Video Compression for the Pandora Multimedia System

DJ Clarke
Olivetti Research Limited

ORL Technical Report 94-6

Contents

1 Introduction

1.1 The Pandora System

In 1990 Olivetti Research Limited in Cambridge began to deploy a network of Multimedia workstations known as Pandora. Each Workstation consisted of a computer running Unix (an ARM-based machine from Acorn Computer) and an add-on box (the 'Pandora Box') to handle Video and Audio traffic. [3] [2] The Video and Audio was carried on the local 50Mbit/s network, the Cambridge Fast Ring (CFR). [1] This set-up allowed multiple streams of video and audio to be sent concurrently between the workstations and, later, a fileserver machine as well. The whole was integrated so that a user could manipulate the Video and Audio using familiarlooking X-Windows tools, with the Video appearing in one or more windows on the Workstation screen. (See Fig 15.)

The Pandora Box is a MIMD (Multiple Instruction Multiple Data) multiprocessor implemented using Inmos Transputers. [4] The architecture of the Box is outlined below in Fig 1.

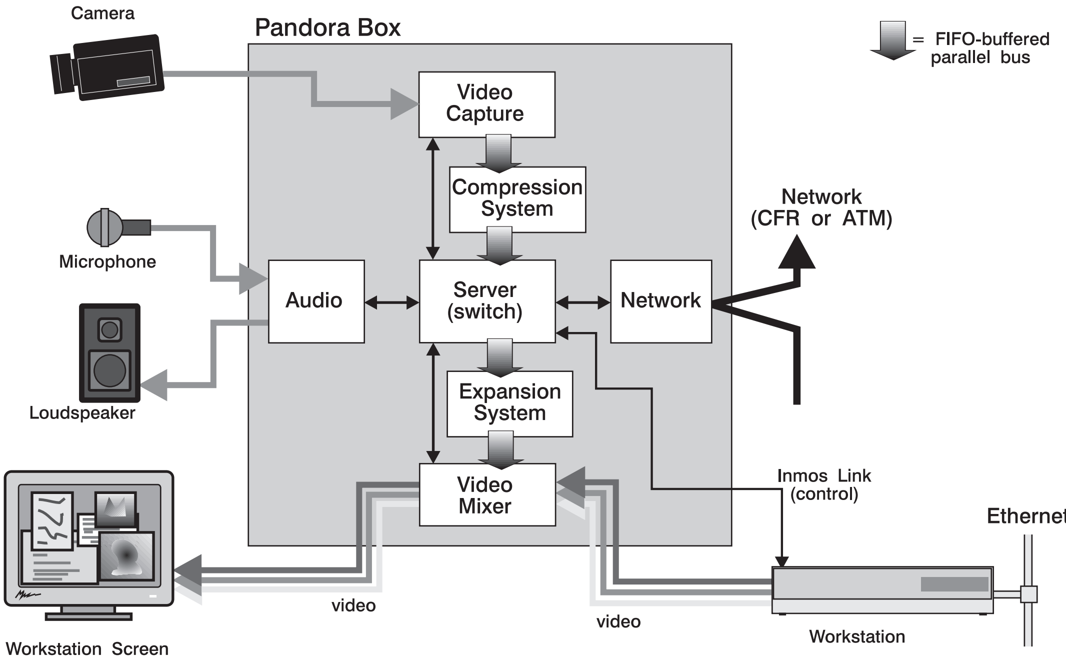


Figure 1: Architecture of a Pandora Box

Video is transferred between the Capture, Server and Mixer subsystems by means of FIFO (First In First Out) buffers. The Compression hardware is located in these buffered datapaths, as shown in the figure.

1.2 Requirements on the Video Compression system

Pandora was designed to handle only monochrome video. Colour video would have increased the data rate and made the system more complex; it was felt at that time, (1989) that we did not need colour video to do research into live networked continuous media.

The chosen network, the CFR, had many good properties for live continuous media streams, including the fine-grain sharing that is now making ATM networking so popular. However, it suffered from a bandwidth limit of about 10Mbit/s in or out of a Station and an aggregate bandwidth limit of about 50Mbit/s for a Ring. Therefore, if Pandora was to support many streams it was clear that some sort of video compression would be needed. Fig 2 shows the sort of data rates involved.

Compression by a factor of around 10 was sought. At the time, the only commercially available video compression systems were videoconference CODECs (COder-DECoders) retailing at tens of thousands of pounds, capable of handling only one video stream, and physically bigger even than the Pandora Box. Clearly something would have to be developed.

Resolution @ Frame-rate	1 stream	10 streams
128x128 @ 12.5Hz	1.6Mbit/s	16Mbit/s
128x128 @ 25Hz	3.3Mbit/s	33Mbit/s
256 x 256 @ 12.5Hz	6.6Mbit/s	66Mbit/s
256x256 @ 25Hz	13Mbit/s	130Mbit/s

for 8-bit monochrome video

Figure 2: Raw Pandora Video data rates

2 Choice of Video Compression algorithm

2.1 Standard Techniques

A truly random sequence of data bits would be impossible to compress without simply discarding some of it. Fortunately, real-world data always has some sort of structure in it, which implies a degree of redundancy in the information. Compression schemes aim to exploit this redundancy and reduce the transmission bandwidth. This has to be done in such a way that an acceptable reconstruction can be done at the receiving end.

Data compression is well-understood, and there are a large number of elegant schemes. They fall into two categories: *lossless* and *lossy.* Practical schemes will usually be a combination of both approaches.

Lossless compression (also known as entropy coding) exploits the data patterns by expressing the *same data* in a form requiring *less bits.* The data stream is regarded as a sequence of *symbols* which occur with differing probability. The trick is to devise an automatic scheme which allocates longer or shorter *codewords* to the symbols so that the overall bit rate is minimised. The best schemes, such as Huffman Coding, can approach the theoretical optimum. Problems will occurr if the mapping from symbols to codewords is not adjusted when the frequency distribution of the symbols changes. A badly tuned system can become a data inflator under these conditions!

Video compression relies heavily on **lossy** techniques. Whenever image data is discarded it produces some effect on the image, so techniques are chosen for their small impact on the perceived image quality. For example, a (monochrome) image will usually be digitised to 8 bits per pixel, but if the least significant bit is discarded, the resulting 7-bit image will often be indistinguishable. However, an image reduced to 4 bits per pixel will show serious 'contouring'. Lossy compression techniques usually involve modifying the image data (passing it through a *transform*) before discarding some of the informaton. Examples of such transforms are pixel-differencing, the Discrete Cosine transform, and Sub-Band Coding. [5]

2.2 Horsepower

Choice of compression algorithm for Pandora depended on the avilability of board area and 'Horsepower'. Size was limited to two 'Single-Eurocard' sized circuit boards.

The outline specification dictated that the system should handle raw video equivalent to a 512 pixel square image at a frame rate of 25Hz. This comes out to a pixel rate of around 7MHz. This rate eliminated any possibility of doing anything useful with a processor chip, so the design had to be realiseable in hardware. Available silicon included:

- Asynchronous FIFO chips 512 deep by 9 bits wide, 30ns access
- SRAM memory 8 bits wide by 8k deep, 35ns access time
- CMOS gate-array chips, about 3ns per gate

The CMOS gate arrays would be a serious investment in time and NRE (Non-Repetitive Engineering) charges, but were the only way to get suitably fast logic into the space available. The SRAMs could be used to implement arbitrary lookup tables, with or without feedback.

2.3 The chosen compression algorithm

Sub-sample the image

The main observation was that, even on the relatively coarse screen being used for Pandora workstations (640 x 480 pixels), a moderately detailed image (say 256 pixels square) would appear quite small. Experiments indicated that acceptable detail could be got from a smaller image (say 128 x 128 pixels) if it could be scaled to cover more of the screen. Simple pixel-doubling would look awful, but some more experiments indicated that bilinear interpolation would usually look acceptable. This approach would give an effective compression ratio of 4 by throwing away 3/4 of the pixels and then making up the missing ones at the other end!

A complication of this sub-sampling would be *aliassing* where high-frequency information is translated down to a lower (spatial) frequency. This effect makes an image appear very gritty, with regular patterns such as striped shirts or railings creating moiré interference patterns. (TV viewers regularly observe this effect on the jackets worn by male weather-forecasters.)

In Pandora, the input TV image is is already somewhat degraded by sub-sampling before the compression system starts work on it. The compression system provides some extra smoothing in the horizontal direction before sub-sampling again. Vertical filtering would have been desireable, but required too much extra circuitry (linestore, adder and control logic).

Difference and non-linearly re-code

The simplest useful transform available is *differential coding*. This method exploits the property of images that ajacent pixels very often have similar values, so that sending the differences can be more economical than sending the actual values. In Pandora the difference is taken between the current pixel and the one immediately to the left. Fig 3 illustrates statistics from an idealised image. It shows each pixel value (brightness) as equally likely, but the distribution of *differences* between *ajacent* pixels shows a bell-shaped distribution.

Unfortunately, the differences can have the same *magnitude* as the original source image, but they now have *sign* as well. The data now contains the same number of samples, but each has an extra bit, so the transform has actually increased the amount of data! However, the data now has quite different properties.

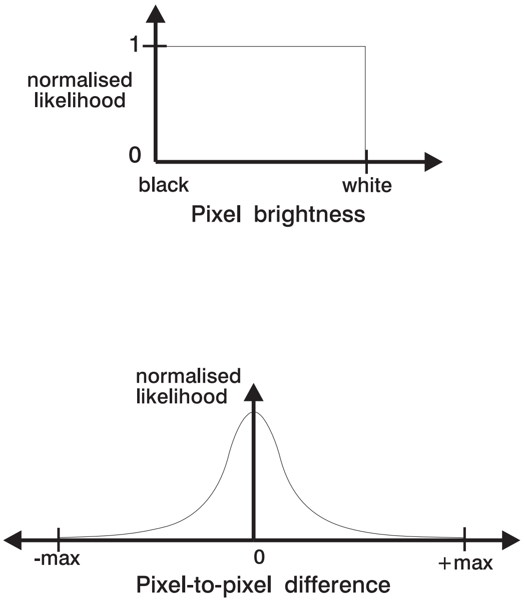


Figure 3: Pixel statistics from an idealised image

Most of the differences will be small. Furthermore, the eye is not good at judging the size of sharp changes in brightness, although it is very sensitive to their location. These two factors allow the differences to be non-linearly re-coded into four bits per difference with very few artefacts (visible errors). The process is illustrated in Fig 4. Note that the diagram shows that large differences are coded much more coarsely than small ones.

Error Propagation

The trouble with differencing and non-linear re-coding is that it introduces errors. Each pixel is reconstructed by adding a value to the previous one. If no corrective action is taken, the errors accumulate leading to unsightly streaks of dark and light which become progressively worse towards the right hand side of the picture.

The solution is to ensure that the reconstructed image always converges on the correct value (see Fig 4) even if it never actually gets there. Fortunately, the correct result can be obtained simply by ensuring that the coder always keeps an accurate account of the current accumulated error. It then transmits a (coded) value for each pixel based on a combination of the pixel-difference data and the current error value. This technique is known as *error propagation.* It works because although the eye can easily detect edges (abrupt changes in brightness) it tends to observe the average value where ajacent pixels only differ by a small amount.

This effect is exploited to great effect in image-dithering techniques such as Floyd-Steinberg error diffusion, which is commonly used to render an image onto a high-resolution screen using a limited colour palette. The best effect is obtained if 'flat' areas can be represented by an exact colour, as the dithering is much more noticeable in the absence of picture detail. In the Pandora Compression system, the code values are chosen such that on 'flat' areas, the exact intensity can be reached in about 3 pixels.

Output format

The input and output datapaths are 32 bits wide. Compressed pixels are packed eight to a word and placed in the output FIFO.

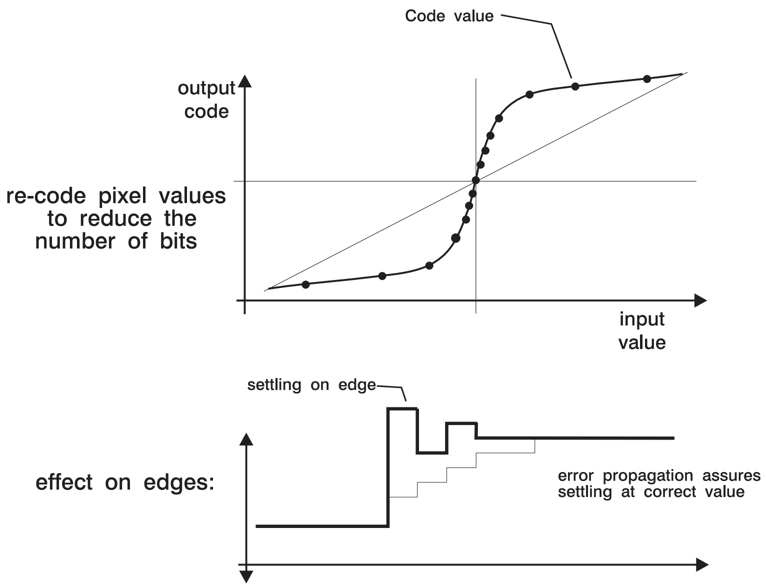


Figure 4: Non-linear re-coding of pixel differences

2.4 Decompression

Image Reconstruction

To decompress the images, the Expansion system looks up each code value to get a difference value, then adds it to the previous pixel value. The process is initialised at the start of each scan line, by starting from a 'previous pixel' value of black. This usually only results in observable errors for the first two pixels, which is not very noticeable.

The differences could have been continued from the previous line without resetting, but since there is usually no correlation between the two edges of the image it would have given little benefit, whilst making the system more sensitive to transmission errors.

Image Interpolation

The Expansion system has 2-D bilinear interpolator hardware. This doubles the number of pixels on a line by making new ones which are the mean of the ajacent 'real' ones. The resulting line of image data is stored in a FIFO memory and replayed in step with the next line, so that the number of lines can also be doubled. In the resulting image, one in 4 pixels is 'real' and the rest are the mean of the ajacent 'real' pixels, as shown in Fig 5. The effect is of a 'soft' picture, in which the original pixels are not readily observable.

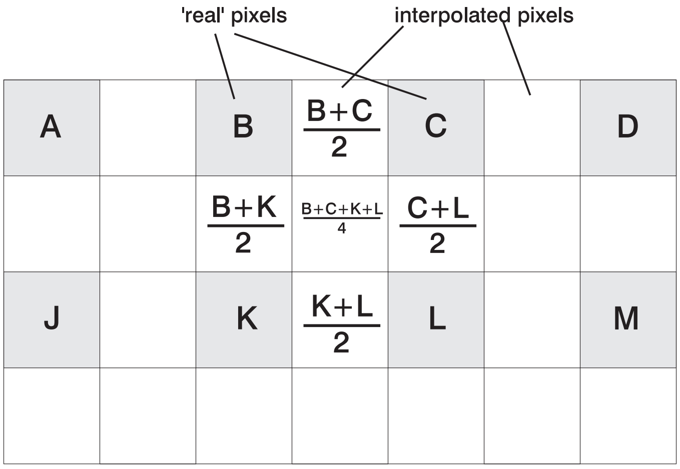


Figure 5: Bi-linear Interpolation of an image

2.5 Test Images

The following images are stills captured during the debug of the Pandora Compression system.

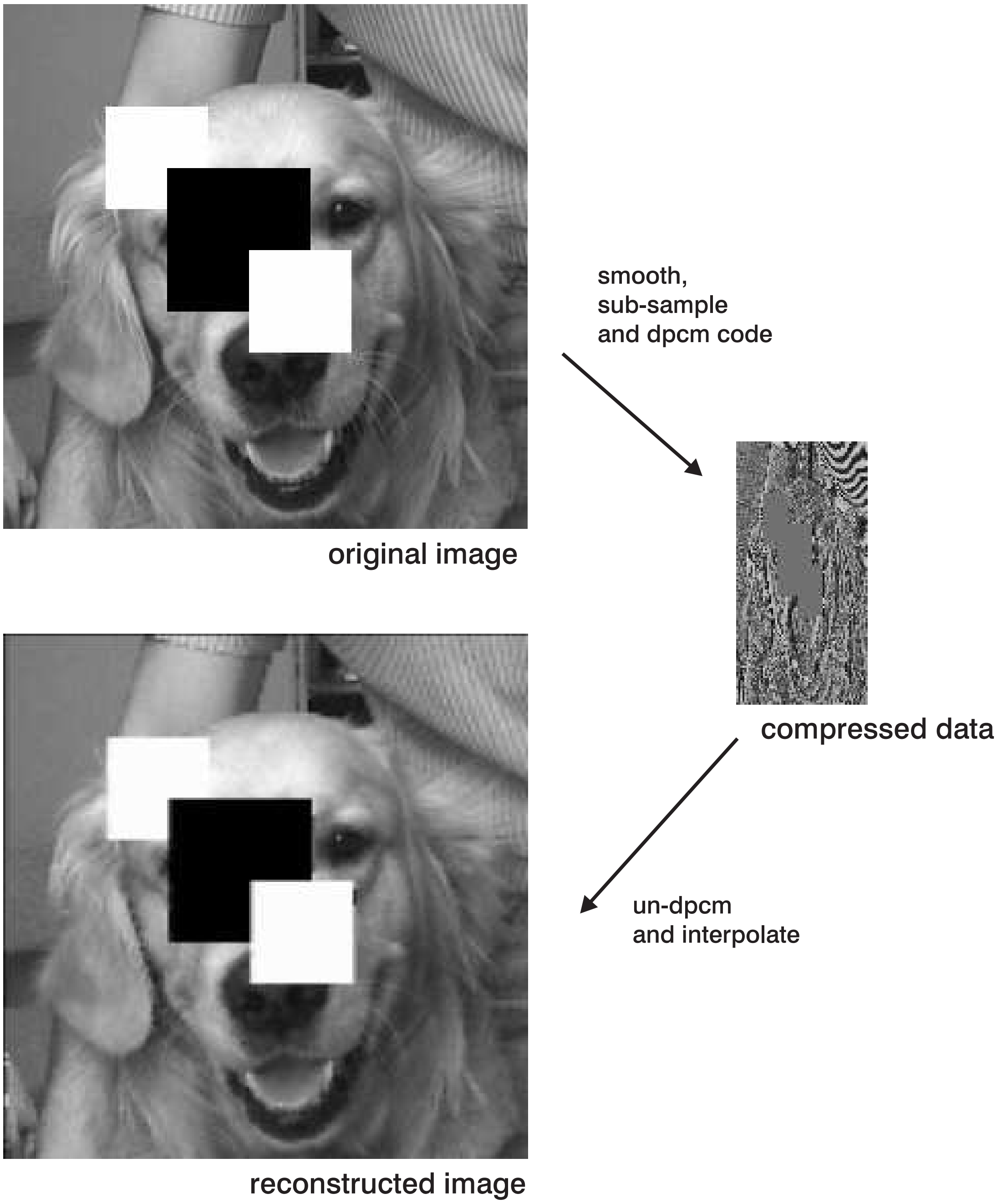


Figure 6: 256 x 256 pixel test image, full compression

Fig 6 shows the effect of the compression algorithm. (The corresponding compressed data is shown at a representative size.)

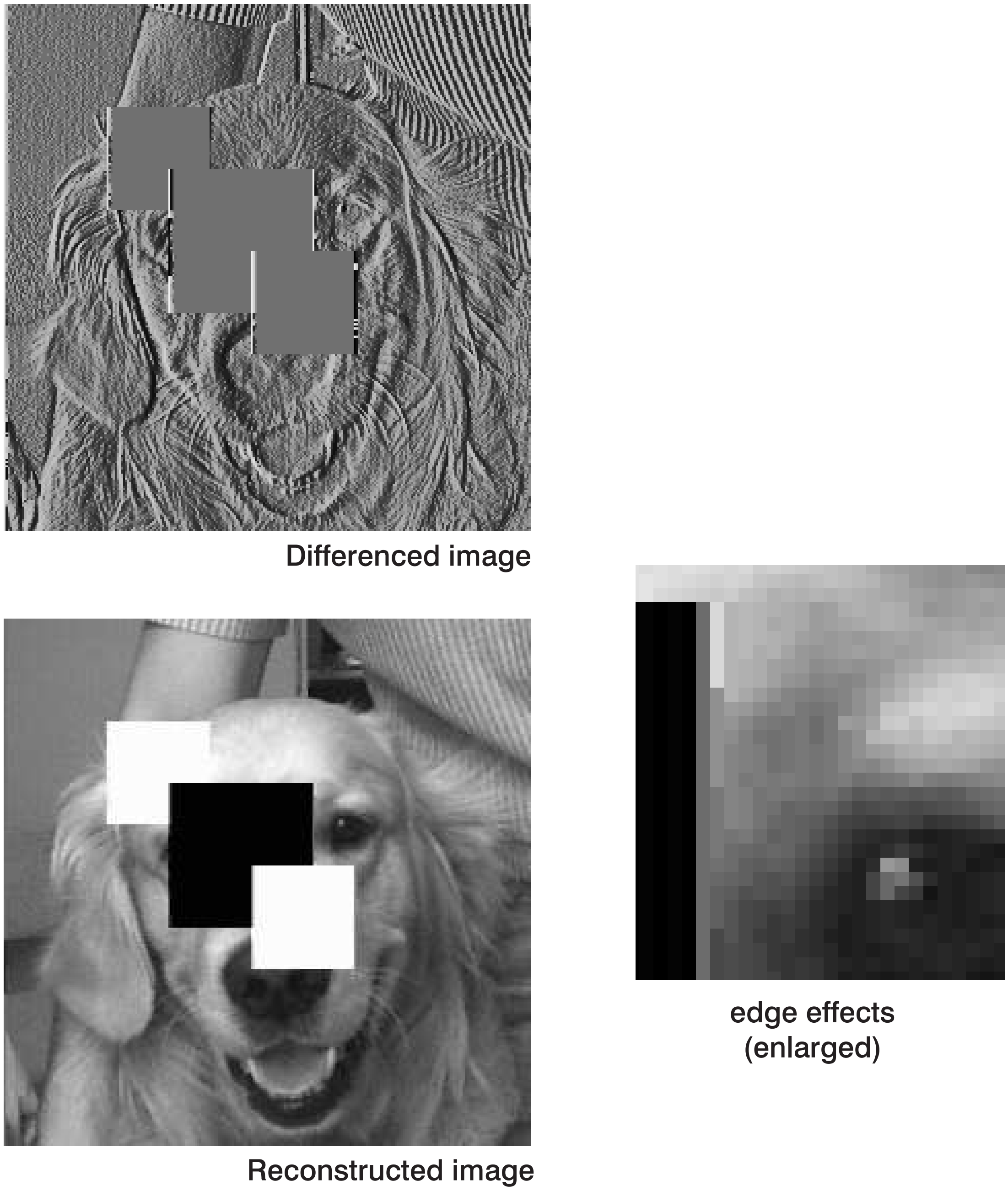


Figure 7: Test image showing the effect of differencing and re-coding.

Fig 7 illustrates what happens to the pixel data when it is differenced and then non-linearly re-coded to 4 bits per pixel. The reconstructed image shows some artefacts, mainly on vertical edges. The black and white squares were deliberately added to the test image to reveal such effects, as real-world images generally lacked the sharpness needed to provoke a clear effect.

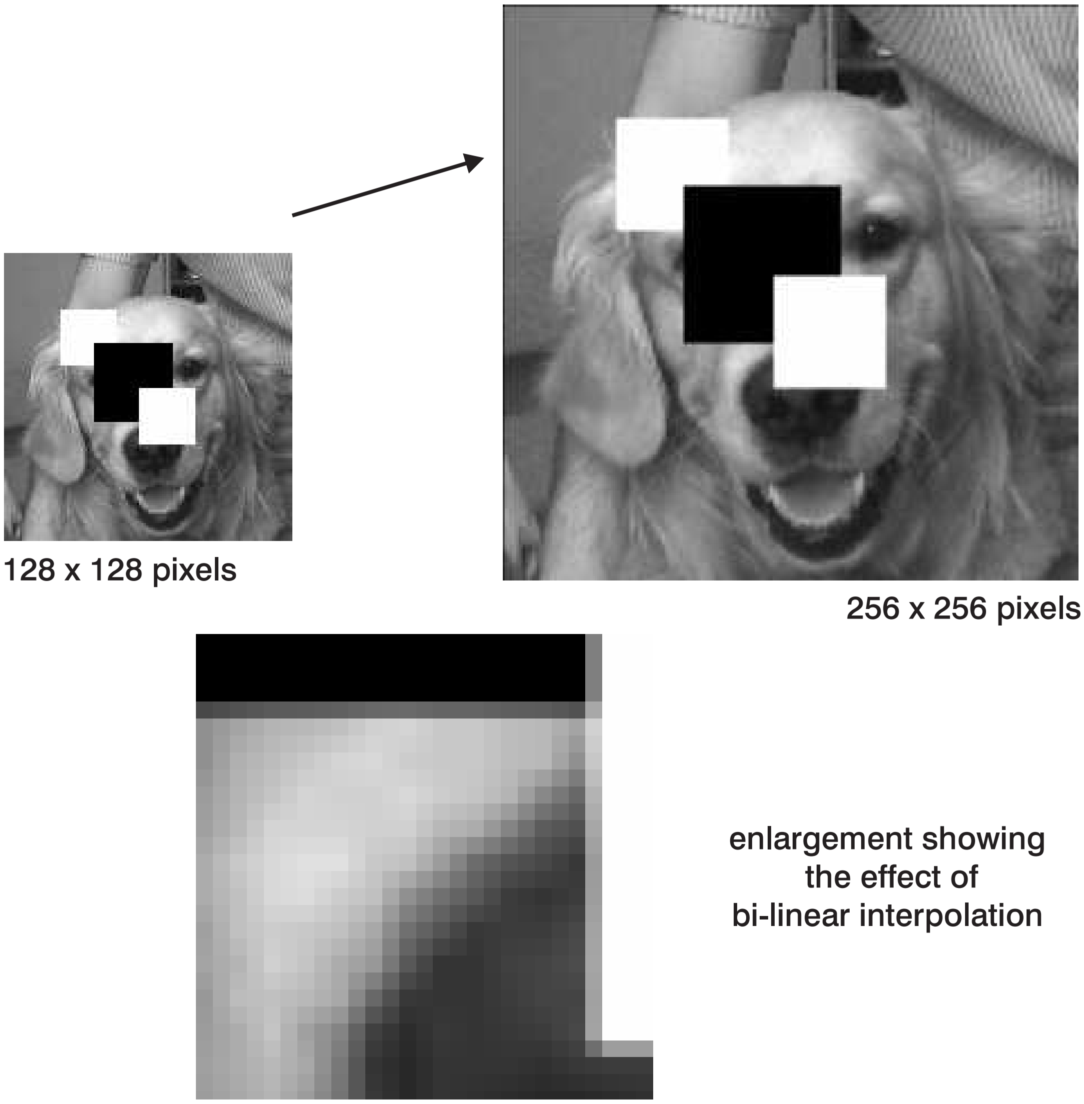


Figure 8: Test image showing bi-linear interpolation.

The bi-linear interpolation used in the Expansion system is illustrated in Fig 8. Some loss of detail is seen. The enlargement shows how the smoothing operates in both x and y directions.

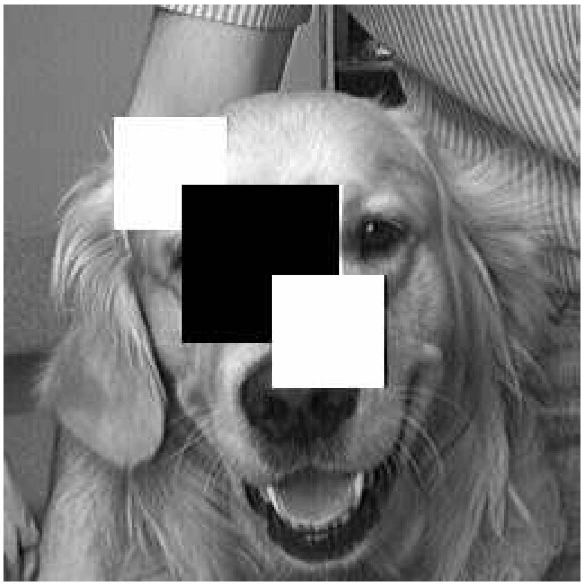


Figure 9: Test images showing horizontal sharpening and smoothing.

Fig 9 shows two possible results from the smoothing/sharpening algorithm in the Compression system. If 30% of the previous pixel is subtracted (and the value normalised) the upper image is produced. In practice, smoothing by 30% is about right; this is shown in the lower image.

3 Hardware Overview

3.1 Integration with the rest of the Pandora Box

At the time design work began on the Compression hardware, the overall architecture of the rest of the Pandora Box was already fixed. This meant that the datapths had to be those shown in Figs 10 and 11.

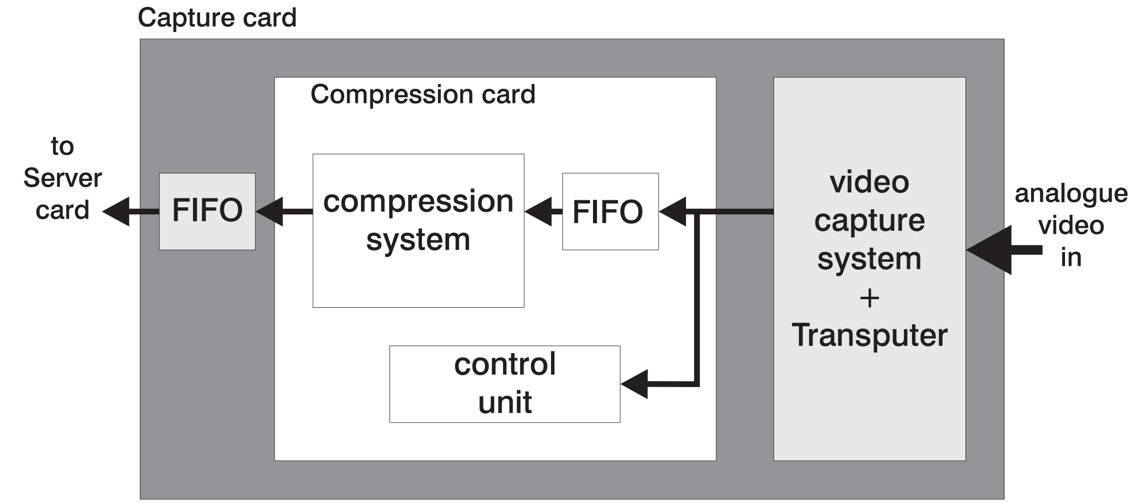


Figure 10: Datapths for Pandora Compression hardware

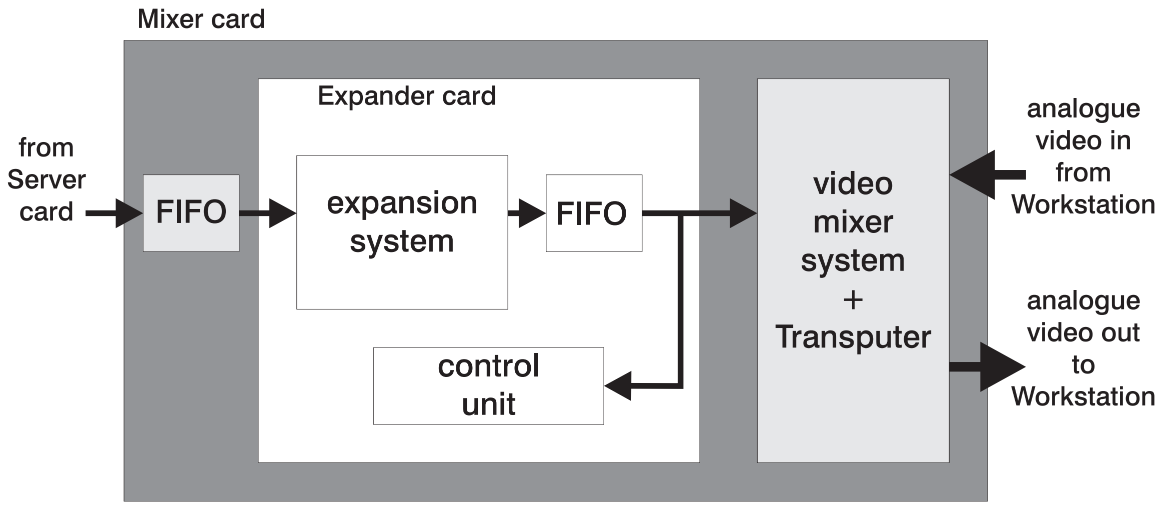


Figure 11: Datapths for Pandora Expander hardware

The compression and expansion hardware fits into a buffered data pipeline. The software handling the data flow has knowledge of the properties of the compression and expansion hardware, so that the correct amount of data is always put into and taken out of the data pipeline.

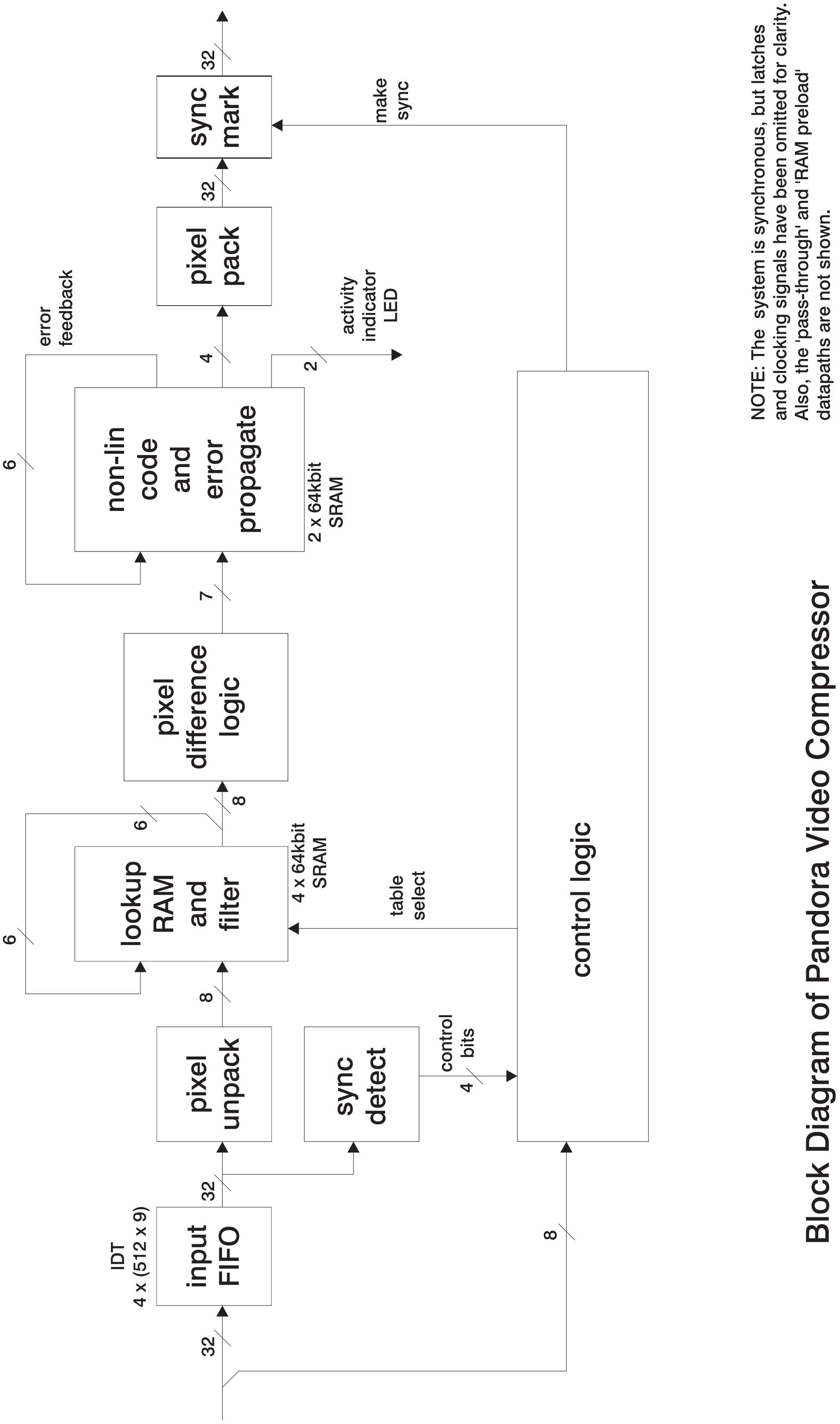

Block Diagram of Pandora Video Compressor
NOTE: The system is synchronous, but latches and clocking signals have been omitted for clarity. Also, the 'pass-through' and 'RAM preload' datapaths are not shown.
input FIFO
IDT 4 x (512 x 9)
pixel unpack
sync detect
control bits
lookup RAM and filter
4 x 64kbit SRAM
table select
pixel difference logic
non-lin code and error propagate
2 x 64kbit SRAM
error feedback
activity indicator LED
pixel pack
sync mark
make sync
control logic
32
32
8
4
6
6
8
7
6
4
2
32
32
8

Figure 12:

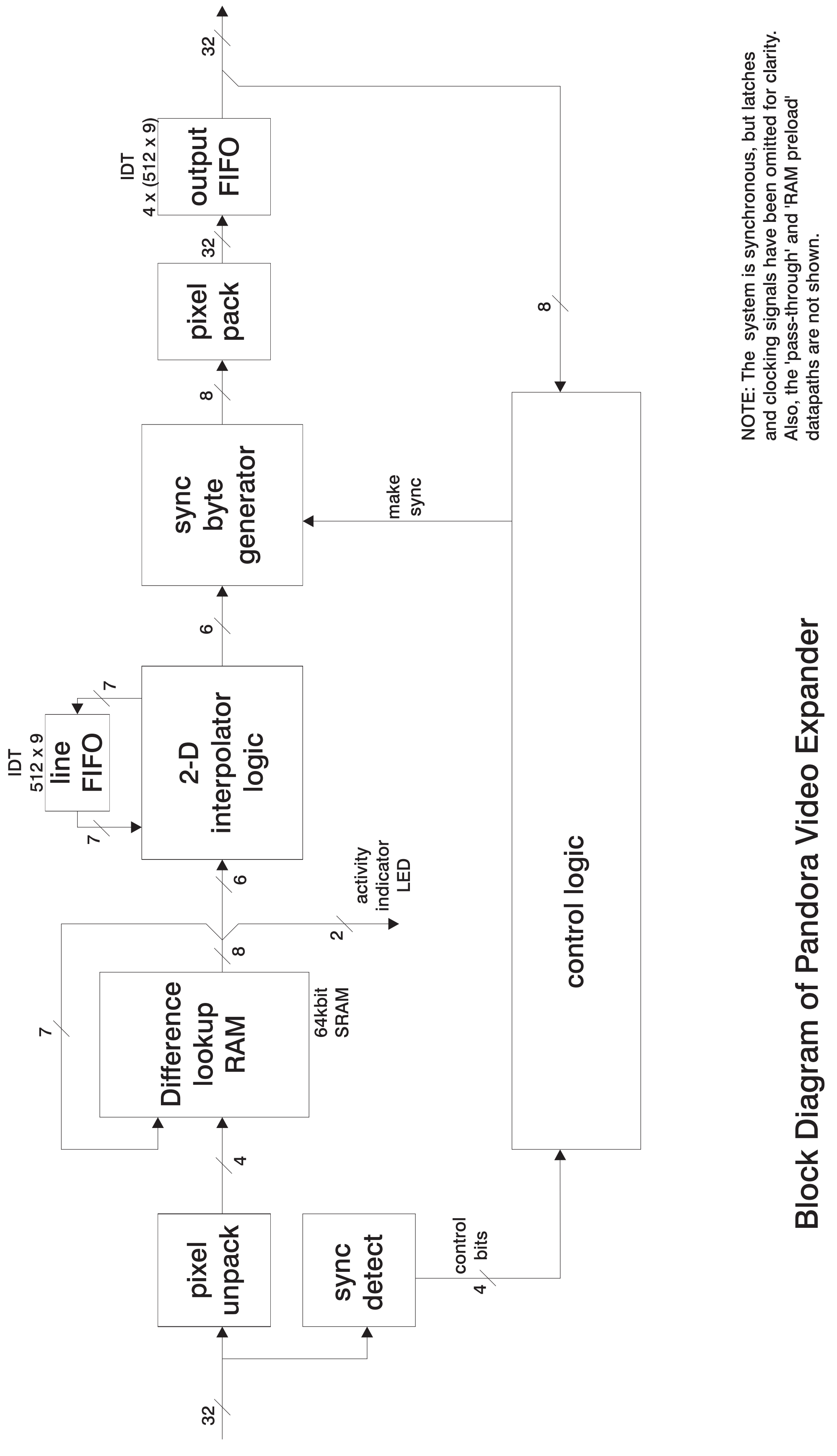

Block Diagram of Pandora Video Expander
pixel unpack
sync detect
Difference lookup RAM
64kbit SRAM
IDT 512 x 9
line FIFO
2-D interpolator logic
activity indicator LED
sync byte generator
make sync
pixel pack
IDT 4 x (512 x 9)
output FIFO
control logic
control bits
32
4
7
8
6
2
NOTE: The system is synchronous, but latches and clocking signals have been omitted for clarity. Also, the 'pass-through' and 'RAM preload' datapaths are not shown.

Figure 13:

3.2 Flexible Hardware

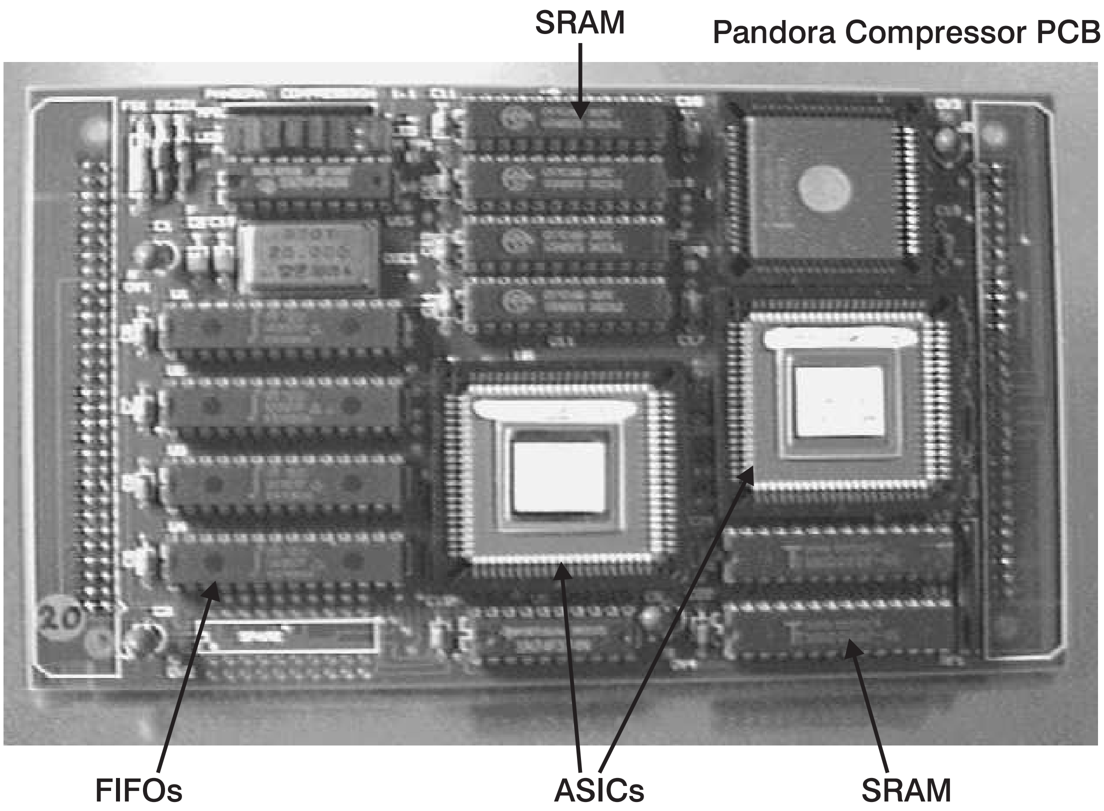
The Pandora Compression and Expansion hardware combines hardwired logic (mostly in custom gate-array ASICs) with RAM-based lookup tables.

The general schemes of the two cards are shown in Fig 12 and Fig 13.

The CMOS gate arrays visible in Fig 14 were used to condense arithmetic and control logic because FPGAs (Field Programmable Gate Arrays) could not deliver the required speed (typically 3ns per gate). Other parts of the algorithm were implemented in SRAM lookup tables. The most interesting way to use SRAMs is with registered feedback, but the number of bits which can be combined in this way was severely constrained by the small size of available SRAM chips. The result can be seen in the Compressor, where the input processing is done by an SRAM lookup table, but the differencing had to be done by hardwired logic, although a (much) larger SRAM could have done that too.

The SRAM solution looks beguilingly elegant. However, in practice, the pain of loading the SRAM tables greatly exceeded that of using them. An embedded SRAM just does not have the right bus connections to get clean data right to it, without the provision of multiplexers, which can consume board space and package pins at an alarming rate.

The ASICs were designed using crayons and paper, along with the chip layout tools from the Cambridge Computer Lab and Qudos. All four chips were extensively simulated, a process which took easily as long as the actual design. The only serious bug was a cumulative error in the interpolator system which only showed up after the third line of the output image. Lessons about simulators: a) you need a good stimulus macro language and b) you need to be able to restart simulations part-way through after altering the stimulus.



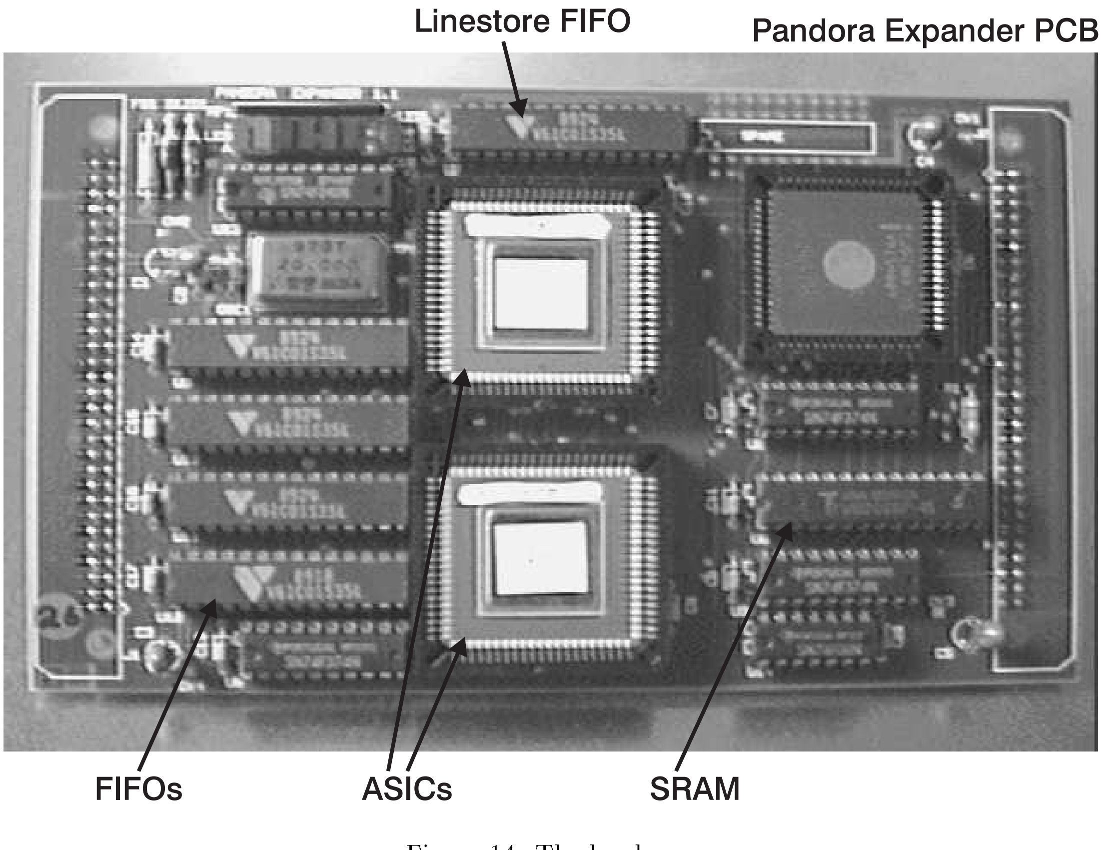


Figure 14: The hardware

The options available from the final hardware are:

Compression system

a) programmable input lookup tables and smoothing (choice of 2)
b) discard alternate pixels (or not)
c) difference and non-linearly re-code (or pass-through)
d) pack as (8 x 4-bit pixels) or (4 x 8-bit pixels) per 32-bit word

Expansion system

a) unpack as (8 x 4-bit pixels) or (4 x 8-bit pixels) per 32-bit word
b) restore image from differenced form (or pass-through)
c) interpolate and magnify by x2 (or pass-through)
d) specify frame-start (or mid-frame)

Each option is contolled by four bits of a control word which is sent at the head of each video line, so that the processing parameters can change on a segment-by-segment basis. ('Segment' is the term coined to decribe part of a video frame. Segments are handled individually, allowing lower latency and finer grain bandwidth sharing than if whole frames were used.) It would have been better to have a header only at the start of each segment, but this would have added to the hardware complexity.

Line dropping at the Compression side is performed by the Capture Card Transputer. The Expander has more 'state' than the compressor, because it stores the *previous line* of video. At the start of a new frame, the frame-start control bit causes the Expander to assume all-black for the previous line. On all other segments the Expander has to be fed a copy of the last line it saw of that video image, so as to preload it with the right state. Normal use of the Expander involves the interleaving of segments from different video streams, so the state of the Expander has to be refreshed at the beginning of each segment.

3.3 Compression Ratio

Strictly speaking, the method described achieves a compression ratio of a little less than two, because the image is sent as four bits per pixel and is represented on the screen as seven bits per pixel.

However, in terms of the area of screen covered, it does rather better. The Pandora Box hardware has no pixel-doubling capability (the processors are not fast enough); the alternative of sending raw video actually uses 8 times the network bandwidth comared to the compressed video.
So in practical terms the compression ratio is 8.

4 Conclusion

Figure 15: A Pandora screen showing multiple video windows

The video compression system described works well and provided an unprecedented degree of flexibility in the handling of video streams. Video segments (fractions of a frame) are interleaved without penalty in performance. Different users can obtain different image sizes and qualities without one user's choice restricting what another can choose (subject to processor and network performance limits).

The software tools developed for Pandora always use the difference-coding as this was found to have negligible effect on image quality. The user can opt to expand the image on-screen to 2x size, or not, giving a choice of small-and-crisp or larger-and-soft. It is normal for a Pandora user to have several 128 x 128 pixel streams or two 256 x 256 pixel moving images on the screen at once. Fig 15 shows a typical Pandora screen.

The technology used in the Pandora Video Compression was quickly overtaken by developments in LSI silicon image compression. By the time Pandora was fully deployed several chip makers were offering Discrete Cosine Transform based products. However, none of this new silicon directly addressed the business of handling multiple video streams interleaved at a fine (sub-frame) grain.

It was clear from our work that successful handling of live video streams requires low latency. If the unit of transmission is a *frame* of video, then up to one frame of delay can be introduced in the transmission path. Compression systems in a serious Multimedia application need to be able to interleave streams at fine-grain, so that they can deliver multiple video streams with the minimum end-to-end delay.

DC

References

[1] Prof. Roger Needham Dr Andy Hopper. The cambridge fast ring networking system. Technical Report 88-1, Olivetti Research Ltd, 1988.

[2] Dr Andy Hopper. Digital video on computer workstations. *Proceedings of Eurographics*, 1992.

[3] Dr Tony King. Pandora: An experiment in distributed multimedia. Technical Report 92-5, Olivetti Research Ltd, 1992. also presented at Eurographics '92.

[4] Inmos Ltd. *The Transputer Reference Manual.* Prentice-Hall, 1988. ISBN 0-13-929001-X.

[5] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, 2nd edition, 1982.