(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4335516号
(P4335516)

(45)発行日　平成21年9月30日(2009.9.30)　(24)登録日　平成21年7月3日(2009.7.3)

(51)Int.Cl.			F I		
H04N	7/26	(2006.01)	H04N	7/13	Z
G06F	9/50	(2006.01)	G06F	9/46	465C

請求項の数 10　(全 55 頁)

(21)出願番号	特願2002-324300 (P2002-324300)
(22)出願日	平成14年11月7日 (2002.11.7)
(65)公開番号	特開2003-244699 (P2003-244699A)
(43)公開日	平成15年8月29日 (2003.8.29)
審査請求日	平成17年8月30日 (2005.8.30)
(31)優先権主張番号	特願2001-370153 (P2001-370153)
(32)優先日	平成13年12月4日 (2001.12.4)
(33)優先権主張国	日本国(JP)
(31)優先権主張番号	特願2001-370154 (P2001-370154)
(32)優先日	平成13年12月4日 (2001.12.4)
(33)優先権主張国	日本国(JP)
(31)優先権主張番号	特願2001-381496 (P2001-381496)
(32)優先日	平成13年12月14日 (2001.12.14)
(33)優先権主張国	日本国(JP)

(73)特許権者　000005821
パナソニック株式会社
大阪府門真市大字門真１００６番地
(74)代理人　100098291
弁理士　小笠原　史朗
(72)発明者　花木　真
愛知県名古屋市中区栄２丁目６番１号白川ビル別館５階　株式会社松下電器情報システム名古屋研究所内

審査官　川崎　優

最終頁に続く

(54)【発明の名称】複数のプロセッサを用いた動画像符号化装置およびその方法

(57)【特許請求の範囲】

【請求項１】

並列に動作する複数の処理部と、前記処理部から共通してアクセスされる共有記憶部とを備えた動画像符号化装置であって、

前記共有記憶部は、

独立して処理を適用するために所定の単位に分割された動画像データと、

分割された動画像データの各単位について、符号化処理に必要なデータと、符号化処理の進行状態と、前記処理部の識別情報を管理するための識別管理情報とを記憶し、

各前記処理部は、

前記共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択手段と、

前記次処理選択手段で選択された一単位の動画像データに対して、前記次処理選択手段で選択された処理を実行する処理実行手段と、

前記識別管理情報に基づき、未使用の識別情報を取得する処理部追加手段と、

前記処理部追加手段で取得された識別情報を解放する処理部削除手段とを含み、

前記次処理選択手段と前記処理実行手段とは、前記取得された識別情報が解放されるまでの間、動作し、

前記処理部削除手段は、前記識別管理情報に基づき、前記処理部追加手段で取得された識別情報が失効していると判断したときに、当該識別情報を解放する、動画像符号化装置

。
【請求項２】
前記処理実行手段における処理で生じた異常を検出する異常検出手段と、
前記異常検出手段で異常が検出されたときに、前記次処理選択手段および前記処理実行手段による符号化処理を停止させる処理停止手段とをさらに備えた、請求項１に記載の動画像符号化装置。
【請求項３】
前記異常検出手段は、前記処理実行手段で求めた動画像データの符号化結果に基づき、前記処理実行手段における処理で生じた異常を検出することを特徴とする、請求項２に記載の動画像符号化装置。
【請求項４】
前記異常検出手段は、前記処理実行手段で求めた動画像データの符号化結果のサイズに基づき、前記処理実行手段における処理で生じた異常を検出することを特徴とする、請求項３に記載の動画像符号化装置。
【請求項５】
前記異常検出手段は、前記処理実行手段で求めた動画像データの符号化結果のシンタックスを検査することにより、前記処理実行手段における処理で生じた異常を検出することを特徴とする、請求項３に記載の動画像符号化装置。
【請求項６】
前記異常検出手段は、前記処理実行手段における処理時間に基づき、前記処理実行手段における処理で生じた異常を検出することを特徴とする、請求項１３に記載の動画像符号化装置。
【請求項７】
前記共有記憶部は、分割された動画像データの各単位について、異常検出用のカウント値をさらに記憶しており、
各前記処理部は、前記カウント値を更新するカウント値更新手段をさらに備え、
前記処理実行手段は、前記次処理選択手段で選択された一単位の動画像データに対して前記次処理選択手段で選択された処理を実行する前に、当該一単位の動画像データについての前記カウント値を初期化し、
前記異常検出手段は、前記処理実行手段における処理が完了したときに、前記処理実行手段で初期化したカウント値が所定値以上であるときに、前記処理実行手段における処理で異常が生じたと判断することを特徴とする、請求項６に記載の動画像符号化装置。
【請求項８】
前記異常検出手段は、前記処理実行手段における処理で異常が生じたと判断したときに、異常が生じた旨を、前記共有記憶部に記憶された符号化処理の進行状態に記録し、
前記次処理選択手段は、前記共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについ て次に実行すべき処理とを選択する際に、異常が生じた旨が記録されている処理を優先的に選択することを特徴とする、請求項２に記載の動画像符号化装置。
【請求項９】
並列に動作する複数の処理部と、前記処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、
前記共有記憶部が、
独立して処理を適用するために所定の単位に分割された動画像データと、
分割された動画像データの各単位について、符号化処理に必要なデータと、符号化処理の進行状態と、前記処理部の識別情報を管理するための識別管理情報とを記憶するデータ記憶ステップを行ない、
各前記処理部が、
前記共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処

理選択ステップと、
　前記次処理選択ステップで選択された一単位の動画像データに対して、前記次処理選択ステップで選択された処理を実行する処理実行ステップと、
　前記識別管理情報に基づき、未使用の識別情報を取得する処理部追加ステップと、
　取得された識別情報を解放する処理部削除ステップとを行い、
　前記次処理選択ステップと前記処理実行ステップとは、前記取得された識別情報が解放されるまでの間、動作し、
　前記処理部削除ステップは、前記識別管理情報に基づき、前記取得された識別情報が失効していると判断したときに、当該識別情報を解放する、動画像符号化方法。
【請求項１０】
　並列に動作する複数の処理部と、前記処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、
　前記共有記憶部に、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと、符号化処理の進行状態と、前記処理部の識別情報を管理するための識別管理情報とが記憶された状態で、各前記処理部によって実行されるべく、
　前記コンピュータに、
　前記共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、
　前記次処理選択ステップで選択された一単位の動画像データに対して、前記次処理選択ステップで選択された処理を実行する処理実行ステップと、
　前記識別管理情報に基づき、未使用の識別情報を取得する処理部追加ステップと、
　取得された識別情報を解放する処理部削除ステップとを実行させ、
　前記次処理選択ステップと前記処理実行ステップとは、前記取得された識別情報が解放されるまでの間、動作し、
　前記処理部削除ステップは、前記識別管理情報に基づき、前記取得された識別情報が失効していると判断したときに、当該識別情報を解放する、という機能を実現させるための、プログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、動画像の符号化装置および符号化方法に関し、より特定的には、複数のプロセッサを用いて符号化処理を並列に実行する動画像符号化装置および動画像符号化方法に関する。
【０００２】
【従来の技術】
従来から、ＭＰＥＧ（Moving Picture Experts Group）などに準拠した動画像符号化処理を、複数のプロセッサを用いて並列に実行し、処理時間を短縮する方法が知られている（例えば、特許文献１）。図３４および図３５は、それぞれ、特許文献１に記載された動画像符号化装置における、親プロセッサおよび子プロセッサの処理を示すフローチャートである。この符号化装置では、装置に含まれる複数のプロセッサのうち、選択された１台のプロセッサが親プロセッサとして動作し、他のプロセッサは子プロセッサとして動作する。各子プロセッサは、ＭＰＥＧ符号化処理のうちで並列に実行可能な処理を並列に実行する。例えば図３５に示す例では、各子プロセッサは、マクロブロックを符号化し、可変長符号化し、局所復号化する処理（ステップＳ３２、Ｓ３４およびＳ３５）を並列に実行する。親プロセッサは、ヘッダの生成（ステップＳ１１～Ｓ１４）や子プロセッサの起動（ステップＳ１５）など、主に符号化処理の実行を制御する。この際、親プロセッサは、自らも子プロセッサと同様の符号化処理を実行してもよい。このように親プロセッサによる実行制御のもとで、複数の子プロセッサを並列に動作させることにより、処理時間を短縮

することができる。
【０００３】
【特許文献１】
特開２０００－３００４７号公報
【０００４】
【発明が解決しようとする課題】
しかしながら、上記符号化装置には、符号化処理の実行制御、符号化処理の内容の切り替え、エラー処理、および、記憶装置の構成などの点において、次のような問題がある。まず、符号化処理の実行制御について言えば、第１に、符号化処理の実行制御が親プロセッサによって行われるため、親プロセッサによる逐次処理が子プロセッサによる並列処理を阻害するという問題がある。第２に、符号化処理中にプロセッサを動的に追加および削除する手段が設けられていないため、プロセッサが１台故障しただけでも、装置全体の動作が停止するという問題がある。第３に、符号化処理の実行制御は親プロセッサと子プロセッサとの間で同期を取ることによって行われるため、符号化処理中にプロセッサを追加または削除するには、追加／削除対象の子プロセッサと親プロセッサとの間で通信を行うか、追加／削除対象の子プロセッサに符号化処理の実行制御を一時的に行わせることが必要となり、これに伴い処理性能が一時的に低下するという問題がある。第４に、親プロセッサと子プロセッサとでは処理内容が異なるため、符号化処理の実行制御が複雑になるという問題もある。
【０００５】
次に、符号化処理の内容の切り替えについて言えば、一般に、動画像符号化装置では、符号化対象となる動画像データの特性などに応じて、符号化処理の内容を切り替えられることが望ましい。例えば、動きの激しい動画像データには精度の高い動きベクトル探索を、動きの乏しい動画像データには簡易な動きベクトル探索をそれぞれ適用すれば、前者については再生画像を高画質化し、後者については処理時間を短縮することができる。
【０００６】
ところが、上記符号化装置では、各プロセッサで実行される処理は固定化されているため、動画像データの特性に応じて符号化処理の内容を切り替えるためには、切り替えられる可能性があるすべての処理を子プロセッサに搭載した上で、親プロセッサが子プロセッサにおける処理の内容を動的に切り替える必要がある。しかし、この方法は、子プロセッサに搭載される機能の大半が無駄になり、親プロセッサと子プロセッサとの間の同期制御が複雑になるので、現実的ではない。また、符号化処理の内容を変更したり、新たな機能を追加するときには、装置本来の目的である符号化処理を中断し、各プロセッサのプログラムをすべて更新する必要が生じる。
【０００７】
次に、エラー処理について言えば、上記符号化装置では、親プロセッサと複数の子プロセッサとが、それぞれ、異なるマクロブロックに対して符号化処理を行うため、符号化結果として生成されるビットストリームには、各プロセッサで求めた結果が混在して出現する。例えば、親プロセッサがスライス層以上のヘッダ生成処理とマクロブロック符号化処理とを行い、「子１」「子２」の２台の子プロセッサがマクロブロック符号化処理を行った場合には、図３６に示すように、３台のプロセッサで求めた結果が混在したビットストリームが生成される。
【０００８】
上記符号化装置では、各プロセッサで求めた結果は、検査することなく連結され、１本のビットストリームとなるため、いずれかのプロセッサで異常が発生すると、生成されたビットストリームを全く使用できないという問題が生じる。例えば、上述した例において「子２」のプロセッサで異常が発生すると、図３６で斜線を付した部分に誤りが生じる。ところが、各プロセッサは任意に選択したマクロブロックを符号化し、各プロセッサにおける処理時間も一定ではないので、生成されたビットストリームにおける誤りの有無や、誤りの位置などを容易に求めることはできない。したがって、プロセッサで異常が発生する

と、生成されたビットストリームは使用できなくなり、正しいビットストリームを得るためには、プロセッサに生じた異常の原因を除去した後に、再び符号化処理を行う必要がある。

【０００９】

また、上記符号化装置では、生成されたビットストリームに基づき、いずれのプロセッサに異常が生じたかを容易に判断することができないため、異常プロセッサを検出するためには、符号化処理を一旦中断した上で、プロセッサごとに異常検出処理を行う必要がある。したがって、装置の保守には、プロセッサの個数に比例した時間が必要となる。また、子プロセッサを追加して符号化処理の高速化を図る場合、プロセッサの追加に伴い装置全体の故障率が上昇するので、信頼性の高い高価なプロセッサを使用しなければ、異常発生時の再符号化処理を含めた実際の処理時間が長くなってしまう。さらに、親プロセッサで異常が発生すると、符号化処理全体が停止し、代替の親プロセッサが準備できるまで符号化処理を再開できないという問題もある。

【００１０】

次に、記憶装置の構成について言えば、上記符号化装置では、各プロセッサからアクセスされる記憶装置には特段の工夫がなされていない。このため、複数のプロセッサが記憶装置に対して並列にアクセスすると、データアクセスがボトルネックになり、動画像符号化処理の速度が低下するという問題がある。

【００１１】

それ故に、本発明の第１の目的は、符号化処理の実行制御を行うプロセッサを備えることなく、符号化処理の実行中でもプロセッサを追加および削除できる動画像符号化装置および方法を提供することである。本発明の第２の目的は、実行中の符号化処理に影響を及ぼすことなく、符号化処理中に符号化処理の内容を切り替え、追加および変更することができる動画像符号化装置および方法を提供することである。本発明の第３の目的は、異常プロセッサを符号化処理中に検出することにより、プロセッサの修理や交換を容易に行え、少なくとも１個のプロセッサが正常に動作していれば符号化処理を継続できる動画像符号化装置および方法を提供することである。本発明の第４の目的は、複数のプロセッサから並列に高速アクセス可能な記憶装置を備え、符号化処理を高速に行える動画像符号化装置および方法を提供することである。

【００１２】

【課題を解決するための手段および発明の効果】

第１の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えた動画像符号化装置であって、共有記憶部は、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とを記憶し、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択手段と、次処理選択手段で選択された一単位の動画像データに対して、次処理選択手段で選択された処理を実行する処理実行手段とを含む。

このような第１の発明によれば、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、次の処理を選択して実行する。各処理部における処理の選択と実行とは、他の処理部の動作とは独立して、任意のタイミングで行われる。このように符号化処理の実行制御が複数の処理部によって分散して行われるので、符号化処理の実行制御を行う処理部を備える必要がなく、複数の処理部の間で同期を取る必要もない。したがって、逐次的に行われる符号化処理の実行制御によって、並列処理による高速化の効果が損われることがなく、装置の性能を向上させることができる。また、各処理部はいずれも同じ構成を有していればよいため、符号化処理の実行制御を簡素化することができる。

【００１３】

第２の発明は、第１の発明において、次処理選択手段は、動画像符号化処理に含まれる複数の処理のうちから一の処理を、次に実行すべき処理として選択することを特徴とする。

このような第２の発明によれば、各処理部は、動画像符号化処理の一部を実行するごとに、次に実行すべき処理を選択する。このように動画像符号化処理を複数の処理に分割し、各処理部が分割された処理をそれぞれ実行することにより、各処理部の負荷を均等化し、各処理部が動画像符号化処理の全体を実行する場合と比べて、装置の性能を向上させることができる。

【００１４】

第３の発明は、第１の発明において、各処理部は、各処理部における処理に必要なデータを記憶する個別記憶手段をさらに備え、処理実行手段は、次処理選択手段で選択された処理を実行する前に、当該処理に必要なデータを共有記憶部から読み出して個別記憶手段に記憶させ、個別記憶手段に記憶されたデータに対して当該処理を実行し、当該処理で得られたデータを個別記憶手段から読み出して共有記憶部に記憶させることを特徴とする。

このような第３の発明によれば、各処理部は、処理に必要なデータを共有記憶部から個別記憶手段に複写して、個別記憶手段に複写されたデータに対して選択した処理を実行する。これにより、処理部から共有記憶部に対するアクセスが減少し、各処理部におけるデータ待ち時間が短縮されるので、装置の性能を向上させることができる。

【００１５】

第４の発明は、第３の発明において、処理実行手段は、次処理選択手段で選択された処理を実行する前に、当該処理に必要なデータが個別記憶手段に記憶されていない場合に、当該処理に必要なデータを共有記憶部から読み出して個別記憶手段に記憶させることを特徴とする。

このような第４の発明によれば、各処理部は、個別記憶手段に必要なデータが記憶されている場合には、データ取得のために共有記憶部にアクセスしない。これにより、処理部から共有記憶部に対するアクセスがさらに減少するので、装置の性能をさらに向上させることができる。

【００１６】

第５の発明は、第１の発明において、共有記憶部は、処理部の識別情報を管理するための識別管理情報をさらに記憶し、各処理部は、識別管理情報に基づき、未使用の識別情報を取得する処理部追加手段と、処理部追加手段で取得された識別情報を解放する処理部削除手段とをさらに含み、次処理選択手段と処理実行手段とは、取得された識別情報が解放されるまでの間、動作することを特徴とする。

このような第５の発明によれば、各処理部は、共有記憶部に記憶された識別管理情報に基づき識別情報の取得と解放とを行い、識別情報を有している間だけ動作する。処理部による識別情報の取得と解放とは、他の処理部の動作とは独立して、任意のタイミングで行われる。したがって、他の処理部が符号化処理を実行中であっても、他の処理部に影響を及ぼすことなく、任意のタイミングで装置に処理部を追加および削除することができる。このため、処理部で障害が発生した場合などに、装置全体の符号化処理を中断することなく、処理部を保守することができる。

【００１７】

第６の発明は、第５の発明において、処理部削除手段は、識別管理情報に基づき、処理部追加手段で取得された識別情報が失効していると判断したときに、当該識別情報を解放することを特徴とする。

このような第６の発明によれば、各処理部は、共有記憶部に記憶された識別管理情報に基づき、必要に応じて識別情報を解放し、処理の実行を停止する。処理部による識別情報を解放すべきか否かの判断は、他の処理部の動作とは独立して、任意のタイミングで行われる。したがって、他の処理部が符号化処理を実行中であっても、他の処理部に影響を及ぼすことなく、各処理部による判断に基づき、処理部を装置から削除することができる。このため、処理部で障害が発生した場合などに、装置全体の符号化処理を中断することなく、障害が発生した処理部を装置から削除することができる。

【００１８】

第７の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有

記憶部とを備えた動画像符号化装置であって、共有記憶部は、符号化対象の動画像データと、動画像データの符号化処理に必要なデータと、処理部の識別情報を管理するための識別管理情報とを記憶し、各処理部は、識別管理情報に基づき、未使用の識別情報を取得する処理部追加手段と、処理部追加手段で取得された識別情報を解放する処理部削除手段と、取得された識別情報が解放されるまでの間、共有記憶部に記憶された動画像データに対して符号化処理を実行する符号化実行手段とを含む。

このような第７の発明によれば、各処理部は、共有記憶部に記憶された識別管理情報に基づき識別情報の取得と解放とを行い、識別情報を有している間だけ動作する。処理部による識別情報の取得と解放とは、他の処理部の動作とは独立して、任意のタイミングで行われる。したがって、他の処理部が符号化処理を実行中であっても、他の処理部に影響を及ぼすことなく、任意のタイミングで装置に処理部を追加および削除することができる。このため、処理部で障害が発生した場合などに、装置全体の符号化処理を中断することなく、処理部を保守することができる。

【００１９】

第８の発明は、第７の発明において、処理部削除手段は、識別管理情報に基づき、処理部追加手段で取得された識別情報が失効していると判断したときに、当該識別情報を解放することを特徴とする。

このような第８の発明によれば、各処理部は、共有記憶部に記憶された識別管理情報に基づき、必要に応じて識別情報を解放し、処理の実行を停止する。処理部による識別情報を解放すべきか否かの判断は、他の処理部の動作とは独立して、任意のタイミングで行われる。したがって、他の処理部が符号化処理を実行中であっても、他の処理部に影響を及ぼすことなく、各処理部による判断に基づき、処理部を装置から削除することができる。このため、処理部で障害が発生した場合などに、装置全体の符号化処理を中断することなく、障害が発生した処理部を装置から削除することができる。

【００２０】

第９の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えた動画像符号化装置であって、共有記憶部は、処理部に転送され、処理部で実行される複数のプログラムと、複数のプログラムの機能を示す機能情報とを記憶し、各処理部は、動画像符号化処理に含まれる複数の処理のうちから一の処理を、次に実行すべき処理として選択する次処理選択手段と、共有記憶部に記憶された機能情報に基づき、共有記憶部に記憶された複数のプログラムから、次処理選択手段で選択された処理を実行するためのプログラムを選択し、選択したプログラムを共有記憶部から転送するプログラム更新手段と、共有記憶部から転送されたプログラムを記憶する個別記憶手段と、個別記憶手段に記憶されたプログラムを用いて、次処理選択手段で選択された処理を実行する処理実行手段とを含む。

このような第９の発明によれば、各処理部は、共有記憶部に記憶された機能情報に基づき、次の処理に必要なプログラムを選択し、選択したプログラムを転送して実行する。各処理部におけるプログラムの選択、転送および実行は、他の処理部の動作とは独立して、任意のタイミングで行われる。このため、各処理部は、他の処理部の動作に影響を及ぼすことなく、次の処理に必要なプログラムを選択し、実行することができる。したがって、実行中の符号化処理に影響を及ぼすことなく、符号化処理中に符号化処理の内容を切り替え、追加および変更することができる。

【００２１】

第１０の発明は、第９の発明において、共有記憶部は、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とをさらに記憶し、次処理選択手段は、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択し、処理実行手段は、次処理選択手段で選択された一単位の動画像データに対して、次処理選択手段で選択された処理を実行することを特徴とする。

このような第１０の発明によれば、各処理部は、共有記憶部に記憶された符号化処理の進行状況に基づき、次の処理を選択して実行する。各処理部における処理の選択と実行とは、他の処理部の動作とは独立して、任意のタイミングで行われる。このように符号化処理の実行制御が複数の処理部によって分散して行われるので、符号化処理の実行制御を行う処理部を備えることなく、動画像符号化処理を行うことができる。

【００２２】

第１１の発明は、第９の発明において、プログラム更新手段は、共有記憶部に記憶された機能情報と次処理選択手段で選択された処理の機能とを比較して、当該処理を実行するためのプログラムを選択することを特徴とする。

このような第１１の発明によれば、各処理部は、共有記憶部に記憶された機能情報と次の処理に必要な機能とを比較して、転送すべきプログラムを選択する。したがって、プログラム選択の際には、共有記憶部に記憶されたデータのうち機能情報のみが使用され、他の処理部は機能情報以外のデータに自由にアクセスできる。したがって、実行中の符号化処理に影響を及ぼすことなく、次の処理に必要なプログラムを選択することができる。

【００２３】

第１２の発明は、第９の発明において、プログラム更新手段は、個別記憶手段に記憶されたプログラムの機能と次処理選択手段で選択された処理の機能とを比較して、選択したプログラムを共有記憶部から転送するか否かを切り替えることを特徴とする。

このような第１２の発明によれば、処理部は、以前に取得したプログラムを用いて次の処理を実行できる場合には、共有記憶部から個別記憶手段にプログラムを転送しない。これにより、処理部から共有記憶部に対するアクセスが減少し、各処理部におけるプログラム転送待ち時間が短縮されるので、装置の性能を向上させることができる。

【００２４】

第１３の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えた動画像符号化装置であって、共有記憶部は、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とを記憶し、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択手段と、次処理選択手段で選択された一単位の動画像データに対して、次処理選択手段で選択された処理を実行する処理実行手段と、処理実行手段における処理で生じた異常を検出する異常検出手段と、異常検出手段で異常が検出されたときに、次処理選択手段および処理実行手段による符号化処理を停止させる処理停止手段とを備える。

このような第１３の発明によれば、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、次の処理を選択して実行するとともに、選択した処理を実行した際に異常が生じたときには自ら停止する。各処理部における処理の選択、実行、異常検出および停止は、他の処理部とは独立して、任意のタイミングで行われる。したがって、符号化処理の実行制御を行う処理部を備えることなく、異常が生じた処理部を符号化処理中に検出することができる。よって、符号化処理中であっても処理部の修理や交換を容易に行うことができ、少なくとも１個の処理部が正常に動作していれば符号化処理を継続することができる。

【００２５】

第１４の発明は、第１３の発明において、異常検出手段は、処理実行手段で求めた動画像データの符号化結果に基づき、処理実行手段における処理で生じた異常を検出することを特徴とする。

このような第１４の発明によれば、動画像符号化処理で生成されたビットストリームに基づき、異常が生じた処理部が検出される。一般に動画像符号化処理で異常が生じたときには、ビットストリームに異常が生じるので、これにより、異常が生じた処理部を符号化処理中に容易に検出することができる。

【００２６】

第１５の発明は、第１４の発明において、異常検出手段は、処理実行手段で求めた動画像データの符号化結果のサイズに基づき、処理実行手段における処理で生じた異常を検出することを特徴とする。
このような第１５の発明によれば、動画像符号化処理で生成されたビットストリームのサイズに基づき、異常が生じた処理部が検出される。動画像符号化処理で異常が生じたときには、ビットストリームのサイズに異常が生じる場合があるので、これにより、異常が生じた処理部を容易に検出することができる。
【００２７】
第１６の発明は、第１４の発明において、異常検出手段は、処理実行手段で求めた動画像データの符号化結果のシンタックスを検査することにより、処理実行手段における処理で生じた異常を検出することを特徴とする。
このような第１６の発明によれば、動画像符号化処理で生成されたビットストリームのシンタックスを検査することにより、異常が生じた処理部が検出される。動画像符号化処理で異常が生じたときには、ビットストリームのシンタックスに異常が生じる場合があるので、これにより、異常が生じた処理部を容易に検出することができる。
【００２８】
第１７の発明は、第１３の発明において、異常検出手段は、処理実行手段における処理時間に基づき、処理実行手段における処理で生じた異常を検出することを特徴とする。
このような第１７の発明によれば、各処理の所要時間に基づき、異常が生じた処理部が検出される。動画像符号化処理に含まれる処理には、定型的な処理や、所要時間を予想可能な処理が多く含まれるので、これにより、異常が生じた処理部を容易に検出することができる。
【００２９】
第１８の発明は、第１７の発明において、共有記憶部は、分割された動画像データの各単位について、異常検出用のカウント値をさらに記憶しており、各処理部は、カウント値を更新するカウント値更新手段をさらに備え、処理実行手段は、次処理選択手段で選択された一単位の動画像データに対して次処理選択手段で選択された処理を実行する前に、当該一単位の動画像データについてのカウント値を初期化し、異常検出手段は、処理実行手段における処理が完了したときに、処理実行手段で初期化したカウント値が所定値以上であるときに、処理実行手段における処理で異常が生じたと判断することを特徴とする。
このような第１８の発明によれば、共有記憶部に記憶されたカウント値が各処理部によって更新され、これを用いて異常が生じた処理部が検出される。したがって、各処理部でタイマーを起動することなく処理の所要時間の概算値を算出し、異常が生じた処理部を検出することができる。
【００３０】
第１９の発明は、第１３の発明において、異常検出手段は、処理実行手段における処理で異常が生じたと判断したときに、異常が生じた旨を、共有記憶部に記憶された符号化処理の進行状態に記録し、次処理選択手段は、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する際に、異常が生じた旨が記録されている処理を優先的に選択することを特徴とする。
このような第１９の発明によれば、処理部で異常が生じた旨は共有記憶部に記録され、各処理部は異常が生じた処理を優先的に実行する。このように異常が生じた処理を他の処理部が早期に実行することにより、動画像符号化処理を安全かつ確実に実行することができる。
【００３１】
第２０の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えた動画像符号化装置であって、共有記憶部は、独立して処理を適用するために所定の単位に分割された動画像データを分散して記憶する複数の画像記憶部と、各画像記憶部とは独立してアクセスできるように構成され、分割された動画像データの各単

位について、符号化処理に必要なデータと処理の進行状態とを記憶するパラメータ記憶部とを含み、各処理部は、パラメータ記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択手段と、次処理選択手段で選択された一単位の動画像データに対して、次処理選択手段で選択された処理を実行する処理実行手段とを含む。

このような第２０の発明によれば、動画像データは複数の画像記憶部に分散して記憶され、各処理部は分散して記憶された動画像データにアクセスする。そこで、処理部からのアクセスが競合しないように動画像データを分散して記憶することにより、処理部からのアクセスが符号化処理のボトルネックになることを防止することができる。よって、動画像符号化処理を高速に行うことができる。

【００３２】

第２１の発明は、第２０の発明において、符号化対象の動画像データを入力する画像入力部をさらに備え、次処理選択手段は、符号化処理の過程において少なくとも２段階に処理の選択方法を切り替え、第１段階では、次に実行すべき処理として動きベクトル探索を含む処理を選択し、第１段階に続く第２段階では、第１段階で求めた動きベクトル探索結果を利用することとして、次に実行すべき処理として既知の動きベクトル探索を含まない処理を選択し、処理実行手段は、次処理選択手段が第１段階にあるときは、次処理選択手段で選択された一単位の動画像データを画像入力部から読み出して、次処理選択手段で選択された処理を実行するとともに、読み出した動画像データを複数の画像記憶部のうち一の画像記憶部に書き込み、次処理選択手段が第２段階にあるときは、次処理選択手段で選択された一単位の動画像データを画像記憶部から読み出して、次処理選択手段で選択された処理を実行することを特徴とする。

このような第２１の発明によれば、画像入力部から入力された動画像データは、動きベクトル探索実行中に複数の画像記憶部に分散して記憶される。その後、処理部は、分散して記憶された動画像データに対して符号化処理を実行する。動きベクトル探索では、計算が処理のボトルネックになるのに対して、動きベクトル探索後の処理では、入出力が処理のボトルネックになる。そこで、入出力に余裕がある動きベクトル探索実行中に、動画像データを複数の画像記憶部に分散して記憶させることにより、入出力に余裕がない動きベクトル探索後の処理を高速に行うための準備をする。これにより、動きベクトル探索後の処理において入出力が処理のボトルネックになることを防止することができる。よって、動画像符号化処理を高速に行うことができる。

【００３３】

第２２の発明は、第２１の発明において、次処理選択手段は、画像入力部から入力されるすべての動画像データについて第１段階における処理を選択した後に、第１段階から第２段階に処理の選択方法を切り替えることを特徴とする。

このような第２２の発明によれば、すべての動画像データについて動きベクトル探索を実行した後に、動きベクトル探索後の処理を開始することができる。

【００３４】

第２３の発明は、第２０の発明において、共有記憶部は、各画像記憶部およびパラメータ記憶部とは独立してアクセスできるように構成され、動画像データの符号化結果を記憶する符号化結果記憶部をさらに含み、処理実行手段は、動画像データの符号化結果を新たに求めたときに、求めた部分より前の部分の符号化結果がすべて符号化結果記憶部に記憶されている場合には、符号化結果記憶部に記憶された符号化結果に新たに求めた符号化結果を連結し、それ以外の場合には、新たに求めた符号化結果を、記憶された符号化結果と連結することなく、符号化結果記憶部に書き込むことを特徴とする。

このような第２３の発明によれば、処理部は、新たに求めた符号化結果が符号化結果記憶部に記憶された符号化結果と連結可能か否かを判断し、連結可能である場合には連結し、連結不可である場合は新たに求めた符号化結果を後で連結すべきものとして符号化結果記憶部に書き込む。後で連結すべきとされた符号化結果の連結処理は、他の処理部によって実行される。したがって、符号化結果の連結順序と生成順序とが異なる場合でも、処理部

は、他の処理部が符号化結果を生成するのを待つことなく、他の実行可能な処理を開始することができる。よって、動画像符号化処理を高速に行うことができる。

【００３５】

第２４の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、共有記憶部が、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とを記憶するデータ記憶ステップを行ない、各処理部が、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップとを行う。

このような第２４の発明によれば、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、次の処理を選択して実行する。各処理部における処理の選択と実行とは、他の処理部の動作とは独立して、任意のタイミングで行われる。このように符号化処理の実行制御が複数の処理部によって分散して行われるので、符号化処理の実行制御を行う処理部を備える必要がなく、複数の処理部の間で同期を取る必要もない。したがって、逐次的に行われる符号化処理の実行制御によって、並列処理による高速化の効果が損われることがなく、装置の性能を向上させることができる。また、各処理部はいずれも同じ構成を有していればよいため、符号化処理の実行制御を簡素化することができる。

【００３６】

第２５の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、共有記憶部が、符号化対象の動画像データと、動画像データの符号化処理に必要なデータと、処理部の識別情報を管理するための識別管理情報とを記憶するデータ記憶ステップを行い、各処理部が、識別管理情報に基づき、未使用の識別情報を取得する処理部追加ステップと、処理部追加ステップで取得された識別情報を解放する処理部削除ステップと、取得された識別情報が解放されるまでの間、共有記憶部に記憶された動画像データに対して符号化処理を実行する符号化実行ステップとを行う。

このような第２５の発明によれば、各処理部は、共有記憶部に記憶された識別管理情報に基づき識別情報の取得と解放とを行い、識別情報を有している間だけ動作する。処理部による識別情報の取得と解放とは、他の処理部の動作とは独立して、任意のタイミングで行われる。したがって、他の処理部が符号化処理を実行中であっても、他の処理部に影響を及ぼすことなく、任意のタイミングで装置に処理部を追加および削除することができる。このため、処理部で障害が発生した場合などに、装置全体の符号化処理を中断することなく、処理部を保守することができる。

【００３７】

第２６の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、共有記憶部が、処理部に転送され、処理部で実行される複数のプログラムと、複数のプログラムの機能を示す機能情報とを記憶するプログラム記憶ステップを行い、各処理部が、動画像符号化処理に含まれる複数の処理のうちから一の処理を、次に実行すべき処理として選択する次処理選択ステップと、共有記憶部に記憶された機能情報に基づき、共有記憶部に記憶された複数のプログラムから、次処理選択ステップで選択された処理を実行するためのプログラムを選択し、選択したプログラムを共有記憶部から転送するプログラム更新ステップと、共有記憶部から転送されたプログラムを記憶する個別記憶ステップと、個別記憶ステップで記憶されたプログラムを用いて、次処理選択ステップで選択された処理を実行する処理実行ステップとを行う。

このような第２６の発明によれば、各処理部は、共有記憶部に記憶された機能情報に基づき、次の処理に必要なプログラムを選択し、選択したプログラムを転送して実行する。各処理部におけるプログラムの選択、転送および実行は、他の処理部の動作とは独立して、

任意のタイミングで行われる。このため、各処理部は、他の処理部の動作に影響を及ぼすことなく、次の処理に必要なプログラムを選択して、実行することができる。したがって、実行中の符号化処理に影響を及ぼすことなく、符号化処理中に符号化処理の内容を切り替え、追加および変更することができる。

【００３８】

第２７の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、共有記憶部が、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とを記憶するデータ記憶ステップを行い、各処理部が、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップと、処理実行ステップにおける処理で生じた異常を検出する異常検出ステップと、異常検出ステップで異常が検出されたときに、次処理選択ステップおよび処理実行ステップによる符号化処理を停止させる処理停止ステップとを行う。

このような第２７の発明によれば、各処理部は、共有記憶部に記憶された符号化処理の進行状態に基づき、次の処理を選択して実行するとともに、選択した処理を実行した際に異常が生じたときには自ら停止する。各処理部における処理の選択、実行、異常検出および停止は、他の処理部とは独立して、任意のタイミングで行われる。したがって、符号化処理の実行制御を行う処理部を備えることなく、異常が生じた処理部を符号化処理中に検出することができる。よって、符号化処理中であっても処理部の修理や交換を容易に行うことができ、少なくとも１個の処理部が正常に動作していれば符号化処理を継続することができる。

【００３９】

第２８の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを用いた動画像符号化方法であって、共有記憶部が、独立して処理を適用するために所定の単位に分割された動画像データを分散して記憶する画像記憶ステップと、画像記憶ステップで記憶された動画像データとは独立してアクセスできるように、分割された動画像データの各単位について、符号化処理に必要なデータと処理の進行状態とを記憶するパラメータ記憶ステップとを行い、各処理部が、パラメータ記憶ステップで記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップとを行う。

このような第２８の発明によれば、動画像データは複数の画像記憶部に分散して記憶され、各処理部は分散して記憶された動画像データにアクセスする。そこで、処理部からのアクセスが競合しないように動画像データを分散して記憶することにより、処理部からのアクセスが符号化処理のボトルネックになることを防止することができる。よって、動画像符号化処理を高速に行うことができる。

【００４０】

第２９の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、共有記憶部に、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とが記憶された状態で、各処理部によって実行されるべく、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップとを備える。

【００４１】
第３０の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、共有記憶部に、符号化対象の動画像データと、動画像データの符号化処理に必要なデータと、処理部の識別情報を管理するための識別管理情報とが記憶された状態で、各処理部によって実行されるべく、識別管理情報に基づき、未使用の識別情報を取得する処理部追加ステップと、処理部追加ステップで取得された識別情報を解放する処理部削除ステップと、取得された識別情報が解放されるまでの間、共有記憶部に記憶された動画像データに対して符号化処理を実行する符号化実行ステップとを備える。
【００４２】
第３１の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、共有記憶部に、処理部に転送され、処理部で実行される複数の処理プログラムと、複数の処理プログラムの機能を示す機能情報とが記憶された状態で、各処理部によって実行されるべく、動画像符号化処理に含まれる複数の処理のうちから一の処理を、次に実行すべき処理として選択する次処理選択ステップと、共有記憶部に記憶された機能情報に基づき、共有記憶部に記憶された複数の処理プログラムから、次処理選択ステップで選択された処理を実行するための処理プログラムを選択し、選択した処理プログラムを共有記憶部から転送するプログラム更新ステップと、共有記憶部から転送された処理プログラムを記憶する個別記憶ステップと、個別記憶ステップで記憶された処理プログラムを用いて、次処理選択ステップで選択された処理を実行する処理実行ステップとを備える。
【００４３】
第３２の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、共有記憶部に、独立して処理を適用するために所定の単位に分割された動画像データと、分割された動画像データの各単位について、符号化処理に必要なデータと符号化処理の進行状態とが記憶された状態で、各処理部によって実行されるべく、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップと、処理実行ステップにおける処理で生じた異常を検出する異常検出ステップと、異常検出ステップで異常が検出されたときに、次処理選択ステップおよび処理実行ステップによる符号化処理を停止させる処理停止ステップとを備える。
【００４４】
第３３の発明は、並列に動作する複数の処理部と、処理部から共通してアクセスされる共有記憶部とを備えたコンピュータを動画像符号化装置として機能させるためのプログラムであって、共有記憶部に、独立して処理を適用するために所定の単位に分割された動画像データが分散して記憶され、さらに、動画像データとは独立してアクセスできるように、分割された動画像データの各単位について、符号化処理に必要なデータと処理の進行状態とが記憶された状態で、各処理部によって実行されるべく、共有記憶部に記憶された符号化処理の進行状態に基づき、分割された一単位の動画像データと、当該一単位の動画像データについて次に実行すべき処理とを選択する次処理選択ステップと、次処理選択ステップで選択された一単位の動画像データに対して、次処理選択ステップで選択された処理を実行する処理実行ステップとを備える。
【００４５】
【発明の実施の形態】
（動画像符号化装置のシステム構成）
図１は、本発明の第１ないし第５の実施形態に係る動画像符号化装置の構成を示すブロック図である。図１に示す動画像符号化装置は、ｎ個のプロセッサ１－１～ｎ、共有記憶部

２、システム管理部３、ネットワーク４、画像入力部１１、画像出力部１２、および、ストリーム記憶部１３を備える。ｎ個のプロセッサ１－１～ｎ、共有記憶部２、および、システム管理部３は、いずれも、装置内に設けられたネットワーク４に接続される。この動画像符号化装置は、ｎ個のプロセッサ１－１～ｎを用いて、共有記憶部２に蓄積された動画像データを並列に符号化する。

【００４６】

プロセッサ１には、例えば、パーソナルコンピュータが使用される。また、プロセッサ１は、動画像符号化専用のコンピュータや、汎用のエンジニアリングワークステーションなどであってもよい。共有記憶部２には、例えば、ハードディスク装置などが使用される。ネットワーク４には、例えば、バスやローカルエリアネットワークなどが使用される。

【００４７】

共有記憶部２は、符号化処理に必要なプログラムとデータとを蓄積している。符号化処理実行前の共有記憶部２には、符号化処理用プログラムと、符号化対象の動画像データと、符号化処理中に参照される制御情報とが蓄積されている。共有記憶部２に蓄積されたプログラムは、各プロセッサ１にダウンロードされる。各プロセッサ１は、ダウンロードされたプログラムに従って、共有記憶部２から動画像データと制御情報とを読み出して所定の符号化処理を行い、符号化処理の中間結果や符号化結果のビットストリームを共有記憶部２に書き込む。符号化処理の中間結果には、例えば、ＧＯＰ（Group of Pictures ）構造や動きベクトルなどが含まれる。

【００４８】

図２は、プロセッサ１の詳細な構成を示すブロック図である。プロセッサ１は、図２に示すように、ＣＰＵ２１、ネットワークインターフェイス部２２、ＲＡＭ２３、および、ローカルバス２４を備える。ＣＰＵ２１、ネットワークインターフェイス部２２、および、ＲＡＭ２３は、いずれも、プロセッサ１内のローカルバス２４に接続される。図１に示すｎ個のプロセッサ１－１～ｎは、いずれも、図２に示す構成を有するものとする。

【００４９】

ＣＰＵ２１は、プロセッサ１の演算処理装置であり、ＲＡＭ２３に蓄積された符号化処理用プログラム３２に従って動作する。ネットワークインターフェイス部２２は、ネットワーク４とローカルバス２４とを接続する。ＲＡＭ２３は、各プロセッサ１ごとに設けられた記憶装置であり、プロセッサ識別情報３１、符号化処理用プログラム３２、および、符号化処理用データ３３を蓄積している。プロセッサ識別情報３１は、各プロセッサ１を識別するために割り当てられた識別情報である。符号化処理用プログラム３２は、共有記憶部２からダウンロードされたプログラムである。符号化処理用データ３３は、各プロセッサ１における符号化処理に必要なデータである。符号化処理用データ３３には、符号化対象の動画像データ、符号化処理中に参照される制御情報、符号化処理の中間結果、および、符号化結果のビットストリームなどが含まれる。

【００５０】

再び図１に戻ると、システム管理部３は、コマンド入力のための入力部（図示せず）を備え、入力されたコマンドに従いデータ入出力などの処理を行う。例えば、システム管理部３は、符号化対象の動画像データを画像入力部１１から共有記憶部２へ転送する処理や、符号化結果のビットストリームを共有記憶部２から画像出力部１２およびストリーム記憶部１３へ転送する処理などを行う。図１では、システム管理部３をｎ個のプロセッサ１－１～ｎとは別に設けたが、システム管理部３をｎ個のプロセッサ１－１～ｎのいずれかに設けてもよい。

【００５１】

画像入力部１１は、デジタルＶＴＲ１４とキャプチャ部１５とを備えている。デジタルＶＴＲ１４には、符号化対象の動画像データが記録されている。キャプチャ部１５は、デジタルＶＴＲ１４から出力された映像信号を所定の形式に変換する。変換後の映像信号は、システム管理部３の制御により、ネットワーク４経由で共有記憶部２に転送される。

【００５２】

画像出力部１２は、デコード部１６とデジタルテレビ１７とを備えている。共有記憶部２に蓄積されたビットストリームは、システム管理部３の制御により、ネットワーク４経由でデコード部１６に転送される。デコード部１６は、転送されたビットストリームを復号し、再生映像信号を出力する。デジタルテレビ１７は、デコード部１６から出力された再生映像信号を画面に表示する。これにより、動画像符号化装置のユーザは、再生映像信号の画質を評価することができる。

【００５３】

ストリーム記憶部１３は、記憶媒体（図示せず）を着脱可能に構成されている。共有記憶部２に蓄積されたビットストリームは、システム管理部３の制御により、ネットワーク４経由でストリーム記憶部１３に転送される。ストリーム記憶部１３は、転送されたビットストリームを着脱可能に構成された記憶媒体に書き込む。ストリーム記憶部１３には、例えば、デジタルＶＴＲやディスク記録装置などが使用される。

【００５４】

図１に示す動画像符号化装置は、以下の特徴を有する。すなわち、図１に示す動画像符号化装置では、符号化対象の動画像データは、独立して符号化可能な単位に分割された状態で共有記憶部２に蓄積される。また、符号化処理は、特徴抽出、シーン検出、ＧＯＰ構造決定、動きベクトル探索、仮符号化、ビット割当て、および、本符号化（仮符号化の後に行われる最終的な符号化）など、複数の処理（以下、各処理を総称して「単位処理」という）に分割される。各プロセッサ１は、ある単位処理を完了したときに、分割された動画像データを任意に選択し、選択した動画像データについてその時点で実行可能な単位処理を選択し、選択した単位処理を実行する。各プロセッサ１における単位処理の選択と実行とは、他のプロセッサの動作とは独立して、任意のタイミングで行われる。

【００５５】

このように符号化処理の実行制御がｎ個のプロセッサ１－１～ｎによって分散して行われるので、符号化処理の実行制御を行う管理用のプロセッサを備える必要がなく、ｎ個のプロセッサ１－１～ｎの間で同期を取る必要もない。したがって、逐次的に行われる符号化処理の実行制御によって、並列処理による高速化の効果が損われることがなく、装置の性能を向上させることができる。また、ｎ個のプロセッサ１－１～ｎはいずれも同じ構成を有していればよいため、符号化処理の実行制御を簡素化することができる。

【００５６】

以下、図１に示す動画像符号化装置の特徴を５つの実施形態に分けて説明する。第１および第２の実施形態では、符号化処理中にプロセッサを追加および削除する機構について説明する。第３の実施形態では、符号化処理中に符号化処理の内容を切り替え、追加および変更する機構について説明する。第４の実施形態では、符号化処理中に異常プロセッサを検出する機構について説明する。第５の実施形態では、共有記憶部の詳細な構成について説明する。以下では、符号化方式としてＭＰＥＧを採用し、動画像データはＭＰＥＧで規定されたシーケンス単位に分割され、その結果ｓ個のシーケンスが得られたとする。

【００５７】

（第１の実施形態）

本発明の第１の実施形態に係る動画像符号化装置は、上述した符号化処理の実行制御の実現方式に加えて、プロセッサ識別情報を動的に割り当て、符号化処理中にプロセッサを追加および削除できること、並びに、世代番号を用いて故障したプロセッサを検出することを特徴とする。

【００５８】

図３は、本実施形態に係る動画像符号化装置のデータ配置図である。図３において、ｎ個のプロセッサ１０１－１～ｎおよび共有記憶部１０２は、それぞれ、図１に示すプロセッサ１－１～ｎおよび共有記憶部２を詳細化したものであり、システム管理部１０３は、図１に示すシステム管理部３に対応する。なお、図３では、本実施形態の特徴を説明するために必要なデータのみを示し、その他のデータについては図示を省略している。図３に示すように、各プロセッサ１０１には、プロセッサ識別情報１１０が蓄積される。プロセ

ッサ識別情報１１０は、図２に示すプロセッサ識別情報３１を詳細化したものであり、識別番号１１１と世代番号１１２とを含んでいる。識別番号１１１は、装置全体で予め用意された１からｍまでの番号のうち、当該プロセッサ１０１が取得した番号である。識別番号１１１は、符号化処理実行中は１からｍまでのいずれかの値となり、符号化処理停止中は０となる。世代番号１１２は、後述するように、プロセッサ１０１に生じた故障を検出するために使用される。

【００５９】

図３に示すように、共有記憶部１０２には、ｓ個の各シーケンスについて、シーケンス処理用データ１４０が蓄積される。第ｉのシーケンス処理用データ１４０には、第ｉのシーケンスを符号化する際に必要とされるすべてのデータ、例えば、動画像データ、制御情報、中間結果、および、ビットストリームなどが含まれる。

【００６０】

加えて、共有記憶部１０２には、シーケンス管理情報１２０と識別管理情報１３０とが蓄積される。シーケンス管理情報１２０は、ｓ個の各シーケンスについて、プロセッサ番号１２１と処理状態１２２とを含んでいる。プロセッサ番号１２１は、当該シーケンスについて現在、単位処理を実行しているプロセッサの識別番号である。処理状態１２２は、当該シーケンスについて、現在実行されている単位処理を示す。

【００６１】

識別管理情報１３０は、現世代番号１３１、世代検査値１３２、世代更新フラグ１３３、および、ｍ個の世代情報１３４を含んでいる。現世代番号１３１は、現在の世代番号を示す。世代検査値１３２は、現世代番号１３１が有効であるか否かを検査するために使用される。世代更新フラグ１３３は、現世代番号１３１が更新されたか否かを示す。世代情報１３４は、ｍ個（識別番号の個数）だけ存在する。第ｉの世代情報１３４には、値ｉの識別番号１１１がいずれかのプロセッサ１０１によって取得されたときの現世代番号１３１が設定される。

【００６２】

図４は、各プロセッサ１０１によるメイン処理を示すフローチャートである。ｎ個のプロセッサ１０１－１～ｎは、いずれも、図２に示す符号化処理用プログラム３２を実行することにより、図４に示すフローチャートに従って動作する。メイン処理の概要は、次のとおりである。プロセッサ１０１は、符号化処理開始前にプロセッサ追加処理（図５）を実行し、予め用意されたｍ個の識別番号から一の識別番号を取得する。その後、プロセッサ１０１は、次に実行すべき単位処理を選択し、選択した単位処理を実行する処理を繰り返す。すべてのシーケンスについて符号化処理を完了したときに、プロセッサ１０１は、プロセッサ削除処理（図６）を実行し、取得した識別番号を解放する。

【００６３】

共有記憶部１０２に蓄積された現世代番号１３１は、適宜１ずつ更新される。プロセッサ１０１は、ある単位処理を完了した時点で現世代番号１３１が更新されていることを検出した場合には、自らの世代番号１１２を現世代番号１３１に合わせて更新する。現世代番号１３１の更新後に所定の時間経過しても、世代番号１１２を更新しないプロセッサ１０１は故障したものとして扱われ、故障したプロセッサが使用していた識別番号は解放される。

【００６４】

図４に示す各ステップの詳細は、次のとおりである。プロセッサ１０１は、起動されると、まず、自プロセッサについて初期化処理を行う（ステップＳ１１０１）。初期化処理では、プロセッサ１０１は、ＲＡＭ内の符号化処理用データ（図２に示す符号化処理用データ３３）を初期化するなどの処理を行う。次に、プロセッサ１０１は、後述するプロセッサ追加処理（図５）によって識別番号１１１を取得する（ステップＳ１１０２）。

【００６５】

次に、プロセッサ１０１は、取得した識別番号１１１が有効か否かを判断する（ステップＳ１１０３）。取得した識別番号１１１が有効でない場合には（ステップＳ１１０３のＮ

Ｏ）、これ以上プロセッサを追加できないため、プロセッサ１０１は処理を終了する。取得した識別番号１１１が有効である場合には（ステップＳ１１０３のＹＥＳ）、プロセッサ１０１は、ステップＳ１１０４へ進み、既に動作中の他のプロセッサとともに符号化処理を並列に実行する。

【００６６】

次に、プロセッサ１０１は、所定の判断基準に従い、次の処理を選択する（ステップＳ１１０４）。すなわち、プロセッサ１０１は、各シーケンスに割当てられた優先度などの情報に基づき、次に処理すべきシーケンスを選択し、選択したシーケンスについて実行可能な単位処理のうち、一の単位処理を次の処理として選択する。

【００６７】

次に、プロセッサ１０１は、次の処理が存在するか否かを判断する（ステップＳ１１０５）。次の処理が存在しない場合には（ステップＳ１１０５のＮＯ）、すべてのシーケンスについて処理を完了したので、プロセッサ１０１は、後述するプロセッサ削除処理（図６）によって識別番号１１１を解放した後（ステップＳ１１１５）、処理を終了する。

【００６８】

次の処理が存在する場合には（ステップＳ１１０５のＹＥＳ）、プロセッサ１０１は、選択した処理を実行するために必要なデータを共有記憶部１０２から読み出し、自らのＲＡＭ（図２に示すＲＡＭ２３）に書き込む（ステップＳ１１０６）。例えば、プロセッサ１０１は、選択したシーケンスについて、動画像データ、制御情報および中間結果などを、共有記憶部１０２から読み出して自らのＲＡＭに書き込む。

【００６９】

次に、プロセッサ１０１は、選択した処理を実行する（ステップＳ１１０７）。具体的には、プロセッサ１０１は、自らのＲＡＭ上のデータに対して、特徴抽出、シーン検出、ＧＯＰ構造の決定、動きベクトル探索、仮符号化、ビット割当て、および、本符号化などの単位処理を実行し、その結果を自らのＲＡＭに書き込む。

【００７０】

次に、プロセッサ１０１は、求めた結果を共有記憶部１０２に書き込む前に、ステップＳ１１０８からＳ１１１３までの世代管理処理を実行する。世代管理処理では、プロセッサ１０１は、まず、識別管理情報１３０をロックする（ステップＳ１１０８）。次に、プロセッサ１０１は、現世代番号１３１が更新されているか否かを判断する（ステップＳ１１０９）。具体的には、プロセッサ１０１は、世代更新フラグ１３３が１であれば更新済み、それ以外の場合は未更新と判断する。現世代番号１３１が更新されていない場合には（ステップＳ１１０９のＮＯ）、プロセッサ１０１は、直ちにステップＳ１１１３へ進む。

【００７１】

現世代番号１３１が更新されている場合には（ステップＳ１１０９のＹＥＳ）、プロセッサ１０１は、変数ｉに自らの識別番号１１１を代入し（ステップＳ１１１０）、自らの世代番号１１２が第ｉの世代情報１３４と一致するか否かを判断する（ステップＳ１１１１）。第ｉの世代情報１３４には、通常は、プロセッサ１０１が識別番号１１１を取得した際の現世代番号１３１が設定されており、プロセッサ１０１が故障した後は、０が設定されている。このため、世代番号１１２と第ｉの世代情報１３４とが一致した場合には（ステップＳ１１１１のＹＥＳ）、プロセッサ１０１は、符号化処理を継続すべく、自らの世代番号１１２と第ｉの世代情報１３４とにいずれも現世代番号１３１を設定する（ステップＳ１１１２）。一方、世代番号１１２と第ｉの世代情報１３４とが一致しない場合には（ステップＳ１１１１のＮＯ）、プロセッサ１０１は、符号化処理を停止すべく、自らの識別番号１１１と世代番号１１２とにいずれも０を設定し（ステップＳ１１１６）、識別管理情報１３０をアンロックした後（ステップＳ１１１７）、処理を終了する。

【００７２】

符号化処理を継続する場合、プロセッサ１０１は、識別管理情報１３０をアンロックし（ステップＳ１１１３）、ステップＳ１１０７で求めた結果を自らのＲＡＭから読み出して、共有記憶部１０２に書き込む（ステップＳ１１１４）。その後、プロセッサ１０１は、

ステップＳ１１０４へ進み、再びステップＳ１１０４以降の処理を実行する。なお、ステップＳ１１０７で所定の処理を完了した後、求めた結果を共有記憶部１０２に書き込む前に世代管理処理を実行しているのは、故障したプロセッサによって求められた結果が共有記憶部１０２に書き込まれることを防止するためである。

【００７３】

以下、本実施形態に係る動画像符号化装置におけるプロセッサ追加処理とプロセッサ削除処理とを説明する。図５に示すプロセッサ追加処理では、プロセッサ１０１は、まず、識別管理情報１３０をロックする（ステップＳ１２０１）。次に、プロセッサ１０１は、現世代番号１３１に対して所定の演算ｆを行い、求めた値と世代検査値１３２とを比較する（ステップＳ１２０２）。所定の演算ｆは、識別管理情報１３０が初期化されていない状態では、現世代番号１３１に演算ｆを施した結果と世代検査値１３２とが一致しないように決定される。例えば、演算ｆとして、ビットごとの否定（ＮＯＴ）を用いることができる。このような演算ｆを選択すれば、現世代番号１３１に対して演算ｆを施した結果と世代検査値１３２とが一致する場合には、識別管理情報１３０は既に初期化されており、両者が一致しない場合には、識別管理情報１３０はまだ初期化されていないと判断することができる。プロセッサ１０１は、両者が一致する場合にはステップＳ１２０４へ、それ以外の場合はステップＳ１２０３へ進む。

【００７４】

ステップＳ１２０２で両者が一致しない場合には（ステップＳ１２０２のＮＯ）、プロセッサ１０１は、自プロセッサが装置全体で最初にプロセッサ追加処理を実行していると判断し、識別管理情報１３０を初期化する（ステップＳ１２０３）。具体的には、プロセッサ１０１は、現世代番号１３１には１を、世代検査値１３２には現世代番号（値は１）に演算ｆを施した結果を、世代更新フラグ１３３およびｍ個の世代情報１３４には０を、それぞれ設定する。

【００７５】

次に、プロセッサ１０１は、ステップＳ１２０４からＳ１２０７の処理により、ｍ個の世代情報１３４のうちで未使用のもの、すなわち、値が０であるものの番号を求める。より詳細に述べると、プロセッサ１０１は、変数ｉに１を設定し（ステップＳ１２０４）、第ｉの世代情報１３４が０であるか否かを調べる（ステップＳ１２０５）。第ｉの世代情報１３４が０である場合には（ステップＳ１２０５のＹＥＳ）、プロセッサ１０１は、ステップＳ１２０８へ進む。それ以外の場合には（ステップＳ１２０５のＮＯ）、プロセッサ１０１は、ステップＳ１２０６へ進み、変数ｉがｍ（識別番号の個数）未満であれば（ステップＳ１２０６のＹＥＳ）、変数ｉに１を加算し（ステップＳ１２０７）、ステップＳ１２０５へ進む。

【００７６】

未使用の世代情報１３４がある場合には（ステップＳ１２０５のＹＥＳ）、プロセッサ１０１は、自らの識別番号１１１および世代番号１１２に求めた番号ｉおよび現世代番号１３１をそれぞれ設定し、第ｉの世代情報１３４に現世代番号１３１を設定する（ステップＳ１２０８）。未使用の世代情報１３４がない場合には（ステップＳ１２０６のＮＯ）、プロセッサ１０１は、自らの識別番号１１１に０を設定する（ステップＳ１２０９）。この場合、他のプロセッサが既に番号ｉを識別番号として使用している可能性があるので、プロセッサ１０１は、第ｉの世代情報１３４には値を設定しない。

【００７７】

最後に、プロセッサ１０１は、識別管理情報１３０をアンロックし（ステップＳ１２１０）、これをもってプロセッサ追加処理を完了する。上述したように、プロセッサ１０１は、プロセッサ追加処理で有効な識別番号を取得した場合、次の処理を選択して、選択した処理を実行する処理（図４のステップＳ１１０４からＳ１１１４の処理）を繰り返す。このようにして、有効な識別番号１１１を取得したプロセッサ１０１は、符号化装置に組み込まれる。

【００７８】

図６に示すプロセッサ削除処理では、プロセッサ１０１は、まず、識別管理情報１３０をロックし（ステップＳ１３０１）、変数ｉに自らの識別番号１１１を代入する（ステップＳ１３０２）。次に、プロセッサ１０１は、自らの識別番号１１１および世代番号１１２、並びに、第ｉの世代情報１３４に、いずれも０を設定する（ステップＳ１３０３）。最後に、プロセッサ１０１は、識別管理情報１３０をアンロックし（ステップＳ１３０４）、これをもってプロセッサ削除処理を完了する。

【００７９】

図６に示すプロセッサ削除処理または図４のステップＳ１１１６の処理により、プロセッサ１０１は、これまで使用していた識別番号１１１を解放する。このようにして、使用していた識別番号１１１を解放したプロセッサ１０１は、符号化装置から切り離される。装置から切り離されたプロセッサ１０１は、任意のタイミングでプロセッサ追加処理を実行し、有効な識別番号１１１を取得すれば、符号化装置に再び組込まれる。

【００８０】

図５に示すプロセッサ追加処理、および、図６に示すプロセッサ削除処理によれば、プロセッサ１０１が識別管理情報１３０に対して排他制御を行うのは、プロセッサ追加処理またはプロセッサ削除処理を実行するときに限られる。このため、他のプロセッサが符号化処理を実行するために共有記憶部１０２の他のデータにアクセスしていても、プロセッサ追加／削除処理がそのアクセスを妨げることはない。したがって、各プロセッサは、他のプロセッサによる符号化処理に影響を及ぼすことなく、任意のタイミングでプロセッサ追加／削除処理を実行することができる。

【００８１】

次に、世代番号を用いて、故障したプロセッサを検出する方法を説明する。図７は、本実施形態に係る動画像符号化装置における世代更新処理のフローチャートである。システム管理部１０３は、故障したプロセッサを装置から切り離すために、所定のコマンドが入力されたときに、あるいは、所定の時間間隔で世代更新処理を実行する。また、図４に示すフローチャートの好適な箇所に世代更新処理を呼び出すステップを挿入し、プロセッサ１０１が、所定の時間間隔で世代更新処理を実行してもよい。以下では、システム管理部１０３が世代更新処理を実行する場合を例示するが、プロセッサ１０１が世代更新処理を実行する場合も全く同様である。

【００８２】

図７に示す世代更新処理では、システム管理部１０３は、まず、識別管理情報１３０をロックする（ステップＳ１４０１）。次に、システム管理部１０３は、現世代番号１３１に所定の演算ｆを施した結果と世代検査値１３２とを比較し、両者が一致するときはステップＳ１４０４へ、それ以外のときはステップＳ１４０３へ進む（ステップＳ１４０２）。

【００８３】

両者が一致しない場合には（ステップＳ１４０２のＮＯ）、システム管理部１０３は、世代番号はまだ初期化されていないと判断し、識別管理情報１３０を初期化し（ステップＳ１４０３）、ステップＳ１４１６へ進む。なお、図７のステップＳ１４０２およびＳ１４０３は、それぞれ、図５のステップＳ１２０２およびＳ１２０３と同じであるので、その説明を省略する。

【００８４】

ステップＳ１４０２で両者が一致する場合には（ステップＳ１４０２のＹＥＳ）、システム管理部１０３は、世代更新フラグ１３３が０か否かを判断する（ステップＳ１４０４）。世代更新フラグ１３３が０でない場合には（ステップＳ１４０４のＮＯ）、システム管理部１０３は、自分以外が世代更新処理を実行中であると判断し、現世代番号１３１を更新することなく、ステップＳ１４１６へ進む。一方、世代更新フラグ１３３が０である場合には（ステップＳ１４０４のＹＥＳ）、システム管理部１０３は、現世代番号１３１を更新すべく、ステップＳ１４０５へ進む。

【００８５】

次に、システム管理部１０３は、世代更新処理を実行中であることを示すため、世代更新

フラグ１３３に１を設定する（ステップＳ１４０５）。次に、システム管理部１０３は、現世代番号１３１に１を加算し、加算後の現世代番号１３１に演算ｆを施した結果を世代検査値１３２に設定する（ステップＳ１４０６）。これにより、現世代番号１３１は１ずつ更新され、これに伴い世代検査値１３２も更新される。

【００８６】

次に、システム管理部１０３は、識別管理情報１３０をアンロックし（ステップＳ１４０７）、所定の時間だけ停止する（ステップＳ１４０８）。ステップＳ１４０８における所定の時間は、正常なプロセッサ１０１がその時間内に図４のステップＳ１１０４で選択した処理を完了できる値に設定される。このため、プロセッサ１０１の識別番号１１１をｉとしたとき、システム管理部１０３がステップＳ１４０８に到達した時点における第ｉの世代情報１３４は、プロセッサ１０１が正常であれば、現世代番号１３１と一致する。一方、プロセッサ１０１が故障したために、システム管理部１０３がステップＳ１４０８に到達した時点で、プロセッサ１０１が選択した処理を完了していない場合には、第ｉの世代情報１３４は現世代番号１３１と一致しない。

【００８７】

そこで、システム管理部１０３は、停止状態から動作を再開すると、識別管理情報１３０をロックした後（ステップＳ１４０９）、現世代番号１３１に一致しない各世代情報１３４に「データ無効」を示す値０を設定する。より詳細に述べると、システム管理部１０３は、変数ｉに１を代入し（ステップＳ１４１０）、第ｉの世代情報１３４が現世代番号１３１と一致しない場合には（ステップＳ１４１１のＮＯ）、第ｉの世代情報１３４に０を設定する（ステップＳ１４１２）。次に、システム管理部１０３は、変数ｉがｓ（シーケンスの個数）未満である場合には（ステップＳ１４１３のＹＥＳ）、変数ｉに１を加算して（ステップＳ１４１４）、ステップＳ１４１１へ進む。ステップＳ１４１０からＳ１４１３の処理により、自らの世代番号１１２を更新していないプロセッサ１０１があった場合、当該プロセッサ１０１の識別番号をｉとしたとき、第ｉの世代情報１３４は０となる。プロセッサ１０１は、自らの世代番号１１２と第ｉの世代情報１３４とが一致しない場合には、取得した世代番号を解放して処理を終了する（図４のステップＳ１１１６およびＳ１１１７）。これにより、故障したプロセッサ１０１は、装置から切り離される。

【００８８】

次に、システム管理部１０３は、世代更新処理を完了したことを示すため、世代更新フラグ１３３に０を設定する（ステップＳ１４１５）。最後に、システム管理部１０３は、識別管理情報１３０をアンロックし（ステップＳ１４１６）、これをもって世代更新処理を完了する。

【００８９】

このように図７に示す世代更新処理によれば、システム管理部１０３またはプロセッサ１０１が現世代番号１３１を順次１ずつ更新し、現世代番号１３１を更新した後に所定の時間経過しても、自らの世代番号１１２を更新しないプロセッサ１０１が使用していた識別番号１１１は解放される。解放された識別番号は、その後に他のプロセッサ１０１によって使用される。したがって、符号化処理の実行中であっても、他のプロセッサ１０１による符号化処理に影響を与えることなく、プロセッサ１０１に生じた故障を検知し、故障したプロセッサ１０１を装置から削除することができる。

【００９０】

なお、世代更新処理を行った際に、更新前の現世代番号１３１を有する第１のプロセッサと、更新後の現世代番号１３１を有する第２のプロセッサとが、同じ識別番号１１１を使用し、同時に符号化処理を実行する場合が考えられる。しかし、共有記憶部１０２は、符号化処理に必要なデータをシーケンスごとに記憶しているので、第１および第２のプロセッサは、それぞれ、共有記憶部１０２の異なる領域に求めた結果を書き込む。したがって、故障したプロセッサを検出するために行われる世代更新処理が、ｎ個のプロセッサ１０１－１～ｎによって行われる符号化処理に悪影響を与えることはない。

【００９１】

以上に示すように、本実施形態に係る動画像符号化装置によれば、符号化処理の実行制御が複数のプロセッサによって分散して行われるので、符号化処理の実行制御を行う管理用のプロセッサを備える必要がなく、プロセッサ間で同期を取る必要もない。また、各プロセッサは、共有記憶部に記憶された識別管理情報に基づき識別情報の取得と解放とを行い、識別情報を有している間だけ動作する。プロセッサによる識別情報の取得と解放とは、他のプロセッサの動作とは独立して、任意のタイミングで行われる。したがって、他のプロセッサが符号化処理を実行中であっても、他のプロセッサに影響を及ぼすことなく、任意のタイミングで装置にプロセッサを追加および削除することができる。このため、プロセッサで障害が発生した場合などに、装置全体の符号化処理を中断することなく、プロセッサを保守することができる。

【００９２】

（第２の実施形態）

第２の実施形態では、次処理選択処理および共有記憶部に記憶される処理状態の詳細について説明する。

【００９３】

図８は、本実施形態に係る動画像符号化装置のデータ配置図である。図８において、ｎ個のプロセッサ１５１－１～ｎおよび共有記憶部１５２は、それぞれ、図１に示すプロセッサ１－１～ｎおよび共有記憶部２を詳細化したものである。なお、図８では、本実施形態の特徴を説明するために必要なデータのみを示し、その他のデータについては図示を省略している。図８に示すように、各プロセッサ１５１には、プロセッサ識別情報１６１、選択シーケンス番号１６２、および、選択処理番号１６３が蓄積される。プロセッサ識別情報１６１は、図２に示すプロセッサ識別情報３１に対応する。プロセッサ識別情報１６１は、各プロセッサ１５１に予め一意に割り当てられていてもよく、第１の実施形態で示したように、各プロセッサが動的に取得するものであってもよい。

【００９４】

図８に示す動画像符号化装置では、符号化処理は複数の単位処理に分割され、各プロセッサ１５１は、一のシーケンスと当該シーケンスについて実行可能な単位処理とを選択して、選択した処理を実行する。選択シーケンス番号１６２は、プロセッサ１５１が選択したシーケンスの番号であり、選択処理番号１６３は、プロセッサ１５１が選択した単位処理の番号である。なお、本実施形態では、符号化処理はｔ個の単位処理に分割されるとする。

【００９５】

図８に示すように、共有記憶部１５２には、ｓ個の各シーケンスについて、第１から第ｓのシーケンス処理用データ１８０が蓄積される。本実施形態に係るシーケンス処理用データは、第１の実施形態と同じであるので、ここではその説明を省略する。

ここではその説明を省略する。

【００９６】

また、共有記憶部１５２には、符号化処理の進行状況を監視制御するために、シーケンス管理情報１７０が蓄積される。シーケンス管理情報１７０は、ｓ個の各シーケンスについて、現処理番号１７１、選択可能処理番号１７２、および、処理状態１７３を含んでいる。現処理番号１７１は、当該シーケンスについて現在実行されている単位処理の番号である。選択可能処理番号１７２は、当該シーケンスについて次のプロセッサが選択することができる単位処理の番号である。現処理番号１７１および選択可能処理番号１７２は、いずれも１からｔ（単位処理の個数）までのいずれかの値を取る。

【００９７】

処理状態１７３は、各単位処理について、プロセッサ番号１７４、開始フラグ１７５、終了フラグ１７６、および、エラーフラグ１７７を含んでいる。各データの第ｊの要素について述べると、プロセッサ番号１７４は、第ｊの単位処理を実行するプロセッサ１５１のプロセッサ識別情報１６１である。開始フラグ１７５は、第ｊの単位処理が開始されるまでは０、開始後は１となる。終了フラグ１７６は、第ｊの単位処理が終了するまでは０、

終了後は１となる。エラーフラグ１７７は、第ｊの単位処理でエラーが発生するまでは０、エラー発生後は１となる。これらのデータは、後述するように、いずれもプロセッサ１５１によって更新される。

【００９８】

図９は、各プロセッサ１５１によるメイン処理を示すフローチャートである。ｎ個のプロセッサ１５１－１～ｎは、いずれも、図２に示す符号化処理用プログラム３２を実行することにより、図９に示すフローチャートに従って動作する。図９に示すメイン処理は、第１の実施形態（図４）とほぼ同様であるが、次処理選択処理（図１０）およびシーケンス状態更新処理（図１１）において、処理状態１７３に含まれる単位処理ごとのデータを更新することを特徴とする。具体的には、プロセッサ１５１は、次処理選択処理では、プロセッサ番号１７４に自らのプロセッサ識別情報１６１を設定するとともに、開始フラグ１７５を１に設定し、シーケンス状態更新処理では、終了フラグ１７６を１に設定するとともに、故障発生時にはエラーフラグ１７７に１を設定する。

【００９９】

図９に示す各ステップの詳細は、次のとおりである。なお、図９のステップＳ１５０１、Ｓ１５０４、Ｓ１５０５およびＳ１５０６は、それぞれ、図４のステップＳ１１０１、Ｓ１１０６、Ｓ１１０７およびＳ１１１４と同じであるので、その説明を省略する。プロセッサ１５１は、起動されると、まず、自プロセッサについて初期化処理を行う（ステップＳ１５０１）。次に、プロセッサ１５１は、後述する次処理選択処理（図１０）を実行し、自らの選択シーケンス番号１６２および選択処理番号１６３、並びに、シーケンス管理情報１７０を更新する（ステップＳ１５０２）。

【０１００】

次に、プロセッサ１５１は、次の処理が存在するか否かを判断する（ステップＳ１５０３）。次の処理が存在しない場合には（ステップＳ１５０３のＮＯ）、すべてのシーケンスについて処理を完了したので、プロセッサ１５１は処理を終了する。次の処理が存在する場合には（ステップＳ１５０３のＹＥＳ）、プロセッサ１５１は、必要なデータを共有記憶部１０２から読み出し（ステップＳ１５０４）、選択した処理を実行し（ステップＳ１５０５）、求めた結果を共有記憶部１０２に書き込む（ステップＳ１５０６）。

【０１０１】

次に、プロセッサ１５１は、後述するシーケンス状態更新処理（図１１）を実行し、ステップＳ１５０５における処理結果に応じてシーケンス管理情報１７０を更新する（ステップＳ１５０７）。その後、プロセッサ１５１は、ステップＳ１５０２へ進み、ステップＳ１５０２からＳ１５０７の処理を繰り返し実行する。

【０１０２】

以下、本実施形態に係る動画像符号化装置における次処理選択処理とシーケンス状態更新処理とを説明する。図１０に示す次処理選択処理では、プロセッサ１５１は、まず、シーケンス管理情報１７０をロックする（ステップＳ１６０１）。なお、シーケンス管理情報１７０がロックされていても、他のプロセッサは、シーケンス処理用データ１８０に対して自由にアクセスできる。

【０１０３】

次に、プロセッサ１５１は、ステップＳ１６０２からＳ１６０７の処理により、ｓ個のシーケンスのうちから一のシーケンスを選択する。より詳細に述べると、プロセッサ１５１は、変数ｉに１を代入し（ステップＳ１６０２）、第ｉのシーケンスについて現処理番号１７１と選択可能処理番号１７２とを比較する（ステップＳ１６０３）。両者が一致する場合には（ステップＳ１６０３のＹＥＳ）、第ｉのシーケンスについて選択可能な処理が存在しないため、プロセッサ１５１は、次のシーケンスを調べる。すなわち、プロセッサ１５１は、変数ｉがｓ（シーケンスの個数）未満であれば（ステップＳ１６０４のＹＥＳ）、変数ｉに１を加算し（ステップＳ１６０５）、ステップＳ１６０３へ進む。

【０１０４】

ステップＳ１６０４で変数ｉがｓ以上となったときには（ステップＳ１６０４のＮＯ）、

いずれのシーケンスについても選択可能な処理が存在しないため、プロセッサ１５１は、シーケンス管理情報１７０を一旦アンロックし（ステップＳ１６０６）、所定の時間だけ停止した後（ステップＳ１６０７）、ステップＳ１６０１へ進む。このように、実行すべき処理は残っているが、その時点でいずれのシーケンスについても選択可能な処理が存在しない場合には、プロセッサ１５１は、所定の時間だけ待機した後に、選択可能な処理を再び探索する。

【０１０５】

ステップＳ１６０３において、現処理番号１７１が選択可能処理番号１７２と異なる場合には（ステップＳ１６０３のＮＯ）、プロセッサ１５１は、第ｉのシーケンスについて現在実行されている単位処理の次の単位処理を、次に実行すべき処理として選択する。このため、プロセッサ１５１は、第ｉのシーケンスの現処理番号１７１に１を加算し（ステップＳ１６０８）、変数ｊに加算後の現処理番号１７１を代入する（ステップＳ１６０９）。次に、プロセッサ１５１は、自らの選択シーケンス番号１６２および選択処理番号１６３に、それぞれ、求めたｉおよびｊの値を設定する（ステップＳ１６１０）。次に、プロセッサ１５１は、第ｉのシーケンスの第ｊの単位処理について、プロセッサ番号１７４にプロセッサ識別情報１６１を、開始フラグ１７５に１を、終了フラグ１７６およびエラーフラグ１７７に０を、それぞれ設定する（ステップＳ１６１１）。最後に、プロセッサ１５１は、シーケンス管理情報１７０をアンロックし（ステップＳ１６１２）、これをもって次処理選択処理を完了する。

【０１０６】

図１１に示すシーケンス状態更新処理では、プロセッサ１５１は、まず、シーケンス管理情報１７０をロックし（ステップＳ１７０１）、変数ｉおよびｊに自らの選択シーケンス番号１６２および選択処理番号１６３をそれぞれ代入する（ステップＳ１７０２）。次に、プロセッサ１５１は、第ｉのシーケンスの第ｊの単位処理について、終了フラグ１７６に１を、エラーフラグ１７７には故障発生状況に応じて０または１を、それぞれ設定する（ステップＳ１７０３）。

【０１０７】

次に、プロセッサ１５１は、第ｉのシーケンスの選択可能処理番号１７２を更新する（ステップＳ１７０４）。具体的には、第ｉのシーケンスについて第ｊの単位処理の実行が完了したことによって新たに選択可能となった単位処理の番号が、第ｉのシーケンスの選択可能処理番号１７２に設定される。

【０１０８】

次に、プロセッサ１５１は、第ｉのシーケンスについて処理完了を待機している他のプロセッサに対して、処理終了を通知する（ステップＳ１７０５）。これにより、第ｉのシーケンスについて処理完了を待機している他のプロセッサは、符号化処理を再開する。最後に、プロセッサ１５１は、シーケンス管理情報１７０をアンロックし（ステップＳ１７０６）、これをもってシーケンス状態更新処理を完了する。

【０１０９】

図１２は、本実施形態に係る動画像符号化装置の各プロセッサが動作する様子を示すタイムチャートである。ここでは、説明を簡略化するために、５個のプロセッサを用いて６個のシーケンスを符号化するものとする。また、各シーケンスに対する符号化処理は、前処理および本符号化の２種類の単位処理からなり、すべてのシーケンスについて前処理を完了した後に、各シーケンスについて本符号化を行うものとする。

【０１１０】

図１２において、５個のプロセッサは、時刻ｔ０以降の任意の時刻でそれぞれ独立に動作を開始し、第４のプロセッサが、時刻ｔ１に初期化処理を最も早く完了したとする。第４のプロセッサは、時刻ｔ１からｔ２までの間に次処理選択処理を行い、次に実行すべき処理として前処理を選択する。その後、第２、第１、第３、第５の各プロセッサが、初期化処理を完了し、次処理選択処理により第１から第４のシーケンスの本符号化を順次選択する。第２、第１、第３、第５の各プロセッサは、次の処理を選択した後、第４のプロセッ

サによる前処理が完了するまで待機する。

【０１１１】

前処理を選択した第４のプロセッサは、各シーケンスについて前処理を完了するごとに、当該シーケンスに対する処理完了を待機していたプロセッサに対して、処理完了を通知する。例えば、第１のシーケンスに対する処理完了を待機していた第２のプロセッサは、第４のプロセッサから処理完了の通知を受けると、第１のシーケンスに対する本符号化を開始する。第１、第３、第５の各プロセッサについても同様である。第４のプロセッサは、時刻ｔ３で第１から第６のシーケンスについて前処理を完了すると、時刻ｔ３からｔ４までの間に次処理選択処理を行い、第５のシーケンスの本符号化を選択する。時刻ｔ４以降、第４のプロセッサは、第５のシーケンスの本符号化を実行する。

【０１１２】

以上に示すように、本実施形態に係る動画像符号化装置によれば、共有記憶部には、各シーケンスの各単位処理について、処理を実行したプロセッサの識別情報と、処理の開始、終了およびエラーを示すフラグとを含んだ処理状態が記憶され、各プロセッサは、次処理選択処理およびシーケンス状態更新処理において処理状態を更新する。したがって、符号化処理の実行中または実行後に、共有記憶部に記憶された処理状態に基づき、プロセッサの故障を検出することができる。

【０１１３】

なお、第１および第２の実施形態では、各プロセッサは、共有記憶部から必要なデータを読み出し、ＲＡＭ上のデータを用いて所定の処理を実行し、求めた結果を共有記憶部に書き込んでいる（図４、図９）。したがって、各プロセッサに含まれるＲＡＭは、データキャッシュとして動作していると言える。このようにＲＡＭをデータキャッシュとして用いることにより、共有記憶部に対するアクセスが減少し、各プロセッサにおけるデータ待ち時間が短縮されるので、動画像符号化装置の性能を向上させることができる。

【０１１４】

（第３の実施形態）

本発明の第３の実施形態に係る動画像符号化装置は、符号化処理中に符号化処理の内容を切り替え、追加および変更することを特徴とする。具体的には、共有記憶部には、プログラムコードとプログラム管理情報とが記憶され、各プロセッサは、自らのＲＡＭに記憶されたプログラムを実行中にプログラム管理情報を参照し、必要に応じてプログラム更新処理を実行する。

【０１１５】

図１３は、本実施形態に係る動画像符号化装置のデータ配置図である。図１３において、ｎ個のプロセッサ２０１－１～ｎおよび共有記憶部２０２は、それぞれ、図１に示すプロセッサ１－１～ｎおよび共有記憶部２を詳細化したものであり、システム管理部２０３は、図１に示すシステム管理部３に対応する。なお、図１３では、本実施形態の特徴を説明するために必要なデータのみを示し、その他のデータについては図示を省略している。図１３に示すように、共有記憶部２０２には、ｓ個の各シーケンスについて、処理状態２４０とシーケンス処理用データ２５０とが蓄積される。第ｉの処理状態２４０は、第ｉのシーケンスの処理状態を示す。第ｉのシーケンス処理用データ２５０には、第ｉのシーケンスを符号化する際に必要とされるすべてのデータ、例えば、動画像データ、制御情報、中間結果、および、ビットストリームなどが含まれる。

【０１１６】

加えて、共有記憶部２０２には、プロセッサ２０１にダウンロードすべきｐ個のプログラムが蓄積される。具体的には、共有記憶部２０２には、異なる機能をサポートするｐ個の各プログラムについて、プログラム管理情報２２０とプログラムコード２３１とが蓄積される。プログラムコード２３１は、プロセッサ２０１にダウンロードされるプログラムコード自体である。プロセッサ２０１は、後述するように（図１８）、ｐ個のプログラムコード２３１のうちから選択した一のプログラムコードを、共有記憶部２０２から自らのＲＡＭ（図２に示すＲＡＭ２３）にダウンロードする。ダウンロードされたプログラムコー

ドは、図１３に示す符号化処理用プログラム２１１となる。従前の符号化処理用プログラム２１１に従って動作していたプロセッサ２０１は、符号化処理用プログラム２１１が更新された後は、再起動することなく、更新後の符号化処理用プログラム２１１に従って動作する。

【０１１７】

プログラム管理情報２２０は、共有記憶部２０２に蓄積されたプログラムを管理するために使用される。図１４は、１つのプログラムについてのプログラム管理情報２２０に含まれるデータの詳細を示す図である。図１４に示すように、プログラム管理情報２２０は、プログラム更新情報２６０、プログラムコード情報２７０、および、プログラム機能情報２８０を含んでいる。

【０１１８】

プログラム更新情報２６０は、有効フラグ２６１、更新フラグ２６２、削除フラグ２６３、および、ダウンロード数２６４を含んでおり、プログラムの登録、更新およびダウンロードの管理に使用される。有効フラグ２６１は、このプログラムが有効である間は１となる。更新フラグ２６２は、このプログラムが更新されている間は１となる。削除フラグ２６３は、このプログラムが削除されている間は１となる。ダウンロード数２６４は、このプログラムを現在ダウンロードしているプロセッサの個数を示す。

【０１１９】

プログラムコード情報２７０は、プログラム名２７１、先頭アドレス２７２、および、プログラムサイズ２７３を含んでおり、プログラムをダウンロードする際に参照される。プログラム名２７１は、このプログラムの名称を示す。先頭アドレス２７２は、このプログラムのプログラムコード２３１が格納されている共有記憶部２０２の先頭アドレスを示す。プログラムサイズ２７３は、このプログラムのプログラムコード２３１の大きさを示す。

【０１２０】

プログラム機能情報２８０は、項目数２８１、および、ａ個の機能情報２８２を含んでいる。項目数２８１は、機能情報２８２の個数を示す。ａ個の機能情報２８２は、共有記憶部２０２に蓄積されたｐ個のプログラムの機能をａ個に分類したときに、このプログラムがａ個の各機能について、どのレベルの機能をサポートしているかを示す。ただし、プログラムがある機能をサポートしていないときには、当該機能に対応した機能情報２８２は０とする。

【０１２１】

例えば、ａ個の機能のうち、第１の機能がフェード機能（フェードする画像に対する機能）で、第２の処理が動きベクトル探索機能であるとする。また、各プログラムは、フェード機能をサポートするか否かのいずれかであり、動きベクトル探索機能については第１ないし第４の探索手法のいずれかをサポートするか、いずれの探索手法もサポートしないかのいずれかであるとする。この場合、フェード機能をサポートしていないプログラムの第１の機能情報は０となり、フェード機能をサポートしているプログラムの第１の機能情報は１となる。また、動きベクトル探索機能をサポートしていないプログラムの第２の機能情報は０となり、動きベクトル探索機能をサポートしているプログラムの第２の機能情報は１ないし４のいずれかの値となる。

【０１２２】

以下、本実施形態に係る動画像符号化装置における符号化処理について説明する。動画像データをＭＰＥＧに準拠して符号化するには、シーケンス層、ＧＯＰ層、ピクチャ層、スライス層、マクロブロック層、および、ブロック層の各階層において、それぞれ所定の処理を行う必要がある。符号化処理を図１５に示すように階層的に分類した場合、ある階層に含まれる処理の全部または一部を終了すれば、それ以下の階層における処理を並列に実行することができる。例えば、スライスヘッダ生成処理を終了した後は、当該スライスに含まれるすべてのマクロブロックについてマクロブロック符号化処理を並列に実行することができる。したがって、例えば、図１５の右欄に示す各処理をそれぞれ、上述した単位

処理としてもよい。あるいは、処理速度の向上を図るため、ピクチャ層以下の処理（ピクチャヘッダ生成、スライスヘッダ生成、および、マクロブロック符号化）をまとめて１つの単位処理としてもよい。

【０１２３】

図１６は、各プロセッサ２０１によるメイン処理を示すフローチャートである。ｎ個のプロセッサ２０１－１～ｎは、いずれも、図２に示す符号化処理用プログラム３２を実行することにより、図１６に示すフローチャートに従って動作する。各プロセッサ２０１は、図１６に示すように、次に実行すべき処理を選択し、必要に応じて自らの符号化処理用プログラム２１１を更新した後に、選択した処理を実行する。

【０１２４】

図１６に示す各ステップの詳細は、次のとおりである。プロセッサ２０１は、起動されると、まず、初期化処理を行う（ステップＳ２１０１）。初期化処理では、プロセッサ２０１は、ＲＡＭ内の符号化処理用データ（図２に示す符号化処理用データ３３）を初期化するなどの処理を行う。次に、プロセッサ２０１は、後述する次処理選択処理（図１７）によって、次に実行すべき処理を選択する（ステップＳ２１０２）。ステップＳ２１０２では、プロセッサ２０１は、次に処理すべきシーケンスおよびＧＯＰを選択し、選択したＧＯＰについて実行可能な処理のうち、一の単位処理を選択する。以下、ステップＳ２１０２で選択された処理を「次処理」という。

【０１２５】

次に、プロセッサ２０１は、現在実行中のプログラムがサポートしている機能と、次処理を実行するために必要な機能とを比較する。具体的には、プロセッサ２０１は、変数ａに現在実行中のプログラムの項目数を、変数ｂに次処理の項目数を、配列型の変数Ｘに現在実行中のプログラムのａ個の機能情報を、配列型の変数Ｙに次処理のｂ個の機能情報をそれぞれ代入し（ステップＳ２１０３）、後述する機能情報比較処理（図１９）を実行する（ステップＳ２１０４）。なお、プロセッサ２０１は、共有記憶部２０２から自らのＲＡＭに現在実行中のプログラムのプログラム機能情報２８０を複写した上で、複写したプログラム機能情報から現在実行中のプログラムの項目数および機能情報を求め、次処理選択処理では、次処理を実行するために必要な機能に関する情報（項目数および機能情報）をプログラム機能情報２８０と同じ形式で求めるものとする。

【０１２６】

次に、プロセッサ２０１は、機能情報比較処理で２つの機能情報が一致したか否かを判断する（ステップＳ２１０５）。両者が一致した場合には（ステップＳ２１０５のＹＥＳ）、プロセッサ２０１はステップＳ２１０７へ進む。両者が一致しない場合には（ステップＳ２１０５のＮＯ）、プロセッサ２０１は、後述するプロセッサプログラム更新処理（図１８）を実行した後（ステップＳ２１０６）、ステップＳ２１０７へ進む。

【０１２７】

次に、プロセッサ２０１は、ステップＳ２１０２で選択した処理を実行する（ステップＳ２１０７）。より詳細には、プロセッサ２０１は、必要なデータを共有記憶部２０２から読み出して自らのＲＡＭに書き込み、自らのＲＡＭ内のデータに対して選択した処理を実行し、その結果を自らのＲＡＭから読み出して共有記憶部２０２に書き込む。次に、プロセッサ２０１は、すべての符号化処理を終了したか否かを判断し（ステップＳ２１０８）、すべての符号化処理を終了した場合には（ステップＳ２１０８のＹＥＳ）、処理を終了する。一方、処理がまだ残っている場合には（ステップＳ２１０８のＮＯ）、プロセッサ２０１は、ステップＳ２１０２へ進み、符号化処理を継続する。

【０１２８】

図１７は、次処理選択処理のフローチャートである。次処理選択処理では、プロセッサ２０１は、まず、変数Ｓに直前に処理したシーケンスの番号を、変数Ｇに直前に処理したＧＯＰの番号を代入する（ステップＳ２２０１）。次に、プロセッサ２０１は、第Ｓのシーケンスに含まれる第ＧのＧＯＰについて、未だ選択されていない処理を探す（ステップＳ２２０２）。

【０１２９】
次に、プロセッサ２０１は、ステップＳ２２０２で未選択の処理を検出したか否かを判断する（ステップＳ２２０３）。未選択の処理があった場合には（ステップＳ２２０３のＹＥＳ）、プロセッサ２０１は、その処理を次に実行すべき処理として選択し、次処理選択処理を終了する。
【０１３０】
一方、未選択の処理がない場合には（ステップＳ２２０３のＮＯ）、プロセッサ２０１は、ステップＳ２２０４へ進み、次のＧＯＰについて未だ選択されていない処理を探すため、変数ＳおよびＧを更新する。より詳細には、以下のとおりである。プロセッサ２０１は、変数Ｇに１を加算し（ステップＳ２２０４）、変数Ｇが第Ｓのシーケンスのｇｏｐ数以下である場合には（ステップＳ２２０５のＹＥＳ）、ステップＳ２２０２へ進む。変数Ｇが第Ｓのシーケンスのｇｏｐ数より大きい場合には（ステップＳ２２０５のＮＯ）、プロセッサ２０１は、変数Ｓに１を加算し、変数Ｇに０を代入する（ステップＳ２２０６）。次に、プロセッサ２０１は、変数Ｓがシーケンス数以下である場合には（ステップＳ２２０７のＹＥＳ）、ステップＳ２２０２へ進む。変数Ｓがシーケンス数より大きい場合には（ステップＳ２２０７のＮＯ）、プロセッサ２０１は、すべての符号化処理を終了したか否かを判断する（ステップＳ２２０８）。すべての符号化処理を終了した場合には（ステップＳ２２０８のＹＥＳ）、プロセッサ２０１は、次に実行すべき処理を選択することなく、次処理選択処理を終了する。実行すべき処理がまだ残っている場合には（ステップＳ２２０８のＮＯ）、プロセッサ２０１は、変数Ｓに１を、変数Ｇに０を代入し（ステップＳ２２０９）、ステップＳ２２０２へ進む。
【０１３１】
図１８は、プロセッサプログラム更新処理のフローチャートである。この処理では、プロセッサ２０１は、まず、プログラム管理情報２２０をロックし（ステップＳ２３０１）、変数ｉに１を代入する（ステップＳ２３０２）。
【０１３２】
次に、プロセッサ２０１は、ステップＳ２３０３からＳ２３０７の処理により、共有記憶部２０２に蓄積されたｐ個のプログラムから、次処理を実行するための機能をサポートしているプログラムを探す。具体的には、プロセッサ２０１は、第ｉのプログラムが有効でない場合（ステップＳ２３０３のＮＯ）、または、第ｉのプログラムの機能と次処理を実行するために必要な機能とが一致しない場合には（ステップＳ２３０６のＮＯ）、変数ｉに１を加算し（ステップＳ２３０７）、ステップＳ２３０３へ進む。一方、プロセッサ２０１は、第ｉのプログラムが有効で（ステップＳ２３０３のＹＥＳ）、かつ、第ｉのプログラムの機能と次処理を実行するために必要な機能とが一致する場合には（ステップＳ２３０６のＹＥＳ）、ステップＳ２３０８へ進む。
【０１３３】
なお、ステップＳ２３０３で第ｉのプログラムが有効であると判断されるのは、第ｉのプログラムの有効フラグ２６１が１で、更新フラグ２６２および削除フラグ２６３がともに０である場合である。また、ステップＳ２３０４からＳ２３０６の処理は、ステップＳ２１０３からＳ２１０５の処理（図１６）において、比較対象のプログラムを現在実行中のプログラムから第ｉのプログラムに変更したものである。
【０１３４】
プロセッサ２０１がステップＳ２３０８に到達した時点で、変数ｉは、プロセッサ２０１にダウンロードすべきプログラムの番号となっている。そこで、プロセッサ２０１は、第ｉのプログラムのダウンロード数２６４に１を加算し（ステップＳ２３０８）、プログラム管理情報２２０を一旦アンロックした後（ステップＳ２３０９）、第ｉのプログラムのプログラムコード２３１を自らのＲＡＭにダウンロードする（ステップＳ２３１０）。
【０１３５】
次に、プロセッサ２０１は、第ｉのプログラムのダウンロード数２６４から１を減算した後に（ステップＳ２３１２）、再初期化することなく、ダウンロードしたプログラムに制

御を移す（ステップＳ２３１４）。なお、プロセッサ２０１は、ダウンロード数２６４から１を減算する際には、事前にプログラム管理情報２２０をロックし（ステップＳ２３１１）、事後にプログラム管理情報２２０をアンロックする（ステップＳ２３１３）。プロセッサ２０１は、これをもってプロセッサプログラム更新処理を完了する。

【０１３６】

図１９は、機能情報比較処理のフローチャートである。機能情報比較処理は、図１６のステップＳ２１０４、および、図１８のステップＳ２３０５から呼び出される。この処理が呼び出される時点では、変数ａには比較対象となるプログラムの項目数が、変数ｂには次処理の項目数が、配列型の変数Ｘには比較対象のプログラムのａ個の機能情報が、配列型の変数Ｙには次処理のｂ個の機能情報が、それぞれ、設定されている。ただし、次処理を実行するために特定の機能をサポートする必要がない場合には、当該機能に対応した機能情報には０が設定されているものとする。

【０１３７】

機能情報比較処理では、プロセッサ２０１は、まず、変数ｉに１を代入する（ステップＳ２４０１）。次に、プロセッサ２０１は、ステップＳ２４０２からＳ２４０５の処理により、第１から第ａの機能情報に不一致があるか否かを調べる。より詳細には、プロセッサ２０１は、変数Ｙのｉ番目の要素Ｙ［ｉ］が０でなく（ステップＳ２４０２のＹＥＳ）、かつ、変数Ｘのｉ番目の要素Ｘ［ｉ］とＹ［ｉ］とが一致しない場合には（ステップＳ２４０３のＹＥＳ）、ステップＳ２４１１へ進む。一方、プロセッサ２０１は、Ｙ［ｉ］が０である場合（ステップＳ２４０２のＮＯ）、または、Ｘ［ｉ］とＹ［ｉ］とが一致する場合には（ステップＳ２４０３のＮＯ）、ステップＳ２４０４へ進む。この場合、プロセッサ２０１は、変数ｉに１を加算し（ステップＳ２４０４）、変数ｉがａ（比較対象となるプログラムの項目数）以下の場合はステップＳ２４０２へ、それ以外の場合はステップＳ２４０６へ進む。

【０１３８】

次に、プロセッサ２０１は、変数ａと変数ｂとを比較し（ステップＳ２４０６）、ａがｂより小さい場合はステップＳ２４０７へ、それ以外の場合はステップＳ２４１０へ進む。前者の場合、プロセッサ２０１は、ステップＳ２４０７からＳ２４０９の処理により、変数Ｙの第（ａ＋１）から第ｂの要素に０以外の要素があるか否かを調べる。より詳細には、プロセッサ２０１は、Ｙ［ｉ］が０でない場合には（ステップＳ２４０７のＹＥＳ）、ステップＳ２４１１へ進む。一方、プロセッサ２０１は、Ｙ［ｉ］が０である場合には（ステップＳ２４０７のＮＯ）、変数ｉに１を加算し（ステップＳ２４０８）、変数ｉがｂ（次処理の項目数）以下の場合はステップＳ２４０７へ、それ以外の場合はステップＳ２４１０へ進む。

【０１３９】

プロセッサ２０１は、ステップＳ２４１０に到達した場合には、比較対象のプログラムの機能と次処理に必要とされる機能とが一致すると判断し、ステップＳ２４１１に到達した場合には、両者は一致しないと判断する。プロセッサ２０１は、上記の判断を行った後、機能情報比較処理を終了する。

【０１４０】

図１９に示す機能情報比較処理によれば、項目数が同じ場合は当然のこと、項目数が異なる場合でも、２つの機能情報を比較することができる。なお、図１９の処理では、Ｘ［ｉ］とＹ［ｉ］とが異なる場合に２つの機能情報は一致しないと判断したが、これに代えて、例えば、Ｙ［ｉ］がＸ［ｉ］より大きい場合に、２つの機能情報は一致しないと判断してもよい。

【０１４１】

次に、本実施形態に係る動画像符号化装置において、共有記憶部２０２にプログラムを登録、削除および更新する方法を説明する。共有記憶部２０２に対するこれらの操作は、システム管理部２０３によって行われる。システム管理部２０３は、入力されたコマンドに従い、以下に示すプログラム登録処理（図２０）、プログラム削除処理（図２１）、およ

び、プログラム更新処理（図２２）を実行する。図２０から図２２に示す処理は、プロセッサ２０１が符号化処理を実行している間も含めて、任意のタイミングで実行される。

【０１４２】

図２０は、プログラム登録処理のフローチャートである。プログラム登録処理では、システム管理部２０３は、まず、プログラム管理情報２２０をロックし（ステップＳ２５０１）、変数ｉに１を代入する（ステップＳ２５０２）。次に、システム管理部２０３は、ステップＳ２５０３からＳ２５０５の処理により、有効フラグ２６１が０であるプログラム管理情報２２０を求める。より詳細には、システム管理部２０３は、第ｉの有効フラグ２６１が０でなく（ステップＳ２５０３のＮＯ）、かつ、変数ｉがｐ（プログラムの個数）未満である場合には（ステップＳ２５０４のＹＥＳ）、変数ｉに１を加算し（ステップＳ２５０５）、ステップＳ２５０３へ進む。また、システム管理部２０３は、第ｉの有効フラグ２６１が０でなく（ステップＳ２５０３のＮＯ）、かつ、変数ｉがｐ以上である場合には（ステップＳ２５０４のＮＯ）、ステップＳ２５１４へ進む。

【０１４３】

システム管理部２０３は、第ｉの有効フラグ２６１が０である場合には（ステップＳ２５０３のＹＥＳ）、共有記憶部２０２内にプログラムコードを配置するための領域を確保する（ステップＳ２５０６）。システム管理部２０３は、領域の確保に失敗した場合には（ステップＳ２５０７のＮＯ）、ステップＳ２５１４へ進む。

【０１４４】

システム管理部２０３は、領域の確保に成功した場合には（ステップＳ２５０７のＹＥＳ）、第ｉの有効フラグ２６１および更新フラグ２６２をいずれも１に設定する（ステップＳ２５０８）。次に、システム管理部２０３は、プログラム管理情報２２０を一旦アンロックし（ステップＳ２５０９）、登録対象のプログラムのプログラムコードを共有記憶部２０２内に確保した領域に書き込み（ステップＳ２５１０）、登録対象のプログラムに関する情報を第ｉのプログラムコード情報２７０およびプログラム機能情報２８０に書き込む（ステップＳ２５１１）。

【０１４５】

次に、システム管理部２０３は、第ｉの更新フラグ２６２に０を設定する（ステップＳ２５１３）。この際、システム管理部２０３は、事前にプログラム管理情報２２０をロックし（ステップＳ２５１２）、事後にプログラム管理情報２２０をアンロックする（ステップＳ２５１４）。システム管理部２０３は、これをもってプログラム登録処理を完了する。

【０１４６】

図２１は、プログラム削除処理のフローチャートである。プロセッサ削除処理では、システム管理部２０３は、まず、プログラム管理情報２２０をロックし（ステップＳ２６０１）、変数ｉに削除対象のプログラムの番号を代入する（ステップＳ２６０２）。

【０１４７】

次に、システム管理部２０３は、第ｉのダウンロード数２６４が０であるか否かを調べる（ステップＳ２６０３）。第ｉのダウンロード数２６４が０でない場合には（ステップＳ２６０３のＮＯ）、いずれかのプロセッサ２０１がプロセッサプログラム更新処理（図１８）により、削除対象のプログラムをダウンロードしている。そこでこの場合、システム管理部２０３は、指定されたプログラムを削除することなく、ステップＳ２６１０へ進む。したがって、プログラムを正しく削除するためには、システム管理部２０３は、プログラム削除処理を再び実行する必要がある。

【０１４８】

第ｉのダウンロード数が０である場合には（ステップＳ２６０３のＹＥＳ）、システム管理部２０３は、第ｉの更新フラグ２６２および削除フラグ２６３に１を設定し（ステップＳ２６０４）、プログラム管理情報２２０を一旦アンロックする（ステップＳ２６０５）。次に、システム管理部２０３は、共有記憶部２０２内で第ｉのプログラムのプログラムコード２３１を配置するために使用していた領域を解放し（ステップＳ２６０６）、第ｉ

のプログラムコード情報２７０およびプログラム機能情報２８０を初期化する（ステップＳ２６０７）。

【０１４９】

次に、システム管理部２０３は、第ｉの有効フラグ２６１、更新フラグ２６２、および、削除フラグ２６３にいずれも０を設定する（ステップＳ２６０９）。この際、システム管理部２０３は、事前にプログラム管理情報２２０をロックし（ステップＳ２６０８）、事後にプログラム管理情報２２０をアンロックする（ステップＳ２６１０）。システム管理部２０３は、これをもってプログラム削除処理を完了する。

【０１５０】

図２２は、プログラム更新処理のフローチャートである。このプログラム更新処理は、プログラム登録処理とプログラム削除処理とを１つの処理で行うものである。プログラム更新処理では、システム管理部２０３は、まず、プログラム管理情報２２０をロックし（ステップＳ２７０１）、変数ｉに更新対象のプログラムの番号を代入する（ステップＳ２７０２）。

【０１５１】

次に、システム管理部２０３は、第ｉのダウンロード数２６４が０であるか否かを調べる（ステップＳ２７０３）。第ｉのダウンロード数２６４が０でない場合には（ステップＳ２７０３のＮＯ）、いずれかのプロセッサ２０１がプロセッサプログラム更新処理（図１８）により、更新対象のプログラムをダウンロードしている。この場合、システム管理部２０３は、指定されたプログラムを更新することなく、ステップＳ２７１３へ進む。したがって、プログラムを正しく更新するためには、システム管理部２０３は、プログラム更新処理を再び実行する必要がある。

【０１５２】

第ｉのダウンロード数が０である場合には（ステップＳ２７０３のＹＥＳ）、システム管理部２０３は、共有記憶部２０２内にプログラムコードを配置するための領域を確保する（ステップＳ２７０４）。システム管理部２０３は、領域の確保に失敗した場合には（ステップＳ２７０５のＮＯ）、ステップＳ２７１３へ進む。

【０１５３】

システム管理部２０３は、領域の確保に成功した場合には（ステップＳ２７０５のＹＥＳ）、第ｉの更新フラグ２６２を１に設定する（ステップＳ２７０６）。次に、システム管理部２０３は、プログラム管理情報２２０を一旦アンロックし（ステップＳ２７０７）、共有記憶部２０２内で第ｉのプログラムのプログラムコード２３１を配置するために使用していた領域を解放する（ステップＳ２７０８）。次に、システム管理部２０３は、登録対象のプログラムのプログラムコードを共有記憶部２０２内に新たに確保した領域に書き込み（ステップＳ２７０９）、登録対象のプログラムに関する情報を用いて第ｉのプログラムコード情報２７０およびプログラム機能情報２８０を更新する（ステップＳ２７１０）。

【０１５４】

次に、システム管理部２０３は、第ｉの更新フラグ２６２に０を設定する（ステップＳ２７１２）。この際、システム管理部２０３は、事前にプログラム管理情報２２０をロックし（ステップＳ２７１１）、事後にプログラム管理情報２２０をアンロックする（ステップＳ２７１３）。システム管理部２０３は、これをもってプログラム更新処理を完了する。

【０１５５】

図２０ないし図２２に示すように、システム管理部２０３は、排他制御を行った上でプログラム管理情報２２０にアクセスする。また、図１８に示すように、プロセッサ２０１も、排他制御を行った上でプログラム管理情報２２０にアクセスする。このようにプログラムコード２３１と分離してプログラム管理情報２２０に対する排他制御を行うことにより、複数のプロセッサが同時に共有記憶部２０２からプログラムをダウンロードすることができる。

【０１５６】
本実施形態に係る動画像符号化装置において、プロセッサ２０１が次処理を実行するためのプログラムを共有記憶部２０２から読み出して実行する一例を説明する。以下では、次のような状況を仮定する。動画像符号化装置のａ個の機能のうち、第１の機能はフェード機能であるとし、共有記憶部２０２に蓄積されたｐ個のプログラムのうち、第８のプログラムのみがフェード機能をサポートしているものとする。また、共有記憶部２０２に蓄積されたｓ個のシーケンスについては、第４のシーケンスでは画像はフェードせず、第５のシーケンスでは画像がフェードするものとする。さらに、第３のプロセッサは、第７のプログラムを用いて、第４のシーケンスに関する処理（例えば、マクロブロック符号化処理）を完了し、次処理として第５のシーケンスに関する処理を選択したとする。
【０１５７】
この場合、第３のプロセッサが使用している第７のプログラムは、フェード機能をサポートしていないので、第４のシーケンスを処理できない。そこで、第３のプロセッサは、共有記憶部２０２に蓄積されたｐ個のプログラムから第８のプログラムを選択し、第８のプログラムのプログラムコード２３１を共有記憶部２０２から自らのＲＡＭにダウンロードする。その後、第３のプロセッサは、ダウンロードした第８のプログラムに従って、第４のシーケンスに対してフェード機能を含んだ動画像符号化処理を実行する。
【０１５８】
以上に示すように、本実施形態に係る動画像符号化装置によれば、各プロセッサは、共有記憶部に記憶された機能情報に基づき、次の処理に必要なプログラムを選択し、選択したプログラムを転送して実行する。各プロセッサにおけるプログラムの選択、転送および実行は、他のプロセッサの動作とは独立して、任意のタイミングで行われる。このため、各プロセッサは、他のプロセッサの動作に影響を及ぼすことなく、次の処理に必要なプログラムを選択し、実行することができる。したがって、実行中の符号化処理に影響を及ぼすことなく、符号化処理中に符号化処理の内容を切り替えることができる。また、符号化処理の内容を変更したり、新たな機能を追加するときでも、符号化処理を中断することなく、これらの操作を行うことができる。
【０１５９】
（第４の実施形態）
本発明の第４の実施形態に係る動画像符号化装置は、各プロセッサが選択した処理を実行する際に異常を検出し、異常検出時には自ら停止することを特徴とする。具体的には、各プロセッサは、選択した処理を実行したときにビットストリームを生成した場合には、生成したビットストリームのサイズやシンタックスなどを検査し、これらの検査結果に基づき、選択した処理で異常が生じたか否かを判断する。
【０１６０】
図２３は、本実施形態に係る動画像符号化装置のデータ配置図である。図２３において、ｎ個のプロセッサ３０１－１～ｎおよび共有記憶部３０２は、それぞれ、図１に示すプロセッサ１－１～ｎおよび共有記憶部２を詳細化したものである。なお、図２３では、本実施形態の特徴を説明するために必要なデータのみを示し、その他のデータについては図示を省略している。図２３に示すように、各プロセッサ３０１には、プロセッサ識別情報３１１が蓄積され、共有記憶部３０２には、ｓ個の各シーケンスについて、シーケンス管理情報３２０とシーケンス処理用データ３３０とが蓄積される。第ｉのシーケンス処理用データ３３０には、第ｉのシーケンスを符号化する際に必要とされるすべてのデータ、例えば、動画像データ、制御情報、中間結果、および、ビットストリームなどが含まれる。
【０１６１】
シーケンス管理情報３２０は、プロセッサ番号３２１、開始フラグ３２２、終了フラグ３２３、エラーフラグ３２４、タイムアウトフラグ３２５、および、カウント値３２６を含んでいる。プロセッサ番号３２１は、このシーケンスについて現在、処理を実行しているプロセッサのプロセッサ識別情報３１である。開始フラグ３２２、終了フラグ３２３、エラーフラグ３２４、タイムアウトフラグ３２５、および、カウント値３２６は、いずれも

各プロセッサ３０１によって設定され、各プロセッサ３０１から参照される。開始フラグ３２２は、このシーケンスに対する処理が開始されたときに１に設定される。終了フラグ３２３は、このシーケンスに対する処理が終了したときに１に設定される。エラーフラグ３２４は、このシーケンスに対する処理で異常が検出されたときに１に設定される。タイムアウトフラグ３２５は、このシーケンスに対する処理でタイムアウトが生じたときに１に設定される。カウント値３２６は、異常が生じたプロセッサを検出するために使用される。シーケンス管理情報３２０に含まれる各データの使用方法の詳細は後述する。

【０１６２】

図２４は、各プロセッサ３０１によるメイン処理を示すフローチャートである。ｎ個のプロセッサ３０１－１～ｎは、いずれも、図２に示す符号化処理用プログラム３２を実行することにより、図２４に示すフローチャートに従って動作する。各プロセッサ３０１は、図２４に示すように、次処理選択処理（図２５）により次に実行すべき処理を選択し、選択した処理でビットストリームが生成されたときは、ビットストリーム検査処理（図２６）、シーケンス管理情報更新処理（図２７）、および、カウント値更新処理（図２８）を実行する。

【０１６３】

図２４に示す各ステップの詳細は、次のとおりである。プロセッサ３０１は、起動されると、まず、初期化処理を行う（ステップＳ３１０１）。初期化処理では、プロセッサ３０１は、自らのＲＡＭ内の符号化処理用データ（図２に示す符号化処理用データ３３）を初期化するなどの処理を行う。次に、プロセッサ３０１は、後述する次処理選択処理（図２５）によって、次に実行すべき処理を選択する（ステップＳ３１０２）。ステップＳ３１０２では、プロセッサ３０１は、次に実行すべきシーケンスおよびＧＯＰを選択し、選択したＧＯＰについて実行可能な処理のうち、一の単位処理を選択する。

【０１６４】

次に、プロセッサ３０１は、ステップＳ３１０２で選択した処理を実行する（ステップＳ３１０３）。より詳細には、プロセッサ３０１は、必要なデータを共有記憶部３０２から読み出して自らのＲＡＭ（図２に示すＲＡＭ２３）に書き込み、ＲＡＭ内のデータに対して選択した処理を実行する。加えて、プロセッサ３０１は、選択した処理でビットストリームが生成された場合を除き、選択した処理で求めた結果を自らのＲＡＭから読み出して共有記憶部３０２に書き込む。次に、プロセッサ３０１は、ステップＳ３１０３の処理でビットストリームが生成された場合はステップＳ３１０５へ、それ以外の場合にはステップＳ３１１１へ進む（ステップＳ３１０４）。

【０１６５】

ステップＳ３１０３の処理でビットストリームが生成された場合には（ステップＳ３１０４のＹＥＳ）、プロセッサ３０１は、後述するビットストリーム検査処理（図２６）により、生成されたビットストリームが正常であるか否かを検査する（ステップＳ３１０５）。次に、プロセッサ３０１は、後述するシーケンス管理情報更新処理（図２７）により、ビットストリーム検査処理の結果に基づき、シーケンス管理情報３２０を更新する（ステップＳ３１０６）。この際、プロセッサ３０１は、ビットストリーム検査処理で異常を検出した場合には、その旨をシーケンス管理情報３２０に記録する。

【０１６６】

次に、プロセッサ３０１は、ステップＳ３１０３の処理が正常終了したか否かを判断し、正常終了と判断した場合にはステップＳ３１０８へ進む（ステップＳ３１０７）。この場合、プロセッサ３０１は、ステップＳ３１０３で生成されたビットストリームを、共有記憶部３０２のシーケンス処理用データ３３０に書き込み（ステップＳ３１０８）、後述するカウント値更新処理（図２８）を実行する（ステップＳ３１０９）。なお、ステップＳ３１０７では、ビットストリーム検査処理で異常が検出された場合に加えて、後述するように、ステップＳ３１０３の処理はタイムアウトしたと他のプロセッサが判断した場合にも、正常終了でないと判断される。

【０１６７】

ステップＳ３１０３の処理でビットストリームが生成されなかった場合には（ステップＳ３１０４のＮＯ）、プロセッサ３０１は、ステップＳ３１０６と同じシーケンス管理情報更新処理（図２７）を行い（ステップＳ３１１１）、ステップＳ３１１０へ進む。なお、ステップＳ３１１１のシーケンス管理情報更新処理では、ステップＳ３１０３の処理は常に正常終了するとして扱うものとする。

【０１６８】

ステップＳ３１０３の処理が正常終了したと判断した場合には、プロセッサ３０１は、ステップＳ３１１０に到達する。この場合、プロセッサ３０１は、すべての符号化処理を終了したか否かを判断する（ステップＳ３１１０）。プロセッサ３０１は、すべての符号化処理を終了した場合には（ステップＳ３１１０のＹＥＳ）、処理を終了する。一方、処理がまだ残っている場合には（ステップＳ３１１０のＮＯ）、プロセッサ３０１は、ステップＳ３１０２へ進み、符号化処理を継続する。

【０１６９】

ステップＳ３１０３の処理が正常終了しなかったと判断した場合には（ステップＳ３１０７のＮＯ）、プロセッサ３０１は、符号化処理を継続できないと判断し、処理を終了する。この場合、プロセッサ３０１は、ステップＳ３１０３の処理でビットストリームが生成されていても、そのビットストリームを共有記憶部３０２に書き込むことなく廃棄する。このように異常を検出して自ら停止したプロセッサ３０１は、保守点検などの作業を終え、ユーザから再起動の指示があるまで停止し続ける。

【０１７０】

図２５は、次処理選択処理のフローチャートである。プロセッサ３０１は、図２５に示すように、共有記憶部３０２に記憶されたｓ個のシーケンスから次に処理すべきシーケンスを選択し、選択したシーケンスのシーケンス管理情報３２０を初期化した上で、選択したシーケンスに対して実行可能な複数の処理から次に実行すべき処理を選択する。図２５に示す処理は、他のプロセッサが実行した際に異常やタイムアウトが生じた処理（エラーフラグ３２４またはタイムアウトフラグ３２５が１である処理）を優先的に選択することを特徴とする。

【０１７１】

図２５に示す各ステップの詳細は、次のとおりである。次処理選択処理では、プロセッサ３０１は、まず、シーケンス管理情報３２０をロックし（ステップＳ３２０１）、変数ｉに１を代入する（ステップＳ３２０２）。

【０１７２】

次に、プロセッサ３０１は、ステップＳ３２０３からＳ３２０５の処理により、ｓ個の要素を有するシーケンス管理情報３２０から、エラーフラグ３２４またはタイムアウトフラグ３２５が１である要素を探す。より詳細には、プロセッサ３０１は、第ｉのエラーフラグ３２４または第ｉのタイムアウトフラグ３２５が１である場合には（ステップＳ３２０３のＹＥＳ）、ステップＳ３２１１へ進む。この場合、第ｉのシーケンスに対する処理が、次に実行すべき処理として選択される。また、第ｉのエラーフラグ３２４と第ｉのタイムアウトフラグ３２５とがともに０であり（ステップＳ３２０３のＮＯ）、かつ、変数ｉがｓ（シーケンスの個数）未満である場合には（ステップＳ３２０４のＹＥＳ）、プロセッサ３０１は、変数ｉに１を加算して（ステップＳ３２０５）、ステップＳ３２０３へ進む。

【０１７３】

第ｉのエラーフラグ３２４と第ｉのタイムアウトフラグ３２５とがともに０であり（ステップＳ３２０３のＮＯ）、かつ、変数ｉがｓ以上である場合には（ステップＳ３２０４のＮＯ）、プロセッサ３０１は、ステップＳ３２０６へ進む。この場合、プロセッサ３０１は、変数ｉに１を代入した後（ステップＳ３２０６）、ステップＳ３２０７からＳ３２０９の処理により、ｓ個の要素を有するシーケンス管理情報３２０から、開始フラグ３２２が０である要素を探す。より詳細には、プロセッサ３０１は、第ｉの開始フラグ３２２が１である場合には（ステップＳ３２０７のＹＥＳ）、ステップＳ３２１１へ進む。この場

合、第ｉのシーケンスに対する処理が、次に実行すべき処理として選択される。また、第ｉの開始フラグ３２２が１であり（ステップＳ３２０７のＮＯ）、かつ、変数ｉがｓ（シーケンスの個数）未満である場合には（ステップＳ３２０８のＹＥＳ）、プロセッサ３０１は、変数ｉに１を加算して（ステップＳ３２０９）、ステップＳ３２０７へ進む。

【０１７４】

第ｉの開始フラグ３２２が１であり（ステップＳ３２０７のＮＯ）、かつ、変数ｉがｓ以上である場合には（ステップＳ３２０８のＮＯ）、プロセッサ３０１は、ステップＳ３２１０へ進む。この場合、プロセッサ３０１は、次に実行すべき処理は存在しないと判断し、変数ｉに「次処理なし」を示す０を代入した後（ステップＳ３２１０）、ステップＳ３２１３へ進む。

【０１７５】

ステップＳ３２０３またはＳ３２０７において次に処理すべきシーケンスを選択した場合、プロセッサ３０１は、第ｉのシーケンス管理情報３２０を初期化する（ステップＳ３２１１）。すなわち、プロセッサ３０１は、第ｉのシーケンス管理情報３２０に対して、プロセッサ番号３２１に自らのプロセッサ識別情報３１を、開始フラグ３２２に１を、終了フラグ３２３、エラーフラグ３２４、タイムアウトフラグ３２５、および、カウント値３２６にいずれも０を、それぞれ設定する。次に、プロセッサ３０１は、その時点で第ｉのシーケンスに対して実行可能な単位処理のうちから一の単位処理を、次に実行すべき処理として選択する（ステップＳ３２１２）。次に、プロセッサ３０１は、シーケンス管理情報３２０をアンロックし（ステップＳ３２１３）、これをもって次処理選択処理を完了する。

【０１７６】

図２５に示す次処理選択処理では、エラーフラグ３２４またはタイムアウトフラグ３２５が１であるシーケンスおよび処理、すなわち、他のプロセッサが実行した際に異常が生じた処理が、優先的に選択される。そのような処理が存在しない場合には、開始フラグ３２２が０であるシーケンスおよび処理、すなわち、他のプロセッサによって実行されていない処理が選択される。このように異常が生じた処理を早期に実行することにより、動画像符号化処理を安全かつ確実に実行することができる。

【０１７７】

図２６は、ビットストリーム検査処理のフローチャートである。ビットストリーム検査処理では、プロセッサ３０１は、まず、ステップＳ３１０３で生成されたビットストリームのサイズ（符号量）が所定の範囲内であるか否かを調べる。より詳細には、プロセッサ３０１は、符号量が所定の最大値以下で（ステップＳ３３０１のＹＥＳ）、かつ、符号量が所定の最小値以上である場合には（ステップＳ３３０２のＹＥＳ）ステップＳ３３０３へ、それ以外の場合はステップＳ３３０６へ進む。

【０１７８】

次に、プロセッサ３０１は、ステップＳ３１０３で生成されたビットストリームの詳細なシンタックス検査を行う（ステップＳ３３０３）。次に、プロセッサ３０１は、生成されたビットストリームがシンタックス検査を通過したと判断した場合にはステップＳ３３０５へ、それ以外の場合はステップＳ３３０６へ進む（ステップＳ３３０４）。

【０１７９】

プロセッサ３０１は、生成されたビットストリームが上記３つの検査をいずれも通過した場合には、ビットストリームは正常で、ステップＳ３１０３の処理は正常終了したと判断する（ステップＳ３３０５）。これに対し、生成されたビットストリームが上記３つの検査のいずれかを通過しなかった場合には、プロセッサ３０１は、ビットストリームは異常で、ステップＳ３１０３の処理は異常終了したと判断する（ステップＳ３３０６）。プロセッサ３０１は、Ｓ３３０５またはＳ３３０６のいずれかで上記判断を行った後、ビットストリーム検査処理を完了する。

【０１８０】

図２７は、シーケンス管理情報更新処理のフローチャートである。シーケンス管理情報更

新処理では、プロセッサ３０１は、まず、シーケンス管理情報３２０をロックし（ステップＳ３４０１）、変数ｉにステップＳ３１０３で処理したシーケンスのシーケンス番号を代入する（ステップＳ３４０２）。

【０１８１】

次に、プロセッサ３０１は、ステップＳ３４０３およびＳ３４０４の処理により、自分が処理したシーケンスを他のプロセッサが処理していないことを確認する。より詳細には、プロセッサ３０１は、第ｉのプロセッサ番号３２１が自らのプロセッサ識別情報３１１と一致し（ステップＳ３４０３のＹＥＳ）、かつ、第ｉのタイムアウトフラグ３２５が０である場合には（ステップＳ３４０４のＹＥＳ）ステップＳ３４０５へ、それ以外の場合はステップＳ３４０８へ進む。

【０１８２】

前者の場合には、プロセッサ３０１は、ステップＳ３１０３の処理の実行結果に応じて、第ｉのシーケンス管理情報３２０を更新する。より詳細には、プロセッサ３０１は、ステップＳ３１０３の処理が正常終了したか否かを判断する（ステップＳ３４０５）。処理が正常終了した場合には（ステップＳ３４０５のＹＥＳ）、プロセッサ３０１は、第ｉの開始フラグ３２２に０を、第ｉの終了フラグ３２３に１をそれぞれ設定する（ステップＳ３４０６）。処理が正常終了しなかった場合には（ステップＳ３４０５のＮＯ）、プロセッサ３０１は、第ｉのエラーフラグ３２４に１を設定する（ステップＳ３４０７）。次に、プロセッサ３０１は、シーケンス管理情報３２０をアンロックし（ステップＳ３４０８）、これをもってシーケンス管理情報更新処理を完了する。

【０１８３】

図２７に示すシーケンス管理情報更新処理によれば、選択した処理を実行した際に異常が生じた場合には、異常が生じた旨が、共有記憶部３０２に記憶されたシーケンス管理情報３２０のエラーフラグ３２４に記録される。エラーフラグ３２４は、上述した次処理選択処理（図２５）において、次に実行すべき処理を選択する際に参照される。

【０１８４】

図２８は、カウント値更新処理のフローチャートである。カウント値更新処理では、プロセッサ３０１は、まず、シーケンス管理情報３２０をロックし（ステップＳ３５０１）、変数ｉに１を代入する（ステップＳ３５０２）。

【０１８５】

次に、プロセッサ３０１は、ステップＳ３５０３からＳ３５０９の処理により、各シーケンスのカウント値３２６を更新し、カウント値３２６を用いて、他のプロセッサにおける処理がタイムアウトしていることを検出する。その詳細は、次のとおりである。プロセッサ３０１は、第ｉの開始フラグ３２２が１で、かつ、第ｉの終了フラグ３２３が０である場合はステップＳ３５０４へ、それ以外の場合はステップＳ３５０８へ進む（ステップＳ３５０３）。ステップＳ３５０４では、プロセッサ３０１は、第ｉのエラーフラグ３２４または第ｉのタイムアウトフラグ３２５が１である場合はステップＳ３５０８へ、それ以外の場合はステップＳ３５０５へ進む。

【０１８６】

前者の場合、プロセッサ３０１は、第ｉのカウント値３２６に１を加算する（ステップＳ３５０５）。次に、プロセッサ３０１は、第ｉのカウント値３２６が所定の上限値以上である場合には（ステップＳ３５０６のＹＥＳ）、第ｉのタイムアウトフラグ３２５に１を設定する（ステップＳ３５０７）。

【０１８７】

次に、プロセッサ３０１は、変数ｉがｓ（シーケンスの個数）未満である場合は（ステップＳ３５０８のＹＥＳ）、変数ｉに１を加算した後（ステップＳ３５０９）、ステップＳ３５０３へ進む。この場合、プロセッサ３０１は、次のシーケンスに対してステップＳ３５０３からＳ３５０８の処理を行う。一方、変数ｉがｓ以上である場合は（ステップＳ３５０８のＮＯ）、プロセッサ３０１は、シーケンス管理情報３２０をアンロックし（ステップＳ３５１０）、これをもってカウント値更新処理を完了する。

【０１８８】
図２８に示すカウント値更新処理によれば、プロセッサで処理されているシーケンスのカウント値３２６が更新され、更新後のカウント値３２６が上限値以上であれば、そのシーケンスのタイムアウトフラグ３２５が１に設定される。タイムアウトフラグ３２５は、上述したシーケンス管理情報更新処理（図２７）において、選択した処理が正常終了したか否かを判断する際に参照される。
【０１８９】
図２４ないし図２８に示すフローチャートによれば、プロセッサ３０１は、ビットストリームのサイズおよびシンタックス、並びに、処理時間を検査することにより、選択した処理を実行した際に生じた異常を検出し、異常検出時にはその旨をシーケンス管理情報３２０に記録した上で自ら停止する。また、プロセッサ３０１は、異常検出時には、生成されたビットストリームを共有記憶部３０２に書き戻すことなく廃棄し、保守点検後にユーザから再起動の指示があるまで停止し続ける。
【０１９０】
また、プロセッサ３０１は、各シーケンスのカウント値３２６を更新し、処理開始時に初期化されたカウント値３２６が所定値以上となっている場合には、その処理はタイムアウトしたと判断し、その旨をシーケンス管理情報３２０に記録する。加えて、プロセッサ３０１は、シーケンス管理情報３２０に基づき、自らの処理がタイムアウトしたと判断されたことを検知した場合には、ビットストリームの検査で異常を検出した場合と同様に、自ら停止する。この場合、タイムアウトしたと判断された処理は、停止したプロセッサ以外のプロセッサ３０１によって優先的に実行される。
【０１９１】
図２９は、本実施形態に係る動画像符号化装置において各プロセッサが動作する様子を示すタイミングチャートである。図２９に示す例では、動画像符号化装置は第１から第６のプロセッサを有し、このうち第１のプロセッサは故障のため停止していると仮定する。また、第２ないし第６のプロセッサは、それぞれ互いに独立して、異なるシーケンスに対して符号化処理を行い、ビットストリームを生成するものとする。図２９に示す例では、第４のプロセッサが、動作中の５個のプロセッサのうちで最初に第１のシーケンスに対する符号化処理を完了している。第４のプロセッサは、符号化処理に続いて、生成したビットストリームが正常であるか否かを検査する。第４のプロセッサは、ビットストリームは正常であると判断し、時刻ｔ１以降で次処理選択処理を行う。この時点では、第１のシーケンスに対する符号化処理は既に完了しており、第２ないし第５のシーケンスに対する符号化処理は他のプロセッサによって実行されている。そこで、第４のプロセッサは、第６のシーケンスに対する符号化処理を次に実行すべき処理として選択し、その処理を開始する。
【０１９２】
次に、第５のプロセッサが、第２のシーケンスに対する符号化処理を完了したとする。第５のプロセッサは、符号化処理に続いて、生成したビットストリームが正常であるか否かを検査する。第５のプロセッサは、ビットストリームは正常でないと判断し、時刻ｔ２に自ら停止する。
【０１９３】
次に、第２のプロセッサが、第４のシーケンスに対する符号化処理を完了したとする。第２のプロセッサは、符号化処理に続いて、生成したビットストリームが正常であるか否かを検査する。第２のプロセッサは、ビットストリームは正常であると判断し、時刻ｔ３以降で次処理選択処理を行う。この時点では、第２のシーケンスに対する処理で異常が生じたことが、共有記憶部３０２に記憶されたシーケンス管理情報３２０のエラーフラグ３２４に記録されている。そこで、第２のプロセッサは、他のプロセッサが実行した際に異常が生じた、第２のシーケンスに対する符号化処理を次に実行すべき処理として選択し、その処理を開始する。
【０１９４】

次に、第３のプロセッサが、第３のシーケンスに対する符号化処理を正常に完了したとする。第３のプロセッサは、時刻ｔ４以降で次処理選択処理を行い、第７のシーケンスに対する符号化処理を開始する。次に、第６のプロセッサが、第５のシーケンスに対する符号化処理を正常に完了したとする。第６のプロセッサは、時刻ｔ５以降で次処理選択処理を行い、第８のシーケンスに対する符号化処理を開始する。

【０１９５】

図２９に示すように、各シーケンスの符号化処理で異常が生じた場合には、その処理を実行したプロセッサは停止し、他のプロセッサがその処理を代替して実行する。なお、図２９では、生成したビットストリームに異常が生じた場合を示したが、符号化処理がタイムアウトした場合も同様である。

【０１９６】

以上に示すように、本実施形態に係る動画像符号化装置によれば、各プロセッサは、共有記憶部に記憶された符号化処理の進行状態に基づき、次の処理を選択して実行するとともに、選択した処理を実行した際に異常が生じたときには自ら停止する。各プロセッサにおける処理の選択、実行、異常検出および停止は、他のプロセッサとは独立して、任意のタイミングで行われる。したがって、符号化処理の実行制御を行うプロセッサを備えることなく、異常が生じたプロセッサを符号化処理中に検出することができる。よって、符号化処理中であってもプロセッサの修理や交換を容易に行うことができ、少なくとも１個のプロセッサが正常に動作していれば符号化処理を継続することができる。

【０１９７】

この際、生成されたビットストリームのサイズやシンタックス、あるいは、選択した処理の処理時間などに基づき異常検出を行うことにより、異常が生じたプロセッサを符号化処理中に容易に検出することができる。また、共有記憶部に記憶されたカウント値が各プロセッサによって更新され、カウント値を用いて異常が生じたプロセッサが検出されるので、各プロセッサでタイマーを起動することなく処理の所要時間を算出し、異常が生じたプロセッサを検出することができる。また、プロセッサで異常が生じた旨は共有記憶部に記録され、各プロセッサは異常が生じた処理を優先的に実行するので、動画像符号化処理を安全かつ確実に実行することができる。

【０１９８】

また、ビットストリームで異常が生じたときには、異常発生前に生成されたビットストリームはそのまま利用され、以後のビットストリームを求めるために必要最小限の符号化処理が再び実行される。この際、異常が生じた処理はシーケンス管理情報に記録されるので、これを解析すれば、異常プロセッサを容易に特定することができる。したがって、装置の保守管理を容易に行い、故障による装置全体の稼働率の低下を抑えることができる。

【０１９９】

また、共有記憶部に記憶されたシーケンスや各プロセッサに記憶されたプログラムに異常やバグなどがある場合には、すべてのプロセッサで同じ異常が生じ、装置全体の動作が停止する。したがって、この場合でも異常が生じた時点で処理が中断されるので、異常なビットストリームを出力することなく、異常の原因を容易に究明することができる。

【０２００】

また、各プロセッサが選択した処理を実行した際の異常を自ら検査し、異常検出時には自ら停止するため、並列処理装置で必要とされる複雑な異常検出処理を行う必要がない。また、各プロセッサが独立して次に実行すべき処理を選択するので、異常発生時の再符号化処理を通常の次処理選択処理と同じ要領で容易に実行でき、任意のタイミングでプロセッサを追加および削除する機能を容易に追加することができる。

【０２０１】

（第５の実施形態）

本発明の第５の実施形態に係る動画像符号化装置は、分散ファイルシステム構成を有する共有記憶部を備えることを特徴とする。具体的には、共有記憶部は互いに独立してアクセスできる複数の画像記憶装置を含み、各画像記憶装置には、動きベクトル探索実行中に、

符号化対象の動画像データが分散して記憶される。

【０２０２】

図３０は、本実施形態に係る動画像符号化装置の共有記憶部の詳細な構成を示す図である。図３０において、共有記憶部４０２およびスイッチングハブ４０４は、それぞれ、図１に示す共有記憶部２およびネットワーク４を詳細化したものである。また、ｎ個のプロセッサ４０１－１～ｎは、図１に示すプロセッサ１－１～ｎに対応し、システム管理部４０３は、図１に示すシステム管理部３に対応し、画像入力部４１１は、図１に示す画像入力部１１に対応する。

【０２０３】

画像入力部４１１は、デジタルビデオカセットレコーダ（ＤＶＣＲ）４１２とディスクレコーダ４１３とを含んでいる。デジタルビデオカセットレコーダ４１２には、符号化対象の動画像データを記録したカセット状の記録媒体が装着される。ディスクレコーダ４１３は、デジタルビデオカセットレコーダ４１２から出力された動画像データを、未符号化状態のまま１フレームずつ所定のファイル形式で記憶する。

【０２０４】

共有記憶部４０２は、パラメータ記憶装置４２１、ｑ個の画像記憶装置４２２－１～ｑ、および、ビットストリーム記憶装置４２３を含んでいる。パラメータ記憶装置４２１は、各プロセッサ４０１における符号化処理に必要とされる制御パラメータを記憶しており、ｎ個のプロセッサ４０１－１～ｎのすべてからアクセスされる。ｑ個の画像記憶装置４２２－１～ｑは、各プロセッサ４０１における符号化処理に必要とされる動画像データを分散して記憶する。動画像データを分散して記憶する方法の詳細は後述する。ビットストリーム記憶装置４２３は、符号化結果のビットストリームを記憶する。

【０２０５】

共有記憶部４０２に含まれる記憶装置（図３０では（ｑ＋２）個の記憶装置）は、いずれも、例えばＮＡＳ（Network Attached Storage）などを用いて構成され、スイッチングハブ４０４にそれぞれ独立に接続される。このため、各プロセッサ４０１は、共有記憶部４０２に含まれる記憶装置に対して独立にアクセスできる。また、画像記憶装置４２２は、プロセッサ４０１からのアクセスが符号化処理のボトルネックにならない数だけ設けられる。画像記憶装置４２２の個数ｑは、上記の条件を満たす限り、プロセッサ４０１の個数ｎより少なくてもよい。例えば、プロセッサ４０１の個数ｎが１０～２０程度である場合に、画像記憶装置４２２の個数ｑが２～６程度であってもよい。

【０２０６】

スイッチングハブ４０４は、ディスクレコーダ４１３と、ｎ個のプロセッサ４０１－１～ｎと、パラメータ記憶装置４２１と、ｑ個の画像記憶装置４２２－１～ｑと、ビットストリーム記憶装置４２３との間を、スイッチ構造で接続する。スイッチングハブ４０４は、プロセッサ４０１からのアクセスが符号化処理のボトルネックにならないよう十分なデータスイッチング容量を有するものとする。なお、スイッチングハブ４０４は、接続対象の装置間をすべてスイッチ構造で接続する必要はなく、少なくとも、動画像符号化処理を実行するためにデータを交換する必要がある装置間をスイッチ構造で接続すれば足りる。

【０２０７】

図３１は、パラメータ記憶装置４２１に記憶される制御パラメータの詳細を示す図である。図３１に示すように、制御パラメータは、ｓ個の各シーケンスについて、シーケンス属性情報４３０とシーケンス管理情報４４０とを含んでいる。シーケンス管理情報４４０は、プロセッサ番号４４１、開始フラグ４４２、終了フラグ４４３、エラーフラグ４４４、タイムアウトフラグ４４５、および、カウント値４４６を含んでいる。シーケンス管理情報４４０に含まれるこれらの要素は第４の実施形態で述べたものと同じであるので（図２３を参照）、ここではその説明を省略する。

【０２０８】

シーケンス属性情報４３０は、シーケンスの属性を示す情報として、シーケンス番号４３１、開始時刻４３２、終了時刻４３３、ＧＯＰ構造情報４３４、サーバ名４３５、目標ビ

ットレート４３６、画像記憶装置番号４３７、および、ビットストリーム作成フラグ４３８などを含んでいる。シーケンス番号４３１は、シーケンスの番号を示す。開始時刻４３２および終了時刻４３３は、それぞれ、シーケンスの開始時刻および終了時刻を１フレームを単位とした時間で示す。ＧＯＰ構造情報４３４は、シーケンスのＧＯＰ構造を示す。図３１に示す例では、第１のシーケンスについてのＧＯＰ構造情報は、ＧＯＰには３枚のＩピクチャを含めて１５枚のピクチャが含まれており、ＧＯＰの先頭にあるピクチャはＩピクチャであることを示している。サーバ名４３５は、符号化処理前にシーケンスが蓄積されていたサーバの名称を示す。目標ビットレート４３６は、シーケンスを符号化するときに目標値して設定されるビットレートを示す。画像記憶装置番号４３７は、符号化処理中にシーケンスが記憶される画像記憶装置の番号を示す。図３１に示す例では、画像記憶装置番号４３７は、１からｑまでのいずれかの値を取る。ビットストリーム作成フラグ４３８は、ビットストリームが既に作成されているか否かを示す。ビットストリーム作成フラグ４３８は、ビットストリームが既に作成されている場合には１、それ以外の場合には０となる。

【０２０９】

一般に、動画像符号化処理において最も多くの計算量が必要とされる処理は、動きベクトル探索である。そこで、本実施形態に係る動画像符号化装置は、動きベクトル探索処理を含む処理を第１のパスで行い、動きベクトル探索処理後の処理を第２のパスで行う、２パス処理で動画像データを符号化する。より詳細には、第１のパスでは、動きベクトル探索の他に、特徴抽出、シーン検出、ＧＯＰ構造決定、仮符号化、ビット割り当てなどの処理が行われ、第２のパスでは、第１のパスで求めた動きベクトル探索結果を用いて本符号化が行われる。本符号化には、フレーム間差分演算、直交変換、量子化、可変長符号化、ローカル復号化、符号化制御などの処理が含まれる。なお、第２のパスにおいて、第１のパスで動きベクトル探索結果が得られていない一部のマクロブロックについて、動きベクトル探索を行ってもよい。このように動画像符号化処理を２パス処理で行う場合、第１のパスでは動きベクトル探索における計算が処理のボトルネックになるのに対して、第２のパスでは共有記憶部４０２に対するアクセスが処理のボトルネックになる。そこで、本実施形態に係る動画像符号化装置は、第１のパスの処理を実行中に、第２のパスの処理を実行するための準備として、共有記憶部４０２に動画像データを分散して記憶させる処理を行う。

【０２１０】

図３２は、各プロセッサ４０１によるメイン処理を示すフローチャートである。ｎ個のプロセッサ４０１－１～ｎは、いずれも、図２に示す符号化処理用プログラム３２を実行することにより、図３２に示すフローチャートに従って動作する。図３２に示す処理を行う前に、システム管理部４０３の制御により、デジタルビデオカセットレコーダ４１２から動画像データが読み出され、ディスクレコーダ４１３にフレームごとにファイル形式で記憶されているものとする。図３２は、システム管理部４０３から起動された後のプロセッサ４０１の動作を示している。

【０２１１】

プロセッサ４０１は、起動されると、まず、初期化処理を行う（ステップＳ４１０１）。初期化処理では、プロセッサ４０１は、自らのＲＡＭ内の符号化処理用データ（図２に示す符号化処理用データ３３）を初期化するなどの処理を行う。次に、プロセッサ４０１は、第１のパスに含まれる処理を次処理として選択し（ステップＳ４１０２）、ディスクレコーダ４１３から処理すべきシーケンスを読み出す。次に、プロセッサ４０１は、ステップＳ４１０３で読み出したシーケンスに対して、ステップＳ４１０２で選択した処理を実行する（ステップＳ４１０４）。ステップＳ４１０４では、特徴抽出、シーン検出、ＧＯＰ構造決定、動きベクトル検出、仮符号化、ビット割り当てなどの処理が行われる。

【０２１２】

次に、プロセッサ４０１は、ｑ個の画像記憶装置４２２－１～ｑのうちから一の画像記憶装置を選択し、選択した画像記憶装置にステップＳ４１０２で読み出したシーケンスを書

き込む（ステップＳ４１０５）。画像記憶装置を選択する方法については、後述する。次に、プロセッサ４０１は、パラメータ記憶装置４２１に記憶された画像記憶装置番号４３７を更新する（ステップＳ４１０６）。例えば、第ｉのシーケンスを第ｊの画像記憶装置４２２－ｊに書き込んだ場合、プロセッサ４０１は、第ｉのシーケンスの画像記憶装置番号４３７に値ｊを設定する。次に、プロセッサ４０１は、第１のパスの処理をすべて終了したか否かを判断する（ステップＳ４１０７）。第１のパスの処理をすべて終了していない場合（ステップＳ４１０７のＮＯ）、プロセッサ４０１は、ステップＳ４０１２へ進み、第１のパスの処理を再び選択して実行する。

【０２１３】

第１のパスの処理をすべて終了した場合（ステップＳ４１０７のＹＥＳ）、プロセッサ４０１は、ステップＳ４１０８へ進む。この場合、プロセッサ４０１は、第２のパスに含まれている処理を次処理として選択し（ステップＳ４１０８）パラメータ記憶装置４２１から、次処理の対象となるシーケンスの画像記憶装置番号４３７を読み出して、その番号の画像記憶装置４２２からシーケンスを読み出す（ステップＳ４１０９）。次に、プロセッサ４０１は、ステップＳ４１０９で読み出した動画像データに対して、ステップＳ４１０８で選択した処理を実行する（ステップＳ４１１０）。ステップＳ４１１０では、フレーム間差分演算、直交変換、量子化、可変長符号化、ローカル符号化、符号化制御などの処理が行われる。

【０２１４】

次に、プロセッサ４０１は、ステップＳ４１１０の処理でビットストリームが生成されたか否かを判断し（ステップＳ４１１１）、ビットストリームが生成された場合（ステップＳ４１１１のＹＥＳ）、生成されたビットストリームをビットストリーム記憶装置４２３に書き込む処理（図３３）を行う（ステップＳ４１１２）。次に、プロセッサ４０１は、第２のパスの処理をすべて終了したか否かを判断する（ステップＳ４１１３）。第２のパスの処理をすべて終了していない場合（ステップＳ４１１３のＮＯ）、プロセッサ４０１は、ステップＳ４０１８へ進み、第２のパスの処理を再び選択して実行する。第２のパスの処理をすべて終了している場合（ステップＳ４１１３のＹＥＳ）、プロセッサ４０１は、処理を終了する。

【０２１５】

このようにステップＳ４１０２からＳ４１０７では第１のパスの処理が実行され、ステップＳ４１０８からＳ４１１３では第２のパスの処理が実行される。また、第１のパスの処理を実行中に、画像入力部４１１から入力された動画像データは、ｑ個の画像記憶装置４２２－１～ｑに分散して書き込まれる。このようにして、第１のパスの処理を実行中に、第２のパスの実行に必要な分散ファイルシステムが構築される。

【０２１６】

ステップＳ４１０５において、ｑ個の画像記憶装置４２２－１～ｑからシーケンスの書き込み先となる画像記憶装置を選択する方法には、次のような方法がある。第１の方法として、プロセッサごとに書き込み先の画像記憶装置を予め決めておく方法がある。第２の方法として、シーケンス番号ごとに書き込み先の画像記憶装置を予め決めておく方法がある。第３の方法として、各画像記憶装置に記憶されているシーケンスの個数あるいはフレームの枚数をパラメータ記憶装置４２１に記憶させておき、各プロセッサ４０１がこれらの情報に基づき、動画像データの記憶量が画像記憶装置間で均一になるように、動的に画像記憶装置を選択する方法がある。プロセッサ４０１は、ステップＳ４１０６において、パラメータ記憶装置４２１に含まれる画像記憶装置番号４３７を更新する。このため、上記いずれの方法を用いた場合でも、プロセッサ４０１は、第２のパスの処理を実行するときに、処理対象のシーケンスがどの画像記憶装置に記憶されているかを知ることができる。したがって、第１のパスでシーケンスを書き込んだプロセッサと、第２のパスでシーケンスを読み出すプロセッサとが異なっていても、動画像符号化処理を正しく実行することができる。

【０２１７】

図３３は、プロセッサ４０１によるビットストリーム書き込み処理のフローチャートである。以下では、パラメータ記憶装置４２１に記憶されているｉ番目のシーケンスのビットストリーム作成フラグ４３８をＣ［ｉ］と表す。ビットストリーム書き込み処理では、プロセッサ４０１は、まず、現在処理中のシーケンスのシーケンス番号４３１を変数ｋに代入し、第ｋのシーケンスのビットストリーム作成フラグ４３８に１を設定する（ステップＳ４２０１）。次に、プロセッサ４０１は、ｋが１であるか否かを判断する（ステップＳ４２０２）。ｋが１である場合（ステップＳ４２０２のＹＥＳ）、プロセッサ４０１は、ビットストリームの先頭部分を書き込むために、ビットストリーム書き込み先ファイルＢＦを作成し、作成したファイルＢＦに生成されたビットストリームを書き込む（ステップＳ４２０３）。

【０２１８】

ステップＳ４２０２においてｋが１でない場合（ステップＳ４２０２のＮＯ）、プロセッサ４０１は、生成されたビットストリームより前の部分が既に生成されているか否かを調べるため、Ｃ［１］からＣ［ｋ－１］がすべて１であるか否かを判断する（ステップＳ４２０４）。Ｃ［１］からＣ［ｋ－１］がすべて１である場合（ステップＳ４２０４のＹＥＳ）、プロセッサ４０１は、ビットストリーム書き込み先ファイルＢＦに生成されたビットストリームを連結する（ステップＳ４２０５）。それ以外の場合（ステップＳ４０２４のＮＯ）、プロセッサ４０１は、ビットストリーム書き込み先ファイルＢＦとは別に新たにファイルF_kを作成し、ファイルF_kに生成されたビットストリームを書き込む（ステップＳ４２０６）。

【０２１９】

プロセッサ４０１は、ステップＳ４２０３、Ｓ４２０５およびＳ４２０６の後は、いずれもステップＳ４２０７へ進む。次に、プロセッサ４０１は、生成されたビットストリームより後の部分が既に生成されている否かを調べるため、Ｃ［ｋ＋１］からＣ［ｋ＋ｒ］がすべて１であるようなｒを求める（ステップＳ４２０７）。次に、プロセッサ４０１は、ｒが１以上であるか否かを調べる（ステップＳ４２０８）。ｒが１以上である場合（ステップＳ４２０８のＹＥＳ）、プロセッサ４０１は、ビットストリーム書き込み先ファイルＢＦにファイルF_{k+1}からF_{k+r}を正しい順序で連結し、その後にファイルF_{k+1}～F_{k+r}を削除する（ステップＳ４２０９）。プロセッサ４０１は、これをもってビットストリーム書き込み処理を完了する。

【０２２０】

以下、本実施形態に係る動画像符号化装置の効果を説明する。一般に、複数のプロセッサを用いて動画像符号化処理を行う場合、データ（特に、原画像データ）に対するアクセスが処理のボトルネックになり、符号化処理時間が長くなる。このため、動画像符号化処理を並列で行うときの目標である実時間処理が行えなくなる。

【０２２１】

本実施形態に係る動画像符号化装置では、動画像データに含まれるシーケンスは、複数の画像記憶装置４２２－１～ｑに分散して記憶される。また、各プロセッサ４０１は、分散して記憶されたシーケンスにそれぞれ独立にアクセスする。したがって、プロセッサ４０１からのアクセスが競合しないようにシーケンスを好適に分散して記憶することにより、プロセッサ４０１からのアクセスが符号化処理のボトルネックになることを防止することができる。

【０２２２】

また、本実施形態に係る動画像符号化装置では、ディスクレコーダ４１３から入力されたシーケンスは、動きベクトル探索実行中に共有記憶部４０２に分散して記憶される。動きベクトル探索終了後、プロセッサ４０１は、分散して記憶されたシーケンスに対して並列に符号化処理を実行する。動きベクトル探索では、計算が処理のボトルネックになるのに対して、動きベクトル探索後の処理では、入出力が処理のボトルネックになる。そこで、入出力に余裕がある動きベクトル探索実行中に、動画像データを複数の画像記憶装置に分散して記憶させることにより、入出力に余裕がない動きベクトル探索後の処理を高速に行

うための準備をする。これにより、動きベクトル探索後の処理において入出力が処理のボトルネックになることを防止することができる。以上のことから、本実施形態に係る動画像符号化装置によれば、動画像符号化処理を高速に行うことができる。よって、プロセッサ４０１の台数を十分に多くすれば、動きベクトル探索以外の動画像符号化処理を実時間で行うことも可能となる。

【０２２３】

また、複数のプロセッサを用いて並列に動画像符号化処理を行う場合、符号化結果のビットストリームは、任意のタイミングで生成される。また、生成されるビットストリームは短く分割され、その長さも一定ではないため、生成されたビットストリームを連結して、１本のビットストリームを作成する必要がある。ところが、あるビットストリームを生成したプロセッサが、そのビットストリームを連結する方法では、ビットストリームの連結順序と生成順序とが異なる場合に、プロセッサ間で同期を取る必要が生じる。よって、この方法では、プロセッサが他のプロセッサの処理を待つ時間が生じ、処理時間が長くなる。

【０２２４】

本実施形態に係る動画像符号化装置では、プロセッサ４０１がビットストリームのある部分を生成したときに、その前の部分が未完成である場合には、プロセッサ４０１は、生成したビットストリームを後で連結すべきものとしてビットストリーム記憶装置４２３に書き込む。この場合、後で連結すべきとされたビットストリームの連結処理は、他のプロセッサによって行われる。したがって、ビットストリームの連結順序と生成順序とが異なる場合でも、プロセッサは他のプロセッサの処理を待つことなく、実行可能な他の処理を開始することができる。よって、動画像符号化処理を高速に行うことができる。

【０２２５】

なお、本発明は上記各実施形態に限られるものではなく、各種の改変が可能である。各実施形態では共通したハードウェア構成が使用されるので、各実施形態の特徴を任意に組み合せて動画像符号化装置を構成することができる。したがって、各実施形態の特徴のうち任意個の特徴を有する動画像符号化装置を構成することができる。例えば、第１および第３の実施形態の特徴を有する動画像符号化装置や、第１、第４および第５の実施形態の特徴を有する動画像符号化装置を構成することができる。また、動画像符号化装置には１以上の任意の数のプロセッサが含まれていてもよく、各プロセッサの演算処理能力が異なっていてもよい。また、第５の実施形態を除き、プロセッサと共有記憶部の間の接続方式や、共有記憶部の構成は、任意に決定してよい。

【０２２６】

第１および第２の実施形態では、各プロセッサで異なる内容の次処理選択処理を実行させてもよく、一部のプロセッサにおける処理を固定化してもよい。第３の実施形態では、一部のプロセッサのみが、選択した次処理に応じてプログラムを更新してもよい。あるいは、各プロセッサのＲＡＭなどに、各プロセッサで過去に使用されたプログラムを幾つか蓄積し、選択した次処理に応じてプログラムを更新する際に、共有記憶部に蓄積されたプログラムに代えて、各プロセッサに個別に蓄積されたプログラムを使用してもよい。第４の実施形態では、選択した処理を実行してもビットストリームが生成されない場合でも、その処理が正常終了したか否かを検査してもよい。あるいは、カウント値更新処理を、ビットストリームを共有記憶部に書き込んだ直後だけでなく、各プロセッサによる処理全体の中から任意に選択した箇所で実行してもよい。

【図面の簡単な説明】

【図１】本発明の第１ないし第５の実施形態に係る動画像符号化装置の構成を示すブロック図である。

【図２】本発明の第１ないし第５の実施形態に係る動画像符号化装置の各プロセッサの詳細な構成を示す図である。

【図３】本発明の第１の実施形態に係る動画像符号化装置のデータ配置図である。

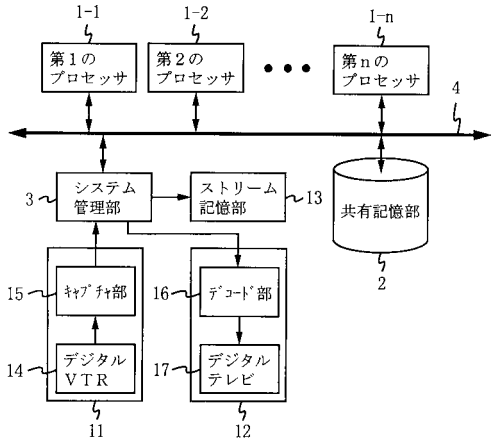
【図４】本発明の第１の実施形態に係る動画像符号化装置の各プロセッサによるメイン処

理のフローチャートである。
【図５】本発明の第１の実施形態に係る動画像符号化装置の各プロセッサによるプロセッサ追加処理のフローチャートである。
【図６】本発明の第１の実施形態に係る動画像符号化装置の各プロセッサによるプロセッサ削除処理のフローチャートである。
【図７】本発明の第１の実施形態に係る動画像符号化装置における世代更新処理のフローチャートである。
【図８】本発明の第２の実施形態に係る動画像符号化装置のデータ配置図である。
【図９】本発明の第２の実施形態に係る動画像符号化装置の各プロセッサによるメイン処理のフローチャートである。
【図１０】本発明の第２の実施形態に係る動画像符号化装置の各プロセッサによる次処理選択処理のフローチャートである。
【図１１】本発明の第２の実施形態に係る動画像符号化装置の各プロセッサによるシーケンス状態更新処理のフローチャートである。
【図１２】本発明の第２の実施形態に係る動画像符号化装置の各プロセッサが動作する様子を示すタイムチャートである。
【図１３】本発明の第３の実施形態に係る動画像符号化装置のデータ配置図である。
【図１４】本発明の第３の実施形態に係る動画像符号化装置におけるプログラム管理情報の詳細を示す図である。
【図１５】本発明の第３の実施形態に係る動画像符号化装置において実行される処理の内容の一例を示す図である。
【図１６】本発明の第３の実施形態に係る動画像符号化装置の各プロセッサによるメイン処理のフローチャートである。
【図１７】本発明の第３の実施形態に係る動画像符号化装置の各プロセッサによる次処理選択処理のフローチャートである。
【図１８】本発明の第３の実施形態に係る動画像符号化装置の各プロセッサによるプロセッサプログラム更新処理のフローチャートである。
【図１９】本発明の第３の実施形態に係る動画像符号化装置の各プロセッサによる機能情報比較処理のフローチャートである。
【図２０】本発明の第３の実施形態に係る動画像符号化装置におけるプログラム登録処理のフローチャートである。
【図２１】本発明の第３の実施形態に係る動画像符号化装置におけるプログラム削除処理のフローチャートである。
【図２２】本発明の第３の実施形態に係る動画像符号化装置におけるプログラム更新処理のフローチャートである。
【図２３】本発明の第４の実施形態に係る動画像符号化装置のデータ配置図である。
【図２４】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサによるメイン処理のフローチャートである。
【図２５】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサによる次処理選択処理のフローチャートである。
【図２６】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサによるビットストリーム検査処理のフローチャートである。
【図２７】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサによるシーケンス管理情報更新処理のフローチャートである。
【図２８】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサによるカウント値更新処理のフローチャートである。
【図２９】本発明の第４の実施形態に係る動画像符号化装置の各プロセッサが動作する様子を示すタイミングチャートである。
【図３０】本発明の第５の実施形態に係る動画像符号化装置の共有記憶部の詳細な構成を示す図である。

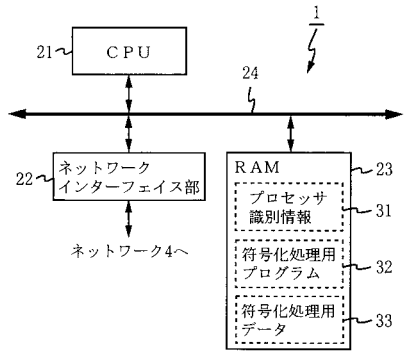
【図３１】本発明の第５の実施形態に係る動画像符号化装置における制御パラメータの詳細を示す図である。
【図３２】本発明の第５の実施形態に係る動画像符号化装置の各プロセッサによるメイン処理のフローチャートである。
【図３３】本発明の第５の実施形態に係る動画像符号化装置の各プロセッサによるビットストリーム書き込み処理のフローチャートである。
【図３４】従来の動画像符号化装置における親プロセッサの処理を示すフローチャートである。
【図３５】従来の動画像符号化装置における子プロセッサの処理を示すフローチャートである。
【図３６】従来の動画像符号化装置によって生成された異常ビットストリームを示す図である。
【符号の説明】
１、１０１、１５１、２０１、３０１、４０１…プロセッサ
２、１０２、１５２、２０２、３０２、４０２…共有記憶部
３、１０３、２０３、４０３…システム管理部
４…ネットワーク
１１、４１１…画像入力部
１２…画像出力部
１３…ストリーム記憶部
１４…デジタルＶＴＲ
１５…キャプチャ部
１６…デコード部
１７…デジタルテレビ
２１…ＣＰＵ
２２…ネットワークインターフェイス部
２３…ＲＡＭ
２４…ローカルバス
３１、１１０、１６１、３１１…プロセッサ識別情報
３２、２１１…符号化処理用プログラム
３３…符号化処理用データ
１１１…識別番号
１１２…世代番号
１２０、１７０、３２０、４４０…シーケンス管理情報
１２１、１７４、３２１、４４１…プロセッサ番号
１２２、１７３、２４０…処理状態
１３０…識別管理情報
１３１…現世代番号
１３２…世代検査値
１３３…世代更新フラグ
１３４…世代情報
１４０、１８０、２５０、３３０…シーケンス処理用データ
１６２…選択シーケンス番号
１６３…選択処理番号
１７１…現処理番号
１７２…選択可能処理番号
１７５、３２２、４４２…開始フラグ
１７６、３２３、４４３…終了フラグ
１７７、３２４、４４４…エラーフラグ
２２０…プログラム管理情報

２３１…プログラムコード
２６０…プログラム更新情報
２６１…有効フラグ
２６２…更新フラグ
２６３…削除フラグ
２６４…ダウンロード数
２７０…プログラムコード情報
２７１…プログラム名
２７２…先頭アドレス
２７３…プログラムサイズ
２８０…プログラム機能情報
２８１…項目数
２８２…機能情報
３２５、４４５…タイムアウトフラグ
３２６、４４６…カウント値
４０４…スイッチングハブ
４１２…デジタルビデオカセットレコーダ
４１３…ディスクレコーダ
４２１…パラメータ記憶装置
４２２…画像記憶装置
４２３…ビットストリーム記憶装置
４３０…シーケンス属性情報
４３１…シーケンス番号
４３２…開始時刻
４３３…終了時刻
４３４…ＧＯＰ構造情報
４３５…サーバ名
４３６…目標ビットレート
４３７…画像記憶装置番号
４３８…ビットストリーム作成フラグ

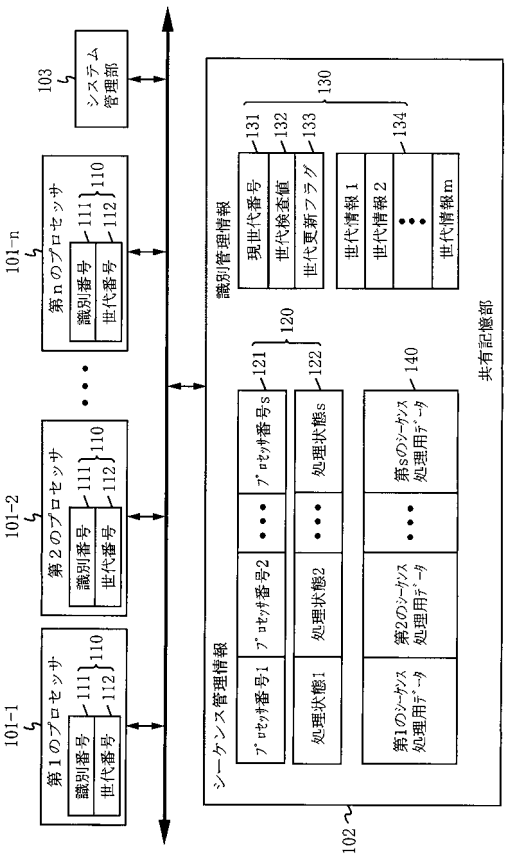
【図１】


1-1
1-2
1-n
第１のプロセッサ
第２のプロセッサ
第ｎのプロセッサ
4
システム管理部
3
ストリーム記憶部
13
共有記憶部
2
15
ｷｬﾌﾟﾁｬ部
16
ﾃﾞｺｰﾄﾞ部
14
デジタルVTR
17
デジタルテレビ
11
12

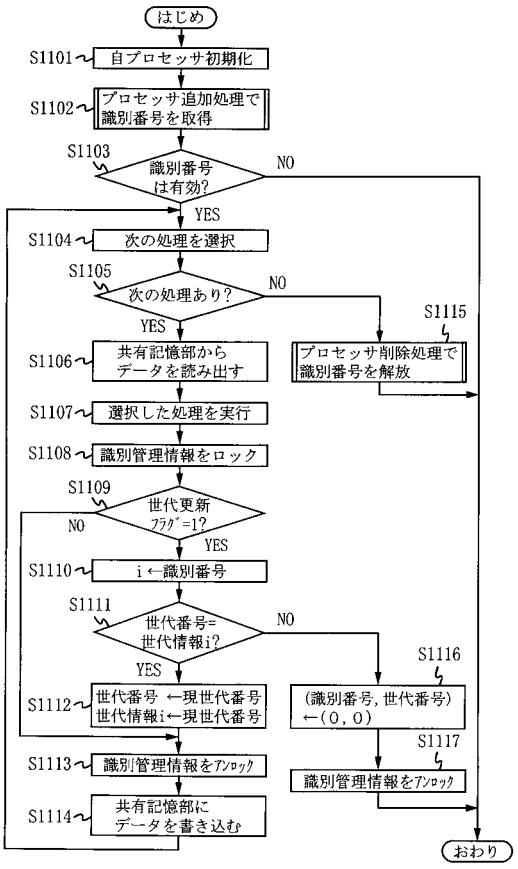
【図２】


1
21
CPU
24
22
ネットワークインターフェイス部
ネットワーク4へ
RAM
23
プロセッサ識別情報
31
符号化処理用プログラム
32
符号化処理用データ
33

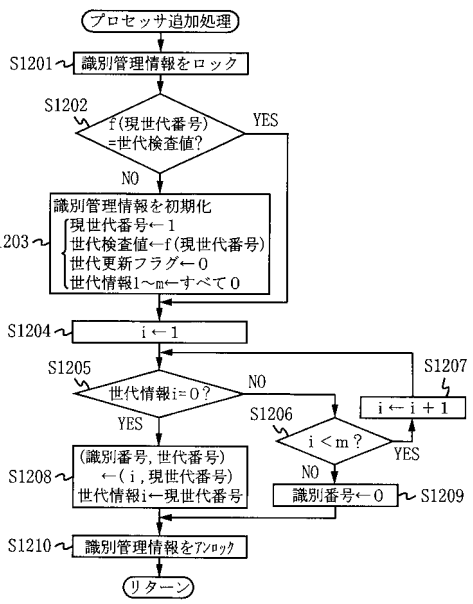
【図３】


103
システム管理部
101-n
第ｎのプロセッサ
識別番号
世代番号
111
112
110
101-2
第２のプロセッサ
101-1
第１のプロセッサ
識別管理情報
130
現世代番号
131
世代検査値
132
世代更新フラグ
133
世代情報１
世代情報２
世代情報m
134
シーケンス管理情報
120
ﾌﾟﾛｾｯｻ番号s
121
処理状態s
122
ﾌﾟﾛｾｯｻ番号2
処理状態2
ﾌﾟﾛｾｯｻ番号1
処理状態1
第sのｼｰｹﾝｽ処理用ﾃﾞｰﾀ
140
第2のｼｰｹﾝｽ処理用ﾃﾞｰﾀ
第1のｼｰｹﾝｽ処理用ﾃﾞｰﾀ
共有記憶部
102

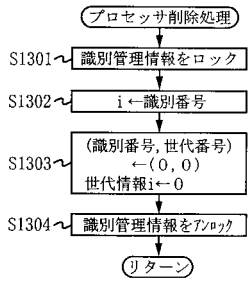
【図４】


はじめ
S1101
自プロセッサ初期化
S1102
プロセッサ追加処理で識別番号を取得
S1103
識別番号は有効?
NO
YES
S1104
次の処理を選択
S1105
次の処理あり?
NO
YES
S1115
プロセッサ削除処理で識別番号を解放
S1106
共有記憶部からデータを読み出す
S1107
選択した処理を実行
S1108
識別管理情報をロック
S1109
世代更新ﾌﾗｸﾞ=1?
NO
YES
S1110
i←識別番号
S1111
世代番号=世代情報i?
NO
YES
S1116
(識別番号,世代番号)←(0,0)
S1112
世代番号←現世代番号
世代情報i←現世代番号
S1117
識別管理情報をｱﾝﾛｯｸ
S1113
識別管理情報をｱﾝﾛｯｸ
S1114
共有記憶部にデータを書き込む
おわり

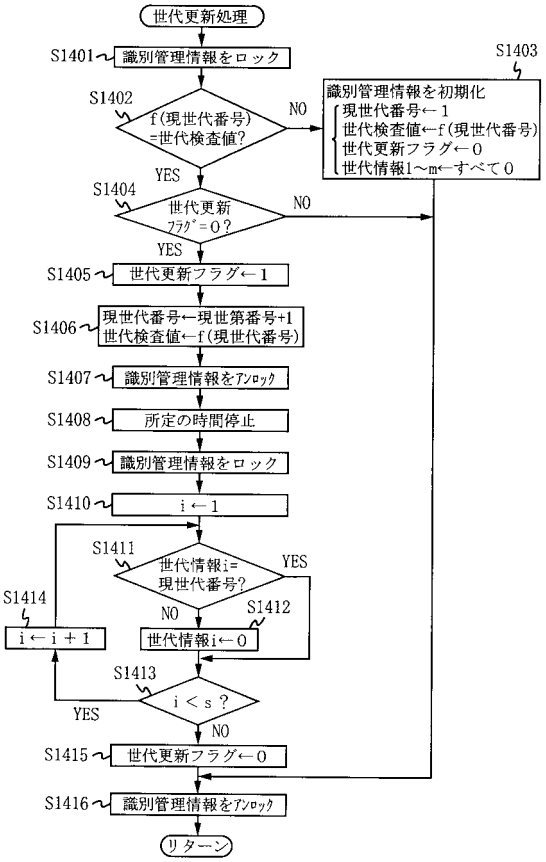
【図５】


プロセッサ追加処理
S1201
識別管理情報をロック
S1202
f(現世代番号)=世代検査値?
YES
NO
S1203
識別管理情報を初期化
現世代番号←1
世代検査値←f(現世代番号)
世代更新フラグ←0
世代情報1～m←すべて0
S1204
i←1
S1205
世代情報i=0?
NO
YES
S1207
i←i+1
S1206
i<m?
YES
NO
S1208
(識別番号,世代番号)←(i,現世代番号)
世代情報i←現世代番号
識別番号←0
S1209
S1210
識別管理情報をｱﾝﾛｯｸ
リターン

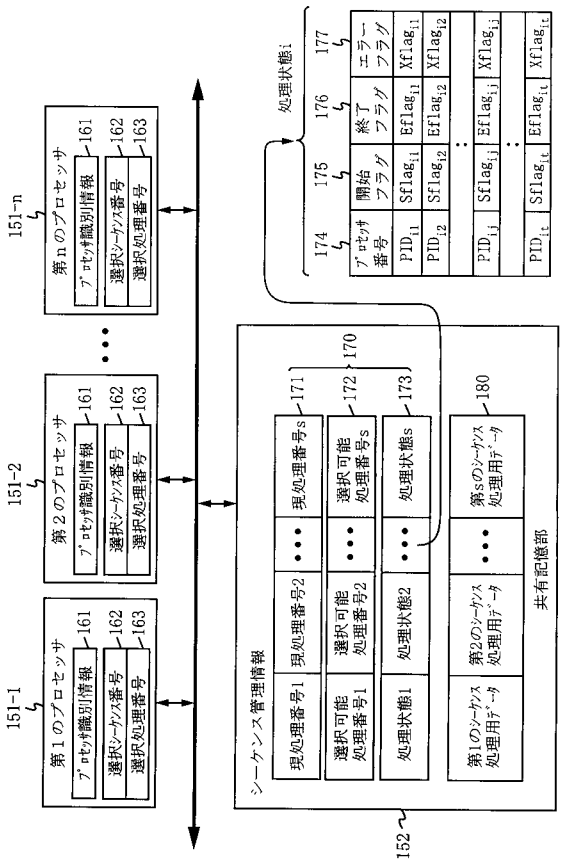
【図６】


プロセッサ削除処理
S1301
識別管理情報をロック
S1302
i←識別番号
S1303
(識別番号,世代番号)
←(0,0)
世代情報i←0
S1304
識別管理情報をｱﾝﾛｯｸ
リターン

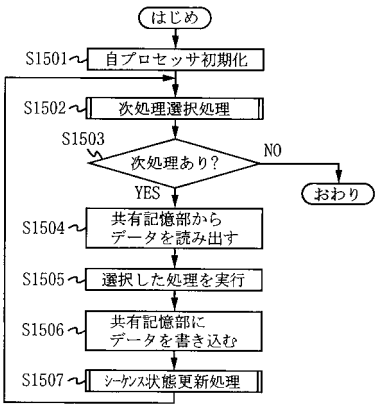
【図７】


世代更新処理
S1401
識別管理情報をロック
S1402
f(現世代番号)
=世代検査値?
NO
S1403
識別管理情報を初期化
現世代番号←1
世代検査値←f(現世代番号)
世代更新フラグ←0
世代情報1～m←すべて0
YES
S1404
世代更新
ﾌﾗｸﾞ=0?
NO
YES
S1405
世代更新フラグ←1
S1406
現世代番号←現世第番号+1
世代検査値←f(現世代番号)
S1407
識別管理情報をｱﾝﾛｯｸ
S1408
所定の時間停止
S1409
識別管理情報をロック
S1410
i←1
S1411
世代情報i=
現世代番号?
YES
NO
S1412
世代情報i←0
S1414
i←i+1
S1413
i<s?
YES
NO
S1415
世代更新フラグ←0
S1416
識別管理情報をｱﾝﾛｯｸ
リターン

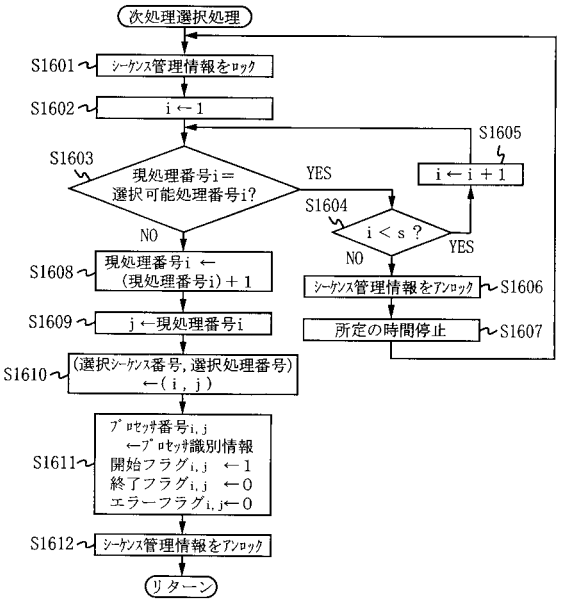
【図８】


処理状態i
プロセッサ番号
開始フラグ
終了フラグ
エラーフラグ
174
175
176
177
PID11
Sflag11
Eflag11
Xflag11
PID12
Sflag12
Eflag12
Xflag12
PIDij
Sflagij
Eflagij
Xflagij
PIDit
Sflagit
Eflagit
Xflagit
151-n
第nのプロセッサ
ﾌﾟﾛｾｯｻ識別情報
選択ｼｰｹﾝｽ番号
選択処理番号
161
162
163
151-2
第2のプロセッサ
151-1
第1のプロセッサ
シーケンス管理情報
現処理番号1
現処理番号2
現処理番号s
171
選択可能処理番号1
選択可能処理番号2
選択可能処理番号s
172
170
処理状態1
処理状態2
処理状態s
173
第1のｼｰｹﾝｽ処理用ﾃﾞｰﾀ
第2のｼｰｹﾝｽ処理用ﾃﾞｰﾀ
第sのｼｰｹﾝｽ処理用ﾃﾞｰﾀ
180
共有記憶部
152

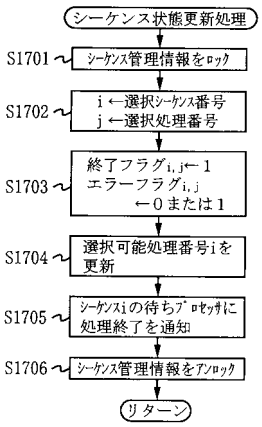
【図９】


はじめ
S1501
自プロセッサ初期化
S1502
次処理選択処理
S1503
次処理あり?
NO
おわり
YES
S1504
共有記憶部から
データを読み出す
S1505
選択した処理を実行
S1506
共有記憶部に
データを書き込む
S1507
ｼｰｹﾝｽ状態更新処理

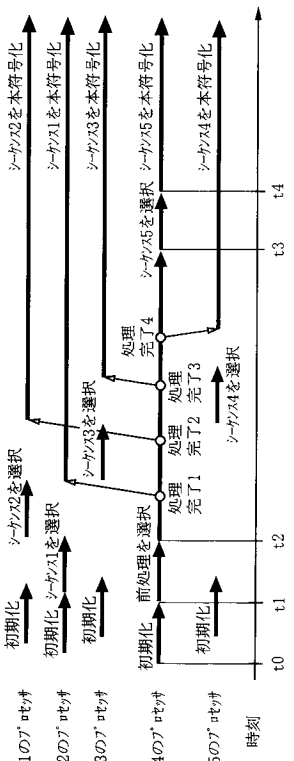
【図１０】



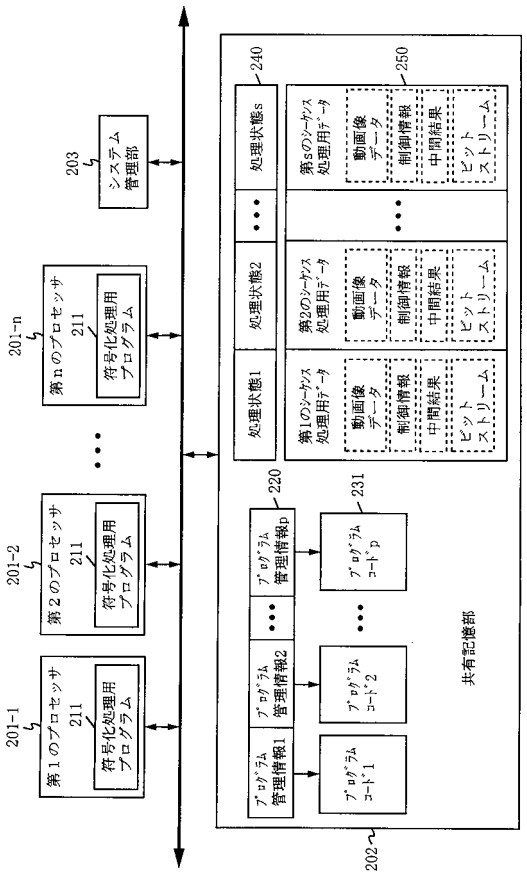
【図１１】



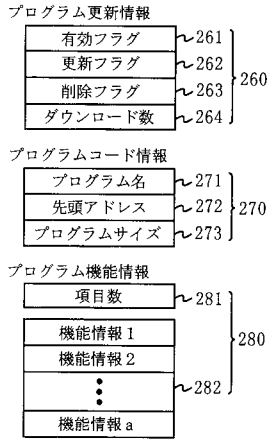
【図１２】



【図１３】



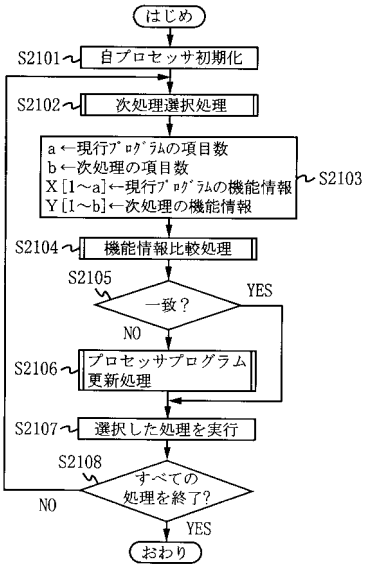
【図１４】



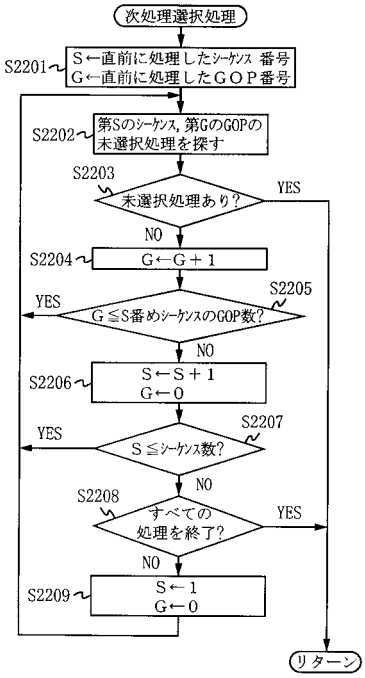
【図１５】

処理の階層	処理内容
前処理	• シーケンス分割処理 • 動画像特徴抽出処理
シーケンス層	• シーケンスヘッダ生成処理 • ＧＯＰ構造決定処理
ＧＯＰ層	• ＧＯＰヘッダ生成処理
ピクチャ層	• ピクチャヘッダ生成処理
スライス層	• スライスヘッダ生成処理
マクロブロック層	• マクロブロック符号化処理

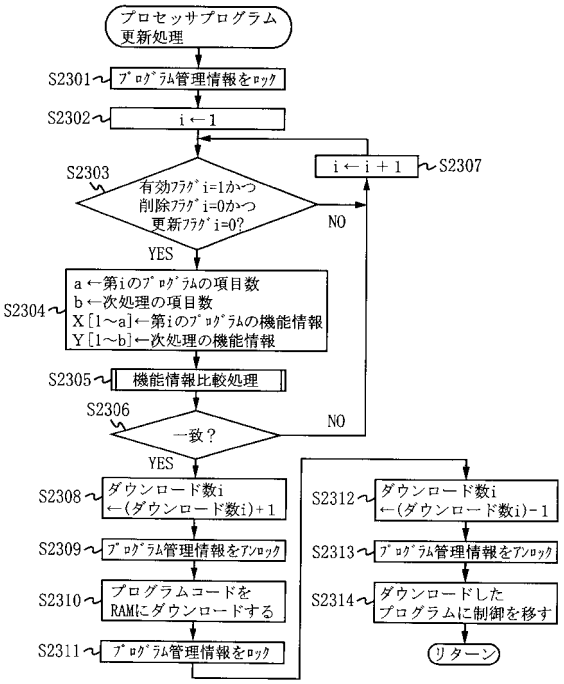
【図１６】



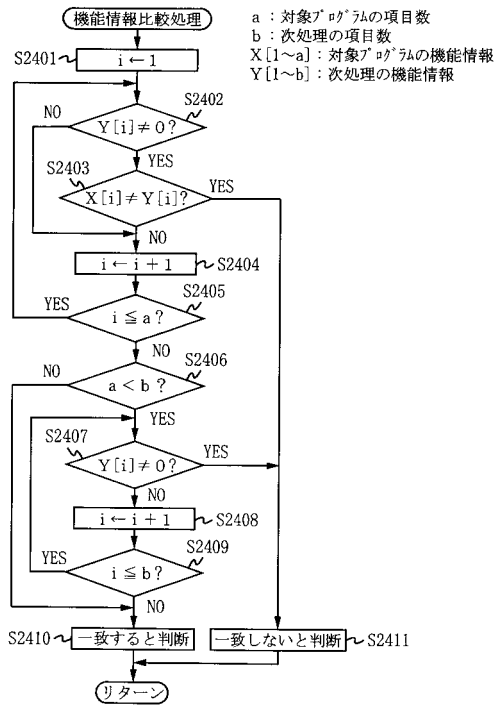
【図１７】



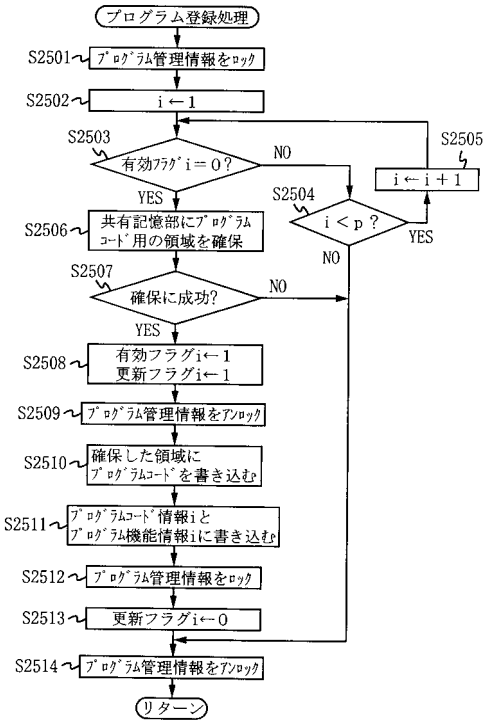
【図１８】



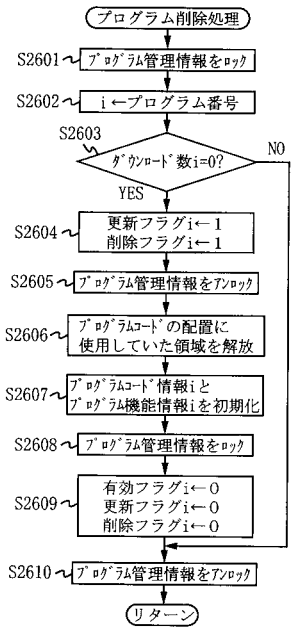
【図１９】



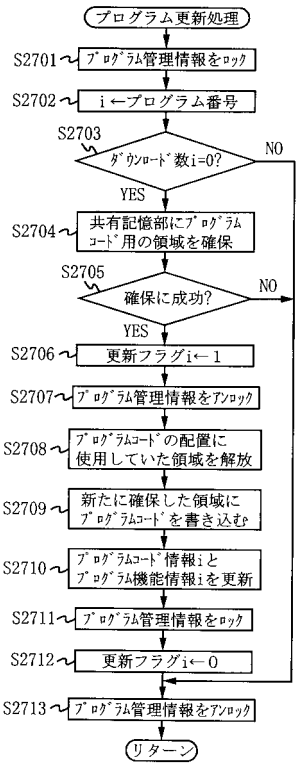
【図２０】



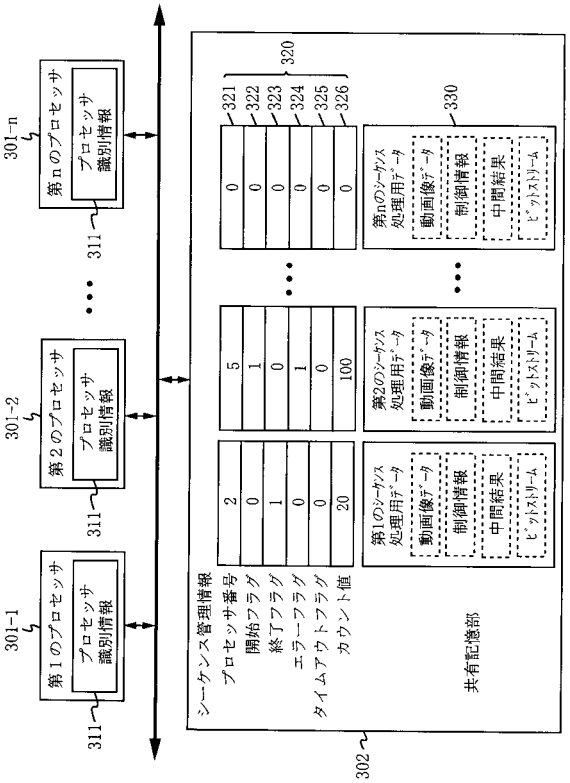
【図２１】



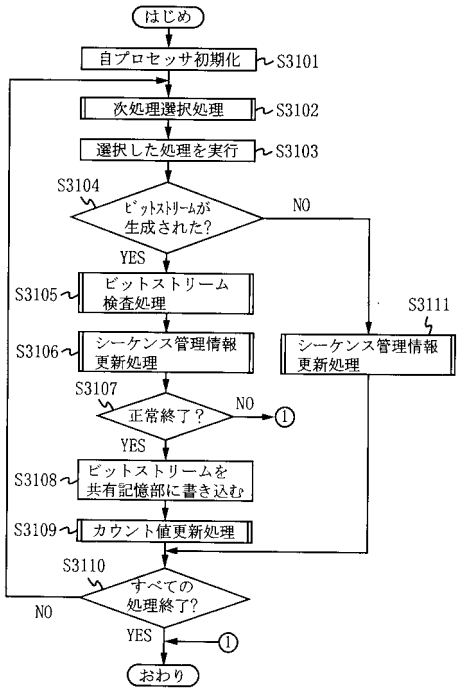
【図２２】



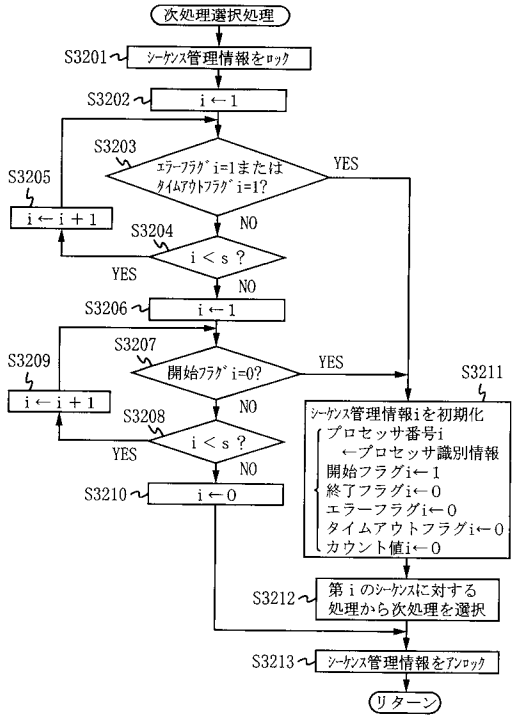
【図２３】



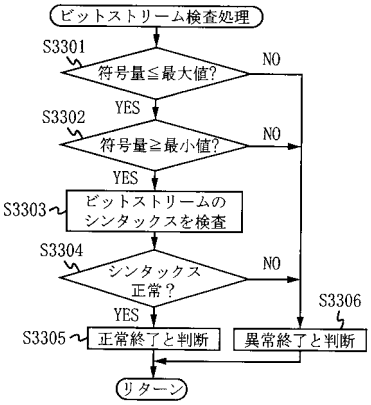
【図２４】



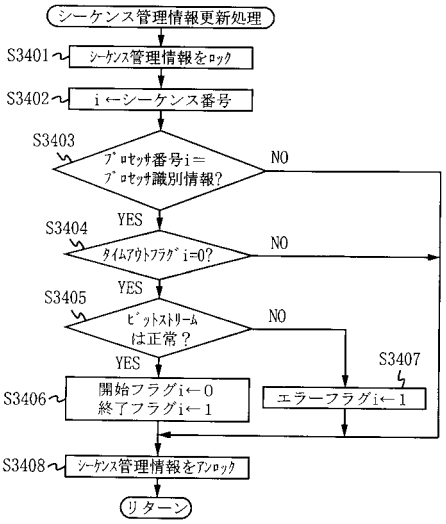
【図２５】



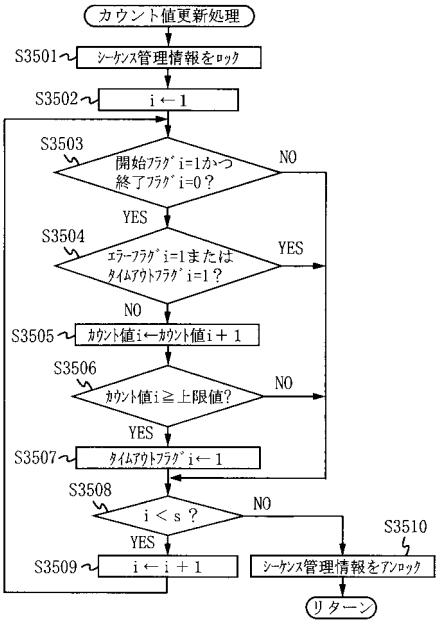
【図２６】



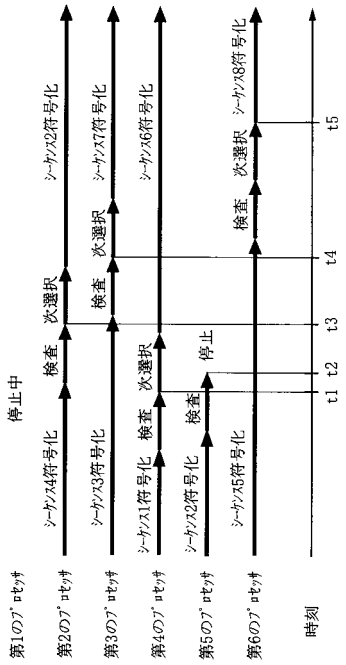
【図２７】



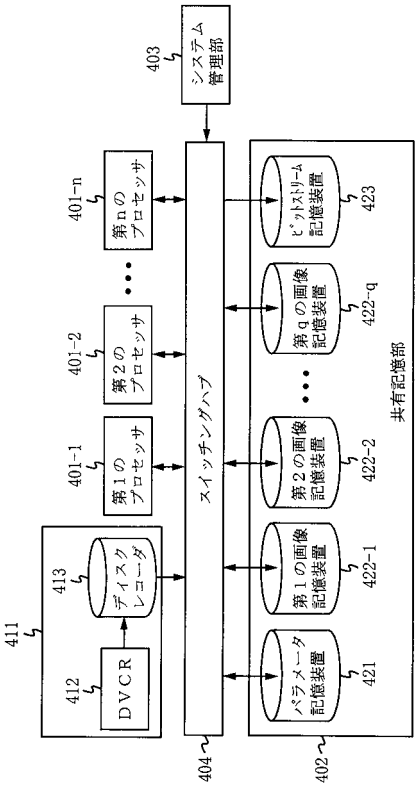
【図２８】



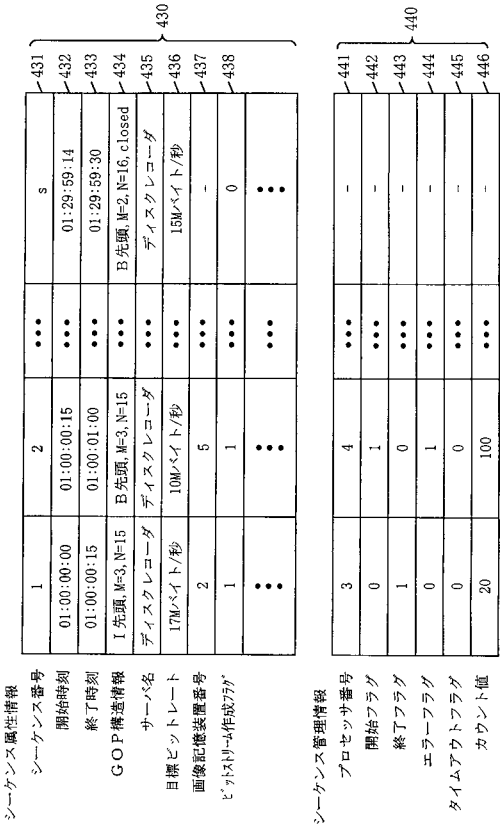
【図２９】



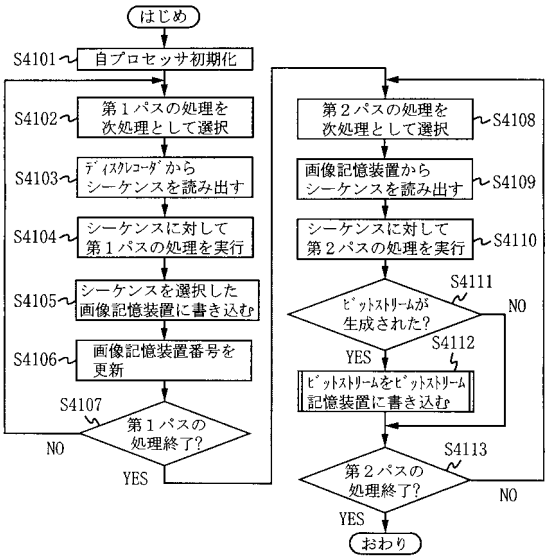
【図３０】



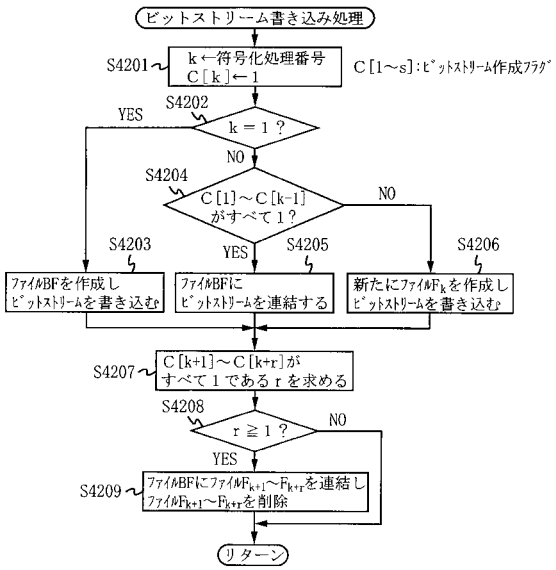
【図３１】



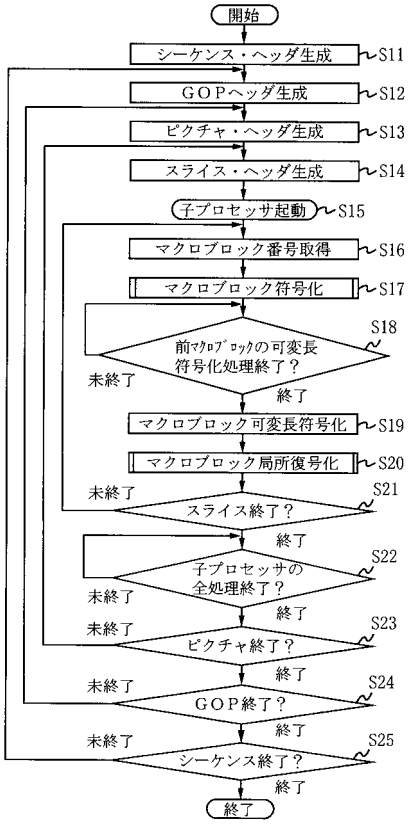
【図３２】



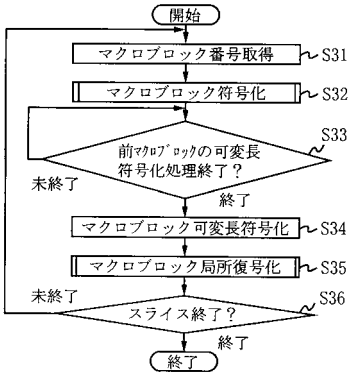
【図３３】



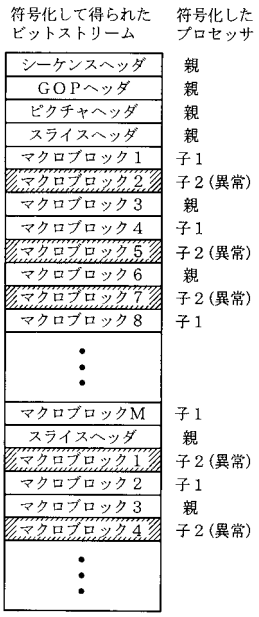
【図３４】



【図３５】


開始
マクロブロック番号取得
S31
マクロブロック符号化
S32
前ﾏｸﾛﾌﾞﾛｯｸの可変長
符号化処理終了？
S33
未終了
終了
マクロブロック可変長符号化
S34
マクロブロック局所復号化
S35
未終了
スライス終了？
S36
終了
終了

【図３６】


符号化して得られた
ビットストリーム
符号化した
プロセッサ
シーケンスヘッダ
親
GOPヘッダ
親
ピクチャヘッダ
親
スライスヘッダ
親
マクロブロック１
子１
マクロブロック２
子２(異常)
マクロブロック３
親
マクロブロック４
子１
マクロブロック５
子２(異常)
マクロブロック６
親
マクロブロック７
子２(異常)
マクロブロック８
子１
マクロブロックＭ
子１
スライスヘッダ
親
マクロブロック１
子２(異常)
マクロブロック２
子１
マクロブロック３
親
マクロブロック４
子２(異常)

フロントページの続き

(56)参考文献 特開２０００－３００４７（ＪＰ，Ａ）
特開平１１－３３１８４２（ＪＰ，Ａ）
Iwata,E., and Olukotun,K., Exploiting Coarse-Grain Parallelism in the MPEG-2 Algorithm, Stanford University Computer Systems Lab Technical Report CSL-TR-98-771, １９９８年９月, P.1-13, ＵＲＬ, http://www-hydra.stanford.edu/publications/MPEG.pdf

(58)調査した分野(Int.Cl., ＤＢ名)
H04N 7/26-50
G06F 9/38,46-54,15/16-177