ボリュームレンダリング専用並列計算機 ReVolver/C40 の シェーディングステージの設計

木綿 芳行

内容梗概

ボリュームレンダリングは、医療画像の生成手法として、また科学技術計算の結果の可視化手段として重要視されている。しかし、この手法は膨大な記憶容量と計算量を要するため、当研究室ではボリュームレンダリング専用並列計算機 ReVolver/C40 を開発中である。ReVolver/C40 は画像生成処理を視線生成/ピクセル値計算/シェーディング処理という 3 つの処理に分割し、パイプライン処理を行うことによって高速表示を可能としている。上記の 3 つの処理をするステージをそれぞれ、視線生成ステージ (RCS), ピクセル値計算ステージ (PCS), シェーディングステージ (SS) とする。

本稿では、ReVolver/C40 においてシェーディング処理を行う、シェーディングステージの構成と実装について述べる。

シェーディングとは、光源情報を用いて生成画像の輝度を補正することにより、画像に現実感を持たせる手法であり、ReVolver/C40 では depth gradient shading の手法を採用している。この手法は、画像生成で得られた深さバッファ、すなわちスクリーン上の各ピクセルごとに視点と表面ボクセル間の距離 (深さ) を格納した 2 次元配列を用いて表面の勾配を計算し、ピクセルの輝度を求めるピクセルアプローチの一種である。また、勾配計算の際に X/Y 方向それぞれについて目的のピクセルを中央に含む数ピクセルの区間領域を取り出し、その両端の深さを用いて勾配を求めているため、処理内容が簡素で高速化に適している。このため、種々のボリュームレンダリング専用計算機において採用されている。

シェーディングステージは、大きく分けて三つの処理があり、Pixel Tranceiver (PT)、Shading Unit (SU)、Video Signal Generator (VSG) と呼ぶ。それぞれの処理内容は、ピクセル値計算ステージから受けとったピクセルのデータを Shading Unit に送信/シェーディング処理/ビデオ信号の生成である。このように 3 つに処理を分割することによって、Shading Unit がピクセル値の受け取りや画像出力という繁雑な処理を気にすることなくシェーディング処理を行うことができる。

Shading Unit において、depth gradient shading の手法を用いて、ピクセル値にシェーディング処理を行うが、上述したようにその際に必要なデータを取り出すのには局所性がある。よって、画面を 4×4 の 16 分割し、それぞれを一つの Processing Element(PE) に割り当てることで並列処理ができ、高速化が計れる。しかし、PE 間で通信が生じると処理速度の低下が起きる。よって、Pixel Transceiver がピクセル値計算ステージから受け取ったデータをもとにして Shading Unit 内のどの PE の担当であるかを計算し、法線計算に必要なデータを PE に余分に持たせることで通信を防ぐことができる。それぞれ一つの PE は、受け取るためのバッファReceive Buffer(RB) と画像出力のためのフレームバッファOutput Buffer(OB) を持ち、中心にシェーディングを行う Digital Signal Processor (DSP) で構成される。Receive Buffer は、Pixel Transceiver によってピクセル値などを書き込まれ、DSP によってピクセル値などを読み出される。また、Output Buffer は、DSP によって RGB 値が書き込まれ、Video Signal Generator によって RGB 値を読み出される。だから、Receive Buffer と Output Buffer はそれぞれダブルバッファ構成になっている。また、Video Signal Generator は、Output Buffer に書き込まれたデータを取りだし、D/A Converter に入力するための CRT Controller と Digital 信号を Analog 信号に変換する D/A Converter から構成される。

上記のことを実現するために、Pixel Transceiver には、FPGA(Field Programmable Gate Array)、Receive Buffer は高速 SRAM を用いて 2 バンク構成にし、Output Buffer には画像用マルチポートメモリ (VRAM) を使用し、Video Signal Generator は、FPGA と D/A Converter を使用している。

この構成により、ピクセル値の書き込み/シェーディング処理/画像表示は独立し、フレーム単位のパイプライン動作を行うことが可能となった。

A Design for the Shading Stage of the Parallel Volume Rendering Computer : ReVolver/C40

Yoshiyuki KIWATA

The volume rendering is an important technique to generate medical images and to visualize results obtained by scientific computation. Since however, this method requires an enormous storage and large computational power, we are developing a parallel machine specially for volume rendering: ReVolver/C40. This machine can display images fast with a pipeline processing of three independent processes:casting rays, calculating pixels and shading. Each of them is called *the Ray Casting Stage*, *the Pixel Calculation Stage* and *the Shading Stage*.

This paper describes a design for the Shading Stage of ReVolver/C40. The goal of the Shading Stage is to shade the images generated by the Pixel Calculation Stage.

Shading is a method to correct intensity at each pixel in a generated image with light source information, and to get the realistic image. The Shading Stage computes the intensity with the *depth gradient shading* method. This is one of the *pixel approaches*, which calculates the surface gradient and the intensity of pixels with a two-dimensional array, called a *depth buffer* which contains a *depth*, the distance from an eye point to a surface voxel at each pixel in a screen. The depth gradient shading is simple and fit for high-speed processing because it calculates a gradient with only two depths at each end pixels of a short segment including the target pixel for both x-coordinate and y-coordinate. So, this method is adopted for various machines specified for volume rendering.

The Shading Stage includes three structures: the Pixel Transceiver (PT), the Shading Unit (SU) and the Video Signal Generator (VSG). Their processes are receiving pixel values and transmitting them to the Shading Unit, shading, and outputting images, respectively. By dividing the Shading Stage into three components, the Shading Unit can do shading without complicated processes of receiving pixel and outputting the images.

When the Shading Unit does shading with the depth gradient shading method, it accesses the limited part of the screen. So,the screen is devided into 16 components and one Processing Element is assigned to each of them. It enables parallel and high-speed processing. But a communication among processing elements causes to down the processing speed. So the Pixel Transceiver calculates which processing element receives the data with the data received from the Pixel Calculation Stage. It enables the communication among the processing elements for a processing element to receive extra data for calculating a normal vector. Each processing element has a pixel value receive buffer, the *Receive Buffer*, and a frame buffer for image output, the *Output Buffer*, and has a *Digital Signal Processor* (DSP) as a kernel for the shading process. The Receive Buffer is written the pixel value on by the Pixel Transceiver and is read by a DSP. The Output Buffer is written the *RGB* value on by a DSP and is read by the Video Signal Generator. Both the Receive Buffer and the Output Buffer adopt the double buffer mechanism. The Video Signal Generator has a *CRT controller* and a D/A converter. The CRT controller reads the data written in the Output Buffer and inputs on the D/A converter. The D/A converter convert a digtal signal to an analog signal.

To realize the above things, A *Field Programmable Gate Array* (FPGA) is used as the Video Signal Generator. A Two-bank memory with high-speed SRAM are used as the Receive Buffer. A multi-port memory specified for images (VRAM) is used as the Output Buffer. A FPGA and a D/A converter are used as the Video Signal Generator.

This structure enabled the Shading Stage to get the three processes independent from each other, and to work on pipeline at a frame unit.

1　はじめに

現実の時空間や計算モデルにおけるオブジェクトや自然現象の多くは、3 次元のボリュームデータで与えられる。ここで、ボリュームデータというのは内部情報を伴った 3 次元のデータであるため、ポリゴンや線分といった幾何要素からは、本質的に構成不可能な場合がある。このようなボリュームデータを可視化する手法として、ボリュームレンダリングがある。ここで、ボリュームデータとは、整数格子座標値をもつボリュームの最小空間単位 (ボクセル) から構成される。ボクセルは、いわばピクセルの 3 次元版であり、自身に関連した一つ以上の数値 (色, 不透明度, 密度など) を持っている。ボリュームレンダリングとは、3 次元空間に配置されたデータを直接 2 次元のスクリーン上に投影することにより、3 次元空間を可視化する手法である。すなわち、ボリュームデータの明確な構造は仮定せず、むしろ空気中に漂う粒子からなる半透明のゲル様の形状モデルを可視化のよりどころとしている。ボリューム表現は、一切のデータセットを維持したままなので、レンダリングの工夫しだいでオブジェクトのいかなる部分領域もみることができる。しかし、ボリュームデータセットの記憶量からくる処理時間の長さが問題となるので、ボリュームレンダリング専用並列計算機 ReVolver/C40 ではボリュームデータのメモリ容量を抑えるために、ボクセル値は 1byte のインデックスとして与えられている。色や透明度の実体は属性テーブルに格納されており、このインデックスを用いて属性テーブルを参照後、色と透明度を得る。

また、ボリュームレンダリングというのは、大きくわけて画像生成/シェーディングという 2 つの処理に分けられる。画像生成は、3 次元のボリュームデータを 2 次元に射影することであり、その射影結果に陰影をつけることがシェーディングである。しかし、ボリュームレンダリングでは、表面がなく法線計算ができない。すなわち、ボリュームデータは直接法線を計算できるような幾何情報をもたないので、シェーディングでは仮想的な表面の勾配をボリュームデータから再生する特別な技法が必要となってくる。

ボリュームレンダリング専用並列計算機 ReVolver/C40 では、*depth gradient shading* という技法を採用している。

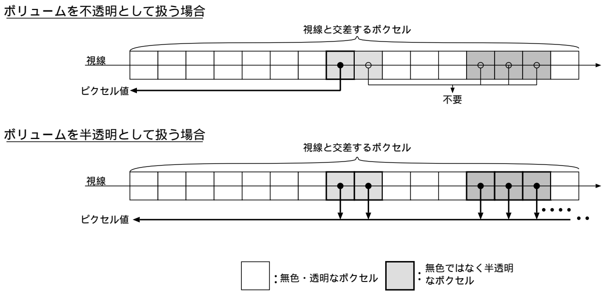


図 1: 参照するボクセルの違い

本稿では、ReVolver/C40 においてシェーディング処理と画像表示を行うシェーディングステージについて述べる。第 2 章において ReVolver/C40 の特徴と構成について述べ、第 3 章では、ReVolver/C40 におけるシェーディングステージの概要について述べ、第 4 章では ReVolver/C40 におけるシェーディングステージの実装について述べる。

2　ReVolver/C40

本節では、ボリュームレンダリング専用並列計算機 ReVolver/C40 の特徴と構成について述べる。

2.1　ReVolver/C40 の特徴

従来のボリュームレンダリングは、主に医療活動の支援として人体の骨格や内蔵などのデータを可視化することを目的としていたため、半透明のボリュームを扱う必要はなかった。ReVolver/C40 では、ボクセル値に色と透明度のデータを持たせることにより、半透明のボリュームを扱う。これは、科学技術計算の結果を可視化するのに有用である。ボリュームを半透明なものとして扱うには、図 1のようにピクセル値を計算する際に視線と交差する無色透明でない全ボクセルを順にアクセスする必要がある。したがって、読み出すボクセル数が増加し、ピクセル値計算の演算が増大するという問題がある。よって、高速表示を行うために ReVolver/C40 が有している特徴を示す。

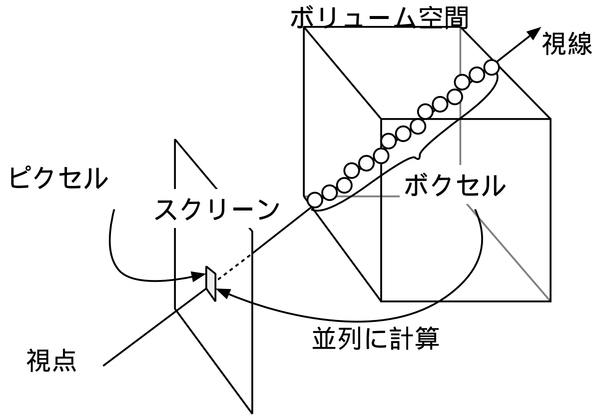


図 2: レイキャスティング法

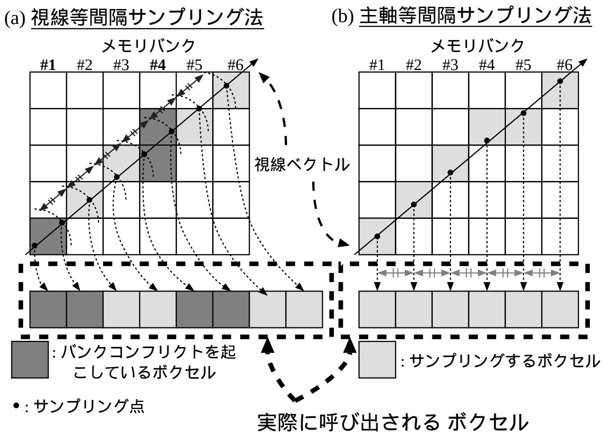

(a) 視線等間隔サンプリング法 (b) 主軸等間隔サンプリング法

図 3: サンプリング方法の違い

レイキャスティング法の採用　視点からスクリーン上のあるピクセルの中心へと向かうベクトルのことを視線ベクトルと呼び、視点を一端としピクセルの中心を通る半直線を視線と呼ぶ。図 2に示すようにレイキャスティング法ではピクセルごとに視線ベクトルを生成し、その視線上のボクセル群をスクリーンに投影する。

サンプリング方法の単純化　従来のサンプリング方法では、視線の方向に対して等間隔にボクセルをサンプリングしてきた。この方法を視線等間隔サンプリング法と呼ぶ。視線等間隔サンプリング法では図 3(a) のように、ボリュームデータをメモリバンクに分割して格納すると、同一視線上の複数のボクセルが同一メモリバンクに格納されている状況が発生する。これは、ピクセル値計算の並列実行を妨げ、高速化の障害となる。

この問題を回避するために ReVolver/C40 では、視線ベクトルの成分の絶対値のうち、最大値をもつ座標軸を主軸と定め、この主軸の方向に対して等間隔にボクセルをサンプリングする方法を採用している。この方法を主軸等間隔サンプリング法と呼ぶ。この主軸等間隔サンプリング法により、図 3(b) に示すようにバンクコンフリクトが生じなくなる。

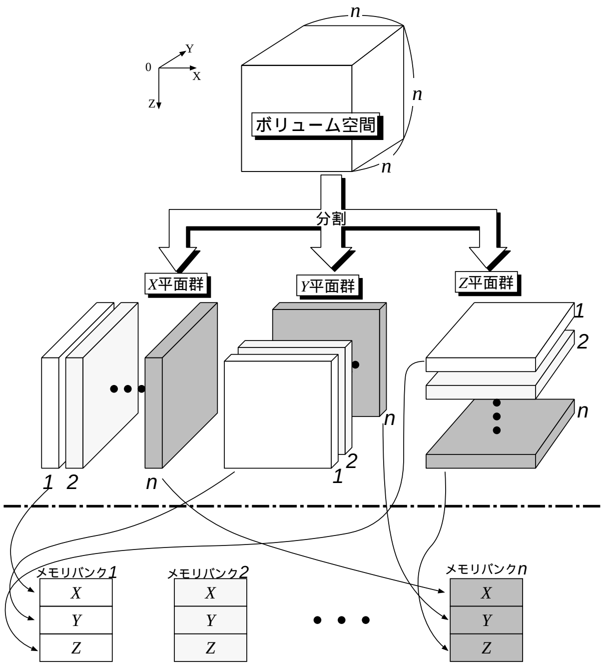


図 4: ボリュームデータの格納方法

バンクコンフリクトのないメモリ構成　主軸等間隔サンプリング法では、ある座標軸に垂直な平面にボリュームデータを分割してメモリバンクに格納すれば、その座標軸を主軸として持つ視線に関してはバンクコンフリクトフリーとなる。しかし、任意方向の視線の主軸が、この座標軸を主軸として持つとは限らない。そこで、x,y,z 軸のいずれにも垂直な平面が存在するように分割すれば任意方

向の視線に対してバンクコンフリクトフリーとすることができる。

そこで、ReVolver/C40 の 3 次元メモリ構造では、図 4 のようにボリューム空間を x,y,z 軸に垂直な n 枚の平面に分割し、X,Y,Z- 平面群とする。ただし、n はボリューム空間の 1 辺の大きさを表している。

各平面群に対して、平面に 1,2,· · ·,n と番号を付け、同じ番号を持つ平面のデータをその番号のメモリバンクの対応する領域に格納する。すなわち、ボリューム内の座標 (p,q,r) のボクセルは、メモリバンク p の X 領域、メモリバンク q の Y 領域、メモリバンク r の Z 領域の 3 箇所に格納されるので、ボリュームデータが三重化されて格納されることになる。

この構成により、各平面群の平面は全て別のメモリバンクに格納されているので、視線が任意方向であっても、主軸等間隔サンプリングを行うことによりバンクコンフリクトが生じないことを保証できる。

並列処理による高速化　ReVolver/C40 においては、図 5 のように画像生成 ·シェーディング ·画像出力からなるフレームを単位としたパイプライン処理を行うことで高速な表示を可能としている。

2.2　Revolver/C40 の構成

ボリュームレンダリングにおける大きな 2 つの処理画像生成/シェーディングに対してそれぞれ視線生成ステージ、ピクセル値計算ステージ/シェーディングステージを割り当てる。

よって、ReVolver/C40 は、3 つのステージと、全体をコントロールする System Control Unit から構成される [1]。これらは図 6 のように接続され、パイプラインで動作する。

ホストインターフェース **(System Control Unit, SCU)** システム全体の制御を行い、ホストマシンから受け取ったコマンド ·プログラム ·ボリュームデータを適切なステージに伝達する。

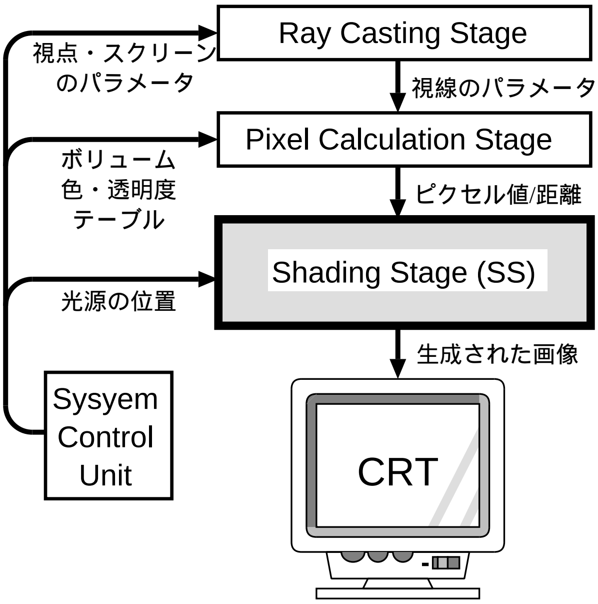


図 6: ReVolver/C40 の全体構成

視線生成ステージ **(Ray Casting Stage, RCS)** スクリーン上の全てのピクセルについて、視点からピクセルを通過する視線ベクトルを生成し、ボリュームと交差する視線をピクセル値計算ステージへ送る。

ピクセル値計算ステージ **(Pixel Calculation Stage, PCS)** ボリュームデータを保持しているステージであり、視線生成ステージからの視線データに基づいてボクセル値を読み出す。ボクセル値は 1byte のインデックス値なので、これを用いて属性テーブルを参照し、ピクセル値計算を行う。

Processing Element(PE) でピクセル値を計算する際、視線上のボクセルのうち視点に一番近い無色 ·透明でないボクセルを表面と判定し、表面と視線との交点の座標を、ピクセル値とともに隣の PE へ送る。この座標値は、深さの情報としてシェーディングステージで用いられる。

シェーディングステージ **(Shading Stage, SS)** ピクセル値計算ステージから受け取ったデータをもとに depth gradient shading を行い、その結果を画面に表示する。

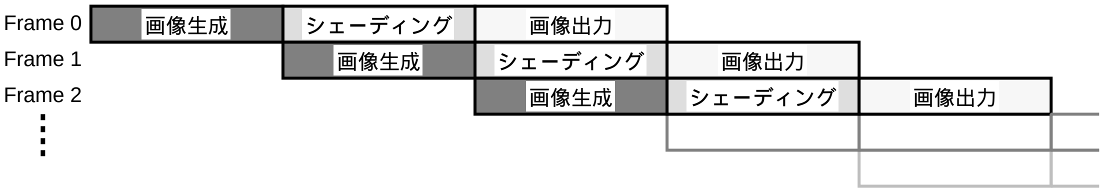


図 5: ReVolver/C40 のパイプライン処理

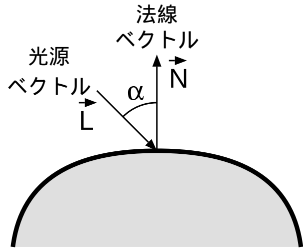


図 7: $\cos\alpha$ の計算方法

3 シェーディングステージの概要

本節では、Revolver/C40 において採用されている *depth gradient shading* について述べ、シェーディングステージの構成と処理について述べる。

3.1 シェーディング処理

シェーディングとは、光源情報を用いて生成画像の輝度を補正することにより、画像に現実感を持たせる手法である [2, 3, 4]。

物体の表面上の 1 点における輝度を求めるには、その点における法線ベクトルと光線の入射ベクトルという二つの要素が必要となる。I_{in} を入射光の強さ、α を入射角、k_d を物体表面の拡散反射率とすると、反射光の強さ I_d は次式で与えられる。

$$I_d = I_{in} \cdot k_d \cdot \cos\alpha \tag{1}$$

なお、$\cos\alpha$ は図 7 に示すように法線方向の単位ベクトル $\mathbf{N}$、光線の入射方向の単位ベクトル $\mathbf{L}$ を用いて次のように表わされる。

$$\cos\alpha = \mathbf{N} \cdot (-\mathbf{L}) \tag{2}$$

ボリュームレンダリングにおいては、ボリュームデータそのものは表面の法線ベクトルに関する情報を持たないため、周りのボクセルの状態から法線情報を推定する必要がある。

この法線情報の求める方法として、ボリュームレンダリング専用並列計算機 ReVolver/C40 は、*depth gradient shading* を採用している。

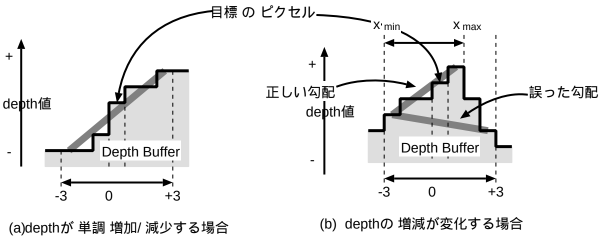
3.2 depth gradient shading

視線上のボクセル値を視点に近い側から調べ、ボクセル値が大きく変化する所をボリュームの表面とみなすことができる。スクリーンからこの表面までの距離を各ピクセルごとに求め、2 次元配列に格納したものを深さバッファ(depth buffer) と呼び、深さバッファに格納される値を depth 値と呼ぶ。

depth gradient shading は、この depth 値から表面の法線を近似計算し、ピクセル値と法線ベクトルをもとに各ピクセルの輝度を補正する手法である。

ある点 P における法線ベクトルを求める場合、X 方向については図 8の (a) のように depth 値が単調増加する場合, 斜線区間の両端ピクセルについて depth 値を取り出し、depth 値の差と座標の差を使って勾配を近似する。また、図 8の (b) のように depth 値の増加が変化する場合は、(a) の場合と同様に差を取ると勾配が平坦になり突起が再現されないので、近似に用いる区間を選択する。Y 方向についても同様に勾配を近似し、2 つの勾配から法線ベクトルを求める。

求められた法線ベクトル N と System Control Unit から受け取る光線の入射ベクトル L と式 (3.1),(3.2) からピクセルの輝度が求められる。



(a)depthが 単調 増加/ 減少する場合

(b) depthの 増減が変化する場合

図 8: 法線ベクトル

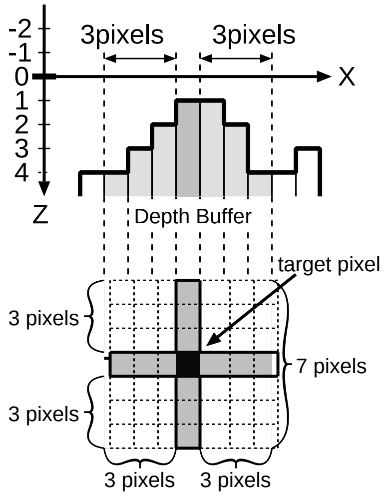


図 9: depth gradient shading の法線計算に必要なピクセル

3.3 Depth Gradient Shading の並列化

ReVolver/C40 の行う depth gradient shading は、図 9 に示した求めるピクセル自身を含む上下左右合わせて 13 ピクセルの領域に対するデータから法線ベクトルを求めて陰影を計算する。だから、スキャンラインごとにサイクリック分割はできない。そこで画面を図 10 のように 16 分割し、それぞれを一つの PE に割り当てることで、並列処理による性能向上が期待できる。

この時、各 PE 上には自分がシェーディング処理を施す領域に加え、図 11 のように上下左右各 3 ピクセル分の周辺領域の depth 情報を割り当てる。このように、depth 情報を一部重複して各 PE に割り当てることで、法線計算時の隣接 PE 間の通信を不要にしている。

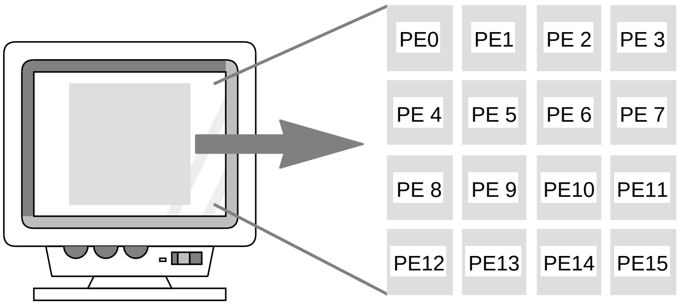


図 10: フレーム・バッファの分割

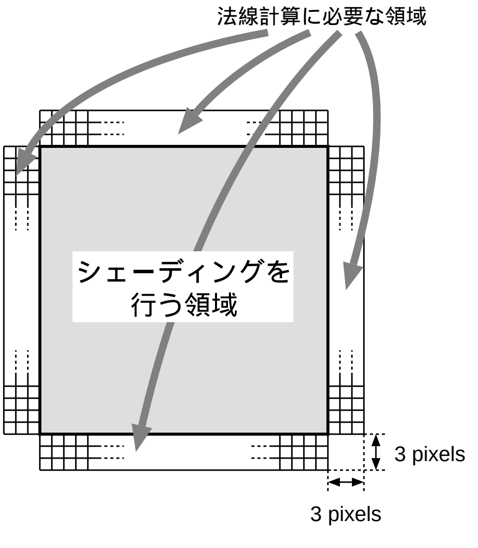


図 11: 法線計算に必要な depth 情報の範囲

4　シェーディングステージの実装

4.1　シェーディングステージの処理と構成

シェーディングステージは、ピクセル値計算ステージから受け取ったピクセル値とその他のデータを元にして、ピクセル値に depth gradient shading を施してピクセル値を補正する。そして、その結果を画面に出力する。

そこで、シェーディングステージを以下の 3 つの部分に分ける。シェーディングステージの構成を図 12 に示す。

Pixel Transceiver (PT)　各ピクセルのデータを前のステージであるピクセル値計算ステージから受けとり、そのピクセルの座標をもとに 16 分割された PE のうち、どの PE の担当であるかを計算し、その PE のもつメモリ (Receive Buffer) へ受け取ったデータを書き込む。

Shading Unit (SU)　Receive Buffer に Pixel Transceiver によって書き込まれたデータを Digital Signal Processor(DSP) が読み出して、シェーディング処理を施し、その結果をメモリ (Output Buffer) に書き込む。図 12に示すように Shading Stage は、中心にシェーディング処理をする DSP、DSP をはさむようにしてメモリバッファとしての Receive Buffer と Output Buffer の 3 つの部分から構成される。

Video Signal Generator (VSG)　CRT controller が Shading Unit 内の Output Buffer から結果を呼び出して、D/A converter に入力して画面に出力される。

4.2　基板設計

メイン基板の構成を図 13に示す。

Shading Unit 内の各 PE はサブ基板 1 枚に実装する。すなわちサブ基板は 16 枚ある。メイン基板には 8 枚までのサブ基板が搭載可能なので、メイン基板を 2 枚使用する。

Pixel Transceiver は Field Programmable Gate Array(FPGA) １個, Video Signal Generator は主に FPGA １個と D/A Converter 1 個で実現される。

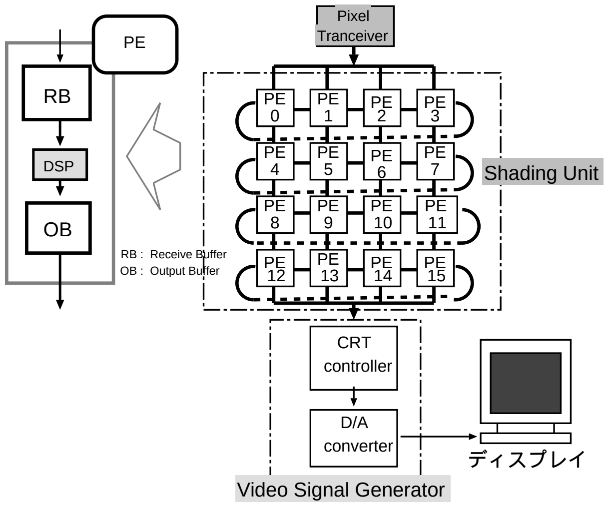


図 12: Shading Stage の全体構成

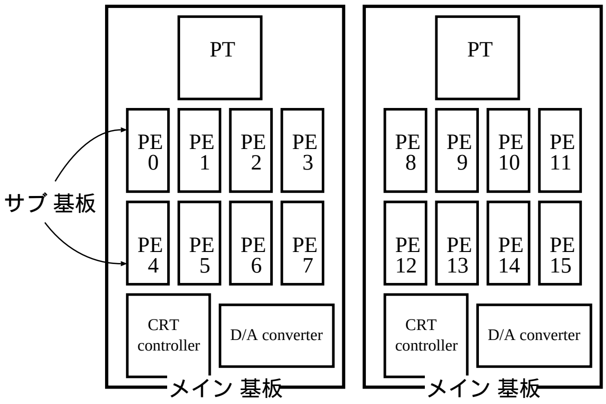


図 13: 基板設計

すなわち、メイン基板 1 枚には、主にサブ基板 8 枚と FPGA2 個と D/A Converter1 個が搭載されることになる。

なお、サブ基板間はコネクタで接続し、各 PE 間のメッシュ結合とするが、PE0 と PE3,PE4 と PE7,PE8 と PE11,PE12 と PE15 については、リング結合として実装する。

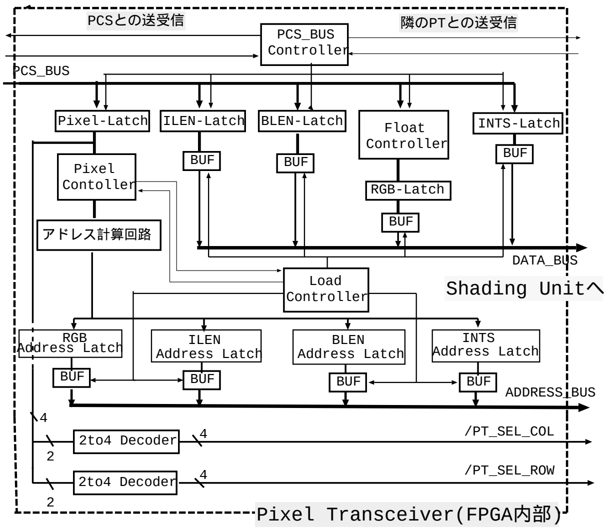


図 14: FPGA 内部

4.3 Pixel Tranceiver の実装

Pixel Transceiver は、FPGA を用いて実現される。

図 14に FPGA 内部の様子を示す。まず、PCS_BUS Controller が、Pixel Calculation Stage に対してデータ送信を要求し、PCS_BUS を通して流れてくるデータを種類ごとにどのラッチに格納するかを制御する。格納されたデータはそれぞれ並列に処理される。

Pixel Latch に格納されたデータ (スクリーン座標) スクリーン座標からどの PE が選択されたかを示す信号 (/PT_SAM_OF_ROW,/PT_SAM_OF_COL) を生成する。

Pixel Controller によって、いくつの PE に共有されているかを判定し、3 つの PE に共有されている場合は、座標値を計算し直して Pixel Controller 内の 3 つの Latch に格納する。

それぞれの座標値に対してひとつずつ Address Latch に格納する。

RGB 値 RGB 値は、Flaot Controller に入力され、データフォーマットの変換が行われ RGB Latch に格納される。Load Controller によって DATA_BUS,ADDRESS_BUS それぞれに流すべきデータとアドレスのバッファの開閉を制御し、Shading Unit 内の選択された PE へ送信する。

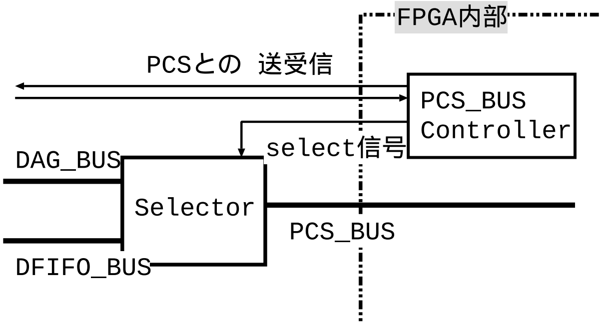


図 15: PCS と PT を流れる 2 本のバスの制御

以下にその詳細について示す。

ピクセル値計算ステージへのデータの送信要求 (PCS_BUS Controller) 図 15 のように、ピクセル値計算ステージから流れてくるバスは 2 本あるが、FPGA の外部にセレクターを設けて、PCS_BUS Controller がセレクト信号を送ることによって、FPGA に入ってくるバスを 1 本にする。そうすることによって、FPGA のピンの数を減らすことができる。また、ピクセル値計算ステージに対してデータの送信要求し、データを受け取る。

ここで、データバス (DAG_BUS) から受け取るデータのフォーマットを 図 16と表 1に示し、データバス (DFIFO_BUS) から受け取るデータのフォーマットを図 17と表 2に示す。

DAG_BUS を流れてくる五つのデータは、一度データの転送要求をするとクロックに同期して五つ連続してながれてくるが、DFIFO_BUS を流れてくるデータは、一度データの転送要求をしても一つずつしかデータがながれてこないので一回一回データを受け取ったことを確認してからデータの転送要求を出す。

また、DAG_BUS から流れてくるデータのうち、PE 内でシェーディング処理を行うのに使用するデータは 3~5 のみであり、1~2 は無視する。

表 1: DAG_BUS を流れるデータ

順番	データ名	解説
1	IN1,BV1	ボクセルの入射点の y 座標と基本ベクトル。
2	IN2,BV2	ボクセルの入射点の z 座標と基本ベクトル。
3	FRMDN,MASK	フレーム終了フラグ, サンプリング開始位置
	AXIS,DIR	主軸名, 主軸方向
	PXL.x,PXL.y	スクリーン座標をあらわす。
4	BLEN	基本距離 (視線ベクトルの基本距離) をあらわす。
5	ILEN	初期距離 (視点から、視線ベクトルがボリューム空間に入射した点までの距離) をあらわす。

表 2: DFIFO_BUS を流れるデータ

データ名	解説
R	ピクセル値の赤成分。(float)
G	ピクセル値の緑成分。(float)
B	ピクセル値の青成分。(float)
DIR、INTS	主軸方向, 交点座標 (視線から一番近い無色透明でないボクセルを表わす)。

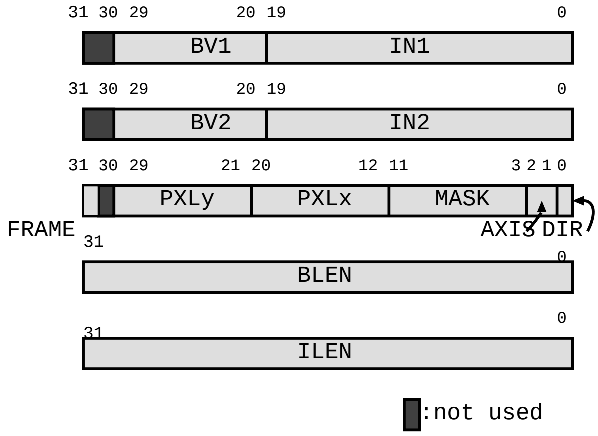


図 16: DAG_BUS から受け取るデータのフォーマット

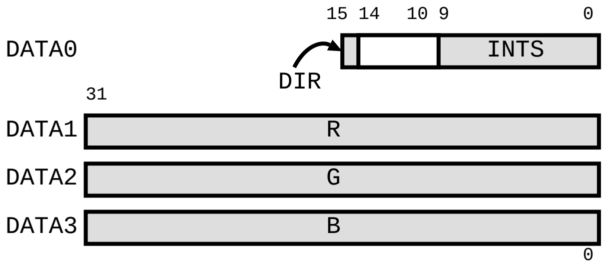


図 17: DFIFO_BUS から受け取るデータのフォーマット

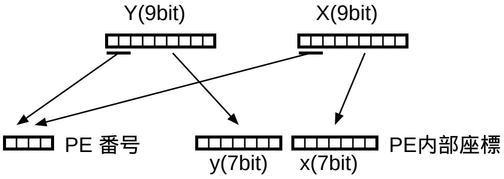


図 18: X, Y座標値から求める PE 番号

データフォーマットの変換 (Float Controller) ピクセル値計算ステージから Pixel Transceiver が受け取るデータのうち、ピクセル値を表わす RGB 値はそれぞれ 32bit の float (DSP の内部形式) で送られてくる。画像表示に用いる分には、RGB それぞれ 8bit integer で十分であるため、Pixel Transceiver において float → integer の変換を行う。この変換については、DSP マニュアルに従う。[5]

PE の選択と Receive Buffer への書き込み (2to 4decoder, アドレス計算回路) 前述したように Shading Unit の持つフレームバッファは分割されてそれぞれの PE が持っている。このため、Pixel Transceiver は、ピクセルの座標値にしたがってそのピクセルを持つ PE を選択し、データを書き込む。したがって、Pixel Transceiver において以下の値を計算する。

- PE の番号
- Receive Buffer 内部のアドレス

前者は、図 18 にしめすように、X, Y 各座標値の上位 2bit がそのまま PE 番号になる。実際に PE に選択されたことを知らせる時は、PE 番号 を X, Y 各座標値の上位 2bit それぞれについて 2to4 デコーダを通す。

図 19 の場合には、スクリーン座標の Y,X の上位 2bit がそれぞれ (10,01) が選択されており、2to4 デコーダを通すことで Y については、横方向の 4 本の信号線 (SAM_OF_ROW) のうち、3 番目だけ Active になり、X については縦方向の 4 本の信号線 (SAM_OF_COL) のうち、2 番目だけ Active になり、両方の信号線が Active になった PE9 が選択されたことになる。

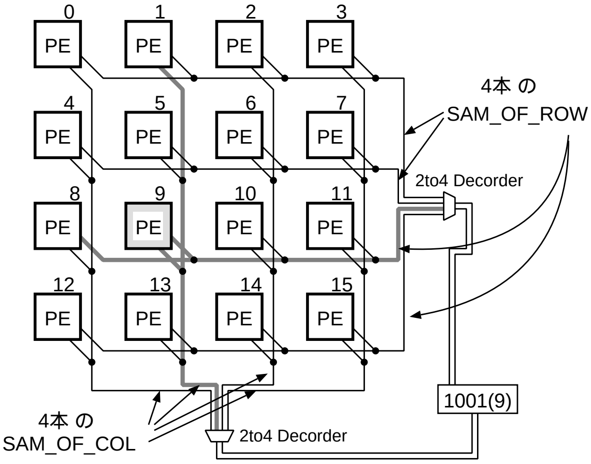


図 19: PE の選択

Receive Buffer のアドレスについては以下のようにして求める。

ひとつの PE はピクセル値については、128^2個のデータでよいが、シェーディングをするのに必要な交点座標や初期距離や基本距離については、上下左右それぞれ 4 ピクセル分の領域を重複 (合計 $(128+8)\times(128+8)$ ピクセル) して所有している。(第 3.3節で述べたように本来,$(128+6)\times(128+6)$ ピクセルでよいが、アドレスの計算がしやすいように上下左右それぞれ 4 ピクセルずつもたせている。

ここで、RGB 値のアドレスを addr1, 交点座標や初期距離や基本距離のアドレスを addr2 とし、スクリーン座標値を (X, Y) とすると、Receive Buffer 内部の座標 (x, y) は図 18 により求められ、Receive Buffer 内部のアドレスは以下の式で求まる。

$$\begin{aligned} addr1 &= y \times 128 + x \\ addr2 &= (y+4) \times 136 + x + 4 \end{aligned}$$

たとえば、136 という定数倍の乗算は、以下のように因数分解される。

$$addr2 = (y+4) \times 136 + x + 4$$

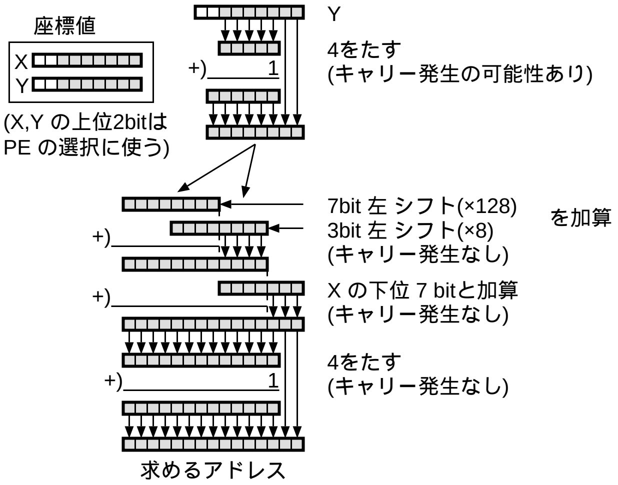


図 20: PE 内アドレスの計算

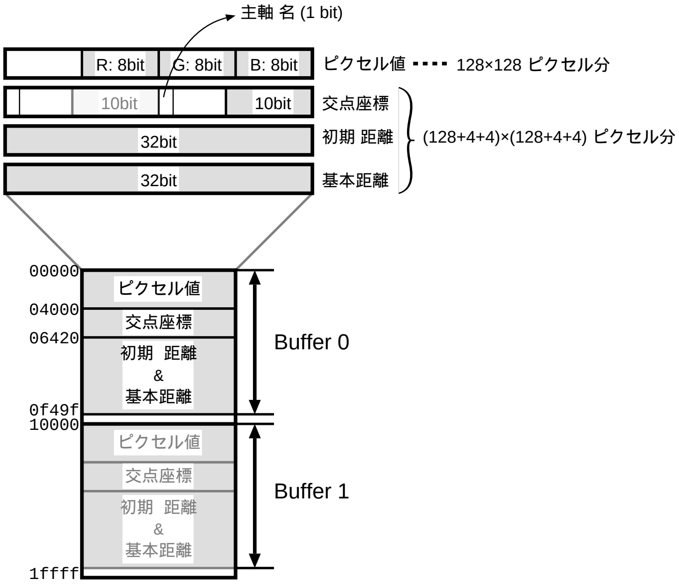


図 21: Receive Buffer(SRAM) のメモリマップ

$$
\begin{aligned}
&= (y+4)\times(128+8)+x+4 \\
&= (y+4)\times 128+(y+4)\times 8+x+4
\end{aligned}
$$

これより、図 20 のように加算器のみで構成される。

PE 内でシェーディング処理をするのに必要なデータ (RGB 値, 交点座標, 初期距離, 基本距離) は、Receive Buffer に書き込まれる。Receive Buffer は、SRAM で実現されているのであるが 図 21 のようにデータを記憶する。すなわち、一番左端の数字はバンクスイッチとよばれるものでこの数字を切り替えることでダブルバッファ構成を実現しており、残り 4 桁のアドレスは、以下のようにしてもとめられる。

$$
\begin{aligned}
RGB\text{値のアドレス} &= addr1 \\
\text{交点座標のアドレス} &= (addr2/2)+4000H \\
\text{初期距離のアドレス} &= (addr2\times 2)+6420H \\
\text{基本距離のアドレス} &= (addr2\times 2)+6421H
\end{aligned}
$$

ただし、交点のアドレスで addr2 を 2 で割る (1bit 右シフト) のは、10bit の交点データを上位 half word と下位 half word につめるためである。また、初期距離, 基本距離は、一つのピクセルに対して連続していれる。

複数の PE に共有されているデータの送信 (Pixel Controller) 図 22 のように 128^2個のピクセルを持つ PE 内の座標を (x,y) とすると $(128+8)\times(128+8)$ 個のデータを持つ PE の座標は (x+4,y+4) で与えられ、(127,127) という座標は、PE0 では (131,131) ,PE1 では (3,131),PE4 では (131,3) という三つの PE に重複して配られる。

すなわち、一つのピクセルの depth 情報は、最大で 3 つの PE に共有されるので、3 通りのアドレスを格納できるように 3 つの Latch を Pixel Controller 内に設ける。以下のいずれか一つの条件を満たすピクセルの場合、複数の PE に重複して配られる。

$$
\begin{cases}
x<4 \text{ ならば、} & \text{左隣の } (x+132, y+4) \\
y<4 \text{ ならば、} & \text{上隣の } (x+4, y+132) \\
x>124 \text{ ならば、} & \text{右隣の } (x-124, y+4) \\
y>124 \text{ ならば、} & \text{下隣の } (x+4, y-124)
\end{cases}
\tag{3}
$$

Receive Buffer,Output Buffer のバンク切替え

Shading Unit 内の Receive Buffer と Output

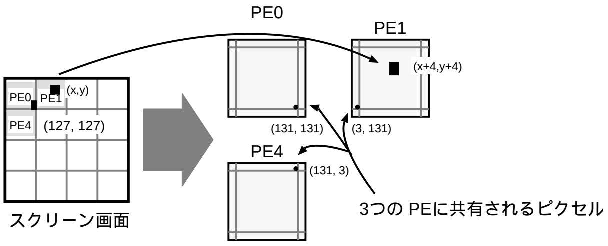


図 22: 複数の PE で共有されるピクセル

Buffer は、ダブルバッファリングを行う。よって、バンクを切替える制御をする。

DAG_BUS から 3 番目に受け取るフレーム終了ビット (FRMDN) が必要な理由は、視線生成ステージ内が、ボリューム空間と交差しない視線データをピクセル値計算ステージに送らない。Pixel Tranceiver ではピクセル値計算ステージからもらったピクセルデータの数からはフレームの終わりが判定できない。そこで、Pixel Transceiver は、視線生成ステージが生成したフレームの終了を示す 1 bit のデータを受信することで判定する。また、PE 内のシェーディング処理を行う DSP が Receive Buffer 内のデータをすべて読み込むことが終了した信号を DSP から受け取る。つまり、Receive Buffer に対する読み込みと書き込みの両方の処理がおわったことを確認してから、バンクを切替える。

また、Output Buffer に関しては、シェーディング処理を施した結果をかきこむ DSP から書き込みが終了したことを示す信号と Video Signal Generator からデータの読み込みの終了を示す信号を受信する。両方の処理が終了したことを確認してからバンクの切替えをする。

データ、アドレスの送受信するタイミングの制御 (Load Controller) Shading Unit 内の Receive Buffer に送信するためのデータバスとアドレスバスを制御すると同時に一つのピクセルに対して処理が終わると PCS_BUS Controller に次のピクセルを受信するように信号を出力するなど FPGA 全体の制御を行う。

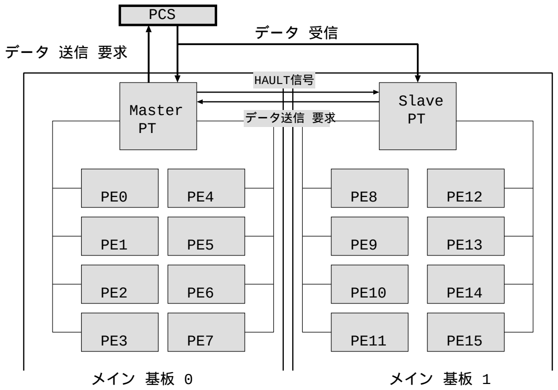


図 23: Master 基板と Slave 基板 (PT)

基板設計上必要な制御 第 4.2 節で述べたようにメイン基板が 2 枚あり、それぞれに Pixel Transceiver の役目をする FPGA がのっており、ピクセル値計算ステージに対するデータ転送要求やピクセル値計算ステージからのデータの受信は、両方の FPGA に対して実行されるように設計されている。よって、2 つの FPGA に対して Master と Slave を決めて、Master の Pixel Transceiver が、Slave の Pixel Transceiver のデータ転送要求信号を受信し、Master と Slave の両方の信号がアクティブになったときに Master が代表してピクセル値計算ステージにデータ転送要求信号を送る。

一方、データの受信は両方の Pixel Transceiver ができるように制御する。

また、送られてきたピクセル値はどこの PE に配るのか Pixel Transceiver 内で求めるが、例えば、図 23のように PE0∼PE7 をメイン基板 0 (Master) に、PE8∼PE15 をメイン基板 1(Slave) に設定すると、Master の Pixel Transceiver で処理していてそのピクセル値が PE10(Slave) にあるピクセル値である場合捨てられる。すなわち、となりの基板上の PE が選択された場合そのデータは捨てられる。

そうすることによって、自分の基板上にある PE にだけ送信すればよく、通信をしなくてすむ。

4.4 Shading Unit の実装

Shading Unit は、16 台の PE を格子上に配置し、それぞれをメッシュ状に結合した構成となる。

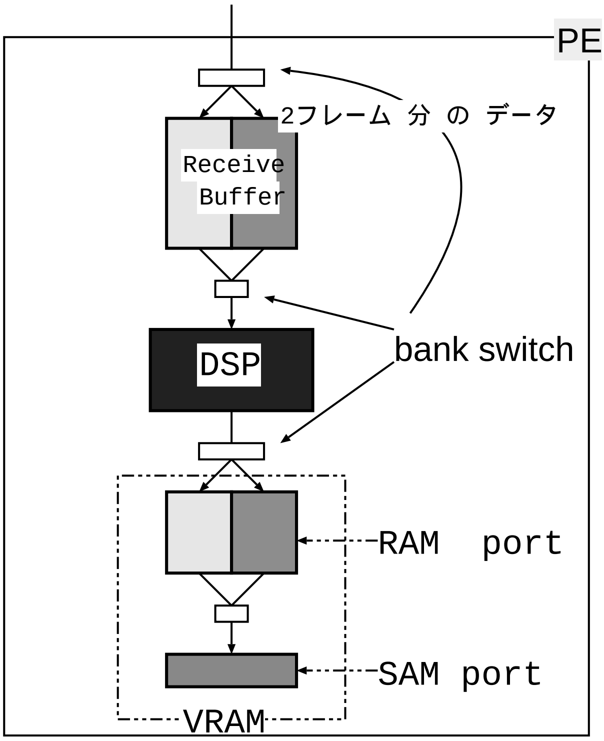


図 24: PE の内部構成

PE の構成は、図 24 に示すようにシェーディング処理を行う DSP を中心に、ピクセル値計算ステージからピクセルデータを受け取るための Receive Buffer と、シェーディング処理を終えたピクセル値を格納する Output Buffer とで構成される。Output Buffer はディスプレイ画面に出力するためのフレームバッファの役割も兼ねている。これを 1 台の PE とし、16 台の PE で並列処理を行う。図 24 のように Receive Buffer に対する Pixel Transceiver からの書き込みと DSP への読み込み、Output Buffer に対する DSP からの書き込みと Video Signal Generator への転送という処理を並列に行うため、Receive Buffer と Output Buffer はそれぞれ 2 バンク構成にし、フレーム単位のシェーディング処理が終わるたびに Pixel Transceiver がバンクを切り換える。

以上の構成により、Shading Unit はピクセル値計算ステージからのピクセル値の受け取り処理や Video Signal Generator への転送処理から完全に独立し、シェーディング処理に最大性能を発揮することができる。

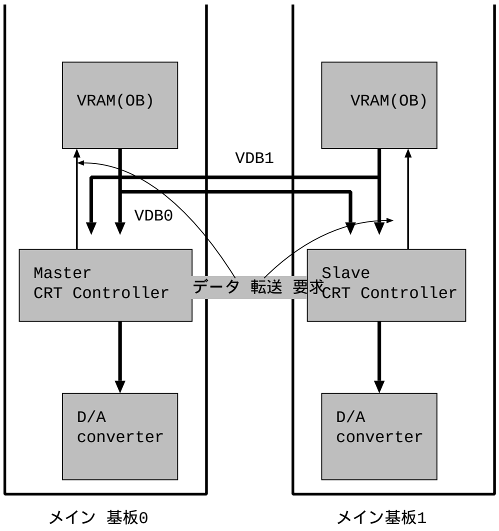


図 25: 実装上の VSG

4.5 Video Signal Generator の実装

図 25 に実装上の Video Signal Generator の構成を示す。第 4.1 節や第 4.2 節で述べたように Video Signal Generator は、主に CRT Controller と D/A Converter から構成し、CRT Controller は、FPGA によって実現される。また、図 25 ように 1 枚のメイン基板上にそれぞれ 1 つずつ実装される。

図 26 に CRT Controller(FPGA 内部) の構成を示す。

テレビの走査線と同じ順番に、今どこのピクセル値が必要かを調べてそのピクセルがどこの PE に所有されているかを調べて、それらの情報を PE 内の Output Buffer に送ると同時にデータを転送するよう要求する。そして、Output Buffer から Video Data Bus を介してデータが流れてくるが、基板設計の都合上 Video Data Bus が 2 本あるためにどちらのバスから流れてくるかを判定してセレクターによって制御し、D/A Converter に送信する。また、同時に D/A Converter に入力すべき種々の信号を生成する。

水平/垂直同期信号の生成　図 27 のようにディスプレイの大きさが 1024×536 であるとすると実際に出力される画面の大きさは、512×512 である。

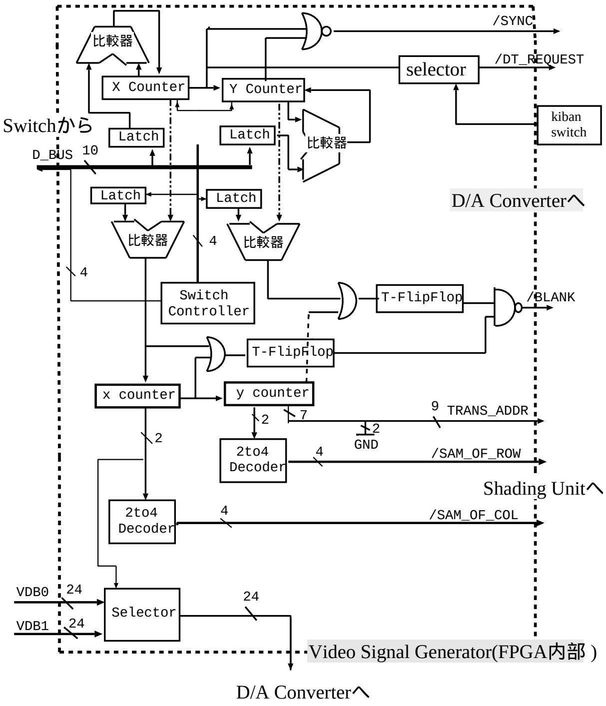


図 26: CRT Controller(FPGA 内部)

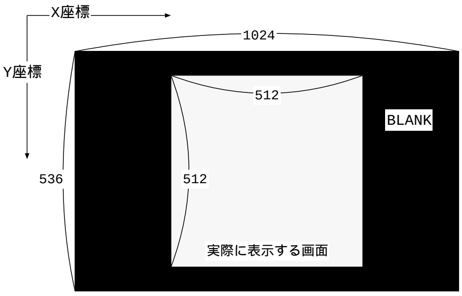


図 27: 表示画面

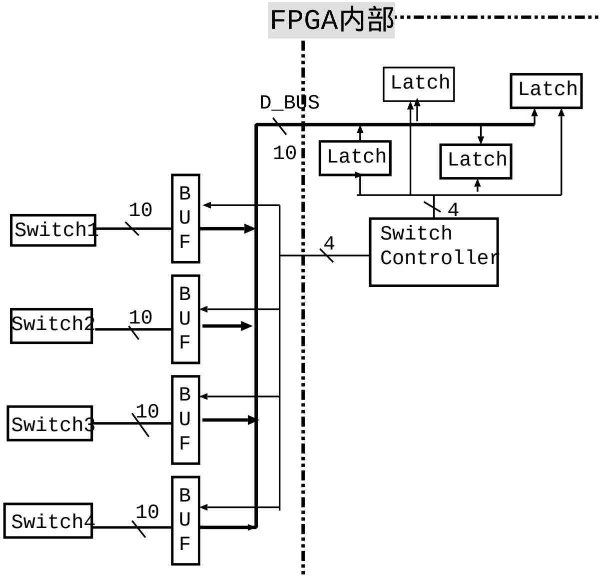


図 28: スイッチによる値の入力

画面上でいまどこを走査線が走っているかということを、すなわちスクリーン座標上のどのピクセルの値がほしいかということを知らせなければならない。よって、X,Y座標上の位置を計数するカウンターが2つ必要になる。また、1024×536のディスプレイと512×512の表示画面を別々にかぞえるカウンターが必要となり、合計4つのカウンターで実現する。

また、ディスプレイの大きさは、わからないのでカウンターに（X counter = 1024,Y counter = 536という値を外部からパラメーターとしてスイッチによって）値をセットする機能を付加する。

その構成を図28に示す。ドットクロックをX counterでかぞえることで1024カウントしたところで桁上がりが生じる。そのCARRY信号が水平同期信号であり、かつY counterのカウント入力とする。これを繰り返すことでY counterが536になると1画面分カウントしおわったことになる。そのときのCARRY信号が垂直同期信号となる。よって値を設定するためのスイッチと今設定された値をどのLatchに格納すべきかということを制御するSwitch Controllerによって実現される。

BLANK 信号の生成　実際に出力される画面の大きさは、図 27 のように 512×512 であるから、それ以外のディスプレイの部分を知らせる信号が BLANK 信号である。

PE 内の Output Buffer に対するピクセル値の転送要求表示すべきピクセルの座標を計算し、そのピクセルをもつ PE の番号を出力する。その求め方は、第 4.3 節に示してある方法 (2to4 decoder) と同じである。

データ転送要求信号と SAM_OE_ROW 信号 (第 4.3 節参照) がともに Active になると PE の VRAM 内で RAM ポート内にある 128 個のデータが SAM ポートにセットされる。すなわち、横一列の 4 つの PE が同時にセットされる。そのサイクルは、4 サイクルかかる。そして、SAM_OE_COL(4 本の縦方向の信号線) が Active になることである一つの PE が選択されたことになり、SAM ポートから 128 個連続で出力される。

基板設計上必要な制御　図 25 のように CRT Controller と D/A Converter は、メイン基板上に 1 個ずつのっているので、メイン基板 0 の CRT Controller を Master とすると Pixel Transceiver の場合と違って両方の CRT Controller を動かすことはできない。よって、Slave の CRT Controller を動かさないようにしなければならない。図 26 にあるように FPGA の外部に 0,1 しか選べないスイッチを設け、データ転送要求信号に selector を設定する。スイッチで 0 が設定された基板をメイン基板とするとメイン基板からしかデータ転送要求信号が送信されないことになり、Slave の CRT Controller には、データが送られてこない。

また、Output Buffer から出力されてくる RGB 値は、基板が 2 枚あるために Video Data Bus0 または Video Data Bus 1 を通ってくる。D/A Converter へ流れるバスは 1 本しかないのでどちらの Video Data Bus から流れてきたかを判断して制御しなければならない。これは、どの PE の列が選択されたかを示す SAM_OF_ROW 信号を selector の入力信号とすることで実現する。

5　おわりに

本稿において、ReVolver/C40 におけるシェーディングステージの構成と実装について述べた。

画像生成部 (視線生成ステージ、ピクセル値計算ステージ) とパイプラインピッチを合わせるために高速な depth gradient shading の手法を採用し、16 台の PE による並列処理を行っている。

また、depth gradient shading における法線計算では、周辺ピクセルの情報を必要とするため、あらかじめそれぞれの PE にピクセル情報を重複して持たせておくことにより、PE 間の通信を不要にしている。各 PE が持つ Receive Buffer と Output Buffer はそれぞれダブルバッファ構成になっている。そうすることで、データの書き込み/読み込みという処理がパイプラインで行える。

実装上では RB は 高速 SRAM に 2 フレーム分のデータを持たせることで 2 バンク構成にし、OB は DRAM と同じ方法でアクセス可能な RAM ポートと、高速なシーケンシャルアクセスが可能な SAM ポートを持っている VRAM を使用することで実現している。

この構成により、ピクセル値の書き込み/シェーディング処理/画像表示は独立し、フレーム単位のパイプライン動作を行うことが可能となった。

また、2 個の Pixel Transceiver で同じピクセルを処理させるとき、そのピクセルを持たない PE がのっている基板の方では、そのデータを捨てることによって PT 間の通信を不要にしている。

一方、Video Signal Generator における CRT Controller も 2 個あるがどちらの CRT Controller が Master かということを外部のスイッチで設定できるようにすることで Slave の CRT Controller からデータ転送要求信号が出力されなくなり、データ受信がおこらないようにすることができる。

このように 2 枚のメイン基板上の Pixel Transceiver も Video Signal Generator も全く同じものとして設計することによって、外部のスイッチによってどちらの基板を Master とするか Slave とするかを設定できる。

今後の課題としては、画像表示の機能を拡張することも考えられる。Output Buffer から CRT Controller へのデータバスは、現在 24bit であるが、これを 32bit とすることで透明度などの情報

を持たせることができる。また、Output Buffer には、大容量の VRAM を用いており、512×512 ピクセルのフレームバッファを最大 4 面までもつことができる。この性質を用いて透明度の値によって、4 面のフレームバッファの合成を行うこともできる。

謝辞

本研究の機会を与えて下さった、本教室教授富田眞治先生に深く感謝の意を表します。

数々の御指導、御助言を頂いた本教室中島浩先生、森眞一郎先生に心から感謝いたします。

本研究の共同研究者である、金喜都氏、中山明則氏、荻野友隆氏、本多俊之氏をはじめ、計算機システム講座の諸氏に感謝いたします。

また、この場をお借りして、筆者を支えて下さいました友人諸氏にお礼申し上げます。

参考文献

[1] 對馬雄次 t: ボリュームレンダリング専用並列計算機 – ReVolver/C40 –, 修士論文, 京都大学 (1995).

[2] 荻野友隆: ボリュームレンダリング専用並列計算機 ReVolver/C40 のシェーディングステージ (1995).

[3] Arie Kaufman(訳: 藤代一成): ボリュームビジュアライゼーション概論, *PIXEL No. 116*.

[4] Chen, L. e. a.: Surface Shading in the Cuberille Environment, *CG&A*, chapter 5(12), pp. 33–43 (1985).

[5] Texas Instruments: *TMS320C4x User's Guide* (1993).