Sepia: scalable 3D compositing using PCI Pamette

Laurent Moll[*] Alan Heirich[†] Mark Shand[*]

[*] Compaq Computer Corporation, Systems Research Center
Palo Alto, California
Laurent.Moll@acm.org, shand@acm.org

[†] Compaq Computer Corporation, Tandem Labs
Cupertino, California
Alan.Heirich@compaq.com

Abstract

We have implemented an image combining architecture that allows distributed rendering of a partitioned data set at interactive rates. The architecture achieves real-time frame rates and low latency through pipelining and the use of a high bandwidth network technology to transfer the image data. It is flexible because it uses programmable FPGA devices to implement the combining logic. The implementation cost is kept low by using only commodity components for the network and graphics, and FPGA logic. The result is a cost-effective interactive visualization system that can be used with a variety of applications running on distributed computing systems such as cluster of workstations and personal computers.

We first motivate the development of a distributed rendering system and we introduce some of the concepts related to the 3D-visualization domain.

We then describe our implementation of this system using the PCI Pamette FPGA-based board. We emphasize the advantages of using a programmable board for the prototype development and also for a potential commercial version.

1 Introduction

A variety of applications in engineering, science, and information processing, are computed in a timely fashion only through the use of high performance computing systems comprising multiple processors. Often these applications partition data sets among processors in such a way that each processor has access to and responsibility for a specific data subset.

In a conventional hardware-accelerated graphics architecture, data must be fed sequentially to a centralized rendering pipeline. In the applications we are interested in, the data subsets must therefore be recombined each time they are to be fed to a visualization engine, which creates a bottleneck. In practical terms this makes interactive visualization of large scale distributed applications unaffordable and prohibitively slow.

A graphics architecture that combines the images produced from individual data subsets would eliminate this bottleneck because it would require of each node only an image of the data subset for which it is responsible. Since this task is assumed to occur at interactive rates, such an architecture makes it possible to combine interactive simulation with visualization. An advantage of this image-based approach is that it can work with a variety of visualization techniques, including ray tracing, volume visualization, and traditional and nontraditional hardware accelerators [1, 2, 3]. Such a graphics architecture combines the high performance of farms of rendering machines and the interactivity allowed by the speed, the scalability and the low latency of the combining network.

2 Sepia architecture

2.1 Smart clusters

Typically in clusters, network adapters are used directly by software and are critical components that must be optimized both in the software run-time library and in the hardware itself for software accesses.

It is possible to embed in the network adapter either a processor [4] or programmable logic that can be used to optimize data transfers for an application, or that can process data by itself. We will call clusters using this technology

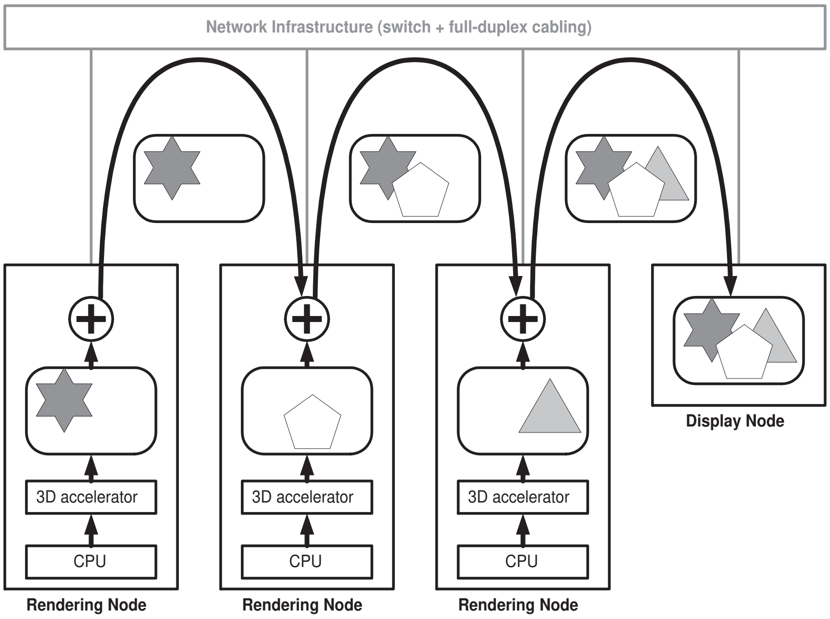

Figure 1: Compositing in a sort-last architecture.

smart clusters. They have many advantages over conventional clusters:

- CPU load is reduced, allowing the CPU to devote more time to other tasks (3D rendering in our case).
- Complex data flows are possible with small overhead.
- Control latency is negligible, as the hardware is directly connected to the network adapter and can react instantaneously.
- The data can flow freely and in pseudo-real-time without having to wait for software.

Our image combining architecture uses such technology in its compositing hardware, which consists of a programmable board directly controlling a standard network adapter.

The PCI Pamette reconfigurable board is well-suited to the task of implementing a smart cluster adapter. Its reconfigurability can be used to adapt the control hardware to the task of the client application. Its standards-based expansion capabilities (CMC/PMC connector) can be used to attach off-the-shelf or custom network adapters. It can be used as a bridge to the network adapter, but also as a direct master, managing data transfers and possibly processing some data.

ServerNet [5] is a network adapter technology developed by Compaq's Tandem division. The ServerNet ASICs have powerful and versatile communication modes that can handle any size of packets with extremely low processing and transmission latencies. They allow the controlling hardware to send data at very high speed and also provide low-latency, low-overhead, small packets transmission capabilities, which is essential for flow control. Off-the-shelf network solutions make the smart cluster inexpensive, fast and reliable as it leverages widely used and debugged technologies.

The latest ServerNet Colorado ASICs use a PCI interface and are therefore easy to interface to an FPGA with a PCI core.

Our compositing hardware is a smart cluster adapter programmed for a specific application, which not only manages the network flow, but also performs some computation on the data. It uses the powerful combination of the programmability and extensibility of the PCI Pamette with the speed and low-latency of ServerNet.

2.2 System components

Sepia (ServerNet Enhanced Parallel Image Accelerator) takes advantage of the combined low prices of modern workstations, 3D-accelerator boards and high-performance network adapters, and of the smart cluster technology to provide a scalable and inexpensive alternative to current high-end visualization systems.

A Sepia cluster consists of:

- Rendering nodes composed of a standard workstation, with typically an OpenGL 3D accelerator and a Sepia

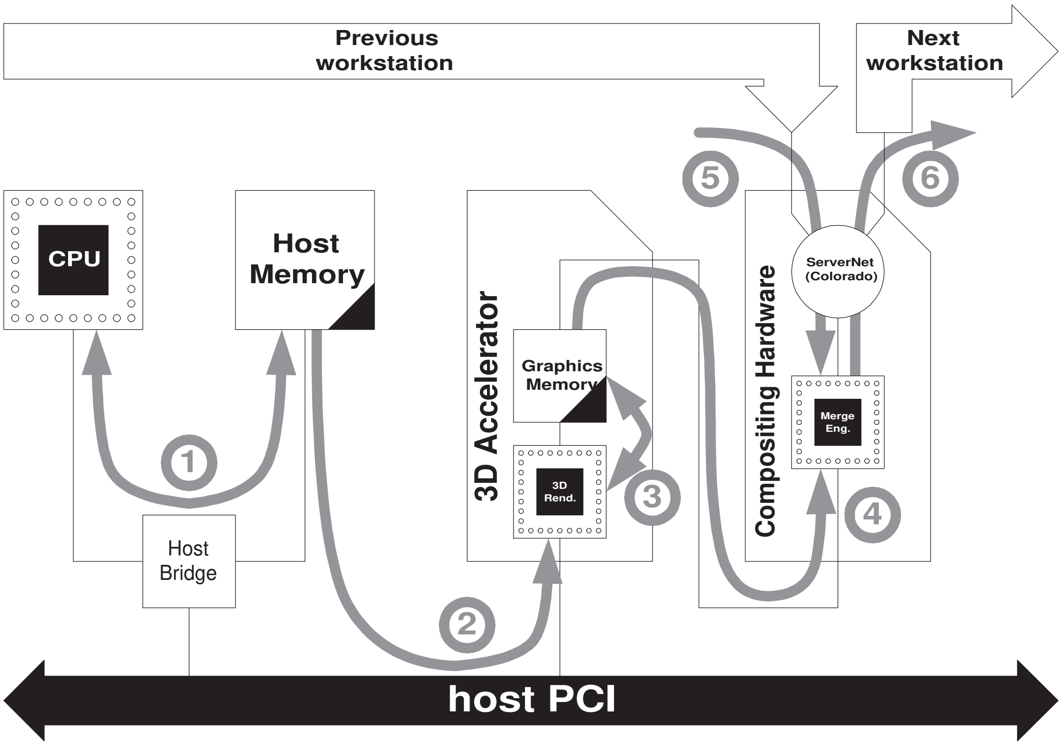

Figure 2: Data flow in a Sepia rendering node.

compositing board (which consists of a merge engine and a network adapter).

- Display nodes composed of a frame buffer (usually attached to a workstation) and a Sepia network adapter.
- The network infrastructure, usually composed of a set of switches to which the nodes are connected or at least of point-to-point connections between the nodes.

When a 3D image is to be computed, the task is spread on all the rendering nodes, each of them computing a subset of the problem. In its intended use, Sepia dispatches the 3D objects on the various rendering nodes and then gets the merge engines to perform standard OpenGL operations like depth comparison or alpha-blending between the partial frames computed by the local 3D accelerators, as shown in figure 1. According to a taxonomy introduced by Molnar et al [6] this is a "sort-last-full" architecture: full images, defined by sets of pixels with color, depth and α (transparency) buffers, are combined after completion of all rendering operations.

Another approach that is possible to implement in Sepia, called sort-middle, tiles the screen in zones and each rendering node renders all the 3D objects, but only on the sub-part of the screen that was attributed to it; then the merge engines put the tiles together to reconstruct the full screen to be sent to the display nodes [7]. The combining function uses the position of the pixel to compute which tile it is in and therefore which pixel color to choose. This tiling approach can also be used for parallel ray tracing.

Sepia is versatile enough to be used for any 3D or 2D work. The local graphics adapter is used as a frame buffer by the CPU and the Sepia merge engine is programmed to perform any task that is useful to the 2D or 3D merging. However, it is not a goal to off-load any rendering task from the 3D accelerator. To keep the price of Sepia low despite the medium volume of production, it is important to be careful with cost efficiency of the various pieces of the system. In the case of 3D rendering, the 3D accelerator will always be more cost-efficient than the FPGA for the functions supported by the hardware.

2.3 3D compositing

The standard datapath within each rendering node is shown in figure 2. Using its local or network data, the processor prepares 3D data and sends OpenGL commands (1)(2). Then the 3D accelerator executes the OpenGL commands and renders the 3D objects in the graphics memory (3). It is at this point that the Sepia compositing hardware is used. Usually, the color data is read from the video memory by a RAMDAC and sent to the screen. Instead, in Sepia, the color (RGB), transparency (α) and depth (Z) data is sent to the merge engine (4). The Sepia merge engine receives data in a scan-line fashion simultaneously from the local video memory (4) and from the upstream node through the Sepia network adapter (5). Using a user-configurable function, the Sepia merge engine combines local and upstream pixels and sends the resulting merged pixels to the downstream node through the network adapter (6).

A typical image combining operation performed in the merge engine is the *Z-comparison*. Z is the depth information associated with each pixel. It is used by the 3D accelerator to deal with hidden pixels. In the compositing network, this depth information can be reused to make the choice between the pixels of the two incoming flows. For example, the choice function could be:

$$\begin{aligned} f_z(a,b) \quad &= \quad a \text{ if } z(a) \leq z(b) \\ & \qquad b \text{ if } z(b) < z(a) \end{aligned}$$

Another popular combining operation is called α-*blending*. It consists in using the α transparency factor of the pixels to blend the colors of the pixels of the two source images to obtain a transparency effect. This typically involves scaling of the pixels color components by a factor related to the transparency components of the source pixels and merging of the scaled colors of the pixels.

2.4 Network characteristics

Besides the network infrastructure, a Sepia cluster consists of rendering nodes (n) and merge engines (n-1). Each rendering node has one output. Each merge engine has two inputs and one output. Even though it is possible to build and use a cluster with network-attached merge engines, it is more attractive to put a merge engine in each node. This way, one of the inputs of the merge engine is connected directly to the 3D-accelerator frame buffer. With an integrated merge engine, nodes need only one input and one output to the network. This lets us use a full-duplex link efficiently.

The logical topology that optimizes the network utilization is the chain: each node only has one input and one output data stream. Other topologies are possible. A ring can be implemented very easily by closing the two ends of a chain; it is a useful topology for applications where the data has to be recirculated in the rendering network. It is also possible to implement a tree but it is more costly in terms of network bandwidth, which is a scarce resource, and its only advantage is a lower latency, which Sepia already keeps very low even in a long chain of machines. Even though the data is flowing in a chain logically, it can be useful to connect all the network inputs and outputs to a switch. This way we can reorder arbitrarily the nodes in the chain. This can be important for merging operations that require a specific order that has to be changed at run-time (as the viewpoint changes for instance).

In Sepia, flow control is completely explicit. Within each node, the data flows with enables and FIFOs are used intensively to regulate the traffic. On the network side, a credit-based scheme is used: each time a new slot is free in a receive FIFO, a message is sent to the upstream node to indicate that a new packet of data can be sent. The ServerNet adapter can provide end-to-end flow control. However, the Sepia network also supports a low-bandwidth control traffic, which must reach its destination even if the data traffic is stalled. Using credit-based flow control, the data traffic is never stalled at the network level.

Sepia is pipelined as aggressively as possible. Ideally, the 3D-accelerator in the rendering nodes have double-buffered color and depth buffers, and the display node has a triple-buffered display. In this case, we obtain pipelining similar to the one shown in figure 3. On each node, rendering can start (with a double-buffer swap) as soon as both the previous rendering and merging are finished. Merging can also start at the same time. However, it will be constrained by the flow-control on the two inputs and one output (the stalls introduced by the flow control are represented in figure 3 by the cross-hashed areas). Thanks to the node-to-node flow-control, no broadcast is needed to stop or start the data flow. This way, even on a system with a high node count (128 or more), stalls stay local and the data flow resumes with a latency of one node. This way, the Sepia infrastructure is fully scalable.

2.5 Performance, bandwidth and latency

The ideal performance goal for a compositing network like Sepia would be to run at HDTV resolution (2.4Mpel) at 60 Hz. For standard 8 byte pixels (3 bytes for RGB the color information, 1 byte for α the transparency factor, and 3 to 4 bytes for Z, the depth information), this leads to a network bandwidth requirement of 1152 MB/s each way (incoming and outgoing).

In 1999, it is not realistic to expect this kind of data rate from inexpensive network components. There are however many ways to bridge the gap between what is available and what we would like.

Firstly, to get interactive rates, it is not necessary to produce new images at monitor refresh rates. Cinema and television successfully use frame rates of 24-30 Hz. Video games are considered to have a fluid animation when they have frame rates in excess of 20 Hz. We can therefore set a more realistic frame rate goal of 20-30 Hz.

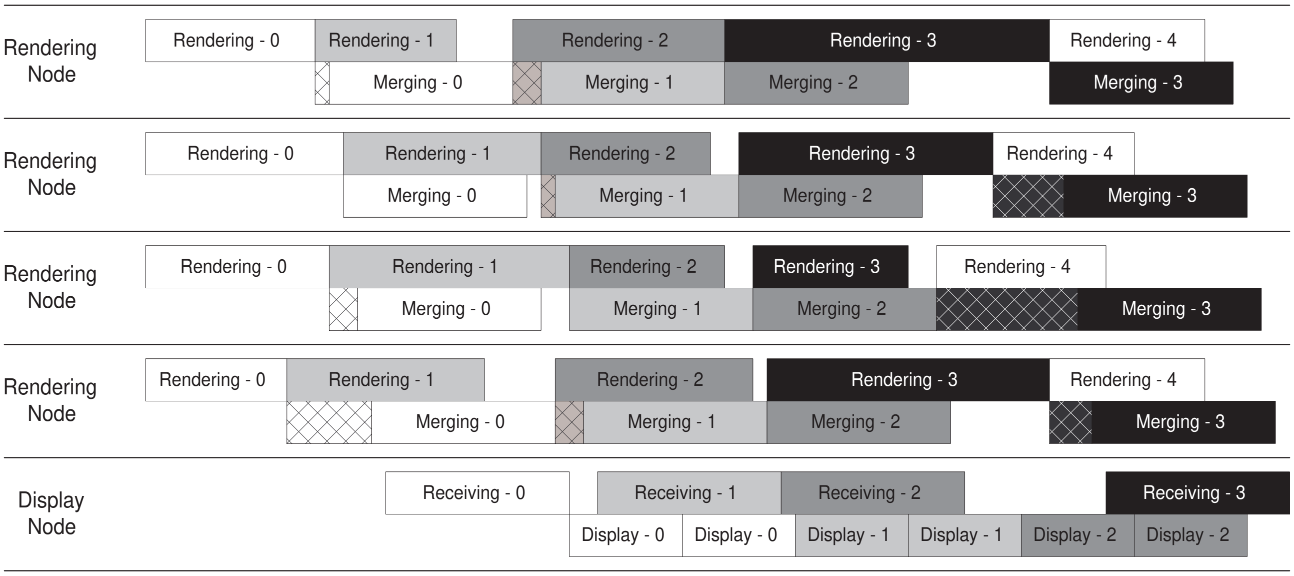

Figure 3: Pipelining in Sepia.

Secondly, we can add on-the-fly compression so that we can reduce the network bandwidth required. Given the high data rates, it is impossible to implement a complex compression algorithm. A simple compression algorithm has been designed for Sepia, using first or second-order differences and a simple entropy coding. From our experiments, this algorithm leads to compression ratios in excess of 1.5.

Thirdly, not all compositing algorithms use RGBαZ. Our sort-last version typically uses only RGBZ with a 24-bit Z, so the pixels are transmitted in 6 bytes per pixel form. Sort-middle algorithms use only color information: RGB, so the pixel size is only 3 bytes in that case.

Finally, if the bandwidth provided by the network is still too low, it is possible to cut the screen into different parts (in a sort-middle fashion) and render the different portions on different pipelines, with either one screen per pipeline at the end or a special combining frame buffer. This solution is a step backward for sort-last algorithms as it involves duplicating data sets and rendering, but it might be the only solution to get the required frame rate.

In summary, for a frame size of 2.4 Mpel, a frame rate of 30 Hz, 6 bytes per pixel and a compression ratio of 1.5, we would need a network bandwidth of 288 MB/s (in each direction with switching capabilities), which is close to what is available today. For smaller frame sizes than HDTV, like 1280×1024, we would need a network bandwidth of 155 MB/s full duplex, which will be available this year.

The data flow architecture of Sepia is designed to minimize the full system latency. This is especially critical on large clusters where the data flows through 100 or 200 nodes. The Sepia adapters use only node-to-node flow control and the link length and packet size are small enough to ensure a maximum latency of 50 μs per node, so that the total cluster latency due to the network is well below a frame time. We see in figure 3 that, compared to a standard rendering machine, Sepia has the extra phase of merging, which has a duration comparable to that of the rendering phase. So, even on large Sepia clusters, the total system latency is small.

3 Sepia (Phase 1) implementation on PCI Pamette V1

3.1 General Architecture

At this time (April 1999), Sepia has been implemented as a prototype using mostly off-the-shelf boards and systems. A Phase 1 Sepia node is composed of:

- A Compaq workstation (IA32 or Alpha based, running Microsoft Windows NT).
- A mid-range 3D accelerator such as the Compaq PowerStorm 300.
- A Compaq PCI Pamette (marketed as "PCI Development Platform") V1R2 using 4 user-programmable Xilinx XC4044XL-1 FPGAs [8].
- A custom Compaq ServerNet PMC daughter-card (for PCI Pamette) using the Colorado 1 PCI-to-ServerNet ASIC [5].

The ServerNet daughter-card is the only custom element in a Phase 1 Sepia node. Besides the Colorado 1 ASIC, it includes some SRAM and two physical link circuits for the two channels, connected to two full-duplex parallel connectors. One connects to the upstream node and the other connects to the downstream node. The daughter-card is plugged into the PMC (PCI Mezzanine Card) slot on the PCI Pamette and is used by the Sepia circuit downloaded in the Pamette FPGAs as a private network interface. The Pamette is plugged on the same PCI bus as the 3D accelerator. It extracts pixel data out of it by doing direct board-to-board DMA.

The complete diagram of the mapping of the Sepia merge engine and network adapter on PCI Pamette is shown in figure 4 and described in the next sections. This diagram applies to the rendering nodes only.

3.2 Interface to the local 3D accelerator

There are three ways to get data from the 3D accelerator frame buffer memory:

- Use the 3D accelerator's DMA engine to send the data directly to the Pamette by doing board-to-board DMA.
- Use the 3D accelerator's DMA engine to send the required data to main memory, then DMA the data from main memory into the Pamette.
- Use the Pamette DMA engine to get data directly from the 3D accelerator by doing board-to-board DMA.

The first solution is tempting, especially since the 3D accelerator's memory system is typically optimized to yield maximum performance in master mode (i.e. when its own DMA engine is used). However, it is not straightforward from the software side to program the 3D accelerator's DMA engine to target another PCI device rather than main memory. This solution also involves tricky flow-control problems between the two boards (given that the Pamette does not have enough storage for a full frame).

The second solution is a good tradeoff for simplicity, as it uses standard datapaths from the 3D accelerator to main memory and from memory to the Pamette.

The third solution is the best of all. It is easy to program the Pamette to DMA from another PCI device rather than memory (it is actually easier as there is no scatter/gather translation to do) and the data is transferred directly, so it is simpler and potentially faster. This is the solution that is implemented in Sepia Phase 1.

This solution has the limitation that it can not be used on an AGP 3D accelerator (PCI-to-AGP DMA reads are not implemented in Intel AGPSets), but this is not crucial in a prototype system.

To measure the performance of this datapath and tune the DMA engine, the *pciperf* tool was used [9]. This example application from the PCI Pamette software kit [8] can be used to exercise and trace any kind of transactions involving PCI traffic (including master-mode transactions using Pamette's programmable DMA engine). From the traces collected from an Evans&Sutherland RealImage 2100 based board such as the Compaq PowerStorm 300, the maximum direct transfer rates obtainable using Pamette's DMA engine are 34 MB/s reading the frame buffer and 57 MB/s writing to it. On other boards, we get performance as low as 8 MB/s for reads.

As shown in figure 4, the left-most FPGAs of the Pamette, both connected to the PCI Interface FPGA, use their local SRAM to store the incoming data. They are used alternatively in a scan line based double-buffer scheme. Knowing the bandwidth and maximum burst size that will be received, it is possible to implement the datapath with minimum buffering and flow-control.

Thanks to preliminary testing using existing tools to perform experiments on hardware configured in that same manner as the proposed system, it was easy to make early design decisions.

3.3 Interface to the network adapter

The interface to the Compaq ServerNet network adapter is potentially the most challenging part of the whole design. Firstly, a custom PCI interface is needed to communicate with the Colorado 1 (PCI-to-ServerNet) ASIC. Secondly, the Colorado 1 circuit itself is quite complex to initialize. Finally, after the first two elements are understood and implemented, the data transmission and flow control management circuits remain difficult to design. Fortunately, due to the programmable nature of the FPGAs, the different parts of the network adapter interface could be partially designed and debugged separately, then merged to be finished and optimized as a whole.

The first version of the PCI interface was a straightforward port of the existing Pamette PCI Interface. PCI Pamette V1 was designed to facilitate reuse of the board's host PCI circuit in communicating with standard PCI devices plugged in the PMC connector. The pinout of the top-right FPGA to the PMC connector is identical to the pinout of the PCI Interface chip to the host PCI connector. Consequently, the first version of the PCI interface to the ServerNet card was working in a matter of days and the Colorado 1 initialization (performed from software) could be written and debugged quite early in the development process. The final PCI interface design is a specialized version of the standard Pamette PCI Interface (with which it

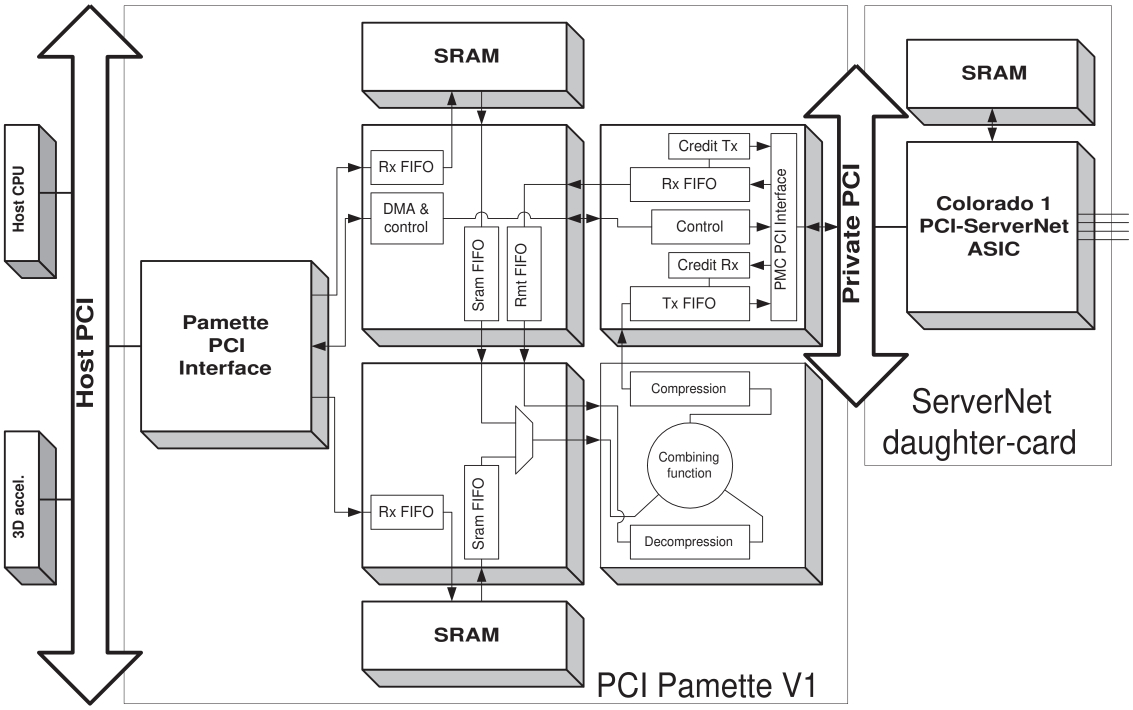

Figure 4: Sepia Phase 1 on PCI Pamette V1.

shares a common source pool) that has more open internal features and more aggressive pipelining. Overall, the design of the special PCI interface was quick and painless.

The Colorado 1 ASIC provides many ways to transfer data and is quite versatile in its addressing modes and translation tables. The initialization code was written and debugged early in the design process, and simple transfers between nodes were performed shortly thereafter.

Section 2.4 justified our use of credit-based flow-control. Given the maximum packet size on Colorado 1 (16 32-bit words), and the amount of buffering available within an XC4044XL (each packet takes less than one of the 40 columns in the FPGA), it is possible to use sufficiently deep FIFOs on input and output to keep packets flowing without interruption. On the Private PCI bus, packets are sent to the Colorado 1 ASIC by writes initiated by the Pamette and incoming data is sent by writes initiated by the Colorado 1. Credits are sent and received using the same mechanism (credits are 32 bits long—the minimum message length). This traffic almost saturates the PCI when handling the expected data rate of 30 MB/s inbound and the same 30 MB/s outbound plus the credit-generated traffic.

3.4 Programmable Merge Engine

Section 2.2 discussed two scenarios each requiring quite different basic operations from the merge engine. One of the design decisions in Sepia is to open the merge engine to advanced users and give them the ability to replace it by one more appropriate to their application if necessary.

In the Phase 1 system, the merge engine occupies most of the bottom-right FPGA on Pamette. It receives two 16b data flows corresponding to the local frame-buffer pixels and the upstream partial pixels. Corresponding pixels in the 2 flows arrive at the same time, with an enable signal. The merge engine must put the data back in RGBαZ format, perform the merging, and serialize the result in 16b words to send them to the network interface located in the FPGA above it.

In addition to the incoming and outgoing data flows, the FPGA contains a register of definable length, to which the software can write to pass information to the merge engine. For instance, for a sort-middle application, the host can send the screen size and the coordinates of the zone for which the node is responsible. This way, the merge engine can use internal X and Y counters and decide which pixel to choose depending on the value of the coordinates. In other scenarios the configurable register may be used to pass the Z-comparison and alpha-blending modes so that they can be performed according to the needs of the software.

The internal logic needed for the merging can be as small as a bit comparison for the sort-middle architecture.

However, as a full chip is available, it is possible to implement much larger merging functions. For instance, to perform blending using a standard $\alpha_{src}c_{src} + (1 - \alpha_{src})c_{dest}$ operator, it is necessary to have four to eight 8×8 parallel multipliers and many multiplexers for the various blending and depth modes.

The fact that no proprietary FPGA code is located in this FPGA makes it easy to allow the user to include his own merging scheme directly in the hardware, with a way to communicate from software to his custom merge engine. The template source code for the sort-last architecture with Z-comparison and blending is available for the user to adapt to specific purposes. It can be modified and recompiled using standard Xilinx tools. Then it can be passed to the runtime software to be downloaded in the relevant FPGA.

There is some non-public code in the other FPGAs, like the PCI Interface. These are provided in bitstream form only, so that there is no special Intellectual Property issue related to opening the hardware to advanced users.

3.5 Mapping on PCI Pamette and performance

The choice of Pamette as a platform to implement the Phase 1 Sepia adapter was guided by very pragmatic goals: being able to have a running prototype for Sepia as fast as possible and being able to reuse as much code as possible for the final version.

The current Phase 1 Sepia is implemented on a Pamette V1R2 using a XC4010E-2 as its PCI interface and four XC4044XL-1 as user FPGAs. With no compression/decompression circuitry and a simple Z comparison combining function, the whole circuit uses about 3000 CLBs (Configurable Logic Blocks, consisting mostly of two 4-input LookUp Tables and two flip-flops), not counting the host-side PCI interface, and runs synchronously with the host PCI clock, usually at 33 MHz. The FPGA design was written in C++ using the PamDC library.

On a 4-node cluster (3 rendering nodes and 1 display node), we measured a sustained transfer rate of 28 MB/s which is close to the maximum bandwidth available from the 3D accelerator (see section 3.2). The measured latency per node is less than 15 μs, which is much better than the target 50 μs.

4 The future: Sepia (Phase 2)

4.1 General architecture

Sepia Phase 1 yields pixel data rates around 4 Mpel/s, which is far from adequate for interactive visualization with large frame sizes. Sepia Phase 2 is targeted at 30 Mpel/s, requiring an order of magnitude faster elements, capable of supporting 160 to 240 MB/s data rates (with 8 bytes per pixel, depending on the effective compression ratio).

In Sepia Phase 1 we can identify three different bottlenecks in the datapath that limit bandwidth:

- The data from the frame buffer is read from the graphics ASIC (and not directly from graphics memory), through an IO bus (PCI) capable of 132 MB/s peak bandwidth.

- The ServerNet network adapter uses 50 MB/s links (30 to 40 MB/s of actual payload) and a 32-bit 33 MHz PCI interface. Both sides are too slow.

- The Xilinx XC4000 (E or XL) family devices with the board level interconnect imposed by PCI Pamette V1 could accommodate higher data rates by clocking at higher speeds (66 MHz), but implementing such circuits in these FPGAs is quite challenging. It is also likely that more internal storage would be needed, which is not available in this FPGA family. The PCI Pamette V1 board itself is near its limits, mainly because of the small pincount of the circuits (208 per chip including power and ground).

To reach the target data rates in Phase 2, some changes are envisaged in the Phase 1 architecture:

- If possible, the frame buffer will be read directly through a specialized port. On a suitable card, it is possible to get between 200 MB/s to 1.5 GB/s this way.

- ServerNet-II [5] will use a gigabit physical layer offering 125 MB/s of bandwidth (100 MB/s of actual payload) full duplex on each link. The Colorado 2 (PCI-to-ServerNet-II) ASIC will have two such links, totaling close to 200 MB/s of bandwidth in each direction. It will also use a 64-bit 66 MHz PCI interface, which matches its data transfer capabilities.

- Use of the new Virtex architecture or high-density XC4000XV circuits from Xilinx will increase merge engine bandwidth by providing faster and bigger parts. In addition, the large quantity of internal RAM will allow more internal buffering and the faster pads will allow implementation of a 66 MHz PCI interface, as is required to fully exploit the Colorado 2 ASIC.

Compared to a Phase 1 rendering node (section 3.1), the Phase 2 rendering node will be composed of:

- A standard workstation.

- A modified 3D accelerator with a dedicated output path from the video output of its graphics RAM.
- A new board derived from the PCI Pamette using Xilinx Virtex or XC4000XV FPGAs.
- A Colorado 2 PCI-to-ServerNet-II ASIC as the back-end of the new board (directly on the board or on a daughter-card).

Individually, the Phase 2 elements are either identical to the Phase 1 elements or natural technology upgrades. The latest technology combined with revised element interconnect will provide the ten-fold increase in performance that we seek, at a commodity price point.

As the elements are still almost identical, there will also be an easy development path from Phase 1 to Phase 2: Phase 2 elements replace Phase 1 elements as they become available, and get debugged and integrated in a stable system.

4.2 Interface to the local 3D accelerator

It is still unclear what the interface to the 3D accelerator will be. The board will still be able to fetch data from main memory or directly from the frame buffer through the PCI bus (and later PCI-X). However, there is much to gain in using a direct connection to the graphics memory. Some graphics card vendors will shortly start providing this kind of features and Phase 2 Sepia will be compatible with at least one of these new boards until a standard starts to emerge. If necessary, Phase 2 Sepia could rely on a custom-made 3D accelerator, that has a built-in direct bus to its graphics memory. This would however defeat the goal of using only commodity pieces besides the merge engine.

4.3 Interface to the network adapter

The network adapter is basically an update of the Phase 1 Sepia network adapter. It uses the Colorado 2 ASIC, which provides more bandwidth and requires a much faster PCI interface (64-bit 66 MHz).

The network in Sepia Phase 2 will be switched, which will permit run-time reordering of the logical merging chain without any physical change. It will also enable the use of the two ServerNet links per board in full-duplex: thanks to the switch, we can create logical point-to point links to different nodes on the same physical wires. This way, the two links will be logically aggregated and the resulting bigger link will be used both to send data downstream and to receive data from upstream.

4.4 Programmable merge engine

The Phase 2 Sepia merge engine will be implemented on a new version of PCI Pamette. The features of the merge engine are likely to be the same as on the Phase 1 system. In this version however, the merge engine will probably be implemented on the same FPGA as the rest of the circuit (including the host interface and the PCI interface to the network adapter). This will lead to Intellectual Property issues if the merge engine interface is still opened to external hardware developments.

4.5 Performance

It is obviously impossible to guess the real performance of such a complex system as Sepia Phase 2 before it is actually built and tested. We believe that for this generation of ServerNet, the network will be the slowest element. We hope to get more than 160 MB/s in each direction from the Colorado 2 ASIC, which will get us very close to the goals set in section 2.5.

At 160 MB/s, we could typically get 30 Mpel/s (depending on the actual compression ratio) for a sort-last algorithm (8 bytes per pixel) or 60 to 70 Mpel/s for a sort-middle algorithm (3 bytes per pixel). In both cases, we would achieve interactive frame rates on large image sizes.

5 Conclusion

Sepia falls into the category of applications that we normally consider as being able to make good use of a versatile reconfigurable board like PCI Pamette: most of the work consists in handling high data throughputs to and from both external and internal sources and some computation is also performed on the data as it flows through the board.

For Sepia, using a programmable board in the prototyping phase (“Phase 1”) had many advantages:

- The tools are simple and well known and most building blocks are already available from other applications, so the development time and the human effort required for Sepia Phase 1 was small (1.5 persons for 3 months). The only physical hardware developed was the custom PMC ServerNet adapter, which was very easy to build from standard specifications.
- Different parts of the FPGA hardware have been designed and tested separately, then merged easily (in particular the PCI interface to the daughter-board, see section 3.3).

- The debugging and runtime infrastructure for the board is already developed and available on multiple platforms, so the software necessary to manage the low-level aspects of a Sepia cluster was developed in a matter of weeks, by the same team that developed the hardware.

- The high-level software development started very early on the first functional system. It can continue with transparent upgrades of the underlying hardware, which bring primarily higher performance and occasionally new functionality.

- The Sepia architecture is opened to external hardware developers who can add their own hardware merge engine and the corresponding cross-platform user-mode driver software.

We strongly believe that it is also very attractive to use a programmable board for an eventual product ("Phase 2" or a derivative):

- Like the prototype, development will be short and will not require much manpower (highly valued in a research environment).

- Large parts of the prototype code will be reused, both in hardware and software.

- Given the price range and the volume for this kind of product, an FPGA-based implementation is a reasonable alternative to the more standard ASIC development.

- In the face of a large number of unknowns (3D accelerator performance and control, Colorado 2 interfacing and performance), the FPGA solution is flexible enough to adapt to all the potential optimizations or problem solutions.

- If correctly engineered, the merge engine is capable of supporting other (future) 3D accelerators or network adapters without knowing their specification a priori.

- The merge engine is still hardware-configurable depending on the application and it will be easy to provide a merge function adapted to any new interesting application of Sepia.

In summary, reconfigurable boards like PCI Pamette prove to be ideal platforms for high-performance data-handling applications like Sepia, even in product versions. Using reconfigurable technology, Sepia can be not only the 3D-compositing network it is designed to be, its hardware can be used as the basis for many distributed graphics processing tasks and more generally for any application that could make use of a smart cluster.

6 Acknowledgements

We would like to thank all the Tandem and Tandem Labs people for their technical support, in particular David Garcia, Michael Knowles, Bob Horst and Albert Tam.

References

[1] H. Pfister and A. Kaufman, "Cube-4 - A Scalable Architecture for Real-Time Volume Rendering," in *1996 ACM/IEEE Symposium on Volume Visualization*, pp. 47-54, San Francisco, CA, October 1996.

[2] C. Silva, A. Kaufman and C. J. Pavlakos, "PVR: High-performance Volume Rendering," in *IEEE Computational Science and Engineering*, Winter 1996, pp 18-28.

[3] A. Heirich and J. Arvo, "A Competitive Analysis of Load Balancing Strategies for Parallel Ray Tracing," in *The Journal of Supercomputing*, vol. 12, no. 1/2, January 1998, pp. 57-68.

[4] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic and W.-K. Su, "Myrinet: A Gigabit per second Local Area Network," *IEEE-Micro*, Vol.15, No.1, February 1995, pp.29-36.

[5] A. Heirich, D. Garcia, M. Knowles and R. Horst, "ServerNet-II: A Reliable Interconnect for Scalable High-Performance Cluster Computing," submitted to *Parallel Computing*.

[6] S. Molnar, M. Cox, D. Ellsworth and H. Fuchs, "A Sorting Classification of Parallel Rendering," in *IEEE Computer Graphics and Applications,* 14(4), July 1994, 23-32.

[7] G. Stoll, B. Wei, D. Clark, E. W. Felten, K. Li and P. Hanrahan, "Evaluating Multi-Port Frame Buffer Designs for a Mesh-Connected Multicomputer," *ISCA '95*.

[8] Compaq Computer Corporation, "PCI Pamette V1," *http://www.research.digital.com/SRC/pamette*.

[9] L. Moll and M. Shand, "Systems Performance Measurement on PCI Pamette," *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 1997.