(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-538710
(P2004-538710A)
(43)公表日　平成16年12月24日(2004.12.24)

(51)Int.Cl.[7]	F I		テーマコード（参考）
H04B 1/707	H04J 13/00	D	5K022

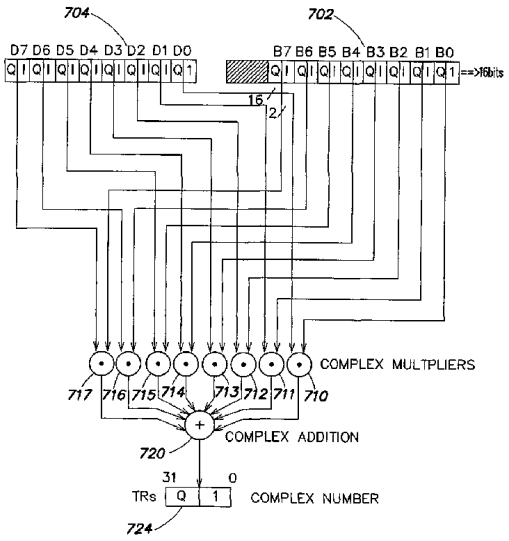
審査請求 未請求　予備審査請求 有　（全 50 頁）

(21)出願番号	特願2003-520104(P2003-520104)
(86)(22)出願日	平成14年8月6日(2002.8.6)
(85)翻訳文提出日	平成16年2月6日(2004.2.6)
(86)国際出願番号	PCT/US2002/024702
(87)国際公開番号	WO2003/015305
(87)国際公開日	平成15年2月20日(2003.2.20)
(31)優先権主張番号	09/925,889
(32)優先日	平成13年8月6日(2001.8.6)
(33)優先権主張国	米国(US)
(81)指定国	EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),CN,JP

(71)出願人　501144003
アナログ・デバイシズ・インコーポレーテッド
アメリカ合衆国マサチューセッツ州ノーウッド，ワン・テクノロジー・ウェイ（番地なし）
(74)代理人　100102842
弁理士　葛和　清司
(72)発明者　リファート，ラセクー
アメリカ合衆国　マサチューセッツ州　０２４４６、ブルックライン、＃１４、ビーコン　ストリート　１４５５
(72)発明者　グリーンフィールド，ズヴィ
イスラエル国　４４４７１　クサーサーヴァ、リヴカ　グーバー　ストリート　５
最終頁に続く

(54)【発明の名称】ディジタル信号プロセッサにおける逆拡散信号回復

(57)【要約】
ディジタル信号プロセッサは、無線電話システムにおける逆拡散復号を実行する。直交符号を使用して、データ信号を結合して、伝送する１つの全体符号化信号とする。この直交符号は、伝送された全体符号化信号から個々のデータ信号を取り出すのに使用される。逆拡散命令は、ディジタル信号プロセッサ機能に含められる。



【特許請求の範囲】
【請求項１】
少なくとも信号値と逆拡散符号を指定する単一の命令に応答して、前記信号値に逆拡散符号を乗じるステップを含む、ディジタル信号プロセッサにおいて信号値を処理する方法。
【請求項２】
乗算の結果を、以前の乗算の結果に加算するステップをさらに含む、請求項１に記載の方法。
【請求項３】
逆拡散符号が、４で割り切れる拡散係数を有する、請求項１に記載の方法。
【請求項４】
逆拡散符号が、符号セグメントに分割されて、各符号セグメントは、１実数ビットと１虚数ビットを含む、２ビット複素符号を含む、請求項１に記載の方法。
【請求項５】
セット符号ビットが、－１の値を表し、クリア符号ビットが＋１の値を表す、請求項４に記載の方法。
【請求項６】
信号値が１６ビットを含む、請求項１に記載の方法。
【請求項７】
信号値が、８実数ビットと８虚数ビットを含む、請求項６に記載の方法。
【請求項８】
少なくとも信号値と１組の符号とを指定する１または２以上の命令に応答して、
前記１組の符号の１つずつについて、前記信号値に前記１組の符号の１つを乗じるステップと、
前記乗算の結果を合計するステップと、
前記合計の結果から得られるデータセットを生成するステップとを含む、ディジタル信号プロセッサにおいてデータセットを計算する方法。
【請求項９】
合計するステップが、乗算の結果を、以前の組の符号を乗算した結果と合計するステップを含む、請求項８に記載の方法。
【請求項１０】
符号の組が、４で割り切れる拡散係数を有する、請求項８に記載の方法。
【請求項１１】
前記１組の符号の１つずつが、１実数ビットと１虚数ビットを含む、２ビット複素符号である、請求項８に記載の方法。
【請求項１２】
セット符号ビットが－１の値を表し、クリア符号ビットが＋１の値を表す、請求項１１に記載の方法。
【請求項１３】
信号値が、１６ビットを含む、請求項８に記載の方法。
【請求項１４】
信号値が、８実数ビットと８虚数ビットを含む、請求項１３に記載の方法。
【請求項１５】
ディジタイル信号計算のための命令およびオペランドを記憶するメモリと、
前記命令の内の選択されたものを、前記メモリからフェッチするための命令アドレスを生成するためのプログラムシーケンサと、
オペランドおよび結果を一時的に記憶するためのレジスタファイル、ならびにデータ信号および符号を指定する復号命令を実行する実行ブロックを含む計算ブロックであって、前記実行ブロックが、データ信号の部分に符号を乗じて、その結果を累算するための複素乗算・累算エンジンを含む、前記計算ブロックとを含む、ディジタル信号プロセッサ。
【請求項１６】

復号命令の実行に応答して、ディジタル信号プロセッサが、
データ信号の部分および符号の部分に対して１組の複素乗算を実行し、かつ
前記複素乗算の結果を合計する、請求項１５に記載のディジタル信号プロセッサ。
【請求項１７】
符号が、４で割り切れる拡散係数を有する、請求項１５に記載の方法。
【請求項１８】
符号が符号セグメントに分割されて、各符号セグメントが、１実数ビットと１虚数ビットを含む、２ビット複素符号を含む、請求項１５に記載の方法。
【請求項１９】
セット符号ビットが－１の値を表し、クリア符号ビットが＋１の値を表す、請求項１８に記載の方法。
【請求項２０】
データ信号が、１６ビットを含む、請求項１５に記載の方法。
【請求項２１】
データ信号が、８実数ビットと８虚数ビットを含む、請求項２０に記載の方法。
【請求項２２】
それぞれが８実数ビットおよび８虚数ビットを含む少なくとも１組の複素第１オペランドと、それぞれが１実数ビットおよび１虚数ビットを含む１組の複素第２オペランドとを指定する、１または２以上の命令に応答して、
第２オペランドの１つずつに対して、第１オペランドの１つと、第２オペランドの１つとの複素乗算を実行するステップと、
前記１組の第２オペランドについて、前記乗算の結果を合計するステップと、
前記合計から得られる１組のデータを、出力として生成するステップとを含む、ディジタル信号プロセッサにおいて出力データを計算する方法。
【請求項２３】
第２オペランドの１つにおけるセットビットが－１の値を表し、第２オペランドの１つにおけるクリアビットが＋１の値を表す、請求項２２に記載の方法。
【請求項２４】
前記１組の複素第２オペランドが、逆拡散符号を含む、請求項２２に記載の方法。
【請求項２５】
前記１組の複素第１オペランドが、入力データ信号を含む、請求項２２に記載の方法。
【請求項２６】
入力データ信号が、音声伝送信号である、請求項２５に記載の方法。
【請求項２７】
命令によって指定される複素信号値と２ビット複素符号セグメントとに応答して、信号値と符号セグメントとの複素乗算を実行して、処理されたデータ値を提供するステップを含む、
ディジタル信号プロセッサにおいて信号値を処理する方法。
【請求項２８】
（ａ）命令によって指定される、１組の複素信号値と対応する組の複素符号セグメントとに応答して、各信号値と対応する符号セグメントとの複素乗算を実行して１組の中間値を提供するステップと、
（ｂ）前記中間値の複素加算を実行して、処理された信号値を提供するステップとを含む、ディジタル信号プロセッサにおいて信号値を処理する方法。
【請求項２９】
複数の組の複素信号値に対して、ステップ（ａ）および（ｂ）を反復して、処理された信号値のストリームを提供する、請求項２８に記載の方法。
【請求項３０】
複素符号セグメントのそれぞれが、２ビット複素符号である、請求項２８に記載の方法。
【発明の詳細な説明】

【０００１】
分野
本発明は、無線電話応用のためのディジタル信号プロセッサに関し、より具体的には準直交符号（near orthogonal codes）を使用する信号成分の分離命令に関する。
【０００２】
背景
セルラー電話システムを実現する場合には、利用可能な周波数において伝送できるデータ量を最大化することが重要である。これには、より大量のデータの伝送を可能にするための、データ信号の変更を必要とすることが多い。このような変更は、受信端において原信号が十分に回復できるように、可逆でなくてはならない。データ変更の１つの方法は、符号分割多重アクセス（ＣＤＭＡ）と拡散によるものである。
【０００３】
拡散とは、通常は狭帯域の信号を、非常に広帯域の信号に変換するデータ変更方法である。信号を広帯域に拡散させると、より小さな電力で信号を送信し、かつ受信することができる。拡散は、信号を拡張し、かつ複製する符号を原信号に乗じることによって達成される。広い周波数帯域にわたって信号が複製される回数は、拡散係数と呼ばれる。例えば、拡散符号（１，１，１，１）は拡散係数（spreading factor）が４であり、この理由は、符号に信号を乗じるときに、各信号部分（「記号」と呼ばれる）が４回反復されることによる。上記の拡散符号の例では、記号「ａｂｃｄ」からなる信号を乗じると、拡散信号（ａａａａｂｂｂｂｃｃｃｃｄｄｄｄ）が得られる。同じ記号を拡散するのに発行される拡散符号が（１，－１，１，－１）の場合には、拡散信号は
【数１】
“$a\bar{a}aa\bar{b}\bar{b}\bar{b}bc\bar{c}c\bar{c}d\bar{d}d\bar{d}$”
となり、ここで
【数２】
$\bar{x}$
は原記号ｘの負の値を示す。拡散が行われると、広帯域信号が受信機に伝送される。この信号は、受信端で逆拡散されて、原信号を回復する。信号を拡散するのに使用される符号は、その信号を逆拡散するのにも使用される。
【０００４】
逆拡散関数に集中的な計算が必要なことを含み、いくつかの要因のために、受信機には、ＡＳＩＣの使用およびディジタル信号プロセッサ内の最も基礎的なコンポーネントを有するハードウエアブロックの使用を含む、逆拡散装置が実装されている。これらの実装には、プログラミングができないという欠点がある。その他の逆拡散装置には、ＦＰＧＡが使用されているが、これらは、コストの理由、およびプログラミングにおける柔軟性がないという理由で、望ましくはない。
【０００５】
ディジタル信号プロセッサは、ソフトウエアのプログラムミングが可能であると理由から、ＡＳＩＣよりも望ましい解決策である。しかしながら、ディジタル信号プロセッサでは、満足な逆拡散装置を実現するために必要な、高速の複素計算を扱うのは困難である。
したがって、伝送された無線電話信号の復号に使用する逆拡散装置の実装を改善することに対する要望がある。
【０００６】
概要
ディジタル信号プロセッサにおいて、信号値を処理する方法が提供され、この方法は、少なくとも信号値と逆拡散符号を指定する単一の命令に応答して、前記信号値に逆拡散符号を乗じるステップを含む。この方法は、乗算の結果を、以前の乗算の結果に加算するステップをさらに含むことができる。逆拡散符号は、４で割り切れる拡散係数を有することが

できる。逆拡散符号は、符号セグメントに分割されて、各符号セグメントは、１実数ビットと１虚数ビットを含む２ビット複素符号を含む。符号ビットに対して、セット符号ビットは、－１の値を表し、クリア符号ビットは、＋１の値を表してもよい。信号値は、８実数ビットおよび８虚数ビットを含む、１６ビットを含むことができる。

【０００７】

ディジタル信号プロセッサにおいてデータセットを計算するためにさらなる方法が提供され、この方法は、少なくとも信号値と１組の符号とを指定する１または２以上の命令に応答して、前記１組の符号の１つずつについて、前記信号値に前記１組の符号の１つを乗じるステップと、前記乗算の結果を合計するステップと、前記合計の結果から得られるデータセットを生成するステップとを含む。この合計のステップには、乗算の結果を、以前の組の符号を乗算した結果と合計することを含めることができる。符号の組は、４で割り切れる拡散係数を有することができる。前記符号の組の１つずつは、１実数ビットおよび１虚数ビットを含む、２ビット複素符号とすることができる。セット符号ビットは、－１の値を表し、クリア符号ビットが＋１の値を表わすことができる。信号値は、８実数ビットと８虚数ビットを含む、１６ビットを含むことができる。

【０００８】

別の一実施態様においては、ディジタル信号プロセッサが提供され、このディジタル信号プロセッサは、ディジタル信号計算のための命令およびオペランドを記憶するメモリと、前記命令の内の選択されたものを前記メモリからフェッチするための、命令アドレスを生成するプログラムシーケンサと、オペランドおよび結果を一時的に記憶するためのレジスタファイル、ならびにデータ信号および符号を指定する復号命令を実行する実行ブロックを含む計算ブロックであって、前期実行ブロックが、データ信号の部分に符号を乗じて、その結果を累算するための複素乗算・累算エンジンを含む、前記計算ブロックとを含む。ディジタル信号プロセッサは、ディジタル信号プロセッサによる復号命令の実行に応答して、データ信号の部分および符号の部分に対して１組の複素乗算を実行し、かつ前記複素乗算の結果を合計することができる。符号は、４で割り切れる拡散係数を有することができる。さらに別の実施態様においては、符号は符号セグメントに分割され、各符号セグメントは、１実数ビットと１虚数ビットを含む、２ビット複素符号を含む。さらに別の実施態様においては、セット符号ビットは、－１の値を表し、クリア符号ビットは＋１の値を表す。データ信号は、８実数ビットおよび８虚数ビットを含む、１６ビットを含むことができる。

【０００９】

別の実施態様においては、ディジタル信号プロセッサにおいて出力データを計算する方法が提供され、この方法は、それぞれが８実数ビットおよび８虚数ビットを含む少なくとも１組の複素第１オペランドと、それぞれが１実数ビットおよび１虚数ビットを含む１組の複素第２オペランドとを指定する、１または２以上の命令に応答して、第２オペランドの１つずつについて、第１オペランドの１つと、第２オペランドの１つとの複素乗算を実行するステップと、前記１組の第２オペランドについて、前記乗算の結果を合計するステップと、前記合計の結果として得られる１組のデータを、出力として生成するステップとを含む。さらに別の実施態様においては、第２オペランドの１つにおけるセットビットは、－１の値を表し、第２オペランドの１つにおけるクリアビットは＋１の値を表す。さらに別の実施態様においては、前記１組の複素第２オペランドが、逆拡散符号を含む。さらに別の実施態様においては、前記１組の複素第１オペランドは、入力データ信号を含む。さらに別の実施態様においては、前記入力データ信号は、音声伝送信号である。

【００１０】

ディジタル信号プロセッサにおいて信号値を処理する方法がさらに提供され、この方法は、命令によって指定される複素信号値と２ビット複素符号セグメントとに応答して、信号値と符号セグメントとの複素乗算を実行して、処理されたデータ値を提供するステップを含む。

ディジタル信号プロセッサにおいて信号値を処理する方法がさらに提供され、この方法は

、（ａ）命令によって指定される、１組の複素信号値と対応する組の複素符号セグメントに応答して、各信号値と対応する符号セグメントとの複素乗算を実行して１組の中間値を提供するステップと、（ｂ）前記中間値の複素加算を実行して、処理された信号値を提供するステップとを含む。この方法は、複数の組の複素信号値に対してステップ（ａ）および（ｂ）を反復して、処理された信号値のストリームを提供するステップをさらに含むことができる。この方法において、複素符号セグメントのそれぞれは、２ビット複素符号とすることができる。

本発明をさらによく理解するために、参照により本明細書に組み入れた、添付の図面を参照する。

【００１１】

詳細な説明

説明の目的で、以下に記述する技法は、セルラー電話信号の伝送という状況において記述される。しかしながら、記述される技法の多くは、セル電話に有用でありながら、その他の高速データ伝送用途にも有用である。

無線電話基地局信号チェインの例のブロック図を図１に示してある。この信号チェインは、記号レート処理１０およびチップレート処理１２を含む。記号レート処理１０をディジタル信号プロセッサに組み込むのが望ましい。送信側では、記号レート処理１０は、ＣＲＣアタッチメントブロック２０、チャネル符号化ブロック２２、レートマッチングブロック２４およびインターリーブブロック２６を含む。受信側では、記号レート処理１０は、逆インターリーブブロック３０、レート決定ブロック３２、チャネル復号ブロック３４およびＣＲＣアタッチメントブロック３６を含む。

【００１２】

チップレート処理１２は、送信側の拡散および変調ブロック４０と、受信側のレーク受信装置４２を含む。逆拡散を実行する、ディジタル信号プロセッサを、受信側に使用することができる。

本発明の実施特徴に適するディジタル信号プロセッサ（ＤＳＰ）１１０の例の、ブロック図を図２に示す。ＤＳＰ１１０の主要構成要素は、計算ブロック１１２および１１４、メモリ１１６、制御ブロック１２４、リンクポートバッファ１２６、外部ポート１２８、ＤＲＡＭコントローラ１３０、命令アライメントバッファ（ＩＡＢ）１３２および１次命令復号器１３４である。計算ブロック１１２および１１４、命令アライメントバッファ１３２、１次命令復号器１３４および制御ブロック１２４は、ＤＳＰ１１０の主要な計算およびデータ処理機能を実行するコアプロセッサを構成する。外部ポート１２８は、外部アドレスバス１５８および外部データバス１６８を介しての外部通信を制御する。リンクポートバッファ１２６は、通信ポート１３６を介する外部通信を制御する。ＤＳＰは好ましくは、単一のモノリシック集積回路として構成される。

【００１３】

メモリ１１６には、３つの独立した大容量メモリバンク１４０、１４２および１４４を含めることができる。好ましい一実施態様においては、それぞれのメモリバンク１４０、１４２および１４４は、それぞれ３２ビットの６４Ｋワードの容量を有する。以下に述べるように、それぞれのメモリバンク１４０、１４２および１４４は、好ましくは１２８ビットデータバスを有する。１クロックサイクルで、それぞれが３２ビットの、最大４つの連続して整列されたデータワードを、各メモリバンクに、またはそこから転送することができる。

【００１４】

ＤＳＰ１１０の要素は、効率の良い高速動作のために、バスによって相互接続されている。バスのそれぞれは、バイナリ情報を並列転送するための多重ラインを含む。第１のアドレスバス１５０（ＭＡ０）は、メモリバンク１４０（Ｍ０）と制御ブロック１２４を相互接続する。第２アドレスバスバス１５２（ＭＡ１）は、メモリバンク１４２（Ｍ１）と制御ブロック１２４を相互接続する。第３のアドレスバス１５４（ＭＡ２）は、メモリバンク１４４（Ｍ２）と制御ブロック１２４を相互接続する。アドレスバス１５０、１５２お

よび１５４のそれぞれは、好ましくは１６ビット幅である。外部アドレスバス１５６（ＭＡＥ）は、外部ポート１２８と制御ブロック１２４を相互接続する。外部アドレスバス１５６は、外部ポート１２８を介して、外部アドレスバス１５８に相互接続されている。外部アドレスバス１５６および１５８のそれぞれは、好ましくは３２ビット幅である。第１のデータバス１６０（ＭＤ０）は、メモリバンク１４０、計算ブロック１１２および１１４、制御ブロック１２４、リンクポートバッファ１２６、ＩＡＢ１３２および外部ポート１２８を相互接続する。第２のデータバス１６２（ＭＤ１）は、メモリバンク１４２、計算ブロック１１２および１１４、制御ブロック１２４、リンクポートバッファ１２６、ＩＡＢ１３２および外部ポート１２８を相互接続する。第３のデータバス１６４（ＭＤ２）は、メモリバンク１４４、計算ブロック１１２および１１４、制御ブロック１２４、リンクポートバッファ１２６、ＩＡＢ１３２および外部ポート１２８を相互接続する。データバス１６０、１６２および１６４は、外部ポート１２８を介して、外部データバス１６８に接続されている。データバス１６０、１６２および１６４のそれぞれは、好ましくは１２８ビット幅であり、外部データバス１６８は好ましくは６４ビット幅である。

【００１５】

第１のアドレスバス１５０および第１のデータバス１６０は、メモリバンク１４０へ、およびそこからのデータ転送用のバスを含む。第２のアドレスバス１５２および第２のデータバス１６２は、メモリバンク１４２へ、およびそこからのテータ転送用の第２のバスを含む。第３のアドレスバス１５４および第３のデータバス１６４は、メモリバンク１４４への、およびそこからのデータ転送用の第３のバスを含む。メモリバンク１４０、１４２および１４４のそれぞれが、別個のバスを有するので、メモリバンク１４０、１４２および１４４は、同時にアクセスが可能である。本明細書における使用では、「データ」とは、ＤＳＰ１１０の動作と関連する、命令またはオペランドを表すバイナリワードを意味する。

【００１６】

通常の動作モードにおいて、プログラム命令はメモリバンクの１つに記憶されており、オペランドは他の２つのメモリバンクに記憶されている。したがって、１クロックサイクルに、少なくとも１つの命令と２つのオペランドを計算ブロック１１２および１１４に供給することができる。メモリバンク１４０、１４２および１４４のそれぞれは、単一クロックサイクルでの複数のデータワードの読み取り、および書き込みを可能にするように構成することができる。単一クロックサイクルで、各メモリバンクから複数のデータワードを同時転送することが、命令キャッシュまたはデータキャッシュを必要とすることなく達成される。

【００１７】

上記のように、それぞれのメモリバンク１４０、１４２および１４４は、それぞれが３２ビットの、６４Ｋワードの容量を有するのが好ましい。各メモリバンクは、１２８ビット幅のデータバスに接続することができる。別の一実施態様においては、各データバスは、６４ビット幅としてもよく、クロックフェーズ１およびクロックフェーズ２のそれぞれにおいて、６４ビットを転送して、これによって実効的に１２８ビットのバス幅を提供する。各メモリバンクにおいて、単一クロックサイクルで複数データワードにアクセスすることができる。具体的には、データは、それぞれが３２ビットのシングル、デュアルまたはクワッドワードとしてアクセスすることができる。

【００１８】

クワッドワード転送を使用すると、１クロックサイクルで、それぞれが３２ビットの４つの命令および８つのオペランドを、計算ブロック１１２および１１４に供給することができる。転送されるデータワード数、およびデータワードが転送される計算ブロック（単数または複数）は、命令中の制御ビットによって選択される。シングル、デュアルまたはクワッドデータワードを計算ブロック１１２、計算ブロック１１４、またはその両方に転送することができる。デュアルおよびクワッドデータワードアクセスによって、単一クロックサイクルに、いくつかのオペランドを計算ブロック１１２および１１４に転送すること

を可能にすることによって、多くの用途でＤＳＰ１１０の性能が改善される。各クロックサイクルにおいて複数の命令にアクセスできる能力によって、各クロックサイクルにおいて複数の動作を実行することが可能となり、それによって性能が向上する。

【００１９】

計算ブロック１１２および１１４の、それぞれの実施態様のブロック図を図３に示してある。多重ポートレジスタファイル２００は、オペランドおよび結果用の一時記憶を提供する。好ましい一実施態様においては、レジスタファイル２００は、それぞれが１２８ビットの８行で構成された、それぞれが３２ビットの３２ワードの容量を有する。レジスタファイル２００は、マルチプレクサおよびラッチ（図示せず）を介してそれぞれのデータバス１６０、１６２および１６４に接続されている（図２）。オペランドがメモリ１１６からフェッチされると、３つのデータバスの２つが選択されて、選択されたバス上のオペランドがレジスタファイル２００に供給される。

【００２０】

図３に示す計算ブロックは、乗算器／累算器２１０、論理演算装置（ＡＬＵ）２１２、シフタ２１４およびアクセラレータ２１６を含む。乗算器／累算器２２０、ＡＬＵ２１２、シフタ２１４およびアクセラレータ２１６は、十分な命令およびオペランドが計算ブロックに供給される程度に、命令を同時に実行することができる。オペランドは、オペランドバス上で、レジスタファイル２００から、乗算器／累算器２１０、ＡＬＵ２１０、シフタ２１４およびアクセラレータ２１６に供給される。乗算器／累算器２１０、ＡＬＵ２１２、シフタ２１４およびアクセラレータ２１６からの結果は、結果バス２２２上を、レジスタファイル２００に戻される。計算ブロックの構成要素は、２次命令復号器２２４からの信号によって、復号された命令に応答して制御される。計算ブロックは、性能の改善のためにはパイプラインアーキテクチャを有するのが好ましい。

【００２１】

ＤＳＰ中のそれぞれの計算ブロック１１２および１１４は、無線電話基地局における性能を向上させるためのアクセラレータ２１６を含む。アクセラレータは、データおよび制御値の一時記憶のためのレジスタ、および指定された命令を実行するためのアクセラレータ回路を含む。アクセラレータは、以下に記述するように、逆拡散関数において必要とされる、複素乗算を実行する。

拡散および逆拡散のプロセスを示す略図を図４に示してある。２つのディジタル化された音声信号３００および３０２が、変調器３０４によって転送の準備のために処理される。変調は、ディジタルビットを無線伝送のための複素数にマッピングするプロセスである。図５に示すように、ディジタルビット０およびディジタルビット１が、複素平面上の点にマッピングされている。高速の伝送速度に対しては、複素平面上の１点が、複数のビットを表すことができる。例えば図６に、ビット対が、複素平面上の点にマッピングされている。このようにして、１＋ｊの値を有する複素信号を受信する受信装置が、その信号をディジタルビット対００と解釈する。このようにして、ディジタルビットの３重項、４重項などを表すように、点を選択することができる。しかしながら、これらの点は、複素平面内で十分離れた位置に選択し、それによって受信装置が受信した複素値を正確に解釈できるようにする必要がある。結果として得られる変調データ信号部分は、記号と呼ばれている。

【００２２】

記号３０６および３０８は、それぞれ、乗算装置３０７および３０９において、それぞれの複素拡散符号３１０、３１２を乗じられる。複素拡散符号３１０および３１２は、互いに直交するように選択される。複素符号を直交とすることによって、結果として得られる符号化信号３１４および３１６を、システムの後に符号を受信する能力と干渉することなく、総和ユニット３１８によって加算することができる。互いに直交する２つの符号の例としては、（１，１）と（１，－１）がある。

【００２３】

図４は、２つの信号のみを含む伝送のための、拡散および逆拡散を示しているが、３つ以

上の信号を結合して伝送することもできることに留意することが重要である。複数信号が結合される場合には、各信号は、記号が受信端で取り出されるように、他のすべての符号に直交する拡散符号が必要である。十分な直交性を有する符号セットを決定するいくつかの方法があるが、好ましい方法を図７のツリーに示してある。ツリー内の、他のすべての符号に直交する符号を生成するためには、基本符号（例中では、（１））を、上部ブランチでは２度反復し、下部ブランチでは１度反復し、１度否定する。例えば、２度反復された符号（１）は、（１，１）となる。反復、かつ否定された符号（１）は、（１，－１）となる。次いで、結果として得られる符号が、図７に示すアルゴリズムに従って、上部ブランチで２回反復されて、底部ブランチで反復かつ否定される。この方法で導いた結果得られる符合はすべて、互いに直交している。直交する符号を選択することによって、受信装置によって信号が取り出されることが保証される。

【００２４】

拡散の後、符号化された信号は合計されて受信装置に伝送される。合計され、かつ拡散された信号部分は、チップと呼ばれる。受信された信号は、逆拡散ブロック３１７および３１９において逆拡散される。逆拡散関数は、乗算・累算関数として簡略化される。受信された信号は、異なるブランチに供給されて、各ブランチにおいて原拡散符合の１つを乗じられる。各ブランチにおける乗算の結果が累算される。複素スクランブル符号は、互いに直交しているので、拡散符号に符号化信号を乗じると、その拡散符号によって拡散された原記号が残り、他の記号は相殺される。逆拡散動作を以下に詳細に記述する。

【００２５】

こうして分離された原記号は、バンドパスフィルタ（ＢＰＦ）３２４に送られる。ＢＰＦ３２４は、信号が属する周波数範囲の外部からのノイズを除去するために使用される。このノイズは、帯域外ノイズと呼ばれる。この帯域外ノイズが除去された後に、信号は、復調器３２６および３２８によって復調されて、原ディジタル信号３００および３０２が回復される。

拡散および逆換算のプロセスによって、多重音声信号を同時に伝送することが可能となる。上記のように、図４の例は、２つの音声信号に限定されない。

【００２６】

記号に対する拡散および逆拡散の効果の詳細な例を図８に示してある。２つの記号セット、ａｂｃｄ５００およびｆｇｈｋ５０２に、それぞれ直交符号５０４および５０６をそれぞれ乗じる。符号５０４は、符号５０６に対して直交している。結果として得られる符号化記号セット５０８および５１０は、互いに加算されて、次いで符号化信号５１２として伝送される。符号化信号部分または「チップ」（符号化信号５１２は１６のチップを含む）が受信されると、それらにそれぞれの符号を乗じ、互いに加算して、拡散係数（図８の例の場合には４である）で除する。こうして、原信号ａｂｃｄおよびｆｇｈｋが、受信側で回復される。

【００２７】

この逆拡散プロセスには、計算が集中している。受信端で結果として得られるデータ信号を得るためには、いくつかの複素乗算実行が必要である。各チップは、逆拡散符号の各セクションで乗じられ、次いでその結果は累算されなければならない。これらの乗算および累算は、受信装置が信号を受信する速度に等しいか、それよりも大きな速度で実行しなくてはならない。セルラー音声伝送に要する高いデータ速度の点から、これまで、ディジタル信号プロセッサは、逆拡散計算を実装するのに必要な計算速度を提供することができなかった。

逆拡散装置のコア動作は、複素乗算および累算である。一実施態様によれば、逆拡散機能は、いくつかのＤＥＳＰＲＥＡＤ（逆拡散）命令の１つを使用して、ディジタル信号プロセッサのアクセラレータ部分２１６で実施される。逆拡散命令のオペランドには、チップ、符号、および前に累算された逆拡散値を含めることができる。

【００２８】

ＤＥＳＰＲＥＡＤ命令によって、ＤＳＰは、逆拡散動作に必要となる複素乗算および累算

機能を迅速に実行することが可能となる。図８の例を参照すると、（ａ＋ｆ）、（ａ－ｆ）、（ａ＋ｆ）などと記されている、信号５１２の部分はチップである（（ａ＋ｆ）は１つのチップである）。それぞれは、８実数ビットと８虚数ビットによってディジタルで表すのが好ましい。（実際には、チップは、サンプリング率またはその他のシステム関連に応じてサイズが変わってもよいが、説明の目的および現在のシステムの好適さから、チップは１６ビットと仮定している。）上記のように、チップは伝送データ信号の一部分である。受信されたチップ値は、ＤＥＳＰＲＥＡＤ命令による処理のために、ＤＳＰレジスタに記憶される。

【００２９】

同様に、ＤＳＰＲＥＡＤ命令に使用するために、複素符号値が、ＤＳＰレジスタ内に記憶される。符号値を±１±ｊに選ぶのが有利である。第１に、これらの符号セグメントは小さく、かつ直交させるのが容易である。第２に、これらの符号セグメントは、２ビットだけ、すなわち１実数ビットと１虚数ビットだけを使用して表すのがはるかに容易であり、ビットがセットされている場合には－１の値を表わし、ビットがクリアされているときには、＋１の値を表す。この表現によって、符号に複素乗算を実施するのに要するハードウエアを大幅に低減することができる。第３に、この符号は十分に小さいために、容易にレジスタに記憶することができ、実行中にメモリからロードする必要がない。

上記のように、拡散係数は、ＤＥＳＰＲＥＡＤ動作毎の複素乗算の数を定義する。ハードウエア上の制約が理由で、本発明の好ましい実施態様では、４で割り切れる拡散係数を使用するが、説明の目的で拡散係数８を使用する。

【００３０】

拡散係数８を有する符号は、ディジタル形式では１６ビット、すなわち、それぞれ１ビット実数部および１ビット虚数部を含む８つの符号セグメントで表される。ディジタイル表現に対して、クリアビットは、＋１の値を表し、セットビットは－１の値を表す。このようにして、２つの１６ビット符号を、３２ビットワード内に記憶することができる。ＤＥＳＰＲＥＡＤ命令の動作を、以下に記述する。

以上に説明したように、符号化データ値から信号を分離するためには、チップに複素符号を乗じる。第１のＤＥＳＰＲＥＡＤ命令は、次の関数を実行することによってこれを達成する。

【数３】

$$\underset{\text{Signal}_{\text{Real}}}{\text{Resulting}} = \sum_{n=0\ \text{to}\ 7} (\text{nCode}_{\text{Real}} * \text{nChip}_{\text{Real}} - \text{nCode}_{\text{Imag.}} * \text{nChip}_{\text{Imag.}}) \qquad (1)$$

$$\underset{\text{Signal}_{\text{Imag.}}}{\text{Resulting}} = \sum_{n=0\ \text{to}\ 7} (\text{nCode}_{\text{Real}} * \text{nChip}_{\text{Imag.}} + \text{nCode}_{\text{Imag.}} * \text{nChip}_{\text{Real}}) \qquad (2)$$

【００３１】

上記の式は、チップと逆拡散符号の複素乗算から得られる信号を計算する。その結果得られる、$Signal_{Real}$は、結果として得られる信号の実数部であり、結果として得られる$Signal_{Imag}$は、結果として得られる信号の虚数部である。$nCode_{Real}$は、ビット値ｎにおける逆拡散符号の実数部分であり、$nCode_{Imag}$は、ビット値ｎにおける逆拡散符号の虚数部分である。同様に、$nChip_{Real}$は、ｎにおける逆拡散符号に対応するチップ値の実数部分であり、$nChip_{Imag}$は、ｎにおける符号値に対応するチップ値の虚数部分である。

【００３２】

この第１のタイプのＤＥＳＰＲＥＡＤ命令は、次の形式を有する。

ＴＲｓ＝ＤＥＳＰＲＥＡＤ（ＲｍＱ，ＴＨｒｄ）

この第１のタイプのＤＥＳＰＲＥＡＤ命令に応答して実行される動作を説明するデータフロー図を図９に示してある。この８つの１６ビットチップ（各チップは、８実数ビットと

８虚数ビットで構成される）の値は、クワッドレジスタＲｍｑ７０４に保持されて（合計１２８ビット）、３２個の２ビット符号セグメントの値（各符号が１実数ビットと１虚数ビットで構成される）が、レジスタペアＴＨｒｄ７０２に保持される。レジスタペア７０２の８つの符号（１６ビット）だけが、この命令の実行中に使用される。この第１のＤＥＳＰＲＥＡＤ命令を実行した後に、符号レジスタＴＨｒｄは、論理的に正しい１６ビットにシフトされて、レジスタの能動部分に新しい符号をロードする。

【００３３】

図９で示すように、複素乗算器７１０～７１７を使用して、７０２に記憶される各符号セグメントに、７０４に記憶された対応するチップを乗じる。これらの乗算の結果は、複素総和ユニット７２０によって加算されて、結果レジスタＴＲｓ７２４に記憶される。図９に示す動作は、上記の式（１）および（２）に対応する。

複素乗算は、ＤＳＰのアクセラレータ２１６（図３）によって実行される。システムを、±１±ｊで表わすことのできる逆拡散符号に限定することによって、複素乗算は、正または負の１の乗算としてＤＳＰで実行することができ、これは、ＤＳＰによって、チップ部分のパスまたはチップ部分の否定として実行することができる。例えば、複素符号が（１，－ｊ）の場合、符号の実数部分は１であり、符号の虚数部分は－１である。実数部分を乗じたチップ部分はすべて、同じままであり、虚数部を乗じたチップ部分はすべて、否定される。

【００３４】

図９に示すように、８つの符号セグメントに、８つのチップを乗じて、互いに加算して結果を得る。この第１のＤＥＳＰＲＥＡＤ命令は、一度に８つの符号セグメントしか使用できないので、第１の逆拡散命令は、拡散係数８を有する逆拡散符号とともに使用するのが好ましい。第１の逆拡散命令はまた、第１の逆拡散命令の結果を、その後の逆拡散命令の結果に加算して、全逆拡散動作を完了することによって、８よりも大きな拡散係数を有する逆拡散符号用に使用することもできる。例えば、逆拡散符号が、拡散係数３２を有するとき、最初の８チップを復号するには、第１の逆拡散命令を１回実行することが必要であり、最終の１６ビット符号セグメントを逆拡散するには、累算を含む逆拡散命令を２回実行することが必要である。

【００３５】

この第２の逆拡散命令は、累算とともに逆拡散されて、次の式を実行する。

【数４】

$$\text{Resulting Signal}_{\text{Real}} = \left(\sum_{n=0\ \text{to}\ 7} \left(\text{nCode}_{\text{Real}} * \text{nChip}_{\text{Real}} - \text{nCode}_{\text{Imag.}} * \text{nChip}_{\text{Imag.}}\right)\right) + \text{Accum. Signal}_{\text{Real}}$$

$$\text{Resulting Signal}_{\text{Imag.}} = \left(\sum_{n=0\ \text{to}\ 7} \left(\text{nCode}_{\text{Real}} * \text{nChip}_{\text{Imag.}} + \text{nCode}_{\text{Imag.}} * \text{nChip}_{\text{Real}}\right)\right) + \text{Accum. Signal}_{\text{Imag.}}$$

ここで、Ａｃｃｕｍ．Ｓｉｇｎａｌ$_{Ｒｅａｌ}$は、先に累算されて逆拡散された信号の実数部であり、Ａｃｃｕｍ．Ｓｉｇｎａｌ$_{Ｉｍａｇ．}$は、先に累算されて逆拡散された信号の虚数部分である。累算とともに逆拡散された、この第２の逆拡散命令は、次の式で表される。

ＴＲｓ＝ＤＥＳＰＲＥＡＤ（ＲｍＱ，ＴＨｒｄ）＋ＴＲｎ

【００３６】

第２のタイプのＤＥＳＰＲＥＡＤ命令に応答して実行される動作を示す、データフロー図を図１０に示してある。図９および１０の同一要素は、同一の参照番号をつけてある。図１０に示す動作は、加算器７２０が、以前のレジスタ値７２２を現在の結果信号に加算して、その結果を出力レジスタ値７２４に配置すること以外は、図９に示すものと同じある

。この第２のＤＥＳＰＲＥＡＤ命令は、累算付きＤＥＳＰＲＥＡＤ命令と呼ぶことができる。

【００３７】

これらの命令を組み込んだソフトウエアコードの例を図１１に示してある。ライン６０２では、信号部分（レジスタＲ３：０に記憶）の最初の８チップに、最初の８つの符号セグメント（ＴＨＲレジスタ１：０に記憶）を乗じて、その結果を結果レジスタＴＲ０に記憶する。同じサイクルにおいて，新規のチップがレジスタＲ３：０にロードされる。（好ましい実施態様のＤＥＳＰＲＥＡＤ命令は、自動的にＴＨＲレジスタをシフトして、新規の符号部分を、レジスタの能動部分に移動させる。）ライン６０４では、（レジスタＲ７：４に記憶された）信号部分の２番目の８チップに、新規の符号部分を乗算して、その乗算の結果を、以前の結果に加算して、新規な結果をレジスタＴＲ０に記憶させる。同一のサイクル中に、新規のチップをレジスタＲ７：４にロードする。乗算および累算関数は、ライン６０６および６０８において反復される。ＤＥＳＰＲＥＡＤ関数を４回実行した後に、レジスタＴＨＲ（３６ビットレジスタ）が、完全にシフトされ（８ビットの４シフト）、新規の符号をＤＳＰによってレジスタＴＨＲにロードしなくてはならないが、レジスタＴＨＲはライン６１０上で再ロードされる。

【００３８】

上記の逆拡散命令の２つの実施態様は、命令が符号の８セグメントを使用して、一時に８チップを復号するように設計されているので、好ましくは、８またはそれよりも大きい拡散係数を有する符号について使用される。これらの命令は、信号部分を逆拡散を完了するのに必要な回数だけ、累算付きＤＥＳＰＲＥＡＤ命令をコールすることによって、８で割り切れる拡散係数を有する信号を逆拡散するのに使用することができる。拡散係数が４（または４で割り切れる）である符号について、逆拡散命令の第３の実施態様について記述する。このＤＥＳＰＲＥＡＤ命令の第３の実施態様は、次の形態を有する。

ＴＲｓｄ＝ＤＥＳＰＲＥＡＤ（ＲｍＱ，ＴＨｒｄ）

【００３９】

この第３のＤＥＳＰＲＥＡＤ命令に応答して実行される動作を示すデータフロー図を、図１２に示してある。図９、１０および１２にある同一の要素は、同一の参照番号をつけてある。図１２に示す動作は、この命令によって逆拡散される符号における拡散係数が４であるので、４つのチップだけを合わせて加算する必要があることを除けば、図１０に示すものと同様である。したがって、この命令は、一度に８チップを復号し、結果を累算しデュアル結果レジスタＴＲｓｄ９２４の別個の半分に記憶することによって、利用可能な８つの複素乗算を使用する。ＤＳＰにおけるＤＥＳＰＲＥＡＤ関数の３つの記述された実施態様は、入信する無線データ信号の処理において、より高い柔軟性と速度を可能にする。

【００４０】

上記のＤＥＳＰＲＥＡＤ命令の例が実行されて、好ましくは８ビット長のビットセグメントで、８つの複素乗算を一度に実行するのが、図９、１０および１２に示してある。これらの命令は、これらの実施態様に限定されるものではなく、より多数の複素乗算器、または異なる長さのビットストリームを使用して実現することが可能であり、また動作を計算ブロック１１２および１１４（図２）の両方で実行することによって処理速度を倍増することができる。

現時点において本発明の好ましい実施態様であると考えられるものを示し、記述したが、添付の特許請求の範囲に定義される本発明の範囲から逸脱することなく、様々な変更および修正ができることが、当業者には明白であろう。

【図面の簡単な説明】

【００４１】

【図１】無線電話基地局信号チェインのブロック図である。

【図２】本発明の一態様によるチャネル復号器を実装するのに好適なディジタル信号プロセッサのブロック図である。

【図３】図２のディジタル信号プロセッサに示した、各計算ブロックの一実施態様のブロ

ック図である。
【図４】直交符号を使用する拡散および逆拡散の処理を示すブロック図である。
【図５】ディジタルビットの複素面上の代表点へのマッピングを示す図である。
【図６】ディジタルビット対の複素面上の代表点へのマッピングを示す図である。
【図７】直交符号ツリーの例を示す図である。
【図８】２組の記号の拡散および逆拡散を示すデータフロー図である。
【図９】逆拡散命令に応答して実行することのできる動作の第１の実施態様を示す、データフロー図である。
【図１０】逆拡散命令に応答して実行することのできる動作の第２の実施態様を示す、データフロー図である。
【図１１】本発明の一態様による逆拡散を実行するためのソフトウエアコードを示す図である。
【図１２】逆拡散命令に応答して実行することのできる動作の第３の実施態様を示す、データフロー図である。

【図１】

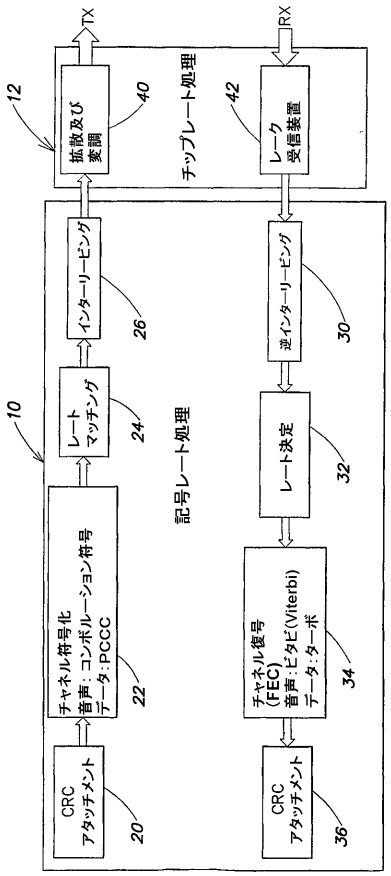


図１

【図２】

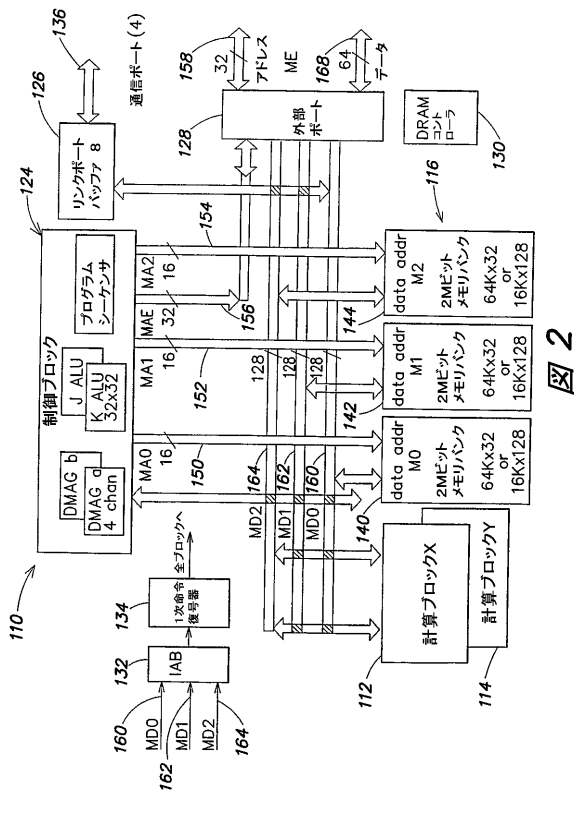


図２

【図３】

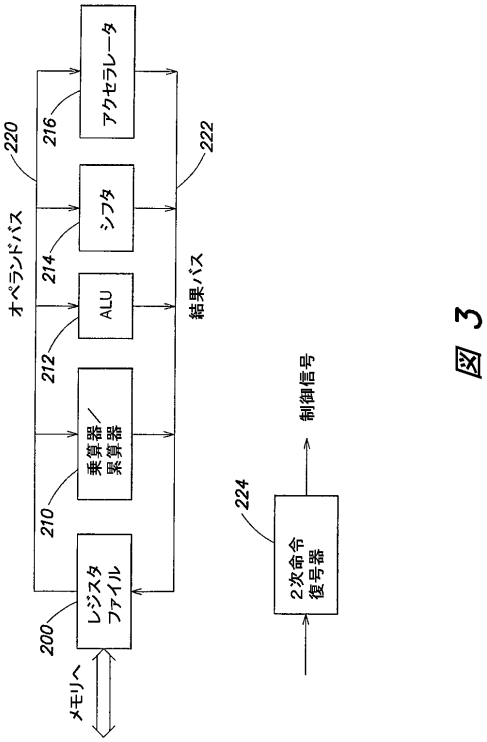


図３

【図４】

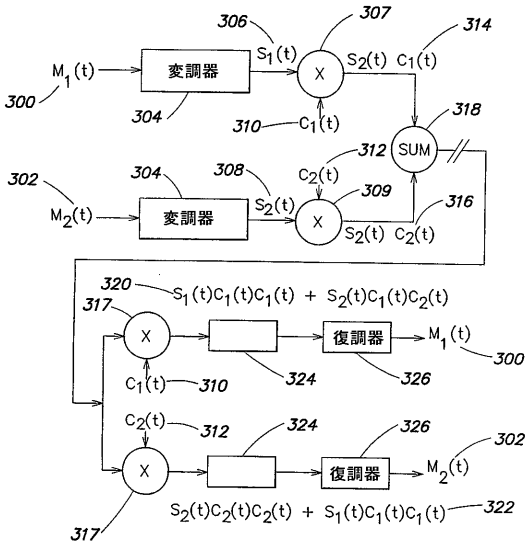


図４

【図５】

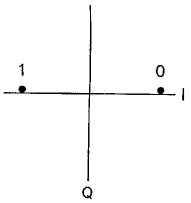


図５

【図６】

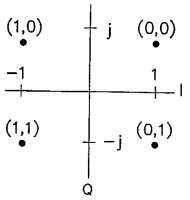


図６

【図７】

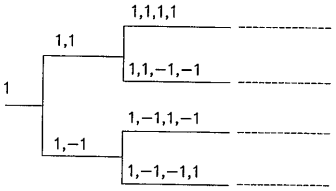


図７

【図８】

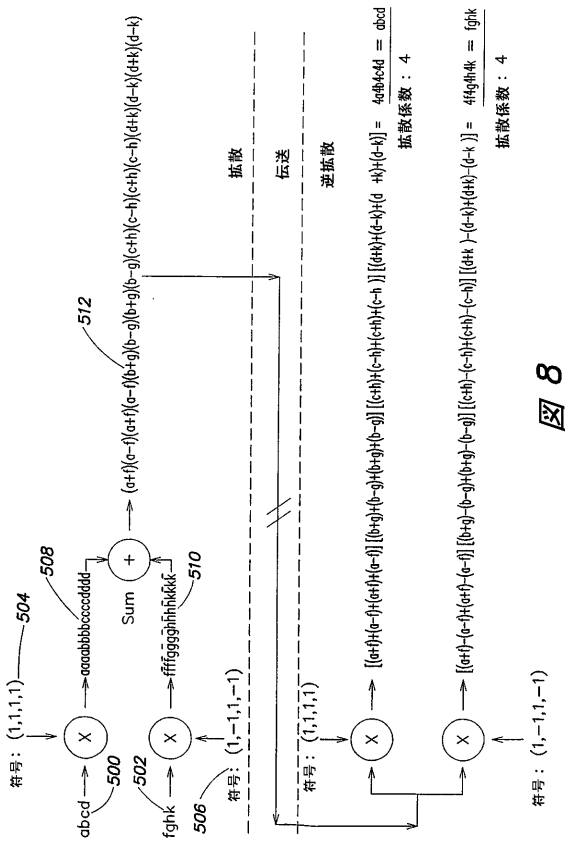


図 8

【図９】

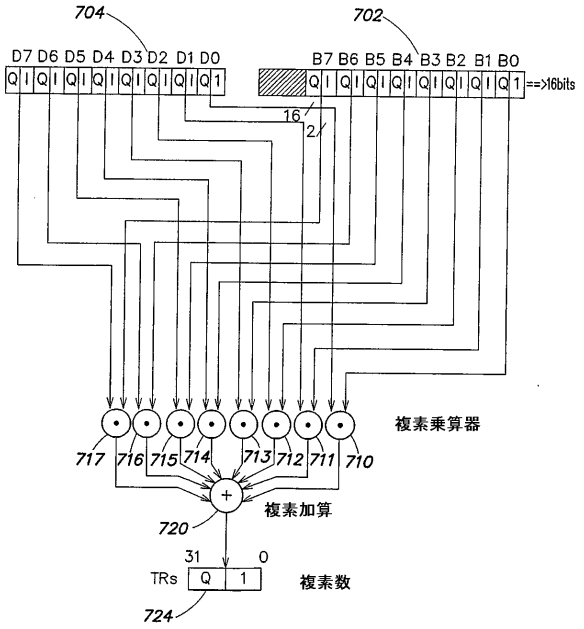


図 9

【図１０】

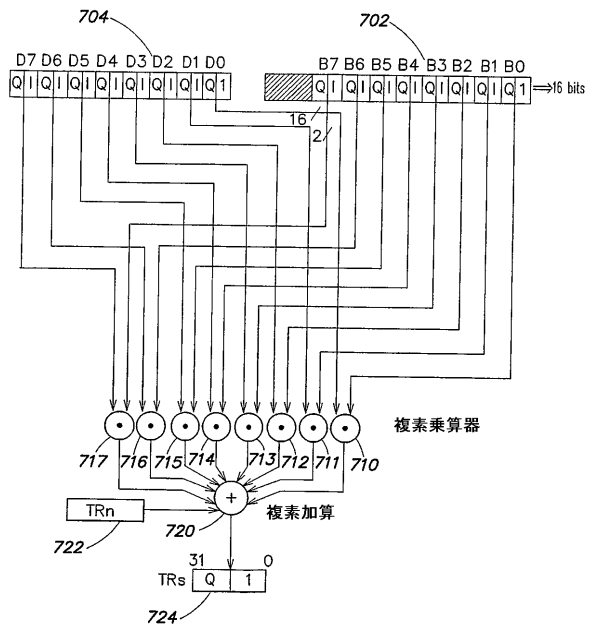


図 10

【図１１】

```
602
TRO = DESPREAD (R3:0, THR1:0)          ; xR3:0 = SDAB Q[JO+=8];;
604
xTRO = DESPREAD (R7:4, THR1:0) + TRO; xR7:4 = SDAB Q[JO+=8];;
606
xTRO = DESPREAD (R3:0, THR1:0) + TRO; xR3:0 = SDAB Q[JO+=8];;
608
xTRO = DESPREAD (R7:4, THR1:0) + TRO; xR7:4 = SDAB Q[JO+=8];
610
xTHR1:0 = R19:18 (I);;
```

図 11

【図１２】

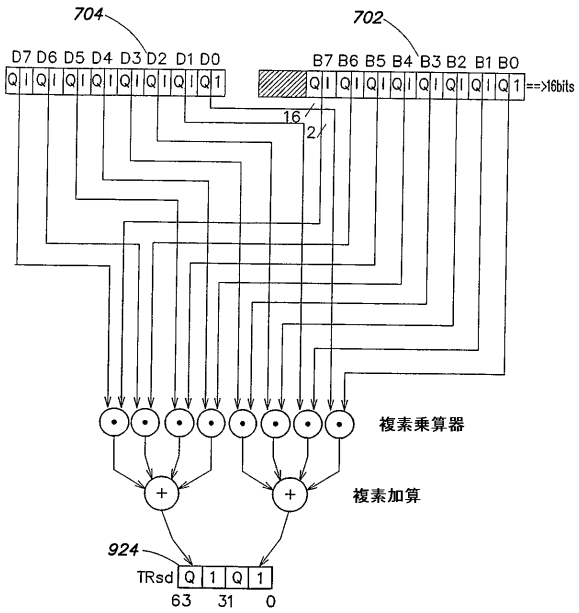

704
D7 D6 D5 D4 D3 D2 D1 D0
Q I Q I Q I Q I Q I Q I Q I Q 1
702
B7 B6 B5 B4 B3 B2 B1 B0
Q I Q I Q I Q I Q I Q I Q I Q 1 ==>16bits
16
2
複素乗算器
複素加算
924
TRsd
Q 1 Q 1
63 31 0

図 12

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 February 2003 (20.02.2003)

PCT

(10) International Publication Number
WO 03/015305 A1

(51) International Patent Classification[7]: H04B 1/707

(21) International Application Number: PCT/US02/24702

(22) International Filing Date: 6 August 2002 (06.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/925,889 6 August 2001 (06.08.2001) US

(71) Applicant: **ANALOG DEVICES, INC.** [US/US]; One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106 (US).

(72) Inventors: **RIFAAT, Rasekh**; 1455 Beacon Street, #14, Brookline, MA 02446 (US). **GREENFIELD, Zvi**; Rivka Goober Street 5, 44471 Ksirsava (IL). **FRIDMAN, Jose**; 1284 Beacon Street, Apt. 220, Brookline, MA 02446 (US).

(74) Agent: **BARZILAY, Ilan, N.**; Wolf, Greenfield & Sacks, P.C., 600 Atlantic Avenue, Boston, MA 02210 (US).

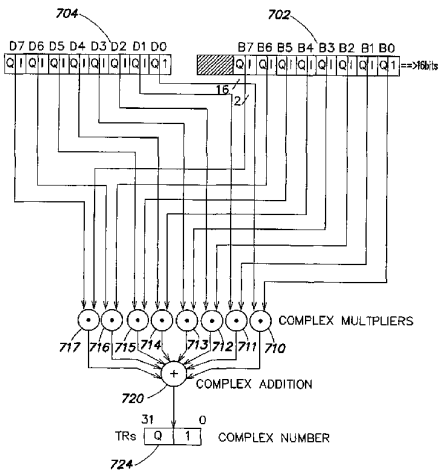
(81) Designated States *(national)*: CN, JP.

(84) Designated States *(regional)*: European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: DESPREAD SIGNAL RECOVERY IN DIGITAL SIGNAL PROCESSORS



(57) Abstract: A digital signal processor performs despread decoding in wireless telephone systems. Orthogonal codes are used to combine data signals into one overall coded signal which is transmitted. The orthogonal codes are used to retrieve individual data signals from the transmitted overall coded signal. Despread instructions are included in the digital signal processor functionality.

WO 03/015305 A1

DESPREAD SIGNAL RECOVERY IN DIGITAL SIGNAL PROCESSORS

Field

This invention relates to digital signal processors for wireless telephone applications and, more particularly, to instructions for isolating signal components using near orthogonal codes.

Background

When implementing cellular telephone systems, it is important to maximize the amount of data that can be transmitted at the available frequency. This often involves altering the data signals to permit the transmission of more data. These alterations must be reversible so that the original signal can be sufficiently recovered at the receiving end. One method of altering the data is through code division multiple access (CDMA) and spreading.

Spreading is a method of altering data whereby a normally narrowband signal is translated into a very wide band signal. Spreading a signal over a wide frequency band allows the signal to be transmitted and received using less power. Spreading is achieved by multiplying the original signal by a code which expands and duplicates the signal. The number of times the signal is duplicated over the wide frequency band is known as the spreading factor. For example, a spreading code of (1, 1, 1, 1) has a spreading factor of 4 because each signal portion (known as a "symbol") is repeated four times when the code is multiplied by a signal. The above example of a spreading code, when multiplied by a signal comprised of symbols "abcd", yields a spread signal of "aaaabbbbccccdddd." If the spreading code is (1, -1, 1, -1) issued to spread the same symbols then the spread signal is "$aa\bar{a}a\bar{b}bb\bar{c}cc\bar{d}dd\bar{d}$" where x represents the negative of the original symbol x. Once spread, the wide band signal is transmitted to the receiver. The signal is despread at the receiver end to recover the original signal. The code used to spread the signal is also used to despread the signal.

Because of several factors, including the intensive computational requirements of a despread function, receivers have implemented despreaders which have included the use of ASICs and the use of a hardware block having the most basic components in a digital signal processor. These implementations suffer from lack of

programmability. Other despreaders have included the use of FPGAs, but these are undesirable because of cost and because they do not offer sufficient flexibility in programming.

A digital signal processor, rather than an ASIC, is a desirable solution because of its software programmability. However, it is difficult for a digital signal processor to handle the high speed complex calculations necessary to implement a satisfactory despreader.

Accordingly, there is a need for improved implementations of despreaders for use in decoding transmitted wireless telephone signals.

Summary

A method is provided for processing a signal value in a digital signal processor, the method comprising the step of, in response to a single instruction that specifies at least a signal value and a despreading code, multiplying the signal value by the despreading code. The method may further comprise the step of adding the result of the multiplying to a result from a previous multiplication. The despreading code may have a spreading factor divisible by 4. The despreading code may be divided into code segments, each code segment comprising a 2 bit complex code including 1 real bit and 1 imaginary bit. For the code bits, a set code bit may represent a value of −1 and a clear code bit may represent a value of +1. The signal value may comprise 16 bits, including 8 real bits and 8 imaginary bits.

A further method is provided for calculating a data set in a digital signal processor, the method comprising the steps of, in response to one or more instructions that specify at least a signal value and a set of codes, for each one of the set of codes multiplying the signal value by one of the set of codes; summing results of the multiplying; and producing a data set resulting from the summing. The summing may comprise summing results of the multiplying with the results of a multiplying by a previous set of codes. The set of codes may have a spreading factor divisible by 4. Each one of the set of codes may be a 2 bit complex code comprising 1 real bit and 1 imaginary bit. A set code bit may represent a value of −1 and a clear code bit may represent a value of +1. The signal value may comprise 16 bits, including 8 real bits and 8 imaginary bits.

In a further embodiment a digital signal processor is provided, the digital signal processor comprising a memory for storing instructions and operands for digital signal computations; a program sequencer for generating instruction addresses for fetching selected ones of said instructions from said memory; and a computation block comprising a register file for temporary storage of operands and results and an execution block for executing a decoding instruction that specifies a data signal and a code, said execution block comprising a complex multiply and accumulate engine for multiplying portions of the data signal by the code and accumulating the results. The digital signal processor may, in response to execution of the decoding instruction by the digital signal processor, perform a set of complex multiplies on portions of the data signal and portions of the code; and sum the results of the complex multiplies. The code may have a spreading factor divisible by 4. In a further embodiment, the code is divided into code segments, each code segment comprising a 2 bit complex code including 1 real bit and 1 imaginary bit. In a still further embodiment, a set code bit represents a value of –1 and a clear code bit represents a value of +1. The data signal may comprise 16 bits, including 8 real bits and 8 imaginary bits.

In another embodiment, a method is provided for calculating output data in a digital signal processor, the method comprising the steps of, in response to one or more instructions that specify at least a set of complex first operands, each one of the first operands comprising 8 real bits and 8 imaginary bits, and a set of complex second operands, each one of the second operands comprising 1 real bit and 1 imaginary bit, for each one of the second operands performing a complex multiplication of one of the first operands by one of the second operands; summing results of the multiplying over the set of second operands; and producing as an output a set of data resulting from the summing. In a further embodiment a set bit in one of the second operands represents a value of –1 and a clear bit in one of the second operands represents a value of +1. In still a further embodiment, the set of complex second operands comprises a despreading code. In yet another embodiment, the set of complex first operands comprises an incoming data signal. In a further embodiment the incoming data signal is a voice transmission signal.

A method is further provided for processing a signal value in a digital signal processor, comprising the step of, in response to a complex signal value and a two bit

complex code segment specified by an instruction, performing a complex multiply of the signal value by the code segment to provide a processed data value.

A method is further offered for processing signal values in a digital signal processor comprising the steps of, (a)in response to a set of complex signal values and a corresponding set of complex code segments specified by an instruction, performing a complex multiply of each signal value by a corresponding code segment to provide a set of intermediate values; and (b) performing complex addition of the intermediate values to provide a processed signal value. The method may further comprise the steps of repeating steps (a) and (b) for a plurality of sets of complex signal values to provide a stream of processed signal values. In the method, each of the complex code segments may be a two bit complex code

Brief Description of the Drawings

For a better understanding of the present invention, reference is made to the accompanying drawings, which are incorporated herein by reference and in which:

FIG. 1 is a block diagram of a wireless telephone base station signal chain;

FIG. 2 is a block diagram of a digital signal processor suitable for implementing channel decoders in accordance with an aspect of the invention;

FIG. 3 is a block diagram of an embodiment of each computation block shown in the digital signal processor of FIG. 2;

FIG. 4 is a block diagram that illustrates the process of spreading and despreading using orthogonal codes;

FIG. 5 illustrates the mapping of digital bits to representation points on a complex plane;

FIG. 6 illustrates the mapping of digital bit pairs to representation points on a complex plane;

FIG. 7 illustrates an example of an orthogonal code tree;

FIG. 8 is a data flow diagram that illustrates the spreading and despreading of two sets of symbols;

FIG. 9 is a data flow diagram that illustrates a first embodiment of operations that may be performed in response to a despread instruction;

FIG. 10 is a data flow diagram that illustrates a second embodiment of operations that may be performed in response to a despread instruction;

FIG. 11 illustrates software code for performing despreading in accordance with an aspect of the invention; and

FIG. 12 is a data flow diagram that illustrates a third embodiment of operations that may be performed in response to a despread instruction.

Detailed Description

For the purposes of illustration, the techniques described below are described in the context of the transmission of cellular telephone signals. However, many of the techniques described, while useful for cell phones, may also be for other high speed data transmission applications.

A block diagram of an example of a wireless telephone base station signal chain is shown in FIG. 1. The signal chain includes symbol rate processing 10 and chip rate processing 12. It is desirable to incorporate the symbol rate processing 10 into a digital signal processor. On the transmit side, the symbol rate processing 10 includes a CRC attachment block 20, a channel coding block 22, a rate matching block 24 and an interleaving block 26. On the receive side, the symbol rate processing 10 includes a de-interleaving block 30, a rate determination block 32, a channel decoding block 34 and a CRC attachment block 36.

The chip rate processing 12 includes a spreading and modulation block 40 on the transmit side, and a rake receiver 42 on the receive side. A digital signal processor which performs despreading may be utilized on the receive side.

A block diagram of an example of a digital signal processor (DSP) 110 suitable for implementing features of the present invention is shown in FIG. 2. The principal components of the DSP 110 are computation blocks 112 and 114, a memory 116, a control block 124, link port buffers 126, an external port 128, a DRAM controller 130, an instruction alignment buffer (IAB) 132 and a primary instruction decoder 134. The computation blocks 112 and 114, the instruction alignment buffer 132, the primary instruction decoder 134 and the control block 124 constitute a core processor which performs the main computation and data processing functions of the

DSP 110. The external port 128 controls external communications via an external address bus 158 and an external data bus 168. The link port buffers 126 control external communication via communication ports 136. The DSP is preferably configured as a single monolithic integrated circuit.

The memory 116 may include three independent, large capacity memory banks 140, 142 and 144. In a preferred embodiment, each of the memory banks 140, 142 and 144 has a capacity of 64 K words of 32 bits each. As discussed below, each of the memory banks 140, 142 and 144 preferably has a 128-bit data bus. Up to four consecutive aligned data words of 32 bits each can be transferred to or from each memory bank in a single clock cycle.

The elements of the DSP 110 are interconnected by buses for efficient, high speed operation. Each of the buses includes multiple lines for parallel transfer of binary information. A first address bus 150 (MA0) interconnects memory bank 140 (M0) and control block 124. A second address bus 152 (MA1) interconnects memory bank 142 (M1) and control block 124. A third address bus 154 (MA2) interconnects memory bank 144 (M2) and control block 124. Each of the address buses 150, 152 and 154 is preferably 16 bits wide. An external address bus 156 (MAE) interconnects external port 128 and control block 124. The external address bus 156 is interconnected through external port 128 to external address bus 158. Each of the external address buses 156 and 158 is preferably 32 bits wide. A first data bus 160 (MD0) interconnects memory bank 140, computation blocks 112 and 114, control block 124, link port buffers 126, IAB 132 and external port 128. A second data bus 162 (MD1) interconnects memory bank 142, computation blocks 112 and 114, control block 124, link port buffers 126, IAB 132 and external port 128. A third data bus 164 (MD2) interconnects memory bank 144, computation blocks 112 and 114, control block 124, link port buffers 126, IAB 132 and external port 128. The data buses 160, 162 and 164 are connected through external port 128 to external data bus 168. Each of the data buses 160, 162 and 164 is preferably 128 bits wide, and external data bus 168 is preferably 64 bits wide.

The first address bus 150 and the first data bus 160 comprise a bus for transfer of data to and from memory bank 140. The second address bus 152 and the second data bus 162 comprise a second bus for transfer of data to and from memory bank

142. The third address bus 154 and the third data bus 164 comprise a third bus for transfer of data to and from memory bank 144. Since each of the memory banks 140, 142 and 144 has a separate bus, the memory banks 140, 142 and 144 may be accessed simultaneously. As used herein, "data" refers to binary words, which may represent either instructions or operands that are associated with the operation of the DSP 110.

In a typical operating mode, program instructions are stored in one of the memory banks, and operands are stored in the other two memory banks. Thus, at least one instruction and two operands can be provided to the computation blocks 112 and 114 in a single clock cycle. Each of the memory banks 140, 142 and 144 may be configured to permit reading and writing of multiple data words in a single clock cycle. The simultaneous transfer of multiple data words from each memory bank in a single clock cycle is accomplished without requiring an instruction cache or a data cache.

As indicated above, each of the memory banks 140, 142 and 144 preferably has a capacity of 64 K words of 32 bits each. Each memory bank may be connected to a data bus that is 128 bits wide. In an alternative embodiment, each data bus may be 64 bits wide, and 64 bits are transferred on each of clock phase 1 and clock phase 2, thus providing an effective bus width of 128 bits. Multiple data words can be accessed in each memory bank in a single clock cycle. Specifically, data can be accessed as single, dual or quad words of 32 bits each.

Using quad word transfers, four instructions and eight operands, each of 32 bits, can be supplied to the computation blocks 112 and 114 in a single clock cycle. The number of data words transferred and the computation block or blocks to which the data words are transferred are selected by control bits in the instruction. The single, dual or quad data words can be transferred to computation block 112, to computation block 114, or to both. Dual and quad data word accesses improve the performance of the DSP 110 in many applications by allowing several operands to be transferred to the computation blocks 112 and 114 in a single clock cycle. The ability to access multiple instructions in each clock cycle allows multiple operations to be executed in each clock cycle, thereby improving performance.

A block diagram of an embodiment of each of the computation blocks 112 and 114 is shown in FIG. 3. A multiple port register file 200 provides temporary storage

for operands and results. In a preferred embodiment, the register file 200 has a capacity of 32 words of 32 bits each, organized as eight rows of 128 bits each. The register file 200 is connected through a multiplexer and latch (not shown) to each of the data buses 160, 162 and 164 (FIG. 2). When operands are fetched from memory 116, two of the three data buses are selected, and the operands on the selected buses are supplied to the register file 200.

The computation block shown in FIG. 3 includes a multiplier/accumulator 210, an arithmetic logic unit (ALU) 212, a shifter 214 and an accelerator 216. The multiplier/accumulator 220, the ALU 212, the shifter 214 and the accelerator 216 are capable of simultaneous execution of instructions to the extent that sufficient instructions and operands can be supplied to the computation blocks. Operands are supplied from the register file 200 to multiplier/accumulator 210, ALU 210, shifter 214 and accelerator 216 on operand buses 220. Results from the multiplier/accumulator 210, the ALU 212, the shifter 214 and the accelerator 216 are returned to register file 200 on result buses 222. The components of the computation block are controlled by signals from a secondary instruction decoder 224, in response to a decoded instruction. The computation block preferably has a pipelined architecture for improved performance.

Each of the computation blocks 112 and 114 in the DSP includes the accelerator 216 for enhanced performance in wireless telephone base stations. The accelerator includes registers for temporary storage of data and control values, and accelerator circuitry for executing specified instructions. The accelerator performs the complex multiplication required in a despread function, as described below.

A simplified diagram illustrating the processes of spreading and despreading is shown in FIG. 4. Two digitized voice signals 300 and 302 are processed by modulators 304 in preparation for transmission. Modulation is the process by which digital bits are mapped to a complex numbers for radio transmission. As shown in FIG. 5 the digital bit 0 and digital bit 1 are mapped to points on a complex plane. For increased transmission rates, one point on the complex plane may represent multiple bits. For example in FIG. 6 bit pairs are mapped to points on the complex plane. In this manner, a receiver which receives a complex signal with a value of 1+ j interprets that signal as the digital bit pair 00. Points may be chosen in this manner to represent

digital bit triplets, quadruplets, etc. However, the points must be chosen to be sufficiently far apart in the complex plane so that the receiver may accurately interpret received complex values. The resulting modulated data signal portions are known as symbols.

Symbols 306 and 308 are each multiplied in multipliers 307 and 309 by a respective complex spreading code, 310 or 312. The complex spreading codes 310 and 312 are chosen so they are orthogonal to one another. By making the complex codes orthogonal, the resulting coded signals 314 and 316 may be added by summing unit 318 without interfering with the system's ability to later receive the codes. An example of two codes which are orthogonal to each other is (1, 1) and (1, -1).

It is important to remember that while FIG. 4 depicts spreading and despreading for a transmission containing only two signals, more than two signals may be combined for transmission. If multiple signals are combined, each signal requires a spreading code which is orthogonal to every other code, so that the symbols may be retrieved at the receiving end. There are several methods for determining code sets with sufficient orthogonality; a preferred method is depicted in the tree of FIG. 7. To create codes which are orthogonal to all other codes in the tree, the base code (in the example, (1)) is repeated twice in a top branch and repeated once and negated once in a bottom branch. For example the code (1) repeated twice become (1, 1). The code (1) repeated and negated becomes (1, -1). The resulting codes are then repeated twice in a top branch and repeated and negated in a bottom branch according to the algorithm shown in FIG. 7. All resulting codes derived in this method are orthogonal to each other. Choosing orthogonal codes ensures that the signals are retrievable by the receiver.

After spreading, the coded signals are summed and transmitted to the receiver. The summed and spread signal portions are known as chips. The received signal is despread at despread blocks 317 and 319. A despread function is simplified as a multiply and accumulate function. The received signal is supplied to different branches and in each branch is multiplied by one of the original spreading codes. The results of the multiplication in each branch are accumulated. Because the complex scrambling codes are orthogonal to one another, when the spreading code is multiplied by the coded signal, the original symbols which were spread by that

spreading code remain and all other symbols cancel. The despread operation is discussed in detail below.

The now isolated original symbols are passed to the band pass filter (BPF) 324. The BPF 324 is used to remove noise from outside of the frequency range in which the signals are located. This noise is referred to as out of band noise. After the out of band noise has been removed the signals are demodulated by demodulators 326 and 328 and the original digital signals 300 and 302 are recovered.

The processes of spreading and despreading allow multiple voice signals to be transmitted together. As described above, the example of FIG. 4 is not limited to two voice signals.

A detailed example of the effect of spreading and despreading on symbols is shown in FIG. 8. Two symbol sets abcd 500 and fghk 502 are each multiplied by orthogonal codes 504 and 506, respectively. Code 504 is orthogonal to code 506. The resulting coded symbol sets 508 and 510 are added together and are then transmitted as coded signal 512. When the coded signal portions, or "chips", (coded signal 512 contains 16 chips) are received, they are multiplied by their respective codes, added together and divided by the spreading factor (which in the case of the example of FIG. 8 is 4). The original signals abcd and fghk are then recovered on the receive side.

This process of despreading is computationally intensive. Several complex multiply executions are necessary to obtain the resulting data signals on the receiver end. Each chip must be multiplied by each section of the despread code and then these results must be accumulated. These multiplies and accumulates must happen at a rate canal to or greater than the rate at which the receiver receives signals. In light of the high data rates with cellular voice transmissions, digital signal processors have, until now, been unable to provide the calculation speed necessary to implement despread calculations.

The core operations for a despreader are a complex multiply and an accumulate. According to one embodiment, the despread function is carried out in an accelerator portion 216 of a digital signal processor by use of one of several DESPREAD instructions. The operands for the despread instructions may include the chips, the codes, and the previously accumulated despread values.

The DESPREAD instructions permit a DSP to quickly perform the complex multiply and accumulate functions required for the despread operation. Referring to the example of FIG. 8, the portions of signal 512 noted as (a+f), (a-f), (a+f), etc. are chips ((a+f) is a chip). Each is preferably represented digitally by 8 real bits and 8 imaginary bits. (In practice the chips may vary in size, depending on sampling rates or other system concerns, but for the purposes of illustration, and for current system preferences, chips are assumed to be 16 bits). As described above, a chip is a portion of the transmitted data signal. The received chip values are stored in the DSP registers for processing by a DESPREAD instruction.

Similarly the complex code values are stored in the DSP registers for use by a DESPREAD instruction. There are advantages to choosing the code values to be ±1 ±j. First, the code segments are small and more easily made orthogonal. Second, they are much easier to represent using only two bits: one real bit and one imaginary bit, if the bit is set it represents a value of −1, if the bit is clear it represents a value of +1. This representation greatly reduces the hardware necessary to perform a complex multiply with the codes. Third, the codes are small enough to be easily stored in a register and don't have to be loaded from memory during execution.

As described above, the spreading factor defines the number of complex multiplies per DESPREAD operation. Because of hardware constraints, the preferred embodiment of the invention utilizes spreading factors divisible by 4, and a spreading factor of 8 is utilized for purposes of illustration.

A code with a spreading factor of 8 is represented digitally by 16 bits: eight code segments, each with a 1 bit real portion and a 1 bit imaginary portion. For digital representation, a clear bit represents a value of +1 and a set bit represents a value of -1. In this manner, two 16 bit codes may be stored in a 32 bit word. The operation of the DESPREAD instructions is described below.

As explained above, to isolate the signals from the coded data values, the chips are multiplied by the complex codes. A first DESPREAD instruction accomplishes this by executing the function:

$$\text{Resulting Signal}_{\text{Real}} = \sum_{n=0\ \text{to}\ 7} (\text{nCode}_{\text{Real}} * \text{nChip}_{\text{Real}} - \text{nCode}_{\text{Imag.}} * \text{nChip}_{\text{Imag.}}) \quad (1)$$

$$\text{Resulting Signal}_{Imag.} = (nCode_{Real} * nChip_{Imag.} + nCode_{Imag.} * nChip_{Real}) \quad (2)$$

The equations above calculate the signal which results from the complex multiplication of chips by despreading codes. Resulting $Signal_{Real}$ is the real portion of the resulting signal and Resulting $Signal_{Imag.}$ is the imaginary portion of the resulting signal. $nCode_{Real}$ is the real portion of the despreading code at bit value n, and $nCode_{Imag.}$ is the imaginary portion of the despread code at bit value n. Similarly $nChip_{Real}$ is the real portion of the chip value which corresponds to the despreading code at n, and $nChip_{Imag.}$ is the imaginary portion of the chip value corresponds to the code value at n.

This first type of DESPREAD instruction has the following form:

TRs = DESPREAD (RmQ, THrd)

A data flow diagram that illustrates operations performed in response to this first type of DESPREAD instruction is shown in FIG. 9. The values of eight 16 bit chips (each chip made up of 8 real bits and 8 imaginary bits) are held in quad register Rmq 704 (128 bits total) and the values of thirty two 2 bit code segments (each code made up of 1 real bit and 1 imaginary bit) are held in register pair THrd 702. Only eight codes (16 bits) of register pair 702 are used during the execution of the instruction. After this first DESPREAD instruction is executed, THrd, the code register, is logically shifted right 16 bits to load the new codes into the active portion of the register.

As shown in FIG. 9 each code segment stored in 702 is multiplied by a corresponding chip stored in of 704 using the complex multipliers 710-717. The results of these multiplies are added by a complex summing unit 720 and stored in a result register TRs 724. The operation illustrated in Fig. 9 correspond to Equations (1) and (2) above.

The complex multiplies are executed by the accelerator 216 (Fig. 3) of the DSP. By limiting the system to despread codes which may be expressed as $\pm 1 \pm j$, the complex multiplies may be executed by the DSP as multiply by a positive or negative 1, which may be implemented by the DSP as a passing of the chip portion or a negation of the chip portion. For example, if the complex code is $(1, -j)$ then the real

portion of the code is 1 and the imaginary portion of the code is –1. Any chip portion multiplied by the real portion remains the same and any chip portion multiplied by the imaginary portion is negated.

As shown in FIG. 9 eight code segments are multiplied by 8 chips and added together to provide a result. Because this first DESPREAD instruction is only able to make use of eight code segments at one time, the first despread instruction is preferably used with despread codes having a spreading factor of eight. The first despread instruction may also be used for despread codes having a spreading factor larger than eight by adding the result of a first despread instruction to the result of later despread instructions to complete a full despread operation. For example, if a despread code has a spreading factor of 32, one execution of the first despread instruction is necessary to decode the first 8 chips, and 2 executions of a despread instruction with accumulate are necessary to despread the final 16 bit code segments.

This second depsread instruction, despread with accumulate, executes the following equations:

$$\text{Resulting Signal}_{Real} = \left(\sum_{n=0 \text{ to } 7} (nCode_{Real} * nChip_{Real} - nCode_{Imag.} * nChip_{Imag.})\right) + \text{Accum. Signal}_{Real}$$

$$\text{Resulting Signal}_{Imag.} = \left(\sum_{n=0 \text{ to } 7} (nCode_{Real} * nChip_{Imag.} + nCode_{Imag.} * nChip_{Real})\right) + \text{Accum. Signal}_{Imag.}$$

Where Accum. $Signal_{Real}$ is the real portion of the previously accumulated and despread signal and Accum. $Signal_{Imag.}$ is the imaginary portion of the previously accumulated and despread signal. This second despread instruction, despread with accumulate, has the following form:

TRs = DESPREAD (RmQ, THrd) + TRn

A data flow diagram that illustrates operations performed in response to this second type of DESPREAD instruction is shown in FIG. 10. Like elements in FIGs. 9 and 10 have the same reference numerals. The operations shown in FIG. 10 are identical

to those shown in FIG. 9, except that the adder 720 also adds the previous register value 722 to the current resulting signal and places the result in the output register value 724. This second DESPREAD instruction maybe referred to as a DESPREAD with accumulate instruction.

An example of software code incorporating these instructions is shown in FIG. 11. In line 602, the first eight chips of a signal portion (stored in registers R3:0) are multiplied by the first eight code segments (stored in THR registers 1:0) and the results are stored into result register TR0. In the same cycle, new chips are loaded into registers R3:0. (The DESPREAD instruction in its preferred embodiment automatically shifts the THR register to move new code portions into the active portion of the register.) In line 604, the second eight chips of a signal portion (stored in registers R7:4) are multiplied by the new code portions, the result of that multiplication is added to the previous result and the new result is stored into register TR0. In the same cycle new chips are loaded into register R7:4. The multiply and accumulate functions are repeated in lines 606 and 608. After four executions of the DESPREAD instruction, register THR (a 36 bit register) has been fully shifted (four shifts of 8 bits) and new codes must be loaded into register THR by the DSP; register THR is reloaded on line 610.

The two embodiments of the despread instruction described above are preferably used for codes with a spreading factor of 8 or greater as the instructions are designed to decode 8 chips at a time using 8 segments of code. These instructions may be used to despread signals with spreading factors divisible by 8 by calling the DESPREAD with accumulate instruction as many times as necessary to complete the despreading of the signal portion. For codes with a spreading factor of 4 (or divisible by 4), a third embodiment of the despread instruction is described. This third embodiment of the DESPREAD instruction has the following form:

TRsd = DESPREAD (RmQ, THrd)

A data flow diagram that illustrates operations performed in response to this third type of DESPREAD instruction is shown in FIG. 12. Like elements in FIGs. 9, 10 and 12 have the same reference numerals. The operations shown in FIG. 12 are similar to

those shown in FIG. 10, except that since the spreading factor in codes despread by this instruction is 4, only four chips need to be added together. Therefore, this instruction makes use of the available eight complex multipliers by decoding 8 chips at once, and accumulating and storing the results into separate halves of the dual result register TRsd 924. The three described embodiments of the DESPREAD function in a DSP allow for much greater flexibility and speed in processing incoming wireless data signals.

The above examples of DESPREAD instructions are shown executed in FIGs. 9, 10 and 12 performing eight complex multiplications at one time on bit segments preferably eight bits long. The instructions are not limited to these embodiments and may be implemented using more complex multipliers or bit streams of different lengths operations can be performed in both comp blocks 112 and 114 (Fig. 2) to double processing speed.

While there have been shown and described what are at present considered the preferred embodiments of the present invention, it will be obvious to those skilled in the art that various changes and modifications may be made therein without departing from the scope of the invention as defined by the appended claims.

CLAIMS

1. A method for processing a signal value in a digital signal processor, the method comprising the step of:

in response to a single instruction that specifies at least a signal value and a despreading code, multiplying the signal value by the despreading code.

2. A method as defined in claim 1 further comprising the step of adding the result of the multiplying to a result from a previous multiplication.

3. A method as defined in claim 1, wherein the despreading code has a spreading factor divisible by 4.

4. A method as defined in claim 1, wherein the despreading code is divided into code segments, each code segment having comprising a 2 bit complex code comprising 1 real bit and 1 imaginary bit.

5. A method as defined in claim 4, wherein a set code bit represents a value of –1 and a clear code bit represents a value of +1.

6. A method as defined in claim 1, wherein the signal value comprises 16 bits.

7. A method as defined in claim 6, wherein the signal value comprises 8 real bits and 8 imaginary bits.

8. A method for calculating a data set in a digital signal processor, the method comprising the steps of:

in response to one or more instructions that specify at least a signal value and a set of codes:

for each one of the set of codes multiplying the signal value by one of the set of codes;

summing results of the multiplying; and

producing a data set resulting from the summing.

9. A method as defined in claim 8, wherein the summing comprises summing results of the multiplying with the results of a multiplying by a previous set of codes.

10. A method as defined in claim 8, wherein the set of codes has a spreading factor divisible by 4.

11. A method as defined in claim 8, wherein each one of the set of codes is a 2 bit complex code comprising 1 real bit and 1 imaginary bit.

12. A method as defined in claim 11, wherein a set code bit represents a value of -1 and a clear code bit represents a value of +1.

13. A method as defined in claim 8, wherein the signal value comprises 16 bits.

14. A method as defined in claim 13, wherein the signal value comprises 8 real bits and 8 imaginary bits.

15. A digital signal processor comprising:

a memory for storing instructions and operands for digital signal computations;

a program sequencer for generating instruction addresses for fetching selected ones of said instructions from said memory; and

a computation block comprising a register file for temporary storage of operands and results and an execution block for executing a decoding instruction that specifies a data signal and a code, said execution block comprising a complex multiply and accumulate engine for multiplying portions of the data signal by the code and accumulating the results.

16. A digital signal processor as defined in claim 15, wherein, in response to an execution of the decoding instruction the digital signal processor:

performs a set of complex multiplies on portions of the data signal and portions of the code; and

sums the results of the complex multiplies.

17. A method as defined in claim 15, wherein the code has a spreading factor divisible by 4.

18. A method as defined in claim 15, wherein the code is divided into code segments, each code segment having comprising a 2 bit complex code comprising 1 real bit and 1 imaginary bit.

19. A method as defined in claim 18, wherein a set code bit represents a value of –1 and a clear code bit represents a value of +1.

20. A method as defined in claim 15, wherein the data signal comprises 16 bits.

21. A method as defined in claim 20, wherein the data signal comprises 8 real bits and 8 imaginary bits.

22. A method for calculating output data in a digital signal processor, the method comprising the steps of:

in response to one or more instructions that specify at least a set of complex first operands each one of the first operands comprising 8 real bits and 8 imaginary bits and a set of complex second operands each one of the second operands comprising 1 real bit and 1 imaginary bit:

for each one of the second operands performing a complex multiplication of one of the first operands by one of the second operands;

summing results of the multiplying over the set of second operands; and

producing as an output a set of data resulting from the summing.

23. A method as defined in claim 22, wherein a set bit in one of the second operands represents a value of –1 and a clear bit in one of the second operands represents a value of +1.

24. A method as defined in claim 22, wherein the set of complex second operands comprises a despreading code.

25. A method as defined in claim 22, wherein the set of complex first operands comprises an incoming data signal.

26. A method as defined in claim 25, wherein the incoming data signal is a voice transmission signal.

27. A method for processing a signal value in a digital signal processor, comprising the step of:

in response to a complex signal value and a two bit complex code segment specified by an instruction, performing a complex multiply of the signal value by the code segment to provide a processed data value.

28. A method for processing signal values in a digital signal processor comprising the steps of:

(a) in response to a set of complex signal values and a corresponding set of complex code segments specified by an instruction, performing a complex multiply of each signal value by a corresponding code segment to provide a set of intermediate values; and

(b) performing complex addition of the intermediate values to provide a processed signal value.

29. A method as defined in claim 28 further comprising the steps of repeating steps (a) and (b) for a plurality of sets of complex signal values to provide a stream of processed signal values.

30. A method as defined in claim 28 further wherein each of the complex code segments is a two bit complex code.

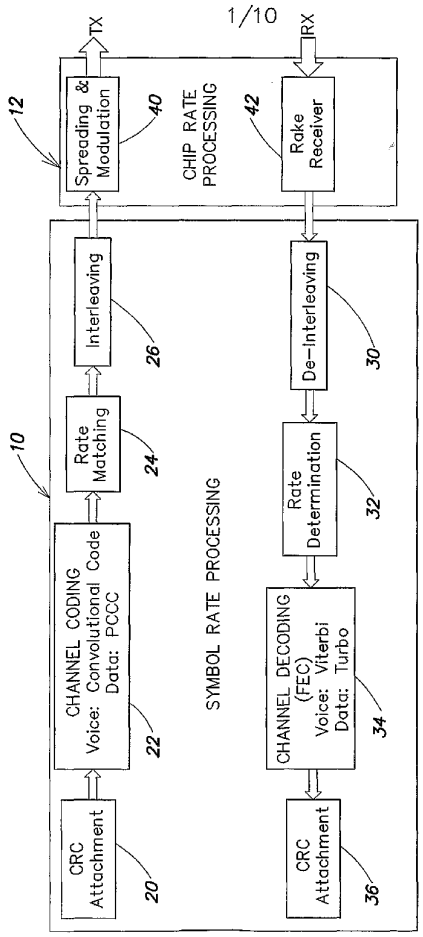

TX
RX
12
Spreading & Modulation
40
CHIP RATE PROCESSING
42
Rake Receiver
Interleaving
26
De-Interleaving
30
10
Rate Matching
24
Rate Determination
32
CHANNEL CODING
Voice: Convolutional Code
Data: PCCC
22
SYMBOL RATE PROCESSING
CHANNEL DECODING
(FEC)
Voice: Viterbi
Data: Turbo
34
CRC Attachment
20
CRC Attachment
36

FIG. 1

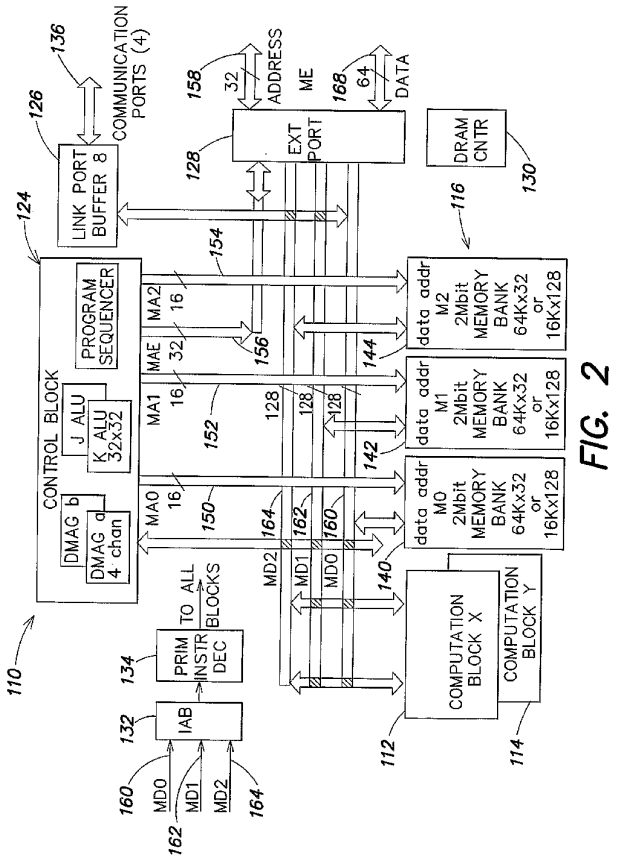

110
CONTROL BLOCK
DMAG b
DMAG a
4 chan
J ALU
K ALU 32x32
PROGRAM SEQUENCER
LINK PORT BUFFER 8
COMMUNICATION PORTS (4)
126
136
124
EXT PORT
ADDRESS
DATA
32
64
158
168
128
DRAM CNTR
130
116
154
156
MA2
MAE
MA1
MA0
16
32
152
150
144
142
140
data addr M2 2Mbit MEMORY BANK 64Kx32 or 16Kx128
data addr M1 2Mbit MEMORY BANK 64Kx32 or 16Kx128
data addr M0 2Mbit MEMORY BANK 64Kx32 or 16Kx128
128
164
162
160
MD2
MD1
MD0
TO ALL BLOCKS
PRIM INSTR DEC
134
IAB
132
COMPUTATION BLOCK X
COMPUTATION BLOCK Y
112
114

FIG. 2

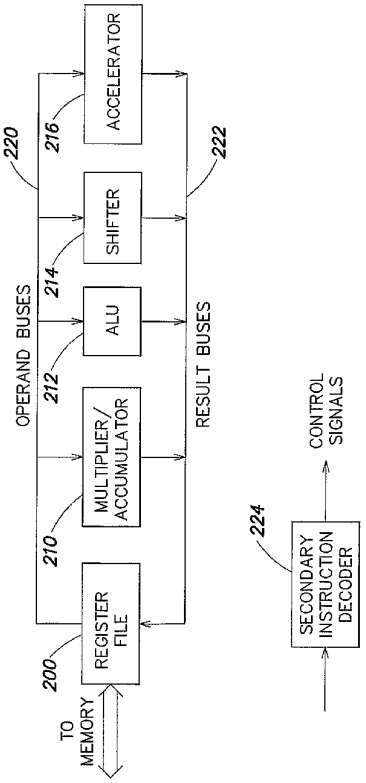

ACCELERATOR
SHIFTER
ALU
MULTIPLIER/ ACCUMULATOR
REGISTER FILE
TO MEMORY
OPERAND BUSES
RESULT BUSES
SECONDARY INSTRUCTION DECODER
CONTROL SIGNALS
200
210
212
214
216
220
222
224

FIG. 3

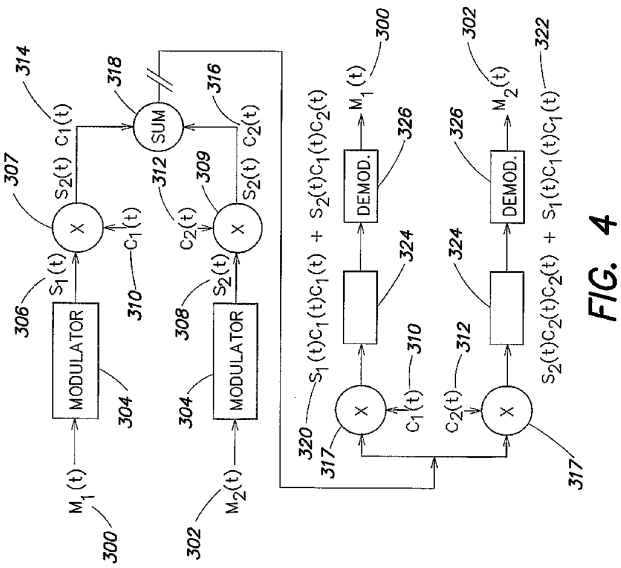

MODULATOR
MODULATOR
SUM
DEMOD.
DEMOD.
$M_1(t)$
$M_2(t)$
$S_1(t)$
$S_2(t)$
$C_1(t)$
$C_2(t)$
$S_1(t)C_1(t) + S_2(t)C_1(t)C_2(t)$
$S_2(t)C_2(t) + S_1(t)C_1(t)C_1(t)$
300
302
304
306
307
308
309
310
312
314
316
317
318
320
322
324
326

FIG. 4

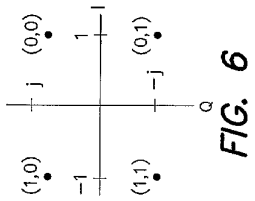


FIG. 6

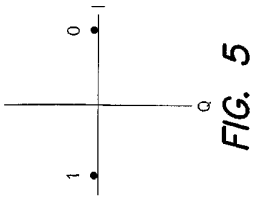


FIG. 5

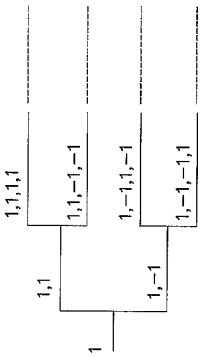


FIG. 7

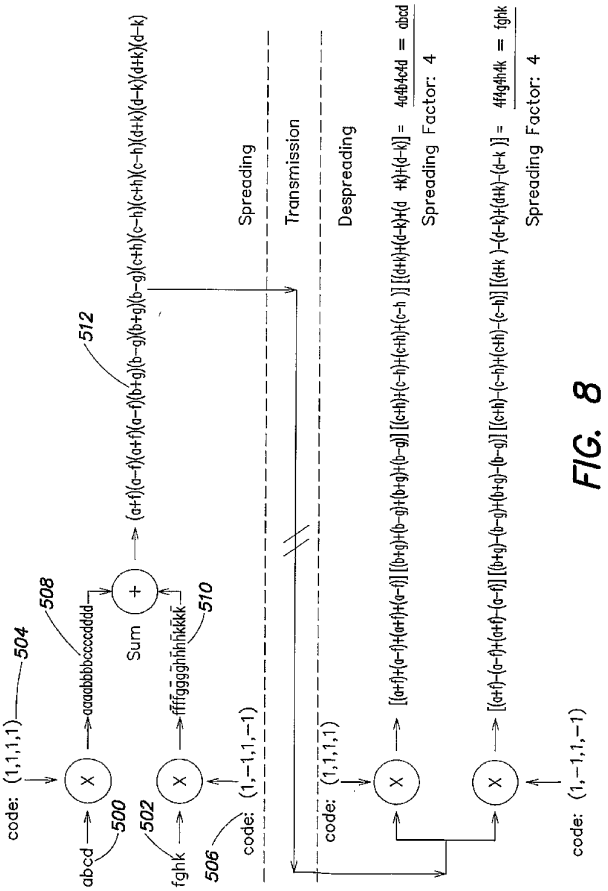

code: (1,1,1,1)
504
508
aaaabbbbccccddddd
512
(a+f)(a−f)(a+f)(a−f)(b+g)(b−g)(b+g)(b−g)(c+h)(c−h)(c+h)(c−h)(d+k)(d−k)(d+k)(d−k)
Sum
+
fffffgggghhhhkkkk
510
abcd
500
X
fghk
502
506
code: (1,−1,1,−1)
Spreading
Transmission
Despreading
code: (1,1,1,1)
[(a+f)+(a−f)+(a+f)+(a−f)][(b+g)+(b−g)+(b+g)+(b−g)][(c+h)+(c−h)+(c+h)+(c−h)][(d+k)+(d−k)+(d+k)+(d−k)] = 4a4b4c4d = abcd
Spreading Factor: 4
code: (1,−1,1,−1)
[(a+f)−(a−f)+(a+f)−(a−f)][(b+g)−(b−g)+(b+g)−(b−g)][(c+h)−(c−h)+(c+h)−(c−h)][(d+k)−(d−k)+(d+k)−(d−k)] = 4f4g4h4k = fghk
Spreading Factor: 4

FIG. 8

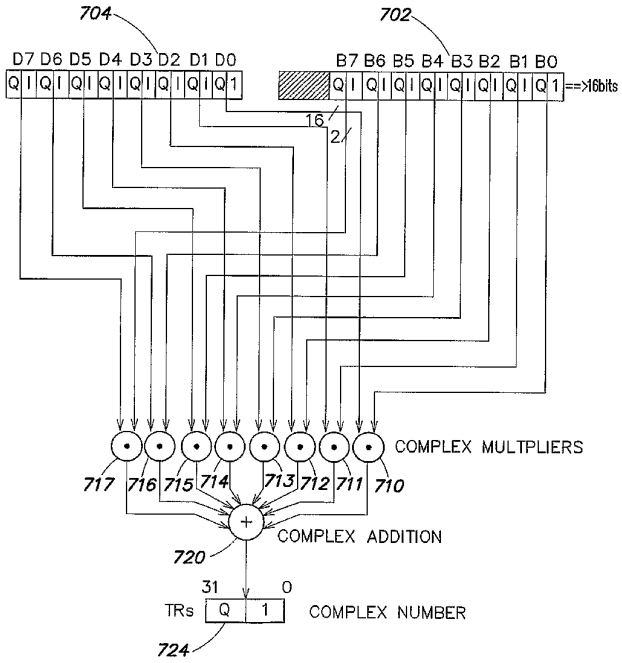

704
702
D7 D6 D5 D4 D3 D2 D1 D0
B7 B6 B5 B4 B3 B2 B1 B0
Q I Q I Q I Q I Q I Q I Q I Q 1
==>16bits
16
2
COMPLEX MULTPLIERS
717 716 715 714 713 712 711 710
COMPLEX ADDITION
720
31 0
TRs Q 1
COMPLEX NUMBER
724

FIG. 9

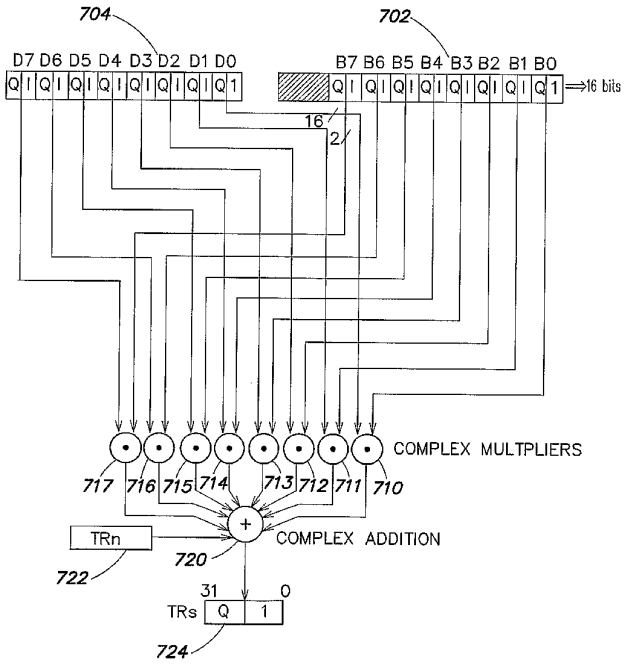

704
702
D7 D6 D5 D4 D3 D2 D1 D0
B7 B6 B5 B4 B3 B2 B1 B0
16 bits
16
2
COMPLEX MULTPLIERS
717 716 715 714 713 712 711 710
TRn
COMPLEX ADDITION
720
722
31
0
TRs
Q
1
724

FIG. 10

```
    602
TR0 = DESPREAD (R3:0, THR1:0)          ; xR3:0 = SDAB Q[J0+=8];;
    604
xTR0 = DESPREAD (R7:4, THR1:0) + TR0; xR7:4 = SDAB Q[J0+=8];;
    606
xTR0 = DESPREAD (R3:0, THR1:0) + TR0; xR3:0 = SDAB Q[J0+=8];;
    608
xTR0 = DESPREAD (R7:4, THR1:0) + TR0; xR7:4 = SDAB Q[J0+=8];
    610
xTHR1:0 = R19:18 (I);;
```

FIG. 11

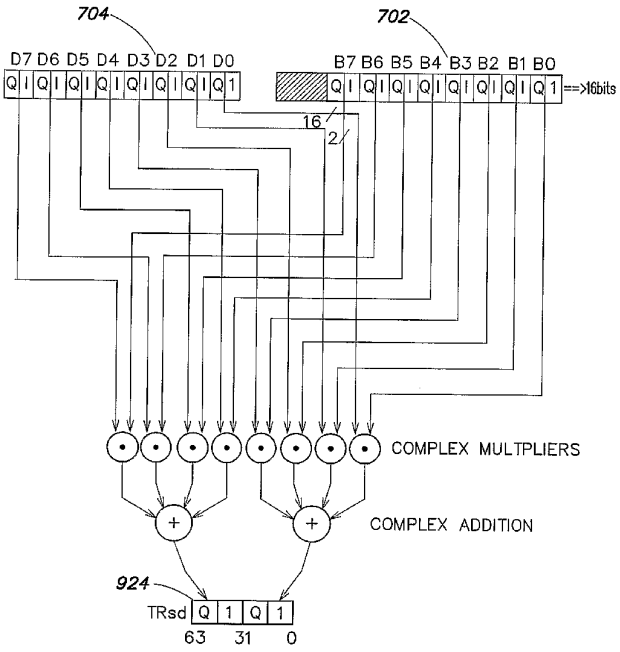

704
702
D7 D6 D5 D4 D3 D2 D1 D0
B7 B6 B5 B4 B3 B2 B1 B0
==>16bits
16
2
COMPLEX MULTPLIERS
COMPLEX ADDITION
924
TRsd
63 31 0

FIG. 12

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/24702

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04B 1/707
US CL : 375/147
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 375/147, 130, 360

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Please See Continuation Sheet

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P	US 6,424,619 B2 (ODENWALDER et al) 23 July 2002 (23.7.2002) figure 4, column 13, line 64 to column 16, line 40.	1-30
X	US 5,930,230 A (ODENWALDER et al) 27 July 1999 (27.7.1999), figure 4, column 5, line 63 to column 6, line 58.	1-30

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 August 2002 (29.08.2002)	12 DEC 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Chi Pham Telephone No. 703-305-4900

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

PCT/US02/24702

Continuation of B. FIELDS SEARCHED Item 3:
EAST
search terms: despreading, complex

フロントページの続き

(72)発明者　フリードマン，ジョセ
アメリカ合衆国　マサチューセッツ州　０２４４６、ブルックライン、アパートメント　２２０、
ビーコン　ストリート　１２８４

Ｆターム(参考) 5K022 EE02　EE14　EE32