジスタ回路の格納データを“１”に設定して転送経路をシフトさせることにより、ワード線デコード回路１５０の出力信号ＷＯ（ｎ＋１）に従って、ワード線ＷＬ（ｎ＋２）が選択される。以降、ワード線とワード線デコード回路１５０の出力ＷＯの対応関係が１つシフトされて、順次メモリマット３０の正常ワード線が選択状態へ駆動される。

【０２２３】

すなわち、不良ワード線およびそれ以降に対応するヒューズデコード情報レジスタにビット“１”を格納することにより、不良ワード線とワード線デコード回路１５０とを分離することができ、不良アドレスが選択されるのを防止することができる。

【０２２４】

図３２は、センスアンプ群およびライトドライバ群に対して設けられる冗長救済回路（１４６，１４８）の構成を概略的に示す図である。図３２において、メモリマット３０のビット線対ＢＬＰｎ－ＢＬＰ（ｎ＋３）に対して設けられる冗長救済回路の構成を代表的に示す。これらのビット線対ＢＬＰｎ－ＢＬＰ（ｎ＋３）は、ビット線対ＢＬＰＡｎ－ＢＬＰＡ（ｎ＋３）またはＢＬＰＢｎ－ＢＬＰＢ（ｎ＋３）のいずれかである。

【０２２５】

ビット線対ＢＬＰｎ－ＢＬＰ（ｎ＋３）それぞれに対応して、センスアンプ・ライトドライバ１７２ｎ－１７２（ｎ＋３）が配置される。これらのセンスアンプ・ライトドライバ１７２ｎ－１７２（ｎ＋３）の各々は、対応のセンスアンプ群およびライトドライバ群に含まれるセンスアンプおよびライトドライバで構成される。

【０２２６】

ビット線対ＢＬＰｎ－ＢＬＰ（ｎ＋２）それぞれに対応して、単位ＡＬＵ回路ブロック１７０ｎ－１７０（ｎ＋２）が設けられる。単位ＡＬＵ回路ブロック１７０ｎ－１７０（ｎ＋２）の各々は、ＡＬＵ３４または、演算器群（ＡＬＵ群Ｂ）８２に含まれる単位ＡＬＵ（１ビット演算を行なう回路）に対応する。

【０２２７】

ＢＬＰ冗長救済回路（１４６，１４８）においては、ビット線対ＢＬＰｎ－ＢＬＰ（ｎ＋２）に対応して、ヒューズデコード情報を格納するヒューズデコード情報レジスタ１８０ｎ－１８０（ｎ＋２）が設けられる。これらのヒューズデコード情報レジスタ１８０ｎ－１８０（ｎ＋２）は、先の図３０に示すヒューズデコード回路１４０から生成される。これらのヒューズデコード情報レジスタ１８０ｎ－１８０（ｎ＋２）は、シフトレジスタ回路を構成し、順次ヒューズデコード情報をシフト動作により転送して、対応のビット線対に対するヒューズデコード情報を格納する。

【０２２８】

これらのヒューズデコード情報レジスタ１８０ｎ－１８０（ｎ＋２）それぞれに対応して、シフト切換用マルチプレクサ１８２ｎ－１８２（ｎ＋２）が設けられる。これらのシフト切換用マルチプレクサ１８２ｎ－１８２（ｎ＋２）は、それぞれ、対応のビット線対

に配置されるセンスアンプ・ライトドライバ１７２ｎ－１７２（ｎ＋２）と、１列上側方向にシフトしたセンスアンプ・ライトドライバ１７２（ｎ＋１）－１７２（ｎ＋３）とに結合される。

【０２２９】

これらのシフト切換用マルチプレクサ１８２ｎ－１８２（ｎ＋２）は、それぞれ、対応のヒューズデコード情報レジスタ１８０ｎ－１８０（ｎ＋２）の格納データがビット“０”のときには、対応のビット線ＢＬＰを対応の単位ＡＬＵ回路ブロック１７０に接続し、ビット“１”が格納されている場合には、対応のビット線ＢＬＰを１列上位側にシフトした単位ＡＬＵ回路ブロック１７０に接続する。

【０２３０】

今、図３２に示すように、ヒューズデコード情報レジスタ１８０ｎにビット“０”が格納され、ヒューズデコード情報レジスタ１８０（ｎ＋１）および１８０（ｎ＋２）にビット“１”が格納されている状態を考える。この状態においては、シフト切換用マルチプレクサ１８２ｎは、ビット線対ＢＬＰｎに対して設けられたセンスアンプ・ライトドライバ１７２ｎを対応の単位ＡＬＵ回路ブロック１７０ｎに結合する。一方、シフト切換用マルチプレクサ１８０（ｎ＋１）は、ビット線対ＢＬＰ（ｎ＋２）に対して設けられたセンスアンプ・ライトドライバ１７２（ｎ＋２）を単位ＡＬＵ回路ブロック１７０（ｎ＋１）に結合し、同様、シフト切換用マルチプレクサ１７２（ｎ＋２）は、ビット線対ＢＬＰ（ｎ＋３）に対して設けられたセンスアンプ・ライトドライバ１７２（ｎ＋３）を単位ＡＬＵ回路ブロック１７０（ｎ＋２）に結合する。

【０２３１】

したがって、ビット線対ＢＬＰ（ｎ＋２）に対して設けられたセンスアンプ・ライトドライバ１７２（ｎ＋１）は、対応の単位ＡＬＵ回路ブロック１７０（ｎ＋１）から分離され、いずれの単位ＡＬＵ回路ブロックにも結合されない。このビット線対ＢＬＰ（ｎ＋１）は、不良ワード線ＷＬ（ｎ＋１）に対応する。したがって、この不良ワード線の冗長置換に連動して、ビット線対の置換を行なうことにより、正確に、正常なメモリセルのみを利用して、単位ＡＬＵ回路ブロック１７０において演算処理を行なうことができる。

【０２３２】

以上のように、この発明の実施の形態１０に従えば、不良冗長置換を行なうことにより、不良セルの救済を行なうことができ、正確な、演算処理を行なう装置が実現され、歩留りが改善される。

【０２３３】

また、メモリマットにおいて、ワード線が直交方向に配列され、またビット線対も直交して配置される構成の場合、不良ワード線の冗長置換に連動して、不良ワード線に対応するビット線対も冗長置換を同様にして行なうことにより、確実に、正常にデータを記憶するメモリセルを用いて演算処理を行なうことができ、装置の信頼性を確保することができる。

【０２３４】

［実施の形態１１］

図３３は、この発明の実施の形態１１に従う基本演算ブロックＦＢｉの要部の構成を概略的に示す図である。図３３において、メモリセルマット３０はエントリＥＲＹとして、番号０からＭＡＸ＿ＥＮＴＲＹが付されたエントリを含む。各エントリは、ビット位置として０からＢＩＴ＿ＭＡＸを有し、そのビット幅が、ＢＩＴ＿ＭＡＸ＋１である。

【０２３５】

演算処理ユニット群（ＡＬＵ群）３２においては、各エントリに対応して演算処理ユニット（以下、適宜ＡＬＵユニットと称す）３４が配置される。この演算処理ユニット群３２に対して、ＡＬＵ間相互接続用スイッチ回路４４が設けられる。

【０２３６】

この主演算回路２０の動作は、プログラム格納メモリ２３に格納されるプログラム（マイクロプログラム）により設定される。コントローラ２１が、このプログラム格納メモリ

２３に格納されたプログラムに従って処理を実行する。
【０２３７】
先の実施の形態１においては、プログラム格納メモリ２３において、マイクロプログラムが格納される。本実施の形態１１においては、このプログラム格納メモリ２３に格納されるプログラム命令は、マイクロ命令でなくてもよく、マクロ命令であってもよい。コントローラ２１が、プログラム命令をデコードし、この命令により指定された動作に必要な処理を実行する。
【０２３８】
レジスタ群２２においては、ポインタレジスタｒ０－ｒ３が設けられ、演算対象のデータのメモリセルマット３０のアドレスが、これらのポインタレジスタｒ０－ｒ３に格納される。コントローラ２１は、このポインタレジスタｒ０－ｒ３に格納されるポインタに従って主演算回路２０におけるエントリまたはエントリ内位置を指定するアドレスを生成して、メモリセルマット３０と演算処理ユニット群３２との間のデータの転送（ロード／ストア）を制御し、また、ＡＬＵユニット３４間の接続指定情報を設定する。
【０２３９】
図３４は、図３３に示す演算処理ユニット３４の構成を概略的に示す図である。図３４において、ＡＬＵ３４においては、内部データ線２００を介してＸレジスタ５４が、ライトドライバ６０およびセンスアンプ６２に結合される。この内部データ線２００は算術演算論理回路５０に結合される。
【０２４０】
この図３４に示す単位ＡＬＵ回路ブロック（ＡＬＵユニット３４）においては、先の図７に示す構成と異なり、Ａレジスタは設けられない。Ｘレジスタ５４が、対応のエントリのメモリセルからのロードデータの一時保存を行ない、かつ算術演算論理回路５０の演算途中の結果の一時保存を行なう。２項演算処理時において、Ｘレジスタ５４に第１の演算データが格納されたとき、次の（別の）演算データは算術演算論理回路５０に直接与えられて演算処理が実行される。
【０２４１】
Ｘレジスタ５４が、ＡＬＵ間接続回路６５を介して他の単位ＡＬＵ回路ブロック（ＡＬＵユニット）に結合され、異なるＡＬＵユニット間でデータ転送を行なうことができる。
【０２４２】
図３４に示すＡＬＵユニット３４の他の構成は、図７に示す単位ＡＬＵ回路ブロック３４の構成と同じであり、対応する部分には同一の参照番号を付し、その詳細説明は省略する。
【０２４３】
図３５は、図３３に示すポインタレジスタｒ０－ｒ３に対する操作命令（レジスタ命令）を一覧にして示す図である。レジスタ命令として、５種類の命令が準備される。
【０２４４】
命令“ｒｅｇ．ｓｅｔ　ｎ，ｒｘ”は、レジスタｒｘに、定数ｎをセットする命令である。定数ｎは、１つのエントリにおけるビット位置を示すものであり、１エントリのビット０からＭＡＸ＿ＢＩＴのいずれかの値を規定する。
【０２４５】
命令“ｒｅｇ．ｃｐｙ　ｒｘ，ｒｙ”は、ポインタレジスタｒｘの内容を、ポインタレジスタｒｙにコピーする命令である。
【０２４６】
命令“ｒｅｇ．ｉｎｃ　ｒｘ”は、ポインタレジスタｒｘの格納値を１増分する命令である。
【０２４７】
命令“ｒｅｇ．ｄｅｃ　ｒｘ”は、ポインタレジスタｒｘの格納値を１減分する命令である。
【０２４８】

命令“ｒｅｇ．ｓｆｔ　ｒｘ”は、ポインタレジスタｒｘの格納値を１ビット左シフトする命令である。

【０２４９】

これらの５種類のレジスタ命令により、ポインタレジスタｒ０－ｒ３の格納値（ポインタ）を操作して、メモリセルマットの演算対象データのアドレスを指定する。

【０２５０】

図３６は、図３４に示すＡＬＵ３４に対する操作命令を一覧にして示す図である。以下、図３６を参照して、各ＡＬＵ命令の操作内容について簡単に説明する。

【０２５１】

命令“ａｌｕ．ｓｅｔ．＃”は、レジスタ＃（Ｘ、ＣまたはＭ）に“１”をセットする命令である。このＡＬＵセット命令は、エントリ単位でレジスタのセットを指定する。

【０２５２】

命令“ａｌｕ．ｃｌｒ．＃”は、レジスタ＃の格納値を“０”にクリアする命令である。

【０２５３】

命令“ａｌｕ．ｃｐｙ．＃１＃２”は、レジスタ＃１の格納値をレジスタ＃２へコピーする命令である。

【０２５４】

このａｌｕコピー命令が実行されると、各エントリに対して設けられたＡＬＵ内でレジスタ間データ転送が実行される。

【０２５５】

図３７は、メモリセルマットとＡＬＵとの間のデータ転送を規定するＡＬＵ命令のロード／ストア命令を示す図である。

【０２５６】

命令“ｍｅｍ．ｌｄ@ｒｘ”は、ポインタレジスタｒｘの示すメモリセル位置からＸレジスタへデータをロードする命令である。

【０２５７】

命令“ｍｅｍ．ｓｔ@ｒｘ”は、Ｍレジスタ（マスクレジスタ５８）にビット“１”が設定されている場合には、Ｘレジスタに格納されたデータを、ポインタレジスタｒｘが指定するアドレス位置へ格納する命令である。

【０２５８】

このメモリロード／ストア命令を利用することにより、ポインタレジスタｒｘの格納値をアドレスとして、メモリセルとＡＬＵユニットとの間でデータ転送を行なうことができる。

【０２５９】

図３８は、ＡＬＵ命令のうち、エントリ間のデータ移動（Ｍｏｖｅ）を行なう命令を一覧にして示す図である。

【０２６０】

命令“ｅｃｍ．ｍｖ．ｎ＃ｎ”は、データ移動命令（ｍｏｖｅ）における移動量を数値＃ｎで規定する。したがって、この命令では、Ｘレジスタ３のデータ転送において、エントリｊ＋ｎのＸレジスタの格納値が、エントリｊのＸレジスタに移動される。エントリ移動量ｎは、０から１２８の範囲の整数値を取り、最大１２８ビット離れた位置のエントリ間でデータ移動（Ｍｏｖｅ）を行うことができる。ただし、ＥＮＴＲＹ＿ＭＡＸは、１２８以上である。

【０２６１】

命令“ｅｃｍ．ｍｖ．ｒ　ｒｘ”は、ポインタレジスタｒｘに格納された値だけエントリ間をデータ移動させる命令であり、この命令が実行されると、エントリｊ＋ｒｘのＸレジスタの格納値を、エントリｊのＸレジスタに転送する。

【０２６２】

図３５から図３８に一覧して示す命令を利用することにより、ＡＬＵユニットに、所望

のエントリのデータを設定することができる。
【０２６３】
図３９は、各単位ＡＬＵ回路ブロック（ＡＬＵユニット）で行なわれる演算を指定する命令を示す図である。
【０２６４】
命令“ａｌｕ．ｏｐ．ａｄｃ@ｒｘ”は、ポインタレジスタｒｘが指定するメモリセルアドレスのデータとＸレジスタに格納されたデータとを加算し、その加算結果をＸレジスタに格納することを指定する命令である。加算演算時、全加算演算が行なわれるため、キャリ発生時、Ｃレジスタにキャリが格納される。Ｘレジスタ（Ｘｊ）には、ポインタレジスタｒｘが指定するアドレスのメモリセルデータＡｊ［ｒｘ］とＸレジスタに格納されたビット値ＸｊとＣレジスタに格納されたキャリＣｊの排他的論理和（“＾”）演算によりサムＳｕｍが生成されて、Ｘレジスタに格納される。
【０２６５】
キャリＣｊは、メモリセルデータＡｊ［ｒｘ］とＸレジスタの格納ビットＸｊとＣレジスタの格納値Ｃｊのビットの各ビットの組のＡＮＤ演算（＆）の論理和（＋）により求められる。
【０２６６】
この加算命令は、マスクレジスタ（ＭレジスタＭｊ）に“１”が設定されたときに実行され、マスクレジスタに“０”が設定されている場合には、このエントリにおいて加算命令は実行されない。
【０２６７】
命令“ａｌｕ．ｏｐ．ｓｂｂ@ｒｘ”は、減算命令であり、この減算命令実行時、ポインタレジスタｒｘが指定するメモリアドレスのデータＡｊ［ｒｘ］からＸレジスタに格納されたビット値を減算する。減算結果がＸレジスタに格納され、Ｃレジスタにはボローが格納される。
【０２６８】
この減算時においては、Ｘレジスタに格納されたビットＸｊの反転値！Ｘｊが用いられ、加算時と同様の処理が行なわれる。したがって、この減算命令が与えられた場合には、Ｘレジスタに格納された値が反転されて加算器へ与えられる（最下位ビットのキャリが１にセットされる）。
【０２６９】
図４０は、ＡＬＵ内で行なわれる論理演算を指定する命令を一覧にして示す図である。
命令“ａｌｕ．ｏｐ．ａｎｄ@ｒｘ”は、ＡＮＤ命令であり、この命令実行時、ポインタレジスタｒｘのポインタが指定するメモリアドレスのデータＡｊ［ｒｘ］とＸレジスタに格納されたビット値Ｘｊの論理積（ＡＮＤ）をとり、その論理積結果がＸレジスタに格納される。但し、マスクレジスタＭｊの格納値（Ｍｊで示す）が“０”の場合には、このＡＮＤ命令は実行されない。以下の論理演算命令についても同様に、マスクレジスタの格納値により。指定された演算の実行／禁止が指定される。
【０２７０】
命令“ａｌｕ．ｏｐ．ｏｒ@ｒｘ”は、ポインタレジスタｒｘのポインタが指定するメモリアドレスのデータＡｊ［ｒｘ］とＸレジスタの格納ビットＸｊの論理和（ＯＲ演算）を行ない、その結果を、Ｘレジスタに格納する。
【０２７１】
命令“ａｌｕ．ｏｐ．ｅｑ@ｒｘ”は、ＥＸＯＲ命令であり、ポインタレジスタｒｘのポインタが指定するアドレスのメモリセルデータＡｊ［ｒｘ］とＸレジスタの格納ビットＸｊの値の排他的論理和演算（ＥＸＯＲ演算）が行なわれ、その演算結果が、Ｘレジスタに格納される。
【０２７２】
命令“ａｌｕ．ｏｐ．ｎｏｔ”は、ＮＯＴ命令（反転命令）であり、Ｘレジスタのビット値Ｘｊを反転し、その反転結果！ＸｊをＸレジスタに格納する。

【０２７３】
ＡＬＵ３４を、マスクレジスタ（Ｍレジスタ）５８、Ｃレジスタ５６、Ｘレジスタ５４、および算術演算論理回路５０で構成し、図３５から図４０に示す命令を組合せて演算処理を記述することにより、種々の演算処理を、ワードパラレルかつビットシリアル態様で実行することができる。
【０２７４】
図４１は、この発明の実施の形態１１に従う基本演算ブロックにおける加算演算を実行するプログラムの一例を示す図である。図４１において、行番号によりプログラム内の各演算命令の行が指定され、その行において、実行される命令が指定され、“／／”の後に、実行される演算命令の内容が説明される。この“／／”後の内容は、演算内容の説明であり、何ら実行命令ではない。以下、図４１に示す加算プログラムは、２項加算処理であり、（ａ＋ｂ）＝ｃの処理が実行される。以下、図４１に示す加算プログラムの処理動作について説明する。
【０２７５】
行番号０において、マスクレジスタ（Ｍレジスタ）に“１”が設定され、キャリレジスタ（Ｃレジスタ）の格納値が“０”にクリアされる。
【０２７６】
行番号１において、ポインタレジスタｒ０に定数ａｓが格納され、ポインタレジスタｒ１に定数ｂｓが格納され、ポインタレジスタｒ２に定数ｃｓが格納される。これらの定数ａｓ、ｂｓ、およびｃｓは、それぞれ２項加算演算（ａ＋ｂ＝ｃ）の各演算数ａ、ｂおよびｃの最下位ビットの対応のエントリ内の位置を示す。
【０２７７】
行番号２および行番号３において、加算命令が指定される。ｉが０から演算データのビット幅（ｂｉｔ＿ｃｏｕｎｔ）－１の間、繰返し加算が実行され、各加算命令実行毎に、ｉが増分される（ｉ＋＋）。ｆｏｒ文の後の中括弧で囲まれる関数の内容が、“ｆｏｒループ命令”の条件が満たされるまで、すなわちｉが演算対象数のビット幅に到達するまで、繰返し実行される。
【０２７８】
このｆｏｒ文で規定されるループ命令においては、ポインタレジスタｒ０の内容が対応のＡＬＵユニットに転送され（ロードされ）てＸレジスタに格納され、次いで、ポインタレジスタｒ１に格納されるポインタ値が示すメモリセルのデータが対応のＡＬＵユニットへ転送されてＸレジスタの格納値と加算される（Ｃレジスタの格納値と合わせて）。加算結果が、ポインタレジスタｒ２のポインタが示すアドレス位置に格納される。この命令列において“ｒ０＋”、“ｒ１＋”および“ｒ２＋”は、この命令実行後、ポインタレジスタｒ０、ｒ１およびｒ２のポインタが１増分されることを示す。
【０２７９】
行番号３において、このループ命令時に実行される命令列の末尾が示される。
このｆｏｒ｛｝のループ命令が完了し、データビット列について加算処理が完了すると、行番号４において、Ｃレジスタの格納値が、Ｘレジスタに転送され、次いで、このＸレジスタの格納値が、ポインタレジスタｒ２が指定するアドレス位置に格納される。この処理により、加算結果のキャリが格納される。
【０２８０】
図４２は、図４１に示す加算操作を概略的に示す図である。まず演算数ａ、ｂおよびｃのエントリＥＲＹの格納領域の最下位ビット位置ａｓ、ｂｓおよびｃｓが、それぞれポインタレジスタｒ０、ｒ１およびｒ２のポインタにより指定される。次いで、このポインタレジスタｒ０、ｒ１およびｒ２のポインタが示すメモリセルのデータａｉおよびｂｉが読出されて加算され、その加算結果が、ポインタレジスタｒ２が示すメモリセル位置に格納される。演算数ａおよびｂが３ビットデータの場合、ｉ＝０～２において、加算、およびストアが実行され、最終的に、Ｃレジスタの格納値がＸレジスタを介してポインタレジスタｒ２の指定するビット位置（ｃｓ＋３）に格納される。

【０２８１】
この演算命令“ａｌｕ．ｏｐ．ａｄｃ@ｒ１＋”により、このＡＬＵユニットにおいてＡＬＵ回路（算術論理演算回路）の実行内容を加算に設定することができる。
【０２８２】
図４３は、演算数ａおよびｂの減算（ａ－ｂ）を行ない、減算結果ｃを生成する減算プログラムの一例を示す図である。以下、図４３を参照して、２項減算処理について説明する。
【０２８３】
まず、行番号０において、ＭレジスタおよびＣレジスタの初期設定が、加算演算処理時と同様に行なわれる。
【０２８４】
行番号１において、加算演算時と同様に、演算数のエントリ内のアドレスの初期設定が行なわれ、ポインタレジスタｒ０、ｒ１およびｒ２に、各対象演算数ａ、ｂおよびｃの最下位ビット位置が設定される。
【０２８５】
行番号２および行番号３において、ループ演算命令が、加算演算実行プログラムと同様に指定される。命令“ａｌｕ．ｏｐ．ｓｂｂ@ｒ１＋”により、演算数ａから演算数ｂを減算する処理が実行される。ロード命令“ｍｅｍ．ｌｄ”およびストア命令“ｍｅｍ．ｓｔ”は、加算時と同様であり、演算データのＡＬＵユニットへの転送および減算結果のメモリセルマットのｃ［ｉ］への格納が実行される。
【０２８６】
行番号３においてループ演算命令の内容の末尾が指定される。
行番号４において、行番号２および３の指定するループ命令の完了後（演算数ａおよびｂの全ビットについての減算が完了後）、Ｃレジスタの内容がＸレジスタに転送され、次いで、Ｘレジスタの内容がポインタレジスタｒ２が指定するメモリ位置に格納されて、ボローが格納される。
【０２８７】
減算処理の場合の各ビットの流れとしては、図４２に示す加算演算において“加算”に代えて、“減算”が行なわれればよく、ビットの流れは同じである。
【０２８８】
図４４は、乗算ａ・ｂ＝ｃを行なう乗算プログラムの一例を示す図である。以下、図４４を参照して、２項乗算演算処理について説明する。
【０２８９】
まず、行番号０において、ポインタレジスタｒ２およびｒ３に、定数ａｓおよびｃｓが設定される。この行番号０に挙げる初期設定時においては、被乗数ａおよび乗算結果ｃの領域の初期設定が行なわれ、乗数ｂの領域の設定はまだ行なわれない。
【０２９０】
行番号１において、ｆｏｒ文において、被乗数ａの格納領域範囲のビット幅だけ乗算を繰返すことが指定される。“ａ＿ｂｉｔ＿ｃｏｕｎｔ”は、被乗数ａのビット幅を示す。
【０２９１】
行番号２の関数文において、ポインタレジスタｒ２の指定する被乗数ビットａ［ｊ］が転送されてＸレジスタに格納される。次いで、このＸレジスタに格納された被乗数ビットａ［ｊ］が、マスクレジスタ（Ｍレジスタ）に格納される（被乗数ビットａ［ｊ］が“０”のときに乗算を行なう必要がないため、乗算を停止するためである。）
行番号３の命令により、ポインタレジスタｒ３のポインタがポインタレジスタｒ０にコピーされ、次いで、ポインタレジスタｒ１に、定数ｂｓが設定され、乗数ｂの初期アドレスが設定される。
【０２９２】
行番号４において、Ｃレジスタのクリアが行なわれる。
行番号５において、ｆｏｒ文により、乗数ｂに対する繰返し処理が指定される。“ｂ＿

ｂｉｔ＿ｃｏｕｎｔ”は、乗数ｂのビット幅を示す。

【０２９３】

行番号６における関数文においては、ポインタレジスタｒ０のポインタが指定するメモリセルデータ、すなわち乗算結果がＸレジスタへ転送される（ロードされる）。次に、ポインタレジスタｒ１が指定する乗数ビットｂ［ｉ］のＡＬＵユニットへの転送が行われ、、Ｍレジスタの格納値が１のときに、Ｘレジスタの乗算結果ｃと乗数ｂの対応のビットｂ［ｉ］との加算が行なわれる。この加算演算命令は、Ｍレジスタ（マスクレジスタ）の格納値が“０”のときには行なわれない。この処理により、乗算ａ［ｊ］ｘｂ［ｉ］が実現され、この乗算結果がそれまでの部分積と加算される。

【０２９４】

この加算結果が、ポインタレジスタｒ０が示す位置に転送されて格納され、ポインタレジスタｒ０のカウンタが１増分される。行番号６の関数文の命令が、行番号５のｆｏｒ文の条件が満たされるまで、すなわち、乗数ｂの全ビットについて、繰返し実行される。この加算処理により、１つのビットａ［ｊ］についての部分積生成とそれまでに生成された部分積との加算が実行される。

【０２９５】

１つの乗数ｂの全ビットについての処理が完了すると、行番号８において、Ｃレジスタの格納値がＸレジスタに格納され、ポインタレジスタｒ０が指定するアドレス位置に、このＸレジスタに転送されたキャリが格納される。これにより、部分積の加算演算処理が完了する。

【０２９６】

次いで、行番号９において、ポインタレジスタｒ３のポインタが１増分され、次の桁の乗数が指定される。行番号２から行番号９の演算処理が、被乗数ａの各ビットについて繰返し実行される。これらの一連の処理により、ビットシリアルに乗算を行なうことができる。

【０２９７】

図４５は、図４４に示す乗算プログラム実行時のビットの流れを模式的に示す図である。図４５において、被乗数ａのビットａｊがマスクレジスタ（Ｍ）に格納される。次いで、乗算結果ビットｃｊが読出されてＸレジスタに格納され、また乗数ビットｂｉが読出されて、加算が選択的に行なわれる。この加算時において、マスクレジスタ（Ｍレジスタ）に格納された被乗数ビットａｊが“１”のときに、乗算結果ビットｃｊと乗数ビットｂｉとの加算が行なわれる。被乗数ビットａｊが“０”のときには、この加算は行なわれず、Ｘレジスタには乗算結果ビットｃｊが維持される。したがって、この加算結果は、ｃｊ＋ａｊ・ｂｉを示しており、この加算結果が元のビット位置ｃｊに格納される。この処理が、乗数ｂの全ビットについて繰返し実行される。したがって、乗数ｂと被乗数ビットａｊの部分積が求められて、その部分積結果が対応の桁の部分積ビットに加算される。したがって、被乗数ａの各ビット毎に部分積生成して、それまでの部分積とを加算する処理が繰返される。

【０２９８】

図４６は、除算ａ／ｂ＝ｃ...ｄを行なう際のエントリのアドレスの割当を概略的に示す図である。被除数ａの開始アドレスａｓがポインタレジスタｒ０により指定され、余りｄの格納領域の開始アドレスｄｓが、ポインタレジスタｒ１に格納される。除数ｂおよび商ｓは開始アドレスが、それぞれ、ｂｓおよびｃｓである。

【０２９９】

図４７は、この除算を行なうプログラムの一例を示す図である。以下、図４７を参照して、除算プログラムの演算内容について説明する。

【０３００】

図４７において、行番号０の命令により、単位ＡＬＵ回路におけるマスクレジスタ（Ｍレジスタ）がセットされ、対応のＡＬＵ回路が演算可能状態に設定される。また、ポインタレジスタｒ０およびｒ１に、それぞれ、演算数ａおよびｂの開始アドレスａｓおよびｂ

ｓが設定される。

【０３０１】

行番号１において、繰返し文（ｆｏｒ文）が記述され指定され、余りの初期設定が行なわれる。すなわち、ポインタレジスタｒ０に従って、被除数ａがポインタレジスタｒ１の指定する余り格納領域にＸレジスタを介して転送されて格納される。被除数ａの全ビットについて、この動作が繰返され、初期状態において、余りｄとして、被除数ａが設定される。この余り格納領域は、被除数ａの上位ビット領域にビット幅拡張されて、そのビット幅が十分に大きくされており、このビット幅拡張された領域に、ビット幅調整された被除数が格納される。

【０３０２】

行番号２の命令において、ポインタレジスタｒ２に、商ｃの開始アドレスｃｓと商ｃのビット幅より１小さい数（ｂｉｔ＿ｃｏｕｎｔ－１）との和が設定される。これにより、ポインタレジスタｒ３には、商ｃの格納領域の最上位アドレスが設定される。この行番号２の命令において、同様、ポインタレジスタｒ２に、余りｄの開始アドレスｄｓと余りｄのビット幅より１小さい値（ｂｉｔ＿ｃｏｕｎｔ－１）との和が設定される。これにより、ポインタレジスタｒ２に、初期値として、最初の被除算対象ビットを格納する領域の最下位アドレスが設定される。

【０３０３】

行番号３において、繰返し文（ｆｏｒ文）が記述される。この行番号３の繰返し文に続いて、行番号４から行番号７までの命令が、繰返し関数として規定される。

【０３０４】

まず、行番号４において、マスクレジスタ（Ｍレジスタ）がセットされ、また、Ｘレジスタがクリアされる。このＸレジスタのクリア値が、ポインタレジスタｒ３の規定するアドレス領域、すなわち商ｃの最上位ビット位置に格納される。これにより、商の初期化（クリア）が実行される。

【０３０５】

行番号５の命令により、ポインタレジスタｒ２の内容が、ポインタレジスタｒ０に格納される。次いで、ポインタレジスタｒ１に、除数の開始アドレスｂｓが設定され、また、Ｃレジスタがクリアされる。

【０３０６】

行番号６において、再び、繰返し文が記述され、繰返し関数として、行番号７の命令が規定される。すなわち、ポインタレジスタｒ０が指定するアドレスのメモリセルのデータがＸレジスタに格納され、このポインタレジスタｒ０のポインタが１増分される。次いで、ポインタレジスタｒ１が指定するアドレスのメモリセルデータが、Ｘレジスタに格納されたデータから減算される。この処理が繰返し実行される。

【０３０７】

この減算が完了すると、次いで、行番号９の命令により、Ｃレジスタの内容が、Ｘレジスタに転送される。このＸレジスタの格納値が反転され、Ｍレジスタにその反転値が格納される。この演算により、除数ｂと最初の被除数との大小が判定される。

【０３０８】

行番号１０において、ポインタレジスタｒ２の内容が、再び、ポインタレジスタｒ０にコピーされ、またポインタレジスタｒ１に、再び、乗数ｂの開始アドレスｂｓが初期設定されて、Ｃレジスタがクリアされる。次の処理の準備が行われる。

【０３０９】

行番号１１において再び繰返し文が指定され、ポインタレジスタｒ０の指定するアドレスのメモリセルデータから、ポインタレジスタｒ１が規定するメモリセルのデータが減算される。このとき、ポインタレジスタｒ１のポインタが１増分される。この演算結果がＸレジスタに格納され、この減算結果が、ポインタレジスタｒ０が規定するメモリセルアドレスの位置、すなわち元の読出位置に格納され、ポインタレジスタｒ０のポインタが１増分される。この動作が、繰返し実行される。

【０３１０】
行番号１４において、ポインタレジスタｒ２の値を１減分し、行番号１５の命令により、Ｘレジスタに１を格納し、このＸレジスタに格納された値をポインタレジスタｒ３が指定するメモリセル位置に格納し、このポインタレジスタｒ３の値が１減分される。
【０３１１】
行番号１６において行番号１の指定する繰返し文の関数の完了が規定されており、したがってこの行番号２から１５に示す処理が、繰返し実行される。
【０３１２】
したがって、この図４７に示す除算プログラムにおいても、ビットシリアル態様で被除数から除数を順次減算し、その減算を、選択的に、除数と被除数の大小関係に応じて実行することにより、除数ビットを生成することができる。また、余りｄの領域に、被除数を格納し、この余りの領域の演算開始位置を順次減分して下位ビット方向へシフトさせることにより、除算時の被除数の桁下げを行って、順次被除数から除数を減算して、商として１が立つかを決定する。この操作を繰返すことにより、除算完了時に、余りを確実に求めることができる。
【０３１３】
図４８は、図４７に示す除算プログラム実行時のデータの流れを示す図である。以下、図４８を参照して、具体的に除算処理について説明する。
【０３１４】
図４８（Ａ）に示すように、行番号０の命令文により、マスクレジスタ（Ｍレジスタ）に“１”が設定され、ポインタレジスタｒ０が、被除数ａの最下位ビットアドレスａｓを指定する。また、ポインタレジスタｒ１は、余りｄの格納領域の最下位ビットアドレス位置ｄｓを指定する。
【０３１５】
行番号１の命令により、ポインタレジスタｒ０およびｒ１を順次増分してメモリロード／ストア動作を実行することにより、被除数ａが、剰余格納領域にコピーされる。この剰余格納領域のビット幅は、被除数ａのビット幅よりも大きい（被除数ａおよび除数ｂのビット幅の和以上のビット幅が準備される）。
【０３１６】
この剰余領域に被除数ａを下位ビット領域にコピーすることにより、被除数ａの上位ビットが拡張され、ビット幅調整された被除数から除数を順次減算して、商を求める準備が行なわれる。
【０３１７】
次いで、行番号２の命令群により、ポインタレジスタｒ３に商格納領域の最上位ビット位置アドレスが設定され、またポインタレジスタｒ２が、剰余格納領域における被乗数ａの最上位ビット位置を指定する状態に設定される。
【０３１８】
次いで、図４８（Ｂ）に示すように、行番号４の命令群により、マスクレジスタ（Ｍレジスタ）が再び、“１”に設定され、Ｘレジスタがクリアされ、“０”を格納する状態に設定され、このＸレジスタの格納値が、商格納領域の最上位ビット位置に格納され、前の演算サイクル時における商のクリアが行なわれる。
【０３１９】
次いで、このポインタレジスタｒ０およびポインタレジスタｒ２のポインタを転送し、剰余格納領域における被除数ａの最上位ビット位置を指定する。この状態で、ポインタレジスタｒ０およびｒ１のポインタを順次増分して、減算動作を実行し、その減算結果が、ＸレジスタおよびＣレジスタに格納される。この操作は、被除数ａの最上位ビットａｍから除数ｂのビット幅分上位のビットで構成される値から除数ｂを減算する操作に対応する。すなわち、桁合わせされた被除数の上位ビット側から除数ｂを減分する操作が実行される。
【０３２０】

次いで図４８（Ｃ）に示すように、商ｃの最初のビットについての比較ループが実行された後に、行番号９の命令群により、Ｃレジスタの内容がＸレジスタに転送され、このＸレジスタの格納値が反転（ＮＯＴ）され、反転値がＭレジスタに転送される。Ｃレジスタの格納値が“１”の場合には、ボローが発生しており、除数ｂの方が、大きく、商ｃの最上位ビットに１を立てることができない状態を示す。Ｃレジスタの格納値が０の場合には、差分値が正であることを示しており、この場合、Ｍレジスタ（マスクレジスタ）に１が格納される。マスクレジスタ（ＭレジスタＭレジスタ）が“０”を格納しているときには、指定された命令は実行されない。マスクレジスタ（Ｍレジスタ）の格納値が“１”のときに、指定された命令に従った演算処理が実行される。すなわち、商として０が立つか１が立つかを、マスクレジスタ（Ｍレジスタ）の格納値により決定する。

【０３２１】

次いで、図４８（Ｄ）に示すように、行番号１０および１１の命令群により、再び被除数ａの最上位ビットａｍを最下位ビットとする数から除数ｂの減算処理が行なわれ、この減算結果が、Ｘレジスタに格納されかつ上位領域の元の領域に格納される。この減算処理は、ポインタｒ１およびｒ０（ポインタｒ２のポインタ値が転送されている）を順次増分することにより行なわれる。この減算処理は、マスクレジスタ（Ｍレジスタ）の格納値が“１”のときに行なわれ、マスクレジスタ（Ｍレジスタ）の格納値が“０”のときには、この減算処理は実行されない。商として０が立つ場合には、この減算処理を行う必要がなく、商ｃの対応のビットに０を格納することが要求される。この不必要な処理についても、分岐を行わずに命令が仮想的に実行されるのは、他のエントリでの除算において１が立つ可能性があり、全エントリにおいて並行して除算処理を実行する必要があるためである（コントローラから共通の制御信号が各エントリに対して生成される）。

【０３２２】

次いで、図４８（Ｅ）に示すように、減算処理が完了すると、行番号１４の命令に従ってポインタレジスタｒ２のポインタが１減分され、次いで、Ｘレジスタに１が設定され、ポインタレジスタｒ３のポインタが示す位置に“１”が格納される。この処理は、Ｍレジスタが１の場合に実行され、Ｍレジスタ（マスクレジスタ）の格納値が０の場合には格納されず、商ｃの格納領域のポインタレジスタｒ３の指定する位置には“０”が維持される。

【０３２３】

これにより、ポインタレジスタｒ３のポインタが１減分され、次の商のビット位置が指定される。

【０３２４】

以降、上述の処理を繰返すことにより、最終的に、図４８（Ｆ）において、ポインタレジスタｒ３が商ｃの最下位ビットｃｓを指定し、またはポインタレジスタｒ０が、剰余格納領域における最下位ビットｄｓを指定する状態に設定される。これにより、減算処理を繰り返し実行することにより、商ｃの最下位ビットについての減算結果が求められる。剰余格納領域においては、被除数ａと除数ｂの減算結果に基づいた減分値が格納される（Ｍレジスタの格納値が１の場合）。

【０３２５】

剰余領域の被除数ａが除数ｂよりも小さい場合には、Ｍレジスタの格納値は“０”となるため、最終的に商ｃ＝０、剰余ｄ＝ａなる演算結果が求められる。

【０３２６】

このＡＬＵユニットにおいて、レジスタを複数個設け、これらのレジスタを用いることにより、除算処理を、ビットシリアル態様で実現することができる。これにより、複数のデータに対する除算処理を並列に実行することができ、各エントリにおける除算内容が、商に１が立つ場合および０が立つ場合においても、その動作演算サイクル数は同じであり、並列除算処理が実現することができる。

【０３２７】

以上のように、この発明の実施の形態１１に従えば、演算処理ユニット群において各単

位ＡＬＵ回路ブロック（演算処理ユニット）にマスクレジスタ、キャリレジスタ、およびＸレジスタを設け、演算回路の演算処理については、プログラム命令に従ってコントローラによりその処理を設定することにより、ビットシリアル態様で大量のワードに対して並列処理を行なうことができる。

【０３２８】

なお、コントローラの構成としては、プログラム命令をデコードし、そのデコード結果に従ってメモリセルマット（主演算回路）のメモリセル選択および書込／読出の制御信号を生成し、またＡＬＵユニットの論理演算処理回路の論理演算内容を、指定された演算状態を実現するようにレジスタ制御信号および演算器選択信号を生成すればよく、また、アドレス算出は、汎用レジスタおよびポインタレジスタを用いて実行することができる。

【０３２９】

［実施の形態１２］

図４９は、この発明の実施の形態１２に従う単位ＡＬＵ回路ブロック（ＡＬＵユニット）３４の構成を概略的に示す図である。図４９においては、ＡＬＵユニット３４は、算術演算論理回路（ＡＬＵ）５０、Ｘレジスタ５４およびＣレジスタ５６に加えて、Ｙレジスタ２００と、Ｙａレジスタ２０１と、Ｄレジスタ２０２と、Ｄレジスタ２０２の格納値に従ってＹレジスタ２００およびＹａレジスタ２０１の格納値の一方を選択して算術演算論理回路５０へ転送するセレクタ（ＳＥＬ）２０３と、Ｚレジスタ２０４とを含む。

【０３３０】

このＺレジスタ２０４は、算術演算論理回路（ＡＬＵ）、Ｘレジスタ５４およびＣレジスタ５６からのデータを受けて、別エントリのＸレジスタまたはメモリセルマット３０の対応のエントリへデータを転送する。また、Ｘレジスタ５４は、他エントリのレジスタとデータを転送することができる。

【０３３１】

ＡＬＵユニット３４は、さらに、Ｆレジスタ２０５と、Ｆレジスタ２０５の格納値に従ってＸレジスタ５４の格納値を選択的に算術演算論理回路５０へ転送するゲート回路２０６と、定数値を格納するＮレジスタ２０７と、算術演算論理回路５０およびＺレジスタ２０４の活性／非活性を制御するマスクビットを格納するＶレジスタ２０８を含む。Ｖレジスタ２０８は、先の実施の形態１１のマスクレジスタ（Ｍレジスタ）と同様の機能を実現する。

【０３３２】

この図４９に示すＡＬＵユニット３４の構成においては、レジスタ回路の数が、実施の形態１１に比べて増加される。これらの増加したレジスタ回路を効果的に利用して、乗算処理を、２次のブースアルゴリズムに従って実行する。２次のブースアルゴリズムは、生成される部分積の個数を半減する。被乗数をＸ、乗数をＹ、積をＺとすると、積Ｚは、次式（１）で表わされる。

【０３３３】

【数１】

$$Z = X \cdot \sum_{j=0}^{(n-1)/2} (y_{2j-1} + y_{2j} - 2 \cdot y_{2j+1}) \cdot 2^{2j}$$

$$Y = (y_n, \cdots, y_0) \qquad \cdots (1)$$

【０３３４】

上式（１）から、乗数Ｙの隣り合う３ビットを同時に見ることにより、被乗数Ｘとの乗算により生成される部分積の個数を半減することができる。また、上式（１）の括弧の中の値は、０、±１、±２の間で変化するため、加算されるべき部分積は、±２・Ｘ・2^{j}、±Ｘ・2^{2j}、０のいずれかとなる。２倍演算は、１ビット左シフトにより実現するこ

とができる。負の演算は、２の補数値を加算することにより実現される。

【０３３５】

図５０は、２次のブースアルゴリズムに従う部分積生成の手順を示す図である。Ｘ２ｊについては、対応の３ビットｙ（２ｊ＋１）、ｙ（２ｊ）、およびｙ（２ｊ－１）がすべて０であるかすべて１の場合には、上式（１）から０であるため、シフトアップは不要であり、０が格納される（演算は行なわない）。ここで、乗数ビットの下付の添え字を括弧内の数字で示す。

【０３３６】

乗数ビットｙ（２ｊ＋１）が０のときに、乗数ビットｙ（２ｊ）またはｙ（２ｊ－１）の一方が１の場合には、被乗数ビットＸ・2^{j}が１倍されるため、元のビット位置に格納する（２ｊビットシフトアップ）。

【０３３７】

乗数ビットｙ（２ｊ＋１）が０であり、乗数ビットｙ（２ｊ）およびｙ（２ｊ－１）が共に１の場合には、この被乗数ビットＸ・2^{j}が２倍され、１ビットさらにシフトアップされるため、（２ｊ＋１）ビット分、そのビット位置がシフトアップされる。

【０３３８】

乗数ビットｙ（２ｊ＋１）が１であり、乗数ビットｙ（２ｊ）およびｙ（２ｊ－１）が共に０の場合には、－２倍となるため、（２ｊ＋１）ビットシフトアップし、かつその２の補数値を求めるかまたは２の補数値を先に求めてから（２ｊ＋１）ビットシフトする。

【０３３９】

乗数ビットｙ（２ｊ＋１）が１であり、乗数ビットｙ（２ｊ）またはｙ（２ｊ－１）のいずれかが１の場合には、被乗数Ｘが－１倍されるため、２ｊビットだけ乗算結果をシフトアップしかつその２の補数を求める（または、乗算結果の２の補数値を２ｊビットシフトアップする）。

【０３４０】

図５１は、図５０に示す部分積生成手順を模式的に示す図である。被乗数Ｘに対し、乗数ビットｙ（２ｊ－１）、ｙ（２ｊ）およびｙ（２ｊ＋１）のデコード結果を乗算して部分積を生成する。この場合、３ビットの乗数の値に応じて、被乗数Ｘに対する係数は、０、±１、±２のいずれかとなる。

【０３４１】

乗数ビットｙ（２ｊ）桁に対応する部分積を生成するため、この被乗数Ｘは、係数±１の場合には、２ｊ桁シフトし、係数±２の場合には、さらに、１桁上位ビット方向にシフトする。２次のブースアルゴリズムに従って、被乗数Ｘをシフトすることにより、部分積Ｐを生成することができる。

【０３４２】

図５２は、この２次のブースアルゴリズムに従う部分積生成の具体例を示す図である。図５２においては、被乗数ａが（０１１１）であり、乗数ｂが（０１１０）である。２次のブースアルゴリズムに従って、乗数ビットの組においては、偶数ビット（ｙ（２ｊ））が、その中心ビットとして利用される。したがって、乗数ｂの第０ビットｂ［０］を乗数ビットｙ（２ｊ）と置く。このとき、乗数ビットｙ（２ｊ－１）は０に設定される。この場合には、図５０に示す表から、－２倍の演算処理を行なうため、被乗数ａを１ビットシフトし、その２の補数を求める。これにより、（１００１０）が、部分積として算出される。乗算結果のビット位置の調整のために、常時ビット方向に符号拡張が行われ、上位ビットに“１”が設定される。

【０３４３】

次の部分積生成においては、乗数ビットｂ［２］が、乗数ビットｙ（２ｊ）として用いられる。したがって、この場合では、乗数ａを２倍することにより、部分積が求められ、ｊが１であるため、３ビット左シフトさせることにより、部分積が得られる。これらの部分積を加算することにより、乗算結果Ｚ＝（００１０１０１０）が求められる。これにより、ａｘｂ＝７ｘ６＝４２が求められる。

【０３４４】
この２次のブースアルゴリズムの場合、４ビットの乗算を行なう場合には、部分積計算が２回であり、各ビットについて部分積を算出する場合に比べて、大幅に部分積算出回数を低減することができる。この２次のブースアルゴリズムに従う乗算を、図４９に示すＡＬＵユニット３４を用いて実現する。以下、この２次のブースアルゴリズムを実行するための演算命令を定義する。
【０３４５】
図５３は、この発明の実施の形態１２におけるレジスタに対する操作を表わすレジスタ命令を一覧にして示す図である。この図５３においては、実施の形態１１のレジスタ命令に加えて、さらに、１命令で２増分する操作を示す命令“ｒｅｇ．ｉｎｃ２　ｒｘ”が準備される。この命令“ｒｅｇ．ｉｎｃ２　ｒｘ”は、ポインタレジスタｒｘのポインタを２増分する命令である。他のレジスタ命令は、先の実施の形態１１において図３５を参照して説明したレジスタ命令と同じである。
【０３４６】
図５４は、ＡＬＵユニットに含まれるＸレジスタ、Ｖレジスタ、Ｎレジスタ、ＣレジスタおよびＦレジスタに対する操作命令を一覧して示す図である。
【０３４７】
命令“ａｌｕ．ｓｅｔ．＃Ｒ”は、レジスタ＃Ｒ（Ｘレジスタ、Ｖレジスタ、およびＮレジスタ）に“１”をセットする命令である。
【０３４８】
命令“ａｌｕ．ｃｌｒ．＃ＲＲ”は、レジスタ＃ＲＲ（Ｘレジスタ、Ｃレジスタ、およびＦレジスタ）をクリアする（０をセットする）命令である。
【０３４９】
これらのセット／クリア命令は、先の実施の形態１１のＡＬＵ命令のセット／クリア命令と同様である。しかしながら、本実施の形態１２においては、Ｘレジスタ、Ｖレジスタ、Ｎレジスタがセット可能であり、またＸレジスタ、ＣレジスタおよびＦレジスタがクリア可能である。
【０３５０】
図５５は、ＡＬＵユニットに含まれるレジスタに対するレジスタ間転送命令を一覧にして示す図である。
【０３５１】
この命令“ａｌｕ．ｃｐｙ．＃Ｒ＃Ｕ”は、レジスタ＃Ｒの内容を、レジスタ＃Ｕへコピーする操作を指令する。この図５５に示すコピー命令も、先の実施の形態１１のレジスタ間転送命令と、単に利用されるレジスタの命名が異なるだけであり、操作内容は同様である。
【０３５２】
図５６は、この発明の実施の形態１２におけるＡＬＵ命令のうちロード／ストア命令を一覧にして示す図である。
【０３５３】
命令“ｍｅｍ．ｌｄ．＃Ｒ@ｒｘ”は、ポインタレジスタｒｘの指定するアドレスのメモリセルデータＡｊ［ｒｘ］をレジスタ＃Ｒ（Ｘレジスタ、Ｙレジスタ）へ格納する命令である。
【０３５４】
命令“ｍｅｍ．ｓｔ@ｒｘ”は、Ｚレジスタの格納値を、ポインタレジスタｒｘが指定するメモリセルアドレスＡｊ［ｒｘ］へ格納する命令である。このストア命令は、Ｖレジスタの格納値が“１”であり、対応のＡＬＵユニットがイネーブル状態に設定されるときに実行される。マスクレジスタ（Ｖレジスタ）Ｖがクリア状態のときには、このストア動作は実行されない。
【０３５５】
図５７は、エントリ間のデータ移動を行なう命令を一覧にして示す図である。

命令“ｅｃｍ．ｍｏｖ．ｎ＃ｎ”は、定数ｎ離れたエントリｊ＋ｎのＺレジスタの格納値がエントリｊのＸレジスタに移動される。このエントリ間データ転送時においては、サイクリックに転送先が決定される（最大エントリ番号を超えると最小エントリ番号のエントリに戻る）。

【０３５６】

命令“ｅｃｍ．ｍｏｖ．ｒ　ｒｎ”は、レジスタｒｘの格納値ｒｎ離れたエントリｊ＋ｒｎのＺレジスタの格納値がエントリｊのＸレジスタに移動される。この移動時においても、転送先は、サイクリックに決定される。

【０３５７】

このレジスタ設定値ｒｎに従うエントリ間データ転送時、用いられるポインタレジスタは、ｒ０からｒ３の４つのポインタレジスタの格納値のいずれかにより設定される。

【０３５８】

このエントリ間データ転送時においては、ＺレジスタからＸレジスタへのデータ転送が行なわれる。

【０３５９】

図５８は、演算処理ユニット（ＡＬＵユニット）における算術演算を規定する命令を一覧にして示す図である。

【０３６０】

命令“ａｌｕ．ｏｐ．ａｄｃ”は、ポインタレジスタｒｘが指定するメモリアドレスのデータをＹレジスタに格納し、このＹレジスタの格納値とＸレジスタに格納された値との全加算を行なう命令である。加算結果（Ｓｕｍ）は、Ｚレジスタに格納され、キャリは、Ｃレジスタに格納される。この加算演算は、ＮレジスタおよびＶレジスタが共にセットされているときに実行される。

【０３６１】

命令“ａｌｕ．ｏｐ．ｓｂｂ”は、ポインタレジスタｒｘに指定されているメモリアドレスのデータをＹレジスタに格納し、このＹレジスタに格納された値とＸレジスタに格納された値との減算を行なう命令である（Ｙ－Ｘ）。減算結果がＺレジスタに格納され、ボローがＣレジスタに格納される。この減算命令も、Ｎレジスタ２０７およびＶレジスタ２０８が共にセットされているときに実行される。

【０３６２】

図５９は、ＡＬＵ命令のうちの２次のブースアルゴリズム実行に関連する算術演算命令を一覧にして示す図である。

【０３６３】

命令“ａｌｕ．ｏｐ．ｂｏｏｔｈ”は、２次のブースアルゴリズムにおける条件分岐に必要な値（ｙ（２ｊ＋１）、ｙ（２ｊ）、ｙ（２ｊ－１））＝（Ｙ，Ｘ，Ｆ）の格納値を用いて２次のブースアルゴリズム実行に必要な条件分岐レジスタＮレジスタおよびＶレジスタの値を決定する。このブース命令“ａｌｕ．ｏｐ．ｂｏｏｔｈ”の実行前に、ロード命令を用いて２ビットの乗数がＸレジスタ５４およびＹレジスタ２００にそれぞれ格納される。これらの処理は、マスクレジスタ（Ｖレジスタ）２０８がセットされているときに実行される。

【０３６４】

Ｎレジスタには、乗算によりシフトアップを行なうか否かを示す情報が設定される。Ｄレジスタ２０２には、（２ｊ＋１）ビットシフトするかの情報が格納される。Ｙレジスタの値が、Ｆレジスタ２０５に格納される。すなわち、Ｎレジスタにおいては、Ｙレジスタの格納値ｙ（２ｊ＋１）が“１”のときには、ＸレジスタおよびＦレジスタの格納値（ｙ（２ｊ）およびｙ（２ｊ－１）の少なくとも一方が０のときに“１”がセットされ、また、Ｙレジスタ２００の格納ビットｙ（２ｊ＋１）が０のときには、ＸレジスタおよびＦレジスタに格納されたビット値ｙ（２ｊ）およびｙ（２ｊ－１）の一方が“１”のときに、このＮレジスタに“１”がセットされ、シフトアップが指定される。

【０３６５】

Ｄレジスタは、Ｙレジスタの格納値ｙ（２ｊ＋１）が０でありかつＸレジスタおよびＦレジスタの格納値ｙ（２ｊ）およびｙ（２ｊ－１）が共に０であるか、またはＹレジスタの格納値が１のときにＸレジスタおよびＦレジスタの格納値が共に０のときに、“１”に設定される。このＤレジスタの格納値は、（２ｊ＋１）ビットのシフトアップを指定する。このＹレジスタの内容をＦレジスタへ転送することにより、乗数ビットｙ（２ｊ＋１）を、ｊが１増分されたときの次の演算時に、乗数ビットｙ（２ｊ－１）として利用することができる。

【０３６６】

命令“ａｌｕ．ｏｐ．ｅｘｅ”は、この２次のブースアルゴリズムの実行命令であり、ＤレジスタおよびＦレジスタに格納値に従って条件分岐を行なう。

【０３６７】

Ｄレジスタの格納値が１の場合には、Ｙａレジスタの値をセレクタ２０３により選択する。Ｄレジスタ２０２の格納値が０のときには、Ｙレジスタの格納値が選択される。この実行命令（ＥＸＥ命令）において、Ｆレジスタの格納値が０の場合には加算命令となり、Ｆレジスタの格納値が１の場合には減算命令となる。

【０３６８】

この実行命令“ａｌｕ．ｏｐ．ｅｘｅ”の有効時、Ｆレジスタ２０５の格納値に従って図４９に示すゲート回路２０６が、Ｘレジスタ５４の格納値の反転または非反転を行なう。ブース命令実行時においては、このゲート回路２０６は、Ｘレジスタ５４およびＦレジスタ２０５の格納値の相補値Ｘ，！ＸおよびＦ，！Ｆを生成する。

【０３６９】

ゲート回路２０６の演算処理内容は、コントローラ含まれる命令デコーダからの制御信号（ＡＬＵ制御）に基づいて決定される。

【０３７０】

図６０は、この図５８に示すブース命令実行時の各レジスタ、すなわちＹレジスタ２００、Ｘレジスタ５４、Ｆレジスタ２０５、Ｄレジスタ２０２およびＮレジスタ２０７の格納値およびその対応の制御内容（部分積生成手順）を一覧にして示す図である。

【０３７１】

上述のように、Ｙレジスタ２００、Ｘレジスタ５４およびＦレジスタ２０５に、それぞれ乗数ビットｙ（２ｊ＋１）、ｙ（２ｊ）、ｙ（２ｊ－１）がセットされる。これらのＹレジスタ、ＸレジスタおよびＦレジスタの格納値に従ってブース命令“ａｌｕ．ｏｐ．ｂｏｏｔｈ”を実行することにより、Ｄレジスタ２０２およびＮレジスタ２０７に、０または１がセットされる。このブース命令により、被乗数Ｘに対する部分積を算出する準備が完了する。

【０３７２】

Ｆレジスタの値は、ＤレジスタおよびＮレジスタの格納値との組合せで、部分積生成時に乗数を補数にするか否かの判定に用いられる。また、ブースアルゴリズム実行時に、Ｆレジスタの格納値に従って、加算および減算を切換えることにより、部分積の選択的な補数生成を行なうことができる（減算操作は、補数の加算と同じである）。

【０３７３】

また命令“ａｌｕ．ｏｐ．ｅｘｅ”は、ブースアルゴリズム乗算以外においても適用用途が存在し、Ｄレジスタ２０２の格納値に従って加算および減算のいずれかを選択的に実行することができ、この実行命令“ａｌｕ．ｏｐ．ｅｘｅ”は、加算命令および減算命令を包含した命令である。

【０３７４】

また、Ｙａレジスタ２０１を用いることにより、乗数のシフト動作が実現される。Ｙａレジスタ２０１には、ブース命令の実行時、前回ロードされたＹレジスタ２００の格納値がコピーされている（ＥＸＥ命令におけるＹａ＝Ｙｊ）。したがって、このＹａレジスタ２０１の初期値を０から開始すれば、Ｙレジスタ、Ｘレジスタ、およびＹａレジスタの格納する３ビットにより、２ビット乗数をロードして１ビット乗数をシフトした状態を作る

ことができる。すなわち、（ｙ１、ｙ０，０）から、ビットｙ１をＹａレジスタに格納することにより、次のブース命令の実行によるロード時に（ｙ３、ｙ２、ｙ１）の３ビットの組を生成することができる。

【０３７５】

図６１は、ＡＬＵ命令のうちの論理演算を行なう命令を一覧にして示す図である。

命令“ａｌｕ．ｏｐ．ａｎｄ”は、ポインタレジスタｒｘのポインタが指定するアドレスのメモリセルデータをＹレジスタに格納し、このＹレジスタの格納値とＸレジスタの格納値に対し論理積演算を行ない、その論理積演算結果をＺレジスタに格納する操作を指定する。Ｖレジスタ（マスクレジスタ）がセットされていないときには、この論理積演算（ＡＮＤ演算）は実行されない。

【０３７６】

命令“ａｌｕ．ｏｐ．ｏｒ”は、ポインタレジスタｒｘのポインタが指定するアドレスのメモリセルデータをＹレジスタに格納し、このＹレジスタの格納値とＸレジスタに格納されている値との論理和演算を行なって、論理和演算結果をＺレジスタに格納する命令である。この論理和演算命令は、マスクレジスタ（Ｖレジスタ）がセットされているときに実行される。

【０３７７】

命令“ａｌｕ．ｏｐ．ｅｘｏｒ”は、ポインタレジスタｒｘのポインタが指定するメモリアドレスのデータをＹレジスタに格納し、このＹレジスタに格納されたデータビットとＸレジスタのビットとの排他的論理和演算を行ない、その演算結果をＺレジスタに格納する操作を指定する。この排他的論理和演算（ＥＸＯＲ演算）も、Ｖレジスタがセットされたときに行なわれ、Ｖレジスタがクリア状態のときには実行されない。

【０３７８】

命令“ａｌｕ．ｏｐ．ｎｏｔ”は、Ｘレジスタの格納値を反転し、その反転結果をＺレジスタに格納する操作を指定する。この反転命令も、Ｖレジスタがクリア状態のときには実行されない。

【０３７９】

命令“ａｌｕ．ｏｐ．ＬＴ”は、Ｃレジスタの格納値に従ってＮレジスタを１にセットまたは０にクリアする命令である。Ｃレジスタの格納値が１のときにＮレジスタが０にクリアされる。

【０３８０】

２次のブースアルゴリズムに従って乗算を行なう操作を、これらの命令を用いて記述したプログラムを図６２に示す。以下、図６２を参照して、２次のブースアルゴリズムに従う乗算操作について説明する。

【０３８１】

まず、行番号０の命令により、マスクレジスタ（Ｖレジスタ）がセットされ、演算の実行を指定される。

【０３８２】

行番号１の命令文により、ポインタレジスタｒ２およびｒ３に、それぞれ、乗数ｂの開始アドレスおよび乗算結果ｃを格納する領域の開始アドレスｃｓがセットされる。また、Ｆレジスタがクリアされ、“０”が格納される。

【０３８３】

行番号２において、繰り返し文が記述され、被乗数ａのビット幅が決定され、この繰り返し文の実行時に、ｊが２倍される。

【０３８４】

行番号３において、ポインタレジスタｒ２に設定されたデータビットがＸレジスタに格納され、またポインタレジスタｒ２が指定するメモリセルのデータが、Ｙレジスタに格納される。このとき、ポインタレジスタｒ２は、命令実行時１増分されており、したがってこの行番号３の命令により、２ビットの乗数が、ｙ（２ｊ＋１）およびｙ（２ｊ）が、それぞれＹレジスタおよびＸレジスタに格納される。

【０３８５】
行番号４において、ブース命令が実行され、図５９の操作内容に示すように、Ｎレジスタおよびレジスタの記憶値が設定され、またＹレジスタの格納ビットがＦレジスタにコピーされる。これにより、部分積生成手順が設定される。
【０３８６】
行番号５の命令文により、レジスタｒ３に格納された乗算結果ｃの最初のビット位置を示すアドレスがポインタレジスタｒ０にコピーされ、またポインタレジスタｒ１に被乗数ａの初期アドレス（最下位ビットアドレス）ａｓがセットされる。
【０３８７】
行番号６において繰返し文が記述され、乗数ｂについて繰返し操作が、ｉについてのｆｏｒ文の条件が満たされるまで実行される。この乗数ｂについては、繰返し回数を示す定数ｉは、１ずつ増分する。
【０３８８】
行番号７の命令文により、ポインタレジスタｒ０に格納されたデータ（乗算結果値）がＸレジスタに格納され、次いで、ポインタレジスタｒ１のポインタが指定するデータすなわち被乗数ａの対応のビットがＹレジスタに格納される。この状態で、ブースアルゴリズム実行命令（ＥＸＥ命令）を実行し、被乗数ビットａｊと乗算結果ビットの加算または減算が実行され、部分積の加算が実行され、加算結果がＺレジスタに格納される。この後、ポインタレジスタｒ０が指定するメモリセル位置に、このＺレジスタに格納された加算または減算結果が格納される。
【０３８９】
行番号８において、この行番号６で規定される繰返し文の終了が示される。したがって、この関数文においては、乗数ｂの３ビットの組を固定して、被乗数ａの部分積生成およびそれまでの部分積との加算が実行される。
【０３９０】
行番号９において再び繰返し文が、乗数ｂのビット幅について規定される。部分積の符号拡張による桁合わせの処理が行う。
【０３９１】
行番号１０の命令文により、ポインタレジスタｒ０が指定するアドレス位置のデータがＸレジスタに格納され、先の行番号７の命令により生成された部分積が読出される。次いで再び、２次のブースアルゴリズムに従って演算が行なわれ、部分積生成が行なわれ、再びポインタレジスタｒ０が指定するメモリセル位置にこの部分積生成結果が格納される。ポインタレジスタｒ０は、乗算結果ｃの格納位置を指定しており、ポインタレジスタｒ０のポインタを増分することにより、先の処理により生成された部分積の上位ビット位置に、符号ビットを記述する。生成される部分積のビット幅を最終的な乗算結果ｃのビット幅に一致させる。
【０３９２】
乗算結果ｃの符号拡張処理が完了すると、行番号１２において、ポインタレジスタｒ３のポインタを２増分する。
【０３９３】
行番号１３の関数文の末尾の記述により、被乗数ａの１つのビットａｊについての一連の処理が完了し、次の被乗数ａの１ビット上位のデータについて処理が実行される。
【０３９４】
図６３は、この図６２に示す符号付きブースアルゴリズム乗算プログラムが初期値の１つのエントリのアドレスを示す図である。乗算結果ｃを格納する領域の先頭位置（最下位ビット位置）は、アドレスｃｓで設定される。被乗数ａは、ビット幅ａ＿ｂｉｔ＿ｃｏｕｎｔを有し、その最下位ビット位置はアドレスａｓで指定される。乗数ｂは、ビット幅ｂ＿ｂｉｔ＿ｃｏｕｎｔを有し、その最下位ビット位置が、アドレスｂｓに設定される。
【０３９５】
図６４に示すように、まず、図６２の行番号１の命令文によりポインタレジスタｒ３に

アドレスｃｓが設定され、またポインタレジスタｒ２にアドレスｂｓが設定される。

【０３９６】

行番号３の命令により、ポインタレジスタｒ２の指定する乗数ビットｙ（２ｊ）およびｙ（２ｊ＋１）がそれぞれＸレジスタおよびＹレジスタに格納される。これにより、Ｆレジスタ、Ｄレジスタおよびレジスタの初期値が、行番号４のブース命令により設定される。これにより、部分積について、デコード結果が指定され、０、±１、±２のいずれの演算を行なうかが設定される。

【０３９７】

次いで、行番号５の命令文により、ポインタレジスタｒ３の内容がポインタレジスタｒ０に転送されて、乗算結果を格納する領域のアドレスがポインタレジスタｒ０に指定される。また、被乗数ａの最下位ビットアドレスａｓが、ポインタレジスタｒ１に設定される。行番号７の命令文により、ポインタレジスタｒ０のポインタにより、Ｘレジスタに先のサイクルの乗算結果ビットｃｉが格納され、またポインタレジスタｒ１のポインタに従って乗数ａのビットａｉがＹレジスタに格納される。Ｙａレジスタには、前のサイクルの被乗数ビットａ（ｉ－１）が格納される。Ｄレジスタの格納内容に従って、ＹレジスタおよびＹａレジスタの一方が選択され、Ｎレジスタが“１”のとき、Ｆレジスタの格納値に従って加算または減算が行なわれる。この演算結果は、この結果ビットｃｉが読出されたビット位置に格納される。

【０３９８】

このＹａレジスタおよびＹレジスタの選択により、2^{j} ビットシフトまたは$2^{j}+1$ビットシフト操作が実現される。

【０３９９】

次に、再び、ポインタレジスタｒ０およびｒ１のポインタを増分して、Ｆレジスタ、Ｄレジスタおよびレジスタの内容を固定して、同様の演算処理が実行され、それまでに求められた部分積に対し、新たな被乗数の加算／減算が、ビットシリアル態様で行なわれる。

【０４００】

これらの部分積生成動作が完了すると、次いで行番号９からの命令に従って、乗算結果格納領域の上位ビット位置において、ポインタレジスタｒ０が指定する位置に対し同様のブース実行命令演算が行なわれる。このときＹレジスタには、被乗数ａの最上位ビットａｍが格納され、Ｙａレジスタには、次の上位ビットａ（ｍ－１）が格納される。従って、先の部分積最上位ビット生成と同様の操作を行って、操作結果をポインタレジスタｒ０が指定する位置へ再度書込む。これにより、符号拡張処理が行なわれ、上位ビット位置に０または１が順次格納される。

【０４０１】

これらの処理の完了後、図６５に示すように、ポインタレジスタｒ３のポインタが２増分され、またポインタレジスタｒ２の値が増分され、次の２次のブースデコード動作が行なえる準備が完了する。乗数ｂについて、偶数ビットとその隣接奇数ビットとの組について、上述の一連の処理を実行することにより、ビットシリアル態様で、順次部分積の生成および前の部分積への加算を行って最終積を求めることができる。

【０４０２】

図６２の行番号２の繰返し文に見られるように、部分積の加算処理は、乗数ｂのビット幅の１／２回で完了し、乗算処理を高速で行なうことができる。

【０４０３】

上述の命令群は、２次のブースアルゴリズムによる乗算のみならず、通常の加減算および除算演算に対しても適用することができる。以下、各演算について説明する。

【０４０４】

図６６は、この発明の実施の形態１２に従う演算命令を用いた加算プログラムを示す図である。図６６（Ａ）において、２項加算（ａ＋ｂ）＝ｃを行う。演算数ａおよびｂおよびｃのそれぞれの最下位ビットアドレスが、ａｓ、ｂｓおよびｃｓに設定される。

【０４０５】
図６６（Ｂ）に、この２項加算を行なうプログラムを示す。この加算プログラムにおいては、用いられるレジスタの名称が異なるだけであり、先の実施の形態１１と同様の演算処理が実行される。
【０４０６】
図６７は、この発明の実施の形態１２における２項減算を行なうプログラムを示す図である。図６７（Ａ）に示すように、（ａ－ｂ）＝ｃの演算を行なう。演算数ａ、ｂおよびｃの最下位ビットアドレスはａｓ、ｂｓおよびｃｓである。
【０４０７】
図６７（Ｂ）に、この減算プログラムを示す。この図６７（Ｂ）に示す減算プログラムは、先の実施の形態１１と同様であり、演算命令の名称が異なるだけであり、同様の減算処理を、減算命令“ａｌｕ．ｏｐ．ｓｂｂ”に従って実行することができる。
【０４０８】
図６８は、この発明の実施の形態１２に従う演算命令を用いる符号なし２項乗算のプログラムを示す図である。この図６８に示すプログラムにおいては、ａ・ｂ＝ｃの２項乗算が行なわれる。演算数ａ、ｂおよびｃのそれぞれの最下位ビットのアドレスはａｓ、ｂｓおよびｃｓである。
【０４０９】
この図６８に示す乗算プログラムにおいても、用いられる命令の名称が異なるものの、実施の形態１１と同様の処理が行なわれており、部分積の同一桁のビットを順次加算して、最終積を求めることができる。
【０４１０】
図６９は、この発明の実施の形態１２に従う演算命令を用いる除算プログラムの一例を示す図である。この図６９に示す除算プログラムにおいては、ａ／ｂ＝ｃ…ｄの演算が行なわれる。被除数ａ、除数ｂ、商ｃおよび余りｄのそれぞれの最下位ビットのアドレスは、ａｓ、ｂｓ、ｃｓおよびｄｓに設定される。
【０４１１】
この図６９に示す除算プログラムにおいても、実施の形態１１と同様の処理が、異なる名称のレジスタを用いて実行されており、順次被除数ａから除数ｂを減算する処理を行なって、商および余りを求めることができる。
【０４１２】
以上のように、この発明の実施の形態１２に従えば、単位ＡＬＵ回路ブロック（演算処理ユニット）内に、複数のレジスタおよびゲート回路を設け、演算命令としてブース命令“ａｌｕ．ｏｐ．ｂｏｏｔｈ”およびブースアルゴリズム実行命令“ａｌｕ．ｏｐ．ｅｘｅ”を設けており、２次のブースアルゴリズムに従って乗算を行なうことができ、高速乗算を実現することができる。
【０４１３】
［実施の形態１３］
図７０は、この発明の実施の形態１３に従う単位ＡＬＵ回路ブロック（演算処理ユニット；ＡＬＵユニット）３４の構成を概略的に示す図である。この実施の形態１３においては、メモリセルマットにおいては、１つのエントリＥＲＹが、偶数アドレスのデータビットＡ［２ｉ］を格納する偶数エントリＥＲＹｅと、奇数アドレスのデータビットＡ［２ｉ＋１］を格納する奇数エントリＥＲＹｏに分割される。偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏの同じアドレスのデータビットに対し並列に演算処理を行なうことにより、処理の高速化を図る。
【０４１４】
ＡＬＵユニット３４においては、演算処理を行なうための縦続接続される全加算器２１０および２１１が演算処理装置として設けられる。このＡＬＵユニット３４における処理データおよび演算内容を設定するレジスタ、すなわちＸレジスタ５４、Ｃレジスタ５６、Ｆレジスタ２０５、Ｖレジスタ２０８、Ｎレジスタ２０７は、先の実施の形態１２と同様

の機能を実現する。

【０４１５】

この実施の形態１３においては、ＡＬＵユニット３４内において、さらに、２ビットデータを並列に格納するためのＸＨレジスタ２２０およびＸＬレジスタ２２１と、レジスタ５４、２２０および２２１からのデータの組の一方の２ビットを、Ｄレジスタ２２２の格納値に従って選択するセレクタ（ＳＥＬ）２２７と、Ｆレジスタ２０５の格納ビットに従ってセレクタ２２７の選択した２ビットに対する反転／非反転（正転）操作を行なう選択反転回路２１７と、レジスタ２０７および２０８の格納データに従って全加算器２１０および２１１の３出力Ｓを選択的に出力するゲート２２３および２２４が設けられる。

【０４１６】

選択反転回路２１７の２ビット出力は、全加算器２１０および２１１のＡ入力へそれぞれ与えられる。ＸＨレジスタ２２０およびＸＬレジスタ２２１は、それぞれ内部データ線２２６および２２８を介して奇数エントリＥＲＹｏの奇数アドレスビットおよび偶数エントリＥＲＹｅと偶数アドレスビットの転送を行なう。Ｘレジスタ５４は、スイッチ回路ＳＷａおよびＳＷｂにより、内部データ線２２６および２２８の一方に選択的に接続される。

【０４１７】

全加算器２１０のＢ入力は、内部データ線２２６に結合され、全加算器２１０のＳ出力を受けるゲート２２３の出力が、また、内部データ線２２６に接続される。全加算器２１１のＢ入力は、スイッチ回路ＳＷｃおよびＳＷｄにより、内部データ線２２６および２２８の一方に選択的に接続される。全加算器２１１のＳ出力を受けるゲート２２４の出力は、また、スイッチ回路ＳＷｅおよびＳＷｆに従って内部データ線２２６および２２８の一方に選択的に接続される。これらのスイッチ回路ＳＷａ－ＳＷｆにより、２ビット並列除算処理を行なう場合の、１ビット単位のビットシリアル処理を実行する。

【０４１８】

ゲート２２３および２２４は、Ｖレジスタ２０８およびＮレジスタ２０７の格納値がともに“１”のときに、指定された演算処理を実行し、それ以外においてはハイインピーダンスを出力する（出力ハイインピーダンス状態となる）。

【０４１９】

また、Ｃレジスタ５６の格納値が、全加算器２１１のキャリ入力Ｃｉｎに接続される。全加算器２１０のキャリ出力Ｃｏは、全加算器２１１のキャリ入力Ｃｉｎに接続され、また、スイッチ２２５を介して、全加算器２１０のキャリ入力Ｃｉｎに接続される。このスイッチ２２５は、１ビット単位での演算処理を行なう場合に、全加算器２１０のキャリ出力Ｃｏを切り離して、全加算器２１１のキャリ入力ＣｉｎをＣレジスタ５６に接続する。

【０４２０】

この図７０に示すＡＬＵユニット３４においては、Ｚレジスタは用いられておらず、Ｘレジスタ５４、ＸＨレジスタ２２０およびＸＬレジスタ２２１が、他のエントリの対応のレジスタとデータ転送を行なうことができる。

【０４２１】

この実施の形態１３においては、メモリセルマットのアドレスを指定するポインタレジスタとして、ポインタレジスタｐ０－ｐ３が用いられる。別に、汎用レジスタ内のポインタレジスタｒ０－ｒ３も利用される。

【０４２２】

図７１は、ポインタレジスタｐ０－ｐ３の操作を行なうポインタレジスタ命令を一覧にして示す図である。

【０４２３】

命令“ｐｔｒ．ｓｅｔ　ｎ，ｐｘ”は、任意の値ｎを、ポインタレジスタｐｘにセットする命令である。この任意の値ｎは、１つのエントリのビット幅（０‐ＢＩＴ＿ＭＡＸ）の範囲で任意の値を取ることができる。

【０４２４】

命令“ｐｔｒ．ｃｐｙ　ｐｘ，ｐｙ”は、ポインタレジスタｐｘの内容を、ポインタレジスタｐｙに転送して格納するコピー命令である。

【０４２５】

命令“ｐｔｒ．ｉｎｃ　ｐｘ”は、ポインタレジスタｐｘのポインタを１増分する命令である。

【０４２６】

命令“ｐｔｒ．ｉｎｃ２　ｐｘ”は、ポインタレジスタｐｘのポインタを２増分する命令である。

【０４２７】

命令“ｐｔｒ．ｄｅｃ　ｐｘ”は、ポインタレジスタｐｘのポインタを１減分する命令である。

【０４２８】

命令“ｐｔｒ．ｄｅｃ２　ｐｘ”は、ポインタレジスタｐｘのポインタを２減分する命令である。

【０４２９】

命令“ｐｔｒ．ｓｆｔ　ｐｘ”は、ポインタレジスタｐｘのポインタを、１ビット左シフトする命令である。

【０４３０】

命令“ｐｔｒ．ｉｎｃ２　ｐｘ”および命令“ｐｔｒ．ｄｅｃ２　ｐｘ”を利用することにより、２ビット並列に処理を行なうことができる（奇数および偶数アドレスを同時に更新する）。

【０４３１】

図７２は、１ビット動作時のロードストア命令を一覧にして示す図である。

図７２において、命令“ｍｅｍ．ｌｄ．＃Ｒ@ｐｘ”は、ポインタレジスタｐｘのポインタが示す位置Ａｊ［ｐｘ］のデータを、レジスタ＃Ｒに格納する（ロードする）命令である。

【０４３２】

命令“ｍｅｍ．ｓｔ．＃Ｒ@ｐｘ”は、レジスタ＃Ｒの格納値を、ポインタレジスタｐｘの指定するメモリセル位置Ａｊ［ｐｘ］へ書込む（ストアする）命令である。このストア命令は、マスクレジスタ（Ｖレジスタ２０８）がクリアされているときには、実行されない。

【０４３３】

命令“ｍｅｍ．ｓｗｐ．Ｘ@ｐｘ”は、Ｘレジスタの格納値とポインタレジスタｐｘの指定するメモリセル位置Ａｊ［ｐｘ］のデータとを交換する命令である。このスワップ命令は、マスクレジスタ（Ｖレジスタ）２０８およびＮレジスタ２０７にともに“１”がセットされているときに実行される。Ｘレジスタのクリア／セットを、メモリセルの格納データで実行することにより、回路構成を簡略化する。

【０４３４】

図７３は、２ビット動作時のＡＬＵユニットに対するロード／ストア命令を一覧にして示す図である。

【０４３５】

図７３において、命令“ｍｅｍ２．ｌｄ．Ｘ@ｐｘ”は、ポインタレジスタｐｘの指定するメモリセル位置Ａｊ［ｐｘ］およびＡｊ［ｐｘ＋１］のメモリセルのデータを、ＸＬレジスタ２２１およびＸＨレジスタ２２０に格納する命令である。すなわち、連続アドレス位置のデータの下位ビットが、ＸＬレジスタに格納され、上位ビットがＸＨレジスタに格納される。

【０４３６】

命令“ｍｅｍ２．ｓｔ．Ｘ@ｐｘ”は、ポインタレジスタｐｘの指定するアドレスの連続アドレスＡｊ［ｐｘ］およびＡｊ［ｐｘ＋１］のメモリセルへ、それぞれ、ＸＬレジスタおよびＸＨレジスタの格納値を格納する命令である。ただし、この動作は、マスクレジ

スタ（Ｖレジスタ）がクリア状態のときには実行されない。
【０４３７】
命令“ｍｅｍ２．ｓｗｐ．Ｘ@ｐｘ”は、ポインタレジスタｐｘの指定するアドレスおよび上位アドレスＡｊ［ｐｘ］およびＡｊ［ｐｘ＋１］のデータが、それぞれ、ＸＬレジスタおよびＸＨレジスタの格納値と交換される命令である。ただし、このスワップ命令は、ＶレジスタおよびＮレジスタがともにクリアされているときには、実行されない。
【０４３８】
この２ビット動作時においては、ポインタレジスタｐｘのポインタを用いて連続アドレスＡｊ［ｐｘ］およびＡｊ［ｐｘ＋１］へ同時にアクセスすることにより、２ビット並列処理を実現する。
【０４３９】
図７４は、１ビット動作時のエントリ間データ移動（ｍｏｖｅ）を行なう命令を一覧にして示す図である。このエントリ間データ移動時には、ポインタレジスタｒｎが用いられる。エントリ間データ移動用ポインタレジスタｒｎの候補レジスタとしては、４つのポインタレジスタｒ０－ｒ３が設けられる。
【０４４０】
命令“ｅｃｍ．ｍｖ．ｎ　＃ｎ”は、定数ｎ離れたエントリｊ＋ｎのＸレジスタの格納値を、エントリｊのＸレジスタに転送することを示す命令である。
【０４４１】
命令“ｅｃｍ．ｍｖ．ｒ　ｒｎ”は、レジスタｒｎの格納値離れたエントリｊ＋ｒｎのＸレジスタの値が、エントリｊのＸレジスタに転送される操作を示す命令である。
【０４４２】
命令“ｅｃｍ．ｓｗｐ”は、隣接エントリｊ＋１およびｊのＸレジスタの格納値を交換する操作を指令する命令である。
【０４４３】
図７５は、２ビット動作時のＡＬＵにおけるエントリ間データ移動（ｍｏｖｅ）の操作を指令する命令を一覧にして示す図である。この２ビット操作時においては、命令記述子“ｅｃｍ２”が命令記述子“ｅｃｍ”に代えて用いられる。この命令記述子“ｅｃｍ２”が指定されると、２ビット単位での演算処理が指定され、ＸＨレジスタおよびＸＬレジスタ間での並列のデータ転送が行なわれ、各レジスタ間の転送内容の指定には、先の１ビット動作時と同じ命令記述子“ｍｖ．ｎ＃ｎ”、“ｍｖ．ｒ　ｒｎ”および“ｓｗｐ”が用いられる。
【０４４４】
図７６は、１ビット動作時のＡＬＵユニットにおける算術演算命令を一覧にして示す図である。
【０４４５】
命令“ａｌｕ．ａｄｃ@ｐｘ”は、加算命令である。ポインタレジスタｐｘのポインタが示すメモリアドレスのＡｊ［ｐｘ］のデータとＸレジスタの格納値Ｘｊとを加算し、その結果ｚを元のメモリセルに格納する。すなわち、アドレスＡｊ［ｐｘ］のメモリセルには、加算後の値Ｓｕｍ（サム）が格納され、Ｃレジスタには、キャリが格納される。
【０４４６】
命令“ａｌｕ．ｓｂｃ@ｐｘ”は、減算命令である。ポインタレジスタｐｘのポインタが示すメモリアドレスのＡｊ［ｐｘ］のデータからＸレジスタの格納値Ｘｊを減算し、その減算結果を元のメモリ位置Ａｊ［ｐｘ］に格納する。減算操作後には、元のメモリセルに減算後の値が格納され、Ｃレジスタにボローが格納される。
【０４４７】
命令“ａｌｕ．ｉｎｖ@ｐｘ”は、反転演算命令である。ポインタレジスタｐｘのポインタが指定するメモリアドレスＡｊ［ｐｘ］のデータを反転して元のメモリセル位置に格納する。
【０４４８】

これらの加算命令、減算命令および反転命令は、ＮレジスタおよびＶレジスタがともにセットされているときに実行され、ＮレジスタおよびＶレジスタの一方がクリア状態のときには実行されない。

【０４４９】

命令“ａｌｕ．ｌｅｔ　ｆ”は、レジスタ設定命令である。Ｆレジスタ、Ｄレジスタ、ＮレジスタおよびＣレジスタに、関数値ｆ（４ビット）で指定される値が、それぞれ対応のレジスタに設定される（ｆ＝Ｆ・８＋Ｄ・４＋Ｎ・２＋Ｃ）。

【０４５０】

図７７は、２ビット演算操作時のＡＬＵユニットにおける算術演算命令を一覧にして示す図である。図７７においては、２次のブースアルゴリズムに従って乗算を２ビット単位で行なう命令が示される。

【０４５１】

命令“ａｌｕ２．ｂｏｏｔｈ”は、ブース命令である。このブース命令実行時、ＸＨレジスタ、ＸＬレジスタおよびＦレジスタの格納値から、次の演算用に、Ｎレジスタ、Ｄレジスタおよびレジスタの格納値を求める。このブース命令は、Ｖレジスタがセットされたときに実行される。このブース命令の実行内容は、以下に説明するブース命令実行時のブースデコード結果に基づいて設定される。

【０４５２】

命令“ａｌｕ２．ｅｘｅ@ｐｘ”は、ブース演算の実行命令（ＥＸＥ命令）であり、ＤレジスタおよびＦレジスタの格納値に従って、シフト動作、および正転（非反転）／反転操作が行なわれる。

【０４５３】

このブースアルゴリズムに従う乗算の操作内容については、以下に具体的に説明する。

図７８は、図７７に示すブース命令実行時のレジスタＤおよびＮの格納値を一覧にして示す図である。

【０４５４】

ブース命令実行時においては、ＸＨレジスタ、ＸＬレジスタおよびＦレジスタには、乗数ビットｙ（２ｊ＋１）、ｙ（２ｊ）、およびｙ（２ｊ－１）が格納される。したがって、ＤレジスタおよびＮレジスタのビット値が、先の実施の形態１２の場合と同じであり、Ｎレジスタがビット“１”を格納する場合には、シフト動作を行なうことが指定され、Ｄレジスタの格納値がビット“１”の場合には、（２ｊ＋１）ビットシフトアップすることが指定される。ＸＨレジスタの格納値がビット“１”のときに、シフトアップ時に補数が生成される。

【０４５５】

このブース命令実行時においては、ＸＨレジスタの格納値が、Ｆレジスタへ転送され、またＣレジスタへ転送される。これにより、次の演算時において、Ｆレジスタに乗数ビットｙ（２ｊ－１）が格納される。

【０４５６】

Ｘレジスタには、初期値“０”が格納される。このＸレジスタの初期格納値を用いることにより、乗数ビットを１ビットシフトした値を生成することができる。

【０４５７】

ブース実行命令（ＥＸＥ命令）の実行時においては、まずポインタレジスタｐｘの指定するメモリセルデータＡｊ［ｐｘ］とＸＨレジスタまたはＸＬレジスタの格納値またはその反転値との加算が行なわれ、加算結果が元のメモリセル位置Ａｊ［ｐｘ］に格納される。キャリは、次のメモリセルアドレスＡｊ［ｐｘ＋１］の演算時のキャリとして利用される。このとき、キャリｃｃを用いて、アドレスＡｊ［ｐｘ＋１］のメモリセルデータと、ＸＬレジスタまたはＸＨレジスタの格納値またはその反転値との加算が行なわれ、加算結果が元のメモリセル位置Ａｊ［ｐｘ＋１］に格納される。また、Ｘレジスタの値が、ＸＬレジスタの格納値にＤレジスタの格納値が１のときに変更される。これにより、（２ｊ＋１）ビットシフトする際に、Ｘレジスタに、ｙｉ（＝２ｊ）を乗数ビットとして格納する

ことができる。

【０４５８】

図７９は、この発明の実施の形態１３におけるブースアルゴリズム乗算処理を示すプログラムを示す図である。被乗数ａ、乗数ｂおよび乗算結果ｃの最下位ビットアドレスは、それぞれ、ａｓ、ｂｓおよびｃｓである。ここでは、簡単化のために、乗数ｂおよび被乗数ａはともに同一のビット幅ｂｉｔ＿ｃｏｕｎｔを有するとする。

【０４５９】

まず、行番号０の命令群により、ポインタレジスタｐ２に、乗数ｂの最下位ビットアドレスｂｓがセットされ、またポインタレジスタｐ３に、乗算結果ｃの最下位ビットアドレスｃｓが設定される。

【０４６０】

行番号１において、ｆｏｒ文により、ブースアルゴリズムの乗数ビットの組の範囲が指定される（２ｉずつｉが増分する）。

【０４６１】

行番号２の命令により、まずポインタレジスタｐ２の指定するアドレスＡｊ［ｐ２］およびＡｊ［ｐ２＋１］のデータが、ＸＨレジスタおよびＸＬレジスタにそれぞれ格納されて、次いで、ブース命令が実行され、Ｎレジスタ、Ｄレジスタおよびレジスタの値が設定される。

【０４６２】

行番号３の命令により、ポインタレジスタｐ３のポインタ値ｃｓが、ポインタレジスタｐ０にコピーされ、また、ポインタレジスタｐ１に被乗数ａの最下位ビットアドレスａｓがセットされる。

【０４６３】

行番号４において、２ビット単位の処理が実行するため、被乗数のアドレスｊの変化範囲および増分量が設定される。

【０４６４】

行番号５の命令により、まずポインタレジスタｐ１の指定する被乗数ビットが、ＸＨレジスタおよびＸＬレジスタにそれぞれ格納される。次いで、ポインタレジスタｐ０の指定するメモリセルのデータ、すなわち先のサイクルにおける部分積とポインタレジスタｐ１により指定された被乗数ビットとを用いて、ブース実行命令が行なわれる。この行番号５の命令実行時においては、ポインタレジスタｐ１およびｐ０は、２ビット処理が行なわれるため、そのポインタ値が２増分される（２アドレス増分される）。

【０４６５】

この処理が、ｊが指定する回数繰返し実行され、したがって被乗数ａの全ビットについて、２ビット単位で２次のブースアルゴリズムに従って部分積の生成およびその前のサイクルにおいて生成された部分積との加算が行なわれ、その加算結果が部分積格納領域に格納される。

【０４６６】

行番号６において、この行番号４で示すｆｏｒ文が規定する関数の終了が指定される。１つの乗数ビットの組が完了すると、行番号７において、また、ｆｏｒ文が記述され、２ビット単位での符号拡張処理が指定される。すなわちこの場合、行番号８に示す実行命令に従って、ポインタレジスタｐ０が指定する領域、すなわち、部分積格納領域の上位ビット領域に、符号拡張処理が行なわれ、最上位ビット位置まで符号拡張が行なわれる。

【０４６７】

行番号９において、この行番号７のｆｏｒ文の関数の完了が指定される。符号拡張処理が完了すると、行番号１０の命令に従って、ポインタレジスタｐ３のポインタ２増分される。

【０４６８】

行番号１２において、行番号１のｆｏｒ文の末尾が指定され、したがって、乗数ｂの次の乗数ビットの組を用いて、再び２ビット単位で部分積の生成および前のサイクルの部分

積との加算が実行される。

【０４６９】

この図７９に示すプログラムにおいては、ポインタレジスタｐ１、ｐ２およびｐ３は、それぞれ、２づつ増分される。しかしながら、主演算回路のメモリセルマットにおいては、先に示すように、偶数エントリおよび奇数エントリの同一ビット位置に偶数アドレスおよび奇数アドレスのビットが格納されており、メモリセルマットに対するアドレス制御においては、１ずつエントリ内のビット位置が更新される。

【０４７０】

図８０は、２ビット演算時における単位ＡＬＵ回路ブロック３４の接続を概略的に示す図である。この２ビット演算時、特にブースアルゴリズムに従って乗算を行なう場合、Ｘレジスタ５４は、スイッチ回路ＳＷａを介して内部データ線２２６に結合される。スイッチ回路ＳＷｂは、ＸＬレジスタ５４と内部データ線２２８を切離す状態に設定される。

【０４７１】

スイッチ回路ＳＷｄが、全加算器２１１のＢ入力を内部データ線２２８に結合し、スイッチ回路ＳＷｃは、全加算器２１１のＢ入力と内部データ線２２６とを切離す。スイッチ回路２２５は、全加算器２１０のキャリ出力Ｃｏと全加算器２１０のキャリ入力Ｃｉｎとを分離する。Ｃレジスタ５６は、スイッチ回路２２５を介して全加算器２１０のキャリ入力Ｃｉｎに結合される。ゲート回路２２４の出力は、スイッチ回路ＳＷｆにより内部データ線２２８に結合される。

【０４７２】

この２ビット演算においては、全加算器２１０および２１１が並列に動作し、図７７に示す実行命令（ＥＸＥ命令）の実行時に、加算結果ｚｚを、ビットＡｊ［ｐｘ］およびＡｊ［ｐｘ＋１］について並列に算出する。

【０４７３】

メモリセルマットにおいては、偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏにそれぞれ、偶数アドレスＡ［２ｉ］および奇数アドレスＡ［２ｉ＋１］のデータビットが格納される。ポインタレジスタｐｘにより、これらの偶数エントリＥＲＹａおよび奇数エントリＥＲＹｏの同一ビット位置メモリセルが指定される。したがって、プログラム実行時において、ポインタレジスタをｐｘカウント値が２増分されることにより、奇数エントリＥＲＹｏおよび偶数エントリＥＲＹｅにおいて、１ビットそのビット位置が上位方向にシフトされる。この操作は、単にポインタレジスタｐｘのポインタに基づいて、メモリセルマットのワード線を選択するアドレスが生成されれば、ワード線切換により、ポインタレジスタｐｘのポインタの２増分を実現することができる。

【０４７４】

図８１は、２次のブースアルゴリズムに従う乗算を行なう際のデータの流れを概略的に示す図である。図８１においては、まずブース命令実行時のデータの流れを示す。エントリは偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏに分割される。乗数ｂの最下位ビットアドレスｂｓがポインタレジスタｐ２に設定され、また乗算結果ｃの最下位ビットアドレスｃｓがポインタレジスタｐ３に設定される。図７９に示す行番号２の命令群を実行することにより、ポインタレジスタｐ２が指定する乗数ｂの２ビット（図８１においてはビットｂ１およびｂ０を示す）が、ＸＨレジスタおよびＸＬレジスタに格納される。初期状態においては、ＸレジスタおよびＦレジスタは“０”に初期設定されている。

【０４７５】

この状態で図７７に示す操作内容に従ってＤレジスタ、Ｎレジスタの格納値が決定され、これらのＤレジスタおよびＮレジスタの値の設定後、Ｆレジスタに、ＸＨレジスタの格納ビット（ｂ１）が格納される。これにより、乗数ビットｙ（２ｊ＋１）が、次の部分積生成時、ｙ（２ｊ－１）ビットとして用いられる状態が準備される。

【０４７６】

図８２は、ブース実行命令（ＥＸＥ命令）実行時のデータビットの流れを概略的に示す図である。ポインタレジスタｐ３の格納値がポインタレジスタｐ０に転送され、また被乗

数ａの最下位ビットアドレスａｓがポインタレジスタｐ１に設定される。次いで、被乗数ａの２ビット（ａ１およびａ０）が、それぞれＸＨレジスタおよびＸＬレジスタに格納される。セレクタ（ＳＥＬ；２２７）が、Ｄレジスタの格納内容に従って、ＸＨレジスタおよびＸＬレジスタの一方およびＸレジスタおよびＸＨレジスタの一方を選択する。

【０４７７】

選択反転器（２１７）は、Ｆレジスタの格納値に従ってセレクタ（ＳＥＬ）の出力ビットを選択的に反転して被加算ビットｘ１およびｘ２を生成する。これらは、全加算器２１０および２２５において、ポインタレジスタｐ０のポインタ値に従って読出された部分積からの２ビット（ｃ０，Ｃ１）と加算（２ビット加算）される。この加算結果は、Ｎレジスタの格納ビットに従って選択的に元の位置へ格納される。

【０４７８】

このセレクタ（ＳＥＬ）による選択動作により、２ｊビットシフト時に、レジスタＸＨおよびＸＬレジスタの格納値に従って±１乗算を行なって、対応の部分積と加算して加算結果を元の部分積ビット位置に格納する。２ｊ＋１ビット加算シフト時には、ＸレジスタおよびＸＨレジスタの格納値を用いずに、前のサイクルで読出された被乗数ビットと対応の部分積との加算が行なわれ、結果として２ｊ＋１ビットシフトが実現される。この場合、Ｄレジスタ、ＦレジスタおよびＮレジスタは、１つの部分積全体の生成時において、その値は固定であるため、セレクタ（ＳＥＬ）および選択反転器の選択内容が固定されるため、２ｊビットシフトおよび２ｊ＋１ビットシフトによる部分積の生成とそれまでの部分積との加算を正確に行なうことができる。

【０４７９】

ポインタレジスタｐ１およびｐ０のポインタは、２ずつ増分され、エントリＥＲＹｅおよびＥＲＹｏにおいては、等価的に１ビットずつその位置がシフトされ２ビット単位での部分積生成および前の部分積との加算が実行される。

【０４８０】

この演算の後、図７９に示す行番号８および９の命令により、この生成された部分積の符号拡張が行なわれ、上位ビット位置に、符号ビットが順次格納される。

【０４８１】

この図７０に示すように、ＡＬＵユニット３４において、２つの全加算器２１０および２１１を設けて２ビット加算を行なうことにより、２ビット単位での部分積生成および前の部分積との加算を行なうことができる。

【０４８２】

なお、図７７に示すように、ブース実行命令（ＥＸＥ命令）は、Ｆレジスタの格納値に従って加算または減算を実行することができ、したがって、この実行命令は、加算および減算を包含する命令である。

【０４８３】

加算および減算も２ビット単位で実行することができ、また、１ビット単位で演算することもできる。しかしながら、除算は、被除数のビット位置をづつ右シフトして減算を行う必要があり、１ビット単位で演算を実行する。この１ビット演算を実現するために図８０においてスイッチ回路２２５が設けられている。

【０４８４】

図８３は、１ビット加減算時におけるＡＬＵユニット３４の接続の一例を概略的に示す図である。１ビット演算の接続時においては、Ｘレジスタ５４が、内部データ線２２６および２２８にスイッチ回路ＳＷａおよびＳＷｂを介してそれぞれ接続され、このＸレジスタ５４出力がセレクタ２２７により選択される。スイッチ回路ＳＷａおよびＳＷｂの接続はポインタｐｘにより決定される。

【０４８５】

Ｆレジスタ２０５の格納ビットに従って、選択反転器２１７により加算／減算が実行される。この反転選択器２１７の出力は、全加算器２１１のＡ入力に与えられる。全加算器２１０のＢ入力は、内部データ線２２６に接続される。この全加算器２１０のキャリ出力

Ｃｏが、スイッチ回路２２５により、全加算器２１０のキャリ入力Ｃｉｎと分離され、この全加算器２１０のサム出力Ｓがゲート２２３を介して内部データ線２２６に結合される。全加算器２１０は加算演算には用いられない。全加算器２１１のキャリ入力ＣｉｎがＣレジスタ５６にスイッチ回路２２５を介して結合される。全加算器のＢ入力は、ポインタｐｘによりスイッチ回路ＳＷｃおよびＳＷｄを介して内部データ線２２６または２２８に選択的に結合される。また、全加算器２１１のサム出力Ｓがゲート２２４およびスイッチＳＷｅおよびＳＷｆを介して選択的に内部データ線２２６および２２８に接続される。

【０４８６】

減算演算を２の補数の加算演算により行なう場合には、Ｃレジスタ５６に、初期値として“１”が格納され、Ｘレジスタ５４からのビット値が、選択反転器２１７により反転される。加算演算を行う場合には、Ｃレジスタ５６は、初期状態として、“０”にクリアされる。

【０４８７】

エントリにおいて内部データ線２２６および２２８に接続される領域には連続アドレスのデータビットＡ（２ｉ）およびＡ（２ｉ＋１）が格納されて、内部データ線２２６および２２８を介してＸレジスタ５４にデータを転送する。

【０４８８】

図８４は、１ビット構成のＡＬＵユニット３４を用いて２項加算を行なうプログラムの一例を示す図である。図８４に示す２項加算プログラムにおいて、命令“ｍｅｍ．ｌｄ．Ｃ　０”は、メモリセルマットの特定の領域に格納されるデータビット“０”を、Ｃレジスタに格納する命令である。Ｃレジスタに、セット／リセット機能を持たせる場合、回路構成が複雑となる。メモリセルマットにおいて、リセット用のクリアビットを特定領域に格納し、このクリアビットを用いてＣレジスタをクリア状態に設定する。

【０４８９】

この２項加算演算処理は、演算ａ＋ｂ＝ａが行われる。Ｆレジスタ２０５により、選択反転器２１７が、非反転状態に設定され、ポインタレジスタｐ１が指定する加算数ｂからのビットと、ポインタレジスタｐ０が指定する被加算数ａの対応のビットの加算が行なわれる。加算命令においては、図７６に示すように、このサムが、元の被加算数ａのビット位置へ格納され、加算（ａ＋ｂ）＝ａが実現される。ポインタレジスタｐｘのポインタにしたがってＸレジスタ５４および全加算器の接続を切り換えることにより、偶数エントリおよび奇数エントリのデータビットについて逐次加算を行うことができる。

【０４９０】

図８５は、２項減算のプログラムの一例を示す図である。この２項減算においては、ａ＝（ａ－ｂ）が実行される。図８５に示すプログラムにおいて、命令“ｍｅｎ．ｌｄ．Ｃ１”は、Ｃレジスタに１をセットする命令であり、減算数ｂの２の補数を生成して補数と被減算数ａとの加算を実行する。

【０４９１】

選択反転器２１７は、Ｆレジスタの格納値に従って反転器として設定される。加算結果が、被減算数ａの元の位置に格納される。スイッチ回路ＳＷａ－ＳＷｆの切換は加算演算時と同様ポインタレジスタｐｘのポインタに従って行われる。

【０４９２】

図８６は、１ビット単位での符号なし乗算を行なう際の乗算プログラムの一例を示す図である。被乗数ａ、乗数ｂおよび乗算結果ｃは、それぞれ最下位ビットアドレスは、ａｓ、ｂｓおよびｃｓである。

【０４９３】

ポインタレジスタｐ２およびｐ３それぞれに最下位ビットアドレスｂｓおよびｃｓがセットされる。ついで、Ｎレジスタ２０７に、ポインタレジスタｐ２が指定する乗数ｂの対応のビットを格納し、全加算器２１１のサムＳｕｍ出力を受けるゲート２２４の処理を設定する。ポインタレジスタｐ０にポインタレジスタｐ３の乗算結果格納位置をコピーする。Ｃレジスタがクリアされて、初期状態が設定される。ポインタレジスタｐ１に被乗数ａ

の先頭アドレスをセットして、被乗数ａのビットをＸレジスタ５４にロードする。全加算器２１１を用いて、乗算結果ｃの対応のビットと被乗数ａのビットの加算を行なう。

【０４９４】

この加算結果は、Ｎレジスタ２０７の格納値が“１”のときにのみ、ポインタレジスタｐ０が指定するアドレス位置に格納される。これにより、ビットｂｉとビットａｉとの乗算および部分積との加算を行なうことができる。部分積生成後、Ｃレジスタに格納されたキャリをこの部分積の最上位ビット位置に格納する。

【０４９５】

この乗算時においては、乗数ｂのビットｂｉを固定して、１つの全体の部分積を生成する。１つの全体部分積生成後、この部分積の最下位ビット位置を指定するポインタｐ３を１つ増分して、新たな部分積を指定する。ついで、乗数ｂのビット位置を増分して次の乗算を行なう。これにより１ビットづつ逐次乗算を行うことができる。この乗算時においてもスイッチ回路ＳＷａ－ＳＷｆの接続が、各ポインタの偶数／奇数に従って制御される。

【０４９６】

図８７は、除算を行なうプログラムを示す図である。この除算時においては、ａ／ｂ＝ｃ... ｄが実行される。この図８７に示す除算プログラムにおいて、ｆｏｒ文が規定する関数文の演算命令“ａｌｕ．ｌｅｔ（０ｂ００１１）”により、Ｆレジスタ、Ｄレジスタ、ＮレジスタおよびＣレジスタの初期設定が行なわれる（これらのレジスタに（００１１）が設定される。

【０４９７】

命令“ｍｅｍ．ｓｔ．Ｃ　ｔｐ”は、Ｃレジスタの格納値ｔｐを、メモリの特定の領域に格納する操作を指定する命令である。

【０４９８】

演算命令“ａｌｕ．ｉｎｖ　ｔｐ”は、この特定のビット値ｔｐを反転して再び元のメモリ位置に書込む操作を指定する命令である。

【０４９９】

命令“ｍｅｍ．ｌｄ．Ｎ　ｔｐ”は、この反転された特定ビット値ｔｐを、Ｎレジスタに格納する操作を指定する命令である。

【０５００】

これらの一連の命令の実行により、セレクタＳＥＬが、Ｘレジスタの出力を選択する状態に設定され、選択反転器２１７が、反転を行なう状態に設定され、減算操作“ａｌｕ．ｓｂｃ@ｐ０”が実現される。

【０５０１】

この除算操作時においては、２ビットのデータがＸＨレジスタおよびＸＬレジスタに格納されてついでエントリの剰余格納領域に格納される。この被除数のコピー動作時に２ビット単位でコピーを実行することにより、剰余生成処理を高速化する。ＡＬＵの回路接続は、先の加減演算および乗算演算時と同様、１つの全加算器２１１を用いる状態に設定される。

【０５０２】

次いで、１ビットごとに、Ｘレジスタに除数ｂを格納し、剰余領域の桁合わせされた被除数ａから減算し、減算結果を、Ｎレジスタの格納値に従って選択的に元の剰余格納領域に格納する。この処理が繰返し実行される。

【０５０３】

これにより、除数ｂと被除数ａの大小関係が決定され（Ｃレジスタのキャリｔｐの値が決定され）、この大小関係に基づいて、Ｎレジスタの格納値が設定される。次の減算処理が、Ｎレジスタの格納値に従って選択的に加算操作を用いて実行される。この加算操作により、剰余領域の元の値を選択的に復元する（商ビットが０のときには、ビット位置をずらせて減算処理を行う必要がある）。商のビットとしてＮレジスタの格納値の反転値を設定する。ついで、商および剰余領域のビット位置を１ビット右シフトして、同様の動作を繰返す。

【０５０４】
このビットシリアルに除算を行なう場合においても、全加算器２１１のＢ入力を、スイッチ回路ＳＷｃおよびＳＷｄを用いて選択的に、内部データ線２２６および２２８に接続し、Ｘレジスタ５４をスイッチ回路ＳＷａおよびＳＷｂにより、交互に内部データ線２２６および２２８に接続する。
【０５０５】
したがって、先の加減算操作および乗算操作と同様にして、偶数エントリおよび奇数エントリの格納領域のデータについて、ビットシリアル態様に除算を行なうことができる。
【０５０６】
なお、加算および減算は、全加算器２１０および２１１をともに用いて２ビット加算処理を行うことにより、２ビット単位で加算および減算を行うことができる。
【０５０７】
図８８は、１エントリＥＲＹの偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏにデータを分散して書込む経路の構成の一例を概略的に示す図である。図８８において、転置メモリ２３０は、外部からワードシリアルかつビットパラレル態様で与えられる外部データＥＷを、ビットシリアルかつワードパラレルなメモリデータＭＷに変換してメモリデータバスＭＤＢに転送する。このメモリデータバスＭＤＢには、列選択ゲートＣＳＧｏおよびＣＳＧｅを介して奇数エントリＥＲＹｏおよび偶数エントリＥＲＹｅを構成するビット線対が結合される。列選択ゲートＣＳＧｏおよびＣＳＧｅは、列選択信号ＣＬｏおよびＣＬｅに応答して選択的に導通する。
【０５０８】
これらの列選択信号ＣＬｏおよびＣＬｅを、ポインタレジスタｐｘ（ｘ＝０－３）の最下位ビットｐｘ［０］の偶数／奇数に従って選択状態へ駆動する。これにより、転置メモリ２３０から読出されたワードパラレルかつビットシリアルなメモリデータＭＷの偶数ビットおよび奇数ビットを、それぞれ、偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏに分散させることができる。
【０５０９】
この図８８に示す構成において、偶数エントリＥＲＹｅおよび奇数エントリＥＲＹｏを、それぞれ別々のエントリとして利用する場合には、ポインタレジスタｐｘの最下位ビットｐｘ［０］を、全エントリに対するデータ書込まで、０または１に固定し、全エントリ書込後、このポインタレジスタｐｘの最下位ビットｐｘ［０］を変更する。これにより、偶数エントリ領域にデータワードが書込まれた後、奇数エントリ領域に別のデータワードを書込むことができ、エントリ単位でデータを格納することができる。
【０５１０】
以上のように、この発明の実施の形態１３に従えば、偶数エントリおよび奇数エントリを設け、また演算処理ユニット内に並列動作する全加算器を配置し、２ビット単位で、ＡＬＵ処理を行なっており、高速の演算処理を実現することができる。
【０５１１】
［実施の形態１４］
図８９は、この発明の実施の形態１４に従う半導体信号処理装置の要部の構成を概略的に示す図である。この図８９に示す構成においては、演算処理ユニット群３２に含まれるＡＬＵユニット３４に対し、コントローラ２１からのＡＬＵ制御信号が共通にＡＬＵ制御線を介して与えられる。このＡＬＵユニット（演算処理ユニット）３４は、メモリセルマット３０のエントリＥＲＹそれぞれに対応して配置される。
【０５１２】
この図８９に示す構成においては、メモリセルマット３０に含まれるエントリＥＲＹに対し、同一の演算処理を並行して実行することができる。これにより、シングル・インストラクション・マルチ・データ（ＳＩＭＤ）構成を容易に実現することができる。
【０５１３】
なお、コントローラ２１からのＡＬＵ制御信号は、図１等において示すマイクロ命令メ

モリに格納されるプログラムのデコード結果に基づいて生成される。

【０５１４】

以上のように、この発明の実施の形態１４に従えば、演算処理ユニット群の単位ＡＬＵユニットに共通にＡＬＵ制御線を配設して共通の制御信号を伝達しており、容易に各エントリに対して同一のＡＬＵ演算を実行することができる。

【０５１５】

［実施の形態１５］

図９０は、この実施の形態１５に従う主演算回路の要部の構成を概略的に示す図である。図９０に示す構成においては、メモリセルマット３０が、エントリ方向に沿って２つのサブマット３０ａおよび３０ｂに分割される。演算処理ユニット群３２も、メモリサブマット３０ａおよび３０ｂに対応して、ＡＬＵ群３２ａおよび３２ｂに分割される。ＡＬＵ群３２ａには、コントローラ２１に含まれるＡＬＵ制御回路２４０ａからのＡＬＵ制御信号が、ＡＬＵ制御線２３２ａを介して与えられる。ＡＬＵ群３２ｂのＡＬＵユニット３４に対しては、コントローラ２１に含まれるＡＬＵ制御回路２４０ｂからのＡＬＵ制御信号が、ＡＬＵ制御線２３２ｂを介して与えられる。

【０５１６】

ＡＬＵ制御回路２４０ａおよび２４０ｂは、このコントローラ２１において命令をデコードする命令デコーダの出力信号に従って選択的に活性化され、それぞれ個々に制御信号を生成する。

【０５１７】

この図９０に示す構成においては、演算処理ユニット群３２において、ＡＬＵ群３２ａおよび３２ｂの演算内容を個々に設定することができ、より複雑な演算処理を実現することができる。

【０５１８】

この場合、メモリセルマット３０において、特に同一アドレスのデータビットを、ＡＬＵ群３２ａおよび３２ｂにおいて演算処理することは要求されない。たとえば、サブメモリセルマット３０ａとＡＬＵ群３２ａの間のデータの転送および演算と、サブメモリセルマット３０ｂとＡＬＵ群３２ｂのデータの転送と演算処理を、交互に実行する。たとえば、ＡＬＵ群３２ａにおけるデータ転送時、ＡＬＵ群３２ｂで演算処理を実行する。ＡＬＵ群３２ａの演算処理時、ＡＬＵ群３２ｂでデータ転送を行なう。演算処理サイクル数などの問題により、このサブメモリセルマット３０ａおよび３０ｂにおけるデータビットのアクセス競合が生じた場合、コントローラ２１において、アクセスの仲裁回路を設け、一方のサブメモリセルマットのデータアクセスを先に完了し、次のサイクルで、他方のメモリセルマットに対してデータのアクセスを行なう。これにより、異なるアドレスのデータビットを用いて演算処理を各ＡＬＵ群３２ａおよび３２ｂにおいて実行することができる。

【０５１９】

また、これに代えて、サブメモリマット３０ａおよび３０ｂの一方にアクセスの優先権を与え、プログラム時に、この優先権の与えられるサブメモリマットのアクセスの有無をフラグでモニタするとともに、このモニタ結果に従って他方のサブメモリマットへアクセスを行なう処理を記述することにより、異なるメモリセルサブマット間での同一アドレス位置へのデータアクセスの競合の問題は、回避することができる。

【０５２０】

以上のように、この発明の実施の形態１５に従えば、ＡＬＵ回路、複数のグループに分割し、各ＡＬＵ群個々に、ＡＬＵ制御信号線を配設しており、各ＡＬＵ群で異なる演算処理を実行することができ、複雑な演算処理を高速に行なうことができる。

【０５２１】

［実施の形態１６］

図９１は、この発明の実施の形態１６に従う半導体装置の要部の構成を概略的に示す図である。図９１において、内部データバス１２（グローバルデータバスＧＢＳ）に基本演算ブロックＦＢ１－ＦＢｎが並列に結合される。このグローバルデータバスＧＢＳに対し

、大容量メモリ２５０が接続される。グローバルデータバスＧＢＳは、入出力回路１０を介して外部に設けられたシステムバス５に結合される。

【０５２２】

大容量メモリ２５０は、たとえば、ダイナミック・ランダム・アクセス・メモリ（ＤＲＡＭ）であり、たとえば１枚の画像のデータを格納することのできる記憶容量を有し、数Ｍビットから数十Ｍビットの記憶容量を有する。

【０５２３】

この大容量メモリ２５０へは、入出力回路１０を介して外部に設けられたホストＣＰＵから演算処理データが格納される。この大容量メモリ２５０におけるデータの格納態様については後に詳細に説明する。基本演算ブロックＦＢ１－ＦＢｎが、グローバルデータバスＧＢＳを介して大容量メモリ２５０とデータを転送する。グローバルデータバスＧＢＳのデータ線は、チップ上配線であり、多くのビット幅を有することができる。したがって、この大容量メモリ２５０と基本演算ブロックＦＢ１－ＦＢｎのいずれかとの間でのデータ転送のバンド幅を大きくすることができ、このデータ転送に要する時間が演算処理に対するボトルネックとなるのを防止することができる。

【０５２４】

図９２は、大容量メモリ２５０と１つの基本演算ブロックＦＢｉとの間のデータ転送経路を概略的に示す図である。主演算回路２０において、メモリセルマット３０Ａおよび３０Ｂの間に、演算処理ユニット群（ＡＬＵユニット群）３２が設けられる。これらのメモリセルマット３０Ａおよび３０Ｂが、グローバルデータバスＧＢＳを介して大容量メモリ２５０とデータ転送を行なうことができる。主演算回路２０においては、メモリセルマット３０Ａおよび３０Ｂとグローバルデータバスの間のインタフェースとなる入出力回路は設けられているが、この入出力回路は図９２においては示していない。

【０５２５】

グローバルデータバスＧＢＳのバス幅が、メモリセルマット３０Ａおよび３０Ｂのエントリ数に等しい場合、たとえば、これらのメモリセルマット３０Ａおよび３０Ｂの１行のデータビットを大容量メモリ２５０との間で転送することができる。この場合、大容量メモリ２５０の入出力データビットが、メモリセルマット３０Ａおよび３０Ｂの１つのエントリのビット幅と同じであってもよい。１つのエントリの内容を１回のデータ転送サイクルで転送することができる（この場合には、メモリセルマットにおいてデュアルポートメモリセルを利用する）。

【０５２６】

図９３は、大容量メモリ２５０と１つのメモリセルマット３０の間のデータ転送を行なう部分の構成の一例を示す図である。大容量メモリ２５０のデータアクセスの制御は、図１に示す集中制御ユニット内の制御ＣＰＵ２５により行なわれる。メモリセルマット３０のデータの入出力は、対応の基本演算ブロック内に含まれるコントローラ２１により行なわれる。制御ＣＰＵ２５およびコントローラ２１の間で、データ転送要求ＲＥＱおよびデータ転送了解ＡＣＫを転送して、データ転送が行なわれる。

【０５２７】

大容量メモリ２５０が、一例として２ポート構成であり、入出力回路１０とポートＥＸＰを介してデータ転送を行ない、また、大容量メモリ２５０が、内部ポートＩＮＰを介してメモリセルマット３０とデータ転送を行なう。したがって、入出力回路１０から外部ポートＥＸＰを介してデータＥＤＴが格納され、ポートＩＮＰを介してそれと直交する方向に整列するデータブロックＴＲＤがメモリセルマット３０に対して転送される。メモリセルマット３０においては、エントリＥＲＹ０－ＥＲＹｍが設けられ、これらのエントリＥＲＹ０－ＥＲＹｍの同一ビット位置のデータのブロックＸＲＤが大容量メモリ２５０との間で転送される。

【０５２８】

したがって、大容量メモリ２５０とメモリセルマット３０の間でのデータ転送時、データブロックＴＲＤおよびＸＲＤがそれぞれ１つのワード線に接続するメモリセルデータの

場合、１つのワード線選択を行うだけで、データブロックＴＲＤまたはＸＲＤを転送することができる。

【０５２９】

大容量メモリ２５０は、外部からは、ポートＥＸＰを介してワード単位でデータの格納が行なわれてもよい。データブロックＥＤＴが、このポートＥＸＰのワード線方向に対応する。従って、大容量メモリ２５０が、データの転置機能を備える。外部バスのインターフェイスの入出力回路が、転置機能を備えている場合には、この大容量メモリ２５０は、データの転置機能を持つ必要はない。デュアルポート構成で、入出力回路１０と大容量メモリ２５０との間のデータ転送バスと大容量メモリと基本演算ブロックとの間のデータ転送バスとが別々のバスであれば、大容量メモリと外部とのデータ転送と機能ブロックとの間のデータ転送を並行して実行することができる。このデュアルポート構成の場合、ポートＩＮＰのビット幅は、メモリセルマット３０のエントリ数（ｍ＋１）またはグローバルデータバスのビット幅となる。

【０５３０】

しかしながら、大容量メモリ２５０は、シングルポートメモリであっても良く、この場合には、大容量メモリ２５０への外部からのデータ転送完了後に、大容量メモリ２５０と基本演算ブロックとの間のデータ転送が行われる。また、大容量メモリ２５０と基本演算ブロックＦＢ１‐ＦＢｎとの間のデータ転送において、１つの基本演算ブロックの演算処理実行時に大容量メモリと別の基本演算ブロックとの間でデータ転送が行われ、データ転送と演算とがインターリーブ態様またはパイプライン的に実行されても良い。

【０５３１】

図９４は、この大容量メモリ２５０とメモリセルマット３０の間のデータ転送の別の態様を概略的に示す図である。図９４において、大容量メモリ２５０は、シングルポートメモリであり、グローバルデータバスＧＢＳに結合される。主演算回路２０のメモリセルマット３０は、デュアルポートメモリであり、ポートＡを介してグローバルデータバスＧＢＳに結合され、またポートＢを介してグローバルデータバスＧＢＳに結合される。ポートＡを介して大容量メモリ２５０とエントリＥＲＹ単位でのデータ転送を行なう。ポートＢは、グローバルデータバスＧＢＳの特定のバスを利用して、外部のホストＣＰＵと入出力回路１０を介してデータ転送を行なう。

【０５３２】

この図９４に示す構成において、大容量メモリ２５０は、外部からのデータＥＤＴを各ワード線ごとに格納する場合、このワード線上の複数ワードのデータＥＤＴをグローバルデータバスＧＢＳを介し、メモリセルネット３０へポートＡから転送することにより、エントリＥＲＹのデータを、一括してメモリセルマット３０に転送することができる。

【０５３３】

ポートＢは、入出力回路１０と直接データの転送を行う場合に利用される。入出力回路１０は、この場合、グローバルデータバスＧＢＳと同一のビット幅を有することは特に要求されない。ポートＢが、入出力回路１０とメモリセルマットの入出力ビット幅の調整を行う。ポートＢを利用する入出力回路とのデータ転送時には、入出力回路１０においてはデータの位置を交換する転置回路が必要とされる。

【０５３４】

この図９４に示す構成の場合には、主演算回路のメモリセルマットが２ポート構成であるものの、ポートＡが大容量メモリ２５０および入出力回路１０とのデータ転送に用いられ、メモリセルのポートＢがＡＬＵ群３２とのデータ転送にのみ利用される場合には、特に、主演算回路２０外部にデータの転置を行なうための構成を設ける必要がない。外部のＣＰＵに対するデータは、ポートＡを介して格納することができる。

【０５３５】

また、この図９４に示す構成における大容量メモリ２５０と主演算回路２０のメモリセルマット３０の間のデータ転送は、図９３に示す構成と同様、制御ＣＰＵ２５とコントローラ２１とを用いて内部のアドレスポインタ（ポインタレジスタ）を参照して行なわれる

。
【０５３６】
以上のように、この発明の実施の形態１６に従えば、複数の基本演算ブロックに対し共通に大容量メモリを設け、大きなビット幅の内部データバスを介して、選択された基本演算ブロックと大容量メモリの間でデータ転送を行なっており、データ転送に要する時間を短縮することができ、高速演算処理を実現することができる。
【０５３７】
［実施の形態１７］
図９５は、この発明の実施の形態１７に従う半導体演算装置の要部の構成を概略的に示す図である。この図９５に示す半導体演算装置においては、内部データバス（グローバルデータバス）ＧＢＳに結合される大容量メモリ２５０が、複数のバンクＢＫ０－ＢＫｋに分割される。これらのバンクＢＫ０－ＢＫｋの各々は、基本演算ブロックＦＢ１－ＦＢｎに含まれるメモリセルマットと同程度の記憶容量を有する。したがって、大容量メモリ２５０の記憶容量は、（バンク数）・（メモリマットの容量）となる。
【０５３８】
図９６は、大容量メモリ２５０と１つの基本演算ブロックＦＢｉの主演算回路２０との接続関係を概略的に示す図である。主演算回路２０において、メモリセルマット３０に係数データを格納する。大容量メモリ２５０において、バンクＢＫ０－ＢＫｋ各々に、画像データが格納され、グローバルデータバスＧＢＳを介してＡＬＵユニット３２とデータ転送を行なう。この場合には、グローバルデータバスＧＢＳのビット幅は、メモリセルマット内のエントリ数に等しくなる。
【０５３９】
画像処理においては、フィルタ処理などを行なう場合、係数と演算処理データとの乗算が実行される。この場合、処理対象のデータと演算処理に必要とされる係数データとでは、係数データよりも処理データの方が圧倒的に数が多い。このため、比較的小容量のメモリセルマット３０に係数データを格納し、処理対象の画像データを、大容量メモリ２５０のバンクＢＫ０－ＢＫｋにそれぞれ分散して格納する。
【０５４０】
演算処理時においては、バンクＢＫ０－ＢＫｋをバンクセレクト信号（図示せず）に従って選択して、選択バンクのデータを演算処理群３２のＡＬＵ群にデータを伝達する。処理完了後の画像データは、外部へ転送し、また新たなデータが対応のバンクに格納される。この外部との間のデータ転送時、別のバンクが選択され、基本演算ブロックＦＢｉにおいて並列演算処理が実行される。
【０５４１】
図９７は、この発明の実施の形態１７に従う半導体演算装置の要部の構成を概略的に示す図である。図９７に示す構成においては、大容量メモリ２５０が、ポートＡ回路２５２ａおよびポートＢ回路２５２ｂと、複数のバンクＢＫ０－ＢＫｋを含む。内部データバス（グローバルデータバス）は、入出力回路１０に結合される内部データバスＧＢＳａと、大容量メモリ２５０のポートＢ回路２５２ｂと主演算回路２０の演算処理ユニット群３２とに結合されるグローバルデータバスＧＢＳｂを含む。グローバルデータバスＧＢＳｂは、演算処理ユニット群３２の各ＡＬＵユニットそれぞれに対して並列にデータビットを転送することのできるビット幅を有する。主演算回路の入出力回路（図示せず）は、内部データバスＧＢＳａに結合される。この内部データバスＧＢＳａは、基本演算ブロックのマイクロ命令メモリへのプログラムデータおよび他のコントローラ２１に対する制御情報等を転送する。
【０５４２】
大容量メモリ２５０のアクセス制御が、メモリコントローラ２５５により行なわれ、主演算回路２０の操作制御は、基本演算ブロック内のコントローラ２１により行なわれる。コントローラ２１は、メモリコントローラ２５５に対してロード／ストアの命令を転送する。メモリコントローラ２５５およびコントローラ２１は、個々に、アドレスポインタを

生成する。

【０５４３】

図９８は、図９７に示すメモリ構成の動作を示すタイミング図である。この図９８に示すように、まず外部のホストＣＰＵからの指令が与えられ、コントローラ２１の制御のもとに、メモリセルマット３０に対し、データが格納される。次いで、外部のホストＣＰＵの指示によりメモリコントローラ２５５が起動され、ホストＣＰＵの制御の下に外部のメモリから入出力回路１０を介してバンクＢＫ０－ＢＫｋに対し処理データのロードが行われる。この大容量メモリ２５０へのデータのロードは、ＤＭＡモードで図１に示すＤＭＡ４の制御のもとに実行されてもよい。

【０５４４】

メモリセルマット３０およびバンクＢＫ０－ＢＫｋへのデータのロードが完了すると、コントローラ２１が、演算処理を開始する。メモリセルマット３０およびバンク＃０（ＢＫ０）のデータについて演算処理が行なわれ、演算結果がバンクＢＫ０に格納される。１つのバンクの処理が完了すると、バンクを切換えて次のバンク＃１、＃２、…のデータの処理が実行される。バンク＃０（ＢＫ０）の処理後のデータは、ポートＡ回路２５２ａを介してメモリコントローラ２５５からの起動を受けたＤＭＡ回路４により、ＤＭＡモードで外部メモリに対して転送され、また、新たな画像データが、このバンク＃０（ＢＫ０）へ格納される。

【０５４５】

したがって、１つのバンクに対し、ポートＡ回路２５２ａを介してデータをストアする処理に時間を要しても、他バンクの演算処理が終了し、次にこのバンクの処理が開始されるまでに、このバンク＃０への新たなデータのロードが完了していることが要求されるだけである。ポートＡ回路２５２ａのビット幅が、内部データバスＧＢＳａのビット幅により制限されるものの、内部データバスＧＢＳａおよびＧＢＳｂは別々に設けられており、外部メモリと大容量メモリ２５０との間でのデータ転送は、何ら主演算回路におけるデータの演算処理に悪影響を及ぼさない。

【０５４６】

また、大容量メモリ２５０の各バンクＢＫ０－ＢＫｋは、グローバルデータバスＧＢＳｂを介して演算処理ユニット群３２とデータを転送しており、各演算サイクルごとに、必要とされるデータを転送でき、高速演算処理を実現することができる。

【０５４７】

以上のように、この発明の実施の形態１７に従えば、大容量メモリをマルチバンク構成とし、１つのバンクを主演算回路のメモリセルマットとみなして演算処理を実行しており、画像データの処理後の画像データおよび新たな画像データの転送をパイプライン的に実行でき、高速演算処理が実現される。

【０５４８】

なお、主演算回路２０において、メモリセルマット３０は、デュアルポート構成であってもよく、また、シングルポート構成であってもよい（ただしシングルポート構成の場合には、メモリセルマット３０への書込データは、ビット位置変更処理を受ける）。

【０５４９】

［実施の形態１８］

図９９は、この発明の実施の形態１８に従う半導体演算装置の要部の構成を概略的に示す図である。この図９９においては、内部データバス（グローバルデータバス）ＧＢＳに複数の基本演算ブロックが並列に結合される。図９９において、これらの基本演算ブロックに含まれる主演算回路（２０）ＭＰＡ０－ＭＰＡ３を代表的に示す。これらの主演算回路（２０）ＭＰＡ０－ＭＰＡ３の各々は、メモリセルマット３０Ａおよび３０Ｂとこれらのメモリセルマット３０Ａおよび３０Ｂの間に配設されるＡＬＵ群を含む。このＡＬＵ群は、図９９においては示していない。ＡＬＵ群に対しては、ＡＬＵユニットを相互接続するためのＡＬＵ間相互接続用スイッチ回路（４４）ＥＣＭが設けられる。

【０５５０】

これらのＡＬＵ間相互接続用スイッチ回路（４４）は、以下の説明においては、符号“ＥＣＭ”を用いて参照する。

【０５５１】

メモリセルマット３０Ａおよび３０Ｂは、各々、複数のエントリＥＲＹに分割されており、これらのエントリＥＲＹそれぞれに対して、ＡＬＵ間接続回路（６５）が設けられる。

【０５５２】

図１において示すように、基本演算ブロック間においては、隣接ブロック間データバス１６が設けられる。この隣接ブロック間データバス１６は、隣接基本演算ブロックのＡＬＵ間接続回路（図７の回路６５）を相互接続する隣接ブロックエントリ相互接続線２６０を含む。この隣接ブロックエントリ相互接続線２６０は、隣接する基本演算ブロックの主演算回路ＭＰＡの同一位置のエントリを相互接続する。

【０５５３】

グローバルデータバスＧＢＳは、入出力回路１０を介して外部のシステムバス５に結合される。

【０５５４】

図１００は、この隣接ブロックエントリ相互接続線２６０の具体的接続態様を示す図である。図１００において、基本演算ブロックＦＢｉおよびＦＢ（ｉ＋１）において、ビット線方向（エントリの延びる方向）に平行に隣接ブロックエントリ接続線２６０が延在して配置され、同一列のエントリＥＲＹｊに対して設けられるＡＬＵユニットＡＬＵｊを相互接続する。この隣接ブロックエントリ相互接続線２６０は、このエントリの延在方向、すなわちビット線と同一方向にメモリセルマット上を延在して配置され、最短距離で、隣接基本演算ブロックＦＢｉおよびＦＢ（ｉ＋１）の同一列のエントリの単位ＡＬＵ回路ブロック（ＡＬＵユニット）ＡＬＵｊを相互接続する。

【０５５５】

図９９に示すグローバルデータバスＧＢＳは、各基本演算ブロックの入出力回路（図１５参照）を介して対応のセンスアンプ群およびライトドライバ群に結合される。主演算回路ＭＰＡ０－ＭＰＡ３それぞれにおいて、センスアンプ群およびライトドライバ群と入出力回路を接続する相互配線（メモリ内部配線）は、これらのセンスアンプ群およびライトドライバ群の上層配線またはメモリセルマット上層配線により形成され、複数ビットのデータを並列に転送する。

【０５５６】

隣接ブロックエントリ相互接続線２６０は、図１００においては、ＡＬＵ間を接続している。しかしながら、この隣接ブロックエントリ間相互接続線２６０は対応のＡＬＵユニットＡＬＵｊを介して対応のＡＬＵ間接続回路６５に結合される。したがって、この隣接ブロックエントリ接続線２６０は、またＡＬＵ間接続回路６５に直接結合されてもよい。このＡＬＵ間接続回路６５は、ＡＬＵユニット内のＸレジスタまたはＺレジスタに結合される。

【０５５７】

［変更例１］

図１０１は、この発明の実施の形態１８の変更例１の隣接ブロック間データバスの配置を概略的に示す図である。図１０１において、グローバルデータバスＧＢＳに並列に、基本演算ブロックに含まれる主演算回路ＭＰＡ０－ＭＰＡｋが結合される。これらの主演算回路ＭＰＡ０－ＭＰＡｋ各々において、メモリセルマット３０Ａおよび３０Ｂの間にＡＬＵ間接続用スイッチ回路ＥＣＭが配置される。隣接ブロックエントリ接続線２６０により、ＡＬＵ間接続用スイッチ回路ＥＣＭ内の隣接する主演算回路の同一位置のエントリに対するＡＬＵ間接続回路（６５）が接続される。

【０５５８】

この図１０１に示す構成において、さらに、最も遠く離れた基本演算ブロックの主演算回路ＭＰＡ０およびＭＰＡｋの同一位置のエントリに対して設けられるＡＬＵユニットお

よびＡＬＵ間接続回路が、フィードバック配線２６２により相互接続される。これにより、主演算回路ＭＰＡ０－ＭＰＡｋの同一位置のエントリのＡＬＵユニットが、リング状に相互接続される。

【０５５９】

このリング状に、各基本演算ブロックの同一位置のエントリに対するＡＬＵユニットおよびＡＬＵ間接続回路を相互接続することにより、任意の位置の主演算回路間でデータ転送を行なうことができる。

【０５６０】

なお、図１０１においては、このフィードバック配線２６２は、主演算回路ＭＰＡ０－ＭＰＡｋのメモリセルマット上部を直線的に延在するように示される。しかしながら、このフィードバック配線２６２は、主演算回路（メモリセルマット）外部を迂回してフィードバックループを形成するように配置されてもよい。

【０５６１】

図１０２は、このリング状隣接ブロック間データバスの構成の他の例を概略的に示す図である。図１０２において、グローバルデータバスＧＢＳの一方側に主演算回路ＭＰＡ０およびＭＰＡ１が配置され、このグローバルデータバスＧＢＳの他方側に、主演算回路ＭＰＡ２およびＭＰＡ３が配置される。グローバルデータバスＧＢＳに関して同一側に配置される隣接主演算回路ＭＰＡ０およびＭＰＡ１は、同一位置のエントリのＡＬＵユニットが隣接ブロックエントリ相互接続バスＮＢＡａを介して相互接続される。この隣接ブロックエントリ相互接続バスＮＢＡａは、隣接する主演算回路ＭＰＡ０およびＭＰＡ１の同一位置のエントリＥＲＹに対し設けられるＡＬＵユニットおよびＡＬＵ間接続回路（６５）を相互接続する隣接ブロックエントリ接続線２６０を含む。

【０５６２】

また、グローバルデータバスＧＢＳの他方側において配置される隣接主演算回路ＭＰＡ２およびＭＰＡ３においても、隣接ブロックエントリ接続バスＮＢＡｂが配置される。この隣接ブロックエントリ相互接続バスＮＢＡｂは、主演算回路ＭＰＡ２およびＭＰＡ３の同一位置のエントリＥＲＹに対して設けられるＡＬＵユニットおよびＡＬＵ間相互接続回路を接続する隣接ブロックエントリ相互接続線２６０を含む。

【０５６３】

この隣接ブロック間相互接続バス（隣接ブロック間データバス１６）は、さらに、グローバルデータバスＧＢＳに対して対向する位置の主演算回路ＭＰＡ０およびＭＰＡ２の対応するエントリＥＲＹに対して設けられるＡＬＵおよびＡＬＵ間相互接続回路を接続する隣接ブロックエントリ相互接続バスＮＢＢａにより相互接続される。この隣接ブロックエントリ相互接続バスＮＢＢａは、主演算回路ＭＰＡ０およびＭＰＡ２の対応の位置のエントリＥＲＹに対して設けられるＡＬＵユニットおよびＡＬＵ間接続回路を相互接続する配線２６２ａを含む。

【０５６４】

同様に、主演算回路ＭＰＡ１およびＭＰＡ３の対応の位置のエントリＥＲＹに対して設けられるＡＬＵユニットおよびＡＬＵ間接続回路が、隣接ブロック相互接続バスＮＢＢｂにより相互接続される。この隣接ブロックエントリ相互接続バスＮＢＢｂは、各ＡＬＵユニットおよびＡＬＵ間接続回路に対して配設される配線２６２ｂを含む。

【０５６５】

したがってこの図１０２に示すように、行列状に整列して基本演算ブロック（主演算回路ＭＰＡ０－ＭＰＡ３）が配設される場合においても、グローバルデータバスの一方側の両端の主演算回路およびグローバルデータバスの他方側の両端の基本演算ブロック（主演算回路）のＡＬＵユニットおよびＡＬＵ間接続回路を相互接続することにより、これらの主演算回路ＭＰＡ０－ＭＰＡ３のＡＬＵユニットをリング状に相互接続することができ、任意の主演算回路間でデータの転送を行なうことができる。

【０５６６】

特に、この図１０２に示すように、各隣接ブロック相互接続バスＮＢＢａおよびＮＢＢ

ｂにおいて、対向して配置される主演算回路の同一位置のエントリＥＲＹに対して設けられるＡＬＵユニットおよびＡＬＵ間相互接続回路を配線２６２ａおよび２６２ｂにより相互接続することにより、相互接続配線２６２ａおよび２６２ｂは、それぞれのバスＮＢＢａおよびＮＢＢｂにおいて同じ長さとなり、信号伝搬遅延を同一とすることができ、信号のスキューを低減でき、高速転送を実現することができる。

【０５６７】

また、この隣接ブロックエントリ相互接続バスＮＢＢａおよびＮＢＢｂは、それぞれメモリセルマット３０Ａおよび３０Ｂ上に延在される第１の配線部分と、グローバルデータバスＧＢＳに関して対向する主演算回路のメモリセルマット上を第１の配線部分と直交する方向に延在する第２の配線部分とで構成し、これらの第１の配線部分および第２の配線部分をメモリセルマット上で相互接続することにより、グローバルデータバスに関して対向するメモリマット間の相互接続配線をコンパクトに配置することができ、配線面積増大によるメモリアレイ面積の増大を抑制することができる。

【０５６８】

以上のように、この発明の実施の形態１８に従えば、隣接する基本演算ブロックの主演算回路の各対応のエントリを内部配線で相互接続しており、特にリング状に相互接続することにより、グローバルデータバスＧＢＳを介することなく、主演算回路間で高速でデータ転送を行なうことができる。

【０５６９】

なお、図１０２においては、ＡＬＵ間相互接続用スイッチ回路ＥＣＭの内部構成を示していない。このＡＬＵ間相互接続用スイッチ回路ＥＣＭにおいては、ＡＬＵユニットおよびＡＬＵ間接続回路が含まれており、各ＡＬＵユニットおよびＡＬＵ間接続回路が、対応の配線２６０または２６２ａ，２６２ｂにより接続される。

【０５７０】

［実施の形態１９］

図１０３は、この発明の実施の形態１９に従うＡＬＵ間相互接続用スイッチ回路（４４）ＥＣＭの接続の態様を概略的に示す図である。図１０３においては、一例として、１つの主演算回路において、８個の単位ＡＬＵ回路ブロック（３４）ＡＬＵ０－ＡＬＵ７が設けられる。この単位ＡＬＵ回路ブロックは、演算回路および各レジスタ回路を含み、内部構成は、実現する演算内容に応じて適宜決定される。この単位ＡＬＵ回路ブロックは、先の実施の形態におけるＡＬＵユニット（演算処理ユニット）に対応する。以下では、説明の煩雑を避けるために、単に「ＡＬＵ」で、この演算処理ユニット（ＡＬＵユニット）を参照する。

【０５７１】

これらの８個のＡＬＵに対して、ＡＬＵ間相互接続用スイッチ回路ＥＣＭにおいては、ＡＬＵ間を１ビットシフト（ｍｏｖｅ）するスイッチ回路および配線を配置する１ビットシフト領域ＡＲ０と、２ビットシフトするスイッチ回路および配線を配置する２ビットシフト領域ＡＲ１と、４ビットシフトするスイッチ回路および配線を配置する４ビットシフト領域ＡＲ２とが設けられる。これらのシフト領域ＡＲ０、ＡＲ１、およびＡＲ２においては、１つの基本演算ブロック内において、図の縦方向に１ビット、２ビットまたは４ビットシフトする位置のＡＬＵ間でデータを転送する。２の乗数ビット、ＡＬＵ間のデータ転送（シフト）を実現することにより、任意のＡＬＵ間のデータ転送を２サイクルで行なうことができる。

【０５７２】

ただし、各データシフトのために配線領域が必要となる。最大２のｎ乗のＡＬＵ間のデータシフトを行なう場合、１ビットシフト領域から２のｎ乗ビットシフト領域まで、合計（ｎ＋１）の配線領域が必要となる。

【０５７３】

図１０４は、このシフト領域ＡＲ０－ＡＲ２において設けられるＡＬＵ間接続回路の構成の一例を示す図である。この図１０４に示すＡＬＵ間接続回路が、各ＡＬＵに対応して

設けられる。この図１０４に示すＡＬＵ間接続回路は、図７に示すＡＬＵ間接続回路６５に対応する。

【０５７４】

図１０４において、ＡＬＵ間接続回路（６５）は、上方シフト指示信号ＵＰおよび下方向シフト指示信号ＤＷに従って選択的に活性化される送受信レジスタ２７０と、ｋビットシフト指示信号ＥＮｋに従って選択的に導通し、送受信レジスタ２７０をｋビット先のＡＬＵ間接続回路に接続するトランスファーゲート２７２および２７３を含む。ｋビット転送指示信号ＥＮｋの活性化により、２のｋ乗ビットのデータシフトが実現される（ｋ＝０、１、…）。

【０５７５】

送受信レジスタ２７０は、シフト指示信号ＵＰおよびＤＷによりデータの送信および受信方向が決定される（送信レジスタおよび受信レジスタの接続が決定される）。この送受信レジスタ２７０は、対応のＡＬＵ間接続回路内の全ｋビットシフトスイッチ（トランスファーゲート）２７２および２７３に共通に配置される。

【０５７６】

この図１０４に示すように、送受信レジスタ２７０を配置することにより、双方向に、ＡＬＵ間でデータビットを転送することができる。この送受信レジスタ２７０が、対応のＡＬＵのＸレジスタに接続される（図７に示す構成の場合）。ＡＬＵ内にＸレジスタおよびＺレジスタが存在し、Ｘレジスタが受信レジスタとして用いられ、かつＺレジスタが送信レジスタとして用いられる場合には（図４９参照）、この送受信レジスタ２７０は、特に設けられなくても良い。

【０５７７】

また、図１０３に示す構成において、１ビットシフト領域ＡＲ０においては、ＡＬＵ０およびＡＬＵ７が相互接続される（スイッチ回路を介して）。これにより、リング状に、双方向に、データビットを同一主演算回路ブロック内において転送することができる。

【０５７８】

［変更例１］

図１０５は、この発明の実施の形態１９の変更例１の構成を概略的に示す図である。図１０５において、ＡＬＵ間相互接続用スイッチ回路ＥＣＭにおいて、ＡＬＵ間を１ビットシフトする１ビットシフト領域ＡＲ０と、２ビットまたは４ビットシフトする配線／スイッチが配置される２／４ビットシフト領域ＡＲ３が設けられる。１ビットシフト領域ＡＲ０においては、先の図１０３に示すように、隣接ＡＬＵ間でデータ転送を行なうことができる。

【０５７９】

一方、２／４ビットシフト領域ＡＲ３においては、プログラマブルスイッチ回路ＰＳＷにおいて、２ビットシフトまたは４ビットシフトが択一的に実現される。この図１０５に示す構成の場合、２ビットシフトおよび４ビットシフトを切換えるために、プログラマブルスイッチ回路ＰＳＷが必要となるものの、転送（ｍｏｖｅ）バス配線数を低減でき、バスの占有面積を低減できる。

【０５８０】

この図１０５に示すプログラマブルスイッチ回路ＰＳＷは、図１０３に示す２ビットシフト領域ＡＲ１における各配線間に配置される。

【０５８１】

図１０６は、このプログラマブルスイッチ回路ＰＳＷの構成の一例を概略的に示す図である。図１０６において、プログラマブルスイッチ回路ＰＳＷは、イネーブル信号ＥＮａに従って配線２７５ａおよび２７５ｂを選択的に分離する転送ゲート２８０と、イネーブル信号ＥＮｂに従って選択的に配線２７５ａを送受信レジスタ２７０の一方のノードに接続するトランスファーゲート２８１と、イネーブル信号ＥＮｃに従って配線２７５ｂを送受信レジスタ２７０の他方ノードに接続する転送ゲート２８２を含む。送受信レジスタおよびプログラマブルスイッチ回路ＰＳＷによりＡＬＵ間接続回路（６５）が構成される

。
【０５８２】
送受信レジスタ２７０は、送信レジスタおよび受信レジスタを含み、上向き矢印に従って上方向シフト、下向き矢印に従って下方向にデータを転送する。この送受信レジスタ２７０は、対応のＡＬＵに含まれるたとえばＸレジスタに結合される。
【０５８３】
この図１０６に示すプログラマブルスイッチ回路ＰＳＷにおいて転送ゲート２８０－２８２を選択的に導通／非導通状態に設定することにより、２ビットシフトおよび４ビットシフトを選択的に実現することができ、また上方向および下方向の双方向にデータビットの転送を行なうことができる。
【０５８４】
切換制御信号ＥＮａ－ＥＮｃは、コントローラ（２１）からのプログラムのデコード結果に基づいて生成されるＡＬＵ制御信号に含まれる。
【０５８５】
図１０７は、このプログラマブルスイッチ回路ＰＳＷの接続状態の１つである分断状態の接続を概略的に示す図である。図１０７において、プログラマブルスイッチ回路ＰＳＷの分断状態ＩＴＰにおいては、転送ゲート２８１および２８２が導通状態となり、転送ゲート２８０が非導通状態となる。この場合、接続配線２７５ａおよび２７５ｂは、転送ゲート２８０により分断され、対応のＡＬＵ間接続回路（６５）を介してデータの転送が行なわれる。データの転送方向は、送信および受信レジスタの接続状態により決定される。
【０５８６】
図１０８は、プログラマブルスイッチ回路ＰＳＷの接続状態のうちのスルー状態の接続を概略的に示す図である。このスルー状態ＴＨＲにおいては、転送ゲート２７０が導通状態に設定され。転送ゲート２８１および２８２が非導通状態に設定される。したがって、接続配線２７５ａおよび２７５ｂが相互接続され、対応のＡＬＵユニットから分離される。従って、このスルー状態ＴＨＲにおいては、接続配線２７５ａおよび２７５ｂ上のデータビットは、対応のＡＬＵを通過して別のＡＬＵへ転送される。このスルー配線により、４ビット転送時、２ビット先のＡＬＵユニットを通過して、４ビット先のＡＬＵユニットへのデータ転送を実現する。
【０５８７】
図１０９は、プログラマブルスイッチ回路ＰＳＷの接続状態のうちの分断／ダミー状態ＩＲＤの接続を概略的に示す図である。この分断／ダミー状態ＩＲＤにおいては、転送ゲート２８０および２８２が非導通状態に設定され、転送ゲート２８２が導通状態に設定される。したがって、接続配線２７５ｂが対応のＡＬＵに接続され、接続配線２７５ａは対応のＡＬＵから分離される。接続配線２７５ａが、対応のＡＬＵとデータのビットの転送を行ない、接続配線２７５ｂは、データ転送に寄与せず、ダミー配線として配置される。
【０５８８】
図１１０は、２ビット／４ビットシフト領域ＡＲ３のプログラマブルスイッチ回路ＰＳＷの接続の一例を示す図である。図１１０において、スイッチ回路ＰＳＷａが分断状態ＩＴＰに設定され、スイッチ回路ＰＳＷｂがスルー状態ＴＨＲに設定される。したがって、ＡＬＵ（ＡＬＵユニット）０、ＡＬＵ２、ＡＬＵ４、およびＡＬＵ６の間でプログラマブルスイッチ回路ＰＳＷａおよびＰＳＷｃによりデータ転送が行なわれ、またＡＬＵ１、ＡＬＵ３、ＡＬＵ５およびＡＬＵ７の間でも、分断状態の。プログラマブルスイッチ回路ＰＳＷａによりデータ転送が行なわれる。同様、スルー状態のプログラマブルスイッチ回路ＰＳＷｂにより、ＡＬＵ６とＡＬＵ０の間およびＡＬＵ１とＡＬＵ７の間でデータ転送が行なわれる。このスイッチの接続により、ＡＬＵ０－ＡＬＵ７において２ビットシフト動作が実現される。
【０５８９】
ＡＬＵ６において、スイッチ回路ＰＳＷｃが、分断・ダミー状態に設定されている。ＡＬＵ６は、ＡＬＵ０とプログラマブルスイッチ回路ＰＳＷｂを介して接続される。従って

、このＡＬＵ６に対して配置されるプログラマブルスイッチ回路ＰＳＷｃは、さらに下方向に向けてデータを転送することは要求されず、片側の接続配線が、常時ダミー配線状態である。従って、このＡＬＵ６に対して配置されるプログラマブルスイッチ回路ＰＳＷｃは、図１０９に示す転送ゲート２８０および２８２が常時非導通状態に設定されても良い。また、これに代えて、ＡＬＵ６に対するスイッチ回路ＰＳＷｃが、図１０９に示す転送ゲート２８１のみで構成されても良い。

【０５９０】

図１１１は、４ビットシフト（ｍｏｖｅ）動作時のプログラマブルスイッチ回路の接続の一例を示す図である。図１１１において、プログラマブルスイッチ回路ＰＳＷａが分断状態ＩＴＰに設定され、プログラマブルスイッチ回路ＰＳＷｂがスルー状態ＴＨＲに設定され、プログラマブルスイッチ回路ＰＳＷｃが分断／ダミー状態ＩＲＤに設定される。

【０５９１】

この図１１１に示す接続状態においては、ＡＬＵ０がＡＬＵ４と結合され、ＡＬＵ４がＡＬＵ６と分離されてスルー状態ＴＨＲのプログラマブルスイッチ回路ＰＳＷｃによりＡＬＵ２に接続される。ＡＬＵ２は、分断／ダミー状態のプログラマブルスイッチ回路ＰＳＷｃによりＡＬＵ０と分離される。

【０５９２】

ＡＬＵ６が、分断／ダミー状態のスイッチ回路ＰＳＷｃを介してＡＬＵ１に結合され、ＡＬＵ７がＡＬＵ３に結合される。ＡＬＵ３は、ＡＬＵ１とプログラマブルスイッチ回路ＰＳＷｃにより分離される。

【０５９３】

したがって、このシフト領域ＡＲ３においてプログラマブルスイッチ回路ＰＳＷを、状態ＩＴＰ、ＴＨＲおよびＩＲＤのいずれかに設定することにより、２ビットシフト（ｍｏｖｅ）動作および４ビットシフト動作を実現することができる。これにより、２ビットシフト用の配線および４ビットシフト用の配線を別々に設ける必要がなく、配線占有面積を低減することができる。

【０５９４】

［変更例２］

シフト領域ＡＲ０－ＡＲ２またはＡＲ０およびＡＲ３の配線のピッチ条件（スイッチ回路ＰＳＷの配置に対する制約条件）を緩和するために、ＡＬＵの配置を、図１１２に示すように設定する。

【０５９５】

図１１２において、４つの単位ＡＬＵ回路ブロック（ＡＬＵ）がエントリ延在方向に整列して配置される。一方側のメモリセルマットのセンスアンプ／ライトドライバ（データレジスタ）ＤＲＧ０ａ－ＤＲＧ７ａと、他方側のメモリセルマットのセンスアンプ／ライトドライバＤＲＧ０ｂ－ＤＲＧ７ｂがそれぞれ各エントリ（図示せず）に対応して配置される。

【０５９６】

この図１１２に示すように、１つのメモリセルマットについて４つのエントリ（センスアンプ／ライトドライバＤＲＧ）の配置ピッチに対し、単位ＡＬＵ回路ブロック（以下、単にＡＬＵと称す）ＡＬＵのピッチを決定する。これにより、ＡＬＵに対するＡＬＵ間シフト用の配線のピッチを緩和することができ、数多くのシフト用配線を余裕を持って各ＡＬＵに対して配置して接続することができる。

【０５９７】

図１１３は、図１１２に示すＡＬＵの配置に対するＡＬＵ間接続用スイッチ回路の配線レイアウトを概略的に示す図である。図１１３においては、４行５列に配置されるＡＬＵを符号“Ｘ［０］－Ｘ［１９］”で示す。通常、ＡＬＵ間データビット転送時においては、Ｘレジスタを用いてデータビットの転送が行なわれるため、このＡＬＵ間接続において各ＡＬＵに含まれるＸレジスタに対する接続が行なわれる状態を一例として示す。

【０５９８】

図のＹ方向（エントリ延在方向）に隣接するＡＬＵの接続は、Ｘ［ｉ］およびＸ［ｉ＋１］を接続する接続配線２９０ａにより実現される。このＹ方向に整列するＡＬＵの最も右に位置するＡＬＵ、すなわちＸ［ｉ＋３］は、次の隣接行の左側に位置するＡＬＵユニット、すなわちＸ［４ｉ＋１］に接続される。これらの配線２９０ａおよび２９０ｂにより、４行５列に整列されるＡＬＵユニットを逐次接続して、１ビットシフト動作を実現することができる。

【０５９９】

Ｘ方向において隣接するＡＬＵが配線２９２により相互接続される（スイッチ回路は示さず）。この配線２９２により、ＡＬＵ（Ｘ［ｉ］）が、４ビット離れたＡＬＵ（［Ｘｉ＋４］）に結合され、４ビットシフト動作を、この配線２９２により実現することができる。

【０６００】

配線２９６は、さらに離れたＡＬＵ間のシフトを行なう配線であり、この配線９２６二より、たとえば３２ビットシフトおよび６４ビットシフトを実現することができる。

【０６０１】

１つの主演算回路において、２５６エントリが設けられている場合、隣接ブロック間データバスＮＢＳ（図１のバス１６）を介して隣接する主演算回路の対応のＡＬＵに接続され、２５６ビットシフトが実現される。

【０６０２】

図１１４は、隣接ブロック間データバスＮＢＳ（図１のバス１６）の配置の一例を概略的に示す図である。メモリセルマット３０においていは、一例として、２５６エントリが配設される。この基本演算ブロック（主演算回路）ＦＢｉのＡＬＵユニット群３２Ｌにおいては、４つのＡＬＵが各行に整列して配置される。したがって、このＡＬＵユニット群３２Ｌにおいて、ＡＬＵ０－ＡＬＵ２５５（Ｘ［０］－Ｘ［２５５］）が配置される。

【０６０３】

基本演算ブロックＦＢ（ｉ＋１）においては、同様、２５６エントリに対応して、４つのＡＬＵが行方向に整列して配置される。基本演算ブロックＦＢｉおよびＦＢｉ＋１は、アドレス領域（外部ＣＰＵのアドレス領域）について連続しており、このＡＬＵユニット群３２Ｒにおける２５６個のＡＬＵに対して連続番号が付され、ＡＬＵ（Ｘ［２５６］－Ｘ［５１１］）が配設される。

【０６０４】

隣接ブロック間データバスＮＢＳにおいて、このＡＬＵユニット郡３２Ｌおよび３２Ｒの同一位置に配設されるＡＬＵが隣接ブロック間データ線２９０により相互接続される。たとえば、ＡＬＵ（Ｘ［３］）は、ＡＬＵユニット群３２ＲのＡＬＵ（Ｘ［２５９］）に接続される。同様、ＡＬＵユニット群３２ＬのＡＬＵ（Ｘ［２５５］）は、ＡＬＵユニット群３２ＲのＡＬＵ（Ｘ［５１１］）に接続される。これにより、データビットを、この隣接ブロック間データバスＮＢＳを介して転送することにより、２５６ビットシフト動作を実現することができる。

【０６０５】

この基本演算ブロックＦＢｉのＡＬＵユニット群３２Ｌの各ＡＬＵは、また隣接ブロック間データバスＮＢＳを介して別の隣接する基本演算ブロックの主演算回路内のＡＬＵに接続される。

【０６０６】

図１１３に示すＡＬＵの配置における８個のＡＬＵを単位として階層的に配置し、かつ各配線の接続に対し、スイッチ回路またはプログラマブルスイッチ回路（図１０５参照）を適用することにより、１ビットシフト動作から２のｎ乗のビットのシフト動作を実現することができる。

【０６０７】

［変更例３］

図１１５は、この発明の実施の形態１９の変更例３の構成を概略的に示す図である。こ

の図１１５に示す構成において、ＡＬＵ間接続用スイッチ回路ＥＣＭの接続を制御するためにＡＬＵシフト制御回路３００が設けられる。このＡＬＵシフト制御回路３００は、主演算回路ＭＰＡ０－ＭＰＡ３に含まれるＡＬＵ間接続用スイッチ回路ＥＣＭ０－ＥＣＭ３それぞれの接続を個々に設定する。これにより、主演算回路ＭＰＡ０－ＭＰＡ３それぞれにおいてＡＬＵ間シフト量を互いに異ならせて演算処理を行なうことができる。

【０６０８】

なお、このＡＬＵシフト制御回路３００におけるシフト制御は、図１に示す基本演算ブロックＦＢ内に含まれるコントローラ２１の制御のもとに実行されてもよい。また、ＡＬＵシフト制御回路３００が、各基本演算ブロック内のコントローラに分散して配置され、対応のＡＬＵ巻相互接続スイッチ回路の接続を制御しても良い。また、これに代えて、ＡＬＵシフト制御回路３００は、システムバス５を介して外部のホストＣＰＵの制御のもとに各ＡＬＵ間相互接続スイッチ回路ＥＣＭ０‐ＥＣＭ３のシフト量を設定してもよい。

【０６０９】

［変更例４］

図１１６は、この発明の実施の形態１９の変更例４の構成を概略的に示す図である。この図１１６に示す構成においては、主演算回路ＭＰＡ０－ＭＰＡ３それぞれにおいて、ＡＬＵ間接続用スイッチ回路ＥＣＭが、上側スイッチ回路ＥＣＭＵおよび下側スイッチ回路ＥＣＭＤに分割される。これらの上側スイッチ回路ＥＣＭＵおよび下側スイッチ回路ＥＣＭＤを、主演算回路ＭＰＡ０－ＭＰＡ３それぞれで個々に制御するため、ＡＬＵシフト制御回路３１０が、主演算回路ＭＰＡ０‐ＭＰＡ３の上側スイッチ回路および下側スイッチ回路に対して個々に制御信号（符号ＵおよびＤ）に与える。符号Ｕで示す制御信号がう上側スイッチ回路ＥＣＭの接続を制御し、符号Ｄで示される制御信号が、下側スイッチ回路ＥＣＭＤの接続を制御する。

【０６１０】

この図１１６に示す他の構成は、図１１５に示す構成と同じであり、対応する部分には同一参照番号を付し、その詳細説明は省略する。

【０６１１】

この図１１６に示す構成の場合、主演算回路ＭＰＡ０－ＭＰＡ３各々において、ＡＬＵ間シフトビット量を上側スイッチ回路ＥＣＭＵおよび下側スイッチ回路ＥＣＭＤそれぞれ個々に設定することができる。したがって、たとえば上側スイッチ回路ＥＣＭＵに対応するＡＬＵにおいて、８ビットシフト動作を行ない、下側スイッチ回路ＥＣＭＤに対応するＡＬＵにおいて２ビットシフト動作を行なうことができ、異なる演算を並列に実行することができる。

【０６１２】

なお、このＡＬＵシフト制御回路３１０は、先の変更例３の構成と同様、主演算回路ＭＰＡ０－ＭＰＡ３に含まれるコントローラ（２１）がそのＡＬＵシフト制御動作を行なうように構成されてもよい。また、ＡＬＵ間シフト動作を制御するために専用に、このＡＬＵシフト制御回路３１０が設けられてもよい。

【０６１３】

また、図１１６に示す構成においては、主演算回路ＭＰＡ０－ＭＰＡ３それぞれにおいて、ＡＬＵ間接続用スイッチ回路が２つのスイッチ回路に分割されている。しかしながら、このＡＬＵ間接続用スイッチ回路の分割数は２に限定されず、さらに多くのブロックに分割されてもよい。各ブロックごとに、ＡＬＵシフト制御回路３１０により、そのシフト量を制御する。

【０６１４】

この図１１６に示す構成の場合、主演算回路それぞれにおいて、個々に複数のＡＬＵ間シフト量を設定することができ、主演算回路それぞれにおいて異なる演算処理を行なうことができ、より複雑な演算処理を実現することができる。

【０６１５】

［変更例５］

図１１７は、このＡＬＵ間接続回路の構成の変更例を示す図である。図１１７において、図１０７に示す配線２７５ａおよび２７５ｂそれぞれとして、＋１ビットシフト線３２０ａｕ、＋４ビットシフト線３２０ｂｕ、＋１６ビットシフト線３２０ｃｕ、およびＮバスシフト線３２０ｄｕ、－１ビットシフト線３２０ａｄ、－４ビットシフト線３２０ｂｄ、－１６ビットシフト線３２０ｃｄ、－Ｎバスシフト線３２０ｄｄが設けられる。Ｎバスシフト線３２０ｄｕおよび３２０ｄｄは、隣接ブロック間データバスＮＢＳの配線である。

【０６１６】

このＡＵＬ間接続回路は、さらに、シフトコマンド信号に従って信号線３２０ａｕ－３２０ｄｕの一方を選択するマルチプレクサ（ＭＵＸ）３１５ｕと、同様、シフトコマンド信号に従って信号線３２０ａｄ－３２０ｄｄのいずれかを選択するマルチプレクサ（ＭＵＸ）３１５ｄと、これらのマルチプレクサ３１５ｕおよび３１５ｄを介して選択された信号線と対応のＡＬＵユニット３１９との間でデータを双方向に送信および受信する送受信データレジスタ３１７を含む。

【０６１７】

送受信データレジスタ３１７へは、＋シフト／－シフトを指定する転送方向指示信号Ｕ／Ｄが与えられる。

【０６１８】

これらの転送方向指示信号を含むシフトコマンド信号は、対応の基本演算ブロックのコントローラからのＡＬＵ制御信号として生成されても良く、また、先の変更例３または４のシフト制御回路３００または３１０から生成されても良い。

【０６１９】

また、送受信データレジスタ３１７は、図１０４および１０６に示す送受信レジスタ２７０に対応する。

【０６２０】

これらのマルチプレクサ３１５ｕおよび３１５ｄが、先の図１０５に示すプログラマブルスイッチ回路ＰＳＷとして利用されてもよい。マルチプレクサ３１５ｕおよび３１５ｄの接続を制御することにより、先の図１０５から図１１１に示すプログラマブルスイッチ回路ＰＳＷのスルー状態、分断状態およびダミー状態を実現することができる。

【０６２１】

図１１８は、図１１７に示す送受信データレジスタ３１７の構成の一例を示す図である。対応のＡＬＵユニット３１９においては、このＡＬＵ間データ転送を行なうレジスタ回路として、Ｘレジスタ３２０を代表的に示す。このＡＬＵユニット３１９において、ＸＨレジスタ、ＸＬレジスタ、またはＺレジスタなどの別のレジスタが、ＡＬＵ間データ転送に用いられてもよい。

【０６２２】

送受信データレジスタ３１７は、Ｘレジスタ３２０の出力ＯＵＴからのデータビットを受ける送信レジスタ３２５と、与えられたデータを取込んで、Ｘレジスタ３２０の入力ＩＮへ伝達する受信レジスタ３２６と、転送方向指示信号Ｕ／Ｄに従って送信レジスタ３２５および受信レジスタ３２６を選択的にプラス（＋）方向シフト配線３２０ｕおよびマイナス（－）方向シフト配線３２０ｄに接続する経路設定回路３３０を含む。

【０６２３】

プラス方向３２０ｕシフト配線は、図１１７に示すシフト配線３２０ａｕ－３２０ｄｕを含み、マイナス方向シフト配線３２０ｄは、図１１７に示すシフト配線３２０ａｄ－３２０ｄｄを含む。この図１１８に示す構成において、図１１７に示すマルチプレクサ３１５ｕおよび３１５ｄは示していない。

【０６２４】

この図１１８に示す構成において、プラス（＋）方向にデータビットを転送する場合、経路設定回路３３０は、送信レジスタ３２５をプラス方向シフト配線３２０ｕに結合し、受信レジスタ３２６をマイナス方向シフト配線３２０ｄに結合する。送信レジスタ３２５

のデータが上方向のＡＬＵに転送されて対応の接続先の受信レジスタに格納される。受信レジスタ３２６が、下側からのＡＬＵの送信レジスタから転送されたデータを格納する。

【０６２５】

マイナス方向にデータ転送を行なう場合には、この経路設定回路３３０は、送信レジスタ３２５を、マイナス方向シフト配線３２０ｄに接続し、受信レジスタ３２６をプラス方向シフト配線３２０ｕに接続する。送信レジスタ３２５が、下側の接続先のＡＬＵの受信レジスタにデータを転送し、受信レジスタ３２６が、上側のソースＡＬＵから送信されたデータを受信する。

【０６２６】

これにより、マイナス方向およびプラス方向いずれの方向においてもデータビットのシフト（転送）が行なわれる場合においても、データの送受信を行なうことができる。この図１１８に示す送信レジスタ３２５および受信レジスタ３２６は、図１０４に示すＡＬＵ間接続回路の構成において、送受信レジスタ２７０として用いられる。

【０６２７】

また、経路設定回路３３０は、スイッチ回路で接続を切り換えるスイッチマトリクスで構成されても良い。また、また、経路設定回路３３０において、送信用トライステートバッファおよび受信用トライステートバッファの組をデータ転送方向に応じて２組設け、トライステートバッファの組を選択的に活性化してデータ転送方向が設定されても良い。

【０６２８】

［変更例６］

図１１９は、この発明の実施の形態１９の変更例６の構成を概略的に示す図である。この図１１９に示す構成においては、ＡＬＵ間の接続配線が、メモリセルマット３０をＹ方向（エントリ延在方向）に延在する配線３４０と、Ｘ方向にメモリセルマット３０を延在してそれぞれ所定ビット離れた配線３４０を相互接続する配線３４２とにより形成される。

【０６２９】

ＡＬＵユニット群３２において、たとえば先の図１１２に示すように４つのＡＬＵが１列に整列して配置される場合（ＡＬＵのピッチがビット線ピッチの４倍の場合）、４ビット離れた配線３４０を相互接続することにより、６４ビット転送の経路を形成することができる。ＡＬＵユニット群３２内に長距離にわたる配線を配置する構成に代えて、メモリセルマット３０上に長距離間データ転送配線を配置することにより、配線レイアウトが容易となる。

【０６３０】

なお、このメモリセルマット３０上を延在して配置される配線３４０および３４２は、たとえば１２８ビット離れたＡＬＵを相互接続する配線であってもよい。

【０６３１】

また、このＡＬＵ間相互接続用スイッチ回路の構成において、ビットシフト量は、2^n（ｎ＝０…８）の構成が一例として示されている。しかしながら、実行される演算内容に応じて、シフトされるビット量およびそのシフトに必要とされるクロックサイクル数に応じてまたメモリセルマット３０に含まれるエントリの数に応じて、適当なシフト量を実現するシフト配線接続が用いられればよい。

【０６３２】

以上のように、この発明の実施の形態１９に従えば、主演算回路におけるＡＬＵ間の接続経路を各主演算回路個々に設定し、また接続経路をプログラマブルに設定することにより、ＡＬＵ間相互接続用配線面積を増大させることなく、効率的に、ＡＬＵ間接続を実現することができ、並列演算性能を、配線レイアウト面積と増大させることなく向上させることができる。

【０６３３】

［実施の形態２０］

図１２０は、この発明の実施の形態２０に従う入出力回路１０の構成を概略的に示す図

である。この入出力回路１０は、図１に示すように、システムバス５を介してホストＣＰＵに結合される。内部データバス１６（グローバルデータバスＧＢＳ）は、内部の複数の基本演算ブロックに共通に結合される。

【０６３４】

入出力回路１０は、ｊビットのホストシステムバス５（バス線ＨＢＳ［（ｊ－１）：０］）と内部のｋビットの第１の内部転送バスＣＢＳ［（ｋ－１）：０］との間でデータ転送を行ないかつデータビット配列の直交変換を行なう直交変換回路４００と、ｋビットの第１の内部転送バスＣＢＳ［（ｋ－１）：０］とｍビットの第２の内部転送バスＳＢＳ［（ｍ－１）：０］の間で選択的にデータ転送経路を設定してデータビットの転送を行なうクロスバースイッチ４０２と、第２の内部転送バスＳＢＳ［（ｍ－１）：０］とｎビットの内部データバス１６（グローバルデータバスＧＢＳ［（ｎ－１）：０］の間でデータの転送を行なうセレクタ４０４を含む。

【０６３５】

ホストシステムバス５は、シングルエンドのバス線で構成され、第１および第２の転送バスＣＢＳおよびＳＢＳ並びにグローバルデータバスＧＢＳ（内部データバス１６）は、各々、ダブルエンドのバス線で構成され、相補信号を伝達する。以下の説明において、これらのバスについて各ビットを特に参照しない場合には、符号ＣＢＳ、ＳＢＳ、ＧＢＳおよびＨＢＳを用いて、バスを参照する。

【０６３６】

入出力回路１０において、ｊビットのワードシリアルかつビットパラレル態様で転送されるデータとｎビットのグローバルデータバスＧＢＳ上に伝達されるワードパラレルかつビットシリアルのデータの間のデータ配列の変換および転送データビット幅の変換が行なわれる。

【０６３７】

図１２１は、図１２０に示す直交変換回路４００の第１の転送バスＣＢＳからホストシステムバスＨＢＳ（バス５）にデータを転送する出力部分の構成を概略的に示す図である。図１２１において、直交変換回路４００のデータ出力部は、ｋ行ｊ列に配列される変換素子ＴＸＦ００－ＴＸＦ（ｋ－１）（ｊ－１）を含む。変換素子ＴＸＦ００－ＴＸＦ（ｋ－１）（ｊ－１）は同一構成を有するため、図１２１においては、変換素子ＴＸＦ００の構成を代表的に示す。変換素子ＴＸＦ００は、クロック入力に与えられる制御信号ＳＤｏｉ［０］に従って相補内部データバス線ＣＢＳ［０］および／ＣＢＳ［０］上の信号を取込みラッチするフリップフロップＦＦａと、出力制御信号ＳＴｏｏ［０］に従ってフリップフロップＦＦａの出力信号をシングルエンドのシステムバス線ＨＢＳ［０］へ伝達するトライステートバッファＢＦを含む。

【０６３８】

変換素子ＴＸＦ（変換素子を総称的に示す）の各行に対応して第１の内部転送バス線ＣＢＳ［ｕ］，／ＣＢＳ［ｕ］が配設され、変換素子各列に対応してホストシステムバス線ＨＢＳ［ｖ］が配設される。ここで、ｕ＝０～（ｋ－１）であり、ｖ＝０～（ｊ－１）である。

【０６３９】

行方向に整列する変換素子ＴＸＦに対して共通に出力制御信号ＳＴｏｏ［ｕ］がそれぞれの出力バッファＢＦに与えられる。列方向に整列する変換素子のフリップフロップＦＦａのクロック入力に対し共通に、入力制御信号ＳＤｏｉ［ｖ］が与えられる。

【０６４０】

これらの制御信号ＳＤｏ［（ｊ－１）：０］およびＳＴｏｏ［（ｋ－１）：０］は、ホストＣＰＵからの制御の下に、図１に示す制御用ＣＰＵ（２５）が生成してもよく、また各基本演算ブロック内に設けられるコントローラからのＤＭＡ転送要求に従って、ＣＰＵ２５が同様に生成してもよい。また、ホストＣＰＵが直接生成しても良く、ＤＭＡモード転送のときには外部のＤＭＡコントローラの制御の下に生成されても良い。

【０６４１】

図１２２は、この図１２１に示す直交変換回路４００の出力部の動作を模式的に示す図である。第１の内部転送バスＣＢＳから、ビットシリアルかつワードパラレルの態様でデータが転送される。今、データビットＡ［ａ］、Ｂ［ａ］・・・Ｇ［ａ］が並列に与えられる。この場合、入力制御信号ＳＤｏｉ（ａ）に従って、第ａ列の変換素子ＴＸＦが、与えられたデータビットを取込みラッチする。したがって、この直交変換回路４００において、第１の内部転送バスＣＢＳからのデータによりフル状態となった場合には、列方向において、ワードＡ、Ｂ、…Ｇの同一ビット位置のデータが配列され、行方向には、各ワードのビットが整列して配置される。

【０６４２】

データ出力時においては、出力制御信号ＳＤｏｏ［ｂ］に従って行方向に整列する変換素子ＴＸＦが同時に出力状態に設定される。したがって、ホストシステムバスＨＢＳには、１つのデータワードＤの各ビット［０］－［ｊ－１］が並列に出力される。これにより、第１の内部転送バスＣＢＳからエントリの同一位置のビットが並列に転送されてきた場合、ホストシステムバスＨＢＳに、各データワードがシリアルに出力される。

【０６４３】

図１２３は、変換素子ＴＸＦの構成の一例を示す図である。図１２３において、フリップフロップＦＦａは、入力制御信号ＳＤｏｉ［ｕ］に従って転送バス線ＣＢＳ［ｓ］および／ＣＢＳ［ｓ］上の信号を伝達するトランスファーゲート４２１ａおよび４２１ｂと、活性化時、トランスファーゲート４２１ａおよび４２１ｂから転送されたデータを差動的に増幅する交差結合型差動増幅回路４２２と、入力制御信号ＳＤｏｉ［ｕ］に従って交差結合型差動増幅回路４２２のＰチャネルＭＯＳトランジスタへ電源電圧を供給するＰチャネルＭＯＳトランジスタ４２３ｐと、この入力制御信号ＳＤｏｉ［ｕ］の反転信号ＺＳＤｏｉ［ｕ］に従って交差結合型差動増幅回路４２２のＮチャネルＭＯＳトランジスタを接地ノードに結合するＮチャネルＭＯＳトランジスタ４２３ｎを含む。

【０６４４】

このフリップフロップＦＦａは、入力制御信号ＳＤｏｉ［ｕ］がＨレベルのときに、対応の転送バス線ＣＢＳ［ｓ］および／ＣＢＳ［ｓ］上の信号を、交差結合型差動増幅器４２２に伝達する。入力制御信号ＳＤｏｉ［ｕ］が非活性状態となると、交差結合型差動増幅回路４２２が、ＭＯＳトランジスタ４２３ｐおよび４２３ｎにより活性化され、トランスファーゲート４２１ａおよび４２１ｂにより転送されたデータを差動的に増幅してラッチする。

【０６４５】

トライステートバッファＢＦは、電源ノードに結合されかつフリップフロップＦＦａの出力Ｑをインバータを介してゲートに受けるＰチャネルＭＯＳトランジスタＰＸ１と、接地ノードに結合されかつフリップフロップＦＦａの出力／Ｑをゲートに受けるＮチャネルＭＯＳトランジスタＮＸ１と、出力制御信号ＳＤｏｏ［ｖ］に従ってＭＯＳトランジスタＮＸ１を対応のホストシステムバス線ＨＢＳ［ｔ］に結合するＮチャネルＭＯＳトランジスタＮＸ２と、出力制御信号ＳＤｏｏ［ｖ］のインバータを通した信号に応答してＭＯＳトランジスタＰＸ１を対応のホストシステムバス線ＨＢＳ［ｔ］に結合するＰチャネルＭＯＳトランジスタＰＸ２を含む。

【０６４６】

この図１２３に示すトライステートバッファＢＦは、出力制御信号ＳＤｏｏ［ｖ］がＬレベルのときには、ＭＯＳトランジスタＰＸ２およびＮＸ２がともにオフ状態であり、出力ハイインピーダンス状態である。

【０６４７】

出力制御信号ＳＤｏｏ［ｖ］がＨレベルとなると、ＭＯＳトランジスタＰＸ２およびＮＸ２がオン状態となり、ＭＯＳトランジスタＰＸ１およびＮＸ１が、対応のホストシステムバス線ＨＢＳ［ｔ］に結合され、フリップフロップＦＦの出力Ｑ、／Ｑの信号に従ってホストシステムバス線ＨＢＳ［ｔ］を駆動する。

【０６４８】

この図１２３に示す変換回路ＴＸＦの構成は単なる一例であり、別の構成が用いられてもよく、入力制御信号ＳＤｏｉ［ｕ］に従って相補信号を取込みラッチし、出力制御信号ＳＤｏｏ［ｖ］の活性化時、この取込んだ相補信号に従ってシングルエンドのシステムバス線を駆動する構成であれば任意の回路構成を利用することができる。

【０６４９】

図１２４は、図１２０に示す直交変換回路４００のデータ入力部の構成を概略的に示す図である。図１２４においては、ホストシステムバスＨＢＳを介して伝達される１ワードのデータに対する入力部の構成を示す。ホストシステムバスＨＢＳのビット幅（ｊビット）に応じて、この図１２４に示す入力部の構成が拡張され、また第１の内部転送バスＣＢＳのビット幅（ｋビット）に従って、この図１２４に示す構成が列方向に繰返し配置される。

【０６５０】

図１２４において、直交変換回路４００の入力部は、ホストシステムバスＨＢＳのバス線ＨＢＳ［０］－ＨＢＳ［７］それぞれに対して設けられる入力変換素子ＴＸＦＩ０－ＴＸＦＩ７と、ワード単位でデータのマスクを指示するマスク信号線ＨＢＳｍ［０］に従って、これらの入力変換素子ＴＸＦＩ０－ＴＸＦＩ７の出力にマスクをかけるワードマスク制御回路４３０を含む。

【０６５１】

入力変換素子ＴＸＦＩ０－ＴＸＦＩ７は、各々、入力制御信号ＳＤｉｉ［ｘ］に従って対応のホストシステムバス線ＨＢＳ［０］－ＨＢＳ［７］上の信号を取込むフリップフロップＦＦｂと、活性化時、対応のフリップフロップＦＦｂのラッチ信号に従って転送バス線ＣＢＳ［ｘ］および／ＣＢＳ［ｘ］に相補信号を伝達するトライステートバッファ４３２と、ワードマスク制御回路４３０からのマスク制御信号と対応の出力制御信号ＳＤｉｏ［ａ］とを受けて対応のトライステートバッファ４３２を活性化するＡＮＤ回路４３１を含む。ここで、出力制御信号ＳＤｉｏ［ａ］において、ａ＝０－７である。

【０６５２】

ワードマスク制御回路４３０は、リセット信号ＳＤｉｒ［ｘ］に応答してリセットされ、対応の１ワード（８ビット）の入力変換素子ＴＸＦＩ０－ＴＸＦＩ７の出力にマスクをかけるフリップフロップＦＦｃと、フリップフロップＦＦｃの出力信号とマスク信号線ＨＢＳｍ［０］上の信号とを受けるＮＯＲゲート４３３を含む。フリップフロップＦＦｃは、ＮＯＲゲート４３３の出力信号を入力制御信号ＳＤｉｉ［ｘ］に従って取込みラッチする。次に、この図１２４に示す直交変換回路４００の入力部の動作について説明する。

【０６５３】

ホストシステムバスＨＢＳからデータが転送されるとき、まず、リセット信号ＳＤｉｒ［ｘ］が活性化され、ワードマスク制御回路４３０のフリップフロップＦＦｃがリセットされる。これにより、入力変換素子ＴＸＦＩ０－ＴＸＦＩ７それぞれにおいて、ＡＮＤゲート４３１がディスエーブルされ、入力変換素子が、出力ハイインピーダンス状態に設定される。

【０６５４】

入力制御信号ＳＤｉｉ［ｘ］とホストシステムバスのマスク信号線ＨＢＳｍ［０］とに従って、ホストシステムバス線ＨＢＳ［０］－ＨＢＳ［７］上のデータビットが、入力変換素子ＴＸＦＩ０－ＴＸＦＩ７のそれぞれのフリップフロップＦＦｂ内に選択的に取込まれる。マスク信号線ＨＢＳｍ［０］上の信号がＨレベルのときには、ワードマスク制御回路４３０のＮＯＲゲート４３３の出力信号が、Ｈレベルとなり、フリップフロップＦＦｃが、入力制御信号ＳＤｉｉ［ｘ］に従ってこのＨレベルの信号を取込みラッチする。フリップフロップＦＦｃのＨレベルの出力信号に従って、入力変換素子ＴＸＦＩ０－ＴＸＦＩ７それぞれのＡＮＤゲート４３１がイネーブルされる。このとき、また、入力制御信号ＳＤｉｉ［ｘ］に従って入力変換素子ＴＸＦＩ０－ＴＸＦＩ７それぞれにおいてフリップフロップＦＦｂが、ホストシステムバス線ＨＢＳ［０］－ＨＢＳ［７］上の信号を取込みラッチする。

【０６５５】
マスク信号ＨＢＳｍ［０］がＬレベルのときには、ワードマスク制御回路４３０のフリップフロップＦＦｃの出力信号はＬレベルであり、入力変換素子ＴＸＦＩ０－ＴＸＦＩ７それぞれにおいて、ＡＮＤゲート４３１はディスエーブル状態に維持される。このときにも、対応のホストシステムバス線上のデータ信号の取込はフリップフロップＦＦｂにおいて実行される。ラッチデータビットの出力時においては、出力制御信号ＳＤｉｏ［０］－ＳＤｉｏ［７］が順次活性化される。ワードマスク制御回路４３０の出力信号がＬレベルのときには、これらのＡＮＤゲート４３１の出力信号がＬレベルであるため、対応の出力制御信号ＳＤｉｏ［０］－ＳＤｉｏ［７］がＨレベルに活性化されても、トライステートバッファ４３２は出力ハイインピーダンス状態である。したがって、この場合には、入力変換素子ＴＸＦＩ０－ＴＸＦＩ７からデータビットの転送は行なわれず、ＣＰＵからのワードに対してマスクがかけられる。

【０６５６】
ワードマスク制御回路４３０の出力信号がＨレベルのときには、ＡＮＤゲート４３１が、出力制御信号ＳＤｉｏ［０］－ＳＤｉｏ［７］に従ってそれぞれ活性化され、トライステートバッファ４３２が対応のフリップフロップＦＦｂのラッチ信号に従って相補転送バス線／ＣＢＳ［ｘ］およびＣＢＳ［ｘ］を駆動する。

【０６５７】
図１２５は、この図１２４に示す直交変換回路４００の入力部のデータ転送動作を模式的に示す図である。直交変換回路４００においては、データ入力時、入力制御信号ＳＤｉｉ［ｘ］に従って、ホストシステムバスＨＢＳ上のデータが取込まれてラッチされる。このホストシステムバスＨＢＳ上のデータはワード単位でマスクデータｍ［０］－ｍ［（ｊ－８）／８］に従って選択的にマスクがかけられる。最終的に入力制御信号ＳＤｉｉ［ｊ－１］が活性化されると、この直交変換回路４００の入力部において格納領域が一杯となる。

【０６５８】
データ出力時においては、出力制御信号ＳＤｉｏ［ｖ］に従って、図の斜線で示す縦方向に１列に整列するデータが第１の内部転送バスＣＢＳ上に並列に転送される。今、マスク信号ＨＳＢＭに従ってマスクが指定されたマスクデータＭＳＤＴが存在する場合、対応の転送バス線ＣＢＳ［ｋ－１］上には、データは転送されず、このマスクデータＭＳＤＴにマスクがかけられる。

【０６５９】
この入力部の構成を用いることにより、ホストＣＰＵからのデータをワード単位でマスクをかけて、内部の基本演算ブロックの主演算回路のメモリセルマットへ格納することができる。

【０６６０】
なお、制御信号ＳＤｉｉ［ｘ］、リセット信号ＳＤｉｒ［ｘ］および出力制御信号ＳＤｉｏ［ｖ］は、外部のホストＣＰＵまたは図１に示す集中制御ユニット１５から、データ入力時に生成されてもよく、また、専用のカウンタ回路（シフトレジスタ）により、これらの制御信号が順次活性化されてもよい。

【０６６１】
この図１２１に示す出力部および図１２４に示す入力部を２セット設け、これらの入力／出力部を、インタリーブ態様で動作させることにより、外部のホストＣＰＵおよび内部データバスの間のデータ転送速度の差を吸収して、連続的に（ギャップレスで）データ転送を行なうことができる。

【０６６２】
なお、この直交変換回路４００においては、変換素子を構成するためにフリップフロップとトライステートバッファ回路が用いられている。しかしながら、先の実施の形態において示したように、デュアルポートメモリを用いて、一方のポートをホストシステムバスに結合し、他方のポートを第１の内部転送バスに結合する構成を利用してもよい。このよ

うなデュアルポートメモリを利用する場合、面積利用効率を改善することができ、チップ面積を低減することができる。

【０６６３】

図１２６は、図１２０に示されるクロスバースイッチの構成を概略的に示す図である。図１２６においては、第２の内部転送バスＳＢＳ［（ｍ－１）：０］のうち１ビットの転送バスＳＢＳ［ｙ］および／ＳＢＳ［ｙ］に対するクロスバースイッチの構成を示す。この図１２６に示す構成が、第２の内部転送バスＳＢＳの各バス線に対して設けられる。

【０６６４】

図１２６において、クロスバースイッチ４０２は、第１の内部転送バス線ＣＢＳ［０］，ＣＢＳ［０］－ＣＢＳ［ｍ－１］，／ＣＢＳ［ｍ－１］のそれぞれに対して設けられるデコーダＤＤＤ０－ＤＤＤ（ｍ－１）と、デコーダＤＤＤ０－ＤＤＤ（ｍ－１）の出力信号に従って、第１の内部転送バス線ＣＢＳ［０］，／ＣＢＳ［０］－ＣＢＳ［ｍ－１］，／ＣＢＳ［ｍ－１］を第２の内部転送バス線ＳＢＳ［ｙ］／ＳＢＳ［ｙ］に接続する選択スイッチ回路ＤＳＷ０－ＤＳＷ（ｍ－１）を含む。

【０６６５】

デコーダＤＤＤ０－ＤＤＤ（ｍ－１）は、接続制御信号ＤＥＣ［０］－ＤＥＣ［４］をデコードするデコード回路４４０と、デコード回路４４０の出力するプリデコード信号に従ってスイッチ制御信号を対応のスイッチ回路ＤＳＷ０－ＤＳＷ（ｍ－１）に出力するＡＮＤ回路４４１を含む。

【０６６６】

接続制御信号ＤＥＣ［０］－ＤＥＣ［４］は、５ビットの接続制御信号であり、第１の内部転送バスＣＢＳが３２ビットの場合を想定する。この接続制御信号ＤＥＣのビット幅は、第１の内部転送バスＣＢＳのバス幅に応じて定められる。デコーダＤＤＤ０－ＤＤＤ（ｍ－１）の１つの出力信号が選択状態となり、対応の選択スイッチ回路ＤＳＷ（ＤＳＷ０－ＤＳＷ（ｍ－１）のいずれか）が導通状態となり、選択された第１の内部転送バス線ＣＢＳ［ｚ］，／ＣＢＳ［ｚ］が、第２の内部転送バス線ＳＢＳ［ｙ］，／ＳＢＳ［ｙ］に接続される。

【０６６７】

第２の内部転送バス線ＳＢＳのバス線それぞれにおいて、このデコード動作に基づいて接続を設定することにより、第１の内部転送バスＣＢＳと第２の内部転送バスＳＢＳとのバス幅を整合させて選択的な接続を形成することができる。

【０６６８】

図１２７は、図１２６に示す接続制御信号ＤＥＣ［０］－ＤＥＣ［４］を発生する部分の構成を概略的に示す図である。図１２７において、接続制御信号発生回路は、行列状に配列されるレジスタ回路ＸＧ００－ＸＧ３４と、第２の内部転送バス線ＳＢＳ［ｙ］および／ＳＢＳ［ｙ］上の信号を増幅してシングルエンドの信号を生成するセンスアンプ回路ＳＡＣと、レジスタ回路ＸＧ００－ＸＧ３４のＹ方向に整列するレジスタ回路それぞれに対応して設けられ、それぞれ選択信号ＳＣｂ［０］－ＳＣｂ［４］に従って選択的に導通し、導通時、センスアンプ回路ＳＡＣの出力信号を伝達する選択ゲートＳＳＧ０－ＳＳＧ４と、レジスタ回路ＸＧ００－ＸＧ３４の各列に対応して設けられ、対応のレジスタ回路の出力信号を増幅して接続制御信号ＤＥＣ［０］－ＤＥＣ［４］をそれぞれ生成するドライバＤＲＶ０－ＤＲＶ４を含む。

【０６６９】

レジスタ回路ＸＧ００－ＸＧ３４においては、Ｘ方向に整列するレジスタ回路に共通に入力制御信号ＳＣｉ［０］－ＳＣｉ［３］および出力制御信号ＳＣｃ［０］－ＳＣｃ［３］がそれぞれ与えられる。

【０６７０】

レジスタ回路ＸＧ００－ＸＧ３４の各々は、対応の入力制御信号ＳＣｉ［ｚ］に従って対応の選択ゲートＳＳＧ（ＳＳＧ０－ＳＳＧ４のいずれか）からのデータを転送するトランスファーゲート４５２と、トランスファーゲート４５２を介して与えられたデータをラ

ッチするラッチ回路４５３と、出力制御信号ＳＣｃ［ｚ］に従って対応のラッチ回路４５３にラッチされたデータを対応のドライバＤＲＶ（ＤＲＶ０－ＤＲＶ４）へ転送する転送ゲート４５４を含む。ラッチ回路４５３は、インバータで構成されるラッチを備え、与えられた信号をラッチする。

【０６７１】

センスアンプ回路ＳＡＣは、センスアンプ活性化信号ＳＣｓに従って活性化され、基本演算ブロックから第２の内部転送バス線ＳＢＳ［ｙ］および／ＳＢＳ［ｙ］上に転送されたデータビットを増幅する。

【０６７２】

この図１２７に示す接続制御信号発生部の構成においては、センスアンプ回路ＳＡＣが生成した１ビットコンテキスト情報が、選択ゲートＳＳＧ０－ＳＳＧ４と入力制御信号ＳＣｉ［０］－ＳＣｉ［３］より選択されたレジスタ回路へ転送されてラッチされる。Ｘ方向に整列するレジスタ回路ＸＧａ０－ＸＧａ４に、クロスバースイッチの接続態様を決定する１つのコンテキスト情報が格納される。したがって、基本演算ブロックから内部転送バス線ＳＢＳ［ｙ］，／ＳＢＳ［ｙ］へ５ビットシリアルに情報を転送し、このデータ転送と同期して選択信号ＳＣｂ［０］－ＳＣｂ［４］を活性状態とすることにより、選択ゲートＳＳＧ０－ＳＳＧ４を介してコンテキスト情報ビットが転送される。このときに、１つの入力制御信号ＳＣｉを選択状態に維持することにより、Ｘ方向に整列するレジスタ回路に転送されたコンテキスト情報ビットが順次ラッチされる。これにより、Ｘ方向に整列するレジスタ回路により、１つのクロスバースイッチの接続態様を決定するコンテキスト情報が格納される。

【０６７３】

したがって、入力制御信号ＳＣｉ［０］－ＳＣｉ［３］それぞれがコンテキスト情報を選択することができるため、４ウェイのコンテキスト情報（４面のコンテキスト情報）を格納することができる。必要な接続態様を決定するコンテキスト情報は、出力制御信号ＳＣｃ［０］－ＳＣｃ［３］のいずれかを活性状態にすることにより読出されて、ドライバＤＲＶ０－ＤＲＶ４を介して伝達される。これにより、図１２６に示すデコーダＤＤＤ０－ＤＤＤ（ａ－１）により、３２対の内部転送バス線ＣＢＳ［０］，／ＣＢＳ［０］－ＣＢＳ［ｍ－１］，／ＣＢＳ［ｍ－１］の１つが選択される。

【０６７４】

このクロスバースイッチの接続態様を決定するために４種類の情報を格納することにより、クロスバースイッチの接続態様をリアルタイムで切換えることができ、データ配列順序を、転送時に容易に変換して演算を行なうことができる。例えば、第１の内部転送バスＣＢＳが８ビットであり、第２の内部転送バスＳＢＳが３２ビットの場合、この４面のコンテキスト情報により、クロスバースイッチ４０２において、８ビットデータの転送を行う経路を順次切り換えることにより、バス幅の調整を行ってデータの転送を行うことができる。

【０６７５】

図１２８は、クロスバースイッチ４０２の全体構成を概略的に示す図である。図１２８において、クロスバースイッチ４０２は、第２の内部転送バス線ＳＢＳ［０］－ＳＢＳ［ｋ－１］それぞれに対応して配置されるスイッチ列４６０ａを含むスイッチマトリクス４６４と、スイッチ列４６４ａそれぞれに対応して配置されるデコーダ群４６２ａを含み、スイッチマトリクス４６４の接続経路を設定するルート決定回路４６２と、デコーダ群４６２ａそれぞれに対応して配置されるコンテキスト情報格納部４６０ａを含み、ルート決定回路４６２の接続ルートを決定する情報を格納するルート情報格納回路４６０を含む。

【０６７６】

スイッチ列４６４ａは、図１２６に示す選択スイッチ回路ＤＳＷ０－ＤＳＷ（ｍ－１）を含み、第１の内部転送バス線ＣＢＳ［０］－ＣＢＳ［ｍ－１］のいずれかを対応の第２の内部転送バス線ＳＢＳ［０］－ＳＢＳ［ｋ－１］に接続する。

【０６７７】

デコーダ群４６２ａは、図１２６に示すデコーダＤＤＤ０－ＤＤＤ（ｊ－１）を含み、対応のスイッチ列４６４ａの選択スイッチ回路の導通／非導通を設定する。

【０６７８】

コンテキスト情報格納回路４６０ａは、図１２７に示す構成を備え、対応のデコーダ群４６２ａに対し、４種類のコンテキスト情報を格納し、出力制御信号ＳＣｃに従って、格納したコンテキスト情報のいずれかを対応のデコーダ群４６２ａに出力する。

【０６７９】

このルート情報格納回路４６０へは、センスアンプ回路群４６６からのｋビットの経路指定情報が与えられる。センスアンプ回路群４６６は、図１２７に示すセンスアンプ回路ＳＡＣを第２の内部転送バス線ＳＢＳ［０］－ＳＢＳ［ｋ－１］それぞれに対応して含み、活性化時、基本演算ブロックから与えられたｋビットのデータを増幅して、それぞれ対応のコンテキスト情報格納回路４６０ａに転送する。

【０６８０】

内第２の内部転送バス線ＳＢＳ［０］－ＳＢＳ［ｋ－１］へは、基本演算ブロックＦＢｉの経路情報格納メモリ４６０からの情報が、内部のコントローラ２１の制御の下に読出されて転送される。このコントローラ２１は、集中制御ユニット１５に含まれる制御用ＣＰＵ２５の制御の下に動作し、メモリ４６０に格納された接続情報を順次出力する。この制御用ＣＰＵ２５は、また経路情報格納回路４６０に対する制御信号ＳＣｂ、ＳＣｃ、およびＳＣｉをそれぞれ出力する。選択制御信号ＳＣｂが５回トグルされることにより、信号ＳＣｂ［４：０］が１回全て選択され、１つのコンテキスト情報の格納が完了する。

【０６８１】

経路情報格納メモリ４６０については、主演算回路内のメモリセルマットの特定の領域が用いられてもよく、またメモリセルマットと別に専用に設けられても良い。

【０６８２】

制御用ＣＰＵ２５はまた、センスアンプ回路群４６６に含まれるセンスアンプ回路（ＳＡＣ）に対するセンス活性化信号（ＳＣｃ）を生成する（この経路は示さず）。

【０６８３】

図１２９は、図１２８に示すデコーダ群４６２ａのデコーダ／スイッチ回路（単位接続回路と以下称す）とデコード信号の対応関係を概略的に示す図である。図１２９において、単位接続回路ＵＣＢＳＷ０は、接続制御信号ＤＥＣが０（十進）のときに、第１の転送バス線ＣＢＳ［ｉ］を第２の転送バス線ＳＢＳ［ｉ］に接続する。単位接続回路ＵＣＢＳＷｘは、接続制御信号ＤＥＣが（０＋ｘ）（十進）のとき、第１の内部転送バス線ＣＢＳ［（ｉ＋ｘ）ｍｏｄ．ｍ］を第２の内部転送バス線ＳＢＳ［ｉ］に接続する。

【０６８４】

第１の内部転送バスＣＢＳはｍビット幅であり、接続制御信号ＤＥＣが“０”のときには、常に、第１の内部転送バス線ＣＢＳ［ｉ］が第２の内部転送バス線ＳＢＳ［ｉ］に接続される。従って、この接続経路がサイクリックに切換える場合においても、接続制御信号ＤＥＣのモジューロｍの演算値により、この第１の内部転送バスＣＢＳの接続を容易に決定することができ、各デコーダ群４６２ａにおいて、デコード信号ＤＥＣが“０”のときには、それぞれ同一番号の転送バス線ＣＢＳ［ｊ］およびＳＢＳ［ｊ］の接続を行なうことができ、接続情報のためのプログラムが容易となる。

【０６８５】

図１３０は、図１２０に示すセレクタ４０４の構成を概略的に示す図である。図１３０においては、１つの第２の内部転送バス線ＳＢＳ［ｚ］，／ＳＢＳ［ｚ］に対するセレクタの構成を示す。この図１３０に示す構成においては、グローバルデータバスＧＢＳは、第２の内部転送バスＳＢＳの４倍のビット幅を有する（ｎ＝４・ｍ）。

【０６８６】

セレクタ４０４は、選択信号ＳＳ［０］に従ってグローバルデータバス線ＧＢＳ［４ｚ］，／ＧＢＳ［４ｚ］を転送バス線ＳＢＳ［ｚ］，／ＳＢＳ［ｚ］に接続する接続ゲートＴＧＷ０と、選択信号ＳＳ［１］に従ってグローバルデータバス線ＧＢＳ［４ｚ＋１］，

／ＧＢＳ［４ｚ＋１］を転送バス線ＳＢＳ［ｚ］，／ＳＢＳ［ｚ］に接続する接続ゲートＴＧＷ１と、選択信号ＳＳ［２］に従ってグローバルデータバス線ＧＢＳ［４ｚ＋２］，／ＧＢＳ［４ｚ＋２］を転送バス線ＳＢＳ［ｚ］，／ＳＢＳ［ｚ］に接続する接続ゲートＴＧＷ２と、グローバルデータバス線ＧＢＳ［４ｚ＋３］，／ＧＢＳ［４ｚ＋３］を選択信号ＳＳ［３］に従って転送バス線ＳＢＳ［ｚ］，／ＳＢＳ［ｚ］に接続する接続ゲートＴＧＷ３を含む。

【０６８７】

これらのグローバルデータバス線ＧＢＳ［４ｚ］，／ＧＢＳ［４ｚ］－ＧＢＳ［４ｚ＋３］，／ＧＢＳ［４ｚ＋３］は、互いに隣接するデータバス線であることは特に要求されない。ｍビット離れたバス線であってもよい。

【０６８８】

選択信号ＳＳ［０］－ＳＳ［３］は、図１に示す集中制御ユニット１５に含まれる制御用ＣＰＵ２５から生成され、データ転送時、順次活性化される。

【０６８９】

図１３１は、この図１３０に示すセレクタ４０４の選択動作を模式的に示す図である。第２の内部転送バスＳＢＳ上では、ｍビットのデータＤ０－Ｄ３が順次転送される。セレクタ４０４は、１／４選択を行っており、本実施の形態においては、選択信号ＳＳ［３：０］に従ってグローバルデータバスＧＢＳのｍビットのバス線を順次選択する。これにより、セレクタ４０４からのデータＤ０、Ｄ１、Ｄ２およびＤ３が、グローバルデータバスＧＢＳのそれぞれｍビットのデータバス線に分配される。

【０６９０】

このグローバルデータバスＧＢＳにおいて、この図１３１に示す選択方式の場合、データＤ０－Ｄ３は、各々、互いに異なるデータワードのビットで構成され、データＤ０－Ｄ３は、各々異なるエントリに格納される。

【０６９１】

データＤ０－Ｄ３が、それぞれデータワードＡＤ－ＤＤの組の異なるビットで構成されるように、同一ワードの組の異なるデータビットで構成され、これらのデータＤ０－Ｄ３を、順次主演算回路内のメモリセルマットの共通のエントリに書込む必要がある場合には、セレクタ４０４の接続経路が固定され、たとえばグローバルデータバスＧＢＳの特定のｍビットのデータ線に連続的にデータＤ０－Ｄ３が出力される。この状態を図１３２に示す。

【０６９２】

図１３２において、グローバルデータバス線ＧＢＳ［４ｚ］にデータＤ０－Ｄ３が順次転送される場合を示す。ただし、４ｚは０から（ｍ－１）であり、隣接データバス線で構成される。これにより、同一ワードのデータビットを順次同一のグローバルデータバス線を介して転送することができ、応じて、主演算回路内のメモリセルマットの共通のエントリに順次格納することができる。セレクタ４０４は双方向スイッチ回路であり、グローバルデータバスＧＢＳから第２の内部転送バスＳＢＳのデータ転送時にも、この図１３１または図１３２に示すデータの転送シーケンスに従ってデータ転送が行なわれる。

【０６９３】

なお、この図１３２に示すデータの分配においても、グローバルデータバスＧＢＳにおいてデータＤ０－Ｄ３は、グローバルデータバスＧＢＳのｍビットの隣接バス線ＧＢＳ［（ｊ－１）：０］に配置されている。しかしながら、隣接バス線ではなく、たとえばＧＢＳ［４ｚ｜ｚ＝０～（ｍ－１）］のように、すなわち、ＧＢＳ［０］、ＧＢＳ［４］・・・のように互いにｍビット離れたバス線に分散して配置されても良い。

【０６９４】

このデータバスの接続制御においては、主演算回路のメモリセルマットのエントリにワードの各ビットが格納されるという条件が満たされ、入出力回路でデータ配列の変換が行われるという条件が満たされる限り、このデータの分配経路の設定は任意に定めることができる。

【０６９５】
以上のように、この発明の実施の形態２０に従えば、外部ＣＰＵに接続されるシステムバスと内部の基本演算ブロックが接続されるグローバルデータバスの間に、データの並べ替えおよびビット幅を調整する入出力回路を設けており、確実に、ホストＣＰＵの処理するデータワードのビット幅にかかわらず、各基本演算ブロックの主演算回路内のエントリに、各ワードをビットシリアル態様で転送することができる。
【０６９６】
[実施の形態２１]
図１３３は、この発明の実施の形態２１に従う半導体集積回路装置の構成を示す回路ブロック図である。図１３３において、この半導体集積回路装置は、複数（ここでは４つとする）の機能ブロックＦＢＡ１～ＦＢＡ４と、４つの機能ブロックＦＢＢ１～ＦＢＢ４と、クロスバースイッチとを備える。機能ブロックＦＢＡ１～ＦＢＡ４は図中Ｘ方向に配列され、機能ブロックＦＢＢ１～ＦＢＢ４は図中Ｘ方向に配列され、機能ブロックＦＢＢ１～ＦＢＢ４はそれぞれ機能ブロックＦＢＡ１～ＦＢＡ４に対向して配置される。
【０６９７】
これらの機能ブロックＦＢＡ１～ＦＢＡ４およびＦＢＢ１～ＦＢＢ４は、これまでの実施の形態において説明した主演算回路に対応する回路ブロックであってもよく、また、別の各々所定の演算処理が割当てられる回路ブロックであってもよい。以下の実施の形態においては、機能ブロックは単に演算処理を行うことのできる構成であればよい。
【０６９８】
クロスバースイッチは、機能ブロックＦＢＡ１～ＦＢＡ４と機能ブロックＢＢ１～ＢＢ４との間に配置され、機能ブロックＦＢＡ１～ＦＢＡ４と機能ブロックＢＢ１～ＢＢ４とを１対１で任意の組合せで接続する。接続の組合せは、４！通りある。
【０６９９】
すなわち、クロスバースイッチは、セレクト信号線対ＬＬＰ１～ＬＬＰ８、データ信号線ＬＬ１～ＬＬ８、デコード回路５０１～５１６、およびワイヤードＯＲスイッチ５２１～５３６を含む。データ信号線ＬＬ５～ＬＬ８の各々は、図中機能ブロックＦＢＢ１から機能ブロックＦＢＢ４にわたって、Ｘ方向に延在するように設けられる。データ信号線ＬＬ１～ＬＬ４の各々は図中Ｘ方向と直交する図中Ｙ方向に延在し、データ信号線ＬＬ１～ＬＬ４の一方端はそれぞれ機能ブロックＦＢＡ１～ＦＢＡ４のデータ信号端子に接続され、データ信号線ＬＬ１～ＬＬ４の他方端はビアホールを介してそれぞれデータ信号線ＬＬ５～ＬＬ８に接続される。
【０７００】
ワイヤードＯＲスイッチ５２１～５２４；５２５～５２８；５２９～５３２；５３３～５３６は、それぞれデータ信号線ＬＬ５～ＬＬ８に対応して配置される。ワイヤードＯＲスイッチ５２１～５２４は、それぞれ対応のデータ信号線ＬＬ５と機能ブロックＦＢＢ１～ＦＢＢ４のデータ信号端子との間に接続され、それぞれデコード回路５０１～５０４の出力信号によって制御される。ワイヤードＯＲスイッチ５２５～５２８は、それぞれ対応のデータ信号線ＬＬ６と機能ブロックＦＢＢ１～ＦＢＢ４のデータ信号端子との間に接続され、それぞれデコード回路５０５～５０８の出力信号によって制御される。
【０７０１】
ワイヤードＯＲスイッチ５２９～５３２は、それぞれ対応のデータ信号線ＬＬ７と機能ブロックＦＢＢ１～ＦＢＢ４のデータ信号端子との間に接続され、それぞれデコード回路５０９～５１２の出力信号によって制御される。ワイヤードＯＲスイッチ５３３～５３６は、それぞれ対応のデータ信号線ＬＬ８と機能ブロックＦＢＢ１～ＦＢＢ４のデータ信号端子との間に接続され、それぞれデコード回路５１３～５１６の出力信号によって制御される。
【０７０２】
セレクト信号線対ＬＬＰ１～ＬＬＰ４の各々は図中Ｙ方向に延在し、セレクト信号線対ＬＬＰ１～ＬＬＰ４の一方端はそれぞれ機能ブロックＦＢＡ１～ＦＢＡ４のセレクト信号

端子対に接続され、セレクト信号線対ＬＬＰ１～ＬＬＰ４の他方端はそれぞれデコード回路５０１，５０６，５１１，５１６に接続される。

【０７０３】

セレクト信号線対ＬＬＰ５～ＬＬＰ８の各々は、図中Ｘ方向に延在し、機能ブロックＦＢＢ１から機能ブロックＦＢＢ４にわたって設けられる。セレクト信号対ＬＬＰ５は、ビアホールを介してセレクト信号対ＬＬＰ１に接続され、機能ブロックＦＢＡ１からのセレクト信号をデコード回路５０２，５０３，５０４の各々に伝達させる。セレクト信号対ＬＬＰ６は、ビアホールを介してセレクト信号対ＬＬＰ２に接続され、機能ブロックＦＢＡ２からのセレクト信号をデコード回路５０５，５０７，５０８の各々に伝達する。セレクト信号対ＬＬＰ７は、ビアホールを介してセレクト信号対ＬＬＰ３に接続され、機能ブロックＦＢＡ３からのセレクト信号をデコード回路５０９，５１０，５１２の各々に伝達させる。セレクト信号対ＬＬＰ８は、ビアホールを介してセレクト信号対ＬＬＰ４に接続され、機能ブロックＦＢＡ４からのセレクト信号をデコード回路５１３，５１４，５１５の各々に伝達させる。

【０７０４】

機能ブロックＦＢＡ１～ＦＢＡ４の各々からセレクト信号が出力されると、デコード回路５０１～５０４の出力信号のうちのいずれか１つの出力信号と、デコード回路５０５～５０８の出力信号のうちのいずれか１つの出力信号と、デコード回路５０９～５１２の出力信号のうちのいずれか１つの出力信号と、デコード回路５１３～５１６の出力信号のうちのいずれか１つの出力信号とが、活性化レベルの「Ｈ」レベルにされる。

【０７０５】

これにより、ワイヤードＯＲスイッチ５２１～５２４のうちのいずれか１つ（たとえばスイッチ５２２）と、ワイヤードＯＲスイッチ５２５～５２８のうちのいずれか１つ（たとえばスイッチ５２８）と、ワイヤードＯＲスイッチ５２９～５３２のうちのいずれか１つ（たとえばスイッチ５２９）と、ワイヤードＯＲスイッチ５３３～５３６のうちのいずれか１つ（たとえばスイッチ５３５）とが導通する。このようにして、機能ブロックＦＢＡ１～ＦＢＡ４と機能ブロックＦＢＢ１～ＦＢＢ４とが１対１で任意の組合せで接続される。

【０７０６】

図１３４は、クロスバースイッチの構成をより詳細に示す回路ブロック図である。図１３４において、ワイヤードＯＲスイッチ５２１～５２４は、それぞれＮチャネルＭＯＳトランジスタ（スイッチング素子）５２１ａ～５２４ａを含む。ＮチャネルＭＯＳトランジスタ５２１ａ～５２４ａの第１の電極（ソースまたはドレイン）はともに対応のデータ信号線ＬＬ５に接続され、それらの第２の電極（ドレインまたはソース）はそれぞれ機能ブロックＦＢＢ１～ＦＢＢ４のデータ信号端子に接続され、それらのゲートはそれぞれデコード回路５０１～５０４の出力信号を受ける。ＮチャネルＭＯＳトランジスタ５２１ａ～５２４ａは、それぞれデコード回路５０１～５０４の出力信号が「Ｈ」レベルにされた場合に導通する。他のワイヤードＯＲスイッチ５２５～５２８；５２９～５３２；５３３～５３６もワイヤードＯＲスイッチ５２１～５２４と同様である。

【０７０７】

セレクト信号線対ＬＬＰ１～ＬＬＰ８の各々は、２本の信号線を含む。セレクト信号は、２ビットのデータ信号を含む。デコード回路５０１～５０４の各々には、予め固有のセレクト信号が割り当てられている。たとえばデコード回路５０１～５０４には、それぞれ００，０１，１０，１１のセレクト信号が割り当てられている。デコード回路５０１は、セレクト信号が００の場合、すなわちセレクト信号に含まれる２ビットの信号がともに「Ｌ」レベルになった場合に「Ｈ」レベルを出力し、他の場合は「Ｌ」レベルを出力する。

【０７０８】

デコード回路５０２は、セレクト信号が０１の場合、すなわちセレクト信号に含まれる２ビットの信号がそれぞれ「Ｌ」レベルおよび「Ｈ」レベルになった場合に「Ｈ」レベルを出力し、他の場合は「Ｌ」レベルを出力する。デコード回路５０３は、セレクト信号が

１０の場合、すなわちセレクト信号に含まれる２ビットの信号がそれぞれ「Ｈ」レベルおよび「Ｌ」レベルになった場合に「Ｈ」レベルを出力し、他の場合は「Ｌ」レベルを出力する。デコード回路５０４は、セレクト信号が１１の場合、すなわちセレクト信号に含まれる２ビットの信号がともに「Ｈ」レベルになった場合に「Ｈ」レベルを出力し、他の場合は「Ｌ」レベルを出力する。他のデコード回路５０５～５０８；５０９～５１２；５１３～５１６も、デコード回路５０１～５０４と同様である。

【０７０９】

機能ブロックＦＢＡ１からセレクト信号が出力されると、デコード回路５０１～５０４のうちのいずれか１つ（たとえばデコード回路１）の出力信号が「Ｈ」レベルになり、そのデコード回路５０１に対応するＮチャネルＭＯＳトランジスタ５２１ａが導通し、機能ブロックＦＢＡ１のデータ信号端子と機能ブロックＦＢＢ１のデータ信号端子とが接続される。

【０７１０】

次に、この実施の形態２１の効果について説明する。このクロスバースイッチでは、Ｙ方向の配線はデータ信号線ＬＬ１～ＬＬ４とセレクト信号線対ＬＬＰ１～ＬＬＰ４の合計１２本であり、Ｘ方向の配線はデータ信号線ＬＬ５～ＬＬ８とセレクト信号線対ＬＬＰ５～ＬＬＰ８の合計１２本である。スイッチ５２１～５３６は１６個であり、デコード回路５０１～５１６は１６個である。

【０７１１】

図１３３のクロスバースイッチと同じ構成で、Ｎ＝２＾ｍ個の機能ブロックＦＢＡ１～ＦＢＡＮとＮ個の機能ブロックＦＢＢ１～ＦＢＢＮとを接続するクロスバースイッチを構成することを考える。記号＾は、べき乗を示し、Ｎは２のｍ乗に等しい。この場合、Ｙ方向の配線は（ｍ＋１）Ｎ本となり、Ｘ方向の配線は（ｍ＋１）Ｎ本となり、スイッチはＮ×Ｎ個となり、ｍ入力のデコード回路はＮ×Ｎ個となる。

【０７１２】

ここで、従来のスイッチマトリクスのように信号線の交差部に対応してスイッチを配置する場合、たとえばＮ＝１２８（ｍ＝７）の場合、一方側の機能ブロックに対して１２８×１２８、他方側の機能ブロックに対して１２８×１２８および制御信号線が、１２８×１２８のＹ方向の信号線がＸ方向に延在する１２８本の信号線を介して結合され、合計Ｙ方向の配線が３×１２８×１２８＝４９１５２本となり、Ｘ方向の配線が１２８本となり、スイッチが３２７６８個となる（＝２×１２８×１２８；一方および他方側の機能ブロックに対してスイッチが配置される）。一方、本発明では、Ｙ方向の配線が８×１２８＝１０２４本となり、Ｘ方向の配線が８×１２８＝１０２４本となり、スイッチが１６３８４となり、デコーダ回路が１６３８４個となる。したがって、本発明は、配線数が少なくなる。また、上述のような単純なスイッチマトリクス構成では、Ｙ方向の配線が密集するとともに、Ｙ方向の配線と機能ブロックとの間のスイッチが密集するのに対し、本発明では配線およびスイッチがＸ方向とＹ方向に均等に配分される。よって、本発明は、従来例よりもレイアウト面積が小さくなる。

【０７１３】

また、本発明では、機能ブロックＦＢＡ１～ＦＢＡ４と機能ブロックＦＢＢ１～ＦＢＢ４との間で双方向のデータ転送を行う場合でも、基本的には図１３３と同じ構成で実現できる。つまり、接続先情報を含むセレクト信号を機能ブロックＦＢＢ１～ＦＢＢ４から発信すればよい。したがって、双方向のデータ転送を行う場合は、本発明と単純スイッチマトリクス構成とのレイアウト面積の差は一層大きくなる。

【０７１４】

次に、本発明の用途について説明する。図１３３で示した機能ブロックＦＢＡ１～ＦＢＡ４，ＦＢＢ１～ＦＢＢ４の各々をＡＬＵ（Arithmetic and Logic Unit）のユニットセルで構成する。ＡＬＵユニットセルは、各種演算ユニットセルで構成されている。複数の基本演算ユニットセル（Ａｄｄ，Ｍｕｌ，Ｄｉｖ，Ｓｕｂ，Ｓｈｉｆｔ等）を有機的に結合することにより、機能エレメントを構成できる。図１３３に示したように、複数のＡＬ

Uユニットセルを上下に配置し、クロスバースイッチの動作をプログラミングすることにより、機能エレメントを構成することができる。この場合、結合の方向を双方向にすることにより、大きな機能エレメントを構成できる。また、クロスバースイッチのプログラミングすなわちＰ＆Ｒ（Place and router)を再構築することにより、リコンフィギュラブルなロジックを構成することができる。

【０７１５】

[実施の形態２２]

図１３５は、この発明の実施の形態２２に従う半導体集積回路装置の要部を示す回路ブロック図であって、図１３４と対比される図である。図１３５を参照して、この半導体集積回路装置が実施の形態２１の半導体集積回路装置と異なる点は、デコード回路５０１～５１６の各々の出力ノードにラッチ回路５３７が追加されている点である。たとえばデコード回路５０１に対応するラッチ回路５３７は、ラッチ信号φＬに応答して対応のデコード回路１の出力信号をラッチして対応のワイヤードＯＲスイッチ５２１に含まれるＮチャネルＭＯＳトランジスタ５２１ａのゲートに与える。

【０７１６】

この実施の形態２２では、ラッチ回路５３７群によってデコード回路５０１～５１６の出力信号をラッチした後は、セレクト信号線対ＬＬＰ１～ＬＬＰ８を開放して他の用途に用いることができる。

【０７１７】

[実施の形態２３]

図１３６は、この発明の実施の形態２３に従う半導体集積回路装置の構成を示すブロック図である。図１３６を参照して、この半導体集積回路装置が図１３３に示す半導体集積回路装置と異なる点は、デコード回路５０１～５１６がデコード回路部ＤＤで置換され、冗長機能ブロックＦＲＢＡ，冗長デコード回路部ＲＤＤ、冗長ワイヤードＯＲスイッチ部ＲＳＳ、および冗長機能ブロックＦＲＢＢが追加されている点である。ワイヤードＯＲスイッチ部ＳＳは、図１３３のワイヤードＯＲスイッチ５２１～５３６を含む。

【０７１８】

デコード回路部ＤＤは、図１３３のデコード回路５０１～５１６に加えて、機能ブロックＦＢＢ１のうちの不良な機能ブロックを指定するセレクト信号を記憶するためのプログラム回路を含む。たとえば機能ブロックＦＢＢ４が不良な場合は、機能ブロックＦＢＢ４を指定するセレクト信号がデコード回路ＤＤのプログラム回路に格納される。

【０７１９】

たとえば機能ブロックＦＢＡ１から正常な機能ブロックＦＢＢ２を指定するセレクト信号が与えられた場合は、デコード回路部ＤＤおよびワイヤードＯＲスイッチ部ＳＳは、図１３３のデコード回路５０１～５１６およびワイヤードＯＲスイッチ５２１～５３６と同様に動作し、機能ブロックＦＢＡ１と機能ブロックＦＢＢ２を接続する。

【０７２０】

たとえば機能ブロックＦＢＡ１から不良な機能ブロックＦＢＢ４を指定するセレクト信号が与えられた場合は、デコード回路部ＤＤおよびワイヤードＯＲスイッチ部ＳＳのうちの不良な機能ブロックＦＢＢ４に対応する部分が非活性化されるとともに、冗長デコード回路部ＲＤＤおよび冗長ワイヤードＯＲスイッチ部ＲＳＳが活性化される。冗長デコード回路部ＲＤＤおよび冗長ワイヤードＯＲスイッチ部ＲＳＳは、機能ブロックＦＢＡ１と冗長機能ブロックＦＲＢＢを接続する。このようにして、不良な機能ブロックＦＢＢ４が冗長機能ブロックＲＦＢＢで置換される。なお、双方向のデータ転送を行う場合は、同様にして、機能ブロックＦＢＡ１～ＦＢＡ４のうちの不良な機能ブロックが冗長機能ブロックＦＲＢＡで置換される。

【０７２１】

この実施の形態３では、機能ブロックおよびクロスバースイッチに冗長機能を設けたので、通常は冗長機能を持たすことができないランダムロジックにも冗長機能を持たすことができ、歩留の向上を図ることができる。

【０７２２】
[実施の形態２４]
図１３７は、この発明の実施の形態２４に従う半導体集積回路装置の構成を示すブロック図である。図１３７において、この半導体集積回路装置は、多数の機能ブロックＦＢＡ１～ＦＢＡｎ（ただし、ｎは２以上の整数である）と、多数の機能ブロックＦＢＢ１～ＦＢＢｎと、クロスバースイッチとを備え、クロスバースイッチは、グローバルデコード回路部ＧＤＤ、ローカルデコード回路部ＬＤＤおよびワイヤードＯＲスイッチ部ＳＳを含む。
【０７２３】
ワイヤードＯＲスイッチ部ＳＳは、図１３３で説明したように、機能ブロックＦＢＡ１～ＦＢＡｎの各々に対応してｎ個のワイヤードＯＲスイッチを含む。ｎ個のワイヤードＯＲスイッチは、それぞれ機能ブロックＦＢＢ１～ＦＢＢｎに対応している。
【０７２４】
ｎ個のワイヤードＯＲスイッチは、各々がＡ個のワイアードＯＲスイッチを含むＢ個のスイッチグループに分割されている。グローバルデコード回路部ＧＤＤは、各機能ブロックＦＢＡからのグローバルセレクト信号に従ってその機能ブロックＦＢＡに対応するＢ個のスイッチグループのうちのいずれかのスイッチグループを選択する。ローカルデコード回路部ＬＤＤは、各機能ブロックＦＢＡからのローカルセレクト信号に従って、その機能ブロックＦＢＡに対応し、かつグローバルデコード回路部ＧＤＤによって選択されたスイッチグループに属するＡ個のワイヤードＯＲスイッチのうちのいずれかのワイヤードＯＲスイッチを選択し、そのワイヤードＯＲスイッチを導通させる。
【０７２５】
たとえばｎ＝１６の場合は、ワイヤードＯＲスイッチ部ＳＳは、機能ブロックＦＢＡ１～ＦＢＡ１６の各々に対応して１６個のワイヤードＯＲスイッチを含む。１６個のワイヤードＯＲスイッチは、それぞれ機能ブロックＦＢＢ１～ＦＢＢ１６に対応している。
【０７２６】
１６個のワイヤードＯＲスイッチは、４個ずつ４個のスイッチグループに分割されている。グローバルデコード回路部ＧＤＤは、図１３８に示すように、それぞれ４個のスイッチグループに対応する４個のグローバルデコード回路５４０を含む。ローカルデコード回路ＬＤＤは、それぞれ１６個のワイヤードＯＲスイッチに対応する１６個のローカルデコード回路５４１を含む。
【０７２７】
対応の機能ブロックＦＢＡからグローバルセレクト信号ＧＧＳ１，ＧＧＳ２が出力されると、４個のグローバルデコード回路５４０のうちのいずれか１つのグローバルデコード回路５４０の出力信号が「Ｈ」レベルになり、そのグローバルデコード回路５４０に対応する４個のローカルデコード回路５４１が活性化される。対応の機能ブロックＦＢＡからローカルセレクト信号ＬＬＳ１～ＬＬＳ４が出力されると、活性化された４個のローカルデコード回路５４１のうちのいずれか１つのローカルデコード回路５４１の出力信号が「Ｈ」レベルになり、そのローカルデコード回路５４１に対応するワイヤードＯＲスイッチが導通する。このようにして、機能ブロックＦＢＡ１～ＦＢＡ１６と機能ブロックＦＢＢ１～ＦＢＢ１６が１対１で所望の組合せで接続される。
【０７２８】
この実施の形態２４では、デコード回路部を階層化したので、セレクト信号用の配線数を低減化することができ、デコード回路部のコンパクト化を図ることができる。
【０７２９】
[実施の形態２５]
図１３９は、この発明の実施の形態２５に従う半導体集積回路装置の構成を示すブロック図である。図１３９において、この半導体集積回路は、複数（図１３９では５つ）の機能ブロックＦＢＡ１～ＦＢＡ５と、５つの機能ブロックＦＢＢ１～ＦＢＢ５と、クロスバースイッチとを備える。機能ブロックＦＢＡ１～ＦＢＡ５は、複数のグループに分割され

ており、図１３９では、機能ブロックＦＢＡ１～ＦＢＡ３が属する第１のグループと、機能ブロックＦＢＡ４，ＦＢＡ５が属する第２のグループに分割される。

【０７３０】

クロスバースイッチのデコード回路部は、第１のグループに対応するデコード回路部ＤＤ１と、第２のグループに対応するデコード回路部ＤＤ２に分割される。クロスバースイッチのワイヤードＯＲスイッチ部は、第１のグループに対応するワイヤードＯＲスイッチ部ＳＳ１と、第２のグループに対応するワイヤードＯＲスイッチ部ＳＳ２に分割される。機能ブロックＦＢＢ１～ＦＢＢ５は、第１のグループに対応する機能ブロックＦＢＢ１～ＦＢＢ３と、第２のグループに対応する機能ブロックＦＢＢ４，ＦＢＢ５に分割される。

【０７３１】

この半導体集積回路装置では、機能ブロックＦＢＡ１～ＦＢＡ３、デコード回路部ＤＤ１、ワイヤードＯＲスイッチ部ＳＳ１および機能ブロックＦＢＢ１～ＦＢＢ３に電源電圧ＶＣＣ１を供給する電源供給線ＰＰＬ１と、機能ブロックＦＢＡ４、ＦＢＡ５、デコード回路部ＤＤ２、ワイヤードＯＲスイッチ部ＳＳ２および機能ブロックＦＢＢ４、ＦＢＢ５に電源電圧ＶＣＣ２を供給する電源供給線ＰＰＬ２とが別々に設けられている。したがって、たとえば機能ブロックＦＢＡ１～ＦＢＡ３およびそれに関連する部分のみを活性化させる場合は、電源供給線ＰＰＬ１に電源電圧ＶＣＣ１を供給し、電源供給線ＰＰＬ２への電源電圧ＶＣＣ２の供給を停止することにより、不要な電力消費を削減することができ、消費電力の低減化を図ることができる。

【０７３２】

また、この半導体集積回路装置では、２つの機能エレメントを同時に構成できるので、２つの同じ機能のエレメントを構成することにより、演算のパラレル処理を行うことができ、高性能なプロセッシング機能を実現することができる。

【産業上の利用可能性】

【０７３３】

この発明は、一般に、データ処理システムに対して適用可能である。特に、画像または音声データなどの大量のデータを処理することが要求される処理システムに対して適用することにより、高速演算処理システムを実現することができる。

【０７３４】

なお、チップ構成としては、１つの主演算回路部分が１チップ（半導体チップ）で構成されてもよく、また１つの基本演算ブロックが１チップ（半導体チップ）で構成されてもよい。また、１つの半導体演算装置が、システムＬＳＩのように、１チップ（半導体チップ）で構成されてもよい。

【図面の簡単な説明】

【０７３５】

【図１】この発明に従う半導体装置の全体の構成を概略的に示す図である。

【図２】図１に示す主演算回路の構成を概略的に示す図である。

【図３】図２に示すメモリセルの構成の一例を示す図である。

【図４】この発明の実施の形態１における半導体装置の演算操作シーケンスを概略的に示す図である。

【図５】この発明の実施の形態１における半導体装置の演算処理動作時の内部タイミングを示す図である。

【図６】この発明の実施の形態１に従う半導体装置の要部の構成をより具体的に示す図である。

【図７】図６に示すＡＬＵ群に含まれるＡＬＵの構成を概略的に示す図である。

【図８】この発明の実施の形態１における半導体装置の主演算回路における演算処理動作の内部タイミングを示す図である。

【図９】この発明の実施の形態１における半導体演算装置のアドレス領域を指定するポインタを概略的に示す図である。

【図１０】この発明の実施の形態２に従う半導体演算装置で利用されるメモリセルの構成

の一例を示す図である。
【図１１】この発明の実施の形態２における半導体演算装置の演算処理動作時の内部タイミングを示す図である。
【図１２】この発明の実施の形態２における半導体演算装置の主演算回路の構成をより具体的に示す図である。
【図１３】この発明の実施の形態３に従う半導体演算装置の主演算回路の構成を概略的に示す図である。
【図１４】図１３に示す主演算回路の演算処理動作時の内部タイミングを示す図である。
【図１５】この発明の実施の形態３に従う主演算回路の構成をより具体的に示す図である。
【図１６】この発明の実施の形態４に従う主演算回路の構成を概略的に示す図である。
【図１７】図１６に示すメモリセルの構成の一例を示す図である。
【図１８】この発明の実施の形態５におけるメモリマットにおける演算対象データの分布を概略的に示す図である。
【図１９】この発明の実施の形態５における半導体演算装置のメモリマット内の演算対象データの分布を概略的に示す図である。
【図２０】この発明の実施の形態５に従う半導体演算装置の主演算回路の要部の構成を概略的に示す図である。
【図２１】この発明の実施の形態５に従う主演算回路の構成をより具体的に示す図である。
【図２２】この発明の実施の形態６に従う主演算回路の要部の構成を概略的に示す図である。
【図２３】この発明の実施の形態７に従う主演算回路の要部の構成を概略的に示す図である。
【図２４】この発明の実施の形態８に従う主演算回路の要部の構成を概略的に示す図である。
【図２５】図２４に示すＡＬＵの可変構成の一例を概略的に示す図である。
【図２６】この発明の実施の形態９における処理システムの構成の一例を示す図である。
【図２７】この発明の実施の形態９に従う処理システムの構成の一例を概略的に示す図である。
【図２８】この発明の実施の形態９における半導体演算装置内の主演算回路に含まれるＡＬＵの構成の一例を概略的に示す図である。
【図２９】この発明の実施の形態９に従う半導体演算装置を利用する処理システムの構成の一例を示す図である。
【図３０】この発明の実施の形態１０に従う主演算回路の要部の構成を概略的に示す図である。
【図３１】図３０に示すワード線冗長救済回路の構成を概略的に示す図である。
【図３２】図３０に示すビット線冗長救済回路の構成を概略的に示す図である。
【図３３】この発明の実施の形態１１に従う基本演算ブロックの要部の構成を概略的に示す図である。
【図３４】図３３に示す単位ＡＬＵ回路ブロックの構成を概略的に示す図である。
【図３５】この発明の実施の形態１１におけるレジスタ命令を一覧にして示す図である。
【図３６】この発明の実施の形態１１におけるＡＬＵ命令を一覧にして示す図である。
【図３７】この発明の実施の形態１１におけるＡＬＵロード／ストア命令を一覧にして示す図である。
【図３８】この発明の実施の形態１１におけるエントリ間データ移動命令を一覧にして示す図である。
【図３９】この発明の実施の形態１１におけるＡＬＵ算術演算命令を一覧にして示す図である。
【図４０】この発明の実施の形態１１におけるＡＬＵ論理演算命令を一覧にして示す図で

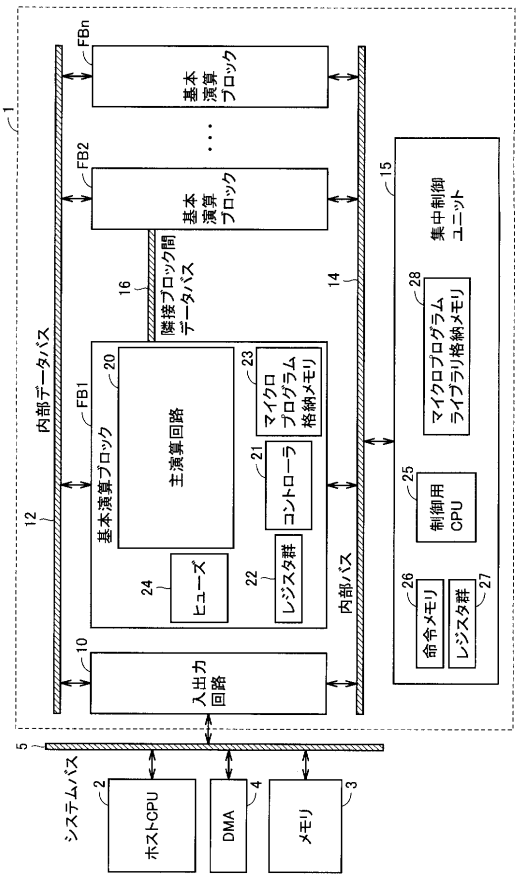
ある。
【図４１】この発明の実施の形態１１におけるプログラムの一例を示す図である。
【図４２】図４１における加算プログラムのデータの流れを概略的に示す図である。
【図４３】この発明の実施の形態１１における減算プログラムを示す図である。
【図４４】この発明の実施の形態１１における乗算プログラムを示す図である。
【図４５】図４４に示す乗算プログラムのデータの流れを模式的に示す図である。
【図４６】この発明の実施の形態１１における除算時のエントリのアドレスの割当を示す図である。
【図４７】この発明の実施の形態１１における除算プログラムを示す図である。
【図４８】（Ａ）－（Ｆ）は、図４７に示す除算プログラムのデータの流れを模式的に示す図である。
【図４９】この発明の実施の形態１２における単位ＡＬＵ回路ブロックの構成を概略的に示す図である。
【図５０】２次のブースアルゴリズムの部分積生成手順を一覧にして示す図である。
【図５１】２次のブースアルゴリズムにおける部分積生成処理を模式的に示す図である。
【図５２】２次のブースアルゴリズムに従う乗算の一例を示す図である。
【図５３】この発明の実施の形態１２におけるレジスタ命令を一覧にして示す図である。
【図５４】この発明の実施の形態１２におけるＡＬＵ命令を一覧にして示す図である。
【図５５】この発明の実施の形態１２におけるＡＬＵセット／クリア命令を一覧にして示す図である。
【図５６】この発明の実施の形態１２におけるＡＬＵロード／ストア命令を一覧にして示す図である。
【図５７】この発明の実施の形態１２におけるエントリ間データ移動命令を一覧にして示す図である。
【図５８】この発明の実施の形態１２におけるＡＬＵ算術演算命令を一覧にして示す図である。
【図５９】この発明の実施の形態１２におけるＡＬＵ算術演算命令を一覧にして示す図である。
【図６０】この発明の実施の形態１２におけるレジスタ格納値とデータ処理との対応を一覧にして示す図である。
【図６１】この発明の実施の形態１２におけるＡＬＵ論理演算命令を一覧にして示す図である。
【図６２】この発明の実施の形態１２における符号付き乗算プログラムを示す図である。
【図６３】この発明の実施の形態１２における乗算時のエントリのアドレスポインタを示す図である。
【図６４】この発明の実施の形態１２における乗算時のデータビットの流れを模式的に示す図である。
【図６５】この発明の実施の形態１２における乗算処理を模式的に示す図である。
【図６６】（Ａ）は、この発明の実施の形態１２における加算処理時のエントリのポインタを示し、（Ｂ）は、加算プログラムを示す図である。
【図６７】（Ａ）は、この発明の実施の形態１２における減算処理時のアドレスポインタを示し、（Ｂ）は、減算プログラムを示す図である。
【図６８】この発明の実施の形態１２における符号なし乗算プログラムを示す図である。
【図６９】この発明の実施の形態１２における除算プログラムを示す図である。
【図７０】この発明の実施の形態１３における単位ＡＬＵ回路ブロックの構成を概略的に示す図である。
【図７１】この発明の実施の形態１３におけるポインタレジスタ命令を一覧にして示す図である。
【図７２】この発明の実施の形態１３におけるＡＬＵの１ビット動作時の命令を一覧にして示す図である。

【図７３】この発明の実施の形態１３における２ビット動作時のＡＬＵロード／ストア命令を一覧にして示す図である。
【図７４】この発明の実施の形態１３における１ビット動作時のエントリ間データ移動命令を一覧にして示す図である。
【図７５】この発明の実施の形態１３における２ビット動作時のエントリ間データ移動命令を一覧にして示す図である。
【図７６】この発明の実施の形態１３における１ビット動作時のＡＬＵ算術演算命令を一覧にして示す図である。
【図７７】この発明の実施の形態１３における２ビット動作時のＡＬＵ算術演算命令を一覧にして示す図である。
【図７８】この発明の実施の形態１３における２次のブースアルゴリズム実行時の各レジスタのビット値と対応の処理を一覧にして示す図である。
【図７９】この発明の実施の形態１３における乗算プログラムを示す図である。
【図８０】この発明の実施の形態１３における単位ＡＬＵ回路ブロックの接続を概略的に示す図である。
【図８１】この発明の実施の形態１３における乗算プログラム実行時のデータの流れを模式的に示す図である。
【図８２】この発明の実施の形態１３における乗算実行時のデータの流れを模式的に示す図である。
【図８３】この発明の実施の形態１３における単位ＡＬＵ回路ブロックの１ビット演算処理時の接続を概略的に示す図である。
【図８４】この発明の実施の形態１３における加算プログラムを示す図である。
【図８５】この発明の実施の形態１３における減算プログラムを示す図である。
【図８６】この発明の実施の形態１３における符号なし乗算プログラムを示す図である。
【図８７】この発明の実施の形態１３における除算プログラムを示す図である。
【図８８】この発明の実施の形態１３におけるエントリへのデータ書込経路を概略的に示す図である。
【図８９】この発明の実施の形態１４におけるＡＬＵ制御の構成を概略的に示す図である。
【図９０】この発明の実施の形態１５に従うＡＬＵ制御の構成を概略的に示す図である。
【図９１】この発明の実施の形態１６に従う半導体演算装置の構成を概略的に示す図である。
【図９２】この発明の実施の形態１６におけるメモリ間のデータ転送経路を概略的に示す図である。
【図９３】この発明の実施の形態１６におけるデータ転送の制御の構成を概略的に示す図である。
【図９４】この発明の実施の形態１６における大容量メモリと主演算回路とのデータ転送経路を概略的に示す図である。
【図９５】この発明の実施の形態１７に従う半導体演算装置の構成を概略的に示す図である。
【図９６】この発明の実施の形態１７のメモリ構成を概略的に示す図である。
【図９７】この発明の実施の形態１７における大容量メモリの構成を概略的に示す図である。
【図９８】図９７に示すメモリ構成のデータ転送動作を示す図である。
【図９９】この発明の実施の形態１８におけるＡＬＵ間接続の構成を概略的に示す図である。
【図１００】図９９に示す隣接ブロック間接続バスの１つのバス線の関連する部分の構成を概略的に示す図である。
【図１０１】この発明の実施の形態１８の変更例１の構成を概略的に示す図である。
【図１０２】この発明の実施の形態１８の変更例２の構成を概略的に示す図である。

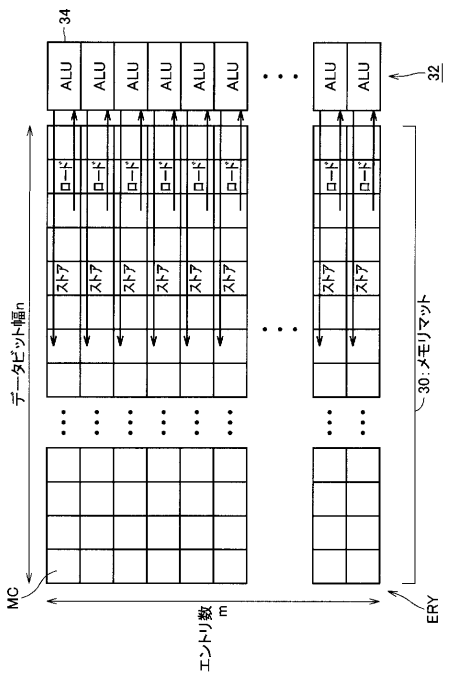
【図１０３】この発明の実施の形態１９に従うＡＬＵ間相互接続用スイッチ回路の構成を概略的に示す図である。
【図１０４】図１０３に示すＡＬＵ間相互接続用スイッチ回路の構成を概略的に示す図である。
【図１０５】この発明の実施の形態１９の変更例１の構成を概略的に示す図である。
【図１０６】図１０５に示すプログラマブルスイッチ回路の構成の一例を概略的に示す図である。
【図１０７】図１０６に示すプログラマブルスイッチ回路の第１の接続状態を示す図である。
【図１０８】図１０６に示すプログラマブルスイッチ回路の第２の接続状態を示す図である。
【図１０９】図１０６に示すプログラマブルスイッチ回路の第３の接続状態を示す図である。
【図１１０】この発明の実施の形態１９の変更例１におけるＡＬＵ間接続用スイッチ回路の接続の一例を示す図である。
【図１１１】この発明の実施の形態１９の変更例１におけるプログラマブルスイッチ回路の他の接続の態様を示す図である。
【図１１２】この発明の実施の形態１９の変更例２のＡＬＵの配置を模式的に示す図である。
【図１１３】図１１２におけるＡＬＵ配置におけるＡＬＵ接続用スイッチ回路の配線を概略的に示す図である。
【図１１４】図１１３に示す隣接ブロック間データバスの接続の一例を概略的に示す図である。
【図１１５】この発明の実施の形態１９の変更例３の構成を概略的に示す図である。
【図１１６】この発明の実施の形態１９の変更例４の構成を概略的に示す図である。
【図１１７】この発明の実施の形態１９の変更例５におけるプログラマブルスイッチ回路の構成を概略的に示す図である。
【図１１８】図１１７に示す送受信データレジスタの構成の一例を概略的に示す図である。
【図１１９】この発明の実施の形態１９の変更例６のＡＬＵ間接続バスの構成を概略的に示す図である。
【図１２０】この発明の実施の形態２０の入出力回路の構成を概略的に示す図である。
【図１２１】図１２０に示す直交変換回路のデータ出力部の構成を概略的に示す図である。
【図１２２】図１２１に示す直交変換回路のデータ変換操作を概略的に示す図である。
【図１２３】図１２１に示す変換素子の構成の一例を概略的に示す図である。
【図１２４】図１２０に示す直交変換回路のデータ入力部の構成を概略的に示す図である。
【図１２５】図１２４に示す直交変換回路のデータ入力部のデータ変換操作を模式的に示す図である。
【図１２６】図１２０に示すクロスバースイッチの要部の構成を概略的に示す図である。
【図１２７】図１２０に示すクロスバースイッチの接続制御信号発生部の構成を概略的に示す図である。
【図１２８】図１２４に示すクロスバースイッチの全体の構成を概略的に示す図である。
【図１２９】図１２６に示すデコーダのデコード信号と接続バスの対応関係を示す図である。
【図１３０】図１２０に示すセレクタの構成の一例を示す図である。
【図１３１】図１３０に示すセレクタのデータ転送経路の一例を示す図である。
【図１３２】図１３０に示すセレクタのデータ転送経路の他の例を示す図である。
【図１３３】この発明の実施の形態２１による半導体集積回路装置の構成を示すブロック

図である。
【図１３４】図１３３に示したデコード回路およびワイヤードＯＲスイッチの構成を示す回路図である。
【図１３５】この発明の実施の形態２２による半導体集積回路装置の要部を示すブロック図である。
【図１３６】この発明の実施の形態２３による半導体集積回路装置の構成を示すブロック図である。
【図１３７】この発明の実施の形態２４による半導体集積回路装置の構成を示すブロック図である。
【図１３８】図１３７に示したグローバルデコード回路部およびローカルデコード回路の構成を示す回路図である。
【図１３９】この発明の実施の形態２５による半導体集積回路装置の構成を示すブロック図である。
【符号の説明】
【０７３６】
１　半導体演算装置、ＦＢ１－ＦＢｍ　基本演算ブロック、２　ホストＣＰＵ、３　メモリ、５　システムバス、１０　入出力回路、１２　内部データバス、１４　内部バス、１５　集中制御ユニット、６　隣接ブロック間データバス、２０　主演算回路、２１　コントローラ、２２　レジスタ群、２３　マイクロプログラム格納メモリ、２４　ヒューズ、３０　メモリマット、４０　センスアンプ群、４２　ライトドライバ群、４４　ＡＬＵ相互接続用スイッチ回路、４６　ロウデコーダ、４８　入出力回路、３２　演算処理ユニット群（ＡＬＵ群）、５０　算術演算論理回路、５２　Ａレジスタ、５４　Ｘレジスタ、５６　Ｃレジスタ、５８　Ｍレジスタ、６０　ライトドライバ、６２　センスアンプ、６５　ＡＬＵ間接続回路、ＮＱ１－ＮＱ８　ＮチャネルＭＯＳトランジスタ、ＰＱ１，ＰＱ２　ＰチャネルＭＯＳトランジスタ、３６ｒ　リード用ロウデコーダ、３６ｗ　ライト用ロウデコーダ、４０Ａ，４０Ｂ　センスアンプ群、４２Ａ，４２Ｂ　ライトドライバ群、２００　ＡＬＵ内内部データ線、２１０，２１１　全加算器、２２０　ＸＨレジスタ、２２１　ＸＬレジスタ、２２２　セレクタ、２１７　選択反転回路、２０８　Ｖレジスタ、２０７　Ｎレジスタ、２０５　Ｆレジスタ、２２２　Ｄレジスタ、ＳＷａ－ＳＷｆ　スイッチ回路、２２５　スイッチ回路、２５０　大容量メモリ、ＧＢＳ　グローバルデータバス、ＢＫ０－ＢＫｑ　バンク、２６０　隣接ブロックエントリ間接続配線、２６２　隣接ブロックエントリフィードバック接続配線、２６２ａ，２６２ｂ　隣接ブロックエントリ間接続配線、ＰＳＷ　プログラマブルスイッチ回路、ＮＢＡａ，ＮＢＡｂ　隣接ブロック間データバス、ＰＳＷ　プログラマブルスイッチ回路、３１５ｕ，３１５ｄ　マルチプレクサ（ＭＵＸ）、３２０ａｕ，３２０ｄｕ，３２０ａｄ－３２０ｄｄ　シフト信号線、３１７　送受信データレジスタ、３１９　ＡＬＵユニット、３２０　Ｘレジスタ、３２５　送信レジスタ、３２６　受信レジスタ、３４０，３４２　メモリセルマット上配線、４００　直交変換回路、４０２　クロスバースイッチ、４０４　セレクタ、ＴＸＦ００－ＴＸＦ（ｋ－１），（ｊ－１）　変換素子、ＴＸＦＩ０－ＴＸＦＩ７　入力変換素子、４３０　ワードマスク制御回路、ＤＤＤ－ＤＤＤ（ａ－１）　デコーダ、ＤＳＷ０－ＤＳＷ（ｍ－１）　選択スイッチ回路、ＳＡＣ　センスアンプ回路、ＸＧ００－ＸＧ３４　レジスタ回路、ＳＳＧ０－ＳＳＧ４　選択ゲート、ＤＲＶ０－ＤＲＶ４　ドライバ、４６０　接続情報保持回路、４６０ａ　コンテキスト情報格納回路、４６２　接続ルート決定回路、４６４　スイッチマトリクス回路、４６４ａ　スイッチ列、４６２ａ　デコーダ群、ＴＧＷ０－ＴＧＷ３　選択ゲート、　ＦＢＡ，ＦＢＢ　機能ブロック、ＦＲＢＡ，ＦＲＢＢ　冗長機能ブロック、ＬＬ，ＬＬＸ，ＬＬＹ　データ信号線、ＬＬＰ　セレクト信号線対、５０１～５１６　デコード回路、５２１～５３６，５５１，５５２，５５３　スイッチ、５２１ａ～５２４ａ，５５１ａ，５５２ａ　ＮチャネルＭＯＳトランジスタ、５３７　ラッチ回路、ＤＤ　デコード回路部、ＧＤＤ　グローバルデコード回路部、ＬＤＤ　ローカルデコード回路部、ＳＳ　ワイヤードＯＲスイッチ部。

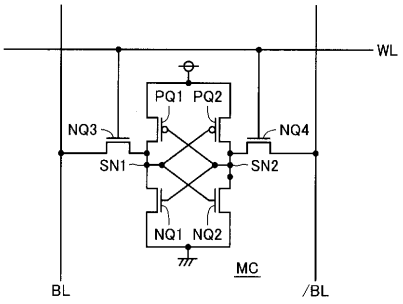
【図１】


1
内部データバス
FBn
基本演算ブロック
FB2
基本演算ブロック
隣接ブロック間データバス
16
14
15
集中制御ユニット
FB1
基本演算ブロック
20
主演算回路
23
マイクロプログラム格納メモリ
21
コントローラ
12
24
ヒューズ
22
レジスタ群
内部バス
28
マイクロプログラムライブラリ格納メモリ
25
制御用CPU
26
命令メモリ
27
レジスタ群
10
入出力回路
5
システムバス
2
ホストCPU
4
DMA
3
メモリ

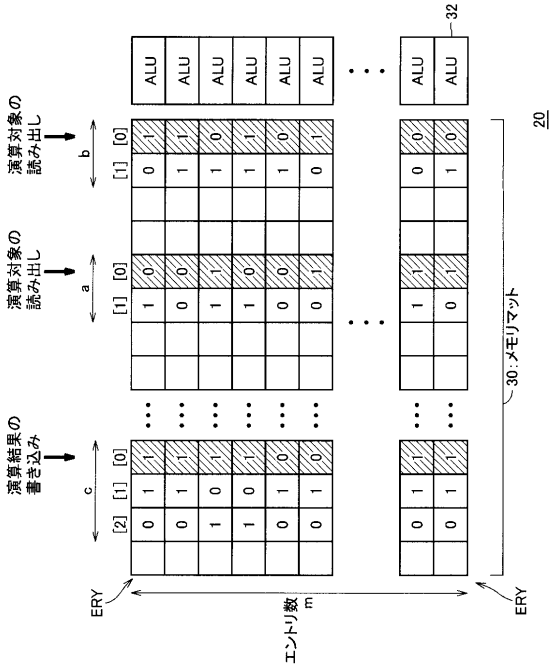
【図２】


34
ALU
32
ロード
ストア
データビット幅n
MC
エントリ数m
ERY
30:メモリマット
20

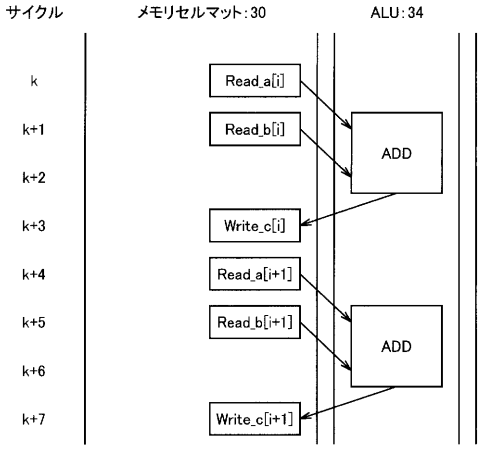
【図３】


WL
PQ1
PQ2
NQ3
NQ4
SN1
SN2
NQ1
NQ2
MC
BL
/BL

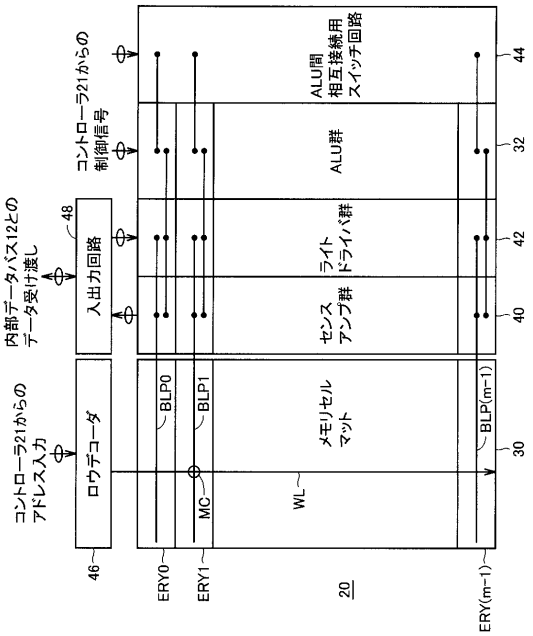
【図４】


ALU
32
演算対象の読み出し
演算結果の書き込み
a
b
c
[0]
[1]
[2]
20
30:メモリマット
ERY
エントリ数 m

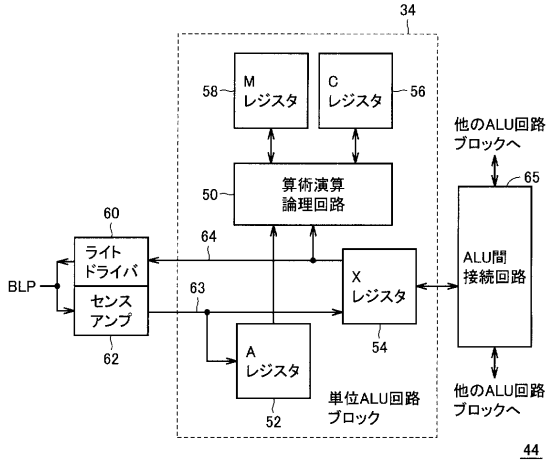
【図５】


サイクル
メモリセルマット:30
ALU:34
k
k+1
k+2
k+3
k+4
k+5
k+6
k+7
Read_a[i]
Read_b[i]
ADD
Write_c[i]
Read_a[i+1]
Read_b[i+1]
ADD
Write_c[i+1]

【図６】


ALU間相互接続用スイッチ回路
コントローラ21からの制御信号
内部データバス12とのデータ受け渡し
入出力回路
ライトドライバ群
センスアンプ群
ALU群
メモリセルマット
ロウデコーダ
コントローラ21からのアドレス入力
BLP0
BLP1
BLP(m-1)
WL
MC
ERY0
ERY1
ERY(m-1)
20
30
32
40
42
44
46
48

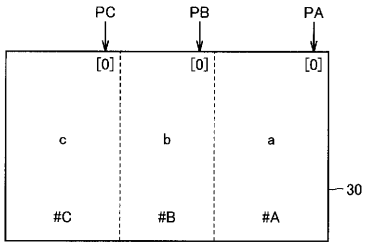
【図７】



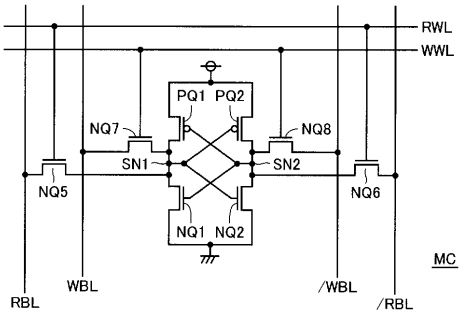
【図８】

サイクル	イベント	レジスタ値 X	A	C	M
k-1	Mレジスタに1をセット、Cレジスタを0にクリア。	0	0	0	1
k	a[i]の読み出し。	0	0	0	1
k+1	b[i]の読み出し。	a[i]	0	0	1
k+2	ALU部で演算実行	a[i]	b[i]	0	1
k+3	c[i]に加算結果を書き込み。	a[i]+b[i]	b[i]	C[i]	1
k+4	a[i+1]の読み出し。	a[i]+b[i]	b[i]	C[i]	1
k+5	b[i+1]の読み出し。	a[i+1]	b[i]	C[i]	1
k+6	ALU部で演算実行	a[i+1]	b[i+1]	C[i]	1
k+7	c[i+1]に加算結果を書き込み。	a[i+1]+b[i+1]	b[i+1]	C[i+1]	1

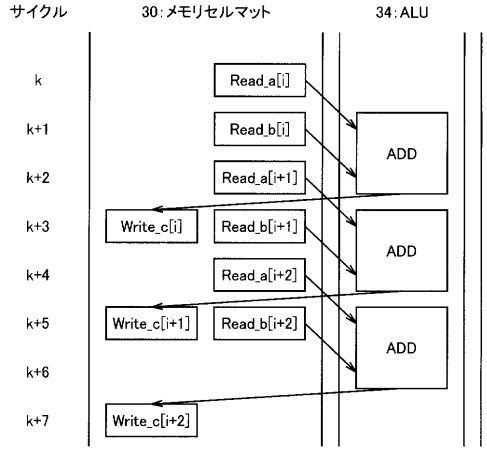
【図９】



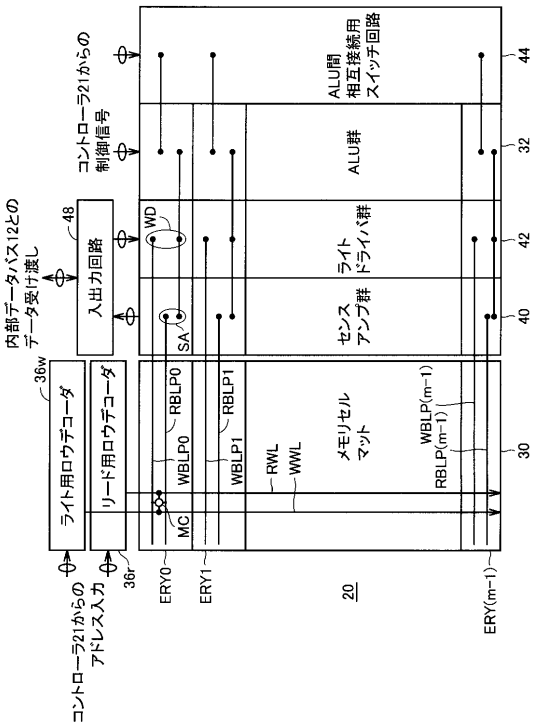
【図１０】



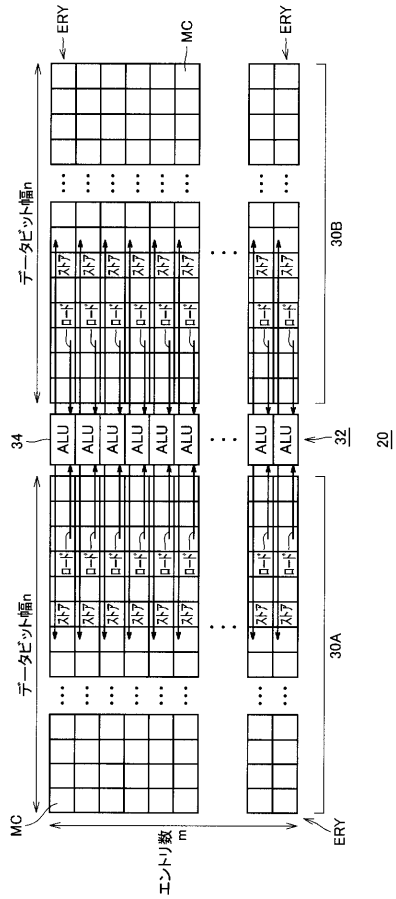
【図１１】



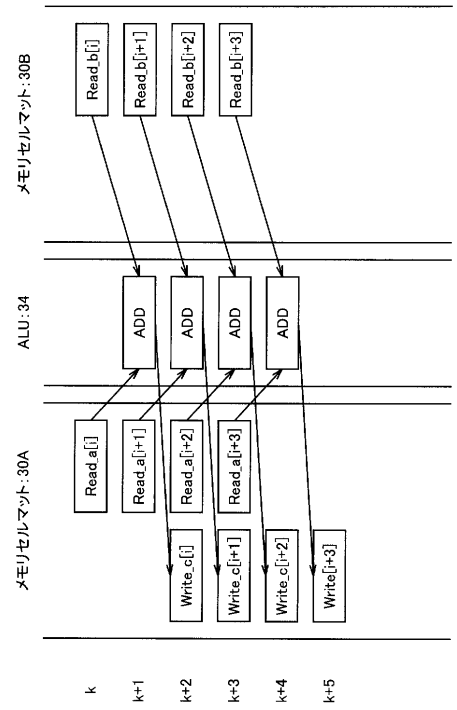
【図１２】



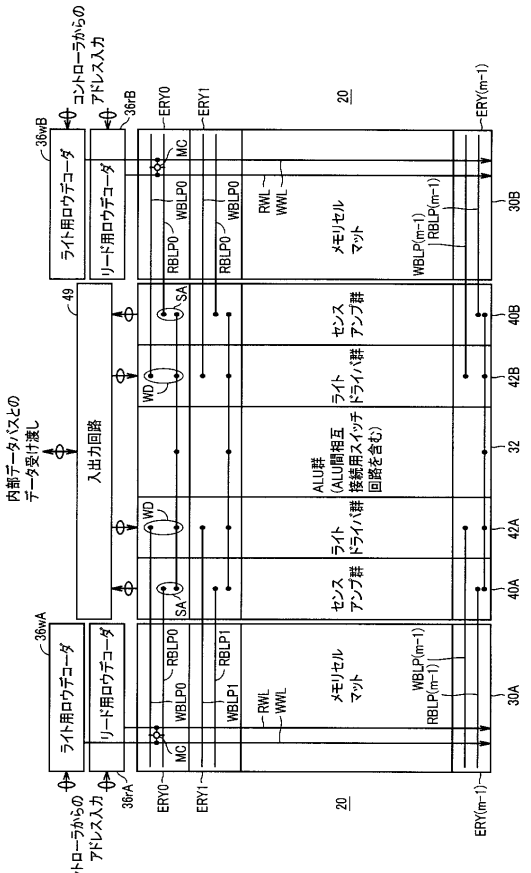
【図１３】



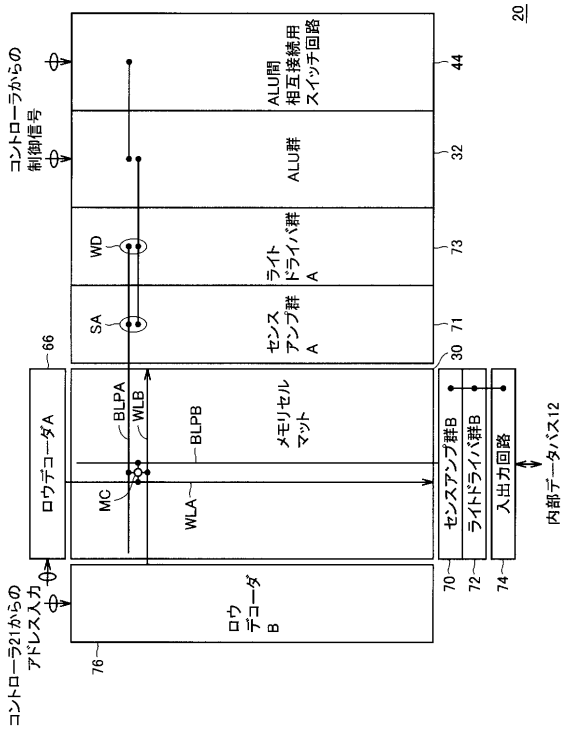
【図１４】



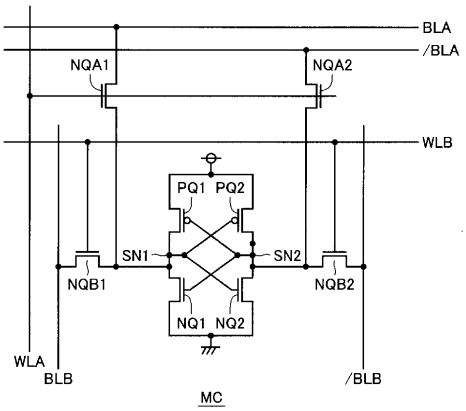
【図１５】



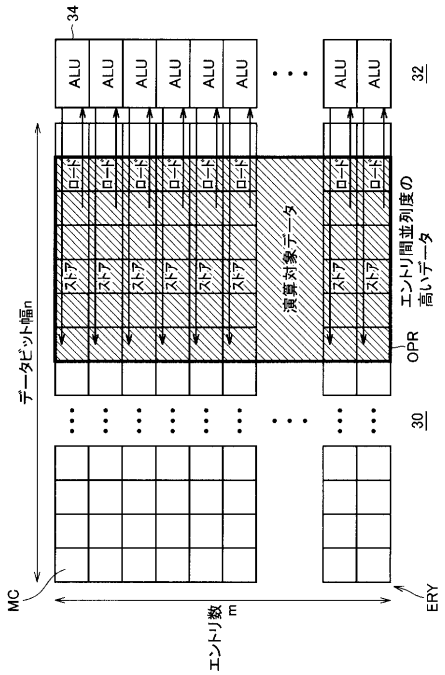
【図１６】



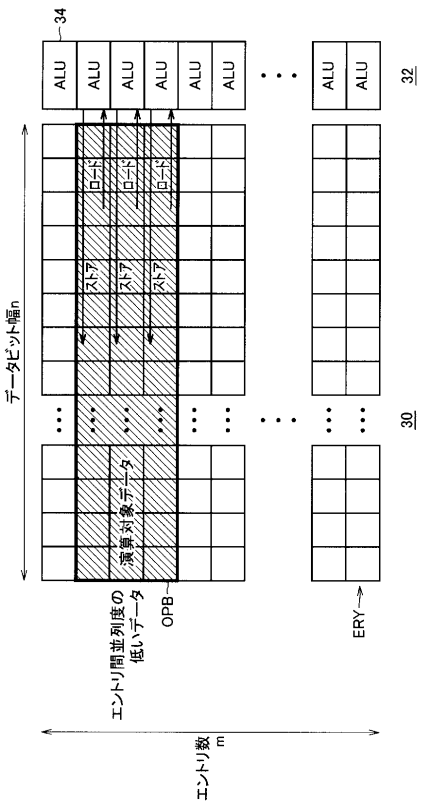
【図１７】



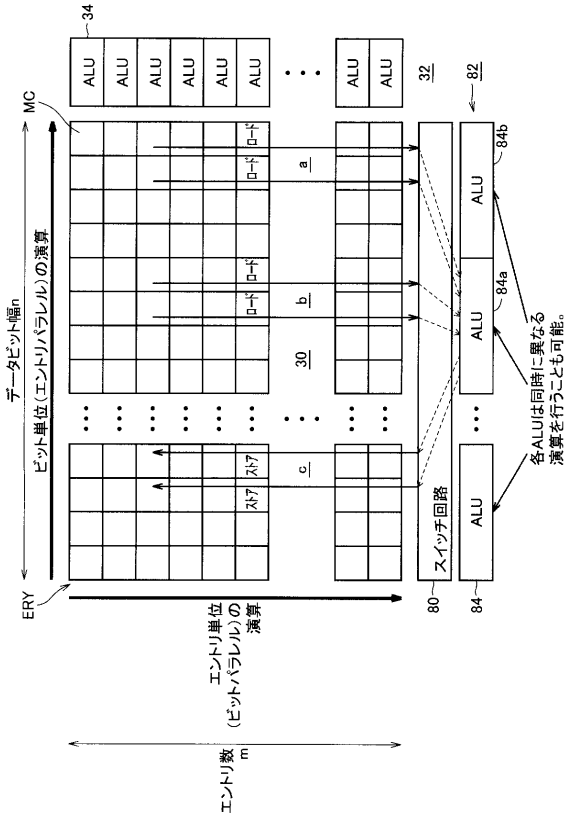
【図１８】



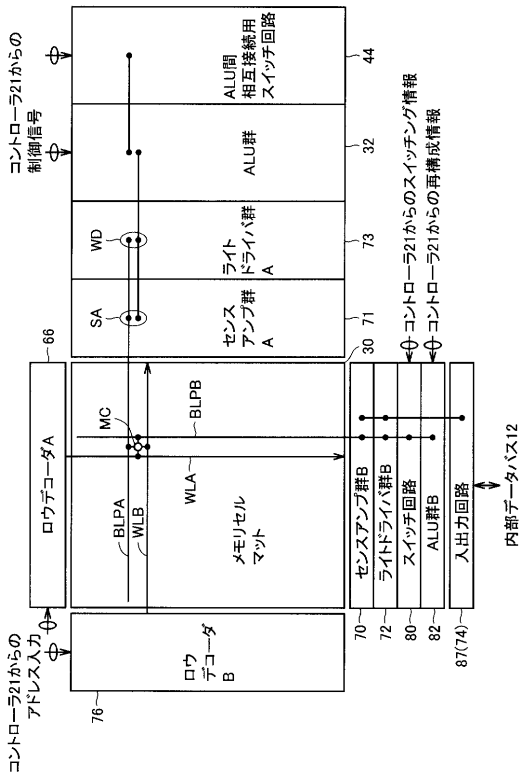
【図１９】



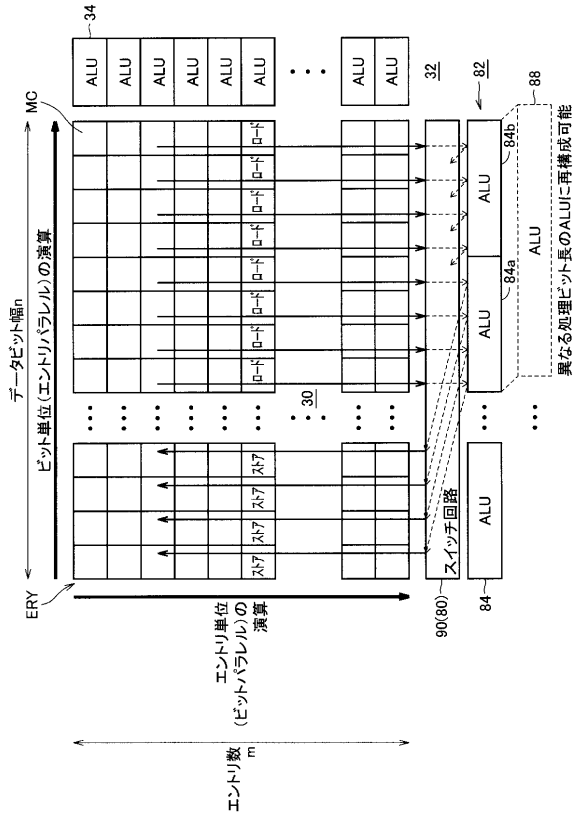
【図２０】



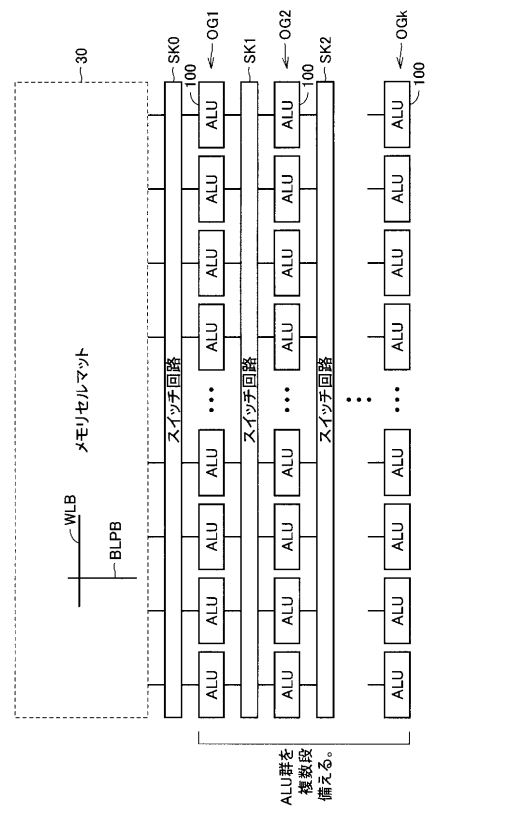
【図２１】



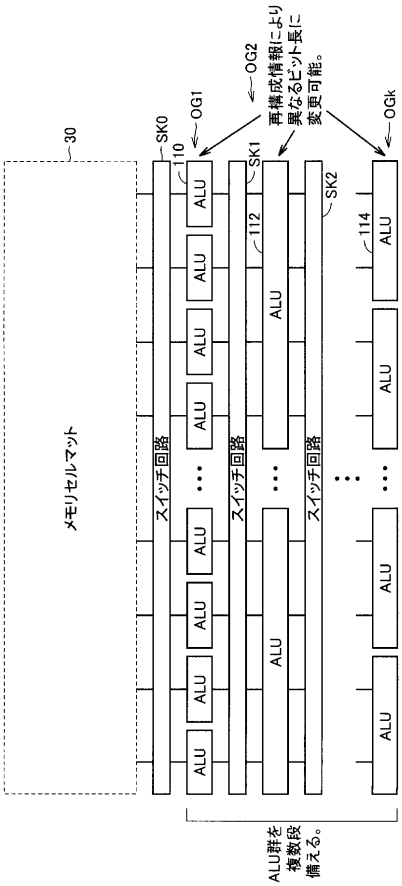
【図２２】



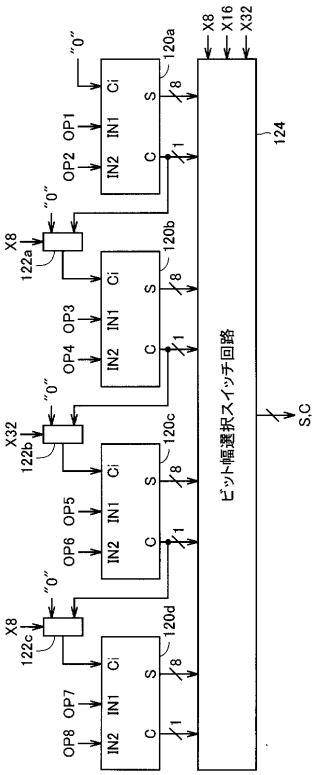
【図２３】



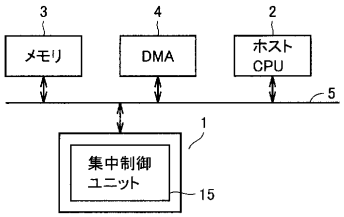
【図２４】



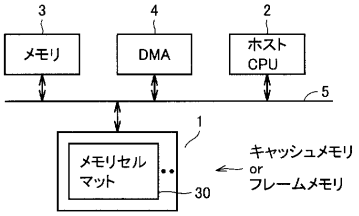
【図２５】



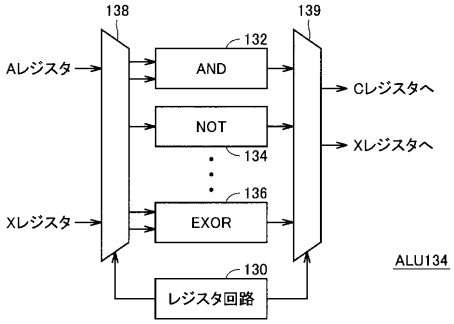
【図２６】



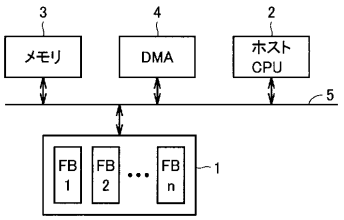
【図２７】



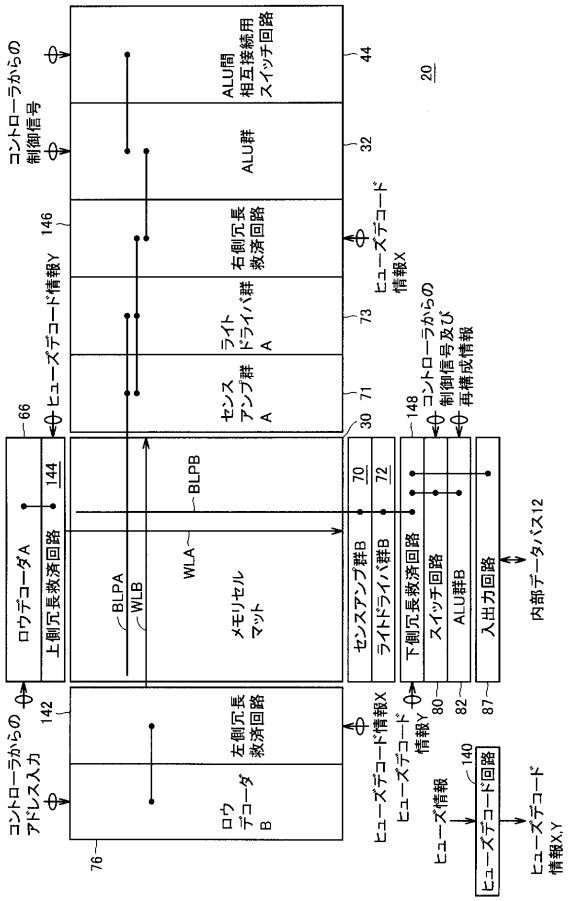
【図２８】



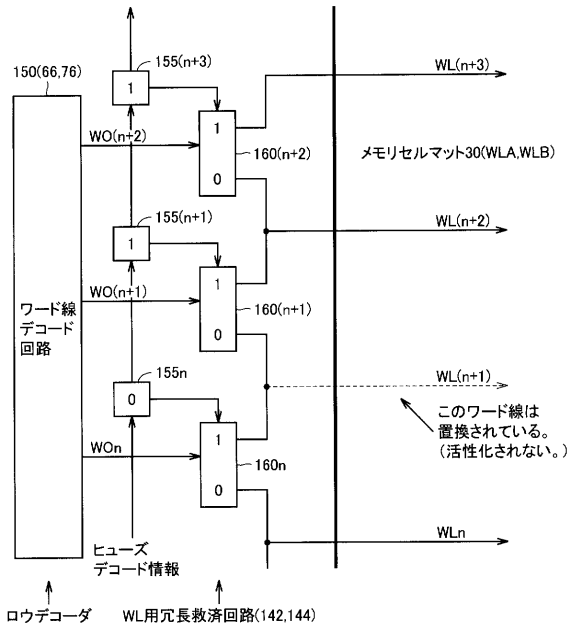
【図２９】



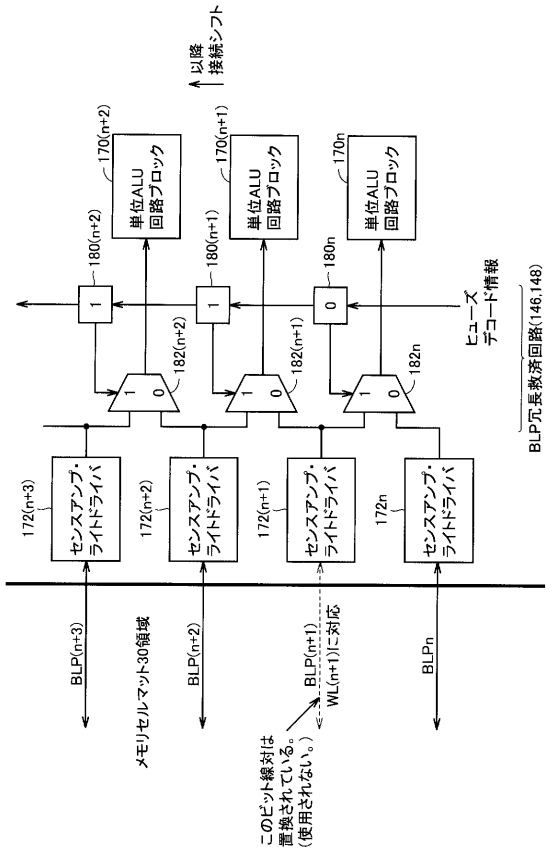
【図３０】



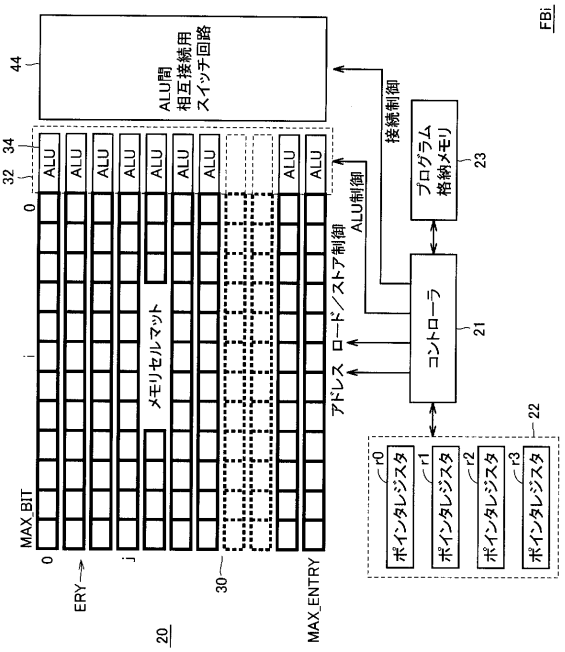
【図３１】



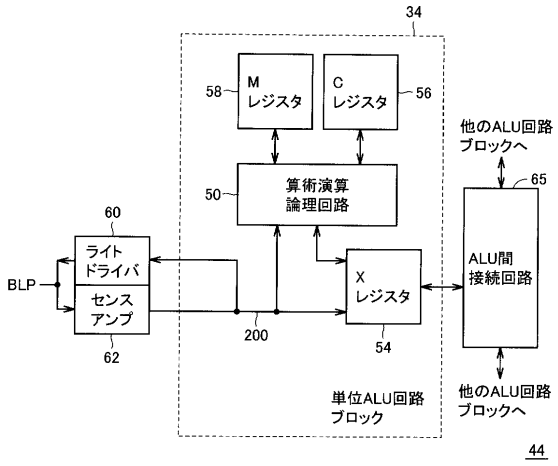
【図３２】



【図３３】



【図３４】



【図３５】

レジスタ命令

命令	動作
reg.set n,rx	rx ← #n (0＜n＜MAX_BIT,x=0...3) :任意の値をレジスタrxにセット。
reg.cpy rx,ry	ry ← rx (x,y=0...3) :2種のレジスタ間のコピー。
reg.inc rx	rx ← rx+1 (x=0...3) :rxの値をインクリメント。
reg.dec rx	rx ← rx−1 (x=0...3) :rxの値をデクリメント。
reg.sft rx	rx ← rx＜＜1 (x=0...3) :rxの値のビットシフト。

【図３６】

ALU命令（セット・クリア）

ALU命令	動作
alu.set.X alu.set.C alu.set.M	Xj ← 1 (j=0...MAX_ENTRY) Cj ← 1 (j=0...MAX_ENTRY) Mj ← 1 (j=0...MAX_ENTRY)
	:エントリjのレジスタに1をセット。
alu.clr.X	Xj ← 0 (j=0...MAX_ENTRY)
alu.clr.C	Cj ← 0 (j=0...MAX_ENTRY)
alu.clr.M	Mj ← 0 (j=0...MAX_ENTRY)
	:エントリjのレジスタを0にクリア。
alu.cpy.XM	Mj ← Xj (j=0...MAX_ENTRY)
alu.cpy.CX	Xj ← Cj (j=0...MAX_ENTRY)
alu.cpy.CM	Mj ← Cj (j=0...MAX_ENTRY)
	:レジスタ間のデータのMove/Copy

【図３７】

ALU命令（ロード・ストア）

ALU命令	動作
mem.ld@rx	Xj ← Aj(rx) (j=0...MAX_ENTRY,x=0...3) :ビットポインタrxの示すメモリ位置からロード
mem.st@rx	Aj(rx) ← Xj if Mj (j=0...MAX_ENTRY,x=0...3) :ビットポインタrxの示すメモリ位置へストア マスクレジスタが0の場合は実行されない。

【図３８】

ALU命令（エントリ間データ移動）

ALU命令	動作
ecm.mv.n#n	Xj ← Xj+n (0＜n＜128) :移動量を数値で指定したmove :エントリj+nのXレジスタの値が エントリjのXレジスタに移動。
ecm.mv.r rx	Xj ← Xj+rx (0＜x＜3) :レジスタrxに格納されている 値に応じて移動

【図３９】

ALU命令（算術演算）

ALU命令	動作
alu.op.adc@rx	Xj=Aj[rx]^Xj^Cj, Cj=Aj[rx]&Xj+Aj[rx]&Cj+Xj&Ci if Mj (j=0...MAX_ENTRY) :加算命令 ポインタレジスタrxに 指されているメモリアドレスの データとXレジスタの値を加算し、 結果をXレジスタに格納する。 :Xレジスタの加算後の値が、Cレジスタにはキャリーが格納される。
alu.op.sbb@rx	Xj=Aj[rx]^!Xj^Cj, Cj=Aj[rx]&!Xj+Aj[rx]&Cj+!Xj&Ci if Mj (j=0...MAX_ENTRY) :減算命令 ポインタレジスタrxに 指されているメモリアドレスの データとXレジスタの値を減算し、 結果をXレジスタに格納する。 :Xレジスタの減算後の値が、Cレジスタにはボローが格納される。

【図４０】

ALU命令（論理演算）

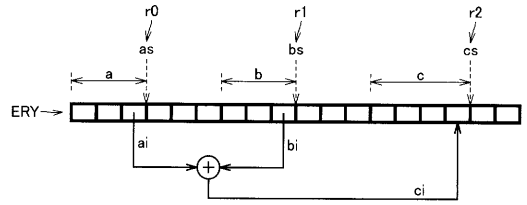
ALU命令	動作
alu.op.and@rx AND命令 （論理積命令）	Xj=Aj[rx]&Xj if Mj (j=0...MAX_ENTRY) :ポインタレジスタrxに指されているメモリアドレスのデータとXレジスタの値をANDし、 結果をXレジスタに格納する。マスクレジスタの値が0の場合は実行されない。
alu.op.or@rx OR命令 （論理和命令）	Xj=Aj[rx]|Xj if Mj (j=0...MAX_ENTRY) :ポインタレジスタrxに指されているメモリアドレスのデータとXレジスタの値をORし、 結果をXレジスタに格納する。マスクレジスタの値が0の場合は実行されない。
alu.op.eq@rx EX-OR命令 （排他的 論理和命令）	Xj=Aj[rx]^Xj if Mj (j=0...MAX_ENTRY) :ポインタレジスタrxに指されているメモリアドレスのデータとXレジスタの値をEX-ORし、 結果をXレジスタに格納する。マスクレジスタの値が0の場合は実行されない。
alu.not NOT命令 （反転命令）	Xj=!Xj if Mj (j=0...MAX_ENTRY) :Xレジスタの値を反転し、結果をXレジスタに格納する。 マスクレジスタの値が0の場合は実行されない。

【図４１】

```
行番号      命令                    コメント
  0 :  alu.set.M;  alu.clr.C;              //set mask & clear carry
  1 :  reg.set#as,r0;reg.set#bs,r1;reg.set#cs,r2;   //set address registor
  2 :  for(i=0;i<bit_count;i++)  {     //bit loop
         mem.ld@r0+;alu.op.adc@r1+; mem.st@r2+;
  3 :  }
  4 :  alu.cpy.CX; mem.st@r2;          //copy back carry
```

【図４２】

a+b=c:



【図４３】

a−b=c:

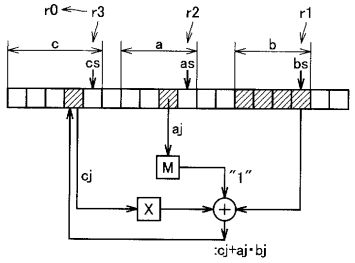
```
  0 :  alu.set.M;  alu.clr.C;              //set mask & clear carry
  1 :  reg.set#as,r0;reg.set#bs,r1;reg.set#cs,r2;   //set address registor
  2 :  for(i=0;i<bit_count;i++)  {     //bit loop
         mem.ld@r0+;alu.op.sbb@r1+; mem.st@r2+;
  3 :  }
  4 :  alu.cpy.CX; mem.st@r2;          //copy back carry
```

【図４４】

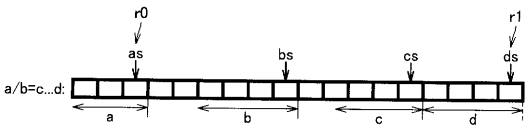
a･b=c

```
  0 :  reg.set#as,r2;reg.set#cs,r3;
  1 :  for(j=0;j<a_bit_count;j++) {        //multiplicand loop
  2 :   mem.ld@r2+; alu.cpy.XM;            //get multiplicand bit
  3 :   reg.cpy r3,r0;reg.set#bs,r1;       //set load/store address
  4 :   alu.clr.C;
  5 :   for(i=0;i<b_bit_count;i++) {       //multiplying loop
  6 :     mem.ld@r0,alu.op.adc@r1+; mem.st@r0+;
  7 :  }
  8 :   alu.cpy.CX; mem.st@r0;     //copy back carry bit
  9 :   reg.inc r3;
 10 :  }
```

【図４５】



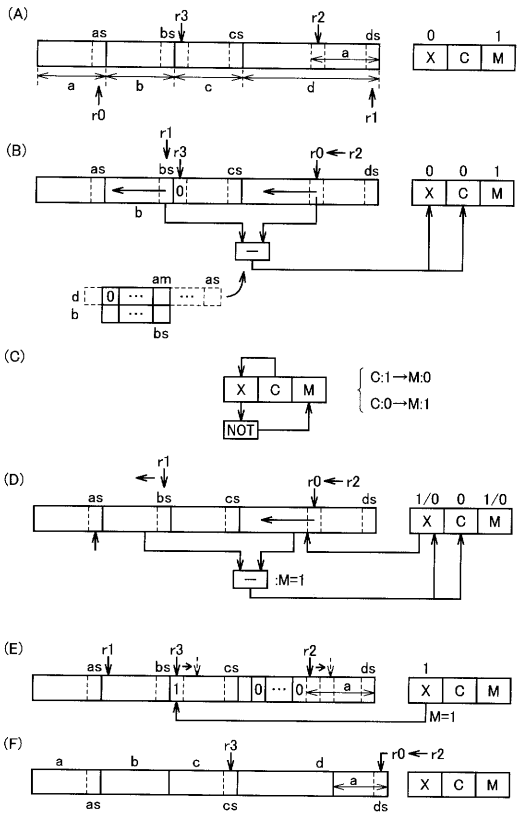
【図４６】



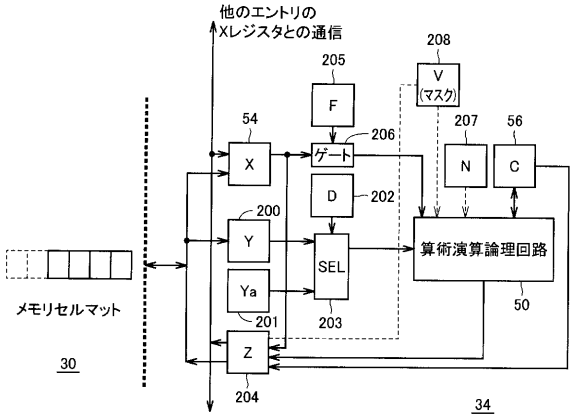
【図４７】

```
  0 :  alu.set.M;reg.set#as,r0;reg.set#ds,r1;
  1 :  for(i=0;i<bit_count;i++) {  mem.ld@r0+;mem.st@r1+;}
  2 :  reg.set#(cs+bit_count-1),r3;reg.set#(ds+bit_count-1),r2;
  3 :  for(j=0;j<bit_count;j++) {
  4 :   alu.set.M;alu.clr.X;mem.st@r3;     //clear result
  5 :   reg.cpy r2,r0;reg.set#bs,r1;alu.clr.C;
  6 :   for(i=0;i<bit_count;i++) {          //compare loop
  7 :     mem.ld@r0+;alu.op.sbb@r1+;
  8 :   }
  9 :   alu.cpy.CX;alu.op.not;alu.cpy.XM;
 10 :   reg.cpy r2,r0;reg.set#bs,r1;alu.clr.C;
 11 :   for(i=0;i<bit_count;i++) {     //substruction loop
 12 :     mem.ld@r0;alu.op.sbb@r1+; mem.st@r0+;
 13 :  }
 14 :  reg.dec r2;
 15 :  alu.set.X;mem.st@r3-              //store result
 16 :  }
```

【図４８】



【図４９】

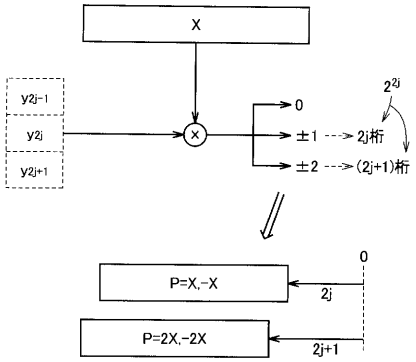


【図５０】

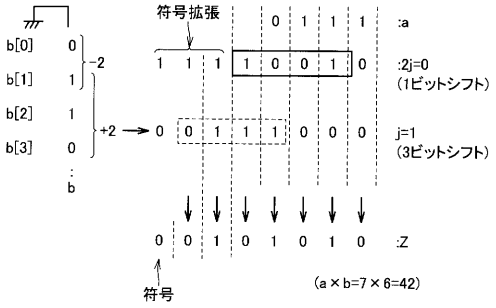
:X・(y2j−1+y2j−2・y2j+1)・2^2j:

y2j+1	y2j	y2j−1	部分積生成の手順
0	0	0	シフトアップ不要
0	0	1	2jビット分だけシフトアップ
0	1	0	2jビット分だけシフトアップ
0	1	1	(2j+1)ビット分だけシフトアップ
1	0	0	(2j+1)ビット分だけシフトアップ 結果の2の補数を求める。
1	0	1	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	0	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	1	シフトアップ不要

【図５１】



【図５２】



【図５３】

レジスタ命令

命令	操作
reg.set n,rx	rx ← #n (0<n<MAX_BIT,x=0...3) :任意の値をレジスタrxにセット。
reg.cpy rx,ry	ry ← rx (x,y=0...3) :2種のレジスタ間のコピー。
reg.inc rx	rx ← rx+1 (x=0...3) :rxの値をインクリメント。
reg.inc2 rx	rx ← rx+2 (x=0...3) :rxの値を2インクリメント。
reg.dec rx	rx ← rx−1 (x=0...3) :rxの値をデクリメント。
reg.sft rx	rx ← rx<<1 (x=0...3) :rxの値のビットシフト。

【図５４】

ALU命令(セット・クリア)

命令	操作
alu.set.X alu.set.V alu.set.N	Xj ← 1 (j=0...MAX_ENTRY) Vj ← 1 (j=0...MAX_ENTRY) Nj ← 1 (j=0...MAX_ENTRY) エントリjのレジスタに1をセット。
alu.clr.X alu.clr.C alu.clr.F	Xj ← 0 (j=0...MAX_ENTRY) Cj ← 0 (j=0...MAX_ENTRY) Fj ← 0 (j=0...MAX_ENTRY) エントリjのレジスタを0にクリア。

【図５５】

ALU命令(セット・クリア)

命令	操作
alu.cpy.XV	Vj ← Xj (j=0...MAX_ENTRY)
alu.cpy.XN	Nj ← Xj (j=0...MAX_ENTRY)
alu.cpy.XZ	Zj ← Xj (j=0...MAX_ENTRY)
alu.cpy.YZ	Zj ← Yj (j=0...MAX_ENTRY)
alu.cpy.CZ	Zj ← Cj (j=0...MAX_ENTRY)
alu.cpy.NZ	Zj ← Nj (j=0...MAX_ENTRY) :レジスタ間のデータのCopy

【図５６】

ALU命令(ロード・ストア)

命令	操作
mem.ld.X@rx mem.ld.Y@rx	Xj ← Aj[rx] (j=0...MAX_ENTRY,x=0...3) Yj ← Aj[rx] (j=0...MAX_ENTRY,x=0...3) :ビットポインタrxの示すメモリ位置からロード
mem.st@rx	Aj[rx] ← Zj if Vj (j=0...MAX_ENTRY,x=0...3) :ビットポインタrxの示すメモリ位置へストア マスクレジスタが0の場合は実行されない。

【図５７】

ALU命令(エントリ間データ移動)

命令	操作
ecm.mov.n#n	Xj ← Zj+n (0<=n<MAX_ENTRY) :エントリj+nのZレジスタの値がエントリjのXレジスタに移動。
ecm.mov.r rn	Xj ← Zj+rn (0<=n<4) :レジスタrxの格納値rn離れたエントリj+rn のZレジスタの内容をエントリjのXレジスタに移動。

【図５８】

ALU命令（算術演算）

命令	操作	
alu.op.adc	Zj=Xj^Yj^Cj Cj=Yj＆Xj+Yj＆Cj+Xj＆Ci	if Nj＆Vj if Nj＆Vj (j=0...MAX_ENTRY)
	:加算命令 ポインタレジスタrxに 指されているメモリアドレスの データとXレジスタの値を加算し、 結果をXレジスタに格納する。 :Zレジスタの加算後の値が、Cレジスタにはキャリー が格納される。	
alu.op.sbb	Zj=Xj^Yj^Cj Cj=Yj＆!Xj+Yj＆Cj+!Xj＆Ci	if Nj＆Vj if Nj＆Vj (j=0...MAX_ENTRY)
	:減算命令 ポインタレジスタrxに 指されているメモリアドレスの データとXレジスタの値を減算し、 結果をXレジスタに格納する。 :Zレジスタの減算後の値が、Cレジスタにはボロー が格納される。	

【図５９】

ALU命令（算術演算）

命令	操作	
alu.op.booth	Nj=(Yj＆(!Xj+!Fj))+(!Yj＆(Xj+Fj)) Dj=(Yj＆!Xj＆!Fj)+(!Yj＆Xj＆Fj) Fj=Yj	if Vj if Vj if Vj (j=0...MAX_ENTRY)
	:ブース命令 X,Y,Fのレジスタ値から N,D,F（次の演算用）を決める。	
alu.op.exe	Zj=Xj^(Dj?Yaj:Yj)^Cj Cj=(Dj?Yaj:Yj)＆(Fj?!Xj:Xj)+(Dj?Yaj:Yj)＆Cj +(Fj?!Xj:Xj)＆Cj Yaj=Yj;	if Nj＆Vj if Nj＆Vj (j=0...MAX_ENTRY)
	:EXE命令 Dレジスタ、Fレジスタの値によって 条件分岐する演算命令 Dレジスタが1の場合はYaレジスタ側を参照する。 （1ビットシフトするかどうかの分岐） Fレジスタが0の場合、加算命令、1の場合、減算命令となる。	

【図６０】

$Y_{(y2j+1)}$	$X_{(y2j)}$	$F_{(y2j-1)}$	D	N	部分積生成の手順
0	0	0	0	0	シフトアップ不要
0	0	1	0	1	2jビット分だけシフトアップ
0	1	0	0	1	2jビット分だけシフトアップ
0	1	1	1	1	(2j+1)ビット分だけシフトアップ
1	0	0	1	1	(2j+1)ビット分だけシフトアップ 結果の2の補数を求める。
1	0	1	0	1	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	0	0	1	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	1	0	0	シフトアップ不要

【図６１】

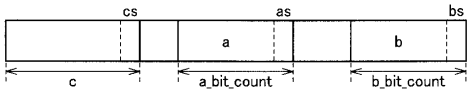
ALU命令（論理演算）

命令	操作	
alu.op.and	Zj = Yj＆Xj if Vj	(j=0...MAX_ENTRY)
alu.op.or	Zj = Yj|Xj if Vj	(j=0...MAX_ENTRY)
alu.op.exor	Zj = Yj^Xj if Vj	(j=0...MAX_ENTRY)
alu.op.not	Zj = !Xj if Vj	(j=0...MAX_ENTRY)
alu.op.LT	Nj = (Cj?0:1);	(j=0...MAX_ENTRY)

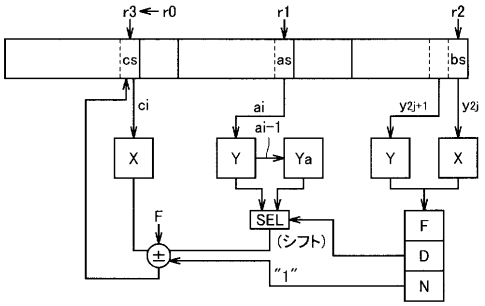
【図６２】

```
//a･b=c（符号付）

 0 :  alu.set.V;
 1 :  reg.set#bs,r2;reg.set#cs,r3;alu.clr.F;
 2 :  for(j=0;j＜a_bit_count;j+=2) {          //multiplicand loop
 3 :   mem.ld.X@r2+; mem.ld.Y@r2+;       //get multiplicand bit
 4 :   alu.op.booth;
 5 :   reg.cpy r3,r0;reg.set#as,r1;              //set load/store address
 6 :   for(i=0;i＜b_bit_count;i++) {          //multiplyer loop
 7 :     mem.ld.X@r0; mem.ld.Y@r1+;alu.op.exe; mem.st@r0+;
 8 :   }
 9 :   for(i=0;i＜b_bit_count-j;i++) {        //sign extention
10 :     mem.ld.X@r0;alu.op.exe; mem.st@r0+;
11 :   }
12 :   reg.inc2 r3;
13 :  }
```

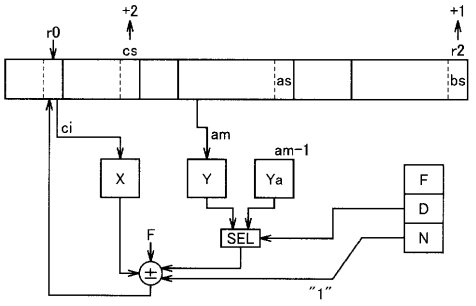
【図６３】



【図６４】



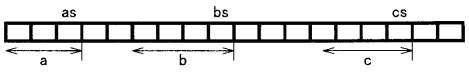
【図６５】



【図６６】

(A)

a+b=c



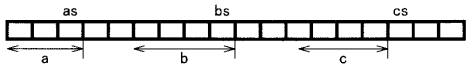
(B)

```
alu.set.V; alu.clr.C;            //set mask & clear carry
reg.set#as,r0;reg.set#bs,r1;reg.set#cs,r2;  //set address registor
for(i=0;i<bit_count;i++) {    //bit loop
   mem.ld.X@r0+; mem.ld.Y@r1+;alu.op.adc; mem.st@r2+;
}
```

【図６７】

(A)

a-b=c



(B)

```
alu.set.V; alu.clr.C;            //set mask & clear carry
reg.set#as,r0;reg.set#bs,r1;reg.set#cs,r2;  //set address registor
for(i=0;i<bit_count;i++) {    //bit loop
   mem.ld.X@r0+; mem.ld.Y@r1+;alu.op.sbb; mem.st@r2+;
}
```

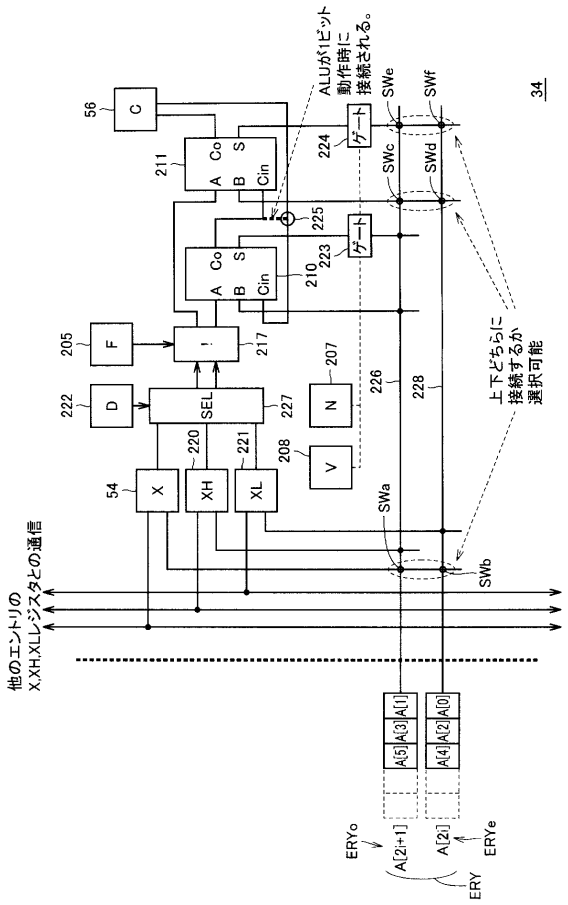
【図６８】

```
//a*b=c(符号なし)

 alu.set.V;
 reg.set#as,r2;reg.set#cs,r3;
 for(j=0;j<a_bit_count;j++) {          //multiplicand loop
  mem.ld.X@r2+; alu.cpy,XN;            //get multiplicand bit
  reg.cpy r3;r0;reg.set#bs,r1;         //set load/store address
  alu.clr.C;
  for(i=0;i<b_bit_count;i++) {         //multiplying loop
    mem.ld.X@r0; mem.ld.Y@r1+;alu.op.adc; mem.st@r0+;
  }
  alu.cpy.CZ; mem.st@r0;
  reg.inc r3;
 }
```

【図６９】

```
//a/b=c...d

 alu.set.V;alu.set.N;reg.set#as,r0;reg.set#ds,r1;
 for(i=0;i<bit_count;i++) { mem.ld.X@r0+;alu.cpy,XZ; mem.st@r1+;}
 reg.set#(cs+bit_count-1),r3;reg.set#(ds+bit_count-1),r2;
 for(j=0;j<bit_count;j++) {
  alu.clr.Z; mem.st@r3;                //clear result
  reg.cpy r2,r0;reg.set#bs,r1;alu.clr.C;alu.set.N;
  for(i=0;i<bit_count;i++) {           //compare loop
    mem.ld.X@r0+; mem.ld.Y@r1+;alu.op.sbb;
  }
  alu.op.LT;alu.clr.C;
  reg.cpy r2,r0;reg.set#bs,r1;
  for(i=0;i<bit_count;i++) {           //substruction loop
    mem.ld.X@r0, mem.ld.Y@r1+;alu.op.sbb; mem.st@r0+;
  }
  reg.dec r2;
  alu.cpy.NZ; mem.st@r3-               //store result
 }
```

【図７０】



【図７１】

ポインタレジスタ命令

命令	操作
ptr.set n,px	px ← #n (0<n<MAX_BIT,x=0...3) :任意の値をレジスタpxにセット。
ptr.cpy px,py	py ← px (x,y=0...3) :2種のレジスタ間のコピー。
ptr.inc px	px ← px+1 (x=0...3) :pxの値をインクリメント。
ptr.inc2 px	px ← px+2 (x=0...3) :pxの値を2インクリメント。
ptr.dec px	px ← px-1 (x=0...3) :pxの値をデクリメント。
ptr.dec2 px	px ← px-2 (x=0...3) :pxの値の2ビットシフト。
ptr.sft px	px ← px<<1 (x=0...3) :pxの値の1ビットシフト。

【図７２】

ALU命令1ビット動作時(ロード・ストア)

命令	操作	
mem.ld.X@px	Xj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
mem.ld.N@px	Nj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
mem.ld.V@px	Vj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
mem.ld.F@px	Fj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
mem.ld.D@px	Dj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
mem.ld.C@px	Cj ← Aj[px]	(j=0...MAX_ENTRY,x=0...3)
	:PXの示すメモリセル位置からロード	
mem.st.X@px	Aj[px] ← Xj if Vj	(j=0...MAX_ENTRY,x=0...3)
mem.st.N@px	Aj[px] ← Nj if Vj	(j=0...MAX_ENTRY,x=0...3)
mem.st.V@px	Aj[px] ← Vj if Vj	(j=0...MAX_ENTRY,x=0...3)
mem.st.F@px	Aj[px] ← Fj if Vj	(j=0...MAX_ENTRY,x=0...3)
mem.st.D@px	Aj[px] ← Dj if Vj	(j=0...MAX_ENTRY,x=0...3)
mem.st.C@px	Aj[px] ← Cj if Vj	(j=0...MAX_ENTRY,x=0...3)
	:pxの示すメモリセル位置へストア マスクレジスタ(V)のクリア時実行されない	
mem.swp.X@px	Aj[px] ⇔ Xj if Vj &Nj :pxの示すメモリセル位置のデータとXレジスタデータ入替	(j=0...MAX_ENTRY,x=0...3)

【図７３】

ALU命令2ビット動作時（ロード・ストア）

命令	操作
mem2.ld.X@px	XLj ← Aj[px] (j=0...MAX_ENTRY,x=0..3) XHj ← Aj[px+1] (j=0...MAX_ENTRY,x=0..3) :ビットポインタpxの示すメモリ位置からロード。 メモリ上の連続する2ビットは同時ロードされ、XH/XLレジスタに格納される。
mem2.st.X@px	Aj[px] ← XLj if Vj (j=0...MAX_ENTRY,x=0..3) Aj[px+1] ← XHj if Vj (j=0...MAX_ENTRY,x=0..3) :ビットポインタpxの示すメモリ位置へストア。 メモリ上の連続する2ビットは同時にXH/XLレジスタからストアされる。 マスクレジスタ(V)が0の場合は実行されない。
mem2.swp.X@px	Aj[px] ⇔ XLj if Vj &Nj (j=0...MAX_ENTRY,x=0..3) Aj[px+1] ⇔ XHj if Vj &Nj (j=0...MAX_ENTRY,x=0..3) :ビットポインタpxの示すメモリ位置とXH/XLレジスタのデータの入れ替え。 メモリ上の連続する2ビットは同時にXH/XLレジスタとデータが入れ替えられる。 マスクレジスタ(V)またはNレジスタが0の場合は実行されない。

【図７４】

ALU命令1ビット動作時（エントリ間データ移動）

命令	操作
ecm.mv.n#n	Xj ← Xj+n (0＜=n＜MAX_ENTRY) :エントリj+nのXレジスタの値がエントリjのXレジスタに移動。
ecm.mv.r rn	Xj ← Xj+rn (0＜=n＜4) :レジスタrnの格納値離れた エントリj+rnのXレジスタがエントリj のXレジスタに移動。
ecm.swp	Xj ⇔ Xj+1 :隣接エントリXレジスタの値を入れ替える。

【図７５】

ALU命令2ビット動作時（エントリ間データ移動）

命令	操作
ecm2.mv.n#n	XLj ← XLj+n (0＜=n＜MAX_ENTRY) XHj ← XHj+n (0＜=n＜MAX_ENTRY) :エントリj+nのXH/XLレジスタの値が エントリjのXH/XLレジスタに移動。
ecm2.mv.r rn	XLj ← XLj+rn (0＜=n＜4) XHj ← XHj+rn (0＜=n＜4) :エントリj+rnのXL/XHレジスタの 値がエントリjのXL/XHレジスタに移動。
ecm2.swp	XLj ⇔ XLj+1 XHj ⇔ XHj+1 :隣接エントリのXH/XLレジスタの値を入れ替える。

【図７６】

ALU命令1ビット動作時（算術演算命令）

命令	操作
alu.adc@px	z=Aj[px]^Xj^Cj Cj=Aj[px]&Xj+Xj&Ci+Cj&Aj[px] Aj[px]=z; if Nj&Vj (j=0...MAX_ENTRY) :加算命令 ポインタレジスタpxに 指されているメモリアドレスの データとXレジスタの値を加算し、 結果をメモリに返す。 メモリセルAjに加算後の値が、Cレジスタにはキャリー が格納される。
alu.sbc@px	z=Aj[px]^!Xj^Cj Cj=Aj[px]&!Xj+!Xj&Cj+Cj&Aj[px] Aj[px]=z; if Nj&Vj (j=0...MAX_ENTRY) :減算命令 ポインタレジスタpxに 指されているメモリアドレスの データとXレジスタの値を減算し、 結果をメモリに返す。 メモリセルAjに減算後の値が、Cレジスタにはボロー が格納される。
alu.inv@px	Aj[px]=!Aj[px];if Nj&Vj (j=0...MAX_ENTRY) :反転命令 ポインタレジスタpxに 指されているメモリアドレスの データを反転してメモリに返す。
alu.let f	f=(F*8+D*4+N*2+C) 与えられた数値に応じてF,D,N,Cのレジスタを設定する。

【図７７】

ALU命令2ビット動作時（算術演算命令）

命令	操作
alu2.booth	Nj=(XHj&(!XLj+!Fj))+(!XHj&(XLj+Fj)) if Vj Dj=(XHj&!XLj&!Fj)+(!XHj&XLj&Fj) if Vj Fj=Cj=XHj; if Vj (j=0...MAX_ENTRY) :ブース命令 XH,XL,Fのレジスタ値から N,D,F(次の演算用)を決める。

alu2.exe@px	x1=(Dj?XHj:XLj);x1=(F?!x1:x1);
	x2=(Dj?Xj:XHj);x2=(F?!x2:x2);
	zz=Aj[px]^x1^Cj
	cc=Aj[px]&x1+x1&Cj+Cj&Aj[px]
	Aj[px]=zz; if Nj&Vj
	zz=Aj[px+1]^x2^cc
	Cj=Aj[px+1]&x2+x2&cc+cc&Aj[px+1] if Nj&Vj
	Aj[px+1]=zz if Nj&Vj
	Xj=(Dj?XLj:Xj) (j=0...MAX_ENTRY)
	:EXE命令
	Dレジスタ、Fレジスタの値に
	よって条件分岐する演算命令

【図７８】

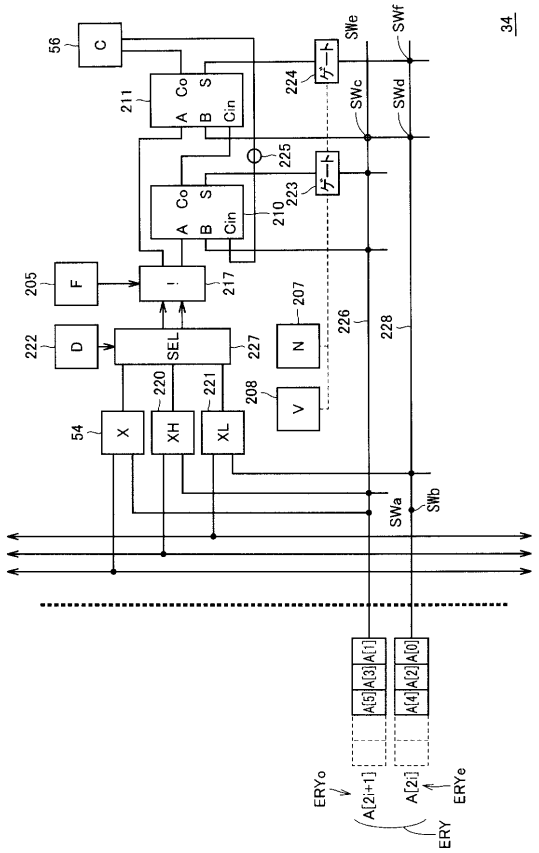
ブース命令実行時のレジスタ値

$XH_{(y2j+1)}$	$XL_{(y2j)}$	$F_{(y2j-1)}$	D	N	部分積生成の手順
0	0	0	0	0	シフトアップ不要
0	0	1	0	1	2jビット分だけシフトアップ
0	1	0	0	1	2jビット分だけシフトアップ
0	1	1	1	1	(2j+1)ビット分だけシフトアップ
1	0	0	1	1	(2j+1)ビット分だけシフトアップ 結果の2の補数を求める。
1	0	1	0	1	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	0	0	1	2jビット分だけシフトアップ 結果の2の補数を求める。
1	1	1	0	0	シフトアップ不要

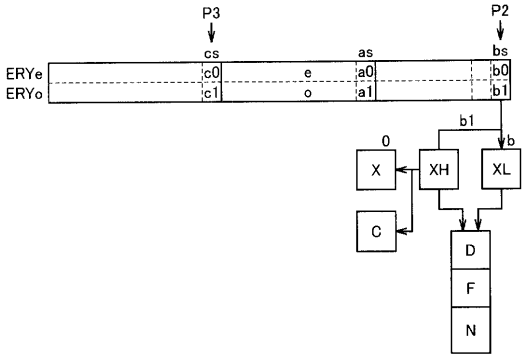
【図７９】

```
//a*b=c(符号付)

 0 :  ptr.set#bs,p2; ptr.set#cs,p3;
 1 :  for(i=0;i<bit_count;i+=2) {
 2 :          mem2.ld.X@p2++; alu2.booth;
 3 :          ptr.cpy p3,p0; ptr.set#as,p1;
 4 :          for(j=0;j<bit_count;j+=2) {
 5 :                  mem2.ld.X@p1++; alu2.exe@p0++;
 6 :          }
 7 :          for(j=0;j<bit_count-i;j+=2) {
 8 :                  alu2.exe@p0++;
 9 :          }
10 :          ptr.inc2 p3;
11 :  }
```

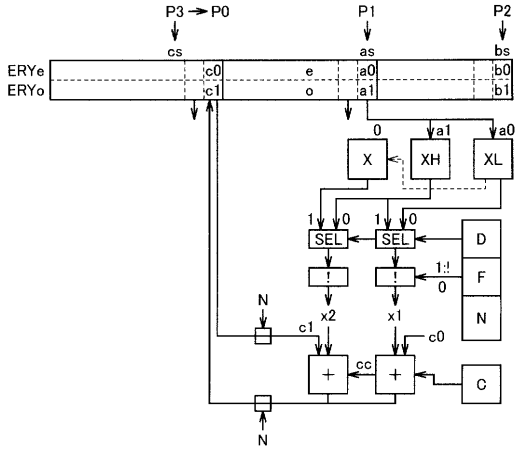
【図８０】



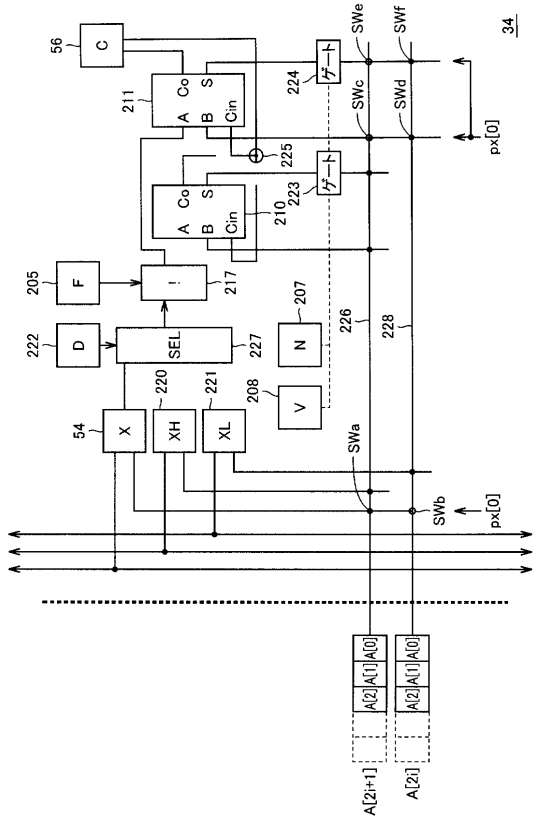
【図８１】



【図８２】



【図８３】



【図８４】

```
//:a+b=a

mem.ld.C 0;                    //clear carry
ptr.set#as,p0; ptr.set#bs,p1;  //set address registor
for(i=0;i<bit_count;i++) {     //bit loop
   mem.ld.X@p1+,alu.adc@p0+;
}
```

【図８５】

```
//a-b=a

mem.ld.C 1;                    //set carry
ptr.set#as,p0; ptr.set#b,p1;   //set address registor
for(i=0;i<bit_count;i++) {     //bit loop
   mem.ld.X@p1+,alu.sbc@p0+;
}
```

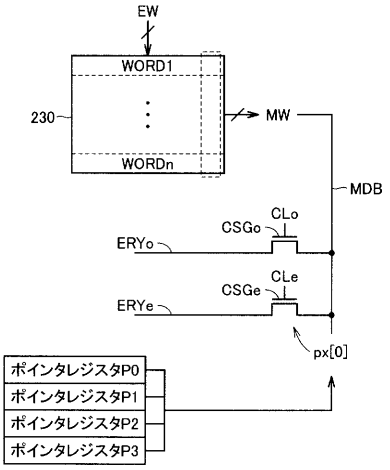
【図８６】

```
//a･b=c(符号なし)

ptr.set#bs,p2; ptr.set#cs,p3;
for(i=0;i<bit_count;i++) {
      mem.ld.N@p2+;
      ptr.cpy p3,p0; mem.ld.C 0;
      ptr.set#as,p1;
      for(j=0;j<bit_count;j++) {
            mem.ld.X@p1+; alu.adc@p0+;
      }
      mem.st.C@p0;
      ptr.inc p3;
}
```

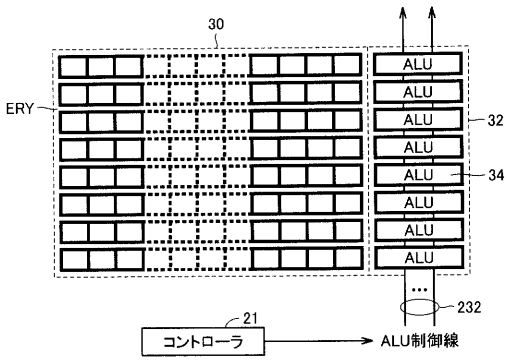
【図８７】

```
//a/b=c...d

ptr.set#as,p0; ptr.set#ds,p1;
for(i=0;i<n;i+=2) { mem2.ld.X@p0++; mem.st.X@p1++;}
ptr.set#(pc+bit_count-1),p3; ptr.set#(pd+bit_count-1),p2;
for(j=0;j<n;j++) {
      ptr.set#bs,p1; ptr.cpy p2,p0;alu.let(0b0011);
      for(i=0;i<n;i++) {
            mem.ld.X@p1+; alu.sbc@p0+;
      }
      ptr.cpy p2,p0; mem.st.C tp; alu.inv tp; mem.ld.N tp;
      ptr.set#bs,p1; mem.ld.C 0;
      for(i=0;i<n;i++) {
            mem.ld.X@p1+; alu.adc@p0+;
      }
      mem.st.N@p3;alu.inv@p3;
      ptr.dec p2; ptr.dec p3;
}
```

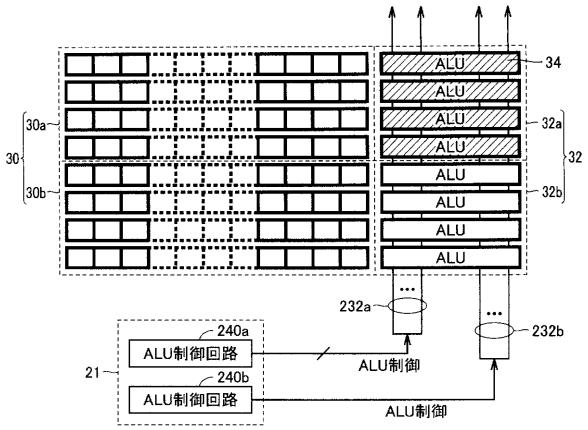
【図８８】



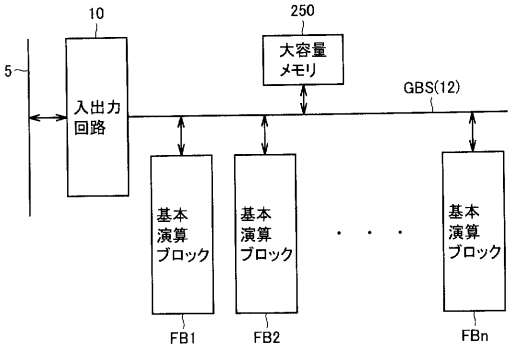
【図８９】



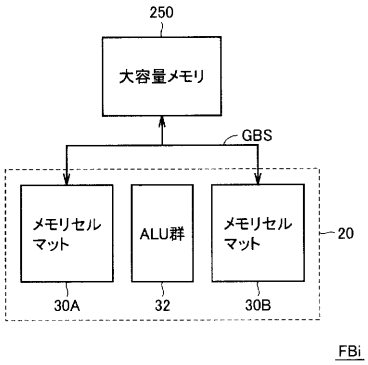
【図９０】



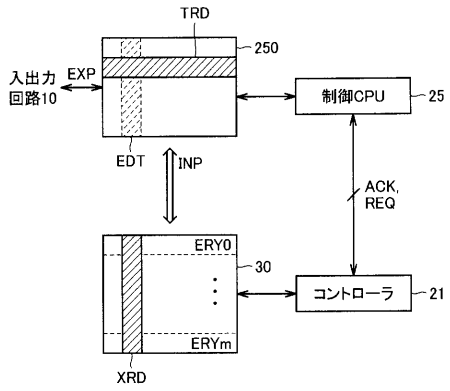
【図９１】



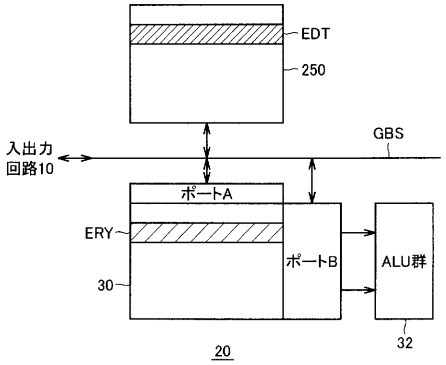
【図９２】



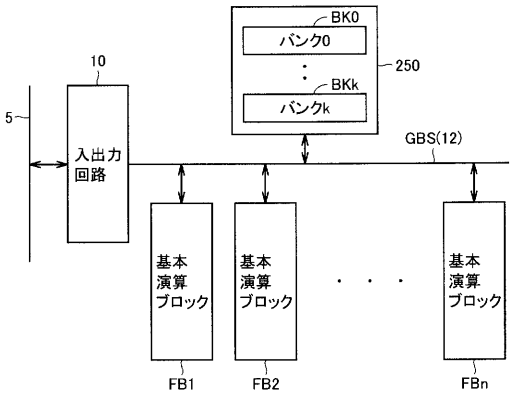
【図９３】



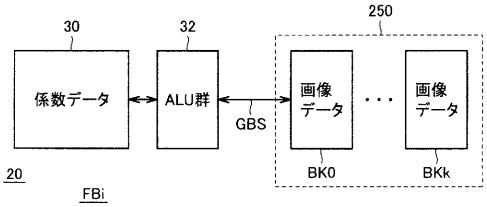
【図９４】



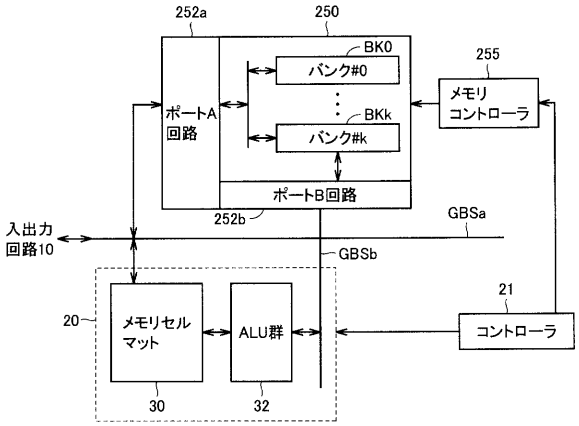
【図９５】



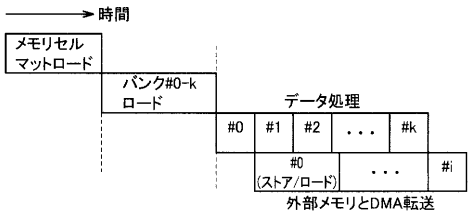
【図９６】



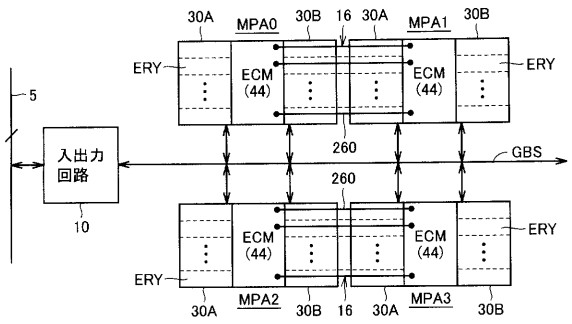
【図９７】



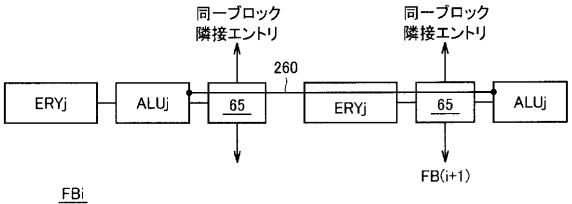
【図９８】



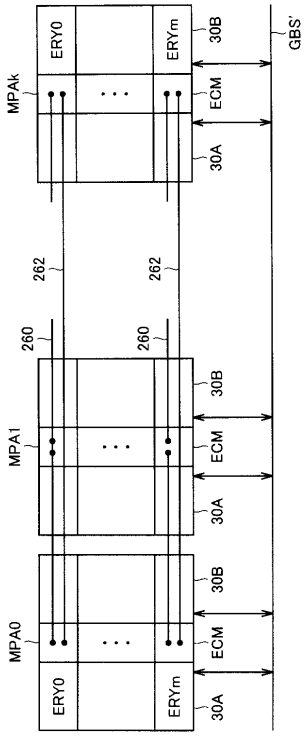
【図９９】



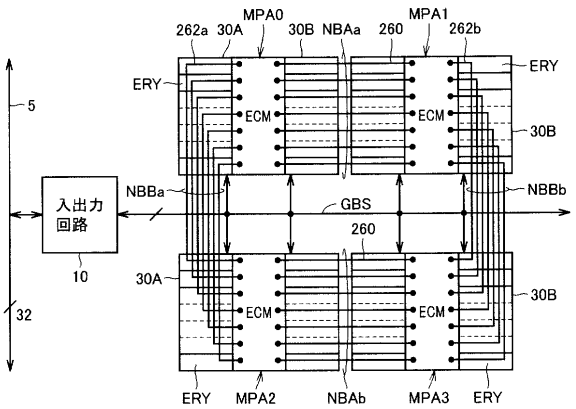
【図１００】



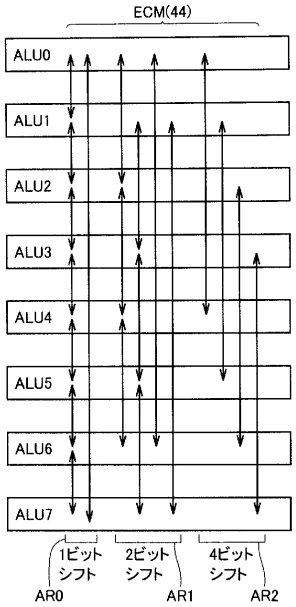
【図１０１】



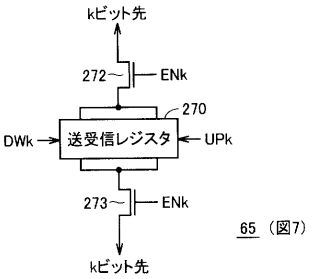
【図１０２】



【図１０３】

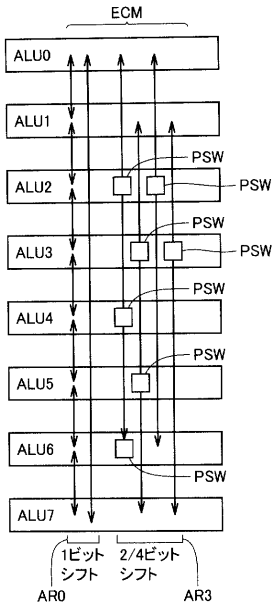


【図１０４】

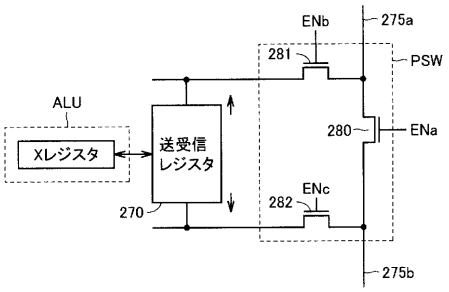


65 (図7)

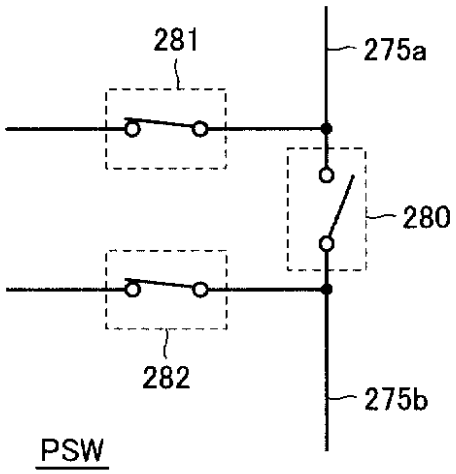
【図１０５】



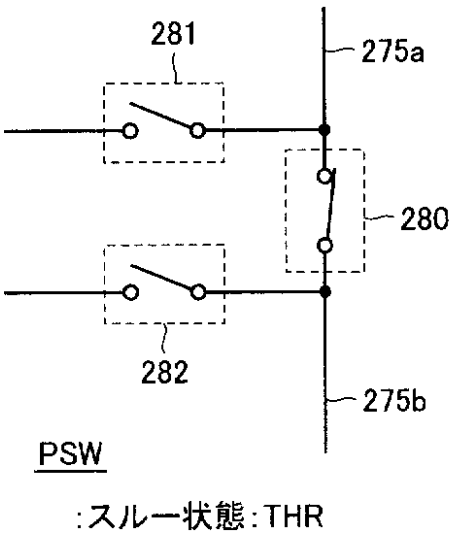
【図１０６】



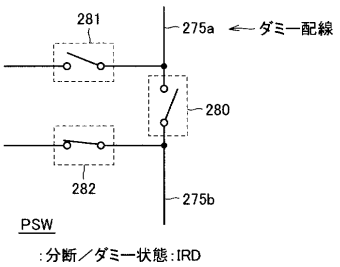
【図１０７】



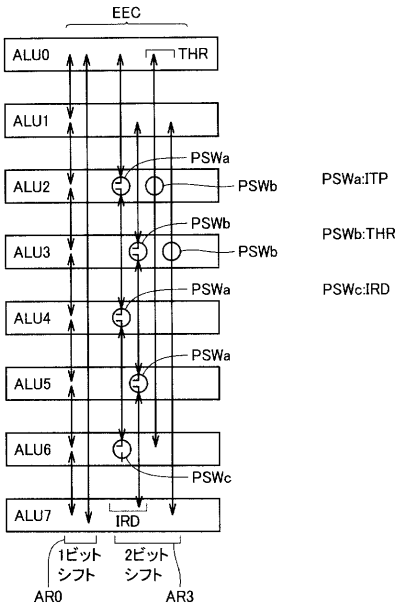
【図１０８】



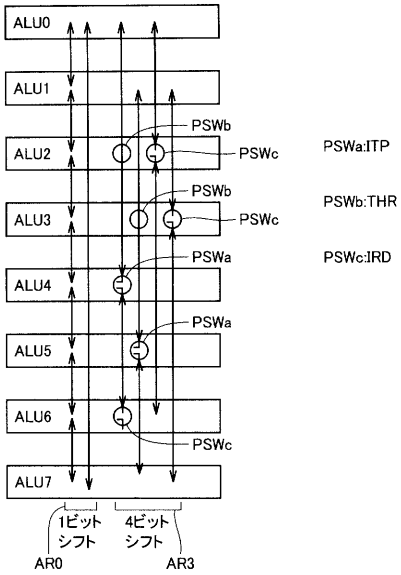
【図１０９】



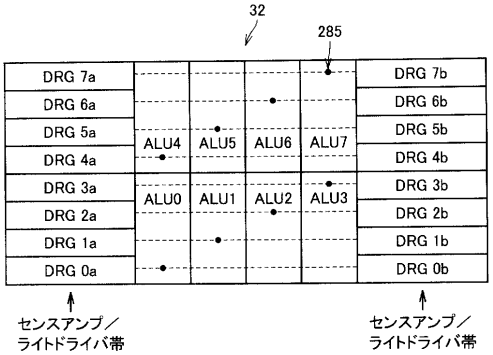
【図１１０】



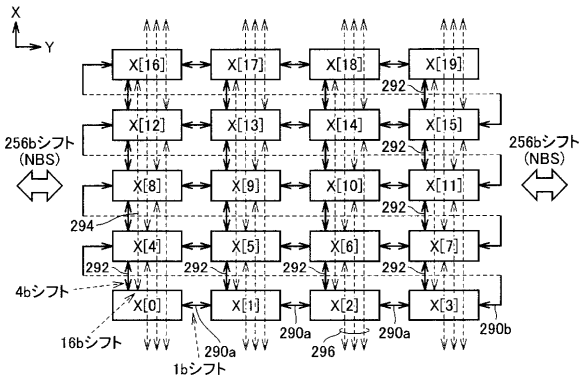
【図１１１】



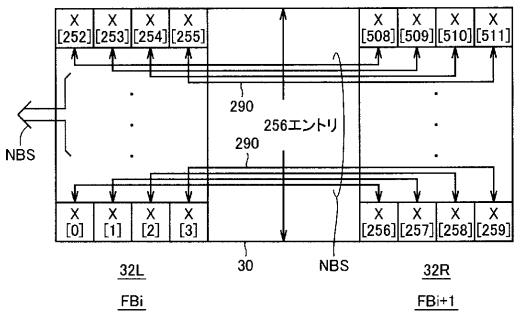
【図１１２】



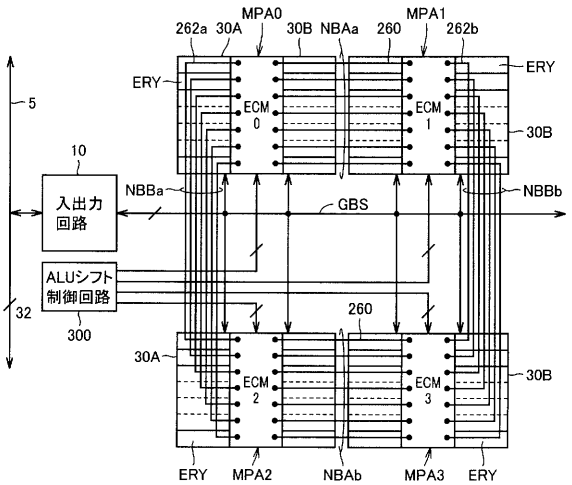
【図１１３】



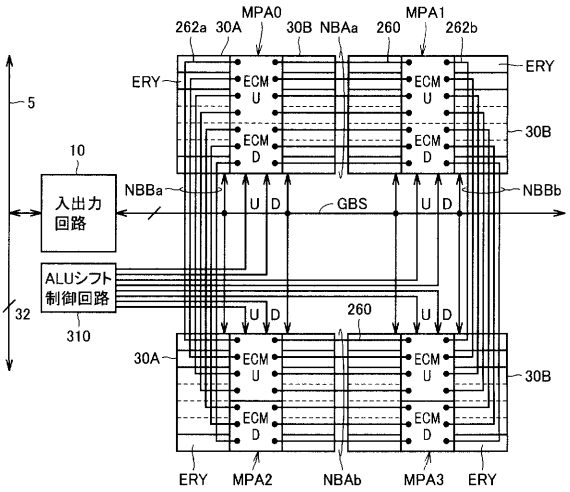
【図１１４】



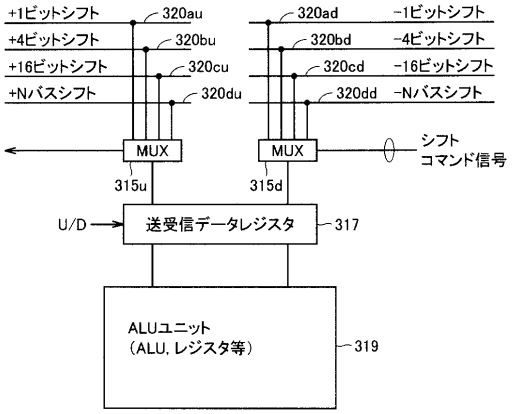
【図１１５】



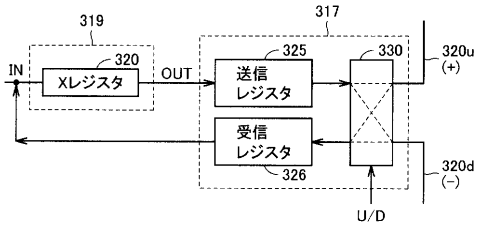
【図１１６】



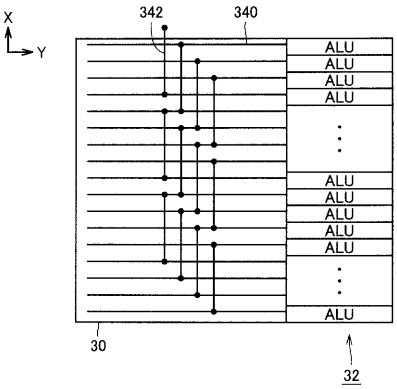
【図１１７】



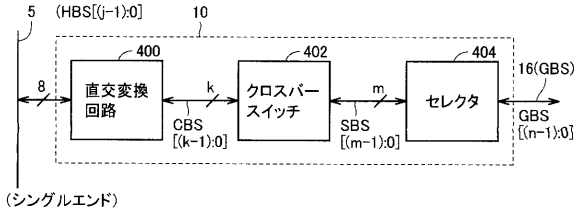
【図１１８】



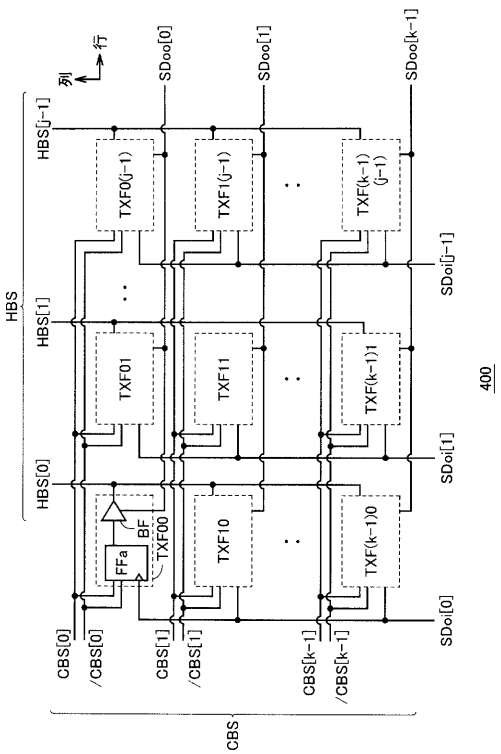
【図１１９】



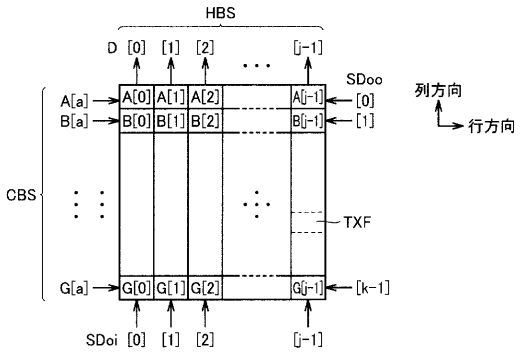
【図１２０】



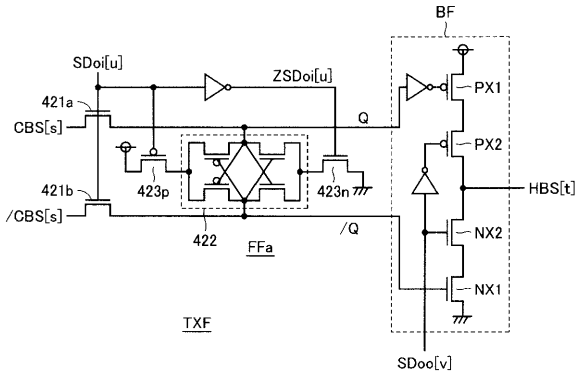
【図１２１】



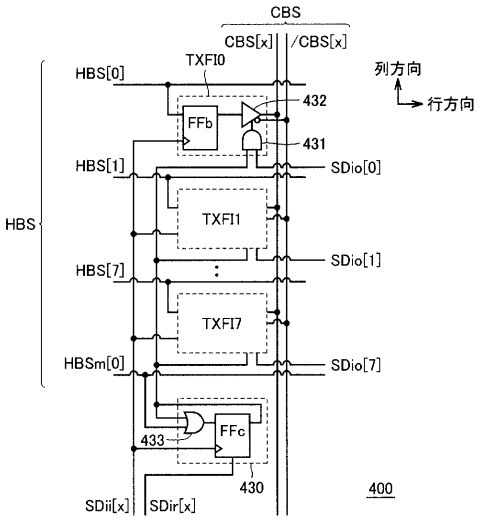
【図１２２】



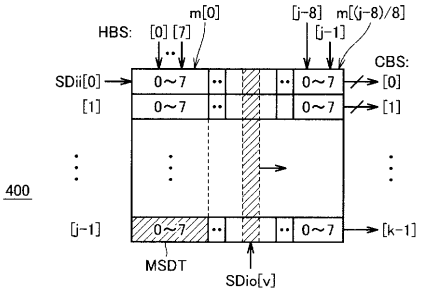
【図１２３】



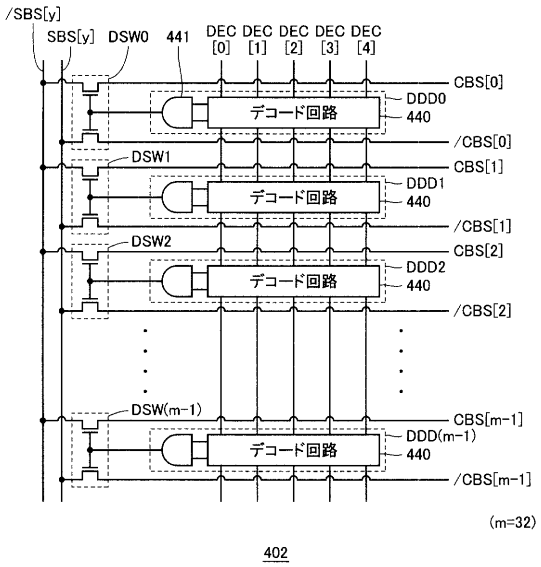
【図１２４】



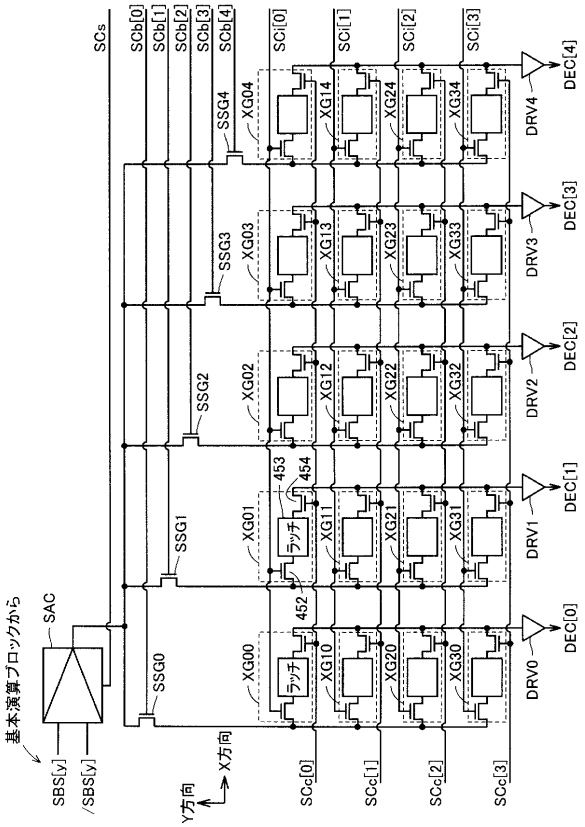
【図１２５】



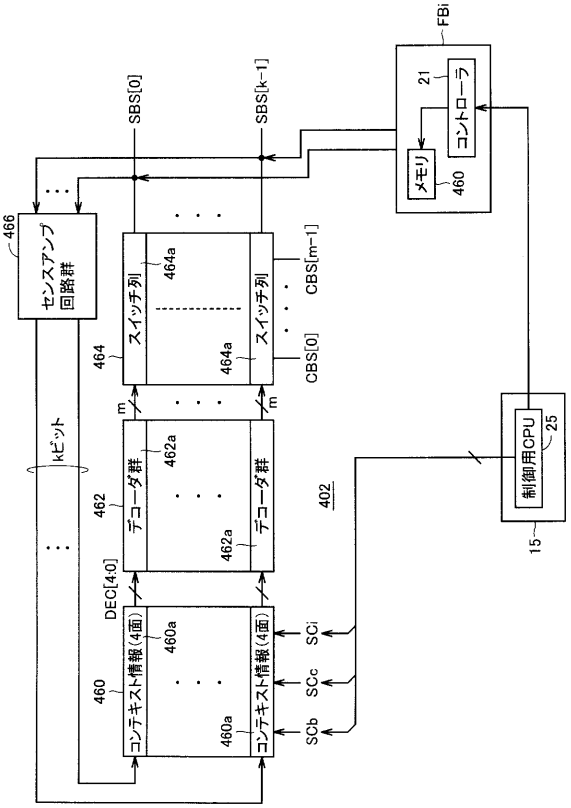
【図１２６】



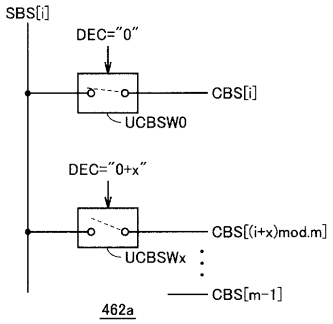
【図１２７】



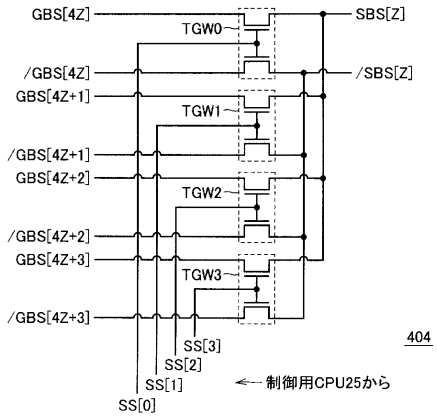
【図１２８】



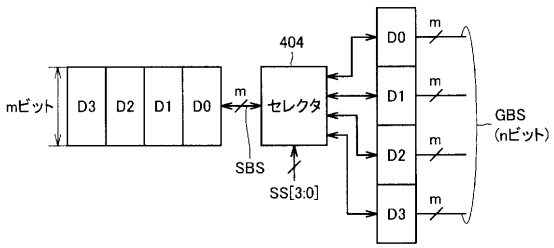
【図１２９】



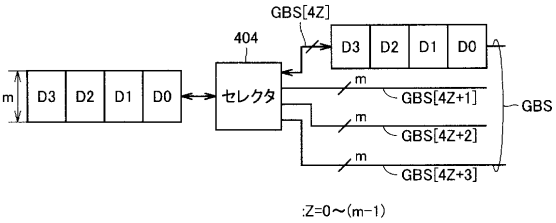
【図１３０】



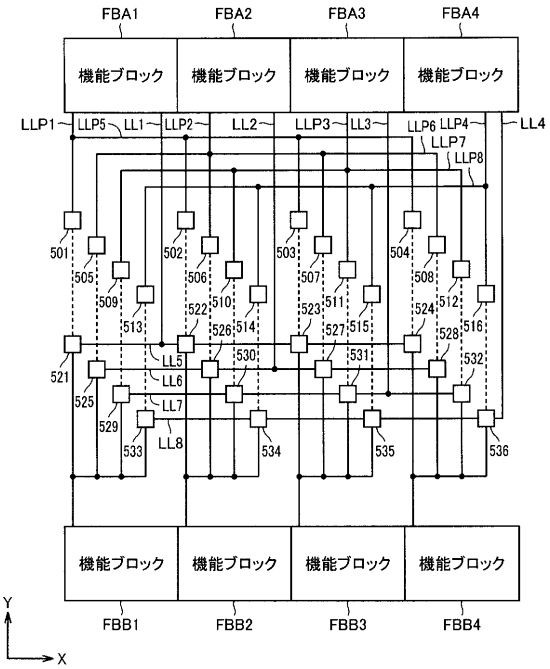
【図１３１】



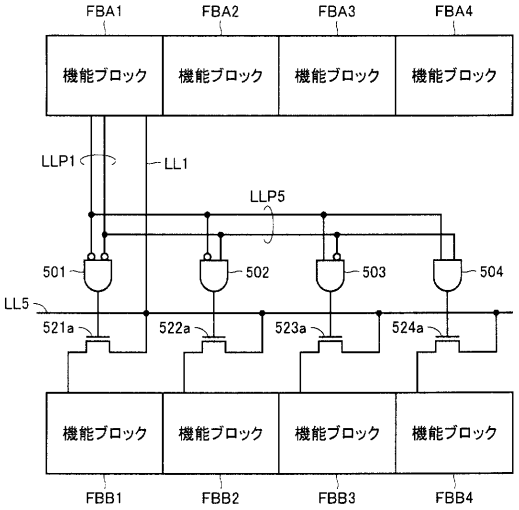
【図１３２】



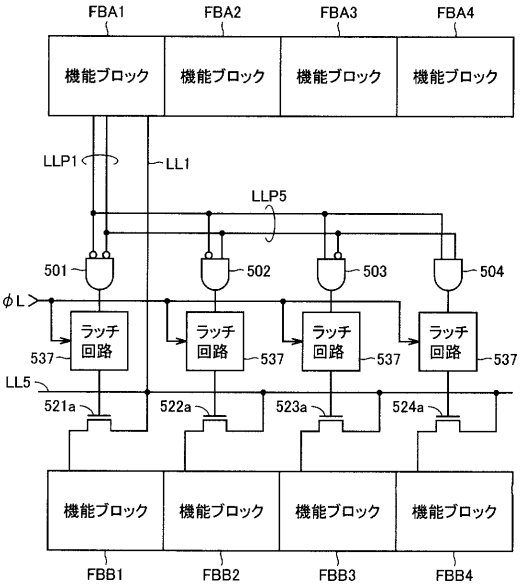
【図１３３】



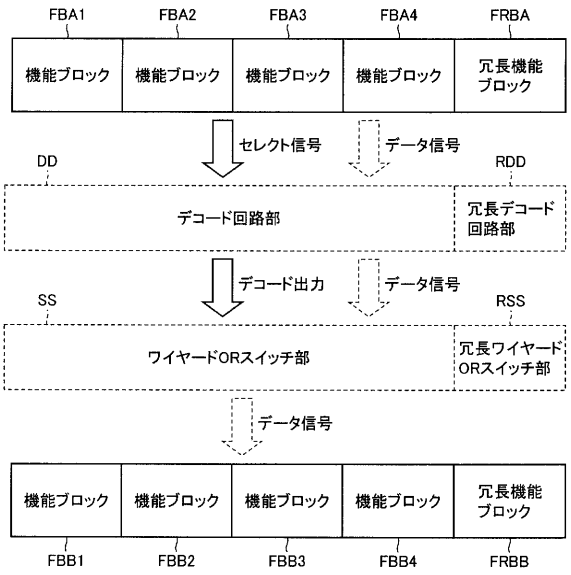
【図１３４】



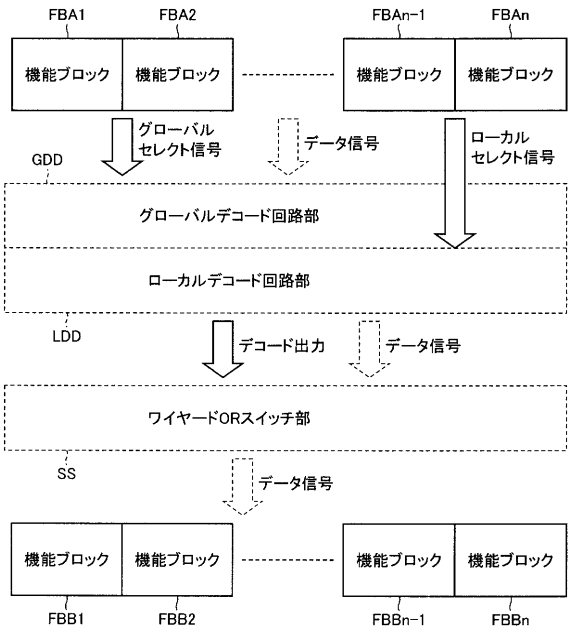
【図１３５】



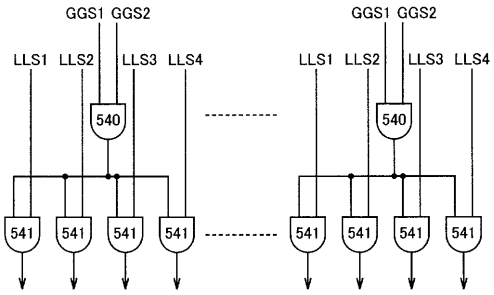
【図１３６】



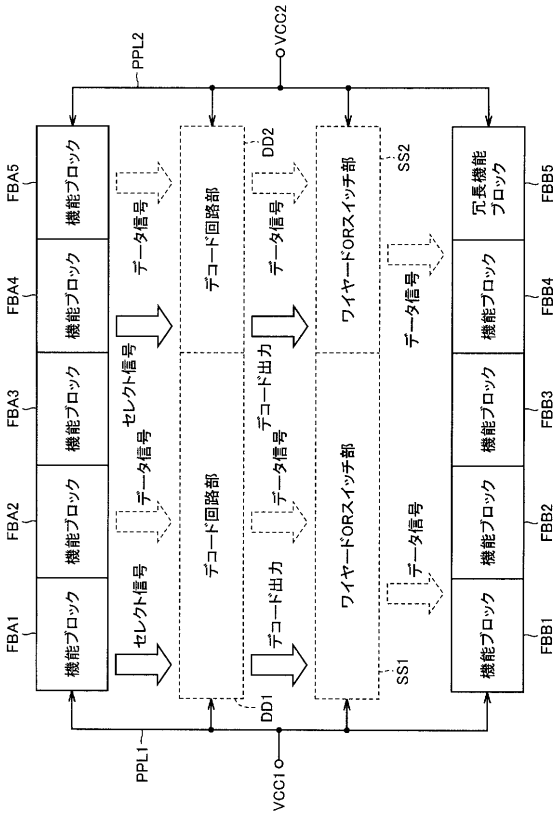
【図１３７】



【図１３８】


GGS1 GGS2
LLS1 LLS2 LLS3 LLS4
540
541

【図１３９】


PPL2
VCC2
FBA5
FBA4
FBA3
FBA2
FBA1
機能ブロック
データ信号
セレクト信号
デコード回路部
DD2
デコード出力
ワイヤードORスイッチ部
SS2
冗長機能ブロック
FBB5
FBB4
FBB3
FBB2
FBB1
DD1
SS1
PPL1
VCC1

フロントページの続き

(72)発明者　野田　英行
東京都千代田区丸の内二丁目４番１号　株式会社ルネサステクノロジ内
(72)発明者　齊藤　和則
東京都千代田区丸の内二丁目４番１号　株式会社ルネサステクノロジ内
(72)発明者　有本　和民
東京都千代田区丸の内二丁目４番１号　株式会社ルネサステクノロジ内
(72)発明者　堂阪　勝己
東京都千代田区丸の内二丁目４番１号　株式会社ルネサステクノロジ内

Ｆターム(参考) 5B056 AA05　BB42　BB71　HH03