平成 14 年度

学士学位論文

NOLI 符号化による DDP パラレル処理実装に関する研究

Parallel Processing of NOLI on DDP

1030313　　三木 勝

指導教員　　島村 和典

2003 年 2 月 24 日

高知工科大学 情報システム工学科

要 旨

NOLI 符号化による DDP パラレル処理実装に関する研究

三木 勝

現在は，さまざまな用途に AV データが利用されている。通信の分野も例外ではない。ディジタル動画通信では MPEG や DVTS が使用されてるが，これらの符号化方式は処理方法にフレーム内圧縮処理を用いるなど，映像遅延が発生するため，リアルタイム通信に適していない．そこで，符号化方式にはフレーミングの要らない NOLI 符号化を用いる．そして，NOLI 符号化方式を応答性の高い DDP に実装し，高速かつ高率な画像処理転送を目指す．

本研究では NOLI 符号化の並列化を図り，処理時間の短縮を検討した．NOLI 符号化の短所となる逐次処理に着目し，DDP による 2 画素並列処理や，投機実行処理を提案し，処理の高速化を図った．

キーワード　NOLI 符号化，DDP，並列処理，高速化

Abstract

Parallel Processing of NOLI on DDP

Recent network progress has made various Audio-Visual(AV) applications popular. They seems to become more popular by ways of digital video compression technologies of MPEG and DVTS. But the typical codings for MPEG and DVTS are heavy and suffering for inherent coding delay. Solving these current problems on AV communications, I focused on NOLI coding without a frame buffering and DDP architecture which enables the fast parallel processing.

This article describes the NOLI mounting techniques on DDP board. I proposed and examined the step reducing mechanism of DDP for NOLI coding.

key words near-optimal linear intarpolation, Data-driven processor , parallel processing, speed up

目次

図目次

表目次

第 1 章

背景

1.1 AV データの多様化

近年，様々な分野に AV データが使われている．通信の分野も例外ではない．文字や静止画に加えて動画，さらにはリアルタイムでの画像通信が行われるようになってきている．テレビ会議や，最近では携帯電話などでも動画像通信が可能になって，より身近なものになった．そして，ますます AV データの多様化が進み，使用される頻度が高まっていくだろう．

1.2 遅延問題の発生

現在，ディジタル動画通信に MPEG2/4 や DVTS などが使用されている．しかし，これらの方式では符号化，復号化処理が複雑であるため，映像の通信応用の際の遅延が大きな問題になると考えられる．

1.3　映像遅延の例

1.3.1　MPEG

1 画面分のデータを蓄積してから処理を行う．DCT（離散コサイン変換）は 8 × 8 ブロックを単位として行う．そのため、画像の 8line 分の画素データが必要になる．

動き補償予測では，符号化の終了した画像を復号しておき，これから符号化すべき画像を予測する．双方向予測は，符号化すべき画像を過去と未来から予測する処理である．未来フレームから予測する場合には，あらかじめ参照する画像を並べ替えて，符号化しておかなければならない．

1.3.2　DVTS

DVTS は DV を実現するもので，DV とは DCT とエントロピー符号化を行う符号化方式である。DV は 80 バイトの DIF ブロックを基本単位として符号化が行われる．この符号化方式も 1 画面の符号化が終了しないと送信が開始されないため遅延の原因になる．

第 2 章

研究目的

本研究では現在，問題になっている通信応用時の映像遅延を減少させる方法を検討する．そこで，符号化方式には MPEG や DV のようなフレーミングが行われる符号化方式ではなく，処理が簡易で処理開始が早い NOLI 符号化方式（near-optimal linear interpolation）を用いることにより，処理にかかる時間を短縮し，遅延問題の解決を図る．さらに DDP（データ駆動型プロセッサ）を用いて NOLI 符号化を実装し，並列処理することによって，より高速な画像処理転送の実現を図る．

第 3 章

NOLI 符号化と DDP について

3.1 NOLI 符号化とは

3.1.1 特徴

NOLI 符号化とは，Near-Optimal Linear Interpolation の略で，画素情報を直線近似で圧縮する方法である．

NOLI 符号化は，処理負荷が軽く，画像 1line 目の 2 番目の画素データが来た時点で処理が行われるため，処理開始時間が早い．これは，処理終了時間が早いと言いかえられる．また，フレーミングが要らないため，リアルタイム通信に適している．

短所は逐次処理であるという点である．また，衛星写真のような複雑な画像は，符号処理を行った方がデータ量が増えるという恐れが出てくる．

3.1.2 アルゴリズム

符号化対象となる幅wピクセル，高さｈピクセルの画像に対し，画面の左上隅から右方向へ走査 (水平走査) し，次に水平走査を上から下方向と走査 (垂直走査) するラスター走査を行い輝度情報を抽出する．これにより，画像情報は連続した 1 次元情報に変換される．輝度値の抽出を図 3.1 に示す．

輝度情報 p は抽出した順に格納する．画像内の位置情報は，(x，y) ではなく，抽出した順番 i($= x + y * w$) で管理を行う．NOLI 符号化は，こうして得られた $w * h$ 個の輝度情報から，復号時に直線近似で使用する輝度情報を抽出することである．以下に抽出方法を説明する。

3.1　NOLI 符号化とは

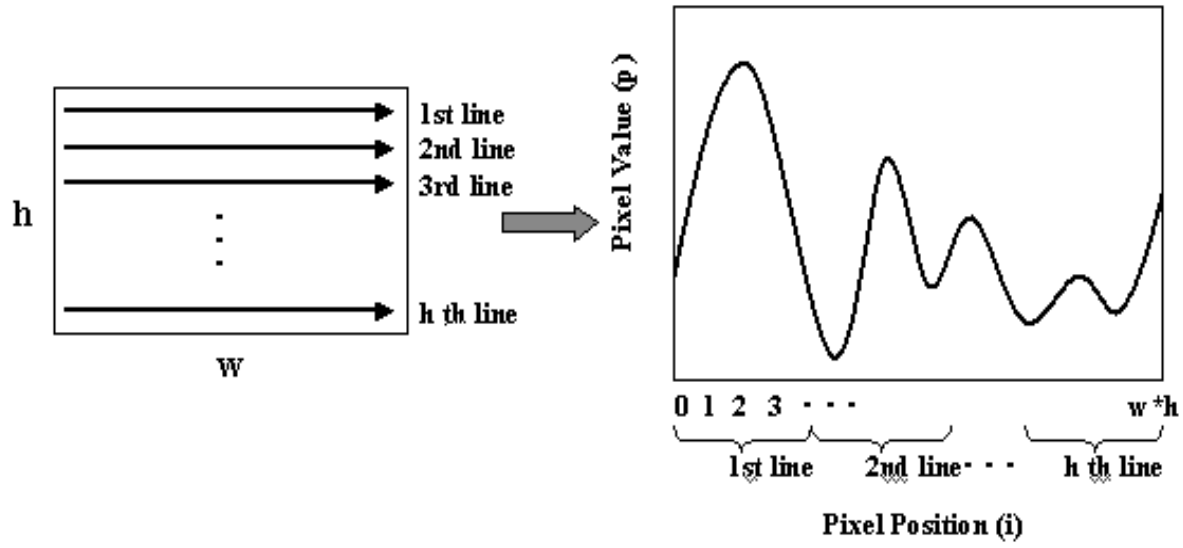

図 3.1　輝度値の抽出

先頭の画素を始点，次の画素を基準点とする．そして，許容閾値 T を定め，始点，基準点と T から Su，Sl の初期値を定める．これが直線近似可能な輝度値の範囲の上限 Su，下限 Sl となる．

直線近似の基準の初期化法を図 3.2 に示す．

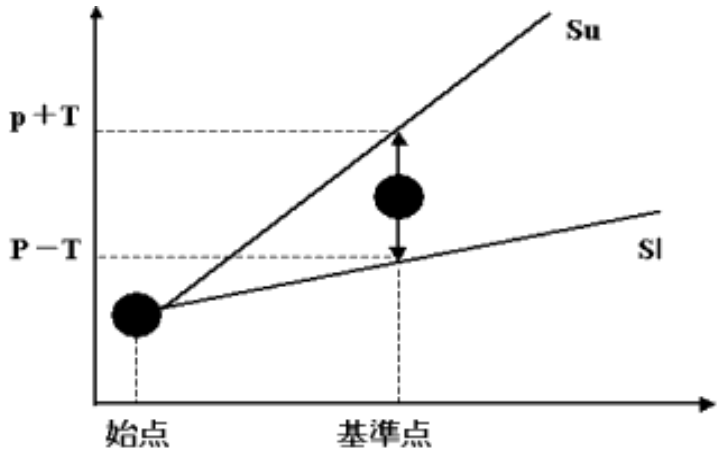

図 3.2　直線近似の基準の初期化

次に，判定点が Su，Sl の間にあるか判定を行う．判定の方法を図 3.3 を例示し，Su，Sl の更新の方法を図 3.4 に示す．

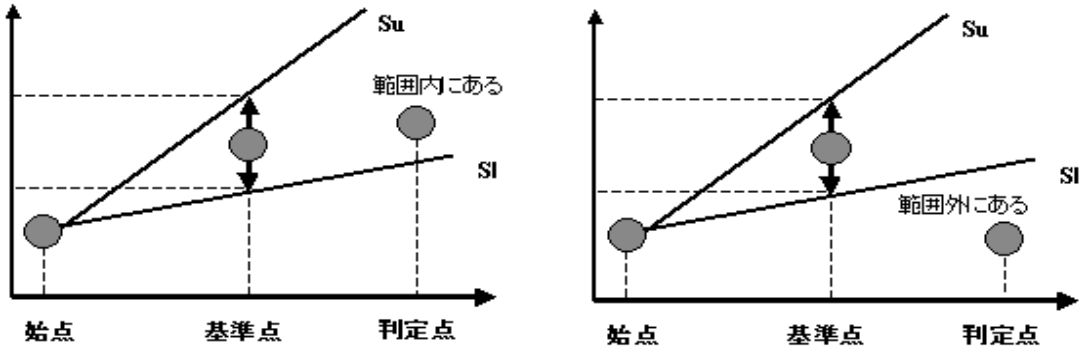

図 3.3　判定

基準点の次の画素 (判定点) が Su と Sl の範囲内にある場合，Su，Sl の更新を行う．Su，Sl 更新は始点，判定点，T から Su1，Sl1 を求め，Su と Su1，Sl と Sl1 を比較することによって，Su，Sl の範囲が狭くなる方の値を新しい Su，Sl の値と定める．そして，判定点を次の画素に移し，判定処理を繰り返す．

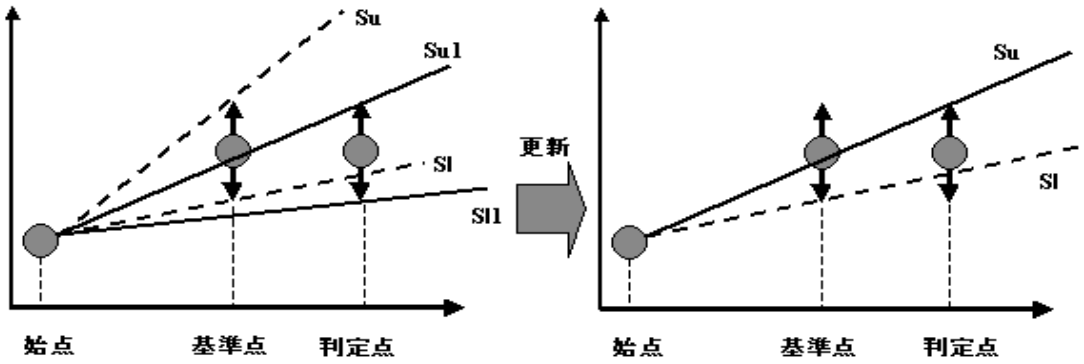

図 3.4　Su,Sl の更新

3.1　NOLI 符号化とは

基準点の次の画素 (判定点) が Su と Sl の範囲外にある場合，判定点の一つ前の画素を近似直線の終点とする．そして，次の近似直線の始点と定められる．この処理を繰り返すことで符号化を行う．

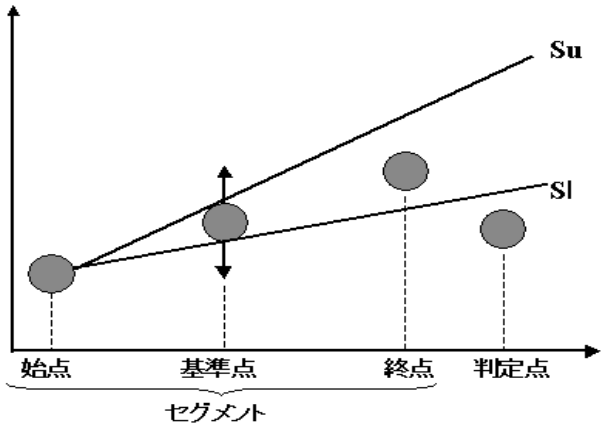

図 3.5　セグメントの定義

こうして得られた，始点と終点 (次の近時直線の始点) の間にある画素は直線近似できる．この一つの近似直線で近似できる画素群の始点から終点の範囲をセグメントと定義する．これを図 3.5 に示す．

このセグメントの境界にある画素情報 (i，p) を符号化後のデータとする．

3.2 DDP とは

3.2.1 特徴

DDP(Data-Driven Processor) はデータ駆動型プロセッサである．データに処理情報を持たせることによって，一度に多数のデータを並列処理することができる．また，スケジューリングが要らず，データを受け取った時点で処理が開始できるため，応答性の高いプロセッサである．

3.2.2 DDP パケット

DDP は情報処理対象となるデータに，データに関するタグ情報 (NODE，OPE，GEN) を付加した DDP パケットで処理を行う．DDP パケットの構成を図 3.6 に示す．

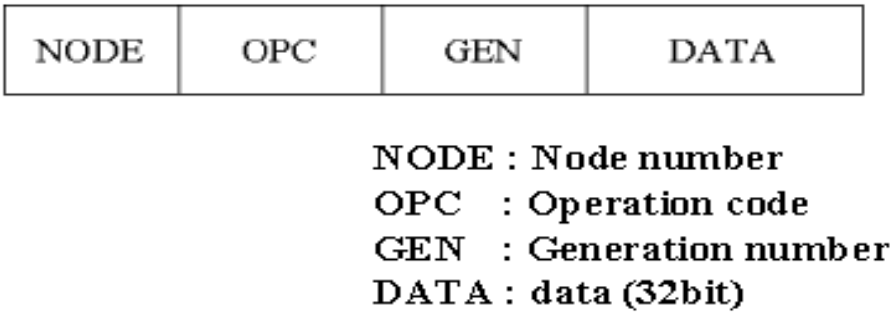

図 3.6 DDP パケット形式

- NODE(ノード番号)

  次に実行される命令が格納されている PS の番地（次の命令を読み出す際のアドレス）である．

- OPC (命令ード)

  FP(演算処理部) で実行される命令を識別するコードである．FP の説明は図 3.7 に示す．

- GEN (世代番号)

  DDP パケット間での演算は，必ず同じ世代番号同士で行われる．

- DATA (データ)

  DDP パケットに格納されているデータ本体である.

3.2.3　DDP の構造

DDP は FC/CST，FP，PS のサブシステムから構成される．その構成を図 3.7 に示し，各サブシステムの機能を以下に述べる.

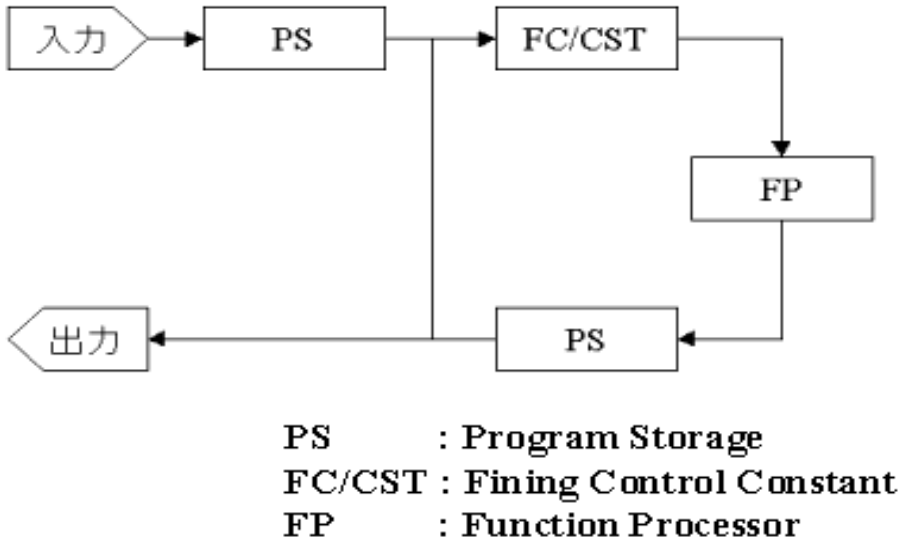

図 3.7　DDP の構造

- FC/CST(待ち合わせ記憶部)

  DDP パケットの待ち合わせを行う．ノード番号と世代番号を見て，一致する場合はその 2 つの DDP パケットを 1 つに結合し，FP へ渡す.

- FP(演算処理部)

  命令コードに従い，演算をして，その結果を PS に渡す.

- PS(プログラム記憶部)

  DDP パケットのノード番号に対応したアドレスに格納されている，命令コードとノード番号を読み出し，DDP パケットへ格納する.

3.3　NOLI 符号化と DDP との相性

NOLI 符号化の処理は輝度成分 (R，G，B や Y，U，V など) に依存性が無く，それぞれの画素成分における画像において，並列に処理ができる．

そして，DDP は応答性の高い画像処理が可能で，リアルタイム通信に向いている．また，様々な画像サイズに対応できる NOLI 符号化処理と，並列処理に対応できる DDP の特性から，単一の高精細画像通信や，複数の低解像度通信を単一システムで実現でき，様々な画像通信に対応可能である．従来の代表的な符号化である MPEG と，NOLI の画像処理転送の比較を図 3.9 に示す．

以上のことから，NOLI 符号化を DDP に実装することによって，高速で高効率な画像処理転送が可能になると考えられる．以下の図 3.8 では，30fps での画像通信を想定する．

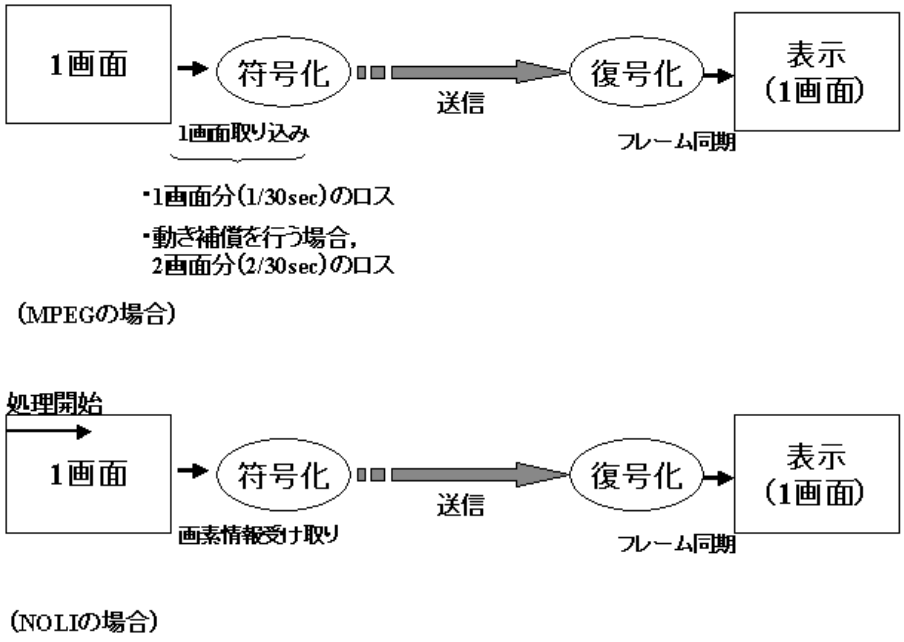

図 3.8　画像処理転送の比較

3.3　NOLI 符号化と DDP との相性

従来のプロッセッサで様々なサイズの画像を扱おうとすると，サイズの大きい画像にあわせて設計される．画像通信の場合，フレームレートは決まっている．つまり，サイズの大きい画像になればなるほど，画素 1 つに対する処理時間が短くなる．その場合に，小さい画像の処理を行おうとすると，処理と処理の間に待ち時間が発生してしまい，画像処理の効率が悪くなる．従来のプロセッサによる D1 と D3 規格のピクセル処理における待ち時間発生を図 3.9 に示す．その点，DDP で画像を扱う場合はデータが来た時点で処理を開始するので，効率的だといえる．

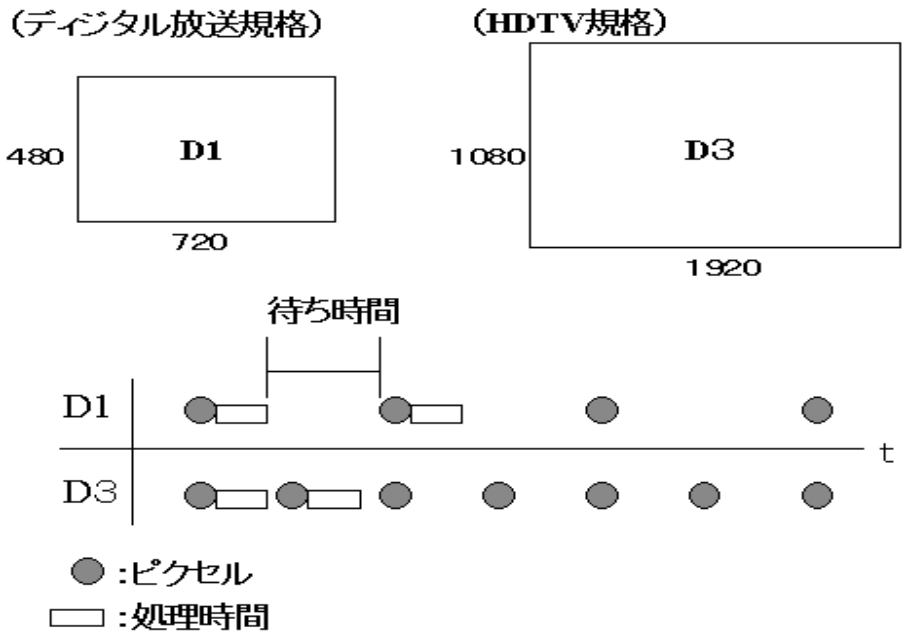

図 3.9　従来のプロセッサの画像処理

第 4 章

検討

NOLI 符号化を DDP に実装と，処理時間の短縮の検討を行った．本研究では，単色画像での検討を進めた．

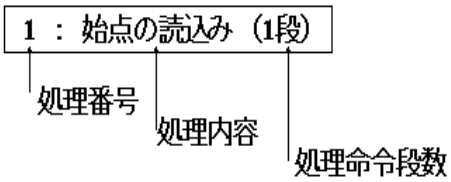

図 4.1　タイムチャートの説明

以下で示す，タイムチャート図は，図 4.1 に示す形式を基準とする．

ここで，処理番号は以下の説明で使用する．処理命令段数は処理にかかった DDP での命令段数を示す．

4.1　NOLI 符号化の実装

DDP に NOLI 符号化処理の実装を行った．

- 世代番号割当ての検討

  DDP は，データとタグ情報を一つの DDP パケットとして扱う．

  また，NOLI 符号化を実行するには，ラスター走査によって得られた，輝度値 p と画素の 1 次元座標情報 i が必要である．そこで, DDP で処理を行う場合，輝度値をデータ (DATA)，1 次元座標情報をタグ情報の識別子の一つである世代番号（GEN）に割り当てて処理を行う．

しかし，実際に使用した DDP 評価システムでは，上記と異なる世代番号が，自動的に割当てられてしまう．そこで，NOLI 符号化の前処理として，自動的に割り振られた世代番号に画素の 1 次元情報を割当て直す処理を行った．

- SDRAM への格納

  1 画素にかかる処理時間と，画素データが DDP に投入される間隔とを比較すると，画素データの投入間隔が短いため，投入された画素データを一時的に SDRAM へ格納して処理を進める．

  そして，1 番目の画素の格納をトリガーとし，SDRAM への画素データの格納と並行して，NOLI 符号化の処理を開始するようにした．

逐次処理の処理の流れを図 4.2 に示す．

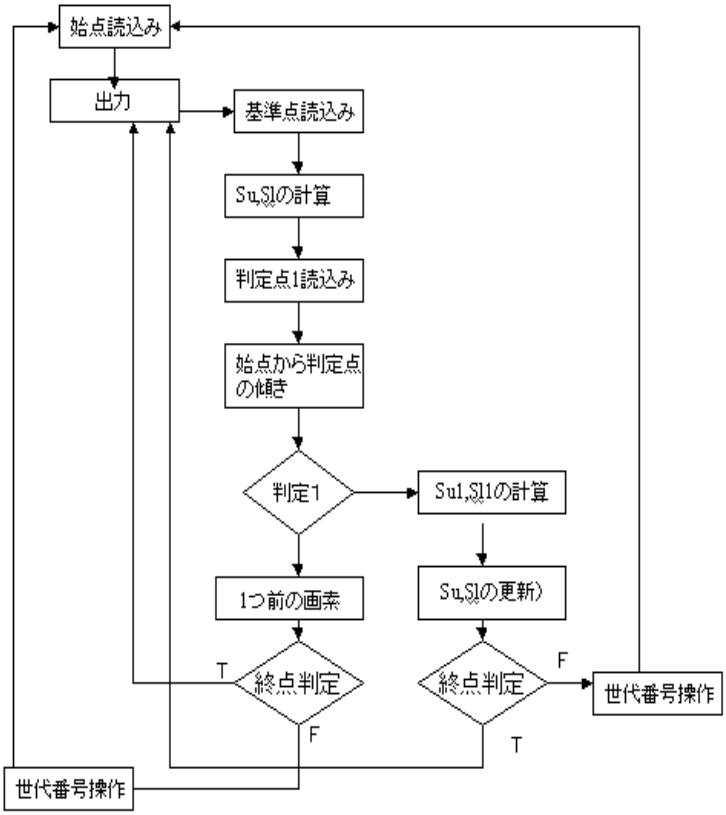

図 4.2　逐次処理のフローチャート

- 求めたパラメータをメモリに格納

  NOLI 符号化では，世代番号の異なるデータ同士で計算する必要がある．

  そこで求めたパラメータをメモリに格納する．そして，計算時にメモリの値を呼び出すことによって，世代番号間の隔たりを無くし，世代番号操作をすることなく，計算を可能にした．

性能評価

NOLI 符号化は逐次処理であり，1 画素単位で処理を進めていくため，非常に処理効率が悪い．次節以降にこの改善案を述べる．

4.2　検討内容

4.2.1　処理の並列化

NOLI 符号化の処理の中で，並列化できる部分を並列処理し，実装を行った．実装した並列処理のフローチャートを図 4.3 に，タイムチャートを図 4.4 に示す．処理の並列化について図 4.4 を基にして，説明を行う．

- セグメント判定の前処理

  判定点が直線近似できるか判定するために必要なパラメータの計算を並列化し，より早い段階での判定を可能にした．

  判定に必要なパラメータである，Su，Sl を処理 4，始点から判定点の傾きを処理 7 で求めている．そこで，これらの処理を並列化する．始点の読込み (処理 1) をトリガーとして，基準点読込み (処理 3) と判定点読込み (処理 6) を並列処理し，始点の画素情報 (i,p) を出力する処理 2 と，処理 4 と処理 7 を並列化し，Su，Sl の計算，判定点までの傾きの計算を同時に行った結果，判定 1 の開始が逐次処理に比べて，処理命令で 9 段，短縮するすることができた．

4.2　検討内容

- セグメント判定の後処理

判定の結果に関わらず，判定後に使用されるパラメータを判定前に計算しておくことによって、判定後に行う処理命令段数を短縮し，高速化を図った．先に述べた，セグメント判定の前処理と並行に，Su1，Sl1 の計算 (処理 9) を行い，Su1，Sl1 を求める．そして，判定 1 と同時に Su，Sl の更新 1（処理 10) を行い，更新した Su,Sl を仮メモリに格納しておく．（通常は同セグメントと判断された後に処理 4 が開始する．）

そうすることによって，判定 1 で同セグメントと判定された時点で仮メモリから Su，Sl を読み出す処理 10-A のみで，判定後の処理が終了する．そして，早い段階で次の判定点に処理を移すことができる．

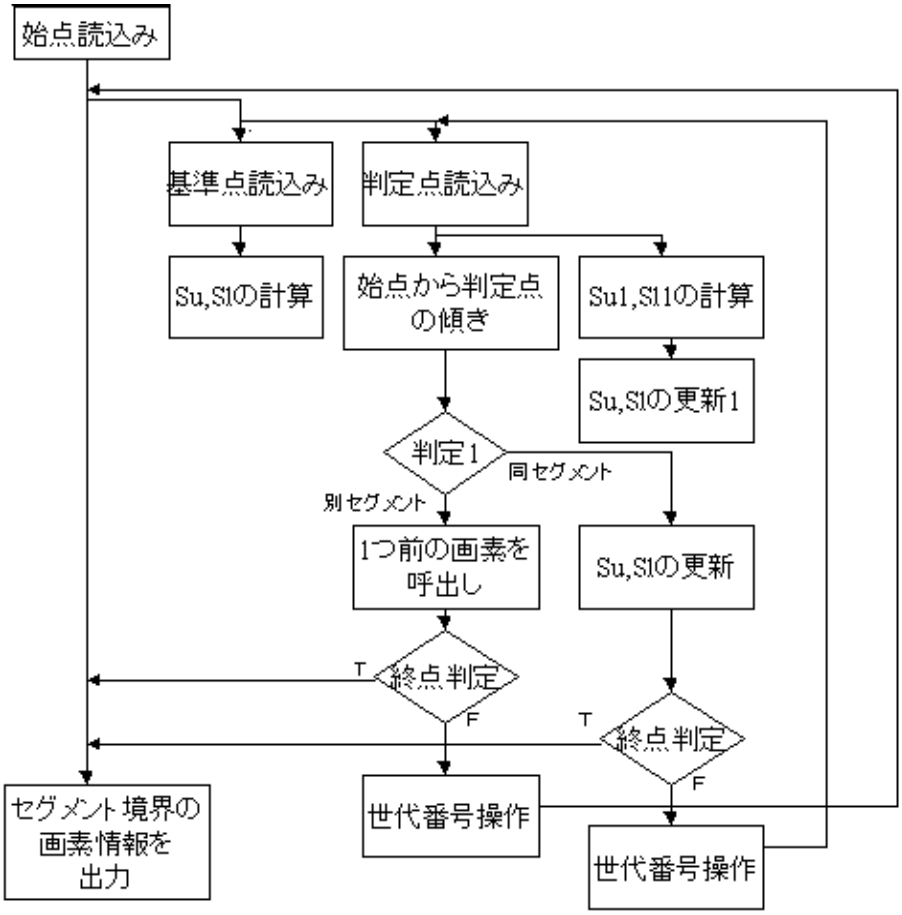

図 4.3　パラメータ計算並列化フローチャート

4.2　検討内容

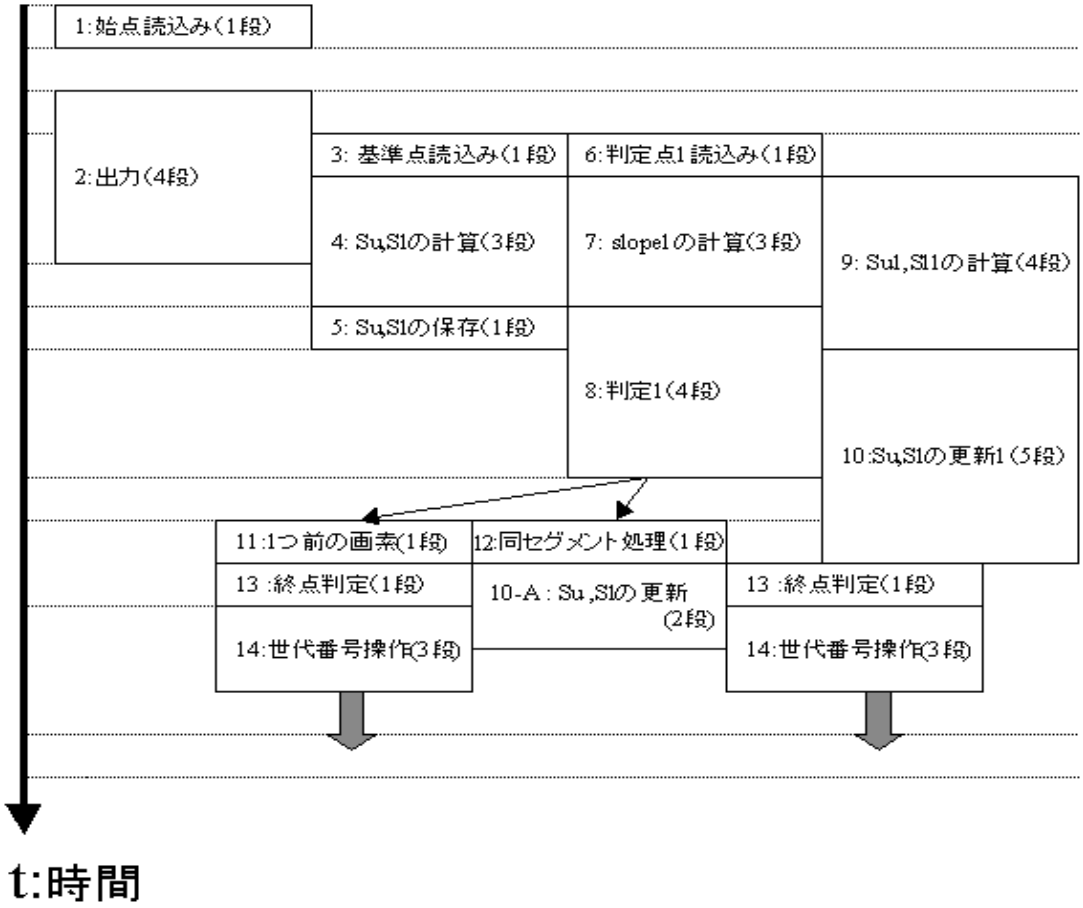

図 4.4　パラメータ計算並列化のタイムチャート

性能評価

処理の並列化により，処理時間を短縮することができた．しかし，逐次処理という性質から，まだまだ理想的性能は見込めない．

そこで，並列性を上げる検討を行った．その点について，次節で説明する．

4.2.2　NOLI の高速化の方法

さらに高速化を図るため，2 画素の並列処理を検討した．

この並列処理のフローチャートを図 4.5 に，タイムチャートを図 4.6 に示す．

- 判定点の並列化

  この説明は図 4.6 を参照して行う．

  判定点を 2 画素読込み，処理を並列に進めることによって，判定 2 の開始を早い段階で実行可能にした．

  処理 6 と処理 6-A で判定点 1 と判定点 2 を同時に読込み，判定 2 に必要なパラメータを求めておき，判定 1 が終了した時点で判定 2 を行うようにする．

  判定 2 に必要なパラメータは，判定点 1 の時点で，更新が行われた Su，Sl，始点から判定点 2 までの傾きである．これらのパラメータは処理 10 と処理 7-A で求めている．そこで，始点の読込み (処理 1) をトリガーとして，処理 3，処理 6，処理 6-A を同時に行い，基準点，判定点 1 に加え，判定点 2 を読込む．そして，7-A で始点から判定点 2 までの傾きを求めておくことによって，処理 10 が終了し Su,Sl の更新結果が出た時点で，判定 2 を実行できるようにした．

4.2　検討内容

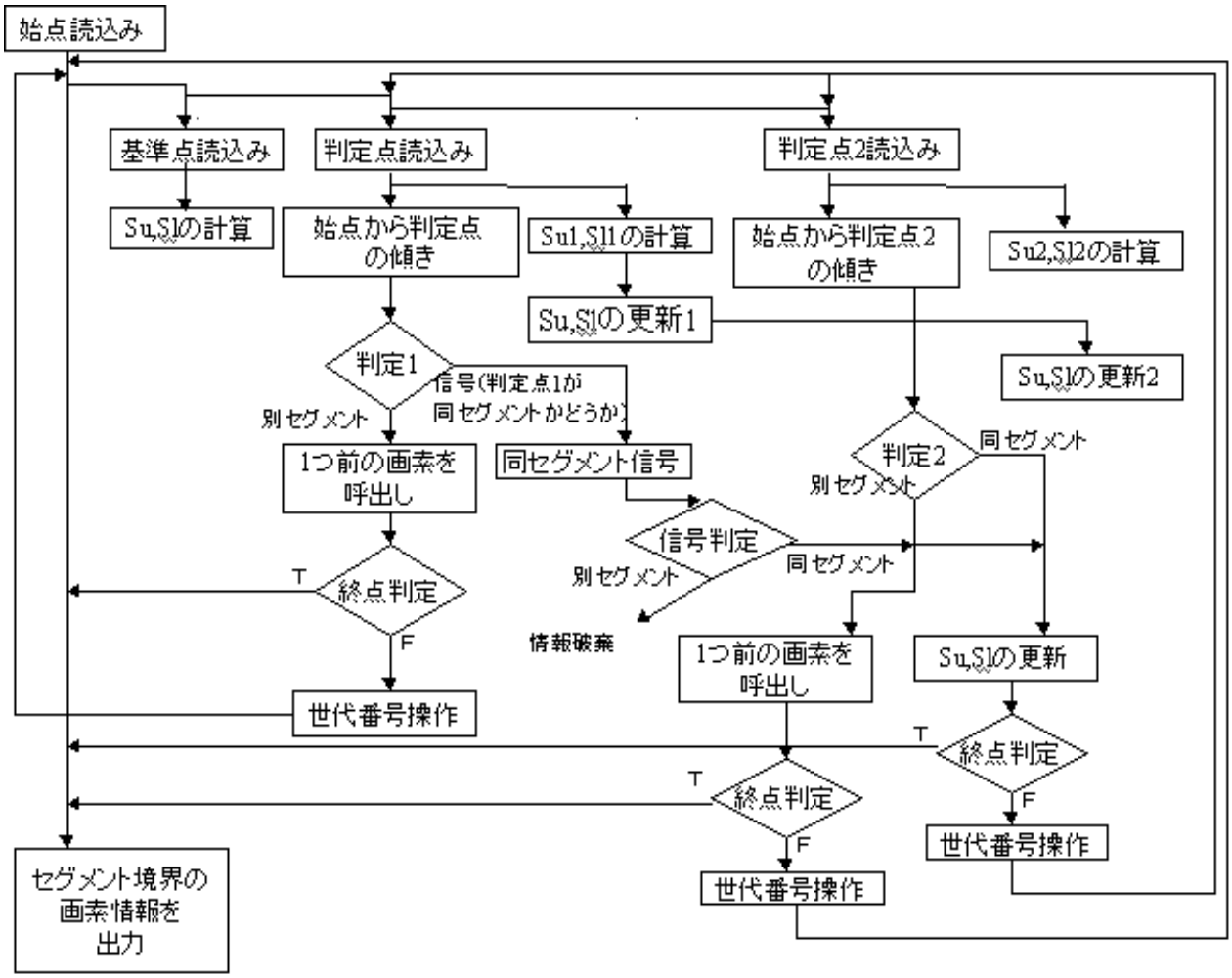

図 4.5　2 画素並列化フローチャート

4.2　検討内容

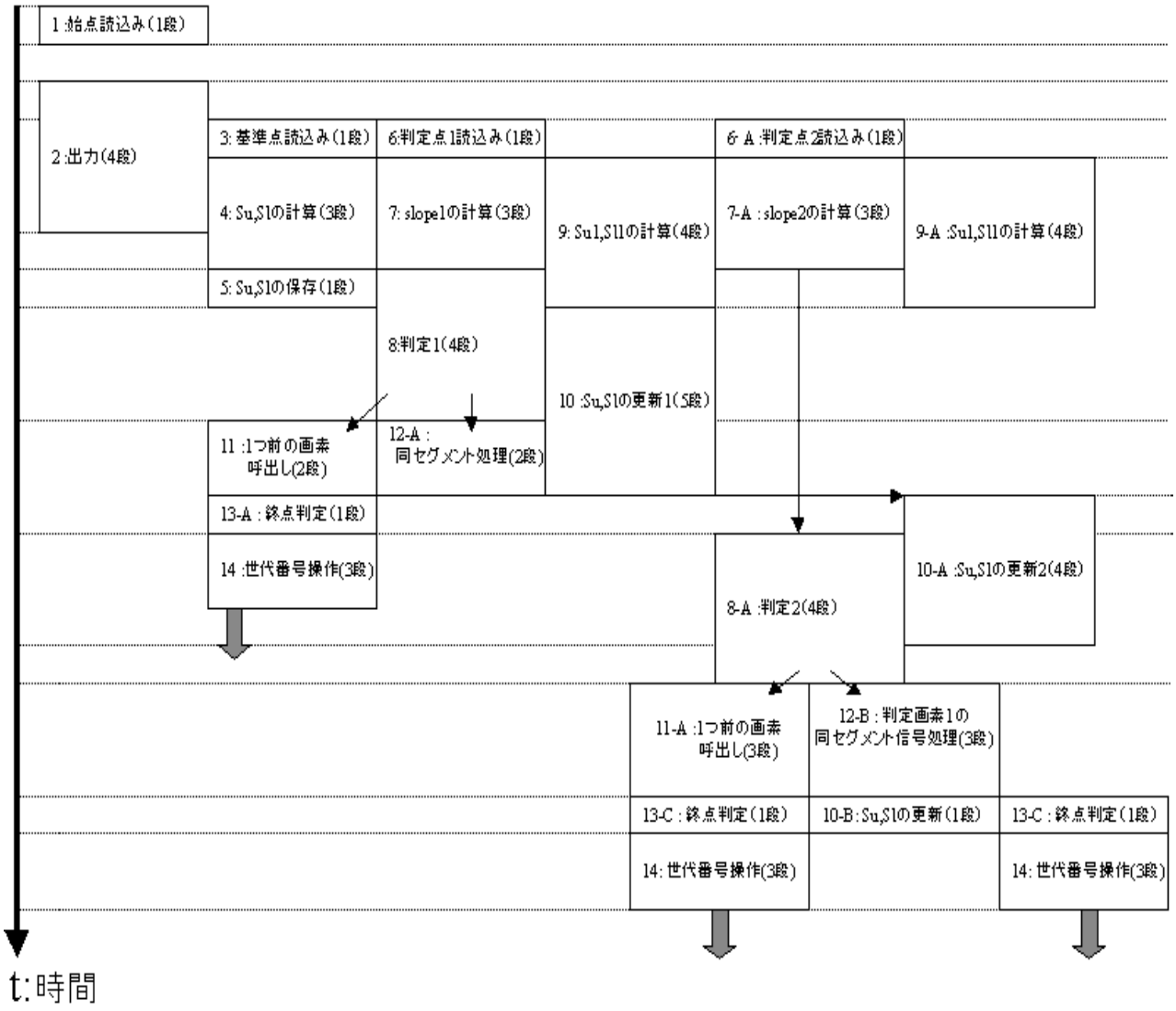

図 4.6　2 画素並列のタイムチャート

性能評価

2 画素を同時に判定することによって，さらに処理時間を短縮することができた．

しかし，判定点 1 での Su，Sl の更新が終了しないと判定 2，判定点 2 での Su，Sl の更新 2 の処理が開始できないため，完全な並列化にはならなかった．すなわち，処理 10 →処理 8-A と処理 10 →処理 10-A は並列処理できない．

4.2.3　DDP 処理命令の統合

頻繁に使われている命令，複雑な処理部分を縮小することによって，高速化を図る．

現在，ネックになっている部分は，図 4.6 の処理 9，処理 10 の部分である．この処理段数が長いため，判定点 2 の判定 (処理 8-A) の処理が遅れてしまう．そこで，処理 9，処理 10 に使われている命令を短縮することで，さらに性能が上がると考えている．

4.3　結果

上記の逐次処理，パラメータ計算の並列化，2 画素同時の判定の 3 パターンについて検証を行った．その結果，2 画素同時に判定する場合が，最も処理時間を短縮することができた．

今回，検証に用いた画像データは，全ての画素が同一セグメントと判定される画像（同セグメント），全ての画素が別のセグメントと判定さる画像 (別セグメント) である．これはそれぞれ，処理時間の最も短いと考えられる画像と，最も長いと考えられる画像である．

同セグメント，別セグメントの場合について，処理命令段数を表 4.1 に示す．

処理	同セグメント	別セグメント
逐次処理	32 段	25 段
並列処理（1 画素）	16 段	16 段
並列処理（2 画素）	12 段	16 段

表 4.1　判定結果別の処理命令段数

また，同一セグメントと判定される輝度値の区間を大きくしていき，処理時間の推移を確認した．それぞれの画像に対する処理時間を図 4.7 に示す．

処理時間の単位は step である（1step=3.5nsec).

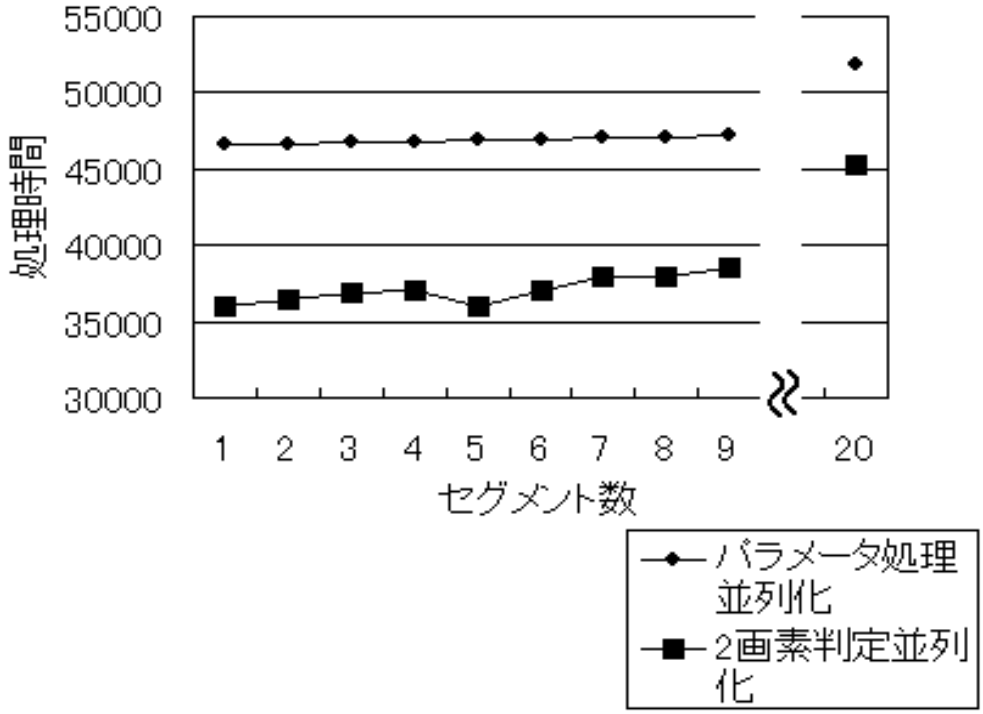

図 4.7　画像に対する処理時間

4.4　考察

図 4.6 において，の判定 1 で別セグメントと判定された場合，2 画素並列処理の効果が現れない．つまり，簡単な画像になればなるほど，2 画素並列処理の効果が発揮される．

2 画素の並列処理について，Su1,Sl1 の計算 (処理 9)，Su,Sl の更新 1(処理 10) の部分の処理が長く，判定点 2 の処理開始時間を遅らせている．この部分の短縮化が必要である．現段階ではまだ処理効率が悪く，利用に耐えない．今後は，さらに処理の効率化を検討する必要がある．

そこで，投機実行を採用することを考えた．

- 投機実行とは

1. DDP パケットのフラグセットを行う命令 (図 4.8 の a) で flag をセットする．
2. セットされた flag は派生する DDP パケットに継承される．図 4.8 では 3 つの処理に分岐している．
3. 分岐した処理のうち，いずれかで delete 命令が実行されると，他の処理 (図 4.8 の b) はナノ PE 上から抹消される．

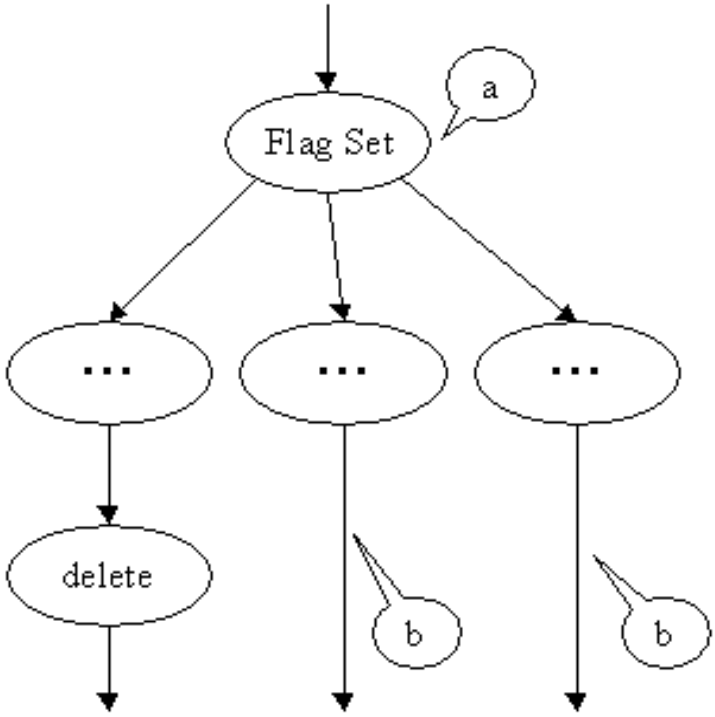

図 4.8 投機実行

この投機実行を応用することによって，セグメント判定と並列して，同セグメントと判定された場合の処理と，別セグメントと判定された時の処理を，同時に実行する．そして，セグメント判定の結果に従って，どちらか一方の処理（同セグメント，別セグメント）を強制終了させる．この投機実行を行えば，判定の後処理と判定を完全に並列化できるため，さらなる処理性能の向上が期待できる．また，画質の改善，伝送プロトコルの面でも検討を進めていく必要がある．

第 5 章

まとめ

本研究では，画像処理転送の符号化の処理時間に着目し，より高速な符号化を目指した．NOLI 符号化方式の欠点である逐次的な処理を，いかに並列化するか検討を行った．

そして，2 画素並列処理することによって，逐次処理と比べ，同セグメントの場合で 20 段の短縮に成功した．これは逐次処理に比べると，処理時間を 60%短縮したことに相当する．別セグメントの場合で 9 段の短縮に成功し，これは 37%の処理時間短縮に相当する．

また今後は，投機実行の採用を検討していくことで，更に高速化を行う．現段階では，2 画素並列処理より，5 段程度の短縮が可能ではないかと推測している．

謝辞

情報システム工学科の島村和典教授にはさまざまな御指導を頂き，誠にありがどうございました．また，本研究発表の副査をしてくださる，岩田誠教授に深く感謝いたします．また，本研究を行うにあたって，お忙しい中，御助力を頂き，副査もお願いした通信・放送機構高知トラフィックセンターの高松希匠研究員には深く感謝いたします，ご迷惑をお掛けしたこともあると思います．ここで御詫びを申し上げます．

また，日頃から親しくしてくださり，いろいろ助言してくださった，本学院生の小林寛征さんに深く感謝いたします．

参考文献

[1] Fei Li,Jason Nieh,“ Optimal Linear Interpolation Coding for Server-based Computing ”,IEEE International Conference on Communication 2002,G05-3,New York,2002

[2] H.Terada, S.Miyata, M.Iwata,“ DDMP's: Self-Timed Super-Pipelined Data-Driven Mulimedia Processors ” , Proceedings of THE IEEE, vol 87, no.2, February,1999

[3] 三菱電機株式会社 情報技術総合研究所, “ MPEG 符号化技術 ” , http://www.eizojoho.co.jp/i/i_pdf/9811shinkijiku.pdf

[4] 小林寛征，高松希匠，島村和典, “ データ駆動型プロセッサによるパケットアセンブリの実装と評価 ”，信学技報, CS2002-138, IE2002-126(2002-12)

[5] シャープ株式会社 IC 事業部 ネットワークシステム LSI 開発センター DDMP 開発室, “ 32bit DDMP 命令の動作説明書 ” 2002 年 4 月 17 日

[6] 谷口慶治, “ 画像処理工学 ” , 共立出版，1996 年 11 月

付録 A

ソースコード

A.1　2 画素並列処理のプログラム

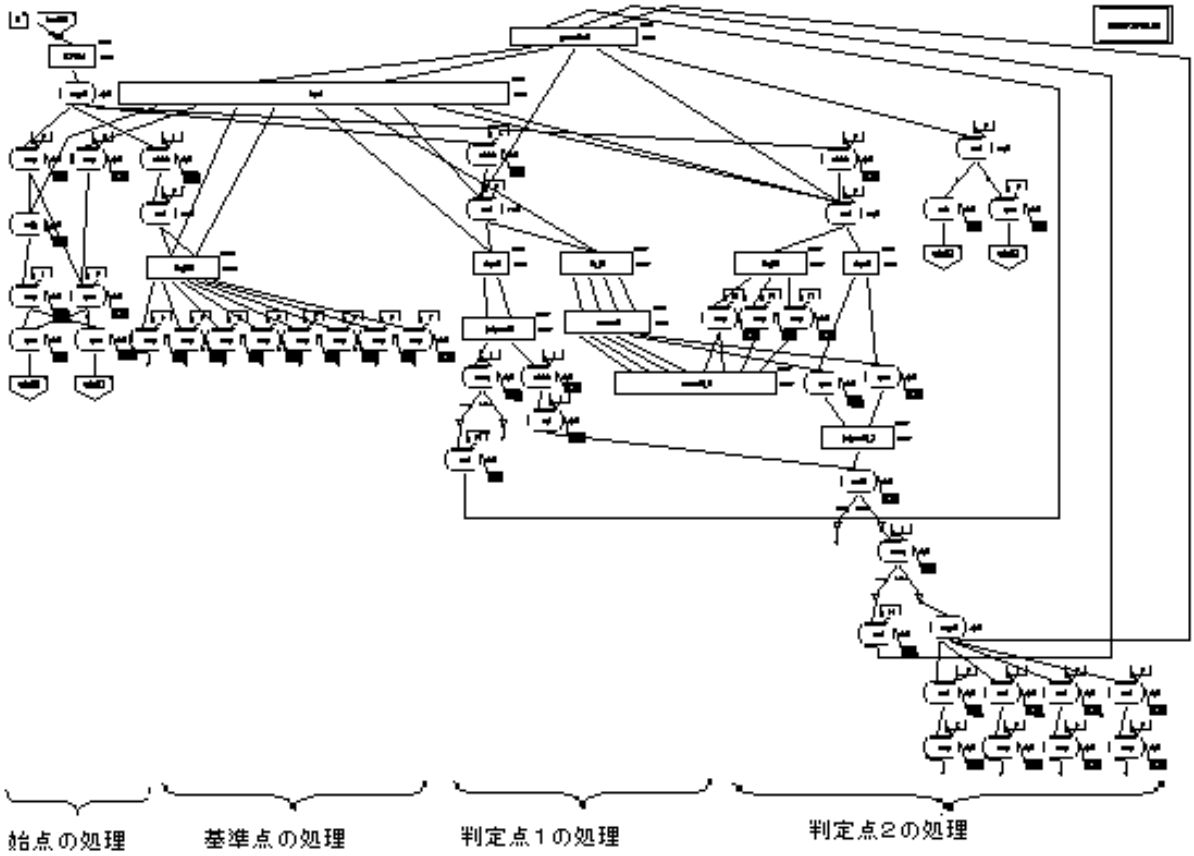

図 A.1　全体のプログラム図

A.1.1　プログラムで使用したメモリ

メモリ	パラメータの内容	メモリ	パラメータの内容
0	始点の輝度	8	Su の分母 (temp)
1	始点の座標	9	Sl の分母 (temp)
2	Su の分子	10	Su2 の分子
3	Sl の分子	11	Sl2 の分子
4	Su の分母	12	Su2,Sl2 の分母
5	Sl の分母	13	判定点の 1 つ前の画素
6	Su の分子 (temp)	14	判定点 1 の 1 つ前の画素
7	Sl の分子 (temp)	15	判定点 2 の 1 つ前の画素

表 A.1　メモリ番号とそのパラメータ

A.1.2　プログラムで使用するモジュール

モジュール名	内容
SDRAM	画素データを SDRAM に格納し， 先頭の輝度値をトリガーとして出力する
Su_Sl2	閾値 T，始点，基準点から，Su，Sl の分子と分母を求める
Su_Sl	閾値 T，始点，判定点から，Su，Sl の分子と分母を求める
slope2	始点と判定点を結ぶ直線の傾きを求める
judgment2	判定点 1 が同セグメントと判断された場合は 0, 別セグメントの場合は 1 を出力する
renewal2	Su，Sl と Su_Sl で求められた Su1，Sl1 を比較し， 更新した値を出力する，と同時に judgment2_2 に更新が 終了したことを知らせる信号を出す.
judgment2_2	判定点 2 が更新された Su,Sl 内にあるかどうか判定する
renewal2_2	判定点 1 と判定点 2 で，それぞれ求められた，Su1，Sl1 と Su2，Sl2 を比較し更新する
generation2	終点判定を行うと共に，世代番号を操作する
input	一度 SDRAM から命令 ssel でデータを読み出すと，読み出すことができない. そこでメモリから読み出す場合と SDRAM から読み出す場合を区別している

表 A.2　モジュール説明

A.1.3　部分的プログラム

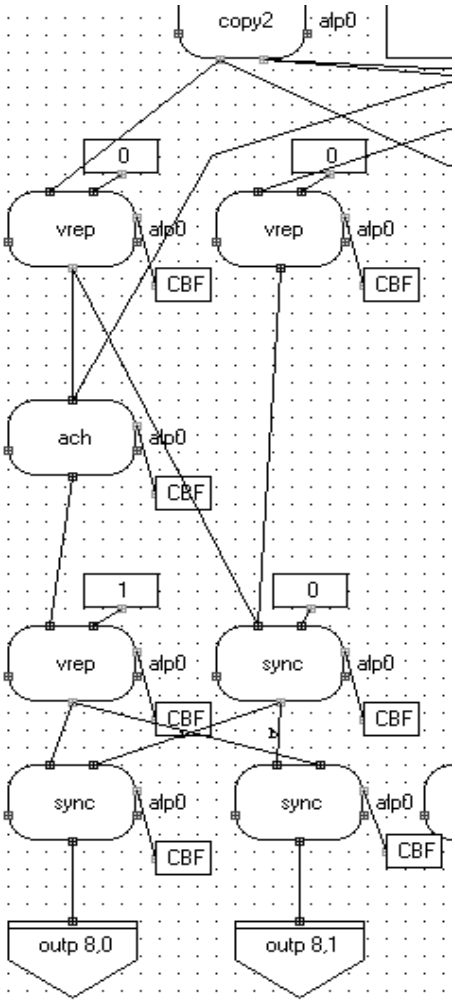

図 A.2　画素情報 (i,p) 出力

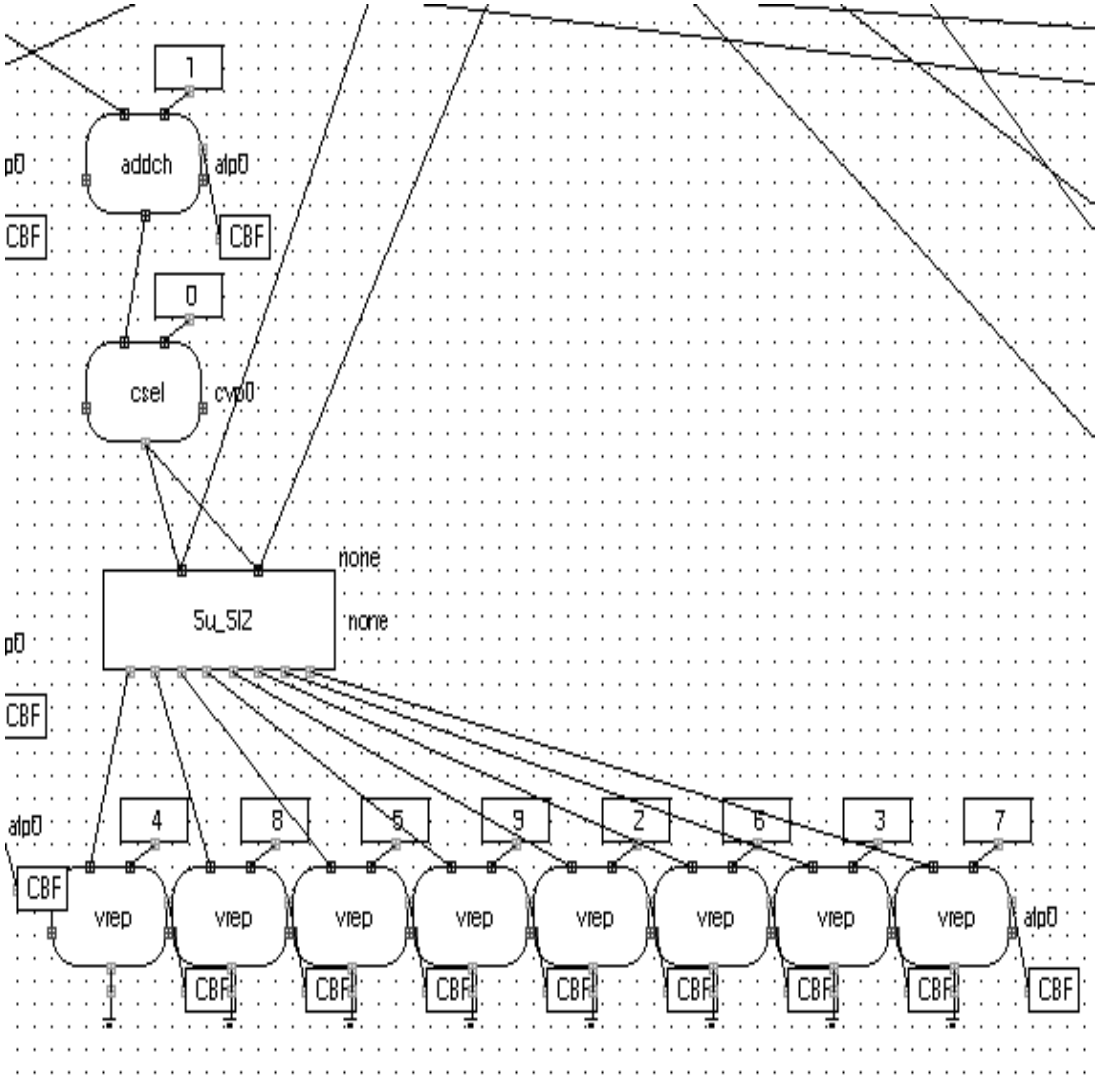

図 A.3　Su,Sl の計算と保存

A.1　2 画素並列処理のプログラム

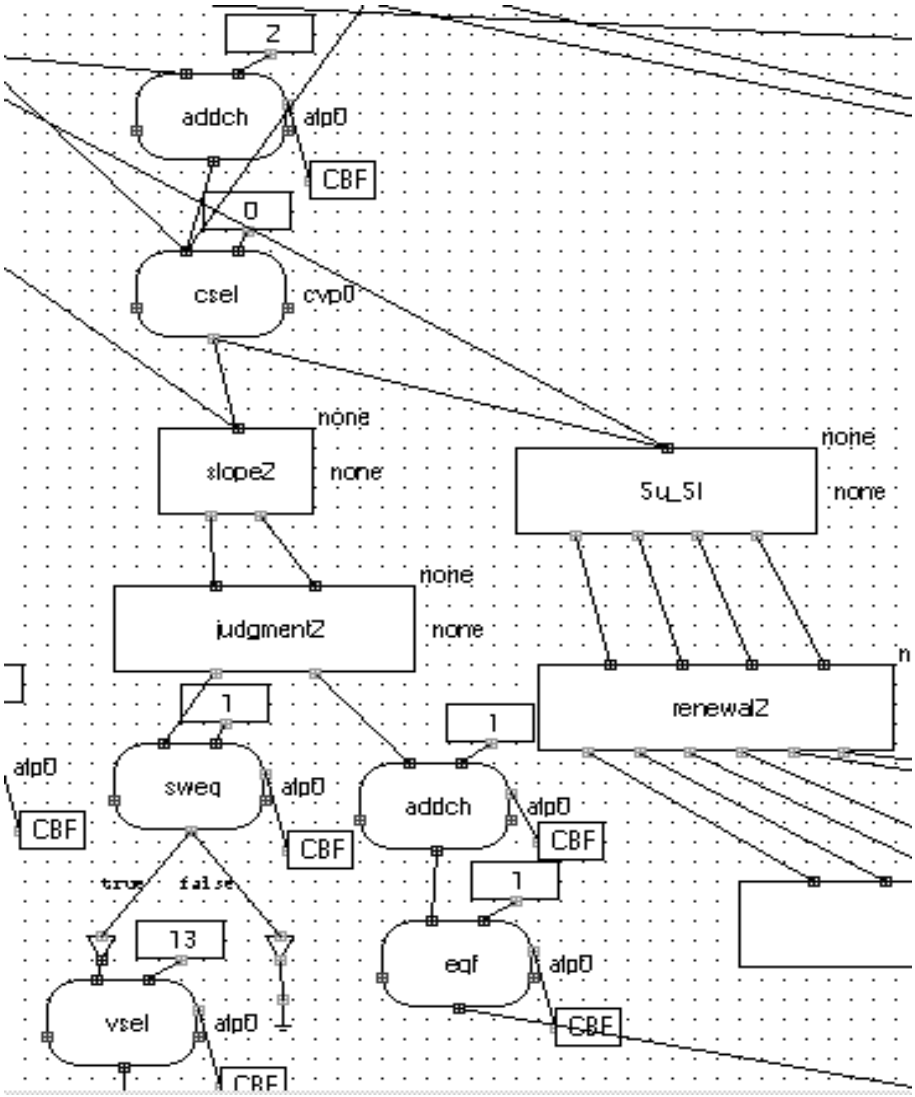

図 A.4　判定点 1 での処理

A.1　2 画素並列処理のプログラム

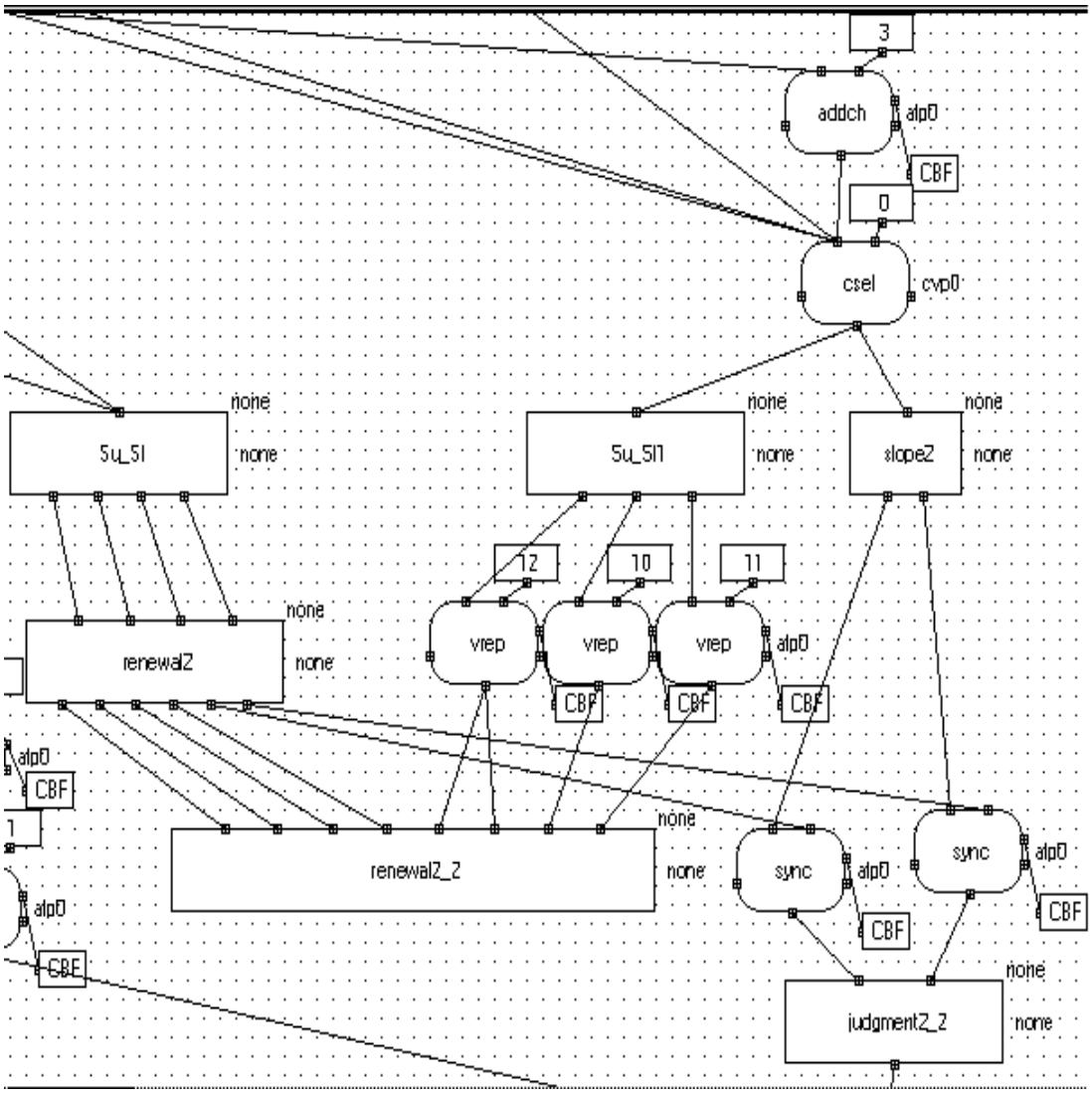

図 A.5　判定点 2 の判定 2 までの処理

A.1　2 画素並列処理のプログラム

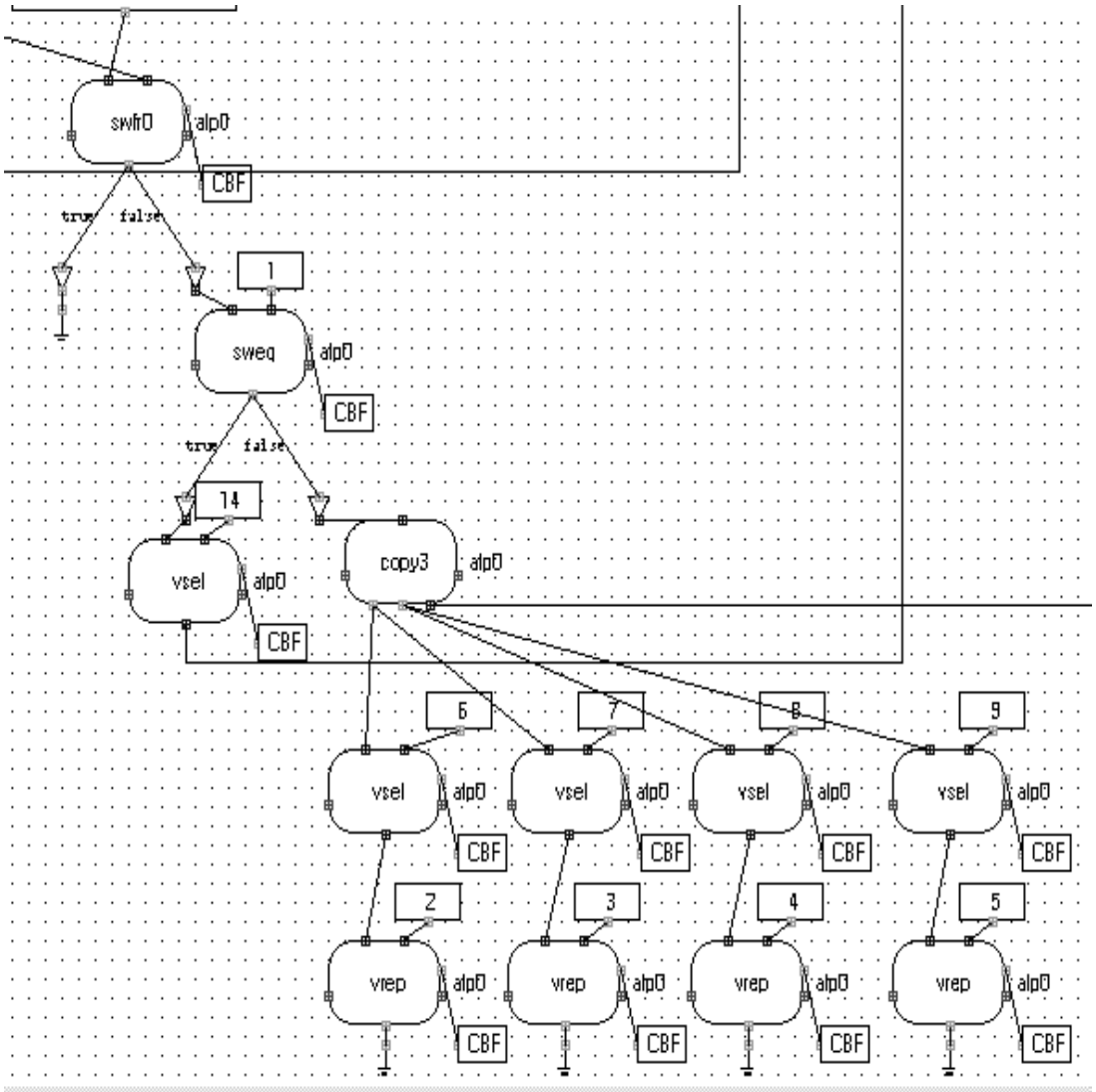

図 A.6　判定点 2 の判定 2 までの処理