A Processing System for Real-Time Holographic Video Computation.

Thomas A. Nwodoh, V. Michael Bove, Jr., John Watlington, and Stephen A. Benton.

MIT Media Laboratory

20 Ames Street, Cambridge MA 02139 USA

ABSTRACT

This paper discusses the Chidi holographic video processing system (called Holo-Chidi) used for real-time computation of Computer Generated Holograms and the subsequent display of the holograms at video frame rates. Chidi is a reconfigurable multimedia processing system designed at the MIT Media Laboratory for real-time synthesis and analysis of multimedia data in general and digital video frames in particular. Holo-Chidi which is an adaptation of Chidi, comprises two main components: the sets of processor cards and the the display interface cards. Each processor card consists of a General Purpose Processor (GPP), a processor to PCI bridge, up to 128MByte DRAM, three SRAM-based Field Programmable Gate Arrays (FPGAs), and high bandwidth data transceivers, all resident on a standard PCI form-factor card. One of the FPGAs, called the RP (Reprogrammable Processor), is dynamically reconfigurable and enables Chidi to be used as a flexible specialized hardware for use in performing computations on streams of data and for the control of the transmission of the results through a High Speed I/O interface port to the display interface cards. The GPP controls the FPGAs and reconfigures the RP as needed by an application. Data archival/retrieval capability is provided through the PCI interface via which the system can download/upload data from/to a host system while real-time input data streams can be received through a FireWire interface whence a camera may be connected. The display interface cards assemble and format data from the multiple processor cards for the display. A nine card Chidi processor system can perform over two billion 8 bit multiplications and two billion 21 bit additions in one second. This throughput is enough to generate a 36MB hologram frame in real time and to display the frame at video rates.

KEY WORDS: Holographic video, reconfigurable processor, parallel processing, stream processing.

1. INTRODUCTION

An optical hologram is a diffractive optical element formed by recording the interference pattern between pairs of light beams. Typically, when recording the hologram, a coherent reference beam and the object beam which is locked in phase with the reference beam are both incident on the hologram plate. The two beams interfere and their “phase footprint” is recorded on the hologram plate. Every light wave from points on the 3-D object(s) in the scene interferes with the reference beam and with one another to produce a “picket fence-like” super-imposition of interference patterns (called fringe patterns) which are imprinted on the plate. These fringe patterns look like an array of curves when the hologram plate is viewed on a microscope. When the recorded hologram is illuminated with an optical replication of the reference beam used during the recording phase, the fringe patterns diffract the illuminating light. The diffracted light produces the holographic image which reconstructs the optical wavefronts that fell on the plate from objects in the scene during the recording phase (see [1] for more details).

A Computer-Generated Hologram (CGH) frame has an array of grey scale data equivalent to the optical hologram’s fringe patterns. The generation of the CGH frame starts with a 3-D description of the scene such as obtained with:

computer-aided design models of objects, stereogram's perspective views of a 3-D scene, or such data as in MRI, CAT, and PET databases. The data are then encoded to generate the CGH fringes. There are two basic approaches to hologram computation - the fully-computed hologram approach and the holographic stereogram approach. The fully-computed hologram approach renders the hologram fringes by simulating the light interference process. At the MIT Media Laboratory, three steps are used for this approach: scene modeling, occlusion processing, and interference modeling [14], [15]. The holographic stereogram approach involves the angular multiplexing of a finite number of two-dimensional perspective views of the scene and the encoding of the result into the holographic fringe pattern. These methods generally involve a series of computations on the data obtained from the scene descriptions.

Large amounts of data are generally required to represent a hologram of even small objects. Consequently, the generation of a CGH frame from scene descriptions require enormous space-bandwidth products which would take today's processors of the order of several minutes to compute. Scientists and engineers have adopted some techniques that make the generation of holograms possible within reasonable computation time limits. These techniques include:

- **Data reduction**. The amount of data in a hologram can be reduced without adversely affecting the sensation experienced by a viewer looking at the resulting CGH. This is because of the limited acuity of the human visual system (HVS) [4],[5]. Researchers at the MIT Media Laboratory have exploited this limited acuity and developed an approach that reduces the amount of data in a hologram. This has made possible the computation and display of holograms within the computational capacity of present day processors. By eliminating vertical parallax, reducing the vertical resolution and the horizontal field of view, and limiting the hologram size, the amount of data required for representation of a hologram is reduced by a factor of up to 50,000 [4]. At the Media Laboratory, a Horizontal Parallax Only (HPO) hologram of an object with size 10cm by 15cm on each side and 30 degrees viewing angle can be represented as a 36MB frame organized as 256KB by 144 lines (called hololines). These HPO holograms have horizontal resolution high enough for smooth binocular parallax and a resolution in the vertical direction comparable to NTSC television [6]. Thus the display provides the human visual system with the information essential for viewing objects with accurate 3-D realism.

- **Parallelism**. With HPO holograms, each hololine has all the information needed to produce the corresponding part of the images from it for display. Hence each hololine can be rendered and displayed independently - thus making it possible to represent the holoframe as an array of independent hololines. The independence of the hololines facilitates processing parallelism where many processing units work in parallel with the overall computation task shared between all the processors. *Mark-II* [12], the MIT hologram display component also uses parallelism at the hologram display stage by exploiting the property of independence of the hololines. Eighteen display channels work in parallel with the outputs from all the channels optically assembled at the display.

- **Specialized Processors**. To achieve the high throughput required, hologram computation demands rapid computations. Specialized hardware is needed to perform these rapid computations. Holo-Chidi and its predecessor, Cheops [10] are specialized hardware developed at the MIT Media Laboratory for holographic video (holovideo).

- **Efficient Computation Algorithms**. Simplification of the computation operations required helps reduce the amount of time spent on computing a hologram frame. The Diffraction-Specific fringe computation algorithm [3] reduces the complexity of the hologram computation operation. Also, a more efficient scheduling of tasks to processors, such as with Stream-Based Computing [7], improves throughput.

With the CGH computed, the holovideo is displayed by having the computed fringes modulate an illumination beam to form the image. The fringes (arrays of "dark and white" gray scales) are first converted into analog form,

Figure 1: Chidi Processor Blocks

further processed, and then converted into sound waves in an Acousto-Optic Modulator (AOM) crystal. Inside the crystal, the sound waves propagating down the crystal change the refractive index of the crystal which also has the illumination beam incident upon it. Because of the change in the refractive index of the crystal, the illumination beam is diffracted according to the fringe patterns now encoded as sound waves. The image is then assembled at the display output volume by a synchronized set of scanning mirrors. See [6] for more details.

Prior work has been done on the computation of holograms in real-time and the display of the frames at video rates. Lucente [8] computed holograms with 6 megasamples using the Connection Machine, Model 2, (CM-2) data-parallel supercomputer in under 1 second. Also, using the Diffraction-Specific Fringe Computation [3] algorithm he computed a 36MB hologram in less than 7 seconds. Bove *et al.* [9], [10] implemented the hologram computation and display system using Cheops, a data-flow parallel processor, which could compute a 36MB hologram of an object using a stereogram with sixteen perspective views in 3 seconds. Holo-Chidi is a successor to Cheops meant to scale up the computational capacity of our holovideo processor.

2. CHIDI SYSTEM ARCHITECTURE

The main architectural blocks in Chidi are shown in figure 1 [11]. The GPP is a PowerPC 604e processor running at up to 350MHZ with 64 data bits and 32 address bits. It's function is to control the operation of the entire Chidi processor card by running programs and reading/writing control registers of the specialized processors.

The PCI bridge interfaces the Chidi card to a PCI highway. Through this interface, the Chidi card can access data on a host machine and the host can access Chidi's memory. The PCI bus which is capable of a sustained bandwidth of 132MB/s was chosen because of its ubiquity and robust bus features. With the PCI interface, Chidi can be interfaced to different host platforms which could furnish the system with data archival capabilities. The PCI interface is implemented with an MPC106 - PowerPC/PCI bridge configured for multiple PowerPC processors. In addition to acting as the PCI bus gateway the MPC106 also acts as the system's memory and ROM controller. The main memory has a 128MB DRAM and 512KB flash EPROM. The EPROM holds the system boot code while the

DRAM stores programs and data. The PCI bus is compliant with PCI standard 2.1 with 32 bit data/address bus running at 33MHz.

The Stream Address Generator (SAG, an FPGA) is configured as an additional processor tied to the PowerPC bus. The SAG generates the stream address for data moving to the RP, DS, and FireWire circuits. Recorded hologram data being displayed from a disk resident on the host flows in a stream fashion from the PCI interface through the DS and RP to the HSIO (High-Speed I/O) port whence they are transmitted to the output. The SAG which is under the control of the GPP sets up this data transmission by generating all the addresses needed by the RP and the DS with all the handshake signals needed to implement the data flow. If the hologram is stored in the local main memory, the SAG which can read/write from/to memory also controls the data flow through the DS and RP to the HSIO port. The SAG also controls the data flow to/from memory from/to the FireWire port though the external FIFOs on the data path. Additionally, the SAG acts as a local bus slave controller on the PowerPC bus. When the bus is in the slave mode, the DS using the SAG's slave control signals can access the PowerPC bus.

The DS (Data Shuffler) controls the flow of stream data between the PowerPC bus and the RP. It basically contains buffers, FIFOs, comparators, data flow state machines, and registers needed to move stream data using the Media Lab's Stream Processing mechanism [7]. 64 bits of data can flow through the DS in either direction simultaneously.

The RP is the specialized computing hardware in this architecture. It is dynamically reconfigurable by the processor and can be used to compute a wide range of algorithms including discrete cosine transform, matrix transpose operation, fast Fourier transform, and superposition stream processing. It can input or output data from/to either the DS or the HSIO port. When it is configured for a given algorithm, it processes the input data, storing intermediate results on the 2MB external SRAM tied to it, and outputs its results either to the HSIO port or to main memory. It can simultaneously take 64 bits of data from or to the DS and has 32 bit data access to and from the HSIO port. Interface to the HSIO is implemented with National's LVDS (Low Voltage Differential Signaling) chipset. Each transmitter or receiver can transmit/ receive 18 bits of TTL data at a rate of up to 170MB/s. Using two transmitters, a CHIDI processor card can transmit data to a display card at rate of up to 340MB/s.

Both the SAG and the DS are implemented with Altera 10K50 SRAM-based FPGA chip which has 274 usable IOs on a 356 pin BGA package while the RP is Altera 10K100 SRAM-based FPGA with 403 usable IOs on 503 pin PGA package. The DS and SAG are programmed after system initialization using a serial EPROM tied to their configuration ports. The RP is programmed by the processor with controls generated by the SAG using configuration data loaded in memory. In effect, the RP is the only dynamically reprogrammable device in this implementation as only it can be reprogrammed as needed by the processor at runtime. The system maintains some system control registers (such as card identification register, memory configuration register, etc) on the SAG.

The IEEE 1394 (FireWire) circuit is implemented using TSB12C01A (link layer) and TSB21LV03 (physical layer) from Texas Instruments. The interface has two ports each of which can be transmitting or receiving 4 bytes of data at a speed of up to 33MHz. The external FIFO between the FireWire circuit and the PowerPC data bus is used as a buffer between the two data ports running at different speeds. There is also an external FIFO on the data path between the RP and the DS serving to synchronize the RP and the DS speeds which can be different.

3. HOLO-CHIDI DESIGN

The Holo-Chidi system uses Chidi cards as its basic data processing platform. The system has two main sets of components: the processor cards and the display interface cards. The processor cards can be up to nine standard PCI form-factor Chidi cards while the display interface cards can be up three PCI form-factor cards. The processor cards are the basic hologram computation units while the display cards control the display of a holoframe by scheduling

Figure 2: Layout of Holo-Chidi System

the transmission to the Mark-II display of the data for each hololine received from each of the different processors working in parallel. Figure 2 shows the layout of Holo-Chidi system.

3.1. The Display Card's Design

The display card interfaces Chidi processor cards with the Mark-II display. The following constraints determined the implementation of the display card:

- Since the Mark-II display has no memory, a holoframe being displayed must be refreshed at above the critical flicker frequency (30 frames per second). A 36MB frame being refreshed at 30 frames per second requires a 1.14 gigabytes per second display bandwidth.

- Mark-II display requires 18 inputs to the Acousto-optic-modulators. The system needs to support this data width so as to transmit the 18 hololines simultaneously to the display. The bandwidth of each AOM is 55Mbytes/s (digital) requiring 110Mcycles/s (analog). The display card must be able to feed each AOM channel with data at a rate of 55MB/s.

- A hololine in Mark-II must be read without interruption till the end of the line. Any gaps in the bit stream would be visible on the image volume [6]. Hence 256KB hololine has to be fed to each channel of the AOM as a continuous data stream. To achieve this, a buffer capable of holding an entire 256K hololine per AOM channel is required on the display card.

- A Chidi card supports a data width of up to 4 bytes at its output port. 4 bytes is thus the maximum width of data that can be sent from a Chidi card to the display card in one Chidi HSIO clock cycle.

Figure 3: Block Diagram of the Display Card

Figure 3 is a block diagram of the display card. Chidi's HSIO port is an LVDS interface circuit, hence the input port of the display card has an LVDS interface chipset. An LVDS receiver, DS90CF364 receives two bytes every LVDS clock cycle from a Chidi card while a DS401 receives the SYNC signal from the transmitting Chidi card. The LVDS receivers convert the signals from LVDS signals to TTL. Nine LVDS channels on the three display cards receive eighteen bytes of TTL data per LVDS clock cycle. The D-to-A Converters (DACs) that feed the Acousto-optic-modulators of Mark-II are loaded 4 bytes at a time. Consequently, each byte lane from the LVDS interface is formated into 4 bytes lanes by using 4 latches. The 4 latched bytes are then loaded into 4 FIFOs that are used as a buffer for data for a hololine. The four FIFOs feeding an AOM channel are deep enough that they can hold 256KB of data. The DACs convert the 8 bit digital data fed from the FIFOs to an analog video signal which are then transmitted to a radio frequency signal processing circuit resident in the Mark-II display.

There is a tight control of the reception of the data streams from the Chidi cards, its formatting into different hololines, and subsequent transmission to Mark-II. An FPGA controls the start/end of both loading and off-loading of the FIFOs. A microcontroller is used to control the entire operation of the display card. It loads control data into the DACs and FPGA, and is used for inter-display-card communication. All three display cards are able to communicate through the USART ports of their microcontroller.

At the display, the computed hologram fringes now in analog form, are passed through a radio frequency signal processing circuit where they are modulated with a high frequency carrier, amplified, and then fed into the AOM. The signal fed into the AOM then phase modulate a spatially filtered, expanded, collimated, coherent illumination laser beam so that when a hololine of the frame is fed through the AOM, it produces a diffraction order conforming to the fringe patterns of the hololine. The AOM is a spatial light modulator that diffracts light from an illumination beam by a change in the refractive index of the crystal caused by sound waves launched across the crystal. In the Mark-II display there are eighteen AOMs , each processing the 256KB data that form a hololine into a holographic

Figure 4: Data Input for Holovideo Computation

image for a line on the image volume. The hololine is a horizontal slice of the frame. The image of the entire object is then optically "assembled" by scanning and spatially multiplexing the diffracted beams from the AOMs to the display using mirrors and lenses. To a viewer looking at the image volume, the image appears tangible, displaying the depth cue of horizontal motion parallax. See [6], [12] for the setup used for MIT Media Lab's Mark-II system.

4. HOLOVIDEO COMPUTATION

In Holo-Chidi, the principle used for the generation of the hologram fringes is the Diffraction-Specific Fringe Computation algorithm reported in [3] which is based on the discretization of space and spatial frequency in the fringe pattern. The 3-D scene is modeled with an array of data called *Hogels* (for holographic element), each having a uniform spectrum and discretized into equal regions. Then, generated from a 3-D description of the scene (such as the stereogram's perspective views) are *Hogel Vectors*, each of which describe the spectrum of the associated Hogel. To generate the fringes, each Hogel Vector is used as an array of coefficients of the associated Hogel. The Hogel-Vector encoding technique which is associated with the Diffraction-Specific Fringe Computation algorithm is used to generate the holographic fringes from the scene description. Each Hogel Vector component (ie. the perspective data) is multiplied with a component from a set of pre-computed *Basis Fringes* (supplies the diffraction component). Each basis fringe is computed such that it contains energy in a particular region of the spectrum and control the directional behavior of diffracted light [13]. By multiplying each Basis Fringe with its corresponding Hogel Vector component and summing the results, the resulting fringes contains the spectral energy specified by the Hogel Vector. This generally involves performing a matrix inner product operation.

The RP configured as a superposition stream processor performs the computationally intensive operation of generating the 36MB fringes by multiplying and accumulating the product of the Hogel Vectors and the Basis Fringes. As shown in figure 4, the Basis Fringe is a 1 by 1024 array of data while the Hogel Vector is a 256 by 144 image frame array representing the individual views of the scene.

4.1. Holovideo Algorithm

The holovideo algorithm was implemented for the RP, written in VHDL, and simulated and synthesized using SYNOPSYS. Instantiations of multipliers, accumulators, and dividers were used to process the data streams coming

Figure 5: Functional Blocks for The Holovideo Application

from memory. Several RP implementation modes were simulated and synthesized, including the Computation mode and Display mode. For the Computation mode, the result of the multiply-accumulate operations are stored in the SRAM whence they are fetched during the subsequent display phase.

The holovideo algorithm involves a series of: additions of the product of a byte, ($\mathbf{basis}_{(i(n))}$ in figure 4), from a 1 by 1024 basis function array and a byte, ($\mathbf{pixel}_{(i(j,k))}$ in figure 4), from a 256 byte by 144 line stereogram view of the scene whose hologram is being computed. In the FPGA simulation discussed here, there are thirty two image frame stereograms representing thirty two different 2-D perspectives views of the scene and there are thirty two pre-computed basis functions. However, Holo-Chidi can support and arbitrarily large number of perspective views and basis functions. The multiplications are carried out between image frame and basis function byte pairs as illustrated in figure 4. Each multiplication results in a 16 bit product whose corresponding values are accumulated over all 32 frame and basis function pairs yielding a 21 bit value, $\mathbf{holo\_value}_{(j*n,k)}$ as in:

$$\mathbf{holo\_value}_{(j*n,k)} = \sum_{i=1}^{32} \mathbf{pixel}_{(i(j,k))} * \mathbf{basis}_{(i(n))} \tag{1}$$

Where j has values from 1 to 256, k is the hololine index with range from 1 to 144, and n has range from 1 to 1024.

Each $\mathbf{holo\_value}_{(j*n,k)}$ is subsequently normalized to yield a byte of the final hologram frame. Normalization serves to fit the value of $\mathbf{holo\_value}_{(j*n,k)}$ to the display system used. Mark-II uses 8 bit fringes. Normalization involve the computation of:

$$\mathbf{holo\_pixel}(j*n,k) = \left(\frac{\mathbf{holo\_value}_{(j*n,k)} - \alpha_{min}}{\alpha_{max} - \alpha_{min}}\right) * \mathbf{255} \tag{2}$$

where α_{max} is the largest un-normalized value for the entire hologram frame (that is maximum value of $\mathbf{holo\_value}_{(j*n,k)}$) and α_{min} is the least un-normalized value (typically zero).

A multiply-accumulate module is used to generate the sum of the product of the image frame - basis function byte pairs while a divider is used for normalization. Figure 5 shows the functional blocks used for holovideo computation in the RP. The comparator is used to generate the value of α_{max}. The multiply-accumulate-compare process use several pipelines each of which is made of 8 bit by 8 bit multipliers producing 16 products, 16 bit by 21 bit adders

Option	**Flow Through**	**Multiply-Accumulate-Compare**	**Normalize** (Divider)	**Chip Utilization**	**Speed**
1	64 Byte FIFO	8 stages (no compare stage, 3 stage pipeline)	8 stages (divide by 8192)	71%	24MHz/ 60MHz
2	256 Byte FIFO	—	—	39%	65MHz
3	—	8 stages (3 stage pipeline)	—	68%	25MHz
4	—	—	32 Stages (divide by 8192 with 16 clocks)	62%	30MHz
5	—	—	20 Stages (divide by 7160 with 10 clocks)	95%	19MHz
6	64 Byte FIFO	8 Stages (3 stage pipeline)	—	79%	23MHz/ 47MHz

Figure 6: Performance of the Holovideo Application on the RP

with 21 bit sums, and 21 bit comparators. Each normalizer pipeline has a divider with 21 bit numerators and 13 bit denominators giving an 8 bit quotient representing the final hologram fringe. The numerator used for the division are the $\textbf{holo\_value}_{(j*n,k)}$ values while the denominator used is $\frac{\alpha_{max}}{255}$ which is a constant. The normalization phase of the computation commence after the value of α_{max} and α_{min} have been determined. The comparator placed at the output of the multiply-accumulate pipeline compares each $\textbf{holo\_value}_{(j*n,k)}$ with the current value α_{max} to determine the α_{max} for the entire hologram frame. α_{min} is assumed to be zero here. Because the holovideo algorithm is not very sensitive to the amplitude of the fringes, a divider denominator value of of 2^{13} is used in this implementation. This is the maximum value that $\frac{\alpha_{max}}{255}$ can be. Figure 6 shows a performance summary for the algorithm when implemented for the RP using different options. This performance is generated by FPGA simulator.

4.2. Performance of Algorithm

The conclusions that can be drawn from the numbers in table 1 are reviewed below. For the computation of a 36MB hologram frame with arbitrary brightness:

- In Option 1, all computation operations are carried out in one single phase. The multiply, accumulate, and normalize operations are all executed in tandem without the storage of intermediate values. Here, there is no compare stage. This Option is used in situations where the value of α_{max} can be approximated or determined a priori. The maximum possible value of α_{max} is 255 * 255 * 32. The divider stage uses 2^{13} as the denominator to divide all $\textbf{holo\_value}_{(j*n,k)}$ values. The flow through module is a FIFO implementation that is used for transmitting pre-computed holograms direct from main memory to the HSIO without any computation performed on the data. With this option, the processor could be loading computed data into the SRAM while transmitting previously computed data though the 64 byte FIFO to the HSIO. 24MHz is the speed of the multiply-accumulate-normalize operations while 60MHz is the speed at which the FIFOs can be clocked. Eight multiply-accumulate-normalize stages generate eight fully computed hologram pixels in 33 clocks running at 24MHz with this option. For a nine processor card system, this translates to 1.3 hologram frames computed per second. Chip utilization is 71%.

- Option 2 involves the transmission of pre-computed holograms from the RP to the HSIO ports. The RP just acts as a FIFO buffer for the data flowing from main memory to the HSIO ports. A speed of 65MHz on all 4 byte lanes of the HSIO port can be achieved.

- Option 3 uses eight multiply-accumulate-compare stages requiring 33 clocks to compute eight values of **holo_value**. With nine Chidi cards computing a 256K by 144 hologram running the multiply accumulate stages at 25MHz, it will take about 0.7 seconds to compute one hologram frame. Option 6 is similar to Option 3 but has an additional 64 byte FIFO running in parallel. With option 6, the system could be computing holograms and storing it at the SRAM while transmitting pre-computed holograms from main memory to the LVDS ports.

- Option 4 assumes that the multiply-accumulate values have been obtained and needs to be normalized prior to transmission to the HSIO port. Up to 32 bytes with 16 clocks at 30MHz can be obtained yielding an HSIO speed of 60 MB/s if only one hololine on one byte lane of the LVDS interface is to be transmitted. This is equivalent to a throughput of 14 frames per second (0.07s per frame). Option 4 uses a divide by 8192 circuit.

- The speed and complexity of the divider circuit is dependent on the number of bit ones in the representation of the denominator. A divide by 7160 is more complex than a divide by 8192 and takes more logic elements to implement. In Option 5, a throughput of 20 bytes in 10 clocks running at 19 MHz can be obtained with the divide by 7160 circuit. For a processor with nine Chidi cards, this implies a normalization capacity of about 9 frames per second (0.11s per frame).

The divider implementation used has a constant denominator (ie. divide by 8192, or 8128, or 7160, etc). Also, having a divider circuit with more logic 1s at its denominator gives a better quotient granularity. But this granularity is not necessary since the holovideo output is reasonably insensitive to the amplitude of the fringes.

One factor that can affect the speed of holovideo computation is the speed at which the data (in memory) that are used for holovideo computation can be loaded into the RP. Also, the speed of accessing the SRAM can affect the speed of normalization if intermediate values are loaded there. For a 66MHz processor bus, the latency for EDO DRAM is 7-2-2-2 clock periods (for the 1st, 2nd, 3rd, and 4th double words fetched from memory) and 7-3-3-3 clock periods for page mode DRAM. That is, for burst DRAM read/write, the access time for loading 8 bytes of data is 2 bus clock periods at best. The SRAM has a one clock period access time and can be clocked at up to 66MHz.

The reprogrammable feature of the RP becomes very useful with both the two phase (computation phase and display phase) and three phase (multiply-accumulate phase, normalization phase, and display phase) approach. The functionality of the RP can be changed by reconfiguration. One could configure the RP for computation, store the results in the SRAM, and reconfigure it for display using the SRAM as the data source. In the three phase approach, any intermediate value obtained after the multiply-accumulate stage of the computation can be stored in the SRAM or main memory. In this case, one could also configure the RP for multiply-accumulate computation only, store the data in memory, reconfigure it for normalization only, store the normalization result in the memory, and then reconfigure it as a FIFO for the display of the fully computed hologram frame. The RP can be reconfigured in about 140ms.

Figure 7: Functional Blocks for Modified Holovideo Application

4.3. Computation with Sparse Input Data

Experience with typical input data for fringe encoding show that there are a lot of zeros in the input data. A great deal of computation time can be saved if the multiplication and addition of the zero terms are skipped. The algorithm in the last subsection do not skip the zero inputs during the multiplication and addition stages. With that, holograms with any level of sparseness in its input data take the same amount of time to compute. In a modified implementation, the input data is pre-processed such that only non-zero components are sent to the multiplicand queue. Figure 7 illustrates data flow for this modified approach.

The perspective and basis function data, $\mathbf{pixel}_{(i(j,k))}$ and $\mathbf{basis}_{(i(n))}$, going into the multiplier stage first go through a comparator stage along the processing pipeline. Only non-zero values of $\mathbf{pixel}_{(i(j,k))}$ and $\mathbf{basis}_{(i(n))}$ are sent to the multiplier. Data for $\mathbf{pixel}_{(i(j,k))}$ coming from main memory are pre-processed on the DS where only non-zero values are forwarded to the RP. Also forwarded from the DS to the RP is a control data with which the RP identifies the data positions that are non-zeros. This control data is used in conjunction with basis function control data stored in the SRAM to determine which $\mathbf{pixel}_{(i(j,k))}$ and $\mathbf{basis}_{(i(n))}$ byte pairs have any zeros. The basis function data are pre-processed prior to loading on the SRAM. Only non-zero values are stored along with their i, n positions of $\mathbf{basis}_{(i(n))}$. With this, only $\mathbf{pixel}_{(i(j,k))}$ and $\mathbf{basis}_{(i(n))}$ values whose i, j, k and i, n, respectively, positions are non-zeros are forwarded to the multiplier. For this modified algorithm, the computation time for a typical hologram is reduced by several orders because of the high sparseness of the input data.

5. CONCLUSIONS

The architecture of Holo-Chidi, a research tool being developed by the Spatial Imaging Group of MIT Media Laboratory for holovideo computation and display has been discussed. Prototype versions of the Chidi cards have been built and tested. A new hardware revision for the Chidi cards, the development of the output cards, and the integration of both sets of cards with a host system are all currently in progress. The system when fully developed will make possible the computation in real-time and display at video rates of holograms of complex 3-D structures providing binocular and monocular cues to a viewer. The architecture which is scalable allows an increase in the number of Chidi cards as higher data throughput is needed. This makes Chidi very suitable for the holovideo program of the

Media Laboratory - bigger sized holograms can be computed, higher resolution can be built in, and full-parallax holograms are possible, just by suitable increase in the number of cards.

ACKNOWLEDGMENTS

The research on Holo-Chidi is supported by the Digital Life Consortium Of the MIT Media Laboratory.

REFERENCES

1. P. M. Hariharam, *Optical Holography*, Camdridge University Press, 1984.

2. Mark Lucente, Steve Benton, Pierre St. Hilaire, "Electronic Holography: The Newest", *International Symposium on 3-D Imaging and Holography,* Osaka, Japan, 1994.

3. Mark Lucente, "Diffraction-Specific Fringe Computation for Electro-Holography", Ph.D Thesis, Dept. of Electrical Engineering and Computer Science, MIT, 1994.

4. Stephen Benton, "Elements of Holographic Video Imaging", *Proceedings of the 4th International Symposium on Display Holography*, 1991.

5. Stephen Benton, "Experiments in Holographic Video Imaging", *Proc. of the SPIE Inst. on Holography*, 1990.

6. Pierre St.-Hilaire, "Scalable Optical Architectures for Electronic Holography", Ph.D Thesis, Program of Media Arts and Sciences, MIT, 1994.

7. J. Watlington, and V. Michael Bove, Jr., "Stream-Based Computing and Future Television", *SMPTE Journal.*

8. Mark Lucente, "Interactive Computation of Holograms Using a Look-up Table", *Journal of Electronic Imaging*, vol 2(1), pp. 28-34, 1993.

9. V. Michael Bove, Jr., and John Watlington, "Cheops: A Reconfigurable Data-Flow System for Video Processing", *IEEE Transaction on Circuits and Systems for Video Technology*, Vol. 5, No. 2, 1995.

10. John Watlington, Mark Lucente, Carlton Sparrell, and V. Michael Bove, Jr., "A hardware Architecture for Rapid Generation of Electro-Holographic Fringe Patterns", *1995 IS&T/SPIE Symposium on Electronic Imaging: Science and Technology. Practical Holography*, SPIE vol. 2406, paper #2406-23.

11. V. Michael Bove, Jr., Mark Lee, Yuan-Min Liu, Christopher McEniry, Thomas Nwodoh, John Watlington, "Media Processing with Field-Programmable Gate Arrays on a Microprocessor's Local Bus", *Proc. of SPIE Media Processors*, vol. 3655, 1999.

12. Steve A. Benton, "The Second Generation of the MIT Holographic Video System", *Invited paper: First International Symposium on Three-Dimensional Image Communication Technologies*, 1993.

13. Mark Lucente, and Tinsley A. Galyean, "Rendering Interactive Holographic Images", *Computer Graphics Proceedings*, pp. 387-394, 1995.

14. Ravikanth Pappu, Carlton Sparrell, John Underkoffler, Adam Kropp, Benjie Chen, and Wendy Plesniak, "A Generalized Pipeline for Preview and Rendering of Synthetic Holograms", *SPIE Conference on Practical Holography XI*, San Jose, 1997.

15. John Underkoffler, "Occlusion Processing and Smooth Surface Shading for Fully Computed Synthetic Holography", *SPIE Proceedings*, vol. 3011, pp. 19-30, 1997.