(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3670160号
(P3670160)
(45)発行日　平成17年7月13日(2005.7.13)　(24)登録日　平成17年4月22日(2005.4.22)

(51)Int.Cl.[7]　F I
G06F 9/52　G06F 9/46　472B
G06F 9/38　G06F 9/38　310E

請求項の数 14　(全 54 頁)

(21)出願番号　特願平11-96570
(22)出願日　平成11年4月2日(1999.4.2)
(65)公開番号　特開平11-327930
(43)公開日　平成11年11月30日(1999.11.30)
審査請求日　平成14年2月6日(2002.2.6)
(31)優先権主張番号　09/055033
(32)優先日　平成10年4月3日(1998.4.3)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　502418767
アプライド　マイクロサーキッツ　コーポレイション
アメリカ合衆国　カリフォルニア　９２１２１，　サン　ディエゴ，　シークエンス　ドライブ　６２９０
(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

最終頁に続く

(54)【発明の名称】タスクに各リソースを割当てるための回路、複数のリソースを共用するための方法、命令を実行するためのプロセッサ、マルチタスクプロセッサ、コンピュータ命令を実行するための方法、マ

(57)【特許請求の範囲】

【請求項１】

複数のタスクによって共用されるべき複数のリソースを含むマルチタスクコンピュータシステムにおいて、データユニットを処理する際に前記タスクのあるものが前記リソースのあるものへのアクセスを終了した後に、他の前記タスクがいずれも前記リソースへのアクセスを終了するまで前記タスクの前記あるものが同じリソースにアクセスしないように、前記タスクに各リソースを割当てるための回路であって、

前記タスクのあるものが前記リソースのあるものにアクセスするステップは、

（１）前記タスクが前記リソースへのアクセスを試みるステップを含み、前記リソースへのアクセスを試みるステップは、前記タスクが前記リソースへのアクセスを試みていることを示す信号を発生するステップを含み、さらに

（２）動作（１）に応答して、前記タスクが前記リソースを利用可能ならば前記回路は前記タスクが前記リソースにアクセスするのを許し、前記タスクが前記リソースを利用可能になるまでは前記回路は前記タスクが前記リソースにアクセスするのを許さないステップと、

（３）前記回路が前記タスクが前記リソースにアクセスするのを許したときに前記タスクが前記リソースにアクセスするステップとを含み、

前記タスクのうちのあるタスクＴ１および前記リソースのうちのあるリソースＲ１に対して、前記タスクＴ１が前記リソースＲ１へのアクセスを既に終了した後に前記タスクＴ１が前記リソースＲ１へのアクセスを試み、かつ前記タスクＴ１が前記リソースＲ１への

アクセスを終了した後に少なくとも１つの他のタスクＴ２が前記リソースＲ１へのアクセスを試みていない場合、前記回路は、前記タスクＴ２が前記リソースＲ１へのアクセスを試み、かつ前記リソースＲ１にアクセスするまでは前記タスクＴ１が前記リソースＲ１にアクセスするのを許さない、回路。

【請求項２】

少なくとも１個のリソースに対して、各タスクは前記リソースをロックしてそれを他のどのタスクに対しても利用不可にすることによって前記リソースへのアクセスを開始し、前記タスクは前記リソースをアンロックすることによって前記リソースへのアクセスを終了する、請求項１に記載の回路。

【請求項３】

各データユニットは、前記リソースのうち少なくとも２個にアクセスして前記データユニットのうち少なくとも１つを処理する、前記タスクのうちの単一のものによって処理される、請求項１または２に記載の回路。

【請求項４】

前記少なくとも２個のリソースは記憶領域であり、その各々は、複数のデータユニットのデータユニット処理情報を記憶する、請求項３に記載の回路。

【請求項５】

前記記憶領域の１つはデータユニットを処理するリクエストを記憶するためのリクエストＦＩＦＯであり、前記記憶領域の別の１つは、前記データユニットの処理のためのコマンドを記憶するためのコマンドＦＩＦＯであり、

各データユニットごとに、前記タスクのうち１つは前記リクエストＦＩＦＯからリクエストを読出し、１つ以上のコマンドが前記データユニットについて前記コマンドＦＩＦＯに書込まれる場合、前記タスクのうち１つは前記１つ以上のコマンドを前記コマンドＦＩＦＯに書込む、請求項４に記載の回路。

【請求項６】

各リクエストは、対応のデータユニットのアドレスを含む、請求項５に記載の回路。

【請求項７】

前記データユニットのうち少なくともいくつかはネットワークを介して受信され、前記記憶領域のうち１つは、前記データユニットを処理するリクエストを記憶するためのリクエストＦＩＦＯであり、前記記憶領域の別のものはネットワークを介した前記データユニットの受信のステータス情報を記憶するためのステータスＦＩＦＯであり、

各データユニットごとに、前記タスクのうち１つは前記リクエストＦＩＦＯからリクエストを読出し、前記ステータスＦＩＦＯから前記ステータス情報を読出す、請求項４に記載の回路。

【請求項８】

複数のコンピュータタスクによって複数のリソースを共用するための方法であって、前記タスクの各々が前記リソースの各々にアクセスするステップを含み、

前記タスクのあるものが前記リソースのあるものにアクセスするステップは、

（１）前記タスクが前記リソースへのアクセスを試みていることを示す信号を前記タスクが発生するステップと、

（２）動作（１）に応答して、前記タスクが前記リソースを利用可能ならば前記タスクが前記リソースにアクセスするのを許し、前記タスクが前記リソースを利用可能になるまでは前記タスクが前記リソースにアクセスするのを許さないステップと、

（３）前記リソースへのアクセスを許されたときに前記タスクが前記リソースにアクセスするステップとを含み、

前記タスクのうちのあるタスクＴ１および前記リソースのうちのあるリソースＲ１に対して、前記タスクＴ１が前記リソースＲ１へのアクセスを既に終了した後に前記タスクＴ１が前記リソースＲ１へのアクセスを試み、かつ前記タスクＴ１が前記リソースＲ１へのアクセスを終了した後に少なくとも１つの他のタスクＴ２が前記リソースＲ１へのアクセスを試みていない場合、回路は、前記タスクＴ２が前記リソースＲ１へのアクセスを試み

、かつ前記リソースＲ１にアクセスするまでは前記タスクＴ１が前記リソースＲ１にアクセスするのを許さない、方法。
【請求項９】
少なくとも１個のリソースに対して、各タスクは前記リソースをロックしてそれを他のどのタスクに対しても利用不可にすることによって前記リソースへのアクセスを開始し、前記タスクは前記リソースをアンロックすることによって前記リソースへのアクセスを終了する、請求項８に記載の方法。
【請求項１０】
各データユニットは、前記リソースのうち少なくとも２個にアクセスして前記データユニットのうち少なくとも１つを処理する、前記タスクのうち単一のものによって処理される、請求項８または９に記載の方法。
【請求項１１】
前記少なくとも２個のリソースは記憶領域であり、その各々は、複数のデータユニットのデータユニット処理情報を記憶する、請求項１０に記載の方法。
【請求項１２】
前記記憶領域の１つはデータユニットを処理するリクエストを記憶するためのリクエストＦＩＦＯであり、前記記憶領域の別の１つは前記データユニットの処理のためのコマンドを記憶するためのコマンドＦＩＦＯであり、
各データユニットごとに、前記タスクのうち１つは前記リクエストＦＩＦＯからリクエストを読出し、１つ以上のコマンドが前記データユニットについて前記コマンドＦＩＦＯに書込まれる場合、前記タスクのうち１つは前記１つ以上のコマンドを前記コマンドＦＩＦＯに書込む、請求項１１に記載の方法。
【請求項１３】
各リクエストは、対応のデータユニットのアドレスを含む、請求項１２に記載の方法。
【請求項１４】
前記データユニットのうち少なくともいくつかはネットワークを介して受信され、前記記憶領域のうち１つは、前記データユニットを処理するリクエストを記憶するためのリクエストＦＩＦＯであり、前記記憶領域の別のものは、ネットワークを介した前記データユニットの受信のステータス情報を記憶するためのステータスＦＩＦＯであり、
各データユニットごとに、前記タスクのうち１つは前記リクエストＦＩＦＯからリクエストを読出し、前記ステータスＦＩＦＯから前記ステータス情報を読出す、請求項１１に記載の方法。
【発明の詳細な説明】
【０００１】
【関連出願への相互参照】
該当なし
【０００２】
【連邦政府資金による研究または開発に関する陳述】
該当せず
【０００３】
【発明の背景】
この発明はデータ処理に関し、より特定的にはパイプライン化された命令実行、マルチタスキングおよびリソースアクセス技術に関する。
【０００４】
パイプライン化およびマルチタスキングによりプロセッサの帯域幅が増加する。これらの技術に伴う時間および複雑さを低減することが望ましい。
【０００５】
特に、命令実行がパイプライン化されると、プロセッサは、命令を実行すべきであるかどうかがわかる前に命令を実行し始めることもあり得る。たとえば、プロセッサが命令Ｉ１を実行し始め、次に、Ｉ１の実行が完了する前に命令Ｉ２を実行し始めたとする。Ｉ１の

実行が完了できないのであれば、命令Ｉ２を実行すべきではなく、命令Ｉ２をパイプラインから除去しなければならない。実際に、どの時点においてもプロセッサはパイプラインから除去すべきである２つ以上の命令を実行しているかもしれない。パイプライン除去に関連する回路の複雑さを低減することが望ましい。

【０００６】

また、マルチタスキング環境においてさまざまなタスク間での切替えに伴うオーバヘッドを低減することも望ましい。タスクを切替えるには、プロセッサにより実行されるオペレーティングシステムが、どのタスクが次に実行されるべきであるかを定めなければならない。また、オペレーティングシステムは、あるタスクにより用いられたレジスタ値をセーブし、別のタスクにより用いられる値をレジスタにロードしなければならない。これらの機能は相当な数のオペレーティングシステム命令を伴うことがある。こうした動作に関連のある命令の数を減少することが望ましい。

【０００７】

また、利用可能でないかもしれないリソースへのアクセスを改善することが望ましい。そのようなリソースの一例としては、プロセッサが読もうとする際には空であり、またはプロセッサが書こうとする際には一杯であるかもしれないＦＩＦＯが挙げられる。ＦＩＦＯにアクセスする前に、プロセッサはＦＩＦＯが利用可能であるかどうかを示すフラグを調べる。利用可能でないかもしれないリソースにアクセスする速度を改善することが望ましい。

【０００８】

また、複数のタスクによるコンピュータリソースの利用を同期化するための単純な同期化方法を提供し、リソースが別のタスクによりアクセスされるよう設定されている際に、あるタスクがそのリソースにアクセスすることによって生じ得るエラーを回避することが望ましい。

【０００９】

【発明の概要】

この発明はいくつかの実施例において、効率のよいパイプラインプロセッサ、マルチタスキングプロセッサおよびリソースアクセス技術を提供する。

【００１０】

いくつかの命令実行パイプライン実施例において、パイプライン除去オーバヘッドはプロセッサがどのタスクに対しても続けて実行できる命令の数を制限することによって低減または消去される。このため、いくつかの実施例では、連続した命令は別々のタスクにより実行される。そのため、命令を実行できない場合でも、次の命令は異なるタスクに属するため当該次の命令は依然として実行されなければならない。このため、次の命令はパイプラインから除去されない。

【００１１】

いくつかの実施例では、同じタスクのどの２つの命令の間でも、プロセッサは別のタスクからの十分な数の命令を実行してパイプライン除去の必要をなくしている。

【００１２】

タスク切替えに関連するオーバヘッドを低減するため、いくつかの実施例では各々のタスクに対する別個のレジスタが含まれており、レジスタ値がタスク切替え動作においてセーブされたり、またはリストアされたりする必要がないようにする。特に、いくつかの実施例では、各タスクは別個のプログラムカウンタ（ＰＣ）レジスタおよび別個のフラグを有する。いくつかの実施例において、タスク切替えはハードウェアによって１クロックサイクルで行なわれる。

【００１３】

いくつかの実施例において、プロセッサは、リソースが利用可能であるかどうかを予めチェックすることなくリソースにアクセスすることができる。プロセッサがリソースにアクセスする命令を実行する際にリソースが利用可能でない場合、プロセッサはその命令をサスペンドする。その命令を実行するはずであったプロセッサ回路は、たとえば別のタスク

の命令など、別の命令を実行するのに利用可能となる。

【００１４】

そのため、いくつかの実施例では、プロセッサはすべてのリソース（たとえばＦＩＦＯなど）の状態を追跡する。（特に断っていない限り、ここで用いている術語「リソース」はある時点において利用可能であるかもしれない、または利用可能でないかもしれない何かを意味する。）生成される信号は各リソースの状態を示しており、特にどのリソースがどのタスクに利用可能であるのかを示す。あるタスクが利用可能でないリソースにアクセスを試みた場合、そのタスクはサスペンドされ、プロセッサは本来そのサスペンドされたタスクが利用できたはずのタイムスロット内で他のタスクを実行できる。リソースが利用可能となると、そのサスペンドされたタスクは再開され、そのリソースにアクセスする命令が再実行される。

【００１５】

複数のタスクが１つ以上のリソースを共用する際の同期エラーを避けるため、いくつかの実施例では、あるタスクがリソースのいずれか１つにアクセスし終わると、そのタスクは、そのリソースを共用する他のすべてのタスクがそのリソースにアクセスし終わるまでそのリソースにアクセスすることができない。このため、いくつかのネットワーク実施例では、さまざまなタスクがＦＩＦＯリソースを共用してデータフレームを処理する。各タスクは別個のデータフレームを処理する。フレームを処理するには、タスクは「リクエスト」ＦＩＦＯからフレームアドレスを読む。次に、タスクはコマンドＦＩＦＯに、チャネルプロセッサがフレームを処理するようにというチャネルプロセッサへのコマンドを書込む。２つ目のタスクは異なるフレームに対して同様の動作を行なう。始めのタスクは再び、さらに別のフレームに対して同じ動作を行なう。あるフレームに対して書かれたコマンドが別のフレームに誤って適用された場合、フレームは誤って処理される可能性がある。

【００１６】

この可能性をなくし、リクエストＦＩＦＯにおけるフレームアドレスとコマンドＦＩＦＯにおけるコマンドとを正確に一致させるため、以下の技術が用いられる。まず、あるタスク（たとえばＴ１）がリクエストＦＩＦＯおよびコマンドＦＩＦＯの双方にアクセスを許されるが、他のどのタスクもこれらのリソースにアクセスすることを許されない。一旦タスクＴ１がいずれかのリソースにアクセスし終わると、そのリソースに別のタスクがアクセスすることが許可され、さらにタスクＴ１は、そのリソースを共用する他のすべてのタスクがそのリソースにアクセスし終わるまでは再びそのリソースにアクセスすることを許されない。このため、リクエストＦＩＦＯにおけるフレームアドレスの順序はコマンドＦＩＦＯにおけるコマンドの順序と対応しており、チャネルがフレームアドレスとコマンドとを正確に一致させることができるようにする。この一致を確立するのに特殊なタグは何ら必要ではなく、この一致は単純なデータ構造であるＦＩＦＯを用いて確立される。

【００１７】

いくつかの実施例では、プロセッサはネットワークのデータフローを処理するいくつかのタスクを実行する。プロセッサは高い帯域幅をもたらすのに、上述のパイプラインおよびタスク切替え技術を用いる。

【００１８】

他の実施例および変更例を以下に説明する。この発明は添付の特許請求の範囲により規定される。

【００１９】

【好ましい実施例の説明】

図１には、パイプライン化されたマルチタスキングプロセッサ（マイクロコントローラ）１６０を含むポートインターフェイス（ＰＩＦ）回路１１０が示される。ポートインターフェイス１１０は、４つの全二重ポートを含み、これらはＡＴＭスイッチ１２０とそれぞれ対応の４つのイーサネットセグメント（図示せず）との間でインターフェイスをもたらし、４つのイーサネットセグメントの各々は対応するＭＡＣ１３０．０－１３０．３に接続される。各ポート「ｘ」（ｘ＝０，１，２，３）において、イーサネットセグメントと

ＡＴＭスイッチ１２０との間のデータは、対応するＭＡＣ１３０．ｘおよび対応するスライサ１４０．ｘを通って流れる。スライサは周知のＡＴＭ　ＳＡＲ機能を実行し、イーサネットフレームをＡＴＭセルに分割し、ＡＴＭスイッチへ向かう途中でセルにＡＴＭヘッダを付加し、イーサネットセグメントへ向かう途中でセルからフレームをアセンブルする。いくつかの実施例では、ＰＩＦ１１０へのＡＴＭスイッチインターフェイスはフレームモードで動作し、このモードにおいてはＡＴＭスイッチは、間に介在するセルのない状態でセルのフレームをスライサ１４０に送る。スライサ１４０はＡＡＬ－５プロトコルを用いる。フレームモードはたとえば、Ａ．ヨッフェ（Joffe ）他による１９９６年８月３０日出願の「ＡＴＭスイッチにおけるセルキューイング」（“Cell Queuing in ATM Switches”）と題された米国特許出願第０８／７０６，１０４号に記述される。また、１９９７年８月２８日出願の、ここに引用により援用されるＰＣＴ出願ＰＣＴ／ＵＳ９７／１４８２１を参照されたい。

【００２０】

ＰＩＦ１１０の他の実施例では、必ずしもＡＴＭまたはイーサネットではない他のネットワーク間でインターフェイスがもたらされる。いくつかの実施例では、スライサ１４０は適当なＭＡＣで置換えられる。

【００２１】

プロトコル変換（たとえば、ＡＴＭ／イーサネット変換）を実行するのに加えて、ＰＩＦ１１０はＩＰルーティング、レイヤー２スイッチングまたは、ＰＩＦマイクロコントローラ１６０により実行されるソフトウェアにより定められる他の処理を行なうことができる。図３、図４に関連して以下の説明を参照されたい。また、本出願と同日出願のＡ．ヨッフェ他による「ネットワークでデータを転送するためのネットワークプロセッサシステムおよび方法」（“SYSTEMS AND METHODS FOR DATA TRANSFORMATION AND TRANSFER IN NETWORKS”）と題された米国特許出願第０９／０５５，０４４号（代理人事件番号Ｍ－４８５５　ＵＳ）も参照されたい。これもここに引用により援用される。

【００２２】

ＰＩＦ１１０はさほど速くないクロック速度においてであってもスループットが高い。そのため、いくつかの実施例では、ＰＩＦ１１０はわずか５０ＭＨｚのクロック速度において４つの１００ＭＢ／秒イーサネットポートおよびそれぞれ対応の４つの１５５ＭＢ／秒ＡＴＭポートに対してＩＰルーティングを行なうことができる。

【００２３】

図１では、各スライサ１４０．ｘとその対応するＭＡＣ１３０．ｘとの間のデータフローは対応するチャネル１５０．ｘ（以下、チャネル「ｘ」、すなわちチャネル０，１，２または３、とも称される）により制御される。チャネル１５０はマイクロコントローラ１６０からのコマンドを実行する。いくつかの実施例では、４つのチャネル１５０．ｘは、時分割多重方式を用いて４つのチャネル１５０の機能を果たす１つのチャネル回路により実現される。ここに引用により援用される前述の米国特許出願第０９／０５５，０４４号の代理人事件番号Ｍ－４８５５　ＵＳ「ネットワークでデータを転送するためのネットワークプロセッサシステムおよび方法」を参照されたい。

【００２４】

チャネル、マイクロコントローラ、スライサ１４０およびＭＡＣ１３０はメモリ１６４を介して通信し、メモリ１６４は以下に説明する内部メモリ（「フレームおよびコマンドメモリ」）１７０およびＦＩＦＯ２３０、２４０を含む。

【００２５】

いくつかのイーサネットの実施例では、マイクロコントローラはＭＩＩ（メディア独立インターフェイス）管理回路１８０に接続され、ＭＩＩ管理回路１８０は当該技術分野において公知であるイーサネット物理層装置に接続される。

【００２６】

サーチマシン（ＳＭ）１９０はメモリ２００内にアドレス導出データベースを保持し、ＩＰルーティングまたはソフトウェアにより定められる他の処理を行なう。ＳＭ１９０はま

た、メモリ２００内に（たとえばＶＬＡＮまたはアクセス制御リストを規定することによって）ネットワーク接続性を制限するデータベースを保持する。サーチマシンはマイクロコントローラ１６０により提示されるキー（たとえばイーサネットまたはＩＰアドレス）を探索することができ、学習アルゴリズムを実行してそのアドレスがデータベースにない場合にレイヤー２またはレイヤー３アドレスを学習する。サーチマシン１９０はいくつかの実施例ではソフトウェアプログラマブルではないが、サーチマシンはフレキシブルなデータベースノード構造をサポートしており、このため、サーチマシンは異なる機能（たとえばＩＰルーティング、レイヤー２スイッチングなど）にたやすく適合できる。サーチマシン１９０は、サーチ、インサート、デリートなどのマイクロコントローラからのコマンドを実行する。サーチマシンはまた、マイクロコントローラにメモリ２００への直接のアクセスを与える。サーチマシンについては、追補８に記載する。

【００２７】

いくつかの実施例では、メモリ２００はフロースルー動作モードにおいてシンクロナススタティックＲＡＭを用いて実現される。いくつかの実施例では複数のメモリバンクが用いられている。

【００２８】

いくつかの実施例では、ＰＩＦ１１０は集積回路である。メモリ２００は集積回路の一部ではないため「外部」と称される。しかしながら、他の実施例では、メモリ２００は同じ集積回路の一部である。この発明はいかなる特定の集積方式によっても限定されるものではない。

【００２９】

ＰＩＦ１１０はまた、シリアル読出専用メモリ（ＲＯＭ）２０４（いくつかの実施例ではシリアルＥＰＲＯＭ）に接続され、ブート時にＲＯＭ２０４からマイクロコントローラへソフトウェア（「ファームウェア」）がロードされるようにする。

【００３０】

図２には、メモリ１６４内の１つのチャネル１５０．ｘおよびその関連のＦＩＦＯリソースが示される。チャネルは以下の２つの類似した部分に分割される。すなわち、対応するＭＡＣ１３０から対応するスライサ１４０へのデータフローを制御する入ってくる（イングレス）サブチャネル１５０Ｉと、スライサ１４０からＭＩＣ１３０へのデータフローを制御する出ていく（イーグレス）サブチャネル１５０Ｅとである。特に断っていない限り、参照番号においてサフィックス「Ｉ」はイングレスサブチャネルに属する回路を示し、サフィックス「Ｅ」はイーグレスサブチャネルに属する回路を示す。

【００３１】

各サブチャネル１５０Ｉ、１５０Ｅにおいてデータ処理は以下のステップを含む。

【００３２】

（１）　対応する入力制御ブロック２１０（すなわち２１０Ｉまたは２１０Ｅ）は対応するデータＦＩＦＯ２２０に入来データをストアする。マイクロコントローラがアドレス変換または他の処理を開始できるようにするのに十分なデータフレームの部分が受取られると（たとえばＩＰルーティング実施例ではＩＰアドレスおよびホップカウントが受取られると）、入力制御２１０はそれぞれ対応のリクエストＦＩＦＯ２３０にリクエストを書く。リクエストがＦＩＦＯ２３０に書かれる前に受取られるフレームバイトの数は、前述の米国特許出願第０９／０５５，０４４号の代理人事件番号Ｍ－４８５５　ＵＳに説明されるようにマイクロコントローラにより書込可能なレジスタにより規定される。

【００３３】

（２）　マイクロコントローラ１６０はリクエストを読み、対応するデータＦＩＦＯ２２０から適当なパラメータ（たとえばイングレス側での発信元および行先アドレスまたはイーグレス側でのＶＰＩ／ＶＣＩ）を読み、適当な処理を行なう。マイクロコントローラはたとえばアドレス導出探索を行なうため、必要に応じてサーチマシン１９０を用いる。

【００３４】

（３）　サーチマシン１９０がマイクロコントローラ１６０に探索結果を返すと、マイク

ロコントローラはそれぞれ対応のコマンドＦＩＦＯ２６０に１つ以上のチャネルコマンドを書き、これはフレームをどのように出力装置（ＭＡＣ１３０またはスライサ１４０）に転送するべきであるかを特定する。

【００３５】

（４）　フレーム全体が受取られた後、入力制御２１０はそれぞれ対応のステータスＦＩＦＯ２４０にステータス情報を書く。ステータスＦＩＦＯはマイクロコントローラ１６０により読まれる。ステータスによりフレームが不良であることが示されると（たとえばチェックサムが不良である場合）、マイクロコントローラはコマンドＦＩＦＯ２６０に「破棄」コマンドを書いて出力制御２５０がそのフレームを破棄するようにさせる。

【００３６】

ステップ（２）、ステップ（３）およびステップ（４）は図３、図４に関連して以下に説明する他の処理を含んでいてもよい。

【００３７】

（５）　出力制御２５０はそれぞれ対応のコマンドＦＩＦＯ２６０からのコマンドを実行する。

【００３８】

いくつかの実施例では、データＦＩＦＯ２２０およびコマンドＦＩＦＯ２６０は内部メモリ１７０内にストアされる。リクエストＦＩＦＯ２３０およびステータスＦＩＦＯ２４０はメモリ２３０、２４０（図１）内にストアされる。

【００３９】

イーグレス出力制御ブロック２５０Ｅの出力はマイクロコントローラに接続され、ＡＴＭスイッチ１２０がプログラム（「アプレット」）をマイクロコントローラにロードして実行できるようにする。アプレットはまず、他のフレームと類似した態様でイーグレス側に転送されるが、これらのアプレットのＶＰＩ／ＶＣＩパラメータはマイクロコントローラを示している。そこで、アプレットはＭＡＣ１３０には転送されない。代わりに、アプレットは回路２５０Ｅの出力からマイクロコントローラプログラムメモリ３１４（図６）にＤＭＡ転送によりロードされる。

【００４０】

マイクロコントローラ１６０はまた、それ自身のフレームを発生し、これをいずれかのデータＦＩＦＯ２２０へ書き、対応するコマンドＦＩＦＯ２６０へコマンドを書くことができる。対応する出力制御２５０はそのコマンドにより特定されるとおりにフレームを転送する。

【００４１】

マイクロコントローラはまた、コマンドＦＩＦＯ２６０に、各サブチャネル１５０Ｉ、１５０Ｅのための別個のメモリ（図示せず）にストアされる統計情報を転送するコマンドを書くことができる。

【００４２】

いくつかの実施例では、マイクロコントローラ１６０は費用のかかるリソースである。注目すべきことは、いくつかの実施例においてマイクロコントローラ命令実行ユニット（図６において３１０として示され以下に説明される）がＰＩＦ１１０のゲートカウントの約７０％を占めることである。このため、マイクロコントローラを常に稼動させることが望ましい。以下のように適当なマルチタスキングによって完全稼動させることができる。

【００４３】

マイクロコントローラは、各ポート０、１、２、３につき１つずつの、４つの「ハードウェアタスク」ＨＴ０、ＨＴ１、ＨＴ２、ＨＴ３を実行する。ハードウェアタスクは以下の表に示すように時分割多重方式で実行される。

【００４４】

【表１】

クロック サイクル	1	2	3	4	5	6
ハードウェア タスク	HT0	HT1	HT2	HT3	HT0	HT1

【００４５】
ハードウェアタスクが利用可能でない場合（たとえば、ハードウェアタスクがサーチマシンを待っているためなど）、それぞれのクロックサイクルにおいてどのマイクロコントローラ命令も開始されない。

【００４６】
各ハードウェアタスクは１つ以上のソフトウェアタスクを含む。各ソフトウェアタスクはフレーム全体を処理するコードを含む。イングレス側のフレームとイーグレス側のフレームとが並列に到着し得るため、いくつかの実施例では各ハードウェアタスクは少なくとも２つのソフトウェアタスクを含んでおり少なくとも２つのフレームを並列処理できるようにする。いくつかの実施例では、異なるソフトウェアタスクがイングレス側とイーグレス側とに提供される。イングレスソフトウェアタスクが、たとえばマイクロコントローラがサーチマシンを待っているため実行できない場合、マイクロコントローラはイーグレスソフトウェアタスクを実行できる。また、その逆も可能である。

【００４７】
以下、術語「タスク」は、特に「ハードウェアタスク」と記載しない限り、ソフトウェアタスクを意味するものとする。

【００４８】
図３には、イングレスタスクによる１つのフレームのレイヤー３処理が示される。ステージ２９０ＤＡでは、サブステージ２９０ＤＡ．１においてマイクロコントローラはフレームから、イーサネット（ＭＡＣ）行先アドレスＤＡを読む。マイクロコントローラはこのアドレスをサーチマシン１９０に供給し、サーチマシン１９０はサブステージ２９０ＤＡ．２において探索を行なう。

【００４９】
サブステージ２９０ＤＡ．３において、マイクロコントローラは探索結果を調べる。ＤＡが見つからなかった場合、フレームは捨てられるか、またはブロードキャストされる。ＤＡが見つかり、サーチマシンがそのＤＡを最終行先ステーションのアドレスであると認識した場合、探索結果は、フレームが最終行先に送られることとなる仮装接続（ＶＣ）のＶＰＩ／ＶＣＩを含むことになる。この場合、ＩＰステージ２９０ＩＰはスキップされる。探索結果からＤＡがＩＰルーティングエンティティに割当てられたアドレスであることが示されると、ＩＰ処理がステージ２９０ＩＰにおいて行なわれる。

【００５０】
そのステージでは、サブステージ２９０ＩＰ．１においてマイクロコントローラはフレームからＩＰ行先アドレスを読む。サーチマシンはステージ２９０ＩＰ．２においてそのアドレスに対して探索を行なう。マイクロコントローラはサブステージ２９０ＩＰ．３において探索結果を調べる。結果にはＶＰＩ／ＶＣＩが含まれ、またアクセス制御制約が含まれることもある。サブステージ２９０ＩＰ．３において、マイクロコントローラはアクセス制御制約をＩＰ発信元アドレスと突き合わせ、フレームが許可されるかどうかを判定する。許可されない場合、フレームは捨てられる。

【００５１】
ステージ２９０ＳＡにおいて、イーサネット発信元アドレスＳＡは処理され、アドレス学習アルゴリズムを実現し、またＶＬＡＮを実現する。より特定的には、サブステージ２９０ＳＡ．１において、サーチマシンはＳＡに対して探索を行なう。サーチマシンは学習アルゴリズムがリクエストするのであればＳＡデータを挿入または修正する。サブステージ２９０ＳＡ．２において、サーチマシンは、ＳＡが属するＶＬＡＮを戻す。サブステージ

２９０ＳＡ．３において、マイクロコントローラはそのＶＬＡＮをステージ２９０ＤＡ．２においてサーチマシンが返したＤＡ　ＶＬＡＮと比較する。イーサネット発信元および行先アドレスが異なるＶＬＡＮに属する場合、フレームは捨てられる。

【００５２】

サブステージ２９０ＤＡ．３、２９０ＩＰ．３、２９０ＳＡ．３のうちの１つまたは２つ以上において、マイクロコントローラはそれぞれ対応のデータフロー（すなわちそれぞれ対応のサブチャネル）に対してコマンドＦＩＦＯ２６０Ｉへコマンドを書く。コマンドはチャネル１５０がフレームを捨てるように指示しても、またはフレームをそれぞれ対応のスライサ１４０に転送するように指示してもよい。フレームが転送される場合、チャネルは、コマンドの指示に従い、スライサにＶＰＩ／ＶＣＩを供給したり、また、ＩＰホップカウントをインクリメントしたり、および／または発信元アドレスをそれぞれ対応のＭＡＣ１３０のアドレスで置換えたりする。

【００５３】

図４には、１つのフレームに対してイーグレスタスクにより行なわれる処理が示される。ステージ２９４ＶＣにおいて、タスクはＶＰＩ／ＶＣＩを調べてフレームがアプレットであるかどうかを判定する。もしそうであれば、タスクはそのフレームをマイクロコントローラプログラムメモリ（以下に説明する図６において３１４として示される）にロードし、アプレットを実行する。ステージ２９４ＩＰはスキップされる。

【００５４】

代わりに、ＶＰＩ／ＶＣＩはフレームがＡＴＭスイッチ１２０からの情報リクエストであることを示していてもよい。そのようなリクエストの例としては、ＰＩＦ１１０内のレジスタを読むリクエスト、または統計情報を読むリクエストが含まれる。イーグレスタスクがそのリクエストを行なう。それが情報のリクエストであれば、イーグレスタスクは、イーグレスタスクを実行するのと同じハードウェアタスクのイングレスコマンドＦＩＦＯ２６０Ｉに１つ以上のコマンドを書く。これらのコマンドにより、チャネルがその情報をスイッチに送ることになる。ステージ２９４ＩＰはスキップされる。

【００５５】

ＶＩＰ／ＶＣＩがスイッチ１２０からのどの管理リクエスト（情報のリクエストなど）をも示さない場合、ステージ２９４ＩＰが行なわれる。サブステージ２９４ＩＰ．１において、タスク（すなわちマイクロコントローラ）はフレームからＩＰ行先アドレスを読み、そのアドレスをサーチマシンに供給する。ステージ２９４ＩＰ．２において、サーチマシンは探索を行ない、イーサネット行先アドレスを返し、場合によってはアクセス制御情報をも返す。ステージ２９４ＩＰ．３において、タスクはそのイーグレスコマンドＦＩＦＯ２６０Ｅへコマンドを書き、フレームのイーサネット行先アドレスをサーチマシンがもたらすアドレスで置換えさせ、イーサネット発信元アドレスをそれぞれ対応のＭＡＣ１３０．ｘのアドレスで置換えさせ、フレームをＭＡＣへ転送させる。また、タスクソフトウェアに応じて他の種類の処理が行なわれてもよい。

【００５６】

マイクロコントローラがステージ２９０ＤＡ．２、２９０ＩＰ．２、２９０ＩＳＡ．２、２９４ＩＰ．２においてサーチマシンを待っている間、マイクロコントローラは同じまたは別のハードウェアタスクにおける別のソフトウェアタスクを実行するのに利用可能である。

【００５７】

いくつかの実施例では、各イングレスフローおよび各イーグレスフローに対して１つのタスクを有しているだけではマイクロコントローラを完全に稼動させることにならず、そのため、各半二重データフローに対して２つ以上のタスクが提供され、マイクロコントローラが各データフローにおいて２つ以上のフレームを並列に処理できるようにする。このことは以下を考慮することによって説明される。イーサネットフレームが短いときにマイクロコントローラの速度が最も要求される。なぜなら、図３、図４に示される同じ処理を短いフレームと長いフレームとの双方に対して行なわなければならないからである。最も短

いイーサネットフレームは６４バイトを有する。たとえば、４つのイーサネットポートが１００ＭＢ／秒ポートであり、ＡＴＭポートが１５５ＭＢ／秒であるとする。１００ＭＢ／秒では、最も短いフレームはイーサネットポートを５．１２マイクロ秒で通過する。このため、マイクロコントローラおよびサーチマシンはフレームを５．１２＋１．６＝６．７２マイクロ秒で処理しなければならない（１．６マイクロ秒はフレーム間ギャップである）。

【００５８】

５０ＭＨｚのマイクロコントローラクロック速度を仮定しよう。これは信頼性のある動作を確実にするためのかなり遅いクロック速度である。他の実施例ではより速い速度（たとえば１００ＭＨｚ）が用いられる。５０ＭＨｚにおいて、６．７２マイクロ秒は３３６クロックサイクルである。このため、１つのハードウェアタスクのイングレスおよびイーグレスタスクのためのクロックサイクル割当は３３６／４＝８４クロックサイクルである。

【００５９】

フレームの処理はマイクロコントローラとサーチマシンとで分けられているが、これらは必ずしも同じフレームに対して並列に作業する訳ではないため、同じハードウェアタスク内の１つのイングレスフレームと１つのイーグレスフレームとに対する処理レイテンシは結線速度での処理においてさえ、８４サイクルより長くてよい。処理するのに８４サイクルよりも長くかかり、６４バイトのフレームがイングレス側およびイーグレス側において連続して到着する場合、同じデータフローにおける前のフレームが処理される前に次のフレームが到着し始めることがあり得る。このため、マイクロコントローラが、同じデータフローにおける前のフレームの処理が完了する前に次のフレームを処理し始めることができるようにすることが望ましい。同じデータフローにおける複数のフレームのこのような並列処理を実現するために、各データフローに対して２つ以上のソフトウェアタスクが提供される。

【００６０】

このように、いくつかの実施例では、各ハードウェアタスクＨＴｘは２つのイングレスタスクＩＧｘ．０、ＩＧｘ．１および２つのイーグレスタスクＥＧｘ．０、ＥＧｘ．１を含む。たとえば、ハードウェアタスクＨＴ１はイングレスタスクＩＧ１．０、ＩＧ１．１およびイーグレスタスクＥＧ１．０、ＥＧ１．１を含む。各タスクは以下のものを含む４ビットのタスク番号により識別される。

【００６１】

ＣＨＩＤ－－それぞれ対応のポート０、１、２、３に対してのチャネルＩＤ（２ビット）＝０、１、２、または３
ＳＮ－－シーケンス番号（ＩＧｘ．０、ＥＧｘ．０では０、ＩＧｘ．１、ＥＧｘ．１では１）
Ｉ／Ｅ－－イングレスには０、イーグレスには１
タスクの総数はすなわち１６である。

【００６２】

１つのフレームは１つのタスクにより処理される。フレームがアプレットである場合、そのアプレットは同じタスクにより実行される。

【００６３】

マイクロコントローラ命令実行はパイプライン化される。このため、上記の表１では、それぞれ対応のハードウェアタスクに対して新しい命令が開始されるクロックサイクルが示される。たとえば、サイクル１では、ハードウェアタスクＨＴ０に対して命令実行が開始される。命令実行は後続のサイクルにおいても続く。

【００６４】

各サブチャネルにおけるＦＩＦＯ２３０、２４０、２６０へのタスクのアクセスは図５の論理図に示される態様で制御される。図５では、「タスク０」および「タスク１」は同じサブチャネルに対する２つのタスクであり、たとえばチャネル１５０．１のサブチャネル１５０Ｉに対するイングレスタスクＩＧ１．０、ＩＧ１．１である。始めはタスク０のみ

がサブチャネルＦＩＦＯ２３０、２４０、２６０へのアクセスを有する。タスク０がリクエストＦＩＦＯ２３０にアクセスすると、スイッチ「ａ」が切替えられ、リクエストＦＩＦＯをタスク１に接続する。タスク０はタスク１がリクエストＦＩＦＯを読むまでリクエストＦＩＦＯを読むことが許されない。

【００６５】

スイッチ「ｂ」はコマンドＦＩＦＯ２６０へのタスクのアクセスを制御する。スイッチ「ｂ」は、タスク０によってあるフレームに対するすべてのコマンドが書かれた際に切替えられる。

【００６６】

ステータスＦＩＦＯ２４０へのタスクのアクセスを制御するスイッチ「ｃ」はステータスＦＩＦＯがタスク０により読まれた際に切替えられる。

【００６７】

サーチマシンへのタスクのアクセスを同期化するため、サーチマシン１９０は次々にコマンドを実行し、結果を同じ順序でもたらす。

【００６８】

各命令において、実行のためのタスクを選択するのには１クロックサイクルしか要さない（以下に説明する図７におけるパイプラインステージＴＳ）。さらに、タスクの選択はパイプライン化されており、そのためスループットに影響を及ぼさない。タスクの選択はハードウェアにより行なわれる。マイクロコントローラにおいてオペレーティングシステムは用いられない。このため、低いレイテンシが達成される。

【００６９】

どの時点においても、各タスクは３つの状態、すなわち、アクティブ、レディまたはサスペンド、のうちの１つにある。アクティブ状態では、タスクは実行されている状態である。最も多くて４つのタスク（各ハードウェアタスクにつき１つ）が同時にアクティブであり得る。各アクティブタスクは４クロックサイクルごとに実行がスケジュールされる（上の表１を参照のこと）。

【００７０】

アクティブタスクは、タスクが利用可能でないリソースにアクセスを試みるとサスペンド状態に移行する。リソースは追補２に記載される。リソースが利用可能となるとタスクはレディ状態に入る。

【００７１】

アクティブタスクがサスペンドされると、同じチャネルにおけるレディ状態にあるタスクのうちの１つがタスク制御３２０（図６）により実行のため選択され、アクティブ状態に移行する。

【００７２】

図６はマイクロコントローラ１６０のブロック図である。実行ユニット３１０はプログラムメモリ３１４にストアされるプログラムを実行する。プログラムはブートの間に、ＲＯＭ２０４（図１）からダウンロードされる。さらに、上述のようにアプレットをロードして動的に実行することもできる。アプレットは実行された後に破棄されてもよく、またはアプレットはメモリ３１４に残されてもよい。

【００７３】

実行ユニット３１０は、汎用レジスタを有するレジスタファイル３１２、特殊レジスタブロック３１５およびデータメモリ３１６を含む。レジスタファイル３１２は、それぞれ対応のバスsa__bus 、sb__bus に接続される２つの３２ビット出力を含み、バスsa__bus 、sb__bus はＡＬＵ３１８の入力に接続される。データメモリ３１６および特殊レジスタブロック３１５の３２ビット出力はsa__bus に接続される。バスsa__bus に別個に接続されているのは特殊レジスタ「ヌル」および「ワン」の出力であり（追補６、表Ａ６－１）、特殊レジスタは定数値をストアする（これらのレジスタは図６において「定数レジスタ」と記される）。

【００７４】

バスsa__bus はまた、プログラムメモリ３１４から読まれる命令の即値フィールド「ｉｍｍ」を受取る。

【００７５】

ＡＬＵ３１８の６４ビット出力は６４ビットバスres __bus に接続され、６４ビットバスres __bus はレジスタファイル３１２、データメモリ３１６および特殊レジスタブロック３１５の入力に接続される。

【００７６】

レジスタファイル３１２、データメモリ３１６および特殊レジスタ３１５は追補６に記載される。そこに記載されるように、レジスタおよびデータメモリはタスク間で分けられているため、タスクが再スケジュールされる際にセーブ／リストア動作は必要ではない。特に、特殊レジスタ３１５は各タスクにつき１つずつの１６個のＰＣ（プログラムカウンタ）レジスタを含む。

【００７７】

ロード／ストアユニット（ＬＳＵ）３３０は実行ユニット３１０、サーチマシン１９０および内部メモリ１７０の間でインターフェイスをもたらす。ＬＳＵ３３０は、メモリからレジスタをロードするか、またはレジスタ内容をメモリにストアするためのロードおよびストアリクエストの待ち行列を維持する。ＬＳＵ３３０の入力はres __bus に接続され、またＬＳＵ３３０の６４ビットの出力ｒｆｉはレジスタファイル３１２の入力に接続される。

【００７８】

ＤＭＡブロック３４０の入力はバスres __bus に接続され、実行ユニット３１０がＤＭＡ３４０をプログラムできるようにする。ＤＭＡ３４０はアプレットをプログラムメモリにロードすることができる。

【００７９】

図７には命令実行パイプラインが示される。パイプラインには７つのステージがある。

【００８０】

（１）　タスクセレクト（ＴＳ）ステージｔ０。このステージでは、アクティブタスクがタスク制御３２０によってそれぞれ対応のチャネル１５０．ｘに対して選択される。いくつかの実施例では、タスク制御ブロックは固定優先順位機構を実現しており、タスクＩＧｘ．０が最高優先順位を有し、次にＩＧｘ．１、その次にＥＧｘ．０、そしてその次にＥＧｘ．１となる。

【００８１】

いくつかの実施例では、タスクは一度アクティブにされると、より高い優先順位のタスクがランする用意ができたというだけではサスペンドされない。その優先順位のより低いタスクは利用可能でないリソースにアクセスを試みるまでアクティブのままである。

【００８２】

（２）　フェッチ（Ｆ）ステージｔ１において、タスク制御ブロック３２０は実行ユニット３１０に対してアクティブタスク番号信号task# __t1（追補１の表Ａ１－１におけるtsk __taskNumt1 と同じ）を駆動する。信号task# __t1は特殊レジスタ３１５内の１６個のＰＣ値のうちの１つを選択する。

【００８３】

どのタスクもアクティブではない場合、タスク制御ブロック３２０は実行ユニット３１０に「アイドル」信号をアサートする。この信号は表Ａ１－１において「tsk __idle」と表わされている。「アイドル」がアサートされると、task# __t1は「ドントケア」であり、命令実行ユニット３１０は残りのパイプラインステージにおいてＮＯＰ（ノーオペレーション）命令を実行する。

【００８４】

「アイドル」がデアサートされると、特殊レジスタブロック３１５においてtask# __t1により選択されたＰＣレジスタ値はプログラムメモリ３１４に与えられる。選択されたＰＣにより示される命令がメモリから実行ユニット３１０へ読出される。

【００８５】
（３）　デコード（Ｄ）ステージｔ２において、命令は実行ユニットによりデコードされる。
【００８６】
（４）　読出（Ｒ）ステージｔ３において、レジスタファイル３１２および／または特殊レジスタ３１５および／またはデータメモリ３１６から命令オペランドが読まれ、ＡＬＵ３１８に提示される。
【００８７】
また、このステージにおいて、以下に図８から図１５に関連してより詳しく説明するように、タスク制御３２０は配線４１０（図６）上にサスペンド信号（表Ａ１－１におけるtsk＿susp）を発生する。サスペンド信号がアサートされると、タスクはサスペンドされ、命令実行はアボートされ、タスクのＰＣレジスタは凍結される。後にタスクがアクティブにされると、同じ命令が再実行されることになる。
【００８８】
また、このステージにおいて、実行ユニット３１０はウェイト信号を発生する。ウェイト信号がアサートされると、命令実行は完了させられずＰＣレジスタは凍結するが、タスクはアクティブのままであり、次のクロックサイクルから命令は再び実行される。たとえば、図７の命令１が、サイクル３においてアサートされるウェイト信号によって遅延された場合、同じ命令はサイクル４において開始される命令番号５として再実行されることになる。
【００８９】
ウェイト信号は、命令を妨害している条件が、同じハードウェアタスクが再スケジュールされるころまでにはなくなりそうである場合にアサートされる。ウェイト条件は追補３に記載される。
【００９０】
サスペンド信号およびウェイト信号がデアサートされると、ＰＣレジスタは次の命令を指すように変えられる。
【００９１】
（５）　実行（Ｅ）ステージｔ４において、命令は実行される。
（６）　ライトバック（ＷＢ）ステージｔ５において、実行ステージの結果は行先がレジスタファイル３１２内にある場合を除いて、その行先に書かれる。
【００９２】
（７）　レジスタ書込（ＷＲ）ステージにおいて、実行ステージの結果は必要であればレジスタファイル３１２に書込まれる。
【００９３】
注目すべきなのは、各命令のＷＲステージ（たとえばサイクル６の命令１）が同じハードウェアタスクの次の命令のＲステージの前に起こることである（サイクル７の命令５を参照のこと）。そのため、たとえば命令５が命令１の結果を用いるとすると、その結果は命令５がそれをサイクル７において読む前にレジスタファイルまたは特殊レジスタに書かれることになる。
【００９４】
図７に示したように、（Ｒステージにおいて）命令がアボートされると、パイプラインはすでに開始された他の命令から除去されなくてもよい。なぜなら、これらの命令は他のタスク（さらには他のハードウェアタスク）に属するからである。たとえば、命令１をアボートしなければならない場合、命令１のＲステージにおいて、またはその前に開始された他の命令とは命令２、３および４だけである。これらの命令は他のタスクによって実行されるため、除去されなくてもよい。
【００９５】
所与のハードウェアタスクに関して、対応する４つのソフトウェアタスクの間で切替えをするのに、タスク切替えがオペレーティングシステムソフトウェアにより行なわれる場合

とは異なり、別個の命令を実行することは必要ではない。このため、高いスループットが達成される。

【００９６】

図８には、１つのリクエストＦＩＦＯ２３０またはステータスＦＩＦＯ２４０に関するタスク同期化のバブル図が示される。下方のダイアグラム７０４では、「タスク０」および「タスク１」は図５と同じ意味を有する。より特定的には、これらはリクエストまたはステータスＦＩＦＯを共用する２つのソフトウェアタスクである。いくつかの実施例では、タスク０はイングレスサブチャネルに対するＩＧｉ．０またはイーグレスサブチャネルに対するＥＧｉ．０である。

【００９７】

ダイアグラム７０４はＦＩＦＯの所有権を示す状態機械である。ＲＥＳＥＴにおいて、ＦＩＦＯは状態７１０ＲＳ．０によって示されるようにタスク０により所有される。

【００９８】

タスク０がＦＩＦＯを読むのに成功すると、ＦＩＦＯは状態７１０ＲＳ．１により示されるように、タスク１により所有されることになる。ＦＩＦＯを読むことは図５の「ａ」または「ｃ」スイッチを切替えることに等しい。タスク１がＦＩＦＯを読むのに成功すると、状態機械は状態７１０ＲＳ．０に戻る。

【００９９】

ＦＩＦＯ読出動作は条件mfsel[x] & ffrd により示される。信号mfsel は追補４に記載される。信号ffrdは、マイクロコントローラによりいずれかのリクエストまたはステータスＦＩＦＯが読まれると、ステージｔ３において実行ユニットによりアサートされる。別個のffrdバージョンが各リクエストおよびステータスＦＩＦＯに対して生成される（ＦＩＦＯ読出が成功すると、追補４の信号mfrdがステージｔ５においてアサートされる）。

【０１００】

リクエストおよびステータスＦＩＦＯは１６個ある。これらＦＩＦＯの各々は０から１５までの一意の数「ｘ」により識別される。ＦＩＦＯ「ｘ」が読まれている際、図８のmfsel[x]により示されるように、その数「ｘ」はラインmfsel 上に駆動される。

【０１０１】

ダイアグラム７２０および７４０はタスク０およびタスク１がＦＩＦＯに関連してどのように状態を変化させるかを示している。上に示したように、各タスクには３つの状態、すなわちレディ（「ＲＤＹ」）、アクティブおよびサスペンドがある。ＲＥＳＥＴにおいて、すべてのタスクはレディとなる。タスクはパイプラインステージｔ０において選択されるとアクティブとなる。

【０１０２】

ここで説明する実施例では、タスクは直接アクティブ状態からレディ状態へ移行することができないが、これは他の実施例では可能である。

【０１０３】

ここで説明する実施例では、各タスクは「サスペンド」条件７３０が生ずるとアクティブ状態からサスペンド状態へ移行する。サスペンドされたタスクはリリース条件７３４が生ずるとレディとなる。可能なサスペンド条件を追補１の表Ａ１－２に列挙する。リリース条件は表Ａ１－３に列挙する。

【０１０４】

ダイアグラム７２０では、サスペンド条件７３０は、タスク０がＦＩＦＯが利用可能でないときにＦＩＦＯにアクセスを試みる際に生じる。より特定的には、条件７３０とは以下のとおりである。

【０１０５】

（１）　タスクがパイプラインステージｔ３にあること（実行ユニット３１０により生成される信号「Ｔ３」により示される）。

【０１０６】

（２）　ffrdがアサートされ、ＦＩＦＯ読出動作を示していること。

（３） mfsel がＦＩＦＯ「ｘ」を識別していること。また、
（４） ＦＩＦＯがタスク１により所有されている（状態機械７０４が状態７１０ＲＳ．１にある）か、または信号cfifordy[x] がローであり、ＦＩＦＯ「ｘ」が空であることを示していること。（信号cfifordyは追補４に記載される。この信号は４つ目のサイクルごとにサンプリングされ、サンプリングされる際には有効である。）
ＦＩＦＯがタスク０によって読まれておりそれ以外のどのタスクによっても読まれているのではないことは、タスク０がパイプラインステージｔ３にあることにより確証される。
【０１０７】
タスク１に対する条件７３０（ダイアグラム７４０）も同様である。
ダイアグラム７２０、７４０における条件７３０は表Ａ１－２（追補１）において、各種類のタスク（イングレスタスク０、イングレスタスク１、イーグレスタスク０、イーグレスタスク１）および各種類のＦＩＦＯ（リクエストおよびステータス）について別々に示される。リクエストＦＩＦＯ条件は４つのセクション「イングレスタスク０」、「イングレスタスク１」、「イーグレスタスク０」、「イーグレスタスク１」の各々において条件番号１として列挙されている。すなわち、イングレスタスク０についての条件は、
exe __RfifoRd & mfsel[x] &（Ireqf ｜～cfifordy[x] ）である。
【０１０８】
信号exe __RfifoRd はffrdと同じである。Ireqf はＦＩＦＯがイングレスタスク１により所有されていることを示す。表Ａ１－２のすべての信号はステージｔ３においてサンプリングされるため、「ｔ３」は表中の条件のいくつかでは省いてある。イーグレスタスク０に関して、信号Ereqf はそれぞれ対応のリクエストＦＩＦＯがイーグレスタスク１により所有されていることを示す。すなわち、Ereqf がIreqf と入れ代わっている。タスク制御３２０は各リクエストＦＩＦＯに対して別個の信号Ireqf またはEreqf を発生する。
【０１０９】
追補１において、信号の否定は（～cfifordyのように）信号名称の前の「～」により示されるか、または（イーグレスタスク１に対する条件１のEreqf __のように）信号名称の後に続く下線によって示される。
【０１１０】
ステータスＦＩＦＯに関して、サスペンド条件７３０は表Ａ１－２において２と番号のつけられた条件である。信号exe __SfifoRd はステータスＦＩＦＯに対するffrdバージョンである。ステータスＦＩＦＯを識別する番号は「ｘ」ではなく「ｙ」で示される。
【０１１１】
ダイアグラム７２０におけるリリース条件７３４は、タスク０がＦＩＦＯを所有している（状態機械７０４が状態７１０ＲＳ．０にある）ことと、cfifordy[x] がハイでありＦＩＦＯが空ではないことを示していることである。タスク１に対するリリース条件７３４（ダイアグラム７４０）は同様である。
【０１１２】
リリース条件は追補１の表Ａ１－３に示されている。各リリース条件は表Ａ１－２の同じスロット内のサスペンド条件に対応する。たとえば、表Ａ１－３における「イングレスタスク０」セクションのリリース条件１は、タスクが表Ａ１－２における「イングレスタスク０」セクションのサスペンド条件１によりサスペンドされた場合にそのタスクをレディ状態にリリースする。このように、表Ａ１－３におけるリリース条件１および２は、リクエストおよびステータスＦＩＦＯに対するダイアグラム７２０および７４０でのリリース条件７３４に対応する。
【０１１３】
図９には、サブチャネルコマンドＦＩＦＯ２６０（すなわち２６０Ｉ）に関するイングレスサブチャネルにおけるタスク同期化が示される。下部のダイアグラム８０４には、イングレスコマンドＦＩＦＯのための状態機械が示される。このＦＩＦＯはイングレスタスクおよびイーグレスタスクの双方により所有され得る。ＲＥＳＥＴにおいて、状態機械は状態Ｓ０にある。この状態において、ＦＩＦＯはイングレスタスク０により所有されている

。イングレスタスク０が、（図５のスイッチ「ｂ」を切替えて）ＦＩＦＯをロックすることなく１つのワードをＦＩＦＯに書くと、ＦＩＦＯは状態Ｓ１に移行し、この状態ではＦＩＦＯはイングレスタスク１により所有される。書込動作は信号IcmdFifoWr[x] により示され、ここで「ｘ」はイングレスコマンドＦＩＦＯに書くことができる４つのイングレスおよびイーグレスタスクのうちの１つを特定する。（IcmdFifoWr[x] がステージｔ３において実行ユニットによりアサートされると、対応するmfloadビット（追補４）がステージｔ５においてアサートされる。）信号IcmdFifoWr[x] は、それぞれ対応のタスクがＦＩＦＯに書くたびに適当な「ｘ」に対してアサートされる。

【０１１４】

ロックされていないことは「アンロック」信号により示され、「アンロック」信号は、コマンドＦＩＦＯを書くのに用いられるマイクロコントローラ命令「ＣＭＤ」（追補７）のＬフラグから実行ユニット３１０によって生成される。

【０１１５】

イングレスタスク１が、コマンドＦＩＦＯをロックすることなく（「ｘ」がイングレスタスク１を示しているIcmdFifoWr[x] により示されるように）そのコマンドＦＩＦＯに書くと、状態機械は状態Ｓ０に戻る。

【０１１６】

イングレスタスク０が状態Ｓ０にあるＦＩＦＯに書き、「ロック」信号がアサートされてそのＦＩＦＯをロックすべきであることを示すと、状態機械は状態Ｓ２に移行する。この状態においては、ＦＩＦＯは依然としてイングレスタスク０により所有されている。ロック信号は、マイクロコントローラ命令ＣＭＤ（追補７）におけるＬフラグから実行ユニット３１０により生成される。ＦＩＦＯは、イングレスタスク０が「アンロック」信号がアサートされている状態でＦＩＦＯに書くまで状態Ｓ２に留まる。書かれた時点でＦＩＦＯは状態Ｓ１に移行する。

【０１１７】

同様に、イングレスタスク１が「ロック」がアサートされている状態で状態Ｓ１にあるＦＩＦＯに書くと、ＦＩＦＯは状態Ｓ３に移行する。この状態においてＦＩＦＯは依然としてイングレスタスク１により所有されている。ＦＩＦＯは、イングレスタスク１が「アンロック」がアサートされた状態でＦＩＦＯに書くまで状態Ｓ３に留まる。書かれた時点でＦＩＦＯは状態Ｓ０に移行する。

【０１１８】

状態機械が状態Ｓ０またはＳ１にあり、イーグレスタスクがコマンドＦＩＦＯをロックすることなくそのコマンドＦＩＦＯに書く場合、状態遷移は起こらない。イーグレスタスク０が状態Ｓ０にあるＦＩＦＯをロックしてＦＩＦＯに書くと、ＦＩＦＯは状態Ｓ４に移行する。この状態において、コマンドＦＩＦＯはイーグレスタスク０により所有される。状態機械は、イーグレスタスク０が「アンロック」がアサートされている状態でコマンドＦＩＦＯに書くまで状態Ｓ４に留まる。書いた時点で、状態機械は状態Ｓ０に戻る。

【０１１９】

状態Ｓ５はＳ４に類似しているが、イーグレスタスク１がコマンドＦＩＦＯに書込をしこれを所有することを表わす。

【０１２０】

状態Ｓ６およびＳ７はそれぞれ対応の状態Ｓ４およびＳ５に類似しているが、状態Ｓ６およびＳ７には状態Ｓ０ではなくＳ１から入る。

【０１２１】

ダイアグラム８２０および８４０には、コマンドＦＩＦＯに関するそれぞれ対応のイングレスタスク０および１の状態遷移が示される。サスペンド条件７３０は表Ａ１－２において番号が３の条件である。信号IcmdFifoWr[x] はイングレスタスク０および１に対する条件３におけるexe __IcmdFifoWr[x] と同じである。表Ａ１－２における信号task# __t3はダイアグラム８２０および８４０における「Ｔ３」と同じである。信号ccmdfull[x] は、コマンドＦＩＦＯ「ｘ」が一杯であるという信号である（追補４を参照のこと）。この信

号はステージｔ３において有効である。信号IcmdfOwnedByI0は、コマンドＦＩＦＯがイングレスタスク０により所有されていることを示す（すなわち、状態機械８０４は状態Ｓ０またはＳ２にある）。信号IcmdfOwnedByI1は、コマンドＦＩＦＯがイングレスタスク１により所有されていることを示す（ダイアグラム８０４における状態Ｓ１、Ｓ３）。

【０１２２】

イーグレスタスクに関して、イングレスコマンドＦＩＦＯに書込むことにより生じたサスペンド条件は表Ａ１－２の条件８である。信号IcmdfOwnedByE0は、コマンドＦＩＦＯがイーグレスタスク０により所有されていることを示す（ダイアグラム８０４における状態Ｓ４、Ｓ６）。信号IcmdfOwnedByE1は、コマンドＦＩＦＯがイーグレスタスク１により所有されていることを示す（ダイアグラム８０４における状態Ｓ５、Ｓ３）。

【０１２３】

リリース条件７３４（図９）は表Ａ１－３におけるイングレスタスクに対する条件３である。

【０１２４】

イーグレスコマンドＦＩＦＯに関するイーグレスタスク同期化も同様である。イーグレスＦＩＦＯについては、状態Ｓ４、Ｓ５、Ｓ６、Ｓ７は存在しない。表Ａ１－２およびＡ１－３において、関係のある条件は番号が３である条件である。信号exe __EcmdFifoWrがexe __IcmdFifoWrと置換わり、イーグレスＦＩＦＯへの書込動作を示す。信号Ecmdf1は、ＦＩＦＯがイーグレスタスク１により所有されていることを示す。

【０１２５】

図１０には、ＤＭＡリソースに関するイーグレスタスク同期化が示される。下部のダイアグラム９０４には、ＤＭＡ状態機械が示される。ＲＥＳＥＴにおいて、ＤＭＡはＩＤＬＥである。図１０において「dmaa__wr」により示されるように、イーグレスタスクがＤＭＡアドレス（プログラムメモリ３１４におけるＤＭＡ転送先アドレス）をＤＭＡ３４０（図６）のＤＭＡアドレスレジスタＤＭＡＡ（追補６）へ書くと、そのタスクはＤＭＡ所有者となり、ＤＭＡ３４０がアクティブとなり内部メモリ１７０からＤＭＡ転送を開始する。図１０の例において、ＤＭＡ所有者はイーグレスタスク０である。

【０１２６】

図１０の「last__word」により示されるように、転送が完了すると、ＤＭＡはレディ（「ＲＤＹ」）となる。

【０１２７】

ＤＭＡがレディ状態にあり、ＤＭＡ所有者タスクがＤＭＡアドレスレジスタを読むと（図１０において「dmaa__rd」として示される）、ＤＭＡは実行状態に移行する。ＤＭＡ所有者はＤＭＡレディ状態においてしかアドレスレジスタを読むことが許されない。非所有者タスクはどのＤＭＡ状態においてもＤＭＡアドレスレジスタを読むことが許される。

【０１２８】

ＤＭＡが実行状態にあると、ＤＭＡ所有者タスクはＤＭＡによりロードされるアプレットを実行する。新たなＤＭＡアクセスは許されない。

【０１２９】

ＤＭＡ所有者タスクがリリースコード１１１をＤＭＡＡレジスタ（追補１）のＯＰフィールドに書込むと、ＤＭＡはアイドル状態に戻る。

【０１３０】

ダイアグラム９２０、９３０には、必ずしも同じハードウェアタスクにあるわけではない２つのイーグレスタスク、すなわちタスク０、タスクＮ、に対する状態遷移が示される。条件７３０は表Ａ１－２におけるイーグレスタスクに対する条件７である。表において、exe __dmaaRdは図１０のdmaa__rdと同じであり、exe __dmaaWrはdmaa__wrと同じである。図１０の「dmaa__rd,wr 」は「dmaa__rdまたはdmaa__wr」という意味である。信号exe __dmaaRd、exe __dmaaWrは実行ユニット３１０により生成される。

【０１３１】

このように、ＤＭＡ所有者タスクは、このＤＭＡ所有者タスクが、ＤＭＡがアクティブで

ある間にステージｔ３においてＤＭＡアドレスレジスタを読もうと試みるか、またはＤＭＡアドレスレジスタに書こうと試みる際にサスペンドされる。所有者タスクはＤＭＡがレディとなるとリリースされる。非所有者タスクはＤＭＡがレディである間にステージｔ３においてＤＭＡレジスタに書こうと試みる際にサスペンドされる。非所有者タスクはＤＭＡがアイドルとなるとリリースされる。

【０１３２】

リリース条件７３４は表Ａ１－２においてイーグレスタスク０および１に対する条件７において「clast __word」として示される。

【０１３３】

図１１には、セマフォレジスタｓｅｍｒ（追補２、６）に関するタスク同期化が示される。サスペンド条件７３０は表Ａ１－２において条件５として示される。各サスペンド条件は以下のとおりである。すなわち、（１）タスクがパイプラインステージｔ３にあることと、（２）ＢＩＴＣまたはＢＩＴＣＩ命令がタスクにより実行され、そのターゲットのオペランドがセマフォレジスタであり、その命令が、その命令の以前からセマフォレジスタビットが有していた値と同じ値をセマフォレジスタビットに書こうとしているためアボートされなければならないこと（これは表Ａ１－２の信号exe __bitcSemRegにより示されており、「exe __」で始まる信号名称はすべて実行ユニット３１０により生成される信号を表わしている）とである。サスペンドが生じると、タスク制御ブロック３２０はフラグＳＰｘを１に設定し、ここで「ｘ」はタスク番号（０－１５）である。

【０１３４】

リリース条件７３０はフラグＳＰｘがクリアされる（すなわち０に設定される）ことである。タスク制御ブロック３２０は以下の２つの条件のうちのいずれか１つでも生じるとすべてのフラグＳＰｘをクリアする。すなわち、

（１）　パイプラインステージｔ３において、命令ＢＩＴＣまたはＢＩＴＣＩの実行がどれか他のタスクＹによって成功したこと。この条件は表Ａ１－３におけるリリース条件５の信号exe __bitcSemAccにより示される。

【０１３５】

（２）　チャネル１５０がセマフォレジスタに書くこと。これはcstrobe がアサートされており（追補４における表Ａ４－１）、ｃｓｅｍ［５］が１にあることによって示される。チャネルは、チャネルコマンドによりコマンドされると、セマフォレジスタにアクセスしてマイクロコントローラ１６０に表示を送る。ここに引用により援用される、上記の米国特許出願第０９／０５５，０４４号の代理人事件番号Ｍ－４８５５　ＵＳ「ネットワークでデータを転送するためのネットワークプロセッサシステムおよび方法」を参照されたい。

【０１３６】

図１２には、サーチマシン１９０に関するタスク状態遷移が示される。サスペンド条件７３０（表Ａ１－２における条件４）は以下の条件（１）および（２）の両方が真であることである。すなわち、

（１）　タスクがパイプラインステージＴ３にあり、そのタスクがサーチマシンにコマンドを書くという命令を実行する（表Ａ１－２においてexe __scmdWrとして示される、信号scmd__wr）か、またはサーチマシンから結果を読むという命令を実行すること（表Ａ１－２においてexe __scmdRdとして示される、信号sres__rd）。追補７のマイクロコントローラ命令ＳＭＷＲ（サーチマシンコマンド書込）および追補６のレジスタscmd、scmde の記載を参照されたい。

【０１３７】

（２）　信号task__ownbit[x] が０であること（「ｘ」はタスク番号）により示されるように、サーチマシンリソースがタスクによって利用可能でないこと。この信号は追補１の表Ａ１－１およびＡ１－２ではsm__task__ownbitとして示される。名称が「sm__」で始まる信号はサーチマシン１９０により生成される。サーチマシンリソースおよびサスペンド条件は追補２に記載される。

【０１３８】
リリース条件７３４とはそれぞれ対応のtask__ownbit[x] が１であることである。
【０１３９】
図１３には、メモリ１７０におけるフリーリストのスクラッチバッファ１６１０（図１８および追補５）に関するタスク同期化が示される。サスペンド条件７３０（表Ａ１－２での条件６）は、以下のすべてが真であることである。すなわち、
（１）　タスクがパイプラインステージｔ３にあること。
【０１４０】
（２）　実行ユニットにより生成される信号ifreel__rdにより示されるように、タスクが内部フリーリストレジスタＩＦＲＥＥＬ（追補６）を読んでいること。この信号は表Ａ１－２においてはexu __ifreelRdと表わされる。ＩＦＲＥＥＬレジスタが読まれてフリーバッファ番号が得られる。
【０１４１】
（３）　「no__free__buffers 」（「no__free__buf 」）信号が特殊レジスタブロック３１５によりアサートされ、フリーバッファがないことを示していること。
【０１４２】
リリース条件７３４は、以下の３つの条件のうちのいずれかが真となることである。すなわち、
（１）　cstrobe （追補４における表Ａ４－１）がチャネル１５０によりアサートされ、ｃｓｅｍ［５］が０であり、チャネル１５０が、信号ｃｓｅｍ［４：０］により特定されるスクラッチバッファ１６１０を内部フリーリストに戻そうとしていることを示していること。
【０１４３】
（２）　信号IfreelWr（表Ａ１－３におけるexu __ifreelWr）は実行ユニットによりアサートされ、マイクロコントローラがＩＦＲＥＥＬレジスタ（追補６）へ書いていることを示していること。このレジスタには、フリーとなったスクラッチバッファの番号が書込まれる。
【０１４４】
（３）　信号IfreerWr（exu __ifreerWr）が実行ユニットによりアサートされ、マイクロコントローラがＩＦＲＥＥＲレジスタへ書いていることを示していること。
【０１４５】
図１４はタスク制御ブロック３２０のブロック図である。タスク制御３２０は、４つの同じラッチのブロック１３０４．０、１３０４．１、１３０４．２、１３０４．３を含む。ラッチ１３０４．０はパイプラインステージｔ０（ＴＳ）におけるハードウェアタスクに関する情報をストアする。その情報はラッチ１３０４．１の入力に与えられる。ラッチ１３０４．１はパイプラインステージｔ１におけるハードウェアタスクに関する情報をストアする。同様にラッチ１３０４．２、１３０４．３はそれぞれ対応のステージｔ２、ｔ３におけるハードウェアタスクに関する情報をストアする。ラッチ１３０４．１の出力はラッチ１３０４．２のそれぞれ対応の入力に接続される。ラッチ１３０４．２の出力はラッチ１３０４．３のそれぞれ対応の入力に接続される。ラッチ１３０４．３の出力は、パイプラインステージｔ３におけるソフトウェアタスクがサスペンドされるべきであるかどうかを判定するのに用いられ、また、以下に説明するようにそれぞれ対応のハードウェアタスクに対するソフトウェアタスクの状態を判定するのに用いられる。
【０１４６】
すべてのラッチは同じクロック（図示せず）によりクロックされる。
各ブロック１３０４において、ラッチ１３２０はそれぞれ対応のハードウェアタスク番号ＨＴ＃（上のＣＨＩＤと同じ）をストアする。ラッチ１３２２はハードウェアタスクに対するアクティブなソフトウェアタスク番号ST#=<SN,I/E>をストアする。そのハードウェアタスクに対していずれのタスクもアクティブではない場合、ラッチ１３２２の出力は「ドントケア」である。

【０１４７】
このように、ブロック１３０４．１のラッチ１３２０、１３２２の出力は信号task# __t1を形成し（図６）、ブロック１３０４．２のラッチ１３２０、１３２２の出力は信号task# __t2を形成する。ブロック１３０４．３のラッチ１３２０、１３２２の出力はラッチ回路１３６０の入力に接続され、ラッチ回路１３６０の出力はラッチ回路１３６２の入力に接続される。回路１３６２の出力は信号task# __t5をもたらす（図６）。
【０１４８】
ブロック１３０４．３のラッチ１３２０の出力はブロック１３０４．０のラッチ１３２０の入力に接続される。
【０１４９】
各ブロック１３０４は、４つのソフトウェアタスクＩＧｘ．０（図１４において「Ｉ０」とも示される）、ＩＧｘ．１（「Ｉ１」）、ＥＧｘ．０（「Ｅ０」）およびＥＧｘ．１（「Ｅ１」）の各々につき１つずつの４つのラッチ回路１３３０を含み、ここで「ｘ」はそれぞれ対応のラッチ１３２０にストアされるハードウェアタスク番号である。各ラッチ回路１３３０は２つのラッチ１３３０Ｓ、１３３０Ｃを含み、これらは簡単にするためタスクＥ１に対してのみ示される。回路１３３０Ｓはタスクの状態（すなわちレディ、アクティブまたはサスペンド）をストアする。回路１３３０Ｃはタスクをレディ状態に移行させるのに必要であるリリース条件７３４をストアする。リリース条件は（表Ａ１－３にあるように）１から７まで、または０から６までのインデックスの形でストアされる。各種のタスク（Ｉ０、Ｉ１、Ｅ０、Ｅ１）に対する可能なリリース条件のインデックスは追補１の表Ａ１－３の左の欄に示される。
【０１５０】
ラッチ１３３０Ｃにある情報は、それぞれ対応のラッチ１３３０Ｓにストアされる状態が「サスペンド」である場合にしか意味をもたない。レディおよびアクティブ状態に対しては、ラッチ１３３０Ｃにある情報は「ドントケア」である。
【０１５１】
各ブロック１３０４は６つのラッチ１３５０を含み、６つのラッチ１３５０は対応するハードウェアタスクに対する６つのそれぞれ対応のリクエスト、ステータスおよびコマンドＦＩＦＯの状態をストアする。可能な状態はダイヤグラム７０４（図８）および８０４（図９）に示され、上に説明したとおりである。
【０１５２】
ブロック１３０４．３のラッチ回路１３３０、１３５０の出力は次の状態および条件発生器１３５４に接続される。回路１３５４はタスクとリクエスト、ステータスおよびコマンドＦＩＦＯとの次の状態を生成し、また次のリリース条件値を生成する。これらの状態および条件信号はバス１３５８を介してブロック１３０４．０の回路１３３０、１３５０の入力へ与えられる。
【０１５３】
図１５には回路１３５４がより詳しく示される。回路１３５４において、リソース次ステージ発生器１３８０はブロック１３０４．３のラッチ回路１３５０からリクエスト、ステータスおよびコマンドＦＩＦＯの状態を受取る。発生器１３８０はまた、いずれかのリソース、ステータスおよびコマンドＦＩＦＯの状態遷移を引き起こし得る、ダイヤグラム７０４および８０４に関連して上に説明した信号のすべてを受取る。発生器１３８０はダイヤグラム７０４および８０４に従ってＦＩＦＯの次の状態を計算し、次の状態を同じクロックサイクルｔ３においてラッチブロック１３０４．０のラッチ回路１３５０へ与える。
【０１５４】
各ラッチ回路１３３０の出力はそれぞれ対応の回路１３９０の入力に接続される。簡単にするため、タスクＥ１のための回路１３９０のみが詳しく示される。タスクＥ１に関して、ラッチ１３３０Ｃのリリース条件出力はマルチプレクサ１３９４のセレクト入力に接続される。マルチプレクサ１３９４のデータ入力はタスクＥ１に対する７つの可能なリリース条件７３４を受取る（表Ａ１－３「イーグレスタスク１」のセクション）。マルチプレ

クサ１３９４に入力される各データは、対応するリリース条件が真である場合にはアサートされ条件が偽である場合にはデアサートされる１ビット信号である。

【０１５５】

マルチプレクサ１３９４により選択されるリリース条件信号（すなわち、ブロック１３０４．３のラッチ１３３０Ｃにストアされるリリース条件に対応する信号）はタスク次状態発生器１３９８に与えられる。発生器１３９８はまた、タスクの現在の状態をラッチ１３３０Ｓから受取り、配線４１０上のサスペンド信号を以下に説明するサスペンド論理およびリリース条件発生器１４０１から受取る。タスク次状態発生器１３９８は、タスクがサスペンドされたままであるかどうか、または代わりにタスクが同じクロックサイクル内にアクティブになれるかどうかを示す信号Ａを発生する。信号Ａは以下の表２に従って生成される。

【０１５６】

【表２】

ラッチ 1330S からの状態	MUX 1394 からのリリース条件	リード 410上のサスペンド信号	A
サスペンド	真	ドントケア	レディ
	偽	ドントケア	サスペンド
レディ	ドントケア	ドントケア	レディ
アクティブ	ドントケア	真	サスペンド
		偽	アクティブ

【０１５７】

アービタ１４０３は４つの回路１３９０からＡ出力を受取り、これらからバス１３５８上に以下の信号を発生する。すなわち、（１）ブロック１３０４．０のそれぞれ対応のラッチ１３３０Ｓに対する各タスクの次のステージと、（２）配線１４０４上のアクティブソフトウェアタスク番号ＳＴ＃とである。ソフトウェアタスク番号はブロック１３０４．０のラッチ１３２２へ送られる。

【０１５８】

アービタ１４０３はまた、信号「アイドル」を発生し、これはアサートされるとどのタスクもアクティブではないことを示す（図６も参照のこと）。

【０１５９】

タスクＩ０、Ｉ１、Ｅ０に対する各回路１３９０は、マルチプレクサへのリリース条件入力を、それぞれ対応のタスク（イングレスタスク０、イングレスタスク１またはイーグレスタスク０）に対応する表Ａ１－３のセクションから取っていることを除いて、タスクＥ１のためのタスク次状態発生器１３９８およびマルチプレクサ１３９４と同じである信号Ａ発生論理を含む。

【０１６０】

サスペンド論理およびリリース条件発生器１４０１はブロック１３０４．３のラッチ回路１３５０の出力を受取り、また、サスペンド条件７３０（図８から図１３および追補１の表Ａ１－２）を計算するのに必要である信号のすべて（たとえばcfifordy、mfsel など）を受取る。ブロック１４０１は、ブロック１３０４．３のラッチ１３２２の出力により識別されるアクティブタスクに対するサスペンド条件を計算する。サスペンド論理１４０１は配線４１０上に、タスク次状態発生器１３９８と他の３つの回路１３９０内の同様の発生器とへサスペンド信号をもたらす。

【０１６１】

さらに、サスペンド論理１４０１は各マルチプレクサ１３９４と他の３つのブロック１３９０内の同様のマルチプレクサ（図示せず）とに対するリリース条件データ入力７３４を生成する。リリース条件は表Ａ１－３の式に従って生成される。

【０１６２】

さらに、サスペンド論理１４０１はブロック１３０４．３におけるすべての状態ラッチ１３３０Ｓの状態出力を受取る。各タスクについて、（１）状態出力がアクティブ状態を示し、かつ（２）タスクに対するサスペンド条件のうちの１つが真である場合、サスペンド論理１４０１は、そのタスクをレディにするのに必要なリリース条件のインデックス７３４＿ｉｎを発生する。別個のインデックス７３４＿ｉｎが表Ａ１－３のそれぞれ対応のセクションに従って各タスクに対して生成される。図１５にはタスクＥ１のためだけのインデックス７３４＿ｉｎが示される。

【０１６３】

これ以外のすべての場合において（すなわち、タスクに対する状態出力が「アクティブ」ではないか、または状態出力はアクティブであるがタスクに対するサスペンド条件のいずれも真ではない場合）、タスクに対するリリースインデックス７３４＿ｉｎは「ドントケア」である。

【０１６４】

タスクＥ１に対するリリースインデックス７３４＿ｉｎはマルチプレクサ１４０６の１つのデータ入力に与えられる。マルチプレクサの他方のデータ入力はタスクＥ１のためのブロック１３０４．３のラッチ１３３０Ｃからの条件出力を受取る。セレクト入力はタスクＥ１に対するブロック１３０４．３のラッチ１３３０Ｓの状態出力から「ａｃｔ」ビットを受取る。状態出力は２ビットを有する。ビット「ａｃｔ」は２ビットのうちの１ビットである。ビット「ａｃｔ」は状態が「アクティブ」であるかどうかを示す。「ａｃｔ」がアクティブ状態を示す場合、マルチプレクサ１４０６はリリースインデックス７３４＿ｉｎを選択する。「ａｃｔ」が非アクティブ状態を示す場合、マルチプレクサ１４０６は条件ラッチ１３３０Ｃの出力を選択する。選択された信号はバス１３５８に与えられ、バス１３５８はその信号をブロック１３０４．０におけるタスクＥ１に対するラッチ１３３０Ｃへ供給する。

【０１６５】

同様に、各タスクのための各回路１３９０は以下のものを選択する同様なマルチプレクサ１４０６（図示せず）を含む。すなわち、同様なマルチプレクサ１４０６は（１）それぞれ対応のタスクに対するブロック１３０４．３のラッチ回路１３３０からの出力「ａｃｔ」がアクティブ状態を示す場合には、サスペンド論理１４０１からのそれぞれ対応のタスクに対するリリース条件インデックス７３４＿ｉｎを選択し、また、（２）「ａｃｔ」が非アクティブ状態を示す場合には、それぞれ対応のタスクに対するブロック１３０４．３のラッチ１３３０の条件出力を選択する。選択された条件インデックスはブロック１３０４．０におけるそれぞれ対応のラッチ１３３０の入力に与えられる。

【０１６６】

いくつかの実施例では、１つのタスクがサスペンドされると、タスク特有の値を有するレジスタはセーブされない。特に、各タスクは、タスクＰＣおよびフラグを有するそれ自体のＰＣレジスタを有する（追補６を参照のこと）。さらに、レジスタファイル３１２は８つのバンクに分割される。各バンクは、同じチャネルからのイングレスタスクおよびイーグレスタスクの対の専用のものである。タスク対により実行されるソフトウェアは、その対の間に共通のレジスタがないような態様で書かれる。このため、レジスタファイルレジスタはタスク特有の値をストアするのだが、これらのレジスタをセーブしたりリストアしたりする必要はない。

【０１６７】

ここで説明する実施例はこの発明を限定するものではない。特に、この発明はポートの数によっても、ポートが全二重または半二重であるということによっても、いかなるタイミング、信号、コマンドまたは命令によっても限定されない。いくつかの実施例では、マイ

クロコントローラは、図７のパイプラインまたは何らかの別のパイプラインを有する複数の実行ユニットを含む。いくつかの実施例では、１つ以上のマイクロコントローラが、スーパースカラまたはＶＬＩＷ（very large instruction word ）プロセッサに存在するような複数の実行ユニットを含む。いくつかの実施例では、マイクロコントローラは複数の集積回路で実現されるプロセッサにより置換えられる。ここで用いた術語「タスク」にはプロセスおよびスレッドも含まれる。他の実施例および変更例も、添付の特許請求の範囲に記載されるこの発明の範囲内に含まれる。

【０１６８】

追補１

タスク制御ブロック

【０１６９】

【表３】

表A1-1:タスク制御ブロック信号リスト

番号	信号名	幅	I/O	ﾀｲﾐﾝｸﾞ	機　能
	SM 190 ｲﾝﾀｰﾌｪｲｽ				
1.	tsk_taskNumt2 [3:0]	4	O	t2	ﾃﾞｺｰﾄﾞｽﾃｰｼﾞの間のﾀｽｸ番号
2.	tsk_taskNumt5 [3:0]	4	O	t5	WB ｽﾃｰｼﾞの間のﾀｽｸ番号
3.	sm_task_ownbit [15:0]	16	I	非同期	ﾀｽｸ所有ﾋﾞｯﾄ(1 ﾘｿｰｽ利用可)
	ﾁｬﾈﾙ 150 ｲﾝﾀｰﾌｪｲｽ				
4.	ccmdfull[7:0]	8	I	非同期	ｺﾏﾝﾄﾞ FIFO フル
5.	cfifordy[15:0]	16	I	非同期	ﾘｸｴｽﾄ/ｽﾃｰﾀｽ FIFO ﾚﾃﾞｨ
	実行ﾕﾆｯﾄ ｲﾝﾀｰﾌｪｲｽ				
6.	tsk_susp	1	O	t4	ｻｽﾍﾟﾝﾄﾞ表示
7.	tsk_taskNumt1 [3:0]	4	O	t0	ﾀｽｸ番号
8.	tsk_idle	1	O	t0	ﾌｪｯﾁの間に NOP を挿入する表示
9.	exu_RfifoRd	1	I	t3	ﾘｸｴｽﾄ FIFO 読出
10.	exu_SfifoRd	1	I	t3	ｽﾃｰﾀｽ FIFO 読出
11.	exu_scmdRd	1	I	t3	SM 結果読出
12.	exu_scmdWr	1	I	t3	SM ｺﾏﾝﾄﾞ書込
13.	exu_IcmdFifoWr	1	I	t3	ｲﾝｸﾞﾚｽｺﾏﾝﾄﾞ FIFO 書込
14.	exu_EcmdFifoWr	1	I	t3	ｲｰｸﾞﾚｽｺﾏﾝﾄﾞ FIFO 書込
15.	exu_lock	1	I	t3	ｺﾏﾝﾄﾞ FIFO ﾛｯｸ表示
16.	edma_done	1	I	非同期	DMA 完了表示
17.	edma_busy	1	I	非同期	DMA ﾋﾞｼﾞｰ表示
18.	edma_suspend	1	I	t3	DMA ｻｽﾍﾟﾝﾄﾞ
19.	edma_sel	1	I	t3	DMA ﾘﾘｰｽ選択
20.	efs_flRelease	1	I	非同期	ﾌﾘｰﾘｽﾄﾘﾘｰｽﾌﾗｸﾞ
21.	efs_semRelease	1	I	非同期	ｾﾏﾌｫﾘﾘｰｽﾌﾗｸﾞ
22.	efs_suspend	1	I	t3	ｾﾏﾌｫまたはﾌﾘｰﾘｽﾄｻｽﾍﾟﾝﾄﾞ
23.	efs_sel	1	I	t3	ｾﾏﾌｫまたはﾌﾘｰﾘｽﾄ rel. 選択
24.	tsk_init_doneE0	1	I	非同期	E0 ﾀｽｸ初期化
25.	tsk_init_doneI0I1E1	1	I	非同期	I0、I1、E1 ﾀｽｸ初期化
	LSU ｲﾝﾀｰﾌｪｲｽ				
26.	ts_taskNum2	4	O	t2	ﾃﾞｺｰﾄﾞｽﾃｰｼﾞの間のﾀｽｸ番号

【０１７０】
【表４】

表A1-2:タスクサスペンド条件

番号	ｻｽﾍﾟﾝﾄﾞ条件
	ｲﾝｸﾞﾚｽﾀｽｸ0
1	exe_RfifoRd & mfsel[x] & (Ireqf | ～cififordy[x])
2	exe_SfifoRd & mfsel[y] & (Isttf | -cfifordy[y])
3	exe_IcmdFifoWr[x] & task#_t3 &(ccmdfull[x]| ～CmdOwnedByI0)
4	(exe_scmdRd | exe_scmdWr) & task#_t3 & ～sm_task_ownbit[x]
5	exe_bitcSemRej & task#_t3
6	exu_ifreelRd & no_free_buf
	ｲﾝｸﾞﾚｽﾀｽｸ1
1	exe_RfifoRd & mfsel[x] & (～Ireqf | ～cfifordy[x])
2	exe_SfifoRd & mfsel[y] & (～Isttf | ～cfifordy[y])
3	exe_IcmdFifoWr[x] & task#_t3 &(ccmfull[x] |～ICmdOwnedByI1)
4	(exe_scmdRd | exe_scmdWr0 & task#_t3 & ～sm_task_ownbit[x]
5	exe_bitcSemRej & task#_t3
6	exu_ifreelRd & no_free_buf
	ｲｰｸﾞﾚｽﾀｽｸ0
1	exe_RfifoRd & mfsel[x] & (Ereqf | ～cfifordy[x])
2	exe_SfifoRd & mfsel[y] & (Esttf | ～cfifordy[y])
3	exe_EcmdFifoWr[x] & task#_t3 & (ccmdfull[x] | ECmdf1)
4	(exe_scmdRd | exe_scmdWr) & task#_t3 & ～sm_task_ownbit[x]
5	exe_bitcSemRej & task#_t3
6	exu_ifreelRd & no_free_buf
7	(exe_dmaaRd | exe_dmaaWr) & task#_t3 & ～dma_idle
8	exe_IcmdFifoWr[x] & task#_t3 &(ccmfull[x] |～ICmdOwnedByE0)
	ｲｰｸﾞﾚｽﾀｽｸ1
1	exe_RfifoRd & mfsel[x] & (Ereqf_ | ～cfifordy[x])
2	exe_SFifoRd & mfsel[y] & (Esttf_ | ～cfifordy[y])
3	exe_EcmdFifoWr[x] & task#_t3 & (ccmdfull[x] | ～ECmdf1)
4	(exe_scmdRd | exe_scmdWr) & task#_t3 & ～sm_task_ownbit[x]
5	exe_bitcSemRej & task#_t3
6	exu_ifreelRd & no_free_buf
7	(exe_dmaaRd | exe_dmaaWr) & task#_t3 & ～dma_idle
8	exe_IcmdFifoWr[x] & task#_t3 & (ccmfull[x] | ～ICmdOwnedByE1

【０１７１】
【表５】

表A1-3:タスクリリース条件

番号	リリース条件
	イング レスタスク０
1	Ireqf_ & cfifordy[x]
2	Isttf_ & cfifordy[y]
3	ccmdfull[x]_ & ICmdfOwnedByI0
4	sm_task_ownbit[x]
5	SPx & (exe_bitcSemAcc | (cstrobe & csem[5]))
6	exu_ifreelWr | exu_ifreerWr | (cstrobe & ～csem[5])
	イング レスタスク１
1	Ireqf_ & cfifordy[x]
2	Isttf_ & cfifordy[y]
3	ccmdfull[x]_ & ICmdfOwnedByI1
4	sm_task_ownbit[x]
5	SPx & (exe_bitcSemAcc | (cstrobe & ～csem[5]))
6	exu_ifreelWr | exu_ifreerWr | (cstrobe & -csem[5])
	イーグ レスタスク０
1	Ereqf_ & cfifordy[x]
2	Esttf_ & cfifordy[y]
3	ccmdfull[x]_ & ～ECmdf1
4	sm_task_ownbit[x]
5	SPx & (exe_bitcSemAcc | (cstrobe & csem[5]))
6	exu_ifreelWr | exu_ifreerWr | (cstrobe & ～csem[5])
7	clast_word
8	ccmdfull[x]_ & ～ICmdOwnedByE0
	イーグ レスタスク１
1	Ereqf & cfifordy[x]
2	Esttf & cfifordy[y]
3	ccmffull[x]_ & Ecmdf1
4	sm_task_ownbit[x]
5	SPx & (exe_bitcSemAcc | (cstrobe & csem[5]))
6	exu_ifreelWr | exu_ifreerWr | (cstrobe & ～csem[5])
7	clast_word
8	ccmdfull[x]_ & ICmdOwnedByE1

【０１７２】

追補２

リソース

全リソースは特殊レジスタまたは専用マイクロコントローラコマンドによってアクセスされる。

【０１７３】

サーチマシン

サーチマシンは、マイクロコントローラによって書込まれるコマンドと結果との２個のリソースを有する。

【０１７４】

書込専用のコマンドリソースは１６個ある（タスクごとに１個）。このリソースが利用可でない唯一の場合は、同じタスクからの先のコマンドが完了されていないときである。

【０１７５】

読出専用の結果リソースは１６個ある（タスクごとに１個）。あるコマンドがサーチマシンに通知されると、そのコマンドが実行されるまで結果が利用不可となる。いくつかのコマンド（たとえば、インサートまたはデリート）は結果を持たない。

【０１７６】

チャネル制御

チャネル制御はコマンドＦＩＦＯ２６０、リクエストＦＩＦＯ２３０およびステータスＦＩＦＯ２４０との３種のリソースを有する。

【０１７７】

コマンドリソースは以下の２つの場合利用不可である。

ａ．リソースが別のタスクに属する。この場合、他のタスクがそのリソースをリリースすると、それはこのタスクに対して利用可となる。

【０１７８】

ｂ．コマンドＦＩＦＯがフルである。この場合、コマンドＦＩＦＯがフルでなくなると、タスクはこのリソースを使用し続けることができる。

【０１７９】

コマンドリソースはセッション保護を有する（すなわち、いくつかのコマンドがあるタスクによって書込まれ得るのはリソースが別のタスクに渡される前である）。これは、最初のアクセスの間にリソースをロックし、最後のアクセスの際にそれをアンロックすることによって達成される。コマンドリソースがロックされると、他のどのタスクもこのリソースにアクセスできない。

【０１８０】

チャネル１５０．ｘのイーグレスタスクＥＧｘは、メッセージをスイッチ１２０に送るために同じチャネル１５０．ｘのイングレスコマンドＦＩＦＯ２６０Ｉにコマンドを書込むことができる。イングレスコマンドＦＩＦＯがアンロックされるたびにイーグレスタスクがイングレスコマンドＦＩＦＯに書込まれ得る。イーグレスタスクがその最初のコマンドをイングレスコマンドＦＩＦＯ２６０Ｉに書込むと、イーグレスタスクからの最後のコマンドが書込まれるまでコマンドＦＩＦＯがロックされた状態となる。

【０１８１】

リクエストまたはステータスＦＩＦＯリソースは以下の２つの場合利用可でない。

【０１８２】

ａ．リソースが別のタスクに属する。この場合、他のタスクがＦＩＦＯを読出すと、リソースはこのタスクに対して利用可となる。

【０１８３】

ｂ．ＦＩＦＯが空である。この場合、ＦＩＦＯがレディになると、タスクはこのリソースを使用し続けることができる。

【０１８４】

ＤＭＡ

データＦＩＦＯからのアプレットをプログラムメモリ３１４にダウンロードするのはＤＭＡブロックである。このリソースは、転送前にＤＭＡアドレスをセットし、転送完了時に最後のワードアドレスを読出すイーグレスタスクによって用いられる。転送の間に最後のワードアドレスが読出されると、最後のワードが転送されるまでタスクがサスペンドされる。また、最初の転送が完了していないときに別のイーグレスタスクによって新しいＤＭＡアドレスを書込む試みもタスクサスペンドを引き起こす。

【０１８５】

内部メモリ１７０管理

内部メモリにおけるスクラッチパッドエリア内のフリーバッファ１６１０（図１７）を管理するのは内部メモリ管理である。メモリには３２個のフリーバッファがある。タスクは、次の利用可能なフリーバッファを得ることを望むとき、フリーリスト（ＦｒｅｅＬ）リソース（追補６のレジスタＩＦＲＥＥＬ）にアクセスする。バッファが残されていなければ、このタスクはサスペンドされる。このバッファを用いたチャネルコマンドによってバ

ッファがリリースされるべきだと示されると、バッファはリリースされてフリーリストに戻る。

【０１８６】

セマフォ

セマフォレジスタｓｅｍｒは３２ビットを有する。その各々がマイクロコントローラの即値ビット変更（ＢＩＴＣＩ）およびＢＩＴＣコマンドを用いて直接的にアクセス可能である。セマフォは保護とタスク間の通信とのために用いられる。

【０１８７】

ＢＩＴＣＩまたはＢＩＴＣコマンドが同じ値を現在のビット値としてこのビットに書込もうと試みる場合、それはアボートされ、そのタスクがサスペンドされる。その後、セマフォレジスタが変更される（レジスタにおける何らかのビットが変更される）と、セマフォを待っている全タスクがレディとなり、Ｂｉｔ＿ｃｈａｎｇｅ＿Ｉｍｍｅｄｉａｔｅ（即値ビット変更）コマンドを再び実行しようとする。

【０１８８】

セマフォレジスタのビット３１－２４はＰＩＦ１１０のそれぞれの予め定められた外部ピン（図示せず）を０から１に変更することによってセットされ得る。

【０１８９】

追補３

タスクウェイト条件

ウエイト信号をアサートできる条件は２つある。

（１）　レジスタスコアボード

マイクロコントローラにおけるレジスタごとに、そのステータスを表示する１スコアボードビットがある。このビットがセットされていれば、レジスタはダーティであり、すなわち、データがＬＳＵ３３０によってロードされるのを待っている。起こり得るシナリオは以下のとおりである。

【０１９０】

（ａ）　タスクがＬＳＵによるレジスタのロードをリクエストする。

（ｂ）　タスクがソースとしてのこのレジスタの使用をリクエストする。しかしながら、スコアボードはダーティである。したがって、ウエイト信号がアサートされる。

【０１９１】

（ｃ）　次に、ＬＳＵがレジスタをロードする。

（ｄ）　タスクがソースとしてのこのレジスタの使用を再びリクエストする。今回は使用が許可される。

（２）　ＬＳＵ　ＦＩＦＯフル

これはウエイト信号を発生するための別の条件である。ロード・ストアリクエストを待ち行列に格納しているＬＳＵ　ＦＩＦＯが一旦レディとなると、この条件はクリアされる。

【０１９２】

追補４

以下の表はチャネル／マイクロコントローラインターフェイスにおいて用いられるいくつかの信号をリストにして挙げる。「Ｉ」は信号がチャネルのための入力であることを意味する。「Ｏ」は信号がチャネル出力であることを意味する。

【０１９３】

【表６】

表A4-1

信号名	幅	I/O	機　能
表示			
csem[5:0]	6	O	ｾﾏﾌｫ ID;CSEM[5] = $\overline{\text{SCRATCH}}$ /NOP 表示
cstrobe	1	O	ｾﾏﾌｫｾｯﾄｽﾄﾛｰﾌﾞ
ｺﾏﾝﾄﾞ FIFO			
mfload[7:0]	8	I	CMD FiFo ﾛｰﾄﾞ ｽﾄﾛｰﾌﾞ （<ﾁｬﾈﾙ>、I/E）
ccmdfull[7:0]	8	O	CMD FIFO ﾌﾙ （<ﾁｬﾈﾙ>、I/E）
要求/ｽﾃｰﾀｽ FIFO			
cfifordy[15:0]	16	O	FIFO RDY （ﾚﾃﾞｨ） （<ﾁｬﾈﾙ>、I/E、 $\overline{\text{Req}}$ /ｽﾃｰﾀｽ）
mfsel[3:0]	4	I	FIFO 選択ｱﾄﾞﾚｽ （<ﾁｬﾈﾙ>、I/E、 $\overline{\text{Req}}$ /ｽﾃｰﾀｽ）
mfrd	1	I	FIFO 読出ｽﾄﾛｰﾌﾞ

【０１９４】

追補５

メモリ

内部メモリ１７０のマップ

内部メモリマップを図１６に示す。

【０１９５】

データエリア１５１０（アドレス０００００－１ＦＦＦ　１６進法）

このエリアはスクラッチパッド１６１０、データＦＩＦＯおよびコマンドＦＩＦＯのために用いられる。このエリアは相対アドレスを用いてアクセスされる。データエリアメモリマップを図１７に示す。

【０１９６】

図１７において、「ＤＢＡＳＥ＿Ｉ」はイングレス側のための（後述する）ＣＦＧＲレジスタの「ＤＢＡＳＥ」フィールドである。同様に、ＤＬＥＮ、ＣＢＡＳＥ、ＣＬＥＮは対応のＣＦＧＲレジスタのフィールドである。サフィックス「＿Ｉ」はイングレスを表わし、「＿Ｅ」はイーグレスを表わす。

【０１９７】

各チャネルのための制御エリア１５２０

このエリアにおけるレジスタタイプの１つは以下のとおりである。

【０１９８】

ＣＦＧＲ－チャネルコンフィギュレーションレジスタ（イングレスおよびイーグレス）

各チャネルの方向ごとに１個、８個のＣＦＧＲレジスタがある。それらのフィールドは以下のとおりである。

【０１９９】

ＤＢＡＳＥ　（９ビット）　データバッファベースポインタ（６４バイト整列）
ＤＬＥＮ　（７ビット）　データバッファ長（６４バイト粒度）
ＣＢＡＳＥ　（９ビット）　コマンドバッファベースポインタ（６４バイト整列）
ＣＬＥＮ　（３ビット）　コマンドバッファ長（６４バイト粒度）

ＧＡＰ （４ビット） フレーム制御ワードが無効であるときのデータ読出ポインタとデータ書込ポインタとの間の最小ギャップ（８バイト粒度）。

【０２００】

データエリア１５３０（アドレス４０００－５ＦＦＦ １６進法）

このエリアは上述の米国特許出願第０９／０５５，０４４号の出願人書類番号Ｍ－４８５５ ＵＳに説明される。

【０２０１】

追補６

マイクロコントローラレジスタ

レジスタファイルマップ

レジスタファイル３１２は８個のバンクへと分割される（図１８）。各バンクは同じチャネル１５０．ｘからの１対のイングレスタスクとイーグレスタスクとの専用のものである。いくつかの実施例では、イングレス処理の方がより複雑なのでイングレスタスクがイーグレスタスクよりも多くのレジスタを用いる。また、いくつかの実施例では、タスクソフトウェアは、２つのタスク間に共通のレジスタがないようなものである。

【０２０２】

各レジスタｒ０．０－ｒ７．７は１バイトの幅である。連続する８バイトがレジスタファイルから並列に読出され得る。８バイトレジスタワードの７ビットアドレスを形成するために、レジスタ番号（０から６３）が、それ自体チャネルＩＤ「ＣＨＩＤ」とタスク対番号（０または１）とを連結したものであるバンクＩＤと連結され、アドレスＭＳＢが（特殊レジスタ３１４に対して）レジスタファイル３１２を表示するために０となる。

【０２０３】

マイクロコントローラレジスタマップ

マイクロコントローラにおける全レジスタがマイクロコントローラコマンドによって直接的にアクセス可能である。レジスタマップはレジスタファイル３１２および特殊レジスタ３１５という２領域に分割される。レジスタアドレスは７ビットからなる。特殊レジスタ３１５ではアドレスＭＳＢが１であり、レジスタファイル３１２ではＭＳＢが０である。

【０２０４】

データメモリ３１６

データメモリ３１６（図１９）は変数の一時記憶と後述するいくつかのパラメータとのために用いられる。

【０２０５】

したがって、データメモリ３１６は以下の３領域に分割される。

ａ．タスクごとのタスクレジスタｔｒ０－ｔｒ５（タスクごとに６個）。これらのレジスタはそれぞれのタスク専用である。

【０２０６】

ｂ．チャネルレジスタｃｒ０－ｃｒ３（チャネル１５０．ｘごとに４個）。これらのレジスタはハードウェアタスク専用である。同じチャネルの全タスク（２個のイングレスタスクおよび２個のイーグレスタスク）がこれらのレジスタにアクセスする。

【０２０７】

ｃ．グローバルレジスタｇｒ（１６個のレジスタ）。これらのレジスタは全タスクに対してグローバルである。

【０２０８】

データメモリ３１６は３２ビットの１２８ワードである。

データメモリ３１６のための７ビットアドレス発生方式を図２０に示す。

【０２０９】

ｔｒはタスクレジスタ番号（０－５）である。

ｔｎはタスク番号（０－１５）である（ｔｒおよびｔｎがタスクレジスタアドレスを形成する）。

【０２１０】

ｃｒはチャネルレジスタ番号である（０－３；「１１０」、ｃｒおよびｃｎがチャネルレジスタアドレスを形成する）。

【０２１１】

ｃｎはチャネル番号（０－３）である。

ｇｒはグローバルレジスタ番号（０－１５）である。

【０２１２】

特殊レジスタ（ＳＲ）３１５（以下の表Ａ６－１参照）は（レジスタファイルと同様に）マイクロコントローラコマンドによって直接的にアクセス可能である。特殊レジスタ３１５は以下の３タイプに分割できる。

【０２１３】

ａ．タスクに属するレジスタ、たとえば、プログラムカウンタ（ＰＣ）、タスク番号（ＴＩＮ）等。

【０２１４】

ｂ．リソースレジスタ、たとえば、リクエストＦＩＦＯ（ｒｅｑｆ）、ステータスＦＩＦＯ（ｓｔｔｆ）、サーチマシンコマンド（ｓｃｍｄ）等（追補２参照）。

【０２１５】

ｃ．データメモリ３１６レジスタ、たとえば、タスクレジスタ（ｔｒ）、チャネルレジスタ（ｃｒ）およびグローバルレジスタ（ｇｒ）。

【０２１６】

リソースおよびデータメモリ３１６（タイプｂおよびｃ）はそのアクセスを簡単にするために特殊レジスタにマッピングされる。

【０２１７】

適切な特殊レジスタを以下の表に手短に述べる。

【０２１８】

【表７】

表A6-1:特殊レジスタ

ｱﾄﾞﾚｽ	名	ﾀｲﾌﾟ	ｱｸｾｽ	幅	合計	注　釈
1000_000	null	-	r	32	--	0ﾃﾞｰﾀ
1000_001	one	-	r	32	--	すべて1のﾃﾞｰﾀ
1000_010	pc	a	rw	16	16	ﾌﾟﾛｸﾞﾗﾑｶｳﾝﾀ
1000_011	tn	a	r	4	1	ﾀｽｸ番号
1000_100	ctl	a	rw	16	1	汎用制御ﾚｼﾞｽﾀ
1000_101	dmaa	a	rw	32	1	ﾌﾟﾛｸﾞﾗﾑﾀﾞｳﾝﾛｰﾄﾞｱﾄﾞﾚｽ
1000_110	reqf	b	r	16	8	ﾘｸｴｽﾄfifo
1000_111	sttf	b	r	16	8	ｽﾃｰﾀｽfifo
1001_000	imp	a	rw	10	16	内部ﾒﾓﾘﾎﾟｲﾝﾀ
1001_001	xmp	a	rw	16	16	外部ﾒﾓﾘﾎﾟｲﾝﾀ
1001_100	cmd_i	b	w	64	fifo	ｲﾝｸﾞﾚｽｺﾏﾝﾄﾞ
1001_101	cmd_e	b	w	64	fifo	ｲｰｸﾞﾚｽｺﾏﾝﾄﾞ
1001_110	cmd_il	b	w	64	fifo	ｲﾝｸﾞﾚｽｺﾏﾝﾄﾞ(ﾛｯｸ)
1001_111	cmd_el	b	w	64	fifo	ｲｰｸﾞﾚｽｺﾏﾝﾄﾞ(ﾛｯｸ)
1010_000	scmd	b	rw	64	16	SMｺﾏﾝﾄﾞ/結果
1010_001	scmde	b	rw	64	16	SMｺﾏﾝﾄﾞ/結果拡張
1010_010	xfreel	b	rw	16	4	外部ﾌﾘｰﾘｽﾄ
1010_011	timer	a	rw	50	1	汎用ﾀｲﾏ
1010_100	smcntl	a	rw	17	1	ｻｰﾁﾏｼﾝ制御ﾚｼﾞｽﾀ
1010_101	flcnt	a	r	17	4	外部ﾌﾘｰﾘｽﾄｶｳﾝﾀ
1010_110	agel0	a	r	16	4	ｴｲｼﾞﾘｽﾄ#0の先頭
1010_111	agel1	a	r	16	4	ｴｲｼﾞﾘｽﾄ#1の先頭
1011_000	semr	a	rw	32	1	ｾﾏﾌｫﾚｼﾞｽﾀ
1011_001	ifreel	b	rw	5	1	内部ﾌﾘｰﾘｽﾄ
1011_010	ifreer	b	rw	32	1	内部ﾌﾘｰﾚｼﾞｽﾀ
1011_011	miir	a	rw	32	1	miiﾚｼﾞｽﾀ
1011_100	msqr	a	rw	32	1	ﾒｯｾｰｼﾞﾚｼﾞｽﾀ
1011_110	thrshl0	a	rw	16	4	ｴｲｼﾞしきい値#0
1011_111	thrshl1	a	rw	16	4	ｴｲｼﾞしきい値#1
1100_iii	tr0-5	c	rw	32	96	ﾀｽｸﾚｼﾞｽﾀ
1101_0ii	cr0-3	c	rw	32	16	ﾁｬﾈﾙﾚｼﾞｽﾀ
1101_111	pmdr	a	r	32	1	ﾌﾟﾛｸﾞﾗﾑﾒﾓﾘﾃﾞｰﾀﾚｼﾞｽﾀ
111i_iii	gr0-15	c	rw	32	16	汎用ﾚｼﾞｽﾀ

【０２１９】

いくつかの特殊レジスタのレジスタフィールドは以下のとおりである。

ＰＣ－プログラムカウンタおよびフラグ

ＰＣ　（１０ビット）　プログラムカウンタ

Ｇ　（１ビット）　フラグ－より大きい

Ｌ　（１ビット）　フラグ－より小さい

Ｅ　（１ビット）　フラグ－等しい

Ｃ　（１ビット）　フラグ－桁上げ

Ｇ、Ｌ、ＥおよびＣは読出専用である。

【０２２０】

ＴＮ－タスク番号

ＣＨＩＤ　（２ビット）　チャネルＩｄ

ＳＮ　（１ビット）　シーケンス番号
Ｉ／Ｅ　（１ビット）　イングレス（０）／イーグレス（１）。
【０２２１】
ＳＣＭＤ、ＳＣＭＤＥ－コマンドおよびコマンド拡張
書込動作の間、これらの３２ビットレジスタはサーチマシンのためのコマンドを形成する。読出動作の間、これらのレジスタは結果を与える。ＳＣＭＤＥはＳＣＭＤの前に書込まれるべきである。
【０２２２】
ＸＦＲＥＥＬ－外部フリーリスト
このレジスタへの書込によって、１ブロックが外部メモリ２００におけるフリーリストスタックに付加される。このレジスタからの読出によって、１ブロックがスタックから取除かれ得る。
【０２２３】
１チャネルごとに１個のフリーリストスタックがある。各レジスタはスタックの先頭を指す１６ビットポインタを含む。
【０２２４】
ＴＩＭＥＲ－汎用タイマ
Ｔｉｍｅｒ　（３２ビット）　タイマ値。タイマはシステムクロックが８計時するごとに進む自走カウンタである。
【０２２５】
ＮＸＴＥ　（１６ビット）　エイジングがあるか調べるための次エントリを指すポインタ。このフィールドは書込専用である。リセット後に初期化されるべきである。
【０２２６】
ＥＴ　（１ビット）　タイマ更新イネーブル。このフィールドは書込動作の間に用いられる。ＥＴ＝１であれば、タイマは書込まれた値をとる。ＥＴ＝０であれば、タイマカウンタは書込によって影響を与えられない。
【０２２７】
ＥＮ　（１ビット）　次エントリ更新イネーブル。このフィールドは書込動作の間に用いられる。ＥＮ＝１であれば、次ポインタが新しい値を得る。ＥＮ＝０であれば、ＮＸＴＥフィールドが無効である。
【０２２８】
ＳＭＣＮＴＬ－サーチマシン制御レジスタ
Ｐｏｉｎｔｅｒ　（１６ビット）　ノードエリア開始ポインタ。このポインタはサーチノードエリアを定義する（このエリアの後部が０ｘＦＦＦＦである）。自動エイジング機構はこのエリア内でのみ実行される。
【０２２９】
ＡＧＥ　（１ビット）　エイジングイネーブル（０－ディセーブル；１－イネーブル）。
【０２３０】
ＦＬＣＮＴ－フリーリストカウンタ
この読出専用レジスタはメモリ１７０のスクラッチパッドエリアにおけるフリーリスト内のエントリ数を含む。
【０２３１】
Ｃｏｕｎｔ　（１７ビット）カウンタ（最大値は０ｘ１００００である）。
ＡＧＥＬ０、ＡＧＥＬ１－エイジリスト０、１の先頭
これらは読出専用レジスタである（チャネルごとに２個）。各々がエイジリストの先頭を含む（チャネルごとに２個のエイジリストがある）。これらのレジスタのどの１つからの読出もレジスタをクリアさせる。ここで、ノード（追補８）におけるＴＳＴＭＰ（タイムスタンプ）がこのリストにおけるノードを互いにリンクするために用いられる。レジスタが０なら、リストは空である。
【０２３２】

Ｐｏｉｎｔｅｒ　（１６ビット）　リストポインタの先頭。
しきい値０、しきい値１－しきい値レジスタ
これらのレジスタの各々は対応のエイジリストと関連したしきい値を含む。
【０２３３】
｜current __time-timestamp｜＞しきい値であり、かつエントリがＬＲＮＤ（学習されたエントリ）のタイプであると、エントリはエイジリストに付加される。
【０２３４】
ｔｈｒｅｓｈｏｌｄ　（１６ビット）しきい値。
ＭＳＧＲ－メッセージレジスタはマイクロコントローラとスイッチ１２０ＣＰＵ（図示せず）との間でメッセージを転送するために用いられる。メッセージはヘッダラインによって転送される。
【０２３５】
ＭＳＧＡ　（１６ビット）　ＭＳＧＲに書込むときはＣＰＵへの、レジスタを読出すときはＣＰＵからのメッセージ。このフィールドは読出の後にクリアされる。
【０２３６】
ＭＳＧＢ　（１６ビット）　レジスタを（テストのために）読出すときのＣＰＵへのメッセージ。
【０２３７】
ＤＭＡＡ－ＤＭＡアドレス
ＯＰ　（３ビット）　動作
０００－ｎｏｐ
００１－ＥＰＲＯＭ２０４からのロード
０１０－スイッチ１２０からのロード
１１１－リリース
ＥＰＡ　（１３ビット）　ＥＰＲＯＭ開始アドレス
ＬＥＲ　（１ビット）　ロードエラー
ＰＭＡ　（１０ビット）　プログラムメモリアドレス。
【０２３８】
ＳＥＭＲ－セマフォレジスタ
Ｓ［ｉ］　（１ビット）　セマフォビット「ｉ」。
【０２３９】
ＩＦＲＥＥＲ－内部フリーレジスタ（１６ビット）
Ｆ［ｉ］　（１ビット）　メモリ１７０のスクラッチパッドエリアにおけるブロック「ｉ」がフリーであるかどうかを示す。
【０２４０】
ＩＦＲＥＥＬ－内部フリーリスト
ＢＬＫＮ　（５ビット）　フリーブロック番号（すなわち、スクラッチバッファ番号：図１７参照）。このレジスタの読出によってスクラッチバッファＢＬＫＮがフリーリストから取除かれる。このレジスタへの書込によって、書込まれるＢＬＫＮ値によって特定されるバッファがフリーリストに戻される。
【０２４１】
ＭＩＲ－ＭＩＩ制御レジスタ
このレジスタはＭＩＩ制御インターフェイスによってイーサネットＰＨＹ装置と通信するために用いられる。
【０２４２】
ＢＳＹ　（１ビット）　ビジー。新しいコマンドでセットされ、コマンド完了時にリセットされる。
【０２４３】
ＣＭＤ　（４ビット）　コマンド
１０００－スキャンオン

０００００－スキャンオフ
０１００－制御情報を送る
００１０－ステータス読出
ＮＶ　（１ビット）　有効でない。ＰＨＹからのデータが有効でないときにセットされる。
【０２４４】
ＦＩＡＤ　（５ビット）　ＰＨＹアドレス
ＲＧＡＤ　（５ビット）　レジスタアドレス
Ｄａｔａ　（１６ビット）　データ。
【０２４５】
追補７
マイクロコントローラ命令
３オペランド命令
これらの命令はオペランド＿A とオペランド＿B との間で算術論理演算を行なう。結果はオペランド＿C に書込まれる。命令は以下のとおりである。
【０２４６】
ＡＤＤ－加算
ＳＵＢ－減算
ＯＲ－論理ＯＲ
ＡＮＤ－論理ＡＮＤ
ＸＯＲ－論理ＸＯＲ
ＳＨＬ－左シフト
ＳＨＲ－右シフト
ＢＩＴＣ－ビット変更
命令サイズフィールドがオペランドサイズを特定する。
【０２４７】
命令における２ビット「ｄｔ」フィールド（行先タイプ）が以下のようにオペランド＿C のタイプを特定する。
【０２４８】
ｄｔ＝００－オペランド＿C はレジスタファイル３１２または特殊レジスタ３１５におけるレジスタである。
【０２４９】
ｄｔ＝１０－オペランド＿C はメモリ１７０内にある。オペランド＿C フィールドはアドレス発生のためにロード／ストアユニットにおいて７ビット即値として用いられる。
【０２５０】
ｄｔ＝ｘ１－オペランド＿C は外部メモリ２００内にある。オペランド＿C フィールドはｄｔ［1］ビットとともにアドレス発生のためにロード／ストアユニットにおいて８ビット即値として用いられる。
【０２５１】
ここで、非ゼロｄｔを有する命令はそのオペランドとしてリソースを用いることができない。
【０２５２】
即値バイトでの２オペランド命令
これらの命令はオペランド＿A と即値バイトとの間で算術論理演算を行なう。結果はオペランド＿C に書込まれる。命令は以下のとおりである。
【０２５３】
ＡＤＩ－即値加算
ＳＢＩ－即値減算
ＯＲＩ－即値論理ＯＲ
ＡＮＤＩ－即値論理ＡＮＤ

ＸＯＲＩ－即値論理ＸＯＲ
ＳＨＬＩ－即値左シフト
ＳＨＲＩ－即値右シフト
ＢＩＴＣＩ－即値ビット変更
サイズフィールドがオペランドのサイズを特定する。
【０２５４】
命令の２ビット「ｄｔ」フィールド（行先タイプ）が３オペランド命令におけるようにオペランド＿Ｃ フィールドのタイプを特定する。
【０２５５】
２オペランド命令
これらの命令は２つのオペランドの間で移動演算および比較演算を行なう。命令は以下のとおりである。
【０２５６】
ＭＯＶＥ－オペランドＡをオペランドＣに移動させる
ＣＭＰ－オペランドＣとオペランドＡとを比較する
命令のサイズフィールドがオペランドのサイズを特定する。
【０２５７】
即値での１オペランド命令
これらの命令はオペランドと即値フィールドとの間で移動演算および比較演算を行なう。命令は以下のとおりである。
【０２５８】
ＭＶＩＷ－即値ワードを移動させる
ＭＶＩＢ－即値バイトを移動させる
ＣＰＩＢ－即値バイトを比較する
ＣＰＩＷ－即値ワードを比較する
命令のサイズフィールドがオペランド＿Ｃ のサイズを特定する。
【０２５９】
即値フィールドでの特殊１オペランド命令
これらの命令は以下のようにオペランドＣに対して演算を行なう。
【０２６０】
ＳＭＷＲ－サーチマシン書込
ＣＭＤ－チャネルコマンド書込
ＣＡＳＥ－ケースステートメント
ＢＴＪ－ビットテストおよびジャンプ。
【０２６１】
ロードおよびストア命令
これらの命令はオペランドＡとメモリ１７０または２００との間でロードおよびストア演算を行なう。命令は以下のとおりである。
【０２６２】
ＬＯＡＤ
ＳＴＯＲＥ
「ｄｔ」フィールド（行先タイプ）が以下のように行先のタイプを特定する。
【０２６３】
ｄｔ＝１０－行先がメモリ１７０である。即値フィールドはアドレス発生のためのロード／ストアユニットにおいて７ビット即値として用いられる。
【０２６４】
ｄｔ＝ｘ１－行先がメモリ２００である。即値フィールドはｄｔ「１」ビットとともにアドレス発生のためのロード／ストアユニットにおいて８ビット即値として用いられる。
【０２６５】
特殊即値命令

この命令はＣＭＤＩ（即値コマンド）である。これはコマンドＦＩＦＯに書込むために用いられる。

【０２６６】

選択された命令

ＡＤＤ、ＳＵＢ、ＡＤＩ、ＳＢＩ

フラグ：

Ｅは結果が０に等しいときにセットされる

Ｃは（ＡＤＤ、ＡＤＩのための）キャリーまたは（ＳＵＢ、ＳＢＩのための）ボローが（オペランドｏｐＣのサイズに基づいて）発生されたときにセットされる。

【０２６７】

ＯＲ、ＡＮＤ、ＸＯＲ、ＳＨＬ、ＳＨＲ、ＯＲＩ、ＡＮＤＩ、ＸＯＲＩ、ＳＨＬＩ、ＳＨＲＩ

フラグ：

Ｅは結果が０に等しいときにセットされる。

【０２６８】

ＢＩＴＣ－ビット変更

オペランド：ビット［３１：２５］＝ｏｐＣ、［２４：１８］＝ｏｐＡ、［１７：１６］＝ｄｔ、［１４：８］＝ｏｐＢ、［７］＝ｖ

演算：opC<-opA[opB]<-v（すなわち、ｏｐＣにおけるビット番号ｏｐＢがｖにセットされること以外では、ｏｐＣがｏｐＡの値を受取る）

フラグ：Ｅは（ｏｐＡ［ｏｐＢ］＝＝ｖ）であるときにセットされる。

【０２６９】

ＢＩＴＣＩ－即値ビット変更

オペランド：ビット［３１：２５］＝ｏｐＣ、［２４：１８］＝ｏｐＡ、［１７：１６］＝ｄｔ、［１２：８］＝ｉｍｍ、［７］＝ｖ

演算：opC<-opA[imm]<-v

フラグ：Ｅは（ｏｐＡ［ｉｍｍ］＝＝ｖ）であるときにセットされる。

【０２７０】

ＣＭＰ－比較

オペランド；ビット［３１：２５］＝ｏｐＣ、［２４：１８］＝ｏｐＡ、［７：５］＝オペランドサイズ

演算：ｏｐＣ？ｏｐＡ

フラグ：

Ｅは（ｏｐＣ＝＝ｏｐＡ）であるときにセットされる

Ｇは（ｏｐＣ＞ｏｐＡ）であるときにセットされる

Ｌは（ｏｐＣ＜ｏｐＡ）であるときにセットされる。

【０２７１】

ＣＰＩＷ－即値ワード比較

オペランド：ビット［３１：２５］＝ｏｐＣ、［２３：８］＝ｉｍｍ

演算：ｏｐＣ？ｉｍｍ

フラグ：

Ｅは（ｏｐＣ＝＝ｉｍｍ）であるときにセットされる

Ｇは（ｏｐＣ＞ｉｍｍ）であるときにセットされる

Ｌは（ｏｐＣ＜ｉｍｍ）であるときにセットされる。

【０２７２】

ＣＰＩＢ－即値バイト比較

オペランド：バイト［３１：２５］＝ｏｐＣ、［２３：１６］＝bit __mask、［１５：８］＝ｉｍｍ

演算：（bit __mask & opC）？ｉｍｍ

フラグ：

Ｅは（（bit __mask & opC）＝＝ｉｍｍ）であるときにセットされる
Ｇは（（bit __mask & opC）＞ｉｍｍ）であるときにセットされる
Ｌは（（bit __mask & opC）＜ｉｍｍ）であるときにセットされる。
【０２７３】
ＬＯＡＤ－内部メモリまたは外部メモリからのロード
オペランド：ビット［３１：２５］＝ａｏｐ、［２４：１８］＝ｏｐＡ、［１７：１６］＝ｄｔ、［７］＝ｉ、［６］＝ｆ
演算：
if [dt==10] opA<-IM[{aop,imp }] ; imp=imp+i ;
if [dt==x1] opA<-XM[{aop,xmp }] ; xmp=xmp+i ;
ＩＭは内部メモリ１７０であり、ｉｍｐは内部メモリポインタレジスタ（表Ａ６－１）であり、ＸＭは外部メモリ２００であり、ｘｐｍは外部メモリポインタレジスタ（表Ａ６－１）である。
【０２７４】
ｆビットがセットされると、ロード命令の実行は同じチャネルからの先のストア演算が完了していなければ遅延される。
【０２７５】
ａｏｐはｉｍｐまたはｘｍｐと連結されるアドレスビットである（「｛｝」は連結を示す）。
【０２７６】
ＳＴＯＲＥ－内部メモリまたは外部メモリへのストア
オペランド：ビット[31:25] = aop, [24:18] = opA, [17:16] = dt, [7]= i
演算：
if [dt==10] opA->IM[{aop,imp }] ; imp=imp+i ;
if [dt==x1] opA->XM[{aop,xmp }] ; xmp=xmp+i ;
ＩＭ、ＸＭ、ｉｍｐ、ｘｍｐおよびａｏｐがＬＯＡＤ命令と同じ意味を有する。
【０２７７】
ＳＭＷＲ－サーチマシンコマンド書込
オペランド：ビット[31:25] = opC, [23:8] = imm
演算：scmd<- {opC[63:16], imm } 。
【０２７８】
ＣＭＤＩ－チャネルへの即値コマンド
オペランド：ビット[31:8] = imm, [7] = L, [6] = P
演算：
Command __port <- {40'b0, imm}
ここで４０'ｂ０は４０個の２進零を示す。
【０２７９】
ifＰ＝０, Command __port = cmd__i ; （イングレスコマンド）
ifＰ＝１, Command __port = cmd__e ; （イーグレスコマンド）
命令Ｌフラグ（１ビット）はロック／アンロック制御である（セットされると、命令のロック状態が変更される）。
【０２８０】
ＣＭＤ－チャネルへのコマンド
オペランド：ビット[31:25] = opC, [23:8] = imm, [7] = L, [6] = P
演算：
Command __port <- {opC[63:16],imm}
ifＰ＝０, Command __port = cmd__i ; （イングレスコマンド）
ifＰ＝１, Command __port = cmd__e ; （イーグレスコマンド）
命令における１ビットＬフラグがロック／アンロック制御である（セットされると、ロック状態が変更される）。

【０２８１】
ＣＡＳＥ
オペランド：ビット[31:25] = opC, [23:16] = bit__mask, [12:8] =シフト
演算：PC<-PC+ （（opC&bit __mask）>>シフト）+1。
【０２８２】
ＢＴＪ－ビットテストおよびジャンプ
オペランド：ビット[31:25] = opC, [24:13] = addr, [12:8] = ビット, [7] = v
演算：if（opC[bit]==v ）then PC<-addr 。
【０２８３】
追補８
サーチマシン
サーチマシンは周知のパトリシアツリー構造を用いる（１９９６年８月１３日にG. C. Stone に発行され、引用によりここに援用される米国特許第５，５４６，３９０号「基数判定パケット処理のための方法および装置」（“Method and Apparatus for Radix Decision Packet Processing ”）を参照されたい。
【０２８４】
図２１はツリーノード２４００を示す。各ノードは６４ビットの４ワードの長さである。ノードフォーマットは以下のとおりである。
【０２８５】
【表８】

サーチノードフォーマット

略語	名	ｻｲｽﾞ	説　明
LCP	左の子へのﾎﾟｲﾝﾀ	16	別の基数ﾉｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ
RCP	右の子へのﾎﾟｲﾝﾀ	16	別の基数ﾉｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ
NAP	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾎﾟｲﾝﾀ	6	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞを指すﾎﾟｲﾝﾀ
BIX	ﾋﾞｯﾄｲﾝﾃﾞｯｸｽ	6	この基数ﾉｰﾄﾞがﾃｽﾄしているﾋﾞｯﾄ
FLG	ﾌﾗｸﾞ	1	ﾋﾞｯﾄ 54-LVD-ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞにおいて左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ有効。 0-無効;1-有効
		1	ﾋﾞｯﾄ55-RVD-ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞにおいて右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ有効。 0-無効;1-有効
		1	ﾋﾞｯﾄ56-LUP-左の子へのﾎﾟｲﾝﾀは上向きﾎﾟｲﾝﾀか下向きﾎﾟｲﾝﾀか。 0-下向き;1-上向き
		1	ﾋﾞｯﾄ57-RUP-右の子へのﾎﾟｲﾝﾀは上向きﾎﾟｲﾝﾀか下向きﾎﾟｲﾝﾀか。 0-下向き;1-上向き

【０２８６】
【表９】

略語	名	ｻｲｽﾞ	説　明
TYP	ﾀｲﾌﾟ	6	ﾋﾞｯﾄ61:58-基数ﾉｰﾄﾞのﾀｲﾌﾟを示す。 0000-ﾌﾘｰﾘｽﾄｴﾝﾄﾘ 0001-ｴｲｼﾞﾝｸﾞを考慮に入れないｽﾀﾃｨｯｸｴﾝﾄﾘ 0010-ｴｲｼﾞﾝｸﾞを考慮に入れる学習されたｴﾝﾄﾘ 0011-ﾙｰﾄｴﾝﾄﾘ 0100-合成ｴﾝﾄﾘが実ｷｰを含まない 0101-ﾈｯﾄﾜｰｸｴﾝﾄﾘ 0110-ｺﾝﾌｨｷﾞｭﾚｰｼｮﾝを待っているﾀﾞｰﾃｨｴﾝﾄﾘ 0111-ﾕｰｻﾞ定義ｴﾝﾄﾘ 1000-ｴｲｼﾞﾝｸﾞしたｴﾝﾄﾘ 1001-ﾃﾞﾘｰﾄされたﾙｰﾄｴﾝﾄﾘ ﾋﾞｯﾄ62はﾀｲﾏを特定する。 0-ﾀｲﾏ0;ﾃﾞﾌｫﾙﾄ値 1-ﾀｲﾏ1 63-予約
KEY	ｷｰ	48	異なるｻｰﾁは異なる数のﾋﾞｯﾄを比較する。DA(ｲｰｻﾈｯﾄ行先ｱﾄﾞﾚｽ)は48ﾋﾞｯﾄであり、IPは32ﾋﾞｯﾄであり、SA(ｲｰｻﾈｯﾄｿｰｽｱﾄﾞﾚｽ)は48ﾋﾞｯﾄである。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	このﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
TSTNP	ﾀｲﾑｽﾀﾝﾌﾟ	16	最後にｴﾝﾄﾘが使われた時
ECNT	ｴﾝﾄﾘｶｳﾝﾄ	16	ｴﾝﾄﾘが使われた回数の#ｷ
UNIFO	ﾕｰｻﾞ情報	64	ﾕｰｻﾞ定義ﾌｨｰﾙﾄﾞ。たとえば UINFO[63:60]-状態。 UINFO[59:56]-ﾌﾗｸﾞ。 UNIFO[23:0]-VPI/VCI。ｲﾝｸﾞﾚｽに対する
NRP	次結果ﾎﾟｲﾝﾀ	16	このﾉｰﾄﾞの結果の一部であるｵﾌﾟｼｮﾝの4ﾜｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ。 0x00-ﾇﾙでありこれ以上ﾘﾝｸが存在しないことを意味する。
NTP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰを指すﾎﾟｲﾝﾀ。 階層的ｻｰﾁが可能。 0x00-ﾇﾙでありこれ以上ﾘﾝｸが存在しないことを意味する。

【０２８７】
【表１０】

ルートノードフォーマット

略語	名	ｻｲｽﾞ	説　明
LCP	左の子へのﾎﾟｲﾝﾀ	16	別の基数ﾉｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ
RCP	右の子へのﾎﾟｲﾝﾀ	16	別の基数ﾉｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ
NAP	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾎﾟｲﾝﾀ	16	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞを指すﾎﾟｲﾝﾀ
BIX	ﾋﾞｯﾄｲﾝﾃﾞｯｸｽ	6	この基数ﾉｰﾄﾞがﾃｽﾄしているﾋﾞｯﾄ。 ﾙｰﾄﾉｰﾄﾞに対してBIX=0x2f
FLG	ﾌﾗｸﾞ	1	ﾋﾞｯﾄ54-LVD-ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞにおいて左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ有効。 0-無効;1-有効
		1	ﾋﾞｯﾄ55-RVD-ﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞにおいて右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ有効。 0-無効;1-有効
		1	ﾋﾞｯﾄ56-LUP-左の子へのﾎﾟｲﾝﾀは上向きﾎﾟｲﾝﾀか下向きﾎﾟｲﾝﾀか。 0-下向き;1-上向き
		1	ﾋﾞｯﾄ57-RUP-右の子へのﾎﾟｲﾝﾀは上向きﾎﾟｲﾝﾀか下向きﾎﾟｲﾝﾀか。 0-下向き;1-上向き
TYP	ﾀｲﾌﾟ		ﾋﾞｯﾄ61:58-基数ﾉｰﾄﾞのﾀｲﾌﾟを示す。 ﾀｲﾌﾟﾌｨｰﾙﾄﾞはﾙｰﾄﾉｰﾄﾞに対して0011に設定される。 ｷｰはこの場合暗に示されており、左の子は0x000000のｷｰを見、右の子は0xffffffのｷｰを見る。 ﾋﾞｯﾄ62-0 ﾋﾞｯﾄ63-（予約）。
NTP	次ﾂﾘｰﾎﾟｲﾝﾀ		次ﾂﾘｰﾎﾟｲﾝﾀﾌｨｰﾙﾄﾞはﾂﾘｰ削除ﾌﾟﾛｾｽの間にいくつかのﾙｰﾄをﾘﾝｸするために用いられる。 SM190がそれに書込むので、このﾌｨｰﾙﾄﾞは基数ﾉｰﾄﾞNTPﾌｨｰﾙﾄﾞとは異なる。ﾏｲｸﾛｺﾝﾄﾛｰﾗはﾙｰﾄﾉｰﾄﾞにおいてこのﾌｨｰﾙﾄﾞにｱｸｾｽしない。これはﾂﾘｰをﾃﾞﾘｰﾄする目的のためにのみ用いられる。

【０２８８】
【表１１】

合成ノードフォーマット

略語	名	サイズ	説　明
LCP	左の子へのポインタ	16	別の基数ノードエントリを指すポインタ
RCP	右の子へのポインタ	16	別の基数ノードエントリを指すポインタ
NAP	ネットワークアドレスポインタ	16	ネットワークアドレスノードを指すポインタ
BIX	ビットインデックス	6	この基数ノードがテストしているビット。 ルートノードに対して BIX=0x2f
FLG	フラグ	1	ビット54-LVD-ネットワークアドレスノードにおいて左ネットワークアドレス有効。 0-無効;1-有効
		1	ビット55-RVD-ネットワークアドレスノードにおいて右ネットワークアドレス有効。 0-無効;1-有効
		1	ビット56-LUP-左の子へのポインタは上向きポインタか下向きポインタか。 0-下向き;1-上向き
		1	ビット57-RUP-右の子へのポインタは上向きポインタか下向きポインタか。 0-下向き;1-上向き
TYP	タイプ	6	ビット61:58-基数ノードのタイプを示す。 タイプフィールドは合成エントリに対して0100に設定される。 キーはこの合成エントリにあるネットワークアドレスから引出される。 ビット62-0 ビット63-0（予約）
KEY	キー	48	キーはそれがストアしているネットワークアドレスノードから引出される。
RTP	ルートポインタ	16	このツリーのルートを指すポインタ

【０２８９】
【表１２】

ネットワークアドレスノードフォーマット

略語	名	ｻｲｽﾞ	説　明
LNA	左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	16	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ
NLRP	次左結果ﾎﾟｲﾝﾀ	16	付加的な結果がｽﾄｱされている 4 ﾜｰﾄﾞﾉｰﾄﾞを指すﾎﾟｲﾝﾀ
LMASK	左ﾈｯﾄﾜｰｸﾏｽｸ	16	ﾈｯﾄﾜｰｸﾏｽｸ。1 の連続的なﾏｽｸを想定する。この値は最後の 1 の位置を示す値である。
TYPE	ﾀｲﾌﾟ	6	ﾋﾞｯﾄ61:58-0101 ﾋﾞｯﾄ62-0 ﾋﾞｯﾄ63-0（予約）
LUINFO	左ﾕｰｻﾞ情報	1	左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽのためのﾕｰｻﾞ定義ﾌｨｰﾙﾄﾞ。たとえば VPI/VCI、状態、ﾌﾗｸﾞ等。
RNA	右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	6	ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ
RMASK	右ﾈｯﾄﾜｰｸﾏｽｸ	48	ﾈｯﾄﾜｰｸﾏｽｸ。1の連続的なﾏｽｸを想定する。この値は最後の1の位置を示す。
NRRP	右次結果ﾎﾟｲﾝﾀ	16	付加的な結果がｽﾄｱされている4ﾜｰﾄﾞﾉｰﾄﾞを指すﾎﾟｲﾝﾀ。
RUINFO	右ﾕｰｻﾞ情報	64	右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽのためのﾕｰｻﾞ定義ﾌｨｰﾙﾄﾞ。たとえば、VPI/VCI、状態、ﾌﾗｸﾞ等。

フリーノードフォーマット

略語	名	ｻｲｽﾞ	説　明
TYP	ﾀｲﾌﾟ	6	ﾋﾞｯﾄ61:58-0000 ﾋﾞｯﾄ62-0 ﾋﾞｯﾄ63-0（予約）
NFP	次ﾌﾘｰﾎﾟｲﾝﾀ	16	ﾌﾘｰﾘｽﾄ上の次ｱｲﾃﾑを指すﾎﾟｲﾝﾀ

【０２９０】
【表１３】

サーチマシンコマンド

A.サーチ

略語	名	ｻｲｽﾞ	説　明
OP	Op ｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0000 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-(予約)
FLAGS	ﾌﾗｸﾞ	8	ﾋﾞｯﾄ8-自動学習 ﾋﾞｯﾄ9-自動ｲﾝｸﾘﾒﾝﾄECNT ﾋﾞｯﾄ15:10-予約
KEY	ｻｰﾁｷｰ	48	ｻｰﾁが32ﾋﾞｯﾄｴﾝﾄﾘのためのものであれば、最上位部分が用いられる。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

注:ルートポインタがヌルに等しいサーチが新しいツリーを生む。

ホストアドレス応答

略語	名	ｻｲｽﾞ	説　明
UINFO	ﾕｰｻﾞ情報	64	求められたｴﾝﾄﾘのUINFOﾌｨｰﾙﾄﾞ。 求められなければ、UINFOは0であろう。
NTP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	階層的ｻｰﾁのための次ﾚﾍﾞﾙﾊﾟﾄﾘｼｱﾂﾘｰを指すﾎﾟｲﾝﾀ。
RXP	ｻｰﾁﾉｰﾄﾞﾎﾟｲﾝﾀ	16	そのｷｰに適合したｻｰﾁﾉｰﾄﾞを指すﾎﾟｲﾝﾀ。
NRP	次結果ﾎﾟｲﾝﾀ	16	付加的な4ﾜｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ
ECNT	ｴﾝﾄﾘｶｳﾝﾄ	16	ｴﾝﾄﾘが用いられた回数の#

ネットワークアドレス応答

略語	名	ｻｲｽﾞ	説　明
UINFO	ﾕｰｻﾞ情報	64	求められたｴﾝﾄﾘのUINFOﾌｨｰﾙﾄﾞ。 求められなければ、UINFOは0であろう。
NAP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	適合したﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞを指すﾎﾟｲﾝﾀ。
NRP	次結果ﾎﾟｲﾝﾀ	16	付加的な4ﾜｰﾄﾞｴﾝﾄﾘを指すﾎﾟｲﾝﾀ。
LRF	左/右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	1	0-左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ;1-右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ

【０２９１】

【表１４】

B.インサートホスト

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0001 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-000（予約）。
KEY	ｻｰﾁｷｰ	48	ｻｰﾁが32ﾋﾞｯﾄｴﾝﾄﾘのためのものであれば、最上位部分が用いられる。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
RXP	ｻｰﾁﾉｰﾄﾞﾎﾟｲﾝﾀ	16	規定のｻｰﾁﾉｰﾄﾞを指すﾎﾟｲﾝﾀ

注:ルートポインタがヌルに等しければ、新しいツリーが生まれる。

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
RXP	ｻｰﾁﾉｰﾄﾞﾎﾟｲﾝﾀ	16	規定のｻｰﾁﾉｰﾄﾞを指すﾎﾟｲﾝﾀ

C.インサートネットワークアドレス

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0010 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-000（予約）。
FLAGS	ﾌﾗｸﾞ	8	ﾋﾞｯﾄ13:8-ﾏｽｸﾚﾍﾞﾙ（16から47） ﾋﾞｯﾄ15:14-予約
KEY	ｻｰﾁｷｰ	48	ｻｰﾁｷｰ。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
NAP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	NTWK ｱﾄﾞﾚｽがｲﾝｽﾄｰﾙされたﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞ
LRF	左/右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	1	0-左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ;1-右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ

【０２９２】

【表１５】

D.デリートホスト

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0011 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-000（予約）。
KEY	ｻｰﾁｷｰ	48	ｻｰﾁｷｰ。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
RXP	ｻｰﾁﾉｰﾄﾞﾎﾟｲﾝﾀ	16	ｻｰﾁﾉｰﾄﾞを指すﾎﾟｲﾝﾀ

E.デリートネットワーク

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0100 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-000（予約）。
FLAGS	ﾌﾗｸﾞ	8	ﾋﾞｯﾄ13:8-ﾏｽｸﾚﾍﾞﾙ（16から48） ﾋﾞｯﾄ15:14-予約
KEY	ｻｰﾁｷｰ	48	ｻｰﾁｷｰ
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
NAP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	NTWK ｱﾄﾞﾚｽがｲﾝｽﾄｰﾙされたﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞ
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
LRF	左/右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	1	0-左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ;1-右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ

【０２９３】
【表１６】

F.デリートツリー

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0101 ﾋﾞｯﾄ7:4-0000（予約）。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

G.ファインドネットワーク

略語	名	ｻｲｽﾞ	説　明
OP	Opｺｰﾄﾞ	8	ﾋﾞｯﾄ3:0=0110 ﾋﾞｯﾄ4-ｷｰ長さ 0-32ﾋﾞｯﾄ;1-48ﾋﾞｯﾄ ﾋﾞｯﾄ7:5-000（予約）。
FLAGS	ﾌﾗｸﾞ	8	ﾋﾞｯﾄ13:8-ﾏｽｸﾚﾍﾞﾙ（16から47） ﾋﾞｯﾄ15:14-予約
KEY	ｻｰﾁｷｰ	48	ｻｰﾁｷｰ。
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ

応答

略語	名	ｻｲｽﾞ	説　明
RTP	ﾙｰﾄﾎﾟｲﾝﾀ	16	ﾊﾟﾄﾘｼｱﾂﾘｰのﾙｰﾄを指すﾎﾟｲﾝﾀ
NAP	次ﾂﾘｰﾎﾟｲﾝﾀ	16	NTWK ｱﾄﾞﾚｽがｲﾝｽﾄｰﾙされたﾈｯﾄﾜｰｸｱﾄﾞﾚｽﾉｰﾄﾞ
LRF	左/右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ	1	0-左ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ;1-右ﾈｯﾄﾜｰｸｱﾄﾞﾚｽ

【図面の簡単な説明】

【図１】この発明によるプロセッサを含むシステムのブロック図である。

【図２】図１のシステムにおけるリソースを示すブロック図である。

【図３】図１のシステムにおけるデータフレーム処理を示すタイミング図である。

【図４】図１のシステムにおけるデータフレーム処理を示すタイミング図である。

【図５】図１のシステムにおいて異なるタスクがどのように共用されるリソースにアクセスするのかを示す論理図である。

【図６】図１のシステムにおいて用いられるプロセッサのブロック図である。

【図７】図６のプロセッサの命令実行パイプラインを示す図である。

【図８】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図９】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図１０】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図１１】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図１２】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図１３】図１のシステムにおけるタスクおよびリソース状態遷移を示す図である。

【図１４】図６のプロセッサのタスク制御ブロック回路のブロック図である。

【図１５】図６のプロセッサのタスク制御ブロック回路のブロック図である。

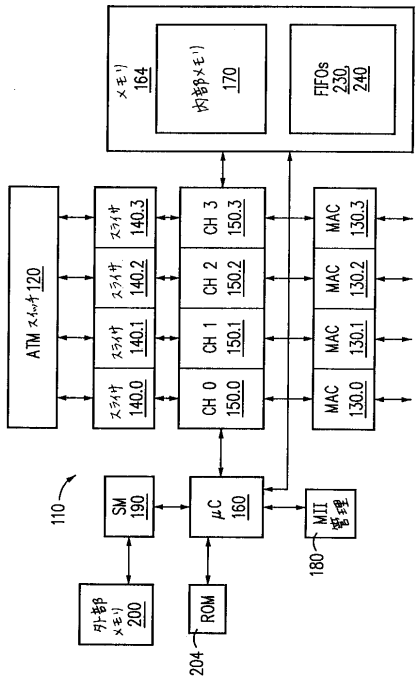
【図１６】図１のシステムのためのメモリマップの図である。

【図１７】図１のシステムのためのデータエリアメモリマップの図である。

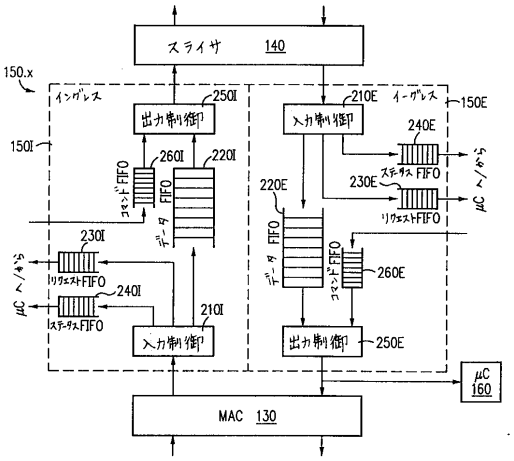
【図１８】図１のプロセッサのためのレジスタファイルマップの図である。

【図１９】図１のプロセッサのためのデータメモリマップの図である。
【図２０】図１９のデータメモリのためのアドレス発生を示す図である。
【図２１】図１のシステムにより用いられるアドレス導出データベースにおけるツリーノードを示す図である。
【符号の説明】
１１０　ポートインターフェイス（ＰＩＦ）回路、１２０　ＡＴＭスイッチ、１３０　ＭＡＣ、１４０　スライサ、１５０　チャネル、１６０　マイクロコントローラ、１７０　内部メモリ、１９０　サーチマシン（ＳＭ）、２００　外部メモリ、２１０　入力制御ブロック、２２０　データＦＩＦＯ、２３０　リクエストＦＩＦＯ、２４０　ステータスＦＩＦＯ、２５０　出力制御、２６０　コマンドＦＩＦＯ、３１０　マイクロコントローラ命令実行ユニット、３１２　レジスタファイル、３１４　マイクロコントローラプログラムメモリ、３１５　特殊レジスタブロック、３１６　データメモリ、３１８　ＡＬＵ、３２０　タスク制御ブロック、３３０　ロード／ストアユニット（ＬＳＵ）、３４０　ＤＭＡブロック、４１０　配線、７０４、７２０、７４０、８０４、８２０、８４０、９２０、９３０　ダイヤグラム、７３０　サスペンド条件、７３４　リリース条件、１３０４、１３２０、１３２２、１３３０、１３５０、１３６０、１３６２　ラッチ、１３５４　次の状態および条件発生器、１３５８　バス、１３８０　リソース次ステージ発生器、１３９０　回路、１３９４、１４０６　マルチプレクサ、１３９８　タスク次状態発生器、１４０１　サスペンド論理およびリリース条件発生器、１４０３　アービタ、１４０４　配線、１６１０　スクラッチバッファ。

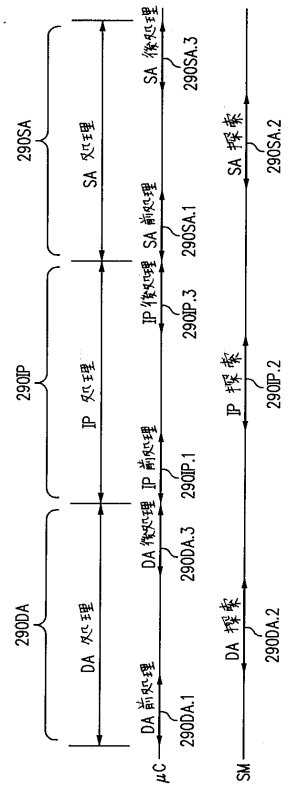
【図１】



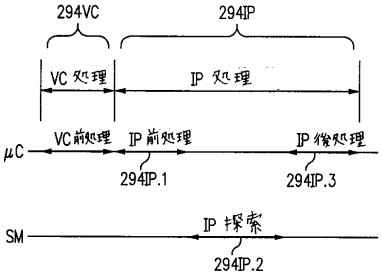
【図２】



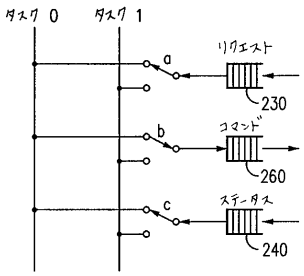
【図３】



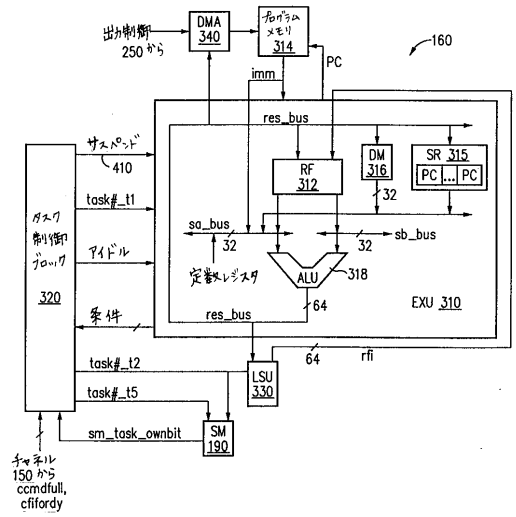
【図４】



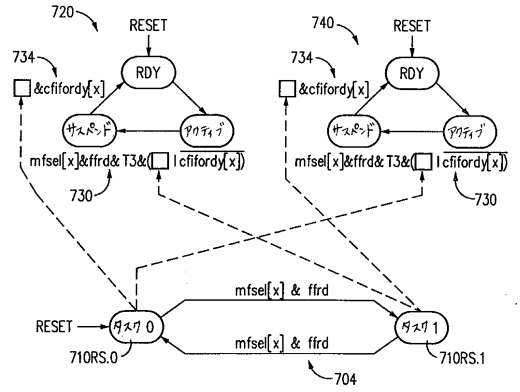
【図５】



【図６】



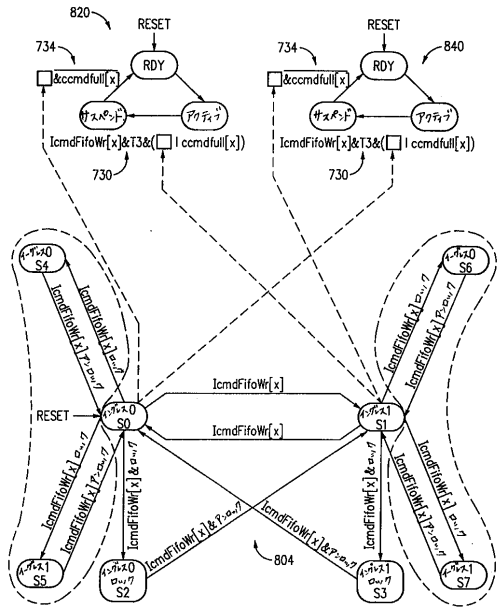
【図８】



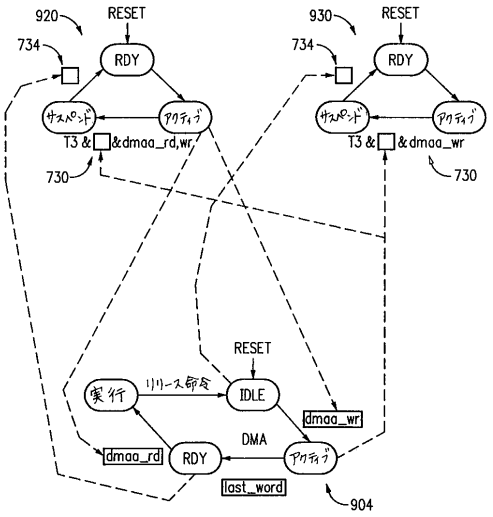
【図７】

命令番号	クロック	0	1	2	3	4	5	6	7	8	9	10
1	HT0	t0 TS	t1 F	t2 D	t3 R(s)	t4 E	t5 WB	t6 WR				
2	HT1		TS	F	D	R(s)	E	WB	WR			
3	HT2			TS	F	D	R(s)	E	WB	WR		
4	HT3				TS	F	D	R(s)	E	WB	WR	
5	HT0					t0 TS	t1 F	t2 D	t3 R(s)	t4 E	t5 WB	t6 WR

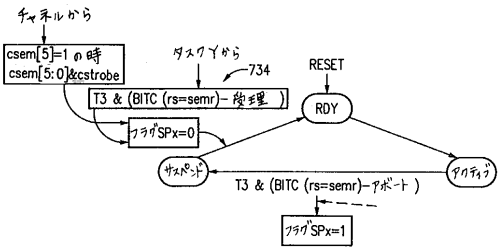
【図９】



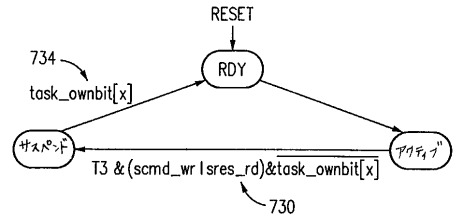
【図１０】



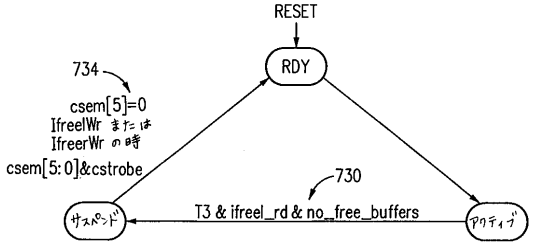
【図１１】



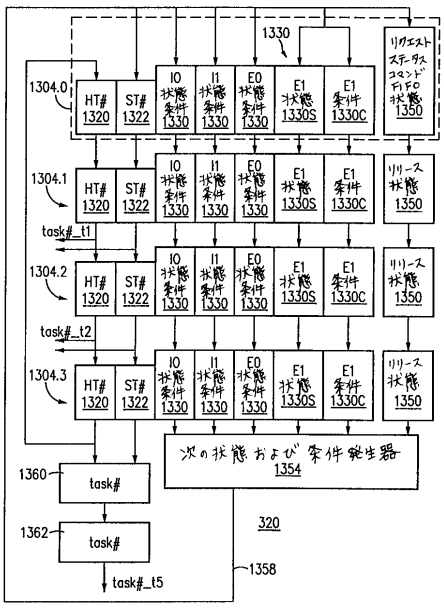
【図１２】



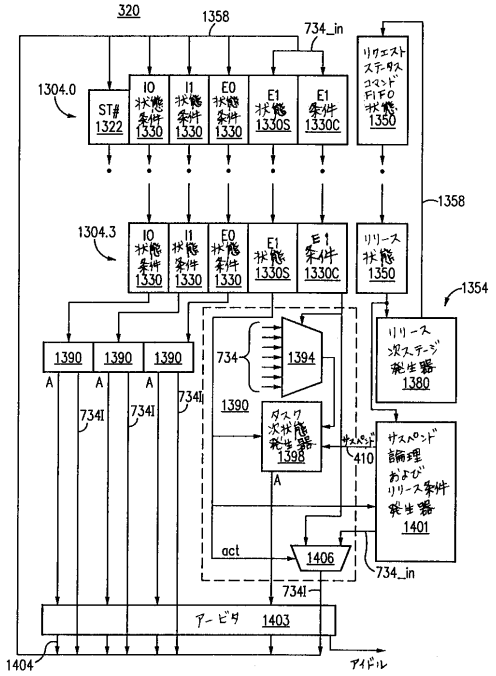
【図１３】



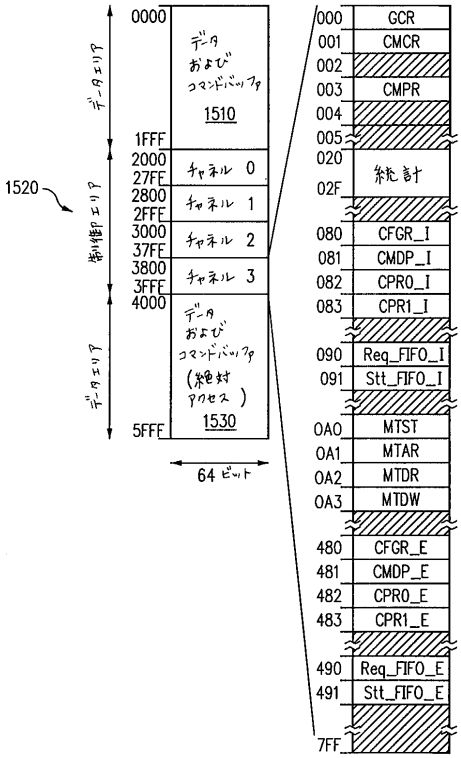
【図１４】



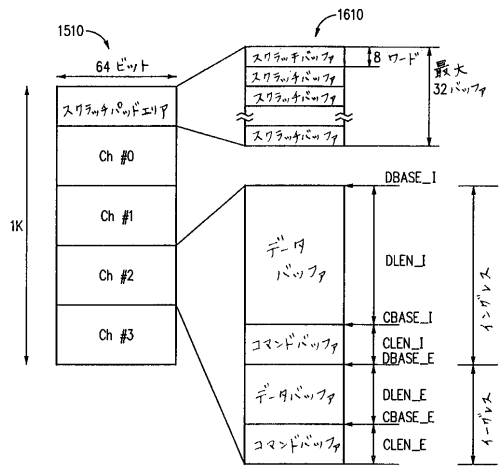
【図１５】



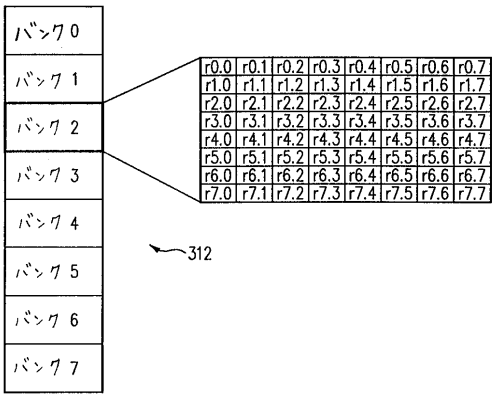
【図１６】



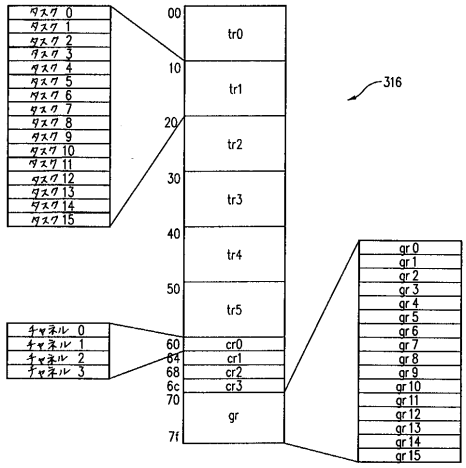
【図１７】



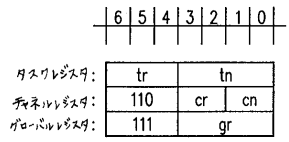
【図１８】



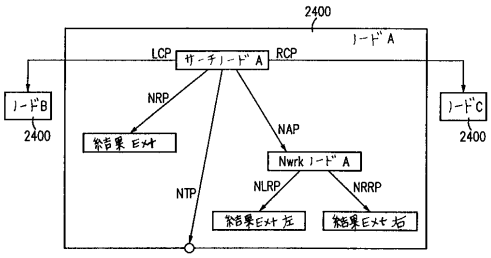
【図１９】


タスク 0
タスク 1
タスク 2
タスク 3
タスク 4
タスク 5
タスク 6
タスク 7
タスク 8
タスク 9
タスク 10
タスク 11
タスク 12
タスク 13
タスク 14
タスク 15
00
tr0
10
tr1
20
tr2
30
tr3
40
tr4
50
tr5
316
チャネル 0
チャネル 1
チャネル 2
チャネル 3
60
cr0
64
cr1
68
cr2
6c
cr3
70
gr
7f
gr 0
gr 1
gr 2
gr 3
gr 4
gr 5
gr 6
gr 7
gr 8
gr 9
gr 10
gr 11
gr 12
gr 13
gr 14
gr 15

【図２０】


6 5 4 3 2 1 0
タスクレジスタ:
tr
tn
チャネルレジスタ:
110
cr
cn
グローバルレジスタ:
111
gr

【図２１】


2400
ノード A
LCP
サーチノード A
RCP
NRP
NAP
NTP
NLRP
NRRP
ノード B
2400
結果 Ext
Nwrk ノード A
結果Ext 左
結果Ext 右
ノード C
2400

フロントページの続き

(74)代理人　100098316
弁理士　野田　久登
(74)代理人　100109162
弁理士　酒井　將行
(72)発明者　アレクサンダー・ヨッフェ
アメリカ合衆国、９４３０６　カリフォルニア州、パロ・アルト、エル・ベラノ・アベニュ、２６０
(72)発明者　ドゥミトゥリー・ビシェツキー
アメリカ合衆国、９５０１４　カリフォルニア州、クペルティーノ、セダー・スプリング・コート、１１６２７

審査官　殿川　雅也

(56)参考文献　特開平０３－１０１５５１（ＪＰ，Ａ）
欧州特許出願公開第００４１３４９０（ＥＰ，Ａ１）
特開平１０－０４９３９０（ＪＰ，Ａ）
欧州特許出願公開第００７７２３２４（ＥＰ，Ａ１）
米国特許第０５５９６５７６（ＵＳ，Ａ）
米国特許第０６１２８２７８（ＵＳ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F　9/46　-　9/54
G06F　9/38

(54)【発明の名称】タスクに各リソースを割当てるための回路、複数のリソースを共用するための方法、命令を実行するためのプロセッサ、マルチタスクプロセッサ、コンピュータ命令を実行するための方法、マルチタスク方法、コンピュータプロセッサを含む装置、複数の所定のグループのタスクを実行するステップを含む方法、ネットワークデータを処理するステップを含む方法、複数のソフトウェアタスクを実行するための方法およびコンピュータプロセッサを含むネットワーク装置