

再帰的演算に基づくステレオマッチングVLSIプロセッサの アーキテクチャ

三浦 清志[†] 張山 昌論[†] 亀山 充隆[†]

[†] 東北大学大学院情報科学研究科 〒980-8579 宮城県仙台市青葉区荒巻字青葉 05

E-mail: †{miura,hariyama,michi}@kameyama.ecei.tohoku.ac.jp

あらまし 3次元計測のためのステレオビジョンでは、画像間の対応を高速に求めることが重要となる。対応点探索の手法として、SAD(Sum of Absolute Differences) 演算に基づく方法がある。本稿では、SAD 演算の中間結果を再利用することにより計算量を最小化するアルゴリズムを提案する。さらに、相互結合網の複雑さを最小化するアロケーションに基づくアーキテクチャを提案する。0.18 μ m CMOS 設計ルールを用いて設計した場合、汎用プロセッサを用いた処理と比較して5000倍以上の高速化が達成できることを明らかにする。

キーワード 3次元計測, ステレオビジョン, Sum of Absolute Differences(SAD), スケジューリング, アロケーション

Architecture of a Recursive-Computation-Based Stereo Matching VLSI Processor

Kiyoshi MIURA[†], Masanori HARIYAMA[†], and Michitake KAMEYAMA[†]

[†] Graduate School of Information Sciences, Tohoku University Aoba 05, Aramaki, Aoba-ku, Sendai-shi,
Miyagi, 980-8579 Japan

E-mail: †{miura,hariyama,michi}@kameyama.ecei.tohoku.ac.jp

Abstract High-speed correspondence matching is essential for stereo vision. A typical method for correspondence matching is one based on SAD(Sum of Absolute Differences) computation. This paper presents a stereo matching algorithm that minimizes the computational amount by re-using the intermediate results of SADs. A VLSI processor architecture is also proposed based on an optimal allocation that minimizes the complexity of the interconnection network. As a result, the performance of the VLSI processor becomes 5000 times faster than that of a general-purpose processor.

Key words Three-Dimensional Instrumentation, Stereo Vision, Sum of Absolute Differences(SAD), Scheduling, Allocation.

1. ま え が き

知能ロボットや、高安全自動車などのリアルワールド応用システムを実現するためには、高速かつ信頼性の高い3次元情報の取得が重要となる[1]。3次元計測の典型的な方法として、2台のカメラを用いて取得された画像間の対応点探索に基づくステレオビジョンがある。対応点探索の手法として、参照画像中の参照ウィンドウと候補画像中の候補ウィンドウとのSAD(Sum of Absolute Differences) 演算に基づく方法がよく知られている。SAD 演算に基づく方法では、SAD 演算の計算量が膨大となる。

本稿では、SAD 間の中間結果を利用することにより計算量を最小化するステレオマッチングアルゴリズムを提案する。本

アルゴリズムは、参照ウィンドウの重なりが最大となるようなSAD 演算間で中間結果を利用することにより、SAD 演算間で共通なAD(Absolute Difference) の数を最大にするという考え方に基づいている。本アルゴリズムを用いることにより、画像サイズ512×512、ウィンドウサイズ15×15の場合、計算量を1.2%以下に減少できる。

また、最小の処理時間(ASAP スケジューリング)の制約下で、演算器-演算器間ならびにメモリ-演算器間の相互結合網の複雑さを最小化する最適アロケーションに基づくアーキテクチャを提案する。0.18 μ m CMOS 設計ルールを用いて設計した場合、汎用プロセッサを用いた処理と比較して5000倍以上の高速化が達成できる。

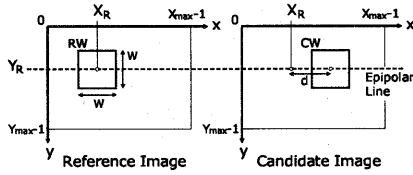


図 1 SAD 演算に基づくステレオマッチング

Fig. 1 SAD-based stereo matching.

2. ステレオマッチングアルゴリズム

2.1 基本的なステレオマッチングアルゴリズム

ステレオビジョンは、図 1 に示すように、2 枚の画像間の対応付けを行った後に、三角測量を行うことにより、物体の 3 次元座標を求める処理である。対応点探索の典型的な手法として、SAD 演算を用いた方法がよく用いられる。SAD 演算に基づく対応点探索では、参照画像上の任意の参照点 (X_R, Y_R) の対応点は以下のように求められる。図 1 に示すように、参照点 (X_R, Y_R) を中心とするサイズ $W \times W$ の参照ウィンドウを設定する。2 台のカメラの光軸を並行にする典型的な配置を用いると、対応点は候補画像のエピポーラライン $Y = Y_R$ 上に存在することが保証される。そこで、候補画像上に、エピポーラライン上の候補点 $(X_R + d, Y_R)$ を中心とするサイズ $W \times W$ の候補ウィンドウを設定する。参照点と候補点の X 座標の差 d を視差と呼ぶ。このとき、参照点と候補点に対する SAD 演算は次式で与えられる。

$$SAD(X_R, Y_R, d) = \sum_{j=-\frac{W-1}{2}}^{\frac{W-1}{2}} \sum_{i=-\frac{W-1}{2}}^{\frac{W-1}{2}} AD(X_R + i, Y_R + j, d) \quad (1)$$

ここで

$$AD(X_R + i, Y_R + j, d) = |I_R(X_R + i, Y_R + j) - I_C(X_R + i + d, Y_R + j)| \quad (2)$$

は参照画像上の画素 $(X_R + i, Y_R + j)$ と候補画像上の画素 $(X_R + i + d, Y_R + j)$ に対する絶対値差分 (AD) であり、 I_R と I_C はそれぞれ、参照画像と候補画像の輝度値である。SAD が小さいほどウィンドウ内の画像が類似していることを意味する。そこで、参照点とエピポーラライン上の全ての候補点に対して SAD を計算し、SAD 値が最小となる点を対応点と決定することにする。対応点探索においては、SAD 演算を膨大な数のウィンドウに対して行う必要がある。画像サイズが $X_{max} \times Y_{max}$ の場合、参照ウィンドウ数は $X_{max} \times Y_{max}$ となる。1 個の参照点の対応点を求めるためには、エピポーラライン上の X_{max} 個の候補ウィンドウに対して行う必要がある。し

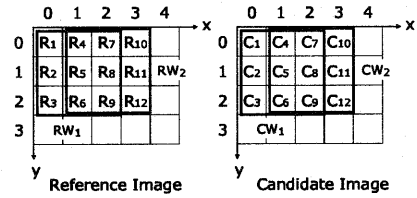


図 2 共通の中間結果を有する SAD 演算

Fig. 2 SAD operation with a common intermediate results.

たがって、参照画像の全ての点に対して対応点を求めるためには、 $X_{max} Y_{max}$ 回の SAD 演算が必要となる。典型的な画像サイズとして $X_{max} = Y_{max} = 512$ とすると、 512^3 回の SAD 演算が必要となる。また、1 回の SAD 演算でも式 (1) からわかるように W^2 回の AD 演算、 $W^2 - 1$ 回の加算が必要とされる (通常ウィンドウサイズは $W = 15$ 程度に設定される)。このように、対応点探索は計算量が多い処理であるため、その高速化が望まれている。

2.2 共通の中間結果に着目した計算量減少

SAD 演算間での共通な中間結果を利用することにより、計算量を減少することを考える。図 2 に、 3×3 のウィンドウサイズの例を示す。参照点 $(1, 1)$ を中心とする参照ウィンドウ RW_1 と、候補点 $(1, 1)$ を中心とする候補ウィンドウ CW_1 に対する SAD 演算 SAD_1 は、式 (1) を用いて、次式のように与えられる。

$$SAD_1 = |R_1 - C_1| + |R_2 - C_2| + |R_3 - C_3| + |R_4 - C_4| + |R_5 - C_5| + |R_6 - C_6| + |R_7 - C_7| + |R_8 - C_8| + |R_9 - C_9| \quad (3)$$

同様に、参照点 $(2, 1)$ を中心とする参照ウィンドウ RW_2 と、候補点 $(2, 1)$ を中心とする候補ウィンドウ CW_2 に対する SAD 演算 SAD_2 は、次式で与えられる。

$$SAD_2 = |R_4 - C_4| + |R_5 - C_5| + |R_6 - C_6| + |R_7 - C_7| + |R_8 - C_8| + |R_9 - C_9| + |R_{10} - C_{10}| + |R_{11} - C_{11}| + |R_{12} - C_{12}| \quad (4)$$

式 (3), (4) の左辺には、共通項 $\sum_{i=4}^9 (R_i - C_i)$ が存在する。したがって、 SAD_1 の中間結果 $\sum_{i=4}^9 (R_i - C_i)$ を用いることにより、3 回の AD 演算と 3 回の加算で SAD_2 が計算可能であり、計算量を減少できる。

参照点 (X_1, Y_1) 、候補点 $(X_1 + d_1, Y_1)$ に対する SAD 演算 $SAD(X_1, Y_1, d_1)$ と、参照点 (X_2, Y_2) 、候補点 $(X_2 + d_2, Y_2)$ に対する SAD 演算 $SAD(X_2, Y_2, d_2)$ が共通の中間結果を有する条件は以下のように与えられる。

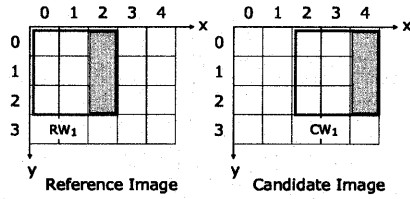


図 3 $SLICE(1, 1, 2)$ ($W = 3$)
Fig. 3 $SLICE(1, 1, 2)$ ($W = 3$).

条件 1: 参照ウィンドウ間に重なりがあり, かつ候補ウィンドウ間にも重なりがある. この条件は次式で表される.

$$\begin{aligned} |X_1 - X_2| &< W \\ |Y_1 - Y_2| &< W \\ |(X_1 + d_1) - (X_2 + d_2)| &< W \end{aligned} \quad (5)$$

条件 2: 各 SAD 演算の視差が等しい. すなわち, $d_1 = d_2$.

条件 1 が必要条件であることは明らかである. また, ウィンドウ間に重複がある (条件 1 が満たされる) 場合でも, 視差が等しくない (条件 2 が満たされない) 場合には, AD 演算への入力の組み合わせが異なるため共通の中間結果が存在しない.

2.3 計算量最小化を指向した VLSI 向きステレオマッチング アルゴリズム

視差 d を有する SAD の集合を S_d とすると, S_d は次式で与えられる.

$$S_d = \{SAD(X, Y, d) | 0 \leq X < X_{max}, 0 \leq Y < Y_{max}\} \quad (6)$$

2.2 節の議論より, 視差が異なる SAD 間には共通の中間結果がない, すなわち, データ依存関係がない. そのため, 視差が異なる SAD の集合 $S_0, S_1, \dots, S_{X_{max}-1}$ 間には並列性が存在する.

以下では, 共通の中間結果を有する, すなわちデータ依存関係を有する SAD の集合 S_d に対して, 計算量最小化を指向したアルゴリズムを説明する.

$SAD(X, Y, d)$ を計算するための中間結果として, 参照ウィンドウと候補ウィンドウの右端の列の画素に対する部分的な SAD (以下では $SLICE$ と呼ぶ) を定義する. $SLICE(X, Y, d)$ は次式で与えられる.

$$SLICE(X, Y, d) = \sum_{j=-\frac{W-1}{2}}^{\frac{W-1}{2}} AD(X + \frac{W-1}{2}, Y + j, d) \quad (7)$$

図 3 は, ウィンドウサイズ $W = 3$, 視差 $d = 2$ の場合の例である. $SLICE(1, 1, 2)$ の計算に用いられる画素を灰色で示している.

はじめに, 図 4 の例を用いて, SAD の再帰的計算の基本的な

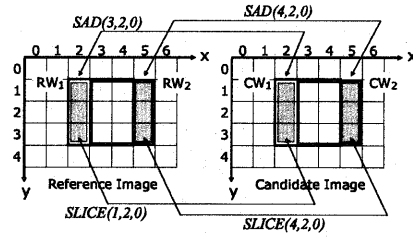


図 4 SAD の再帰的計算 ($W = 3$)
Fig. 4 Recursive Computation of an SAD.

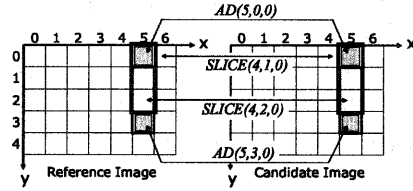


図 5 $SLICE$ の再帰的計算 ($W = 3$)
Fig. 5 Recursive Computation of a SLICE.

手順を説明する. 参照点 $(3, 2)$, $(4, 2)$ を中心とする参照ウィンドウを, それぞれ RW_1 , RW_2 とする. また, 候補点 $(3, 2)$, $(4, 2)$ を中心とする候補ウィンドウをそれぞれ CW_1 , CW_2 とする. ウィンドウ RW_1 と CW_1 に対して $SAD(3, 2, 0)$ が計算されるとする. また, ウィンドウ RW_2 と CW_2 に対して $SAD(4, 2, 0)$ が計算されるとする. $SAD(4, 2, 0)$ を計算する際に, $SAD(4, 2, 0)$ と共通の中間結果を有する SAD は複数存在するが, 計算量を最小化するためにウィンドウの重なりが最大になる (X 座標が 1 だけ異なる) ような $SAD(3, 2, 0)$ の結果を用いる. これは, ウィンドウの重なりを最大にすることにより, 共通な AD の数を最大にできるためである. $SAD(3, 2, 0)$ と $SAD(4, 2, 0)$ の共通の中間結果は, $SAD(3, 2, 0) - SLICE(1, 2, 0)$ により計算できる. また, $SAD(4, 2, 0)$ に存在し, $SAD(3, 2, 0)$ に存在しない部分画素は $SLICE(4, 2, 0)$ で与えられる. 結果として, $SAD(4, 2, 0)$ は次式で与えられる.

$$\begin{aligned} SAD(4, 2, 0) &= SAD(3, 2, 0) \\ &\quad - SLICE(1, 2, 0) + SLICE(4, 2, 0) \end{aligned} \quad (8)$$

同様に, $SLICE(4, 2, 0)$ の計算においても, 計算量を最小とするためには画素の重なりが最大となる $SLICE(4, 1, 0)$ の結果を用いればよい. したがって, $SLICE(4, 2, 0)$ は次式で与えられる.

$$\begin{aligned} SLICE(4, 2, 0) &= SLICE(4, 1, 0) \\ &\quad - AD(5, 0, 0) + AD(5, 3, 0) \end{aligned} \quad (9)$$

式 (8), (9) より, $SAD(4, 2, 0)$ は次式で計算できる.

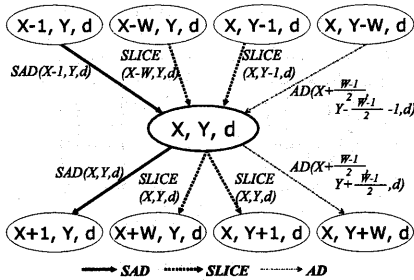


図6 再帰的SAD計算のDFGのノード

Fig. 6 Node of a DFG of the recursive SAD computation.

$$\begin{aligned}
 SAD(4, 2, 0) &= SAD(3, 2, 0) \\
 &\quad - SLICE(1, 2, 0) + SLICE(4, 1, 0) \\
 &\quad - AD(5, 0, 0) + AD(5, 3, 0)
 \end{aligned} \quad (10)$$

式(10)は、もし、 $SAD(3, 2, 0)$ 、 $SLICE(1, 2, 0)$ 、 $SLICE(4, 1, 0)$ 、 $AD(5, 0, 0)$ が計算済みであれば、1回のSAD計算がAD計算1回($AD(5, 3, 0)$)と4回の加算(または減算)で行えることを示している。式(10)を一般化した式を、式(11)に示す。

$$\begin{aligned}
 SAD(X, Y, d) &= SAD(X-1, Y, d) \\
 &\quad - SLICE(X-W, Y, d) + SLICE(X, Y-1, d) \\
 &\quad - AD(X + \frac{W-1}{2}, Y - \frac{W-1}{2} - 1, d) \\
 &\quad + AD(X + \frac{W-1}{2}, Y + \frac{W-1}{2}, d)
 \end{aligned} \quad (11)$$

次に、SADの再帰的計算の全体の手順を示す。並列性、規則性を明確に表すため、データフローグラフ(DFG)によって記述する。 $AD(X + \frac{W-1}{2}, Y + \frac{W-1}{2}, d)$ は、 $SAD(X, Y, d)$ を計算する際にだけ計算されるとする。すなわち、他のSAD演算で $AD(X, Y, d)$ を用いる場合には、新たに計算するのではなく、 $SAD(X - \frac{W-1}{2}, Y - \frac{W-1}{2}, d)$ の中間結果を用いる。 $SLICE(X, Y, d)$ も同様に、 $SAD(X, Y, d)$ を計算する際にだけ計算されるという条件を設定する。

図6に、再帰的SAD計算のDFGの基本となる式(11)のDFG表現を示す。各ノード (X, Y, d) はSAD演算 $SAD(X, Y, d)$ を表す。また、ノード間の有効辺はデータの依存関係を表す。有効辺の種類(実線、鎖線、一点鎖線)はそれぞれ、中間結果の種類(SAD, SLICE, AD)に対応する。例えば、ノード $(X-W, Y, d)$ から (X, Y, d) への有効辺(鎖線)は、 $SAD(X-W, Y, d)$ の中間結果である $SLICE(X-W, Y, d)$ が、 $SAD(X, Y, d)$ の入力として使用されることを意味する。図6に示す依存関係を全てのSAD演算に対して網羅すれば、再帰的SAD計算のDFGを求められる。図7に、ウィンドウサイズ $W=3$ の場合の、 $S_d = \{SAD(X, Y, d) | 0 \leq X < X_{max}, 0 \leq Y < Y_{max}\}$ に対する再帰的SAD計算のDFGを示す。

提案手法と、従来の手法の計算量の比較を表1に示す。提案手法と従来の手法において、SAD演算回数は同じである。1回のSAD演算について見てみると、提案手法では1回のSAD

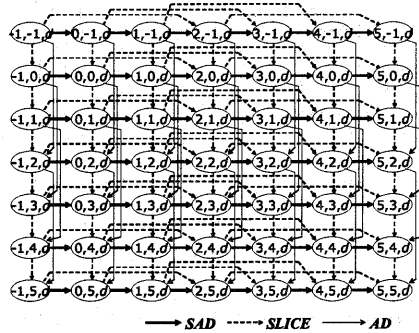


図7 再帰的SAD計算のDFG ($W=3$, $X_{max}=Y_{max}=7$ の例)

Fig. 7 DFG of the recursive SAD computation.

	Conventional	Proposed
Number of SADs	$X_{max}^2 Y_{max}$	$X_{max}^2 Y_{max}$
Number of ADs per SAD	W^2	1
Number of additions per SAD	$W^2 - 1$	4

表1 ステレオマッチングの計算量の評価

Table 1 Evaluation of computational amount.

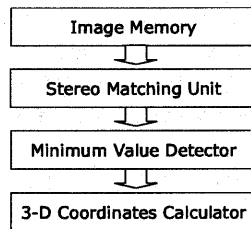


図8 ステレオビジョンVLSIプロセッサ

Fig. 8 Stereo Vision VLSI Processor.

演算をAD演算1回と4回の加算(または減算)で行える。一方、従来の手法では、2.1節で述べたように、1回のSAD演算に W^2 回のAD演算と、 W^2-1 回の加算が必要となる。典型的なウィンドウサイズとして、 $W=15$ とすると、提案手法の計算量は従来の方法に比べ、AD演算回数において0.44%、加算回数において1.8%となる。AD演算は減算と排他的論理和により行われるため、AD演算1回を加算1回と同等と見なすと、 $W=15$ の場合に計算量は従来の1.2%以下となる。

3. VLSIプロセッサアーキテクチャ

図8は、ステレオビジョンVLSIプロセッサのブロック図である。画像メモリには参照画像と候補画像のデータが格納され、ステレオマッチング部へ画像データが供給される。ステレオマッチング部では、全ての参照ウィンドウと候補ウィンドウの組み合わせに対してSADが計算される。

図9に示すように、ステレオマッチング部は N 個のSAD演算部(SADU)から構成される。各SADUには、同数のSADの集合 $S_d(d=0, 1, 2, \dots, X_{max}-1)$ が割り当てられる。 $S_0, S_1, S_2, \dots$ 間ではデータ依存関係がないため、SADU間では通信のない並列処理が可能である。1個のSADUは、割り当てられた複

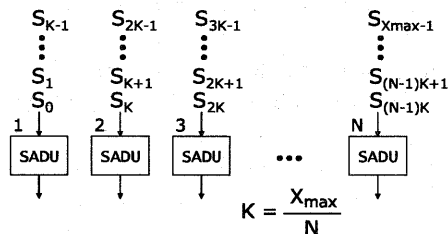


図 9 ステレオマッチング部
Fig. 9 Stereo Matching Unit.

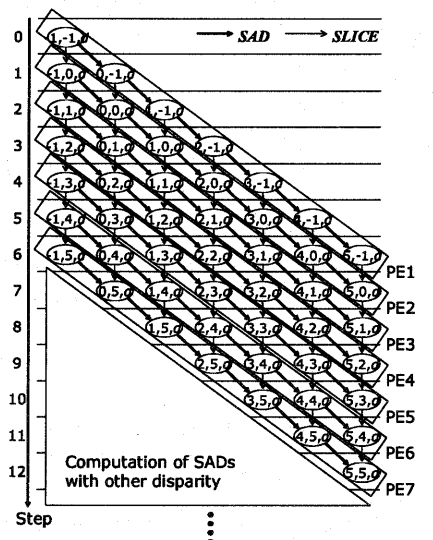


図 10 ASAP スケジューリングおよび PE アロケーション
Fig. 10 ASAP Scheduling and PE Allocation.

数の S_d に対しては直列的に処理を行う。このような構成では、SADU 数に比例して処理時間を減少できる。

SADU の設計においては処理時間最小化を最優先し、その上で演算器-演算器間およびメモリ-演算器間の相互結合網の複雑さを最小化する。はじめに、処理時間最小化のため図 7 の DFG に対して ASAP (As Soon As Possible) スケジューリングを行うことを提案する。DFG のノード 1 個あたりの処理時間を単位ステップとすると、ASAP スケジューリング結果は図 10 で示される。簡単のため、図 10 ではデータフローを示す有効辺の一部を省略している。再帰的 SAD 演算の DFG においては、ノード間の依存関係により、どの 1 個のノードの処理ステップを遅らせても全体の処理ステップ数は増加する。したがって、処理ステップ数最小となるスケジューリングは ASAP スケジューリングだけとなる。

次に、ASAP スケジューリングにより与えられる処理ステップ数制約の下で、相互結合網の複雑さを最小化する最適アロケーションを提案する。相互結合網の複雑さを、全マルチプレクサ (MUX) の入力総数によって評価する。図 11 に、SADU のアーキテクチャモデルを示す。DFG のノード 1 個を処理するモ

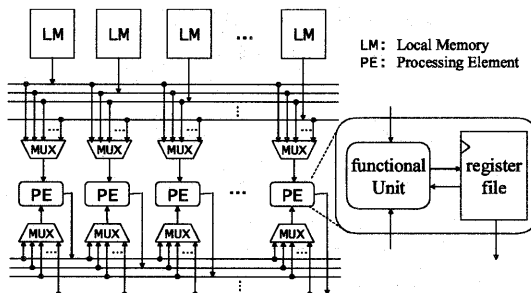


図 11 SADU のアーキテクチャモデル
Fig. 11 Architecture Model.

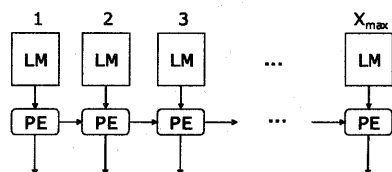


図 12 最適アロケーションに基づく SADU の構成
Fig. 12 SADU Based on the Optimal Allocation.

ジュールを PE (Processing Element) と呼ぶ。各 PE は、ノード 1 個を 1 ステップ以内に処理すると仮定する。スケジューリング結果から、処理ステップ数制約を満足するためには少なくとも X_{max} 個の PE が必要となる。そのため SADU は、相互結合網に接続された X_{max} 個の PE をもつ。また、各ノードにはそれぞれ画像データが入力されるため、PE と同数のローカルメモリ (LM) を用意する。

提案の最適アロケーションは、PE-PE 間相互結合網と LM-PE 間相互結合網の両方の MUX 入力総数を最小化することができる。図 12 は、本アロケーションにより得られる SADU の構成である。本アロケーションは、PE アロケーションおよびメモリアロケーションからなる。提案する PE アロケーションを図 10 に示す。この PE アロケーションでは、PE 間で転送されるデータの語長をできるだけ小さくするために、語長が最も大きい SAD の結果を同一 PE で再利用するようにしている [2]。1 個の PE は、どのステップにおいても特定の 1 個の PE だけに中間結果を転送する。したがって、図 12 に示されるように各 PE はリニアアレイ状に接続され、PE-PE 間相互結合網の MUX 入力総数は $X_{max} - 1$ となる。本 PE アロケーションには、同じ y 座標を有する画素だけが各 PE へ入力されるという性質がある。この性質に着目し、同じ y 座標を有する画素を同一の LM に記憶するメモリアロケーションを提案する。その結果、図 12 に示すように LM と PE は 1 対 1 に接続され、LM-PE 間相互結合網の MUX 入力総数は X_{max} となる。

本アロケーションの最適性を証明する。はじめに、PE アロケーションに基づく PE-PE 間相互結合網の MUX 入力総数 $X_{max} - 1$ が最小であることを示す。ASAP スケジューリングの処理ステップ数制約により、PE 数は X_{max} 個である。ASAP スケジューリングでは、他ノードからの入力が必要としないノー

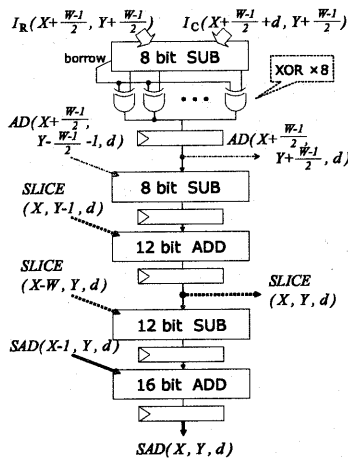


図 13 PE 内の演算ユニットの構成
Fig.13 Functional unit.

ドは必ず 1 ステップ目に割り当てられる。本稿で扱う ASAP スケジューリングに基づく SDFG では、他のノードの出力を必要としないノード（独立なノード）は 1 ステップ目に割り当てられ、その数は 1 となる（図 10）。他の PE の中間結果を必要としない PE（独立な PE）に対しては独立なノードが割り当てられる必要があるため、独立な PE の数は最大 1 となる。SADU 内の PE 数が X_{max} であることから、独立な PE 以外の PE の数は少なくとも $X_{max} - 1$ となる。これらの PE は他の PE の中間結果を必要とするため、各 PE の MUX の入力数は少なくとも 1 となる。したがって、SADU 内の PE-PE 間相互結合網の MUX の入力総数の下限は $X_{max} - 1$ となり、提案の PE アロケーションの最適性が証明される。次に、メモリアロケーションに基づく LM-PE 間相互結合網の MUX 入力総数 X_{max} が最小であることを示す。PE 数が X_{max} であることと各 PE は少なくとも 1 個の LM からの入力が必要であることから、LM-PE 間における MUX 入力総数の下限が X_{max} となることは明らかである。したがって、提案のメモリアロケーションの最適性が証明される。

図 13 は、PE 内の演算ユニット（FU）の構成である。FU では、式 (11) に従って 1 回の AD 演算と計 4 回の加減算から 1 個の SAD を計算する。各演算器間にはレジスタを挿入し、演算器レベルでパイプラインを活用する。

以下では、画像サイズが $X_{max} = 512$, $Y_{max} = 512$ であるステレオビジョン VLSI プロセッサの性能見積もりを行う。はじめに、中間結果を格納するレジスタファイルと、画像データを保持するローカルメモリの容量を考える。256 階調のグレースケールでは 1 画素が 8bit となるため、 $W = 15$ における各中間結果の語長は、(AD の語長) = 8bit, (SLICE の語長) = $(8 + \log_2 W)$ bit = 12bit, (SAD の語長) = $(8 + \log_2 W^2)$ bit = 16bit となる。PE1 個あたりのレジスタファイルの容量は、図 10 に示すスケジューラされた DFG から得られるライフタイムより、SAD が 1word = 16bit, SLICE が W word = 180bit,

AD が W word = 120bit となる。画像サイズより、SADU1 個あたりの PE 数は $512 (= X_{max})$ となる。ローカルメモリの個数は、PE 数と同じく 512 個となる。各ローカルメモリには画像データが重複なく保持されているとすると、参照画像・候補画像の画像サイズは合計で $512 \times 512 \times 8 \times 2$ bit (= 512Kbyte) となり、ローカルメモリ 1 個あたりの容量は 1Kbyte となる。

次に、VLSI プロセッサの処理時間の見積もりを行う。データパスにおけるクリティカルパスは、図 13 より、16bit 加算器であると考えられる。0.18 μ m CMOS 設計ルールを用いることを想定し、16bit 加算器とレジスタ遅延時間の和を 5nsec とする。処理時間は $(S_d$ の処理時間) $\times$ (1 個の SADU が処理する視差 d の数) $\times$ 5nsec。ここで、図 10 において、 S_d の実行開始ステップは 0, 終了ステップは 12 であるが、各 PE ごとに見るとそれぞれ開始ステップ・終了ステップが異なる。そこで、図 10 では、次の入力である $d' \neq d$ に対する S_d の開始ステップを 7 とすることで、全体の処理終了を待たずに次の DFG の処理を開始することができる。このことを考慮すると、SADU が 1 個の場合の処理時間は、 $512 \times 512 \times 5$ nsec = 1.3msec となる。汎用プロセッサ (AthlonXP 1800+) では処理時間が約 7.6sec となるため、VLSI 化により 5000 倍の高速化が見込まれる。

4. むすび

SAD 演算の共通な中間結果に着目し、計算量の最小化を指向した VLSI 向きステレオマッチングアルゴリズムを提案した。また、最適アロケーションに基づくプロセッサアーキテクチャを示し、PE-PE 間およびメモリ-PE 間相互結合網の複雑さを最小化した。提案のプロセッサは、ローカルメモリと PE のペアがリニアアレイ構造に接続される単純で規則的な構成であるため、FPGA のようなプログラマブルデバイスによっても容易に実現できると考えられる。

文 献

- [1] 亀山充隆, 藤岡与周, “ロボット用 VLSI プロセッサシステム,” 日本ロボット学会誌, Vol.14, No.1, pp.22-25, 1996.
- [2] 三浦清志, 張山昌論, 亀山充隆, “再帰的計算に基づくステレオマッチングと VLSI 化” 電子情報通信学会論文誌, Vol.J86-C, No.8, pp.752-759, 2003.
- [3] T. Kanade: Development of a Video Rate Stereo Machine. In Proc. of the 1994 ARPA Image Understanding Workshop, pp.14-16, Nov 1994.
- [4] 張山昌論, 李昇桓, 亀山充隆, “転送ボトルネックのないセンサ・メモリアーキテクチャに基づくモーションステレオ VLSI プロセッサの構成” 電気学会センサマイクロ部門専門誌, vol.20, no.5, pp.237-244, 2000.