特別研究報告書

DVI-Dインタフェースによる
高速低遅延データ転送

指導教員　　富田 眞治 教授

京都大学工学部情報学科

野田 裕介

平成18年2月10日

DVI-D インタフェースによる
高速低遅延データ転送

野田 裕介

内容梗概

近年，急速な計算機性能の向上に伴い，大規模な 3 次元ボリュームデータの高精度な数値シミュレーションへの期待が高まっている．中でも医療などの分野においては，大規模な 3 次元データの実時間数値シミュレーションの可視化技術の研究が現在進められている．

3 次元データの可視化手法としては，ボリュームレンダリングがある．ボリュームレンダリング処理の特性から，大規模な 3 次元データ実時間可視化に際しては，並列処理による高速化が必須となる．我々は，大規模データの可視化を実現する並列ボリュームレンダリング環境の構築を目標としている．大規模データの可視化環境の実現方法として，研究資源の投資量に応じて複数の方針が考えられるが，本稿では，可視化専用のハードウェア (VisA) による実現方法について述べる．また，実装対象となる，汎用 FPGA 搭載 PCI カード (VisA Pro カード) について，DVI-D インタフェースを重点的に，その特徴と仕様について紹介する．

並列ボリュームレンダリングにおいて，その高速化を考える場合，各ノード間の中間値の合成に伴う，データ転送がボトルネックになる．そのデータ転送の高速化のために，DVI-D Dual Link インタフェースを用いて実現する手法を提案する．本稿で提案する VisA は，128 ノードでの並列パイプライン処理を想定している．よって，そのノードスケーラビリティから，フレーム単位の並列処理ではなく，ピクセル単位での並列処理を想定する．

DVI-D インタフェースは，シングルリンク時で最大 3.96*Gbps* の高速データ転送が行えるが，Myrinet・Imfiniband と比較すれば，その高速性では劣る．しかし，安価な通信路として，本稿の提案に対しては，性能は優れていると言える．

DVI-D インタフェースは本来，ディスプレイ表示を目的としたインタフェースである．本稿では，ピクセル単位でのデータ転送を考えるので，ディスプレイ表示に伴うブランクやマージンによるオーバヘッドを削減した手法を提案する．汎用 PC 通信で利用するような，TCP/IP プロトコルでの Ethernet による通信でのパケット通信に対して，プロトコルのオーバヘッドがないため，低レ

イテンシのデータ転送が行える．

VisA Pro カードを用いた，データ転送の予備実装及び評価を行った．データ転送実験はまず，1 ノードにより，データの送受信をループさせて行った．高速データ転送の実装に伴い，転送開始時に不定な値の出力が検知されたり，その後でのメインデータの受信のタイミングがずれるということが起こった．そこでデータ転送の開始を伝送する REQ 信号を転送することで，初期に出力される不定なデータを読み込むことを防いだ．またスタンバイ用のデータを複数回ブランクを挟んで転送することで，その後のメインデータを安定して転送することに成功した．その結果，伝送路の最大動作周波数は 165MHz とされているが，VisA Pro カードに内蔵されているクロック，166MHz でのデータ転送を実現させ, 最大帯域幅以上である 3.97$Gbps$ でのデータ転送が可能であることを確認した．また REQ 信号や，スタンバイデータの転送量が最小限となるようにに最適化することで，データ転送開始から実際にデータが出力されるまで，102ns という低レイテンシでのデータ転送を実現した．

さらに 2 ノードで実装を行ったところ，1 ノードでは検知されなかった REQ 信号の不定な動作により，転送が正しく行われない場合が確率的に生じた．この不定動作はボードの個々によってばらつきがあると考えられる．よってノード数が増大することで，不定動作をするボードが増える恐れがある．しかし，これらはデータ転送の初期状態のみのバグであるので，ボリュームレンダリングでの初期の出力画像にバグを与えることはあっても，その後の出力画像には影響を与えないと考えられる．

ノード間の高速大容量データ転送の予備実装の完成により，種々の並列ボリュームレンダリングへの適用が期待できる．

High Speed Low Latency Data Transmission on DVI-D Interface

Yusuke NODA

Abstract

Recently, expectation of large-scale three-dimensional volume data for a highly accurate numerical simulation is rising as a computer performance improves rapidly. In the medical field etc., research of real-time visualization technologies of numerical simulation of large-scale three-dimensional data is being advanced.

Volume rendering is a technique of three-dimensional visualization. Because of the characteristic of the processing, for real-time visualization of a large-scale data set, speedup by parallel computation must be required. We aim for construction of parallel volume rendering environment for real-time visualization of a large-scale data set. Now, some policies can be thought as a method of achiving an enviranment of visualization of a large-scale data set, corresponding to the amount of investment of the research resources. In this thesis, we describe a way to realize an environment by using a special hardware for visualization (VisA). And, we introduce the characteristic and the specification of the general-purpose PCI card loaded FPGA (VisA Pro card), which is the implementation object of this thesis, and introduce DVI-D Interface chiefly.

Considering speedup in parallel volume rendering, data transmission for composition of intermediate value between each nodes is a bottleneck. For realizing the high speed data transmission, we propose a technique with DVI-D Dual Link interface. In this thesis, VisA assumes the parallel pipeline processing with 128 nodes. Because of the node scalability, the parallel computation is not by a frame unit, but by a pixel unit.

DVI-D interface can perform high speed data transmission of max $3.96Gbps$ at single link time. Compared with Myrinet and Imfiniband, it is inferior at the point of high-speed data transmission. But DVI-D Interface is low price and it is good enough in the proposal of this thesis.

DVI-D interface is originally an interface aimed for display. In this thesis, we propose data transmission by a pixel unit. So we propose the technique that cuts the overhead of the blank and margin with display. Opposed to

packet transmission by Ethernet with TCP/IP protocol in general-purpose PC Communication, Because of the reduction of the protocol overhead, low latency data transmission is enabled.

We implemented and evaluated the data transmission with VisA Pro card. We implemented with 1 node first. The transmission is looped. When the high speed data transmission is implemented, at the time of the initial transmission, output of an unsettled value was detected and at the time of main data transmission, the data reception was out of timing. We trasfered the REQ signal that tells the begining of the data transmission so that we prevented reading unsettled data which is the initial output. We transfer standby data with some blank several times so that we could transfer the main data stably. As the result, though max movement frequency of transmission channel is specified to be 165MHz, data transmission was relized at 166MHz by the clock which is built-in VisA Pro card. We confirmed that Data transmission in $3.97Gbps$ that is more than the max bandwidth of the DVI-D Interface specification. REQ signal and standby data transmission was optimized to minimum so that it took $102ns$ from the begining of the data transmission to the really data output, so low latency transmission is realized.

We implemented with the 2 nodes. Because of an unsettled REQ signal movement that was not detected with the 1 node implementation, the case that transmission was not performed correctly occurred randomly. It is thought that this random unsettled movement occurs unevenly in each of board, and the more number of the node increases, the more boards do unsettled movement. But this is the bug that occur at the initial data transmission state, so this give only the initial output of volume rendering some bug.

The implementation of high speed transmission of a large-scale data between nodes is completed, so it is expected to apply to various kinds of parallel volume rendering.

DVI-D インタフェースによる
高速低遅延データ転送

目次

第1章　はじめに

近年，急速な計算機性能の向上に伴い，大規模な3次元ボリュームデータの高精度な数値シミュレーションへの期待が高まっている．中でも医療などの分野において，大規模な3次元データの実時間数値シミュレーションの可視化技術の研究が現在進められている．大規模な数値データを単一の計算機で実時間処理することが不可能な場合，処理の分散，並列化により実現する手法は必須である．これまでも，PCクラスタ等の並列計算機環境を利用した，シミュレーションとその実時間可視化のシステムが開発されており，次世代シミュレーション技術として，従来の実験の代替手段となりうる「仮想実験型/仮想体験型のシミュレーションシステム環境」の構築が望まれている．このような次世代のシミュレーション環境では，オペレータによるシミュレーション対象へのインタラクティブな操作に対応して実時間シミュレーションを行うとともに，即刻その結果を可視化などの手段により提示することが求められる．我々は，実時間インタラクティブシミュレーション環境の実現に向けて，個人あるいは小規模な組織単位で占有利用可能なPCクラスタを用いて，インタラクティブな数値シミュレーション及びその可視化を実時間処理する環境について研究を行っている．

上記の可視化機構を実現可能なハードウェアアーキテクチャとして，我々はVisA(Visualization Accelerator)を提案した[1]，[2]．そのプロトタイプとして処理の手順をそのままにボリュームデータの処理可能サイズを半分に制約したVisA Proを開発している．VisAは専用ASIC (Application Specific Integrated Circuit)で作ることを前提としたアーキテクチャであるが，実装段階に適したFPGA(Field Programmable Gate Array)上への実装を考え，高速入出力リンクと大容量高速メモリを備えた汎用FPGA搭載PCIカード(以下VisA Proカードと呼ぶ)を東京エレクトロンデバイス(株)との共同企画で開発した．本稿では，並列処理を行ったデータの再構成の際に必要不可欠な機構である，高速低遅延な大容量データ転送の実現として，上記ボードの保持する高解像度の液晶表示に対応するためのインタフェースである，DVI(Digital Visual Interface)-D Dual Linkインタフェースを用いた実装を提案するものである．その際に，DVI-Dインタフェースを本来のディスプレイ表示のためのデータ転送方法とは異なる転送手法を用いて実装を行った．

章構成としては以下の通りである．2章において，ボリュームレンダリングの概要について説明し，専用ハードウェアであるVisAのアーキテクチャの説明，さらに，実装対象であるVisA Proカードの仕様を述べる．3章では本稿で提案するDVI-Dインタフェースを用いたデータ転送方法やその特徴，利点・欠点などを述べる．4章では今回行ったデータ転送の予備実装について説明，および評価を行い，5章でまとめる．

第2章　並列ボリュームレンダリング

本章ではまず，ボリュームレンダリング(Volume Rendering)について述べる．その後，専用ハードウェアを用いた並列ボリュームレンダリングの実現として提案した可視化専用グラフィクスカードVisAの説明，そのプロトタイプであるVisA Proカードの説明を行う．

2.1　ボリュームレンダリング

ボリュームレンダリングとは，3次元対象の内部構造や動的特性を可視化するものである．ボリュームレンダリングにおいて，可視化の対象となる3次元空間を，ボリューム空間(Volume Space)と呼ぶ．ボリューム空間には，科学技術計算の結果や自然現象などのデータセットが半透明のボリュームとして色や透明度などに変換されて置かれる．

ボリューム空間を単位立方体に分割し，離散的にデータを与えたものをボクセル(Voxel)と呼ぶ．各ボクセルには色や透明度の値(ボクセル値)が与えられる．レンダリング結果を表示するスクリーン(Screen)はピクセル(Pixel)と呼ばれる単位正方形の集合として捉えられる．

ボリュームレンダリングは，ボリューム空間に対して，スクリーンからの視線と交差するボクセルのサーフェイスを算出してレンダリングを行うサーフェイスレンダリング(Surface Rendering)とは異なり，ボリュームの色や透明度を直接スクリーンに射影する手法である．図1に示すように，サーフェイスレンダリングと，ボリュームレンダリングでは参照するボクセルが異なる．

ボリュームレンダリングでは，シミュレーションにより得られた3次元空間上の数値データとしてのボクセル値を，色Cと透明度tに対応づけることで，3次元空間内部のデータの分布状況を可視化する．具体的なボリュームレンダリ

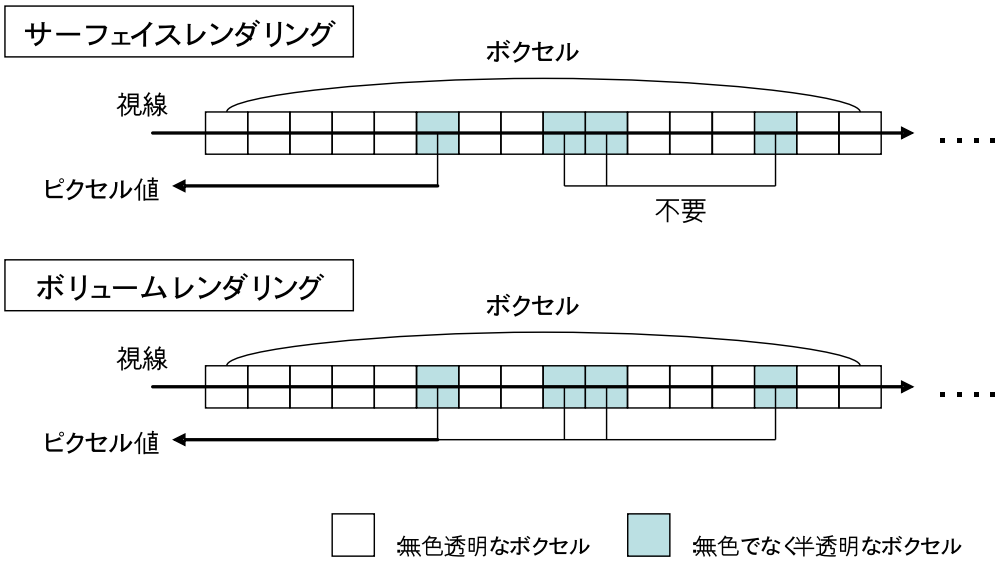

図 1: 参照するボクセルの違い

ングの手法としては，レイキャスティング (Ray Casting) があるが，これは図 2 に示すように，まず視点からスクリーンの各ピクセルに対して視線 (Ray) を出し，視線上のボクセルの値を視点に近い順に $v_o, v_1, v_2, \cdots$ として，ピクセル値は次の式 (畳み込み演算) で計算することで得られる．

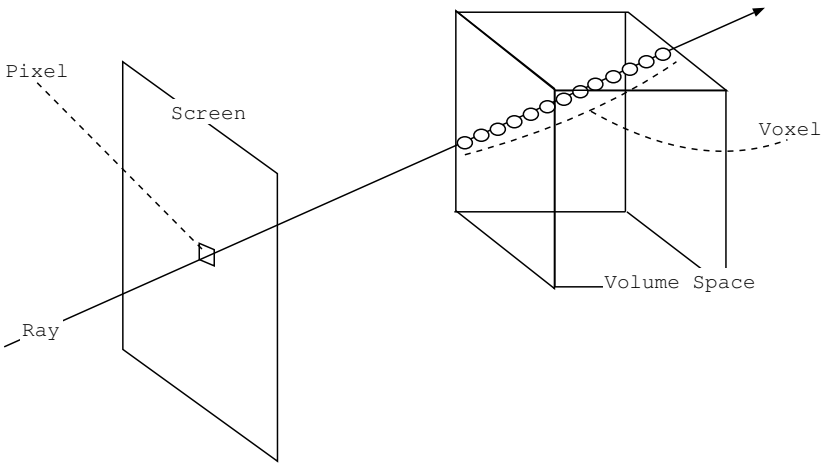

図 2: ボリュームレンダリング

$$C_k = \sum_{i=0}^{k}(1 - t(v_i)) \cdot c(v_i) \cdot \prod_{j=0}^{i-1} t(v_j)$$

$$T_k = \prod_{i=0}^{k} t(v_i) \tag{1}$$

ここで $c(v_i), t(v_i)$ はそれぞれボクセル値 v_i を，色，透明度に変換した値であることを示している．さらに，式 (1) は以下のような漸化式に変換できる．

$$\begin{aligned} C_k &= C_{k-1} + (1 - t(v_k)) \cdot c(v_k) \cdot T_{k-1} \\ T_k &= t(v_k) \cdot T_{k-1} \end{aligned} \tag{2}$$

この畳み込み演算には，演算区間をいくつかの部分区間に分割し，それぞれの区間で得られる計算結果に対して，再度畳み込み演算を行うことで正しい結果を得ることが可能であるという性質がある．そこで，1) シミュレーションサーバの各ノードが，自身が生成したデータに対する部分 3 次元空間 (以下，サブボリュームと呼ぶ) に対して畳み込み演算を行い，2) そこで得られた画像とボクセル毎の透明度を，視点からの距離の順番に従って合成する，という手順で並列処理が可能である．

並列ボリュームレンダリングの実装に際しては，可視化処理に対してどれだけの投資が可能かに応じて，いくつかの方法が考えられる．まず，可視化用のハードウェア・アクセラレータを用いるか，そうせずにソフトウェアのみで行うかという選択がある．また，ハードウェア支援を行う場合においても，テクスチャベースのボリュームレンダリングを行う汎用グラフィクスカードを用いるか，あるいは直接ボリュームレンダリングを行うボリュームレンダリング専用のハードウェアを実装するかの選択が可能である．可視化環境の構築に対して，研究資源の投資量に応じて複数の方針が考えられる．

本稿では，その中でもボリュームレンダリング専用ハードウェアを用いた実装という手法を採択している [3]．以下で専用グラフィクスカード VisA について詳細に述べる．

2.2 専用ハードウェアを用いた可視化システム

専用ハードウェアによる可視化システムは，$N \times N \times L$ のサブボリューム単位で並列化し，レイキャスティングアルゴリズムを用いて，1 ピクセル分のピクセル値計算をサブボリューム単位でパイプライン処理することで目標とする描画速度を得る．$N = 4096$ とした場合，可視化システムの構成としては 128 ノード構成の PC クラスタに，ボリュームレンダリング向けアクセラレータ (VisA)

を装備し，VisA 間を双方向高速リンクで接続する．計算結果は VisA 間リンクと同一規格のケーブルによりフレームバッファに送られ，ディスプレイへ表示される．生成された 2 次元画像に対して，さらに高度な後処理が必要な場合は，ホスト PC に送り処理を行う．図 3 に VisA の構成の概略図を示す．

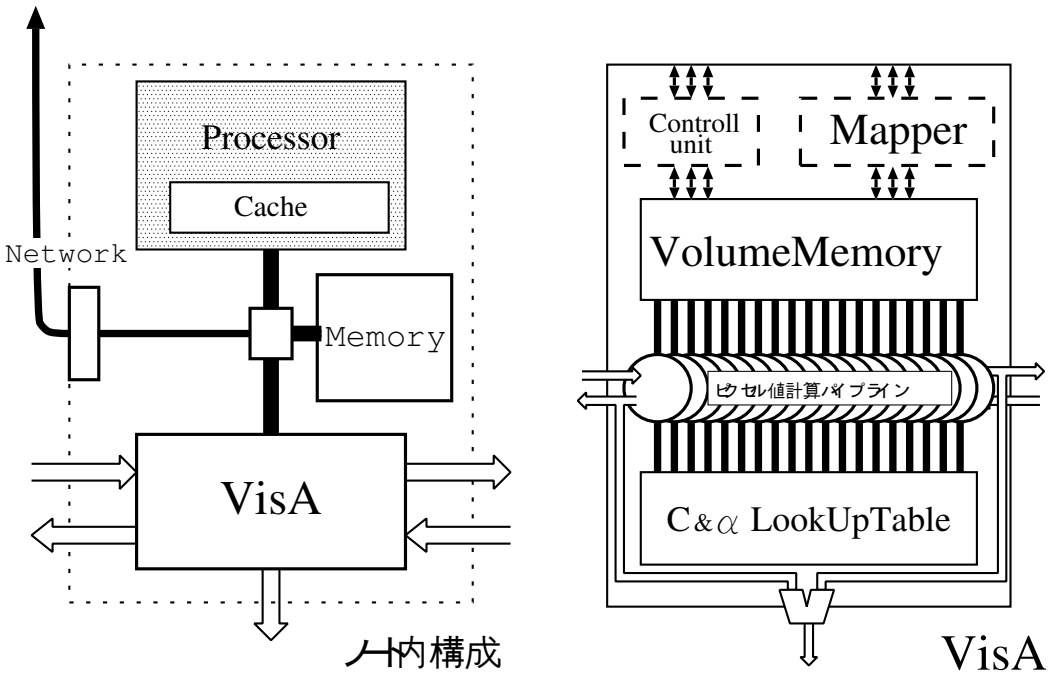

図 3: VisA の構成の概略図

ボリュームレンダリングでは，視線生成，ピクセル値計算，シェーディングの 3 ステージでの処理が必要となるが，VisA ではピクセル値計算のみを重点的に専用ハードウェア化し，その他のステージで必要であった処理は各 PC の CPU や汎用グラフィクスカードを用いて実行する．以下，VisA の主要構成要素について簡単に説明を行う．

- PC クラスタ部

  128 ノードの PC クラスタ部は，視線生成処理を行うとともに，シミュレーションの母体となる．ここで生成されたボリュームデータは PCI バスを経由して，それぞれ各ノードに装備された VisA に送られる．
- VisA 間リンク

  このリンクはレンダリングパイプラインにおいて計算途中の色情報，透明度情報，Z 値を送るためのリンクであり，フレームバッファの入力側スルー

プットと同様約 1GB/s の転送速度が必要である．各ノードに隣接ノード間の双方向リンクと，フレームバッファへの出力ポートを設けることで，種々のレンダリングアルゴリズムに対応する．具体的には DVI 用 LVDS インタフェースを用いてネットワークを構成する．この部分の詳細は 2.3.3 節で述べる．

- ボリュームメモリ
  ボリュームデータを格納するための容量 2GB のメモリで $4000^2 \times 32$ のサブボリュームを 4 セットまで格納可能とする．ノード内のピクセル値計算ユニットに対して 4GB/s のバンド幅を確保するため，PC600 規格相当の Rambus chanel 4 チャネル (合計 4.8GB/s) で構成する．
- ピクセル値計算パイプライン (PCU)
  32 個の PCU をパイプラインに接続して構成する．表示に必要な RGB は 8bit であるが，誤差伝播の影響を軽減するため 16bit 固定小数点として色と透明度の演算を行う．パイプライン周波数は画面サイズ ($2048^2 pixel$) とフレームレート ($30 frame/second$) から, 以下のようになる．

$$2048^2(pixel) \times 30(frame/second) = 128MHz$$

- Look Up Table (C& α LUT)
  RGB 各 8bit の色情報と，8bit の透過率を保持する 256 エントリのルックアップテーブルである．各ピクセル値計算ユニットが 1 ボクセルの演算を行う度に 1 回参照されるため，スループット的にはボリュームメモリの 4 倍の性能が要求されるが，小容量のメモリであるため，マルチポート RAM を複数用いて実装する．
- 制御ユニットおよび Mapper
  制御ユニットは視線情報を始めとする描画に関する情報を CPU から受け取り，VisA 全体の制御を行う．Mapper は CPU 側からのボリュームデータを受け取り，ボリュームメモリに格納する．シミュレーション結果からのボリュームデータへのマッピング処理が定型的かつ簡易なものであれば，CPU 側でのマッピング処理を省略し，Mapper に直接マッピング処理を行わせることで高速化を図ることができる．この目的のために Mapper には FPGA 的な機能を持たせる．
- プリフェッチ機構

ボリュームメモリはパイプライン構成している全てのピクセル値計算ユニットから同時にアクセスされるため，所望のパイプライン周波数を実現するために，LUT とボリュームメモリの間にプリフェッチバッファを装備し，ボリュームメモリへのアクセス遅延に伴うパイプラインストールを最小化する．

2.3 開発対象 FPGA 搭載ボード

本ボードは東京エレクトロンデバイス (株) との共同企画で開発を行ったもので，高速大容量の DDR-SDRAM を 2 系統，DVI-D Dual Link の入出力チャネルを搭載した，高いメモリバンド幅を要求する並列画像処理向けの FPGA 搭載ボードである．

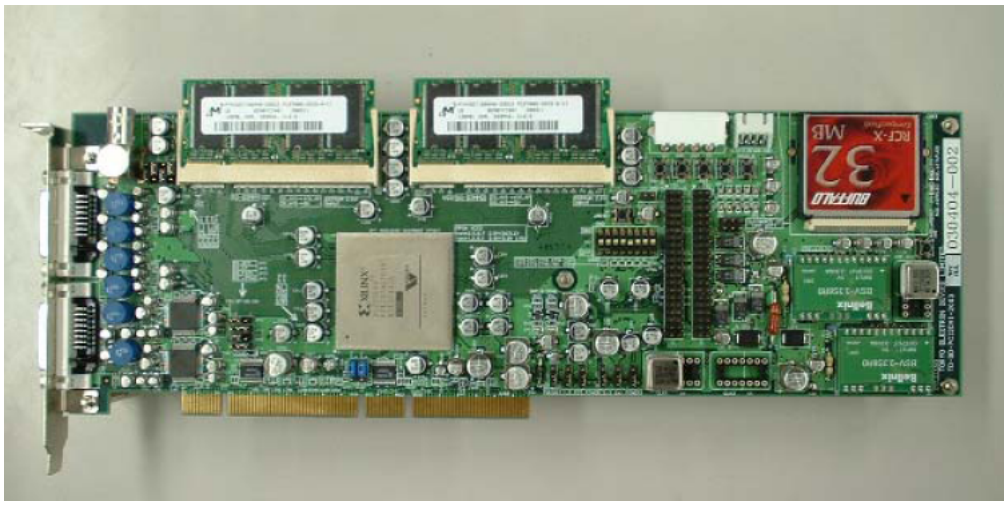

図 4: VisA Pro カード (写真提供 東京エレクトロンデバイス (株))

以下にその仕様の特徴を示し，特に本稿で扱う DVI-D Dual Link の入出力チャネルについては詳しく触れる．

2.3.1 高速大容量メモリ

DDR-SDRAM SO-DIMM を 2 枚搭載し，別系統クロックにて最大 DDR333MHz(PC2700 規格メモリ使用時) にて独立に動作可能である．各メモリスロットには，ノート PC 用に市販されているメモリ容量 1GB までの DDR SO-DIMM モジュールを搭載可能である．従って，128bit 幅の 2GB メモリ，あるいは，64bit 幅の 1GB メモリ 2 チャネル等のメモリ構成が可能である．最大メモリバンド幅は 2 チャ

ネル合計で 5.2GB/s に達する．

2.3.2 高性能大容量 FPGA

本ボードは，Xilinx 社の VirtexII シリーズ FPGA，XC2V6000-5(600 万ゲート) を搭載する．同 FPGA は，内部に 18bit × 18bit の高速乗算器と 18Kbit のオンチップメモリブロックをそれぞれ 144 個内蔵し，単体ではともに 100MHz 以上で動作可能である．これにより画像処理で必要とされる高い整数演算性能とメモリバンド幅を確保することができる．また，ソフトウェアプロセッサコアを内部に置くことでより柔軟な数値処理が可能となる．

2.3.3 DVI-D Dual Link の入出力チャネル

高解像度液晶ディスプレイに対応した DVI-D Dual Link の入出力チャネルを持つ．これにより，本ボードで生成した画像データを直接ディスプレイに出力することができる．またはグラフィクスカードのディジタル出力を一旦取り込んで加工したのちに出力すること等も可能である．

DVI-D インタフェースでは，高速データ転送に伴うクロックスキューを防ぐために差動信号である LVDS(Low Voltage Different Signaling) を用いて転送する．パラレルデータをシリアルデータに，またはその逆の変換するインタフェースである SerDes(Serializer/Deserializer) としては，Texas Instruments 社の TFP401/TFP410 を各 1 セットずつ搭載している．この SerDes チップを用いての通信のブロック図を図 5 に示す [4]，[5].

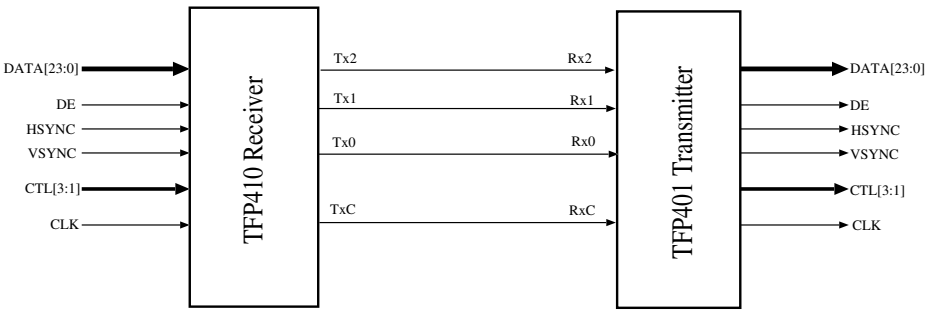

図 5: SerDes によるデータ通信

8bit の Blue のデータ，同期信号である HSYNC，VSYNC を 1 配線に，8bit の Green のデータとコントロール線である Ctl1 を 1 配線に，8bit の Red のデータと Ctl2，Ctl3 を 1 配線にそれぞれ割り当て，シリアルデータに変換している．このシリアル転送を行うために，パラレル転送での基本周波数 (VisA Pro カー

ドの場合は，FPGA が内蔵するクロックの周波数) を PLL(Phase Lock Loop) を用いて逓倍し，この逓倍した周波数でシリアル化したデータを送信している．受信側では，同様に逓倍したサンプリング周期でシリアルデータを取り込み，パラレルデータに変換する．この 3 配線に CLOCK を加えた 4 配線でデータ通信を行う．

Dual Link 構成時には入出力それぞれ伝送路の最大動作周波数である 165MHz で，24bit のパラレルデータ線により以下の最大転送速度を実現可能である．

$$165M(cycle) \times 24(bit/cycle) \times 2(dual) = 7.92Gbps$$

また，Single Link 構成時には最大 $3.96Gbps$ の転送速度を実現可能である．

この入出力チャネルを並列処理のためのネットワークとして使用することを目指し，本稿ではデータ転送を提案する．

2.3.4 その他の特徴

- PCI インタフェース：ホスト PC とのインタフェースとして PCI64/66 並びに PCI32/33 に対応している．
- SSRAM：166MHz 動作の 512KB(128K × 36bit)SSRAM を搭載している．ZBT(Zero Bus Turnaround) タイプなので，Read/Write アクセス切替えの Idle サイクルが不要である．

2.4 構成概略

VisA Pro は VisA の機能を約半分に縮小したもので，基本的な動作は VisA と同様である．図 6 に VisA Pro の概略構成を示す.

一枚の VisA Pro カードには，ピクセル値計算パイプラインの 1 ステージを担当するピクセル値計算ユニット (PCU) を 16 段実装し，連続する 8 つの PCU がボリュームメモリ (VisA Pro カードでは 2 バンク構成) の 1 バンクを共有する．隣接する VisA Pro 間は DVI-D Dual Link ケーブルによりリング接続する．

2.5 関連研究

近年，画像処理の並列処理としては，ATI 社製のマルチ GPU プラットフォームである CrossFire や，同じくマルチ GPU 構成に対応するためのアーキテクチャである nVIDIA 社製の SLI がある．両者ともに，2 台の GPU により画像処理を分散して行うことで，高解像度の画像処理，高速な画像処理を達成して

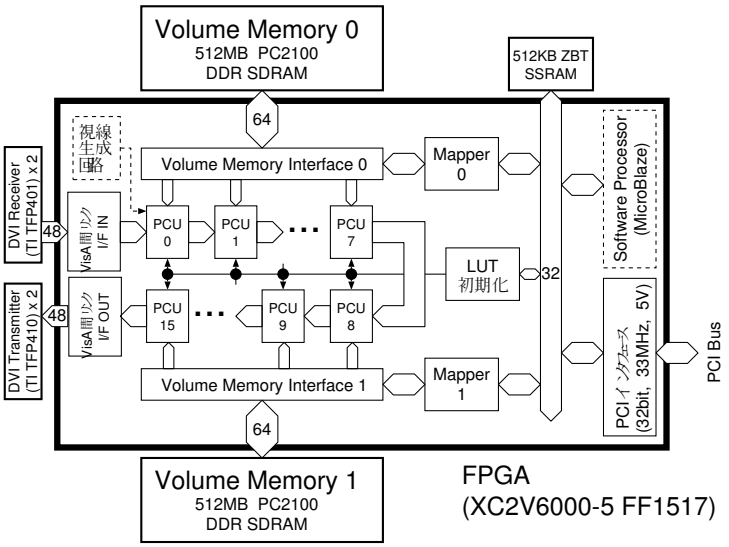

図 6: VisA Pro カードのブロック図

いる．

これらは共に，2 台の GPU を用いることによって，フレーム単位での処理分散を行っている．画像合成としては，本稿と同様に，DVI インタフェースを使用して行っている．

ATI 社製の CrossFire，nVIDIA 社製の SLI が 2 台の GPU による並列処理を行っていることに対して，VisA では 128 ノードによる規模で並列処理を提案するものである．よって並列化のノードスケーラビリティという点で，優位であると言える．

また，DVI インタフェースを用いてデータ転送を行い，画像合成を行う際にも，本稿では第 3 章で述べる手法により，ピクセル単位での転送を想定している．一方，CrossFire，SLI ではフレーム単位での転送となっている．この点で，本稿では粒度の細かいデータ転送を実現することで，低レイテンシでの高速データ転送を達成している．

第 3 章　DVI-D インタフェースを利用したデータ転送の提案

DVI-D インタフェースは本来，ディスプレイへのディジタル接続用のインタフェースである．そのインタフェースの RGB 各 8bit のデータ線を，24bit のパ

ラレルデータの通信路と見なし，高速なデータ転送を提案する．

まず本来の DVI-D インタフェースのデータ転送の手法について述べ，その後本提案による新たなデータ転送手法を示す．またその提案による利点・欠点を各ポイントにおいて述べる．

3.1 DVI-D インタフェース本来のデータ転送手法

DVI-D インタフェースは本来の動作としては，ビデオメモリ上に用意された表示データをある決まったタイミングに従ってディスプレイに送り出すインタフェースである [6].

ディスプレイへの画素の描画の様子を図 7 に示す．画素の描画は左上から順に右端まで終了したら，帰線処理を行い，次のスキャンラインの左端から描画を始める．最後のスキャンラインの右端まで描画が終わると，次の画面の左上に戻り，その動作を繰り返す．1 ライン分の水平方向軸に対しての復帰のタイミングを示す水平同期信号 (HSYNC)，同様に 1 画面の垂直方向軸に対しての復帰のタイミングを示す垂直同期信号 (VSYNC)，この 2 つの同期信号を画素データを転送する間に転送しなければならない．

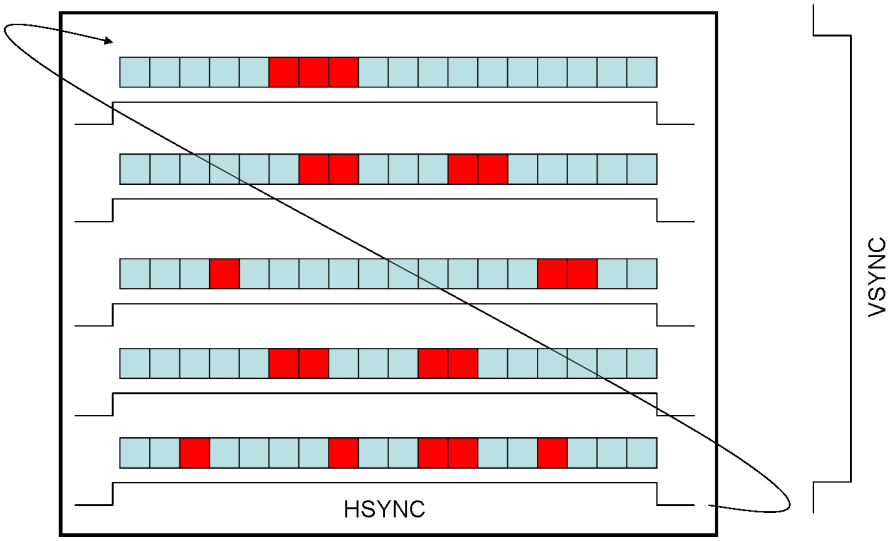

図 7: 液晶への画像の描画

同期信号を転送する間，RGB 各 8bit による 24bit の画素データは無表示区間として，データ転送をストップする．また再びディスプレイに描画し始めると

きに，HSYNC の直後の描画であれば左側の画像が，VSYNC の直後の描画であれば最上段の数ラインの画像が乱れる．データの乱れを防ぐために，それぞれの同期信号の無効化の直後に若干の無表示区間 (ブランキング期間) をいれておく必要がある．

よって実際に描画が行われるときの画面への画素データ・同期信号・ブランキング期間の転送は図 8 に示すようなものとなる．

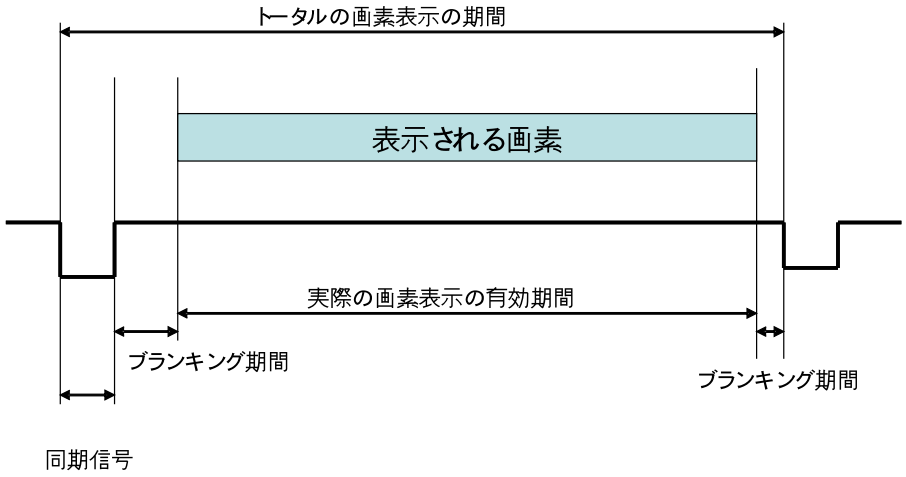

図 8: 画素データ・同期信号・ブランキング期間

具体例として UXGA(1600 × 1200) で，リフレッシュレート (1 秒間に何画面切替えるのか) を 60Hz と考えての，CRT ディスプレイ・LCD ディスプレイへの表示の場合の画素データ・同期信号・ブランキング期間の水平・垂直それぞれは，VESA 規格により規定されている．それぞれを画素数，またはスキャンラインで表すと表 1 のようになる．

この場合，実際の画像データ帯域としては，

$$1600 \times 1200(pixel/frame) \times 60(frame/sec) \times 24(bit/pixel) = 2.76Gbps$$

という帯域幅のデータ転送を行っていることになる．

今，同期期間やブランキング期間を考えずに，60fps でデータを送り続けることが出来たと仮定すると，CRT ディスプレイ・LCD ディスプレイに表示する場合，トータルでのデータ転送の帯域幅は，表 1 より以下の計算で得られる．

$$2160 \times 1245(pixel/frame) \times 60(frame/sec) \times 24(bit/pixel) = 3.87Gbps$$

表 1: UXGA(1600 × 1200) の表示，ブランキング期間

UXGA(1600 × 1200)	CRT	LCD
水平トータル画素数	2160(pixel)	1760(pixel)
水平表示画素数	1600(pixel)	1600(pixel)
水平同期信号幅	168(pixel)	32(pixel)
水平ブランキング画素数	392(pixel)	128(pixel)
垂直トータル画素数	1245(pixel)	1245(pixel)
垂直表示画素数	1200(pixel)	1200(pixel)
垂直同期信号期間	4(scanline)	4 (scanline)
垂直ブランキング期間	41(scanline)	41(scanline)

$$1760 \times 1235(pixel/frame) \times 60(frame/sec) \times 24(bit/pixel) = 3.13Gbps$$

よって，それぞれ 2.76*Gbps* に対して，1.40 倍・1.14 倍の高速化が可能である．

この性質を利用して，従来の DVI-D インタフェースよりも，さらに高速なデータ転送を行うというのが，本提案の趣旨である．

3.2 新たなデータ転送手法

新たなデータ転送の手法としては，24bit のパラレルデータを RGB 信号線に載せて高速データ転送するということを考える．本提案手法の適用を考えている並列ボリュームレンダリングシステムでは，それぞれのノードでサブボリュームをレンダリングしてできた中間画像データ (以下，サブイメージと呼ぶ) を次のノードに転送し，前後関係を考慮してそのノードでレンダリングされたサブイメージと再合成する．中間画像合成の過程においては，ディスプレイに表示するわけではないため，水平同期信号 (HSYNC) は，転送する必要はなく，それに伴うブランキング期間なども削減することが可能である．さらに垂直同期信号 (VSYNC) に関しても，単に 2 つの画像間の識別が可能な最低限の同期情報があればよいので，同期に関する不必要なマージンを一切排除した，効率の高いデータ転送帯域幅を確保することを期待したアーキテクチャを提案することができる (図 9，図 10 参照)． 提案する回路としては図 11 のようなブロック図のものを考える．

受信側のボードからのデータとクロックを，SerDes チップより受け取り，供

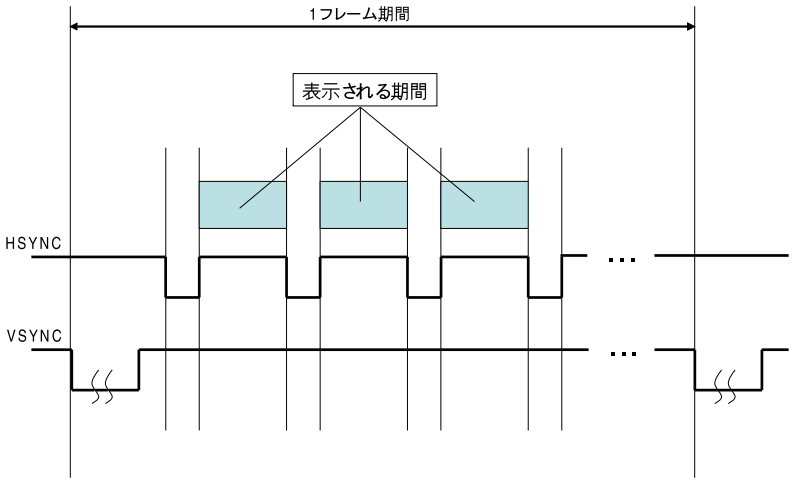

図 9: ディスプレイ表示対応のデータ転送

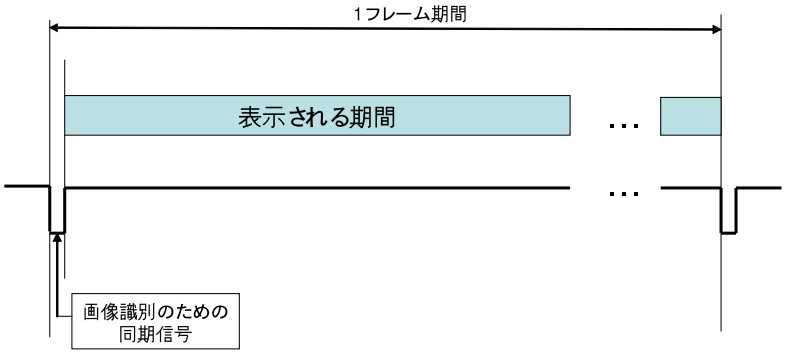

図 10: 提案する方式によるデータ転送

給されるクロックに同期して FIFO に格納する．その後，今度はボード上の基本クロックに同期してデータを FIFO から取り出し，画像データを加工，再合成などの操作をした後，送信用の FIFO に格納する．送信のためのクロックに同期してデータを取り出し，SerDes チップにデータとクロックを引き渡し，送信側のボードへ転送する．

3.3 本提案による実装の利点・欠点

本提案によるデータ転送のアーキテクチャの利点を他のデータ通信と比較した形で，以下に述べる．

3.3.1 高速性

DVI-D インタフェースを用いて，3.1 節で述べた帯域幅での高速データ転送が可能である．さらに同期信号やブランキング期間をデータ転送の始めに REQ

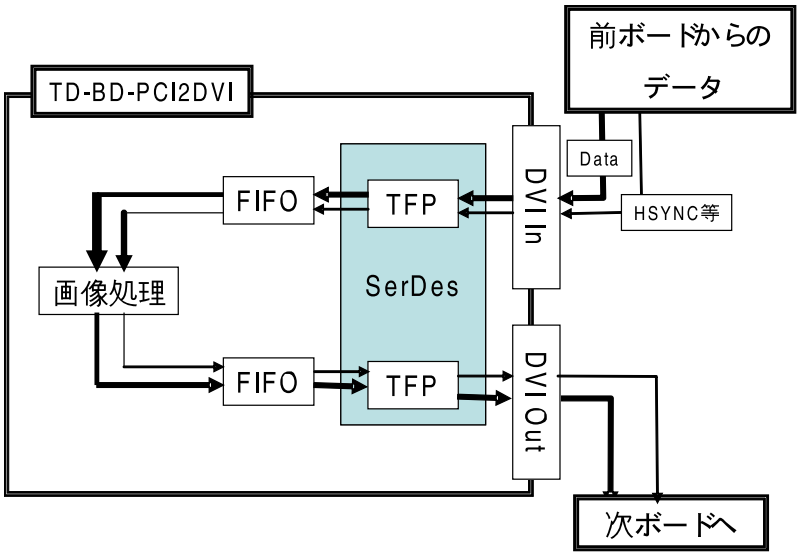

図 11: 提案する回路

として送る以外は転送しないため，その分のオーバヘッドを省いた帯域幅でのデータ転送が可能になる．

一方 Myrinet や Infiniband などの 10$Gbps$ 以上の高速，低レイテンシデータ転送を実現する技術があるが，これらはいずれも高価なものである．それに対して DVI-D インタフェースは比較的安価ながら，本稿で提案する手法に適応する性能を満たしている点で，優れていると言える．

3.3.2 低遅延

ノード間を Ethernet で通信する汎用 PC クラスタでの並列処理と比較した場合，そのプロトコルは TCP/IP であり，通信がパケット単位であるため，μs オーダのレイテンシでのデータ転送であることに対して，本提案のデータ転送ではそのプロトコルの複雑さからのオーバーヘッドを簡略化により解消でき，ns オーダのレイテンシでのデータ転送が可能である．よって粒度の細かいピクセル単位でのデータ転送が行えるという優位性を示せる．並列ボリュームレンダリングにおいて，本稿での提案では，粒度の細かいデータ転送を行えるため，並列ボリュームレンダリングのパイプライン処理を RGB，24bit のピクセル単位で行うことを可能にする．

3.3.3 応用性

FPGA で実装することで，データ転送の手法を FPGA を再構成することにより，容易に再構成することが可能である．また他の FPGA により構成されたレ

ンダリング回路などとの互換の可能性を考えることができる．

3.4　欠点

主な用途を画像処理に限定するために，単方向通信である点で，データの確実な到着を保証しない，Unreliable 通信路であることを許容する．よって汎用の通信インタフェースとは異なり，一般の並列/分散処理のための通信インタフェースとしての利用は困難である．

第 4 章　予備実装とその評価

前章で提案した回路の実装により，高速データ転送が可能であることを，2.3 節で述べた FPGA 搭載ボードで行い確認する．また，その実装過程で新たに生じた問題点や，その解決法について述べる．さらに，実装により達成された転送のレイテンシや帯域幅について評価を行う．

4.1　連続の大容量データ転送

前述の通り，本来 DVI-D インタフェースは，ディスプレイへのディジタル接続用のインタフェースであり，1 スキャンライン分の連続データまでしか想定されていない．

例として UXGA(1600 × 1200) であれば，1 スキャンライン分の連続データとは，1600 画素分で以下のデータ量となる．

$$1600(pixel) \times 24(bit) = 38.4kbit$$

そこで，この値を大きく越える大容量の連続データを転送した場合の回路の耐性を評価する．

並列ボリュームレンダリングにおいては，RGB 各 8bit の計 24bit のデータ以外に透明度を表すアルファ値や前後関係を表す Z 値等も転送する必要がある．また上記の UXGA の例よりさらに大容量のボリュームデータから大規模なスクリーンサイズへのボリュームレンダリングに対応するサブイメージデータを転送する場合を想定する．我々は最終的には 2048^2 のスクリーンに毎秒 30 フレーム出力することを目標としている．その場合にサブイメージデータも最大で同様のサイズとなる．その場合 RGB 各 8bit，加え透明度を表す 8bit の α 値を考

ると，以下の通信帯域が必要となる．

$$2048^2(pixel/frame) \times 30(frame/seconf) \times 32(bit) = 4.03Gbps$$

この値よりも大幅に容量の大きい連続データを転送することで実験を行った．転送したデータは，$1Gpixel$ 分のデータである．

転送するデータは全て同一のデータで，24bit 全てが 1 というデータである．データ量としては，以下のものとなる．

$$1G(pixel) \times 24(bit) = 24Gbit$$

このデータを DVI-D インタフェースの伝送路の最大動作周波数は 165MHz 以上である，VisA Pro カードに内蔵されているクロック，166MHz でのデータ転送を行った．その結果，24bit のパラレルデータ (DATA)，及び DE(Data Enable) がそれぞれブランクなく，1G サイクルに渡って受信できたことで確認した．しかしこれらは 4.2.2 節で詳細に述べるが，所望のタイミングでの正しいデータ転送ではなかった．ここで示されたことは，1G サイクルに渡って途絶えずに DATA，DE を連続に転送可能ということである．

4.2 ランダムデータの正確な高速データ転送

4.1 節の実装は，連続にデータ転送が可能であることを示したが，正確なタイミングで転送が行えることを示したわけではない．次に考える実装は，ランダムなデータを正確なタイミングで転送することを実験したものである．

図 12 は実装した回路のモジュール構成を示したブロック図である．

幅 24bit，深さ 8 の ROM を用意して，ランダムな 8 個のデータを持たせておく．このデータを順にトランスミッタチップ (Transmitter) に引き渡して転送する．

このとき供給するクロックとしては，本ボードに内蔵されている 166MHz で動作するクロックを利用する．クロックは直接他のモジュールの入力として使用せず，一旦 CLK-Generator と呼ばれるモジュールへ引き渡され，その中の DCM(Digital Clock Manager) を介して使用する．また受信側で受け取るクロックに関しても DCM を介して使用するものとした．

DCM では完全にデジタル化された DLL(Delay Locked Loop) 回路であり，クロックの位相と周波数を高精度で確実に制御できる．主なクロック管理機能と

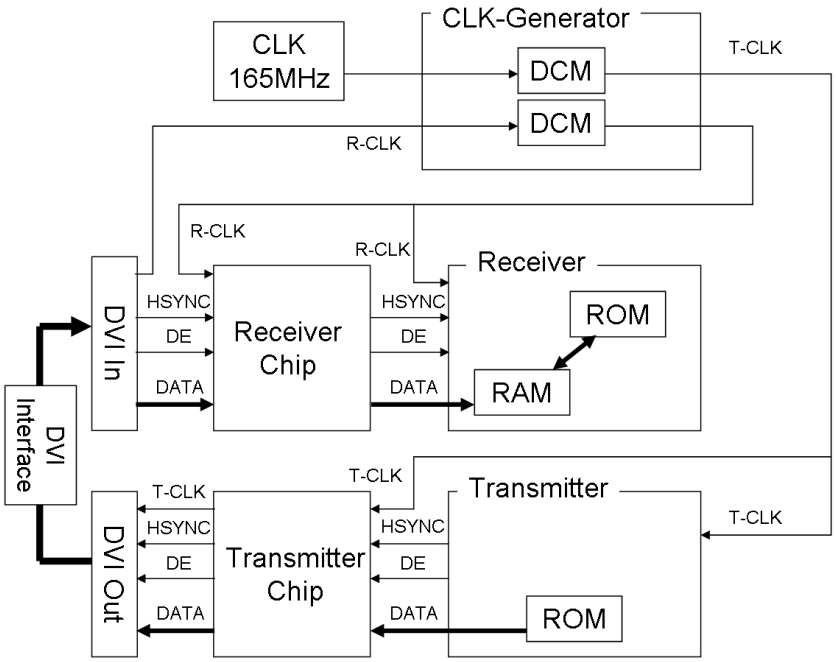

図 12: ランダムデータの正確な転送を実験する回路のブロック図

しては以下のようなものがある.

- クロックスキューの調節
  入力クロックに位相を合わせた新しいシステムクロックを FPGA の内部, または外部に供給する.
- 周波数合成
  クロックを乗算および除算することにより，広範囲の出力クロック周波数を生成する.
- 位相シフト
  設定の変更が可能な位相シフトにより，大幅または微細に位相をシフトできる.
- EMI の削減
  出力クロックの周波数スペクトルを広げることにより，電磁波による障害 (EMI) を削減する機能を備えている.

本実装は本来別のボードへのデータ転送を想定しての回路であるが，今回は予備実装として，1 ノードでデータの送受信をループさせて行った．受信したデータはレシーバチップ (Receiver) を介して，受信クロックと供に受け取り，幅

24bit，深さ 8 の RAM に書き込まれる．その RAM のデータが全て正しいか，前述と同様の ROM を用意してデータを比較することで確認する．

4.2.1 単純な手法によるデータ転送

まずデータ転送のスイッチが入ると 24bit のパラレルデータ (DATA) と DE を同時に 8 サイクル転送することを試みた．受信側での RAM の制御としては，DATA と同時に送信される DE を WE(Write Enable) の入力とした．

結果としては，RAM に正確なデータは書き込まれなかった．これはボードへのコンフィギュレーション，またはリセットによる初期化の際に数十サイクル程度に渡って，いくらかの不定の DATA，DE がレシーバより受信されてしまうことで，その不定のデータを RAM に書き込んでしまうために起こるものであることを確認した．

4.2.2 HSYNC を REQ として利用してのデータ転送

上記の実装結果からデータ転送の開始を知らせる信号 (以下，この信号を REQ 信号と呼ぶ) が必要である．そこで本来のディスプレイへのデータ転送で使われていた水平同期信号である HSYNC を REQ として利用することを試みた．

また上記の実装の追加として，同期信号である HSYNC，VSYNC がボードの初期化の際に，DATA や DE のように不定値を出すことはないことが確認できた．つまり HSYNC，VSYNC は所望のタイミングで転送開始することが可能である．しかし，4.4 節で後述するが，本実装で使用したボードでは，HSYNC，VSYNC は安定していたため，REQ 信号として使うことは問題なかった．しかし，個々のボードによって HSYNC，VSYNC が不安定となるものが存在した．

HSYNC は 1 サイクル程度の短いサイクルでは，受信側に認識されなかった．3.1 節で述べた UXGA の例でもあるように，同期信号はある程度の連続期間，転送されることを想定されていることが考えられる．余裕を持たせて HSYNC を 4 サイクル，その後ブランキング期間として 4 サイクルをおいてから，DATA，DE を転送するという手法とした，(図 13 参照)．

ゲート遅延や，配線遅延を考慮するために，タイミング制約をかけ，それを満たす配線を自動配線により行った．その結果，166MHz 以上で動作する配線ができた．しかし結果としては，最大クロック周波数である 166MHz でのデータ転送では，正しいデータ転送を確認することができなかった．そこで DCM を用いて周波数を落として実装したところ，100MHz という周波数でしか正常な動作は確認できなかった．

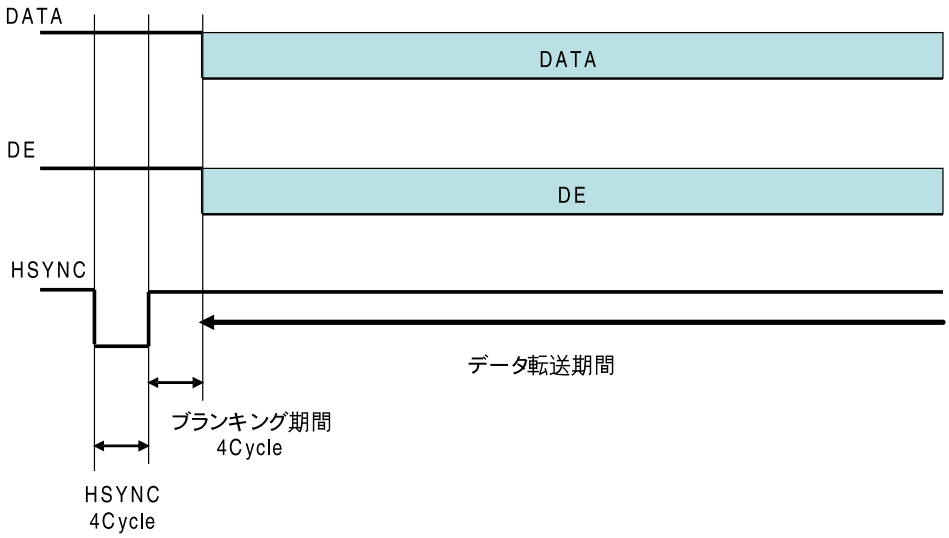

図 13: HSYNC を REQ として利用してのデータ転送

具体的には，133，166MHz とクロック周波数を大きくしていくと，データが正しく転送される確率が低くなるということが起こった．転送実験を繰り返すと正しくデータ転送ができている場合もあるが，8〜9 割以上の確率で正しい転送に失敗した．

さらに詳細に調査した結果，DATA と DE は同時に送信しているが，レシーバから受信した DATA と DE を比較すると，そのタイミングがずれてしまうということが起こった．そのずれのタイミングは，どちらかが 1，2 サイクル遅いということがランダムで起こっていた．よって受信されたデータが格納された RAM のデータには，1，2 サイクルずれたアドレスにデータが格納されていた．さらにそのずれは 24bit のパラレルデータで並列に一様ではなく，RGB，8bit 毎に生じていた．特に DATA[7:0] である Blue のデータが高い確率でずれを生じさせていた．図 14 にランダムなずれの生じる様子を示す．

DATA と DE のタイミングのずれは，データの転送，ブランク期間，ということを複数回繰り返すことで，それ以降はタイミングのずれを生じることなく，安定してデータを転送できることが分かった．

本実装を行う前に，通常の計算機からの画像データをディスプレイに表示する，または所望の範囲に所望の画像を表示するなどの実験を行ったが，DATA, DE のずれに伴う画像のずれや，乱れは生じていなかった．これは，ずれの影響は始めの 1 フレーム目の数行のみに起こることなので，実際のディスプレイ上

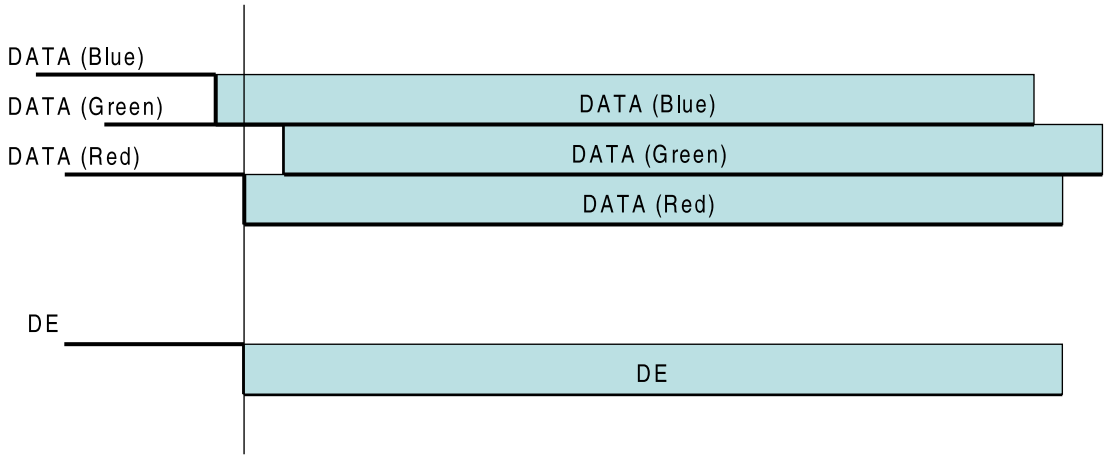

図 14: DE と DATA ずれの様子

には，目に見えては現れないためであったと考えられる．

4.2.3　データ転送開始直後の不確定性を回避するデータ転送プロトコル

4.2.2 節でのデータ転送開始直後の不確定性を回避するために，転送開始直後には，スタンバイ用のデータとブランク期間を複数回繰り返して行い，その後データ転送が安定してから，メインとなるデータ転送を行う手法とした．

安定したデータ転送を行うための，スタンバイデータの転送・ブランク期間のサイクル数・その繰り返し回数を決定するために，それらのパラメータを変えながら実験を行った．その結果，ブランク期間のサイクル数・スタンバイ用のデータの転送のサイクル数・ブランクとデータ転送を繰り返す回数を，安定してデータ転送できる限界まで削減し，最適化したものは，表 2 のようになることが分かった．具体的に，ブランク期間としては信号が何も出力されないブランクが 4 サイクルあり，スタンバイ用のデータとしては 24bit が全て 1 となるデータを転送している．

表 2: 最適化したスタンバイ操作のコスト

ブランク期間	4(cycle)
スタンバイ用のデータ転送	2(cycle)
ブランクと転送の繰り返し回数	2(回)

上記のスタンバイ操作を行った後に，HSYNC による REQ 信号を転送し，DE, DATA を同時に転送することで 166MHz のクロック周波数でデータ転送を安定

して行うことが可能となった.

HSYNC の転送のサイクル数の削減により最適化を計ったところ，4 サイクルの転送とブランク期間 1 サイクルを合わせた 5 サイクルが，最小のサイクル数であることを確認した.

よって最終的なデータ転送の手法としては，図 15 に示すようなものとなった.

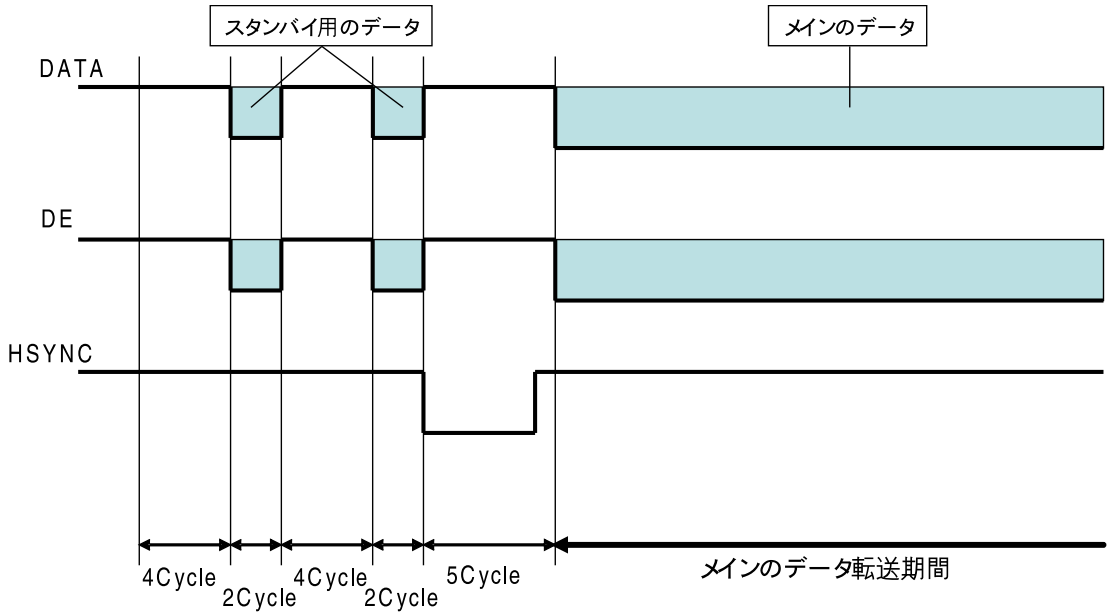

図 15: 最終的なデータ転送手法の様子

4.2.4 HSYNC 等の信号をヘッダーとして使用する拡張への考察

4.2.3 節で述べた，スタンバイや REQ としての HSYNC は，有効であることが認識されるというレベルのデータ転送であるため，正確に数サイクルに渡って転送する保証はされないものである．HSYNC 以外のコントロール信号である VSYNC・Ctl[3:1] を利用して，画像データのヘッダーとしての拡張が考えられるが，数サイクルに渡る複雑なデータを転送するということは不可能である．HSYNC・VSYNC・Ctl[3:1] の信号を用いて，5bit までのデータを転送することは可能であると思われるが，それ以上の情報を持たせたデータ転送を望む場合は，メインとなるデータの始めのサイクルで 24bit の DATA 線を利用して，別途転送する必要があると思われる.

4.2.5 大容量のデータ転送

4.2.3 節で述べた転送手法により，大容量データでの転送を試みた.

幅 24，深さ 8 の ROM を用意して，ランダムなデータを初期値として与えた．そのデータを 1000 回ループさせて転送し，幅 24，深さ 8000 の RAM を用意して受信したデータを格納し，正しくデータ転送されていることを確認する．ブロック図としては，図 12 とほぼ同じ構成である．

スタンバイにかかるサイクル数としては，表 2 に従って以下のようになる．

$$(4(branking) + 2(stanby)) \times 2(\text{回}) + 5(REQ) = 17cycle$$

上記のスタンバイ期間の後，8000 サイクルに渡って 166MHz の周波数でのデータ転送を安定して行うことを確認した．よってデータ転送の実効転送速度は，転送にかかった総サイクル数 8017 に対して有効なデータを転送したサイクル数が 8000 となるため，以下のようになる．

$$8000(cycle)/8017(cycle) \times 166M(Hz) \times 24(bit) = 3.97Gbps$$

また，転送開始から実際にデータ転送されるまでのレイテンシとしては，166MHz で 17cycle なので，以下のようになる．

$$17(cycle)/166M(Hz) = 102ns$$

さらに 4.1 節で述べたことと併せて，UXGA(1600×1200) や 2048^2 のディスプレイを想定した，さらに大容量データへの転送も安定して行うことが期待できる．

4.3 非同期 FIFO をバッファとして利用した実装

前述までの回路に，非同期 FIFO をバッファとして追加した回路を提案，実装した．モジュール構成のブロック図を図 16 に示す．

非同期 FIFO を加えることによって，まず Receiver からの DATA を受信クロック (R-CLK) の同期させて FIFO に溜める．FIFO からデータを取り出す際は，データ転送で使用したボード上に内蔵されているクロック (T-CLK) に同期させているが，他の所望のクロックを使用することも可能である．

このような実装によって FIFO がバッファの役割を担い，所望のタイミングでデータを取り出すことが可能となる．よって他の回路がそのデータを使用して処理を行う場合，その回路でのクロックに同期させて，データを扱うことができる．

本実装では，4.2.5 節で述べたものと同様のデータを送信し，受信したデータ

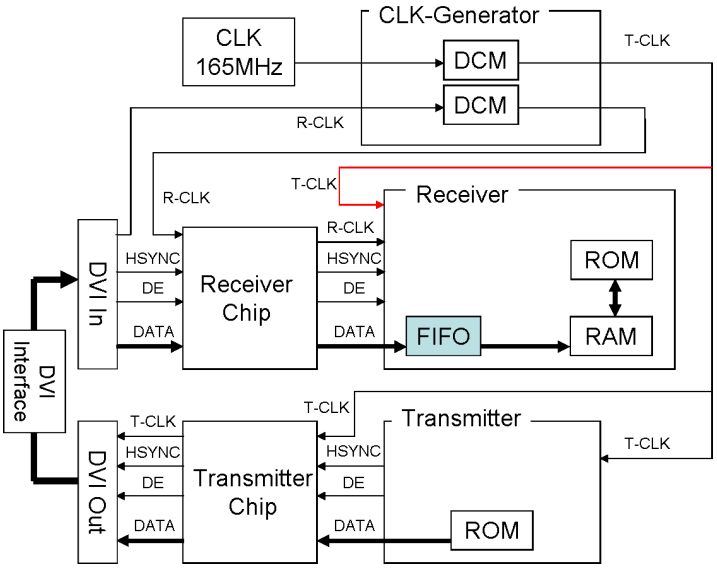

図 16: FIFO を加えたブロック図

を受信クロック (R-CLK) を同期させて FIFO に格納し，その後ボード上に内蔵されたクロック (T-CLK) に同期させて，データを取り出し RAM に格納し，正しくデータが転送されていることを確認する．

結果としては，正常にデータ転送が所望のタイミングで行われることを確認できた．

4.4 ノード数を 2 にしてのデータ転送

前述までのデータ転送実装は全て 1 ノードによるデータをループさせることでの転送実験であったが，ノード数を 2 として，一方のノードからデータ送信，他方のノードでは受信して，正しくデータ転送されることを確認する実装を行った．

その結果，いくらかの例外を除いては上記のデータ転送と同様に，正しくデータ転送できていることを確認できた．

この例外というのは，データ転送側のボードのコンフィギュレーション，またはリセットを行った場合，DVI-D インタフェースを介して，受信側のノードに REQ 信号としている HSYNC 信号がいくらかの不定のサイクルに渡って検知された．この不定な出力は前述までの 1 ノードでの実装では起こらなかったことである．HSYNC が不定な値を出力するということは，1 つのボード内でも確率的に生じることであり，個々のボードによって，その確率の傾向は異なるものであると考えられる．

4.2.1 節で述べた通り，ボードのコンフィギュレーション，またはリセットの際には，DATA，DE から不定の値が出力されてしまうために，HSYNC が同様に不定の値を出力することになれば，その値を受信してしまうことにより，正しいデータ転送が行えないということが起こった．

よって，図 17 に示すように，REQ 信号 (図 17 では HSYNC) がその初期状態から数サイクルに渡って不定な値が出力される可能性がある．それはボードによって個々に差がある可能性があるため，本実装では確認できていないが，他の信号である VSYNC, Ctl 信号を HSYNC の代わりに用いて REQ としても，ノード数を増やせば不定なデータを出力するノードが存在する可能性がある．

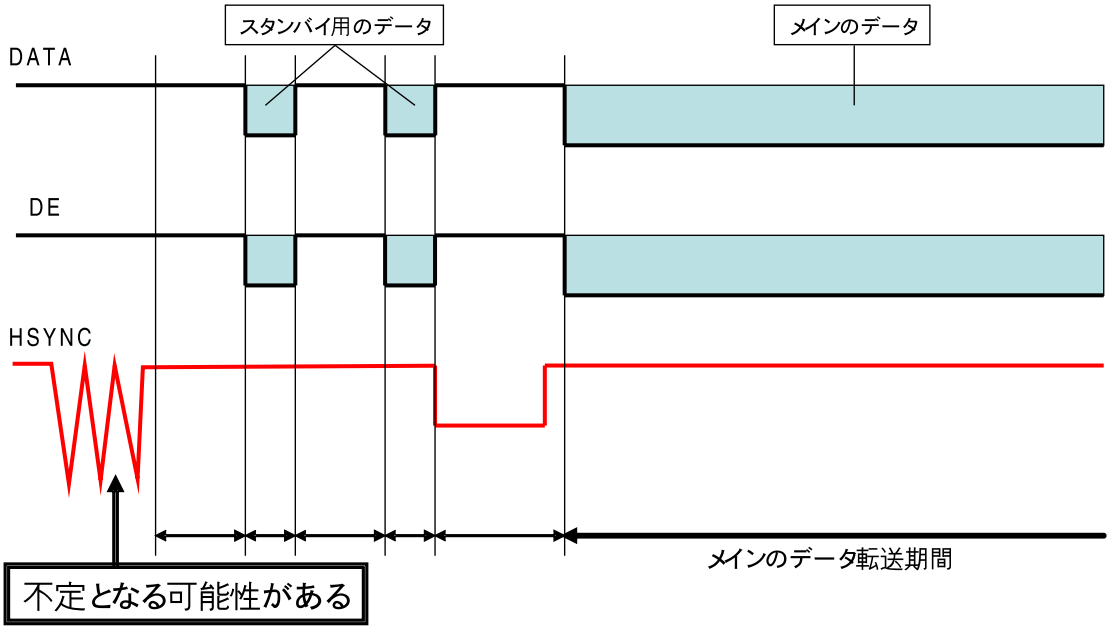

図 17: REQ 信号 (ここでは HSYNC) が不定となる可能性

一方で，HSYNC が不定の値を出力しない場合は，安定したデータ転送が行われることを確認できた．

4.5 起動直後での不確定なデータ出力に対する考察

4.4 節で述べた通り，ボードの個々の性質のばらつきにより，起動時に不確定な出力を確率的に生じると考えられる．しかし，これらは起動時のみのバグであるため，並列ボリュームレンダリングにこのデータ転送を適用した場合，そのバグの影響により，出力される初期段階の画像にバグを与える可能性はある

が，その後は安定してデータ転送を行えると考えられる．

第5章　まとめ

並列ボリュームレンダリングを行う手法として，本稿では専用ハードウェア VisA による処理について述べた．

そのプロトタイプである VisA Pro カードを用いて実装する場合，並列ボリュームレンダリングにおける，ノード間の中間画像データであるサブイメージデータをピクセル単位で合成するための，データ転送を FPGA 搭載ボードの DVI-D インタフェースを用いて行う手法について述べ，その予備実装を行った．

本実装により，DVI-D インタフェースを用いて，シングルリンク時の伝送路の最大通信帯域幅，3.96$Gbps$ 以上の帯域幅である 3.97$Gbps$ でのデータ転送が可能となり，大容量高速データ転送が可能であることを示した．またデータ転送に伴うブランキング期間の削減，プロトコルの最適化により，データ転送開始から，実際のデータが出力されるまで 102ns という，低レイテンシでのデータ転送が可能であることを示した．

しかし，ノード数を増大させてのデータ転送を想定すると，REQ 信号などが不確定要素となり得るため，安定したデータ転送が行えない可能性が出てくることが分かった．しかしこの不確定データは起動時のみに生じるものであるため，その後は安定させることは可能であると考えられる．よって，本稿で提案する並列ボリュームレンダリングに対応するデータ転送は安定して行うことができると考えられる．

本実装では，ランダムデータの転送を行っただけであるが，種々の並列ボリュームレンダリングの手法に合わせて，RGB データに加えて，アルファ値，その他ヘッダーなどを加えることについても，柔軟な対応が期待できる．

謝辞

本研究の機会を与えていただいた富田眞治教授に甚大なる謝意を表します．

本研究に関して適切なご指導を賜わった森眞一郎助教授，中島康彦助教授，嶋田創助手，三輪忍助手に心から感謝致します．

本研究に当たって，日頃から様々な助言，助力を下さり，ReVolver グループ，修士 2 回生である，岡村 大先輩，小松原 誠先輩， 篠本 雄基先輩，吉村 知晋先

輩に心から感謝致します.

また，同学であり，互いに助言，激励し合った学部 4 回生である，岩田 浩明, 木村 英雄，吉田 智一に感謝致します.

さらに日頃からご助力いただいた京都大学大学院情報学研究科通信情報システム専攻富田研究室の諸兄に心から感謝致します.

参考文献

[1] 生雲公高: 時変ボリュームデータの実時間処理のための専用グラフィックカード VisA の開発, 修士論文, 京都大学大学院情報学研究科修士課程通信情報システム専攻 (2003).

[2] 岡村大, 五島正裕, 森眞一郎, 中島康彦, 富田眞治: 並列可視化処理向け FPGA 搭載 PCI カードへのボリュームレンダリングの予備実装, 電子情報通信学会技術報告 CPSY2004–76, pp. 1–6 (2005).

[3] 村木茂, 緒方正人, 越塚健児, 梶原景範, 劉学振, 永野靖忠, 下川和郎, Ma, K.-L.: VG クラスタ: スケーラブルビジュアルコンピューティングシステム, Visual Computing グラフィックスと CAD 合同シンポジウム 2001 (2001).

[4] 小畑正貴: FPGA における差動信号入出力を用いた PC クラスタ用ネットワークインタフェース, 情報処理学会論文誌:コンピューティングシステム (2003).

[5] 西本桃子: 低コスト FPGA による 640Mbps LVDS インタフェースの実現, *Design Wave Magazine* (2005).

[6] 井倉将実: アナログ RGB ビデオ出力回路設計入門, *Interface* (2006).