NASA CONTRACTOR REPORT 166500

Development, Design, Fabrication and Evaluation of a Real-Time Video Compression System

(NASA-CR-166500) DEVELOPMENT, DESIGN, FABRICATION AND EVALUATION OF A REAL-TIME VIDEO COMPRESSION SYSTEM (Hein Engineering Corp.) 22 p HC A02/MF A01 CSCL 09C

N83-30911

Unclas
G3/33 13097

D. N. Hein

CONTRACT NAS2-10481
July 1983

NASA

NASA CONTRACTOR REPORT 166500

Development, Design, Fabrication and Evaluation of a Real-Time Video Compression System

David N. Hein
Hein Engineering Corp.
Sunnyvale, California

Prepared for
Ames Research Center
under Contract NAS2-10481

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

Development, Design, Fabrication and Evaluation of a Real-Time Video Compression System
Final Report

By David N. Hein

June 25, 1983

Prepared under contract No. NAS2-10481 by
HEIN ENGINEERING CORP.
Sunnyvale, California

for

AMES RESEARCH CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

1.0 INTRODUCTION

This is the final report on the work done by David Hein at the NASA-Ames research center. This work was done by me under contract number NAS2-10481 between the Hein Engineering Corporation and Ames. This contract spans a period of three and one-half years from December 1, 1979 to May 31, 1983.

The main emphasis of this report is on the work done during the last six months on the Conditional Replenishment Emulator. The progress for May is given in the next section and a description of the emulator is given in the following section.

Brief summaries of the work that was done in the other areas covered by the contract over the entire contract period are also provided. For more detailed descriptions of the work done under the contract I refer you to the three yearly reports and 38 monthly reports that were delivered during the contract period.

2.0 PROGRESS FOR MAY, 1983

During the month of May I integrated the second frame memory and dumb controller into the test setup. This was done by modifying the integration/kludge controller to allow it to handle two sets of frame memory boards. The timing generator also had to be modified to allow it to provide the necessary timing signals to the second frame memory and dumb controller. After talking with Harry Jones about the matter, I decided to permanently use the integration/kludge controller as the smart memory controller. Initially I was planning on having the system controller load the dumb controllers, but using the kludge controller was a much simpler solution.

We also had to modify the interface to the RAM quantizer to talk to the M-bus instead of directly to the HSD. I put Mike Koop on the problem and he came up with a solution in one day. He then made the modifications and had the RAM quantizer talking on the M-bus in another day.

I worked for a couple of weeks on getting the ALU/Sequencer debugged and working. The debugging task consisted of writing system controller programs and running them on both the hardware and the software simulator. I found very few wiring errors on the ALU/Sequencer.

After I completed the debugging of the system controller I added this to the integration test bed. I wrote a simple conditional replenishment program that would keep track of the number of blocks that were being updated and maintain a constant data rate by using the appropiate number of frame repeats. Once I got this program working the integration was completed. This was done in the first week of June.

We then tore down the integration test bed and began the final packaging process. Mike Koop and Mark Leon were working full-time during this period and with there invaluable assistance the packaging took less then two weeks.

I continued writing system controller programs to simulate various conditional replenishment algorithms during the period when the hardware was being packaged. After the hardware was completed I began running these conditional replenishment programs, and tried various ideas to obtain the best overall picture.

There are two main components in the condition replenishment programs that I worked on that control the quality of the picture.

These are the change threshold control and the adaptive quantizer. These were tweaked until an acceptable picture quality was achieved at 1.5 Mbps. The following sections describe the operation of the Conditional Replenishment Emulator, the algorithm that was programmed, and the adaptive quantizers that were developed.

3.0 CONDITIONAL REPLENISHMENT EMULATOR

3.1 EMULATOR HARDWARE

The Conditional Replenishment Emulator is a piece of hardware that can simulate in real-time various conditional replenishment algorithms. This hardware interfaces to the Intraframe Compressor that was previously built at NASA-Ames.

The Intraframe compressor encoder takes in analog color video signals and digitizes it at a rate of 8 Million samples per second. The Intraframe encoder then converts the raster scanned data into blocks of 8 samples by 8 lines. A two-dimensional Hadamard transform is applied to this data and the resulting transform coefficients are rounded to 8 bits. The color information, which is in the form of I and Q signals, is subsampled by a factor of 16 to 1 and a 2x2 Hadamard transform is performed on each of the signals.

The 8 color coefficients that are used for each 8x8 block are inserted in place of 8 of the high sequency luminance coefficients. In this way the intraframe encoder produces a single 8 Mega-byte per second data stream that contains both the luminance and the color information. In the normal operation of the intraframe encoder this data stream is fed to a 2 bit per pel quantizer, rate buffered, and then sent out as a 16 Mbps data stream.

When using the Conditional Replenishment Emulator the transform coefficients are fed directly to the emulator. This is shown in Figure 1. The emulator then produces a processed version of the transform coefficients which are then passed to the intraframe decoder. The intraframe decoder performs the inverse operations of the encoder by doing inverse Hadamard transforms and rescanning the data back into a raster scan format. The decoder produces a analog video signal which can be fed to color monitors or other video equipment.

A block diagram of the Conditional Replenishment Emulator is shown in Figure 2. The transform coefficients go through an adaptive quantizer where each value is quantized to a certain number of bits. The quantized data is stored in a frame memory which acts as an input buffer.

The output of the first frame memory is passed to another frame memory which acts as the display buffer. Various data rates can be simulated by controlling the flow of data from the first frame memory to the second one.

The rest of the functional blocks in the emulator are used to control the updating of the memories. The change detector is used to compare the data that is coming in with the data stored in the display buffer. The mode detector is used to measure the amount of information in a block and produces a mode value that controls the RAM quantizer.

The system controller is a micro-programmable computer that implements the various conditional replenishment algorithms. It consists of three main parts. They are the Data memory, the Program memory, and the ALU/Sequencer. The ALU/Sequencer is the central processing portion of the system controller. It performs the actual computations and controlling functions. It consists of a bit slice

ORIGINAL PAGE IS
OF POOR QUALITY

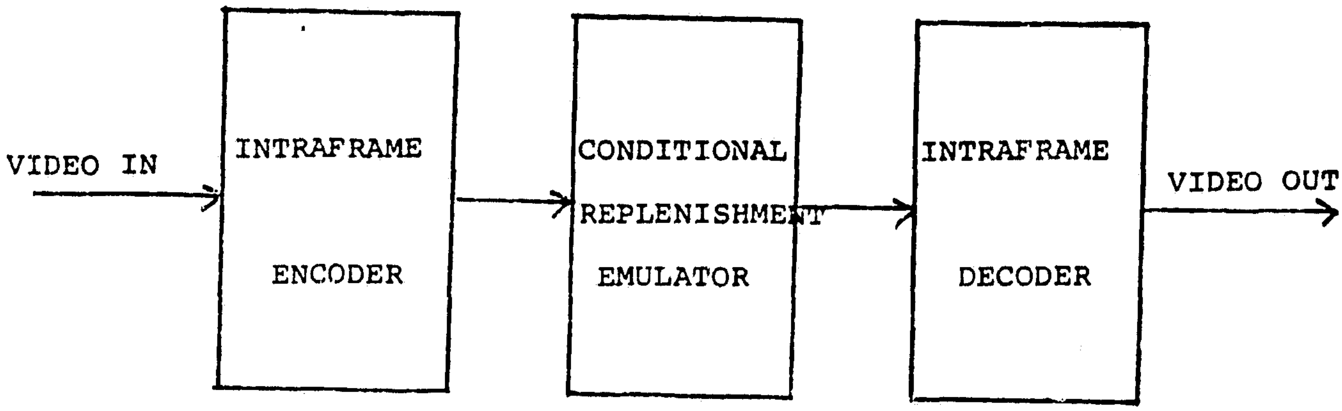

Figure 1. Connections between the Emulator and the Intraframe coder.

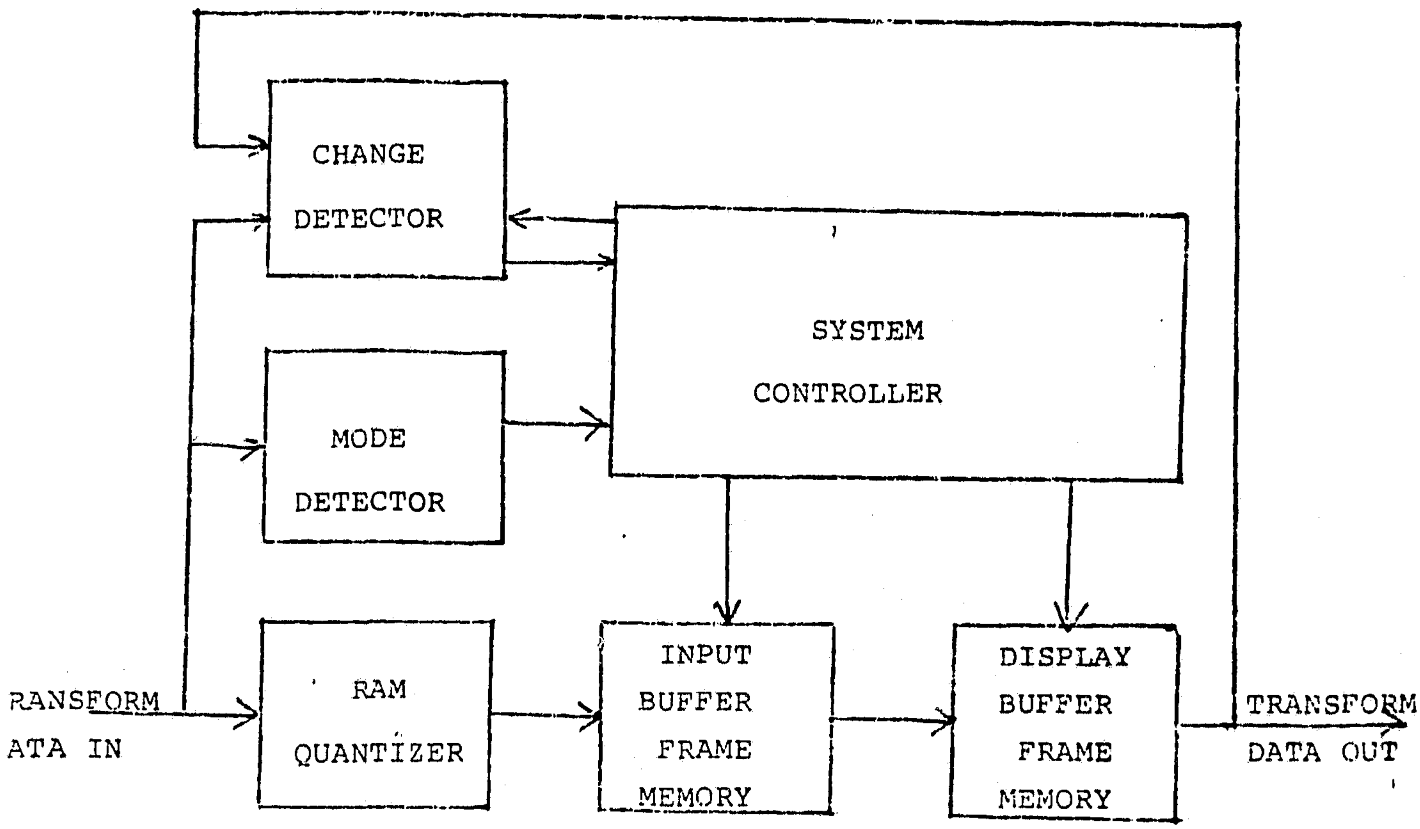

Figure 2. Block diagram of the Conditional Replenishment Emulator.

microprocessor with a word size of 16 bits. The instruction execution time is 250 nano-seconds. Each instruction word is 64 bits wide.

The program memory is used to hold the instructions for the ALU/Sequencer. The program memory is a 1K by 64 bit memory made up of high speed RAM chips.

The data memory is used as scratch pad memory to store the necessary information that is needed by the ALU/Sequencer. The data memory is configured as a 4K by 16 bit block of memory.

The three modules in the system controller and the RAM quantizer are all connected to the M-bus. The M-bus consist of a 16 bit data path which is controlled by four signals. These four signals control the flow of data to and from the ALU. Two of the four signals are used to put a device address on the bus and to automatically increment the last device address that was sent.

The system controller is interfaced to a Gould SEL 32/77 computer. The SEL can down-load programs, quantizers and other data into the memories in the emulator. Diagnostic programs have been written on the SEL which enable it test the various portions of the emulator and verify that they are working properly.

3.2 ADAPTIVE QUANTIZER

The quantizer that was use in the 1.5 Mbps simulation consisted of a 7 mode adaptive zonal coder. The quantizer is made adaptive by comparing the coefficients to be coded with six sets of thresholds. The mode that is used is the lowest one for which all of the coefficients are less than their corresponding threshold. If all of the theshold sets are exceeded in value by at least one of the coefficients then mode 6 is chosen.

The threshold sets that were used for the 1.5 Mbps simulation are shown in Table 1. Locations that are left blank in the threshold tables indication that those coefficients are not tested. Notice that the 00 coeficient is never tested.

Each of the seven modes have a bit assignment associated with it. This is show in Table 2. For the lower valued modes the bit assignments reflect the range that is set by the thresholds. That is, for larger thresholds more bits are required. For the higher valued modes coarser quantizers are used to represent the coefficients. These bit assignments take advantage of the fact that quantization errors are less visible when there are high contrast edges in the picture. Eight of the coefficients in the lower right quadrant of the bit assignment tables represent the eight chrominance transform coefficients.

Table 3 shows the quantizer assignments that were made to produce the bit assignments in Table 2. The major features of the corresponding quantizers are listed in Table 4. There are two types of quantizers that were used. These were uniform and gaussian types. The parameter listed in the table for the uniform quantizers is the width of the quantization bin. The parameter for the gaussian quantizers is the standard deviation.

All of these tables were coded as look-up tables and stored in high speed RAM in the hardware. The hardware uses these tables to determine the quantizing mode number for each block of data, and then determines the correct quantizing table and the quantized value to be used for each transform coefficient.

Since each mode requires a different amount of rate the average

ORIGINAL PAGE IS
OF POOR QUALITY

Table 1. Thresholds for mode detection.

SET 0

-	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	-	-	-	-
2	2	2	2	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-

SET 1

-	4	2	2	2	2	2	2
4	2	2	2	2	2	2	2
2	2	2	2	-	-	-	-
2	2	2	2	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-

SET 2

-	8	4	4	2	2	2	2
8	4	2	2	2	2	2	2
4	2	2	2	-	-	-	-
4	2	2	2	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-
2	2	-	-	-	-	-	-

SET 3

-	16	8	8	4	4	4	4
16	8	4	4	2	2	2	2
8	4	2	2	-	-	-	-
8	4	2	2	-	-	-	-
4	2	-	-	-	-	-	-
4	2	-	-	-	-	-	-
4	2	-	-	-	-	-	-
4	2	-	-	-	-	-	-

SET 4

-	32	16	16	8	8	8	8
32	16	8	8	4	4	4	4
16	8	4	4	-	-	-	-
16	8	4	4	-	-	-	-
8	4	-	-	-	-	-	-
8	4	-	-	-	-	-	-
8	4	-	-	-	-	-	-
8	4	-	-	-	-	-	-

SET 5

-	64	32	32	16	16	16	16
64	32	16	16	8	8	8	8
32	16	8	8	-	-	-	-
32	16	8	8	-	-	-	-
16	8	-	-	-	-	-	-
16	8	-	-	-	-	-	-
16	8	-	-	-	-	-	-
16	8	-	-	-	-	-	-

Table 2. Bit assignments.

MODE 0 - 26 BITS

8	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	7	7
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MODE 1 - 38 BITS

8	3	2	2	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	7	7
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MODE 2 - 89 BITS

8	4	3	3	2	2	2	2
4	3	2	2	0	0	0	0
3	2	0	0	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	7	7
2	0	0	0	0	0	5	5
2	0	0	0	0	0	0	0
2	0	0	0	0	0	5	5

MODE 3 - 132 BITS

8	5	4	4	3	3	3	3
5	4	3	3	2	2	2	2
4	3	2	2	0	0	0	0
4	3	2	2	0	0	0	0
3	2	0	0	0	0	7	7
3	2	0	0	0	0	5	5
3	2	0	0	0	0	0	0
3	2	0	0	0	0	5	5

MODE 4 - 165 BITS

8	5	4	4	3	3	3	3
5	4	3	3	2	2	2	2
4	3	2	2	2	2	2	2
4	3	2	2	2	2	2	2
3	2	2	2	0	0	7	7
3	2	2	2	0	0	5	5
3	2	2	2	0	0	0	0
3	2	2	2	0	0	5	5

MODE 5 - 195 BITS

8	6	5	5	4	4	4	4
6	5	4	4	3	3	3	3
5	4	3	3	2	2	2	2
5	4	3	3	2	2	2	2
4	3	2	2	0	0	7	7
4	3	2	2	0	0	5	5
4	3	2	2	0	0	0	0
4	3	2	2	0	0	5	5

MODE 6 - 242 BITS

8	7	6	6	5	5	5	5
7	6	5	5	4	4	4	4
6	5	4	4	3	3	3	3
6	5	4	4	3	3	3	3
5	4	3	3	0	0	7	7
5	4	3	3	0	0	5	5
5	4	3	3	0	0	0	0
5	4	3	3	0	0	5	5

Table 3. Quantizer assignments.

MODE 0 - 26 BITS

8	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	10	10
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MODE 1 - 38 BITS

8	3	2	2	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	10	10
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

MODE 2 - 89 BITS

8	4	3	3	2	2	2	2
4	3	2	2	0	0	0	0
3	2	0	0	0	0	0	0
3	2	0	0	0	0	0	0
2	0	0	0	0	0	10	10
2	0	0	0	0	0	9	9
2	0	0	0	0	0	0	0
2	0	0	0	0	0	9	9

MODE 3 - 132 BITS

8	5	4	4	3	3	3	3
5	4	3	3	2	2	2	2
4	3	2	2	0	0	0	0
4	3	2	2	0	0	0	0
3	2	0	0	0	0	10	10
3	2	0	0	0	0	9	9
3	2	0	0	0	0	0	0
3	2	0	0	0	0	9	9

MODE 4 - 165 BITS

8	16	15	15	14	14	14	14
16	15	14	14	13	13	13	13
15	14	13	13	12	12	12	12
15	14	13	13	12	12	12	12
14	13	12	12	0	0	10	10
14	13	12	12	0	0	9	9
14	13	12	12	0	0	0	0
14	13	12	12	0	0	9	9

MODE 5 - 195 BITS

8	6	16	16	15	15	15	15
6	16	15	15	14	14	14	14
16	15	14	14	13	13	13	13
16	15	14	14	13	13	13	13
15	14	13	13	0	0	10	10
15	14	13	13	0	0	9	9
15	14	13	13	0	0	0	0
15	14	13	13	0	0	9	9

MODE 6 - 242 BITS

8	7	6	6	16	16	16	16
7	6	16	16	15	15	15	15
6	16	15	15	14	14	14	14
6	16	15	15	14	14	14	14
16	15	14	14	0	0	10	10
16	15	14	14	0	0	9	9
16	15	14	14	0	0	0	0
16	15	14	14	0	0	9	9

Table 4. Quantizer tables.

TABLE NUMBER	NUMBER OF BITS	QUANTIZER TYPE	PARAMETER
0	0	UNIFORM	1
1	-	-	-
2	2	UNIFORM	1
3	3	UNIFORM	1
4	4	UNIFORM	1
5	5	UNIFORM	1
6	6	UNIFORM	1
7	7	UNIFORM	1
8	8	UNIFORM	1
9	5	GAUSSIAN	8.75
10	7	GAUSSIAN	30.0
11	-	-	-
12	2	UNIFORM	1
13	2	UNIFORM	2
14	3	GAUSSIAN	2.8
15	4	GAUSSIAN	4.75
16	5	GAUSSIAN	8.75
17	6	GAUSSIAN	16.0
18	7	GAUSSIAN	30.0

number of bits needed to represent an image depends on the frequency of useage of each of the modes. Images that use mode 0 for most of its block requires a small number of bits to represent it. Images that use mode 6 for most of its blocks will require a large number of bits to represent it.

3.3 CONDITIONAL REPLENISHMENT ALGORITHM

Conditional replenishment works by sending only the changing areas of a video image. The emulator works on small 8x8 blocks of images. Each block is compared to the block from the previous frame that is being displayed at the decoder. If the block is close enough to the one that is at the decoder then it doesn't have to be transmitted. If there is a large difference then the block is transmitted after it has been coded by the adaptive quantizer. The rate that is used to send a block is accumulated. A simulated rate buffer is determined by defining the number of bits that it can hold. At the end of each frame the number of bits that can be transmitted in one frame time is subtracted from the accumulated number of bits during that frame. If the number is less than zero then a buffer underflow has occurred and the change threshold is set to zero as well as the buffer pointer. If the number is greater than the a certain fraction of the buffer then the change threshold is set to a value that will disable any further updating of blocks. This condition is maintained until a sufficient number of frames times have occurred to reduce the level of the buffer below the full mark. If, at the end of a frame, the buffer level is between zero and the full mark, then the change threshold is adjusted proportionately to the buffer level. The minimum value of the change threshold in this region is zero and the maximum value is some previously determined value.

This maximum value was determined experimentally such that an acceptable picture is still produced when the change threshold is set to this value.

The change threshold is modified for each block dependent on the mode that it is being coded under. This is because small differences are much more visible in the lower mode blocks than they are in the higher mode blocks. The reasoning used is the same as that used in designing the quantizers. The higher mode blocks contain high contrast edges which tend to mask out quantization errors as well as frame-to-frame difference errors.

4.0 ADDITIONAL TASKS PERFORMED UNDER THE CONTRACT

4.1 INTRAFRAME TRANSFORM COMRESSOR

I worked on this project for about a year and eight months, six of those months under a previous contract. The intraframe transform compressor encodes color video at a data rate of 16 Mbps.

The transform compressor encoder takes in a color video signal in the form of red, green and blue signals. These are then converted to YIQ signals and digitized at 512 times the video line rate, or about 8 million samples per second. An 8x8 Hadamard transform is performed on the Y signal, while the I and Q signals are each sub-sampled by a factor of 16 and are processed by a 2x2 Hadamard transform. These signals are then quantized by a four mode adaptive zonal coder down to an average rate of 2 bits per pel. The quantizer codes are then serialized and sent out at a data rate of 16 Mbps.

This data is then fed to the intraframe compressor decoder where the quantizer codes are replaced by representative values. An inverse transform is then performed on this data and the reconstructed red, green and blue analog video signals are produced.

4.2 FIELD/FRAME REPEAT ADAPTER

The Field/Frame Repeat Adapter took about seven months to develop and construct. This device takes the 16 Mbps data from the intraframe encoder and produces an 8 Mbps data rate. This is done by transmitting only one out of every two frames or fields, depending on the setting of a front panel switch. This is the same idea that was use in the modified Link-a-bit video compressors.

The decoder will take in the 8 Mbps serial data and by repeating each frame or field that it received it fills in the missing frame or field that was not transmitted. This produces a 16 Mbps data rate which can be fed to the intraframe decoder to produce the reconstructed video signal.

4.3 5770 VIDEO DISPLAY

During the entire period of the contract I maintained the 5770 video display. This equipment was constructed by Dr. Scott Knauer as a replacement for a old and unreliable system that used rotating magnetic discs for the frame storage media.

Even though the 5770 display does have a few persistent problems it has operated quite reliably. As it is now, the 5770 display can continue to be used for Landsat image analysis purposes without too much trouble.

During the contract period we added two enhancements to the 5770 display. These are the color mapper and the internal video sync generator.

The color mapper is mostly used for psuedo coloring of classified Landsat images. This feature is being used in the ELAS image analysis system. It is also used to perform flickering to compare up to three different images, and contrast stretching for image enhancement. The color mapper was designed and constructed by Steve Keating and Mike Koop under my supervision.

The internal sync generator was designed and constructed by Mike Koop. This circuit provides an internal sync reference for the 5770 display and allows the display system to operate on its own. In the past we had to provide an external sync to the display which would sometimes cause problems when we needed to operate some of the other video equipment in the lab.

We also constructed a sync adder and video buffer circuit for the Comtal display used on the IDIMS system. This circuit combines the horizontal and vertical drive signals produced by the Comtal display into a composite sync signal which is added to the video signals. This allows the Comtal to drive standard video monitors instead of the special one that it originally was using. This circuit was designed by me and constructed by Mike Koop and Annie Kong.

Another piece of hardware that was constructed in the lab was the M-80 microcomputer. This was a kit that we purchased and it was assembled by Mark Leon. This device was used to drive the echo disc from the SEL computer. Mark did some of the programming on the M-80 for this purpose. The M-80 was also used in integrating and testing the Conditional Replenishment Emulator.

4.4 SEL COMPUTER

During the early part of the contract part of my duties were to maintain the system software on the SEL computer. During that period we went through several hardware upgrades which usually required software upgrading as well.

During the contract we also upgraded the operating system several times. This required modifying the HSD handler and some of the applications programs.

4.5 COMPUTER SIMULATIONS

During the contract period I managed to perform some research work and did some computer simulations of video compression algorithms. Most of my research was done on motion compensation and condition replenishment algorithms within the frame-work of a transform compression system.

I did some research on very low rate systems that worked by doing subsampling both in the spacial domain as well as the temporal domain. This system used the conditional replenishment ideas that were developed during earlier simulations. I also did some studies on frame difference coding with conditional replenishment and motion compensation. I presented this work at four different conferences and it is also the basis for my doctoral dissertation.

4.6 PROCESSING OF THE IRIS STS-3 IMAGE

During July of 1982 I worked on restoring the IRIS STS-3 image. This is an infared image of the space shuttle as it was re-entering during the third mission. I spent about a month working on this image and found that it was very distorted. I also looked at some calibration data that was taken in the laboratory and found some distortion in this data as well. The calibration data exhibits a loss in resolution which was demonstrated by the rourding of the edges which should have been only one pixel wide but were actually about six pixels wide.

The distortion in the STS-3 image included this artifact. However, there was also a multiple image effect that was much more severe. This multiple image distortion appears to be quite nonstationary. It was concluded that this image cannot be sufficiently restored because of this distortion.