DESY DV-78/02
April 1978

Digital Video System for Real-Time Processing[+] of Image Series

G.C. Nicolae and K.H. Höhne

Deutsches Elektronen-Synchrotron DESY

[+] Research supported by grant DVM 130 of the Bundesministerium für Forschung und Technologie.

A short version of this paper has been submitted to IEEE Transactions on Computers.

Content

Zusammenfassung

Im Rahmen eines Projektes zur Auswertung von Röntgenbildfolgen wurde ein neuartiges System zur digitalen Verarbeitung von Video-Bildfolgen entwickelt. Es erlaubt, Videobildfolgen in Echtzeit zu digitisieren (bis 512 Punkte/Zeile, 8 bits/Punkt), mit hoher Geschwindigkeit zu verarbeiten und die Ergebnisse in graphischer Form - auch in Farbe - darzustellen. Die extremen Anforderungen für die Digitisierungs- und Verarbeitungsgeschwindigkeit (mit Datenraten bis 80 Mbaud) führten zu einer Realisierung in Form eines Multiprozessor-Systems. Die mikroprogrammierten Spezialprozessoren kommunizieren über einen asynchronen Hochgeschwindigkeitsbus mit einer Transferrate von 15 MWorte/sek. Ihre Funktionen werden durch einen Kontrollrechner (PDP 11/45) gesteuert. Um die beschriebenen Systemleistungen zu erbringen, wurden strukturierte Entwurfsmethoden wie Petri-Netze angewandt, welche zu einem transparenteren Entwurf, einer besseren Dokumentation und kürzeren Implementationszeiten führten. Dieser Bericht beschreibt die Struktur, die Programmierung und die Anwendung des Systems.

Abstract

While studying the analysis of X-ray picture sequences a new system for digital processing of video images has been developed. This system allows real time digitization (up to 512 pixels/line, 8 bits/pixel), high speed processing of video images and presentation of the results in a readily interpretable way on a colour TV display. The extreme requirements of speed of digitization and processing (with data rates up to 80 Mbaud) led to a realization in form of a multiprocessor system. The special microprogrammed processors communicate via a high speed asynchronous bus at a throughput of 15 Mwords/s. Their functions are controlled by a host computer (e.g. PDP 11/45). To achieve the above performance, structured design methods, such as Petri-nets have been applied, which gave rise to a more transparent design, better documentation and shorter implementation times. This paper describes the structure, the programming and the application of the system.

1. Introduction

Among the variety of medical image processing applications the analysis of X-ray pictures is of growing importance. Whereas in the past 10 years effort has been concentrated on the analysis of static pictures, there is an increasing interest of physicians in the extraction of functional parameters from time sequenced x-ray pictures (angiodensitometry). The procedure for an angiodensitometric analysis (e.g. for the kidney) is as follows: The physician applies a radio-opaque contrast medium to the kidney under examination. This medium propagates through the kidney vessels at the speed of circulation of the blood. This scene (see fig. 1) is viewed by the image-intensifier video system of the x-ray equipment. For a quantitative analysis, the time-sequenced video images have to be digitized, stored and subsequently processed. The result of the computation may be local parameters such as blood velocity or blood volume per unit time in a blood vessel selected by the physician, or maps of functional parameters such as delay times of the contrast medium for any point of the x-ray picture field (functional images).

In a collaboration of DESY with the Radiology Department of the University-Hospital Hamburg-Eppendorf, a research project for the quantitative analysis of the blood flow in such organs as the kidney has been started. For this project we had to design and implement an interactive user-guided system for this application in a clinical environment [1]. This led to the following essential requirements: The system had to be fast. The acquisition of the picture series had to be performed at a speed allowing for immediate analysis. As a consequence, we chose the video signal as input and aimed at a real-time digitization of the video information.Also the analysis had to be fast because it had to be carried out as an interactive process [2]. The choice of each step of analysis depends on the result of the foregoing one. A routine application of this technique is only possible if it is not time consuming. Should furthermore, the result of a step require acquisition of new data, it has to be available during the time that the patient is still present.

In detail the following functions had to be performed by the system:

- Picture data acquisition, storage, display and transfer of pictures (or regions of pictures) at standard video rates (50 frames/s, 10 M samples/s).
- Picture data processing such as contrast enhancement, quantitative analysis as computation of blood flow at a speed allowing interactive analysis.

- Suitable communication between user and system by
  - adequate means of (graphic) command input,
  - pictorial and graphic output which is
    - easily interpretable by use of appropriate geometrical (shape), optical (grey level, colour) and mark (blink) attributes, and
    - fast enough for interactive use and animation of pictures.

Similar demands may be found in other fields of multitemporal image processing such as production and traffic control or desctructive material testing.

Earlier approaches to this problem have used analog techniques [3] . Despite their real time features the lack of accuracy and flexibility proved to be a severe drawback. Digital techniques, however, potentially meet these requirements. The major difficulty with digital processing of video images lies in the high data rates. The video samples have to be digitized, stored and processed to obtain digital video images with space, time and intensity resolution comparable to those of analog video images. The commercial European CCIR standard requires 625 scan lines/frame, 50 half-frames/s, 6 MHz bandwidth. With a minimum intensity resolution of 8 bits, the resulting data rates of about 80 Mbaud associated with the corresponding frame buffer size of 256 kbytes reach the technological limits of the exisiting digitization and memory technology. Whereas in the past these requirements of bandwidth and storage capacity could not be met, the threshold of feasibility has now been reached by the rapid development of solid state technology.

In order to reduce the data rates, the first digital systems used time multiplexed acquisition techniques in conjunction with analog video tape and/or disk recorders [4] . Deterioration of the output video signal from the analog recorders - due, for example, to large scan jitters and small signal /noise ratios - in the multiple replay mode lead to the development of real-time systems which digitize video images directly in a single play.

Read et al. [5] have designed a system which works at 10 frames/s. Gilbert et al. [6] have described a system which manages 60 frames/s at a rate of 40 M samples/s. When designing a system, it is not sufficient to have fast data channels such as acquisition, storage, display and processing devices. The main

problem is to find a structure in which these data channels cooperate to give the performance required by the application. Gilbert et al. chose a star-like structure, in which the data channels are gouped around a processor called "data-interchange", in which the entire *processing power* is *concentrated*. It connects pairs of data channels and performs the data processing required for each configuration needed by the user. This structure allows a high throughput (up to 40 M words/s) especially when working in synchronous mode, but needs relatively long times (up to 5 µs/instruction) for reconfiguration.

Considering the most urgently required features of flexibility and parallelism we chose for our Digital Video System (DVS) a structure with *distributed processing power*.

This introductive description is to provide information on

- the architecture
- the programming and
- the application fields

of the system.

2. Architecture of the System

2.1. Design Considerations

A system which satisfies the above requirements can be considered as a pool of different processes such as picture generation, picture display, etc. Table I contains the list of the processes which occur in the DVS, with their semantics and a set of associated attributes. Looking at Table I in view of our objective of achieving a throughout compatible with the real-time requirements and a maximum of flexibility we make the following observations:

1. A high throughput can be achieved by *paralleling* some processes such as picture generation, picture display and picture processing. In some cases the processes can be decomposed again into several concurrent subprocesses. This situation is illustrated in fig. 2 which shows a model of the processes. The processes connected with a double line are concurrent. There are three hierarchical levels on which the processes run. They can be characterized by the level of programming required: problem oriented, macro- and micro-programming.

2. There is a wide spectrum of *flow speeds*. The means of realizing a given process, whether by hardware, firmware, software or a combination of those, must be chosen accordingly. Clearly the picture generation and picture display processes are a matter of hardware. Picture processing, however, may be done by a general purpose computer or by dedicated processors depending on their complexity. The system actually implemented should, therefore, offer the flexibility of choosing between these according to the user's needs.

3. Depending on the user's needs and the inherent time constraints of the various processes they have different *priorities* in using common system resources such as frame buffers, data paths, etc. The picture frame generation process - for example - must have a higher priority in using the system resources than any other process. Otherwise picture data will be lost.

Considering the above structure of the design task we decided to realize the system as a multiprocessor system.

2.2. Structure of the System

Fig. 3 shows the implemented system structure. It consists of three main parts: the host computer, the communication interface and the multiprocessor system for real time processing of video images. Although the further description will concentrate on the last of these, the other parts are briefly described.

2.2.1. Host Computer

The host computer is a PDP11/45 mini-computer equipped with a floating point processor, 80 k 16 Bits memory, two disk drives with 12.5 Mwords 16 Bits storage capacity, two DEC-tapes and one I/O terminal. The multiprocessor system is connected through the unibus [7].

2.2.2. Communication Interface

The communication between the host computer and the multiprocessor system takes place via the communication interface, consisting of a channel-interface unit on the host computer side, and of a special data transmission system - called data-line network - on the multiprocessor system side.

The PDP 11 channel-control logic is a complex microprogrammed unit which can manage simultaneously up to eight independent transfer tasks programmed in the

PDP11/45 host computer. There are four transfer modes possible, depending on two criteria:

- on the transfer direction: read from or write to the PDP11
- on the transfer mode: direct memory access or dialog transfer

The PDP11 channel-control logic contains subunits such as: address counter, word-count register, status register, function registers, protocol-control unit, etc. A detailed description of the PDP11 channel interface can be found in ref. 8.

The data-line network+performs the data transmission between the PDP11 channel interface and the multiprocessor system. In the data-line network each data vector (16 Bits) to be transmitted has an associated identity vector (16 Bits) which contains the coded addresses of the source and destination of the transfer. This identity vector - also called the FROM-TO Address Vector - permits the implementation of nodes in the data-transmission system, where the data are distributed, according to their associated identity vector, to the desired destination. Inside the nodes the data are transmitted in parallel format (2 x 16 Bits) over the so called data-line bus, which is controlled by a synchronous control unit named data-line supervisor. Between the nodes the data are transmitted serially, using a pulse width modulation at a rate of 210 Kwords/s . A detailed description of the data-line network is found in ref. 8.

2.2.3. Multiprocessor System

Fig. 4 shows the detailed structure of the multiprocessor system. It consists of several dedicated processors, which are connected through a high speed asynchronous bus. Looking at optimization of the system, we have found that the number of physically implemented processors can be smaller than the number of processes defined in the process set. For example, it is economical for the video display processor to carry out both frame display and picture processing processes. Table II shows our choice of physically implemented processors with the corresponding processes.

The high-speed bus dynamically switches data paths between pairs of processors, and on the other hand performs the process synchronization in the system. It has a structure which supports the work of the dedicated processors in a real-time environment. It is, therefore, called a "Real-Time-Asynchronous Bus" (RTA Bus). Two frame buffers of 32 k x 16 Bits for the digitized video images

+) Designed by E.L. Bohnen, DESY

are used as common resources of the system by all processors.

2.3. Techniques for Realization

The complexity of the structure as well as the speed requirements prompted the utilization of advanced design tools and concepts.

- Since the interactions between the concurrent processes and sub-processes in the system are very complex, a formal description of their communications between the parallel processes can be given by the *Petri-Net model* [9] . This model is considered as a further development of the state-transition graph which is normally used for description of sequential processes [10] . Here the communication rules are physically represented by the edge transitions of the communication signals. Petri-net representation of the communications rules allows a relatively easy and reliable check for deadlock and unsafe conditions [11] . This yields a valuable tool for testing the correctness of the communication rules initially conceived by the designer.
- An economical and flexible implementation of the dedicated processors has been achieved using the concept of *microprogramming* [12,13] .
- The improvement of processor throughput, necessary for a real-time system, has been achieved using *pipe-line techniques* on both, data- und control-flow levels. The use of this more complex processor structure allows the use of simple, low cost TTL logic throughout the system, in spite of the high throughput requirements.

3. Functional Description of the Components

In this more detailed description we concentrate on the structure and function of the different components of the system: RTA-bus, real-time digitizer, video-display processor, video-symbol generator and communication processor.

3.1. The Real-Time Asynchronous Bus (RTA-Bus)

The RTA-bus has to perform a twofold function: to provide for sharing the communication paths between pairs of processors, and to manage the process synchronisation according to the priorities defined by the user. It has to provide this service for variable processor speeds, especially under the constraints of real-time application. Fig. 5 shows the structure of the implemented bus. As in conventional bus structures [14], each processor is connected to the bus via a communication block. A dialog - i.e. data or instruction transfer - takes place between an active communication block (master) and a passive communication block (slave). The communication blocks are principally masters and/or slaves. Each potential master has to request the bus controller for authorization to use the bus. The bus controller issues a bus grant, which is looped asynchronously over the master blocks in a daisy-chain manner. The master block with the highest priority requesting the bus will receive control. The request from a master block with a higher priority than the currently active master block will interrupt communication over the bus within one bus cycle.

Each communication block contains the following dedicated units:

- The bus request unit, which carries out the dialogue with the bus controller for the permission to use the bus.
- The active transfer unit, which initiates the data transfer as master of the communication process.
- The passive transfer unit, which completes the data transfer as slave of the communication process.
- The dialogue handler, which carries out the bidirectional communication with the kernel. The kernel lies outside the communication block and is user specific. Generally it is a dedicated processor such as real-time digitizer, video-display processor or graphical display processor.

These units carry out simultanously the communication (sub)processes with their environment. The communication rules between the units or the RTA-bus, which are physically represented as the sequence of the signal edge transitions in the bus system, were formally described and verified by Petri-nets. The Petri-net description of the communication rules between the units or the RTA-bus was a decisive help for the quick and consistent design of the bus[+].

[+] A detailed description of Petri-net modelling is being prepared for publication.

Its structure shows the following particular properties:

- By *paralleling* the (sub)processes needed for the bus communication, administration times were minimized (60 ns bus cycle). There are three main processes which are carried out concurrently: requesting of the bus and selection of the master, data transfer over the bus, and dialogue with the kernel. The race problems arising from the attempt for a rapid synchronisation of these processes have been solved using special arbiters +).
- By choosing an *asynchronous transfer* mode an optimum adaption to the characteristic transfer rates has been achieved. The transfer rates depend on the bandwidth of the processors and on the bandwidth of the bus itself, which is 15 Mwords/s.
- The data transfer over the bus occurs wordwise or blockwise. As a decisive property for the real-time application the choice about the transfer mode is done *automatically* depending on the transfer rate of the communicating processors themselves. Processors with a high transfer-rate, such as the real-time digitizer, transfer data in blocks, whereas processors with a low transfer-rate, such as the general purpose computer, transfer data in words.
- The time necessary to grant or release the bus for one of the dedicated processors is not longer than one transfer cycle of the bus. This special property allows extremely short times for the switching of the bus resources between pairs of communicating processors and guarantees the processor actually having the highest priority an *upper limit of the delay time* for bus grant.

This feature was implemented by chaining an auxiliary priority line (PN) over the communication blocks in the RTA-bus, which inhibits the bus need (BN) request of the acutally active communication block (see fig. 5).

The implemented RTA-bus has the following technical properties:

- Bus control technique: daisy chaining, centralized control with guaranteed bus grant acknowledge.
- Bus communication technique: request/acknowledge asynchronous type, fully interlocked, with 60 nsec bus cycle.
- Data transfer technique: single words or variable length blocks.
- Bus width: 40 lines of 120 Ω characteristic impedance.

+) A detailed description of Petri-net modelling is being prepared for publication.

For compatibility with the host computer, the width of the data vector in the RTA-bus is acutally 16 Bits. Our experience with the system has shown that an increase of the data vector to 32 or even 64 Bits would be an improvement resulting in a diminution of the required band width of the frame buffers, which are the major cost factor in the system.

3.2. The Real-Time Digitizer (RTD)

The Real-Time Digitizer (RTD) has to perform the real-time acquisition, digitizing and storage of video images. In its design we aimed at a maximum data reduction immediately at acquisition time. It is achieved by the programmability of the acquisition parameters such as region of interest, time and spatial resolution. Thus only limited regions are digitized with no more spatial and time resolution than necessary.

3.2.1. Structure and Function of the RTD

Fig. 6 shows the structure of the RTD. It consists of several subordinated units:

- An analog to digital converter (conversion time = 66 ns, aperture time = 1 ns) which digitizes the video signal.
- A microprogrammed digitizing unit, which controls the acquisition format and the spatial and time resolution, and performs the synchronisation with the composite video signal.
- A FIFO buffer register, which ameliorates the fluctuation of the incoming data from the digitizing unit and outcoming data to the RTA bus.
- A microprogrammed bus interface unit, which communicates with the RTA bus.

These subordinate units are connected together in a pipe-lined structure on the data-flow level which enables the overlap of the different phases of the picture generation. Effective control is performed by a second pipe-lined structure called the action network, which works on the control-flow level and effects the overlapping of the control phases corresponding to those on the data-flow level [10]. It is hierarchially situated above the data-flow level. On one side the action network exchanges information with the host computer about the beginning and the end of the digitization process. On the other side it coordinates the activities of the subordinated units by simultaneously sending instructions to them and receiving as feedback status information about the data-flow during the digitizing process.

This pipe-lined structure on the both control-flow and data-flow levels allows the non-critical satisfaction of the high throughput requirements of the RTD.

The RTD can be considered as a Mealy automaton with the following specifications:

Input-set X: = Instruction Set

Output-set Y: = The set of the digitized pixels with the associated grey level and storage address for the frame buffer

State-set Z: = The set of all possible contents for the following storage cells, which exist in the RTD:

- IREG: = Instruction Register (16 Bits)
- X_{min}REG: = absolute x-coordinate of the beginning of the window to be digitized (8 Bits)
- X_{max}REG: = absolute x-coordinate of the end of the window to be digitized (8 Bits)
- Y_{min}REG: = absolute y-coordinate of the beginning of the window to be digitized (8 Bits)
- Y_{max}REG: = absolute y-coordinate of the end of the window to be digitized (8 Bits)
- XCNT: = raster-scan counter for the x-coordinate (8 Bits)
- YCNT: = raster scan counter for the y-coordinate (8 Bits)
- SFF: = Start Acquisition Flipflop
- FFF: = Status Flipflop for the spatial resolution
- GFF: = Status Flipflop for the full picture acquisition format (i.e. 256 x 256)
- MMREG: = Status for the selection of the frame buffer as destination of the digitized frame (2 Bits)

STREG: = Status Register, which contains information about the current activities of the RTD and error conditions detected during the digitization process.

The behaviour of the RTD as a synchronous processor is described by the state-transition function δ:

$$\underline{Z}^{n+1} = \delta(\underline{Z},\underline{X})^n \qquad (3.1)$$

and by the output function ω:

$$\underline{Y}^n = \omega(\underline{Z},\underline{X})^n \qquad (3.2)$$

where n is the index of the discrete time of the automaton. Both equations are described in Table III, which is ordered by the elements of the instruction set X.

The practical experience with the DVS has shown that also the grey level resolution should be programmable. This would permit a better use of the frame buffers according to user's requirements for spatial, time and grey level resolution. For example, the grey level resolution can be reduced in favour of the time resolution. Fig. 7 shows an example for the digitization accuracy of a video image digitzed with low (5 M Samples/s) and high (10 M Samples/s) spatial resolution in comparison with the original analog picture.

3.2.2. Programming the RTD

According to the functional description of the RTD, the instruction set of the RTD contains three classes of instructions:

- Instructions controlling the acquisition format. The user can select any rectangular window with $0 \leq x \leq 255$ and $0 \leq y \leq 255$ for the acquisition. This feature allows data reduction at acquisition time.
- Instructions controlling the spatial resolution (sampling rates of 5 MHz or 10 MHz) and the time resolution (up to 50 frames/s).
- Instructions controlling the destination of the digitized data.

The code of the instruction set and of the status register is described in Table IV.

The semantics of the status register is the following:

DONE = 0 Beginning of the window to be digitized
DONE = 1 End of the window to be digitized
PAR = 1 Parity error during the transfer of data
SP = 1 Frame buffer overflow due to incorrect programming of the acquisition format and spatial resolution (digitized pixels > 64 k)
ZP = 1 FIFO buffer register overflow
ERR = 1 General error. The ERR is defined as ERR = PAR v SP v ZP
BUSY = 1 RTD is active (waits for the beginning of the video frame or digitizes the actual video frame)

The user has to take care of the following special features:

- The status register is sent at the beginning of the acquisition of a frame (DONE = 0) and at the end of acquisition (DONE = 1)
- The organisation and capacity of the frame buffers (32 k x 16 Bits) lead to the following limitations during the practical application:
  - Since two digitized pixels are necessary to fill a word of the frame buffer

    $X_{max}REG - X_{min}REG$ = odd number

    must be always valid.
  - Since the maximum storage capacity of the frame buffers is 64 K pixels for high spatial resolution (F = 1)

    $X_{max}REG - X_{min}REG \leq 128$

    has to be fulfilled.

3.3. The Video-Display Processor (VDP)

The Video Display Processor (VDP) is used for the display of the data of the frame buffers on the colour video monitor. The frame buffers can contain video frames generated by the real-time digitizer and/or graphical frames generated by the graphical display processor (see Section 3.4.). To give an easily interpretable optical presentation for the different semantics of the data a trans-

formation into a grey level and/or colour code is necessary. Thus the main feature of the VDP is its *programmable processing facility*, which provides these transformations in *real time*.

3.3.1. Structure and Function of the VDP

Fig. 8 shows the architecture of the video display processor. It is a complex microprogrammed processor, which can be decomposed into specialized units such as:

- A display format control unit, which controls the rectangular window to be displayed on the colour video monitor.
- A RTA-bus interface unit, which manages the communication with the RTA-bus.
- A FIFO buffer register, which compensates for the fluctuation of the incoming data from RTA-bus and outgoing data to video monitor.
- The programmable processing unit, which carries out real-time transformations on the data to be displayed.
- A video-interface, which connects the digital part of the video display processor with the video colour monitor. It contains the digital analog converters with 40 MHz bandwidth and the circuits for generating the composite-video signal.
- A light-pen and cursor control and interpretation unit, which supports interactive communication between user and system.

On the data-flow level the VDP features a pipe-lined structure, necessary to manage the high data rates occuring during the display process. For example, the arithmetic logical operations, necessary for the picture transformation, are simultaneously performed by the different sections of the Programmable Processing Unit (PPU). In the following discussion we will concentrate on the structure of the PPU. The implemented structure is shown in Fig. 9.

For the realization of the processing we chose the technique of look up tables. Here the transformation function from frame buffer contents to grey levels or colours is written into a RAM in tabular form. As the transformation function is often composed of two functions (e. g. contrast enhancement and colour assignment) we chose for reasons of transparency a cascade of two look-up tables (W-RAM, C-RAM). Monadic and dyadic operations are applicable: Monadic operations work on one of the frame buffers or the concatenation of both:

OPERATION(BUFFER N) $\Longrightarrow$ PICTURE
OPERATION(BUFFER1_BUFFER2) $\Longrightarrow$ PICTURE

A typical monadic operation on one buffer is a grey-level transformation. As an example Fig. 10 shows a linearly digitized picture (a) in comparison with its logarithmic (b) and its exponential transformation (c). Obviously the logarithmic transformation enhances the low grey levels, whereas the exponential transformation enhances the high grey levels of the displayed picture.

Monadic operations on one frame-buffer are also used for pseudo-colour assignment. It performs a transformation of the frame buffer content into the colour domain. Fig. 11 shows a functional image [15] of a kidney. In this case the frame buffer content was a map of the time blood takes to reach each point of the kidney. The *pseudo-colour* representation enhances the information contained in the picture and, therefore, helps for easier interpretation.

As an example of a monadic operation on the concatenation of both buffers one can write the picture intensity into one buffer and the associated colour into the other. The appropriate function in the look-up tables would provide the true-colour coded picture on the TV screen. Fig. 12 shows a true-colour picture of a carpet.

The dyadic operations work on the two frame-buffers.

(BUFFER1)OPERATION(BUFFER2) $\Longrightarrow$ PICTURE

As an example one may fill the RAM with a function which performs a subtraction of the two pictures, as shown in Fig. 17.

As the look-up tables can be updated up to 50 times/s, a dynamical optical interpretation of the picture data is possible, which even allows the presentation of moving scenes. Fig. 13 shows two phases in emulating blood propagation in the kidney using a functional image of the type described above.

The VDP can be considered as a Mealy-automaton with the following specifications:

Input Set $\underline{X}$ = Instruction set.

Output Set $\underline{Y}$ = The set of pixels displayed on the colour video monitor (max. 256 x 256 pixels). Each pixel is defined as a triple $\underline{P}_j$ (R_j, G_j, B_j) where the R_j, G_j and B_j are the values of the red, green and blue colour primitives.

State Set $\underline{Z}$ = The set of the all possible contents for the following storage cells, which exist in the VDP:

IREG = Instruction Register (16 Bits).

X_{min}REG = Absolute x-corrdinate of the beginning of the window to be displayed (8 Bits).

X_{max}REG = Absolute x-coordinate of the end of the window to be displayed (8 Bits).

Y_{min}REG = Absolute y-coordinate of the beginning of the window to be displayed (8 Bits)

Y_{max}REG = Absolute y-coordinate of the end of the window to be displayed (8 Bits).

XCNT = Raster-scan counter for the x-coordinate (8 Bits).

YCNT = Raster-scan counter for the y-coordinate (8 Bits).

WRAM = Look-up table: 256 Words x 8 Bits.

WADRCNT = W-RAM-address counter.

CRAM = Look-up table: 4096 Words x 9 Bits.

CADRCNT = C-RAM-address counter.

SFF = Start-display flipflop.

FFF = Status flipflop for spatial resolution.

GFF = Status flipflop for full picture-display format (i.e. 256 x 256)

MMREG = Status for the selection of the frame buffers as source of the displayed data (2 Bits).

DMREG = Status for the display mode (2 Bits).

LPREG = Status for the light-pen interaction (2 Bits).

CURREG = Cursor register (16 Bits).

The state-transition function δ and the output function ω, which describe the behavior of the VDP, are contained in TABLE V.

The following features should be noticed:

- The display format code is identical to that of the RTD. Programming is thus facilitated.
- The limitations due to organisation and capacity of the frame buffers described for the RDT (see 3.2.2.) are also valid for the VDP.
- The instructions concerning the display mode and the frame buffer selection (see No. 11 and 12 on TABLE V) are dependent on each other. Therefore, there is a limited number of meaningful combinations, which are listed in TABLE VI.
- The semantics of the data contained in the both frame buffers depends on the user's choice, except Bit DØØ and DØ1 which are dedicated for blink and intensify mark, respectively. For true colour display mode the frame buffer 1 could contain the luminance information and frame buffer 2 the chrominance information.

3.3.2. Programming the VDP

With respect to the functional description of the VDP, the instruction set of the VDP contains four classes of instructions:

- Instructions controlling the display format: any rectangular window with $0 \leq X \leq 255$ and $0 \leq Y \leq 255$ can be selected by the user.
- Instructions controlling the spatial resolution: (256 or 512 pixels/line).
- Instructions for filling the look-up tables.
- Instructions controlling the data paths.

TABLE VII contains the implemented code for the instruction set of the VDP.

3.4. The Video-Symbol Generator (VSG)

The Graphical Display Processor (GDP) mentioned in Chapter 2 is to support the graphical presentation of the results. It generates a graphical frame which is stored in one of the frame buffers and displayed by the VDP. The GDP contains picture primitive generators in particular character and vector generators. It has the capability of carrying out Euclidian transformations such as translation and rotation, and domain transformations such as clipping and windowing. In contrast to the previously described processors, the GDP is designed but not yet implemented. Presently we use, as a preliminary solution, the Video Symbol Generator (VSG) to perform a subset of the graphical presentation, which will be done finally by the GDP. The VSG only has the capability of displaying alphanumerical information on the colour video monitors.

3.4.1. Structure and Function of the VSG

Fig. 14 shows the structure of the VSG. The alphanumerical information to be displayed is contained in the page buffer (2K Words x 16 Bits). The page buffer can be accessed over the data path multiplexer from two sources:

- On one side from the CRT controller, which reads the page buffer and performs the data processing necessary to achieve the required optical representation on the colour-video monitor. The CRT controller contains subunits such as character generator, colour generator and video interface.
- On the other side, by means of the I/O interface, from the host computer which updates the alphanumerical information from the page buffer. The updating takes place only during the vertical retrace time of the video beam, i. e. for the European CCIR standard a time slice of 1.5 ms which appears every 1/50 s.

Since the user communicates over the I/O interface only with the page buffer and not with the CRT controller, a functional description of the VSG can neglect the CRT controller itself.

The VSG exposes the following technical properties:

- Display format:
  The monitor screen contains a matrix of 2048 fields, organized in 32 rows/frame with 64 symbol-fiels/row.

- Symbol format:
  Each field contains a symbol. The symbol has a 5 x 7 matrix representation.
- Symbol repertoire:
  There are 128 symbols, coded in the US-ASCII-Code (7 Bits/symbol).
- Optical attributes of the field:
  - 8 colours for the symbol.
  - 8 colours for the background.
  - 4 gray levels for the field (common for the symbol and background).
- Mark attributes:
  Each field can be marked by blinking with a frequency of 2 Hz.

The alphanumerical frame produced by the VSG is mixed in an analog way with the picture frame produced by the VDP and displayed on the video monitor.

Fig. 15 shows as an example of a VSG output an assembler listing of the SIMPL11 programming language [16] using the VSG. In this example optical attributes such as symbol and background colours and intensity have been used to mark lexical and syntactical properties of the program.

The VSG can be formally described as a Mealy automaton with the following specifications:

Input set $\underline{X}$:= The instruction set

Output set $\underline{Y}$:= The set of all data written in and/or read from the page buffer

State set $\underline{Z}$:= The set of all possible contents for the following storage cells which exist in the VDP
IREG: = Instruction register
ADRCNT:= Address counter
RDREG: = Read-data register
WDREG: = Write-data register
W/$\overline{R}$FF: = Write/$\overline{\text{read}}$ flipflop

The state-transition function δ (see 3.1.) and the output function ω (see 3.2.) are described in TABLE VIII.

3.4.2. Programming the VSG

For programming the VSG the programmer just has to choose the position on the TV screen by setting the appropriate address count and the read or write instructions according to TABLE IX.

The symbols have the following meaning:

M	Mark bit in the data-line system
(A10,...A0)	ADRCNT vector
(I1,I0)	Grey level for the selected field
(B_S,G_S,R_S)	Colour of the symbol in the field
(B_H,G_H,R_H)	Colour of the background of the field

both with the code

B	G	R	colour
0	0	0	black
0	0	1	red
0	1	0	green
1	0	0	blue
1	0	1	cyan
1	1	0	violet
1	1	1	white

B = 1 Blink

3.5. Communication Processor (CP)

The Communication Processor (CP) performs the transfer of data and instructions between the DVS and the host computer. It is the only means of communication of the DVS with its environment. Further attention will be paid to structure, function and programming of the CP.

3.5.1. Structure and Function of the CP

Fig. 16 shows the structure of the CP. It consists mainly of two independent controllers used for write operations into the DVS and for read operations from the DVS respectively. Using two separate controllers permits a better adaption of the bandwidth of the RTA-bus to the unibus of the host computer. Each controller contains the following units:

- Interface logic, which controls the communication with the host computer.
- Operational registers such as instruction register, address counter, word-count register and write and read-data register.
- Control logic, which controls the transfer protocol.

The communication with the RTA-bus takes place over a multiplexer, which is controlled by the multiplexer-control unit. It decides which controller's bus request is to be switched to the RTA-bus.

For the functional description of the CP we will concentrate on the write and read controllers.

The user can consider the write controller as a Mealy automaton with the following specifications:

Input Set $\underline{X}$: = Instruction set.

Output Set $\underline{Y}$: = The set of control signals, data and address vectors, which are created for each instruction.

State Set $\underline{Z}$: = The set of all possible contents of the following storage cells, which exist in the write controller:

IREG : = Instruction register (32 Bits)
ADRCNT:= Address counter for the address space of the RTA-bus (16 Bits)
WDREG : = Write-data register (16 Bits)
MMREG : = Frame-buffer select register (2 Bits)

The state-transition function δ (see 3.1.) and the output function ω (see 3.2.) are described in TABLE X.

Similarly the read controller can be considered as a Mealy automaton with the following specifications:

Input Set $\underline{X}$: = Instruction set.

Output Set $\underline{Y}$: = The set of control signals, data and address vectors, which are created for each instruction.

State Set $\underline{Z}$: = The set of all possible contents of the following storage cells, which exist in the read controller:

IREG : = Instruction register (32 Bits)

ADRCNT: = Address counter for the address space of the RTA-bus (16 Bits)

WCNT: = Word-count register, for controlling the block length during the transfer

FRTOREG: = "FROM-TO" address register, which contains the address of the destination module for the data to be read (16 Bits)

$\overline{S}$/PFF: = "Serial-Parallel" flipflop, which controls whether the data read are to be sent serially or parallel to the UNIBUS.

MMREG : = Frame-buffer select register (2 Bits)

RDREG : = Read-data register (16 Bits)

State transition and output functions are contained in TABLE XI.

3.5.2. Programming the CP

The instruction sets of the write and read controllers are presented with their codes in TABLE XII and TABLE XIII, respectively.

4. Fields of Application

The system described offers itself for any application where multitemporal scenes proceeding at the video picture rate have to be analyzed in a fast and/or interactive way. Our special application deals with the analysis of the contrast medium flow in an organ, which is viewed by the image identifier video equipment of an x-ray device [15] .

The real time digitization facility offers the physician the ability quickly to digitize selected regions of interest with the required time and space resolution. The graphic properties of the system enable him simply to point to interesting blood vessels and to call analysis programs such as those for the computation of blood flow. If experience in routine use shows that they take

too long, the structure of the system allows the easy addition of a special processor carrying out the algorithm in hardware. The flexibility and speed of the picture and graphical data output finally allow a presentation of results which is easily interpretable by the physician.

The flexibility and the speed of the system allow the analysis of a large number of x-ray picture series in a reasonable time. This is necessary for getting significant medical results.

5. Conclusions

Design and implementation of a multiprocessor system for the real-time digital processing of video images have been described.

The architecture of the system, which is based on the concept of distributed intelligence, has the following features:

- It enables a high degree of *parallel processing* using dedicated processors. For example the real-time digitizer can digitize a video image and at the same time the communication processor can read a part of the digitized frame at the transfer speed of the general purpose computer.
- The process synchronization is done *automatically* by the RTA-bus according to the processor priorities defined by the user. For example the video display processor will be automatically interrupted by the acquisition of a new video image and afterwards restarted without any specific action of the user.
- The *throughput* can be improved by moving processes from software to firmware or hardware. New processors can be added directly to the RTA-bus without changes in the original configuration. For example, a two-dimensional filtering of the digital video image, actually done by the general purpose computer, can be substantially speeded up by adding a special Fast-Fourier-Transformation Processor (FFTP) to the RTA-bus.
- The processors, e.g. real-time digitizer, video-display processor etc. are fully programmable at both macro- and micro-instruction levels. *Efficient programming* of the processors is therefore facilitated.
- The system is *portable* because it has a single I/O communication port with its environment (communication processor).

- Using *structured design* methods throughout the system we have achieved improved transparency of the design, better documentation, shorter implementation times and, last but not least, superior resulting products.

Working with the system we have learned that some additional features, such as programmable grey-level resolution and increased size of the data vector in the RTA-bus, should favourably influence the performance/cost ratio of the system. We intend to redesign the system with this in mind.

The development of the system was started in 1976 and it is scheduled to be completed at the end of 1978. Although the system has been conceived for application in a clinical environment, the implemented system is suitable for a large number of other application fields.

6. Acknowledgment

The authors wish to thank Profs. H. Schopper and G. Weber (DESY) for their continuous support of the project, to M. Böhm, Dr. W.-R. Dix, W. Ebenritter, G. Pfeiffer, V. Riemer, Dr. B.Sonne and V. Wolf (DESY), for their valuable contributions during the definition and testing phase, and to Prof. S. Wendt (University of Kaiserslautern) for many stimulating discussions concerning the design philosophy. The authors also appreciate the efforts of K. Schmoeger, R. Siemer and V. Fischer in preparing the camera-ready version of this text.

7. References

1. K. H. Höhne, G. Nicolae, G. Pfeiffer, W.-R. Dix, W. Ebenritter, D. Novak, M. Böhm, B. Sonne, E. Bücheler; "An Interactive System for Clinical Application of Angiodensitometry", Digital Image Processing, Informatik-Fachberichte Vol.8, Berlin-Heidelberg-New York: Springer, 1977, pp.232-243.

2. K. H. Höhne, G. Pfeiffer; "The Role of the Physician-Computer Interaction in the Acquisition and Interpretation of Scintigraphic Data", Meth. Inform. Med. Vol. 13, pp. 65-70, April 1974.

3. N. R. Silverman, "Videometry of Blood Vessels", Radiology, Vol.101, pp. 597-604, December 1971.

4. R. A. Robb, S. A. Johnson, J. F. Greenleaf, M. A. Wondrow and E. Wood; "An operator-interactive, computer-controlled system for high fidelity digitisation and analysis of biomedical images", Proc. Soc. Photo-Optical Instrumentation, Vol.40, pp.11-26, Aug. 1973.

5. J. S. Read, R. T. Borovec, R. C. Amendola, A. C. Petersen, M. H. Goldbaum, M. Kottow, B. H. McCormick and M. F. Goldberg, "The Television Ophthalmoscope Image Processor", Proc. of the IEEE workshop on Picture Data Description and Manangement, Chicago, pp. 64-67, April 1977.

6. B. K. Gilbert, M. T. Storma, C. E. James, L. W. Hobrock, E. S. Yang, K. C. Ballard, E. H. Wood, "A Real Time Hardware System for Digital Processing of Wide Band Video Images", IEEE Trans. Comput., Vol.C25, pp.1089-1100, Nov. 1976.

7. PDP11 Peripherals and Interfacing Handbook, Digital Equipment Corporation, 1971.

8. S. Wendt, F. Hüller, "Data-Line Kanalwerk für die PDP11", Internal Report No.22, Institute for Computer Science, Hamburg, Nov. 1975.

9. C. A. Petri, "Kommunikation mit Automaten", Ph.D. Dissertation, Darmstadt, 1962.

10. S. Wendt, "Petri-Nets and Asynchronous Sequential Circuits", Elektronische Rechenanlagen, Vol.16, pp. 208-216, 1974.

11. J. L. Peterson; "Petri-Nets", ACM Computing Surveys Vol. 9, No. 3, September 1977.

12. S. S. Husson, "Microprogramming: Principles and Practices", Englewood Cliffs: Prentice Hall, 1970.

13. S. Wendt, "On Structures of Microprogramm Control Units", Elektronische Rechenanlagen , Vol.13, pp.22-26, 1971.

14. K. J. Thurber, E. P. Jensen, L. A. Jack, "A Systematic Approach to the Design of Digital Bussing Structures", in 1972 Fall Joint Computer Conference, AFIPS Conf. Proc. Vol.41, part II, pp. 719-740.

15. K. H. Höhne, M. Böhm, W. Erbe, G. C. Nicolae, G. Pfeiffer, B. Sonne, "Functional Imaging - A New Tool for X-Ray Functional Diagnostics", DESY-Report Nr. DESY DV-78/01, 1978.

16. G. Pfeiffer, "Simpl 11, eine einfache Implementierungssprache für PDP11-Rechner", DESY Report Nr. DESY DV-76/02, 1976.

8. Figure Captions

Fig. 1 A frame of an X-ray picture series showing a kidney

Fig. 2 Hierarchical model of the Digital Video System

Fig. 3 Actual configuration of the Image Processing System

Fig. 4 Basic structure of the Digital Video System

Fig. 5 Structure of the RTA-bus

Fig. 6 Structure of the Real Time Digitizer

Fig. 7 Video presentation of a picture as
a) an analog image
b) a digitized image with 5 M samples/s
c) a digitized image with 10 M samples/s (section)

Fig. 8 Structure of the Video Display Processor

Fig. 9 Basic structure of the Programmable Processing Unit

Fig. 10 Example of gray level transformations
a) linearly digitized image
b) logarithmic transformation of image a)
c) exponential transformation of image a)

Fig. 11 Example of a pseudo-color image

Fig. 12 Example of a true-color image

Fig. 13 Two phases of the real-time playback of the blood propagation in the kidney

Fig. 14 Structure of the Video Symbol Generator

Fig. 15 Example of using optical attributes for a better transparency of program structure

Fig. 16 Structure of the Communication Processor

Fig. 17 Example of a real time arithmetic operation performed by the programmable precessing unit.
a) object with background
b) background
c) subtraction of the images a) and b)

TABLE I Set of Processes

Process Name	Semantics	Priority	Flow Speed	Realization
Picture-frame generation	Picture - acquisition digitizing transfer to frame-buffer storage	1	real-time	hardware + firmware
Frame display	Frame data (i.e. picture and/or graphic) - transfer from frame buffer - display in raster-scan format	4	real-time	hardware + firmware
Picture processing (for viewing)	Picture transformation for - enhancement (e.g. gray-level manipulation) - arithmetic logical operations (e.g. marking, subtraction) - marking (e.g. colour coding)	3	real-time	firmware
Graphic-frame generation	Generation of graphical primitives as point, vector and area Transformation of graphical items - euclidian: translation rotation scaling - domain: windowing clipping	4	partially real-time + partially inter-active	hardware + firmware
DVS Communication with environment	Exchange of data and control-information with the environment	2	real-time	hardware + firmware
User Communication with DVS	Exchange of data and control information with DVS	2	real-time	software + hardware
Picture analysis	User programmable processing - functional images - blood velocity - blood flow	2	inter-active	software
Data management	Retrieval of pictures	2	inter-active	software

TABLE II Process Set Versus Processor Set

Process / Processor	Frame Generation	Frame Display	Picture-processing	Graphic-Frame Generation	User Communication	DVS Communication	Picture Analysis	Data Management
Real-Time Digitizer	X							
Video-Display Processor		X	X					
Graphic Display Processor				X				
Communication Processor						X		
General Purpose Computer (e.g.PDP11)			X	X	X		X	X
Special Processor (e.g. FFTP)							X	

TABLE III Description of the RTD Instructions

No.	Instruction $\underline{X}^n$	State-Transition Function $\underline{Z}^{n+1} <= \underline{Z}^n$	Output Function $\underline{Y}^n$
1	SET X_{min}-REGISTER	X_{min}REG<= IREG ($2^7 \div 2^0$)	no output
2	SET X_{max}-REGISTER	X_{max}REG<= IREG ($2^7 \div 2^0$)	no output
3	SET Y_{min}-REGISTER	Y_{min}REG<= IREG ($2^7 \div 2^0$)	no output
4	SET Y_{max}-REGISTER	Y_{max}REG<= IREG ($2^7 \div 2^0$)	no output
5	SET FULL PICTURE FORMAT	GFF <= IREG (2^2 = 1) X_{min}REG<= Ø X_{max}REG<= 255 Y_{min}REG<= Ø Y_{max}REG<= 255 FFF <= Ø	no output
6	SET RESOLUTION	FFF <= IREG (2^1) with FFF = Ø = 256 pixels/scan line FFF = 1 = 512 pixels/scan line	no output
7	SELECT MEMORY	MMREG <= IREG (2^4,2^3) with IREG(2^4)=1:= SELECT FRAME BUFFER 2 IREG(2^3)=1:= SELECT FRAME BUFFER 1	no output
8	START ACQUISITION	SFF <= IREG (2^0 = 1) j[XCNT <= XCNT + 1] j = Ø to 255 k[YCNT <= YCNT + 1] k = Ø to 255 STREG = current value	(1) Sequence of the digitized pixels, which lie in the programmed window on the screen, with gray level value and associate frame buffer address. (2) Send the status register at the beginning and at the end of the window to be digitized.

TABLE IV Code of the RTD Instructions

Instruction	Bit No. 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Ø
SET X_{min}REG	1	Ø	Ø						x_7	x_6	x_5	x_4	x_3	x_2	x_1	$x_Ø$
SET X_{max}REG	1	Ø	1						X_7	X_6	X_5	X_4	X_3	X_2	X_1	$X_Ø$
SET Y_{min}REG	1	1	Ø						y_7	y_6	y_5	y_4	y_3	y_2	y_1	$y_Ø$
SET Y_{max}REG	1	1	1						Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	$Y_Ø$
CONTROL INSTR	Ø	-	-									MMREG		GFF	FFF	SFF
STATUS REGISTER	DONE	ERR	ZP	SP	PAR						BUSY					

TABLE V Description of the VDP Instructions

No.	Instruction $\underline{X}^n$	State Transition Function $\underline{Z}^{n+1}$ <= $\underline{Z}^n$	Output Function $\underline{Y}^n$
1	SET X_{min}-REGISTER	X_{min}REG <= IREG $(2^7 \div 2^0)$	no output
2	SET X_{max}-REGISTER	X_{max}REG <= IREG $(2^7 \div 2^0)$	no output
3	SET Y_{min}-REGISTER	Y_{min}REG <= IREG $(2^7 \div 2^0)$	no output
4	SET Y_{max}-REGISTER	Y_{max}REG <= IREG $(2^7 \div 2^0)$	no output
5	WRITE FIRST WORD IN WRAM	WADRCNT <= Ø WRAM | Ø <= IREG $(2^7 \div 2^0)$ WADRCNT <= WADRCNT + 1	no output
6	WRITE WRAM	WRAM | WADRCNT = j; j=0÷255 <= IREG $(2^7 \div 2^0)$ WADRCNT <= WADRCNT + 1	no output
7	WRITE FIRST WORD IN CRAM	CADRCNT <= Ø CRAM | Ø <= IREG $(2^8 \div 2^0)$ CADRCNT <= CADRCNT + 1	no output
8	WRITE CRAM	CRAM | CADRCNT = j; j=Ø÷4095 <= IREG $(2^8 \div 2^0)$ CADRCNT <= CADRCNT + 1	no output

TABLE V (continued)

No.	Instruction $\underline{X}^n$	State-Transition Function $\underline{Z}^{n+1}$ <= Z^n	Output Function $\underline{Y}^n$
9	SET FULL PICTURE FORMAT	GFF <= IREG (2^2 = 1) X_{min}REG <= Ø X_{max}REG <= 255 Y_{min}REG <= Ø Y_{max}REG <= 255 FFF <= Ø	no output
10	SET RESOLUTION	FFF <= IREG (2^1) with: FFF=Ø :=256 pixels/scan line FFF=1 :=512 pixels/scan line	no output
11	SELECT FRAME BUFFER	MMREG <= IREG (2^4, 2^3) For the semantics of the code see TABLE VI	no output
12	SELECT DISPLAY MODE	DMREG <= IREG (2^6, 2^5) For the semantics of the code see TABLE VI	no output
13	SELECT LIGHT-PEN STATUS	LPREG <= IREG (2^8, 2^7) where IREG (2^7):= VISUAL IREG (2^8):= TRACK and if VISUAL = 1 CURREG <= (YCNT, XCNT)	if VISUAL = 1, CURSOR intensified if TRACK = 1, CURREG will be sent to the host computer

14	START DISPLAY	SFF <= IREG (2^0 = 1)
		j(XCNT <= XCNT + 1);
		j = 1 to 255
		k(YCNT <= YCNT + 1);
		k = 1 to 255
			Frame Display

TABLE VI List of Meaningful Display Modes

No.	DMREG		MMREG		Display Modus
	L1	LØ	MM2	MM1	(see Fig. 9)
1	Ø	Ø	Ø	1	Black/white display mode for the data contained in the frame buffer 1 Data transformations: WRAM : D2 = W(D1); D2 = (W7, W6, ... WØ) D3 = D2 CRAM : D4 = C(D3); D4 = (C7, C6, ... CØ) RED = GREEN = BLUE := D4
2	Ø	Ø	1	Ø	Black/white display mode for the data contained in the frame buffer 2 Data transformations: see 1
3	Ø	1	Ø	1	Pseudocolour display mode for the data contained in frame buffer 1 WRAM : D2 = W(D1); D2 = (W7, W6, ... WØ) D3 = D2 CRAM : D4 = C(D3); D4 = (C8, C7, ... CØ) RED: =(C2, C1, CØ) GREEN: =(C5, C4, C3) BLUE: =(C8, C7, C6)
4	Ø	1	1	Ø	Pseudocolour display mode for the data contained in the frame buffer 2 Data transformations: see 3
5	1	1	1	1	True colour-display mode for the data - contained in the frame buffers 1 and 2 Data transformations: WRAM : D2 = W(D1); D2 = (W7, W6, ... WØ) D3 = (DØ7, DØ6, ... DØ2, W7, W6, ... W2) CRAM : D4 = C(D3); D4 = (C8, C7, ... CØ) RED: =(C2, C1, CØ) GREEN: =(C5, C4, C3) BLUE: =(C8, C7, C6) DØØ = 1, BLINK MARK DØ1 = 1, INTENSIFY MARK

TABLE VII Code of the VDP Instructions

No.	Instruction	Bit No. 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	SET X_{min} REG	1	0	0						x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
2	SET X_{max} REG	1	0	1						X_7	X_6	X_5	X_4	X_3	X_2	X_1	$X_\emptyset$
3	SET Y_{min} REG	1	1	0						y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
4	SET Y_{max} REG	1	1	1						Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	$Y_\emptyset$
5	WRITE FIRST WORD WRAM	0	1	0	1					W_7	W_6	W_5	W_4	W_3	W_2	W_1	W_0
6	WRITE WRAM	0	1	0	0					W_7	W_6	W_5	W_4	W_3	W_2	W_1	W_0
7	WRITE FIRST WORD CRAM	0	1	1	1				C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
8	WRITE C. RAM	0	1	1	0				C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
9	CONTROL INSTRUCTION	0	0	-	-			Track		Visual		DMREG		MMREG		GFF	FFF SFF
	CURREG	YCNT								XCNT							

TABLE VIII Description of the VSG Instructions

No.	Instruction $\underline{X}^n$	State-Transition Function $\underline{Z}^n$ <= $\underline{Z}^{n+1}$	Output Function $\underline{Y}^n$
1	SET ADRCNT	ADRCNT <= IREG (2^{10} ÷ 2^0) W/$\overline{R}$FF <= IREG (2^{11})	no output
2	WRITE	PAGE BUFFER | ADRCNT <= IREG (2^{15} ÷ 2^0) ADRCNT <= ADRCNT + 1	no output
3	READ	RDREG <= PAGE BUFFER | ADRCNT ADRCNT <= ADRCNT + 1	character in RDREG

TABLE IX Code of the VSG Instructions

No.	Instruction	Bit No. M 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 Ø
1	SET ADRCNT	1 W/$\overline{R}$ A10 A9 A8 A7 .....AØ
2	WRITE (W/$\overline{R}$=1)	0 B_H G_H R_H B_S G_S R_S I1 IØ B US-ASCII
3	READ (W/$\overline{R}$=Ø)	D U M M Y

TABLE X Description of the Write-Controller Instructions

No.	Instruction $\underline{X}^n$	State-Transition Function $\underline{Z}^{n+1}$ <= $\underline{Z}^n$	Output Function $\underline{Y}^n$
1	SET ADRCNT	ADRCNT <= IREG ($2^{15} \div 2^0$)	no output
2	SET MEMORY SELECT REGISTER	MMREG <= IREG (2^1,2^0) with IREG(2^1) = 1 := SELECT FRAME BUFFER 2 IREG(2^0) = 1 := SELECT FRAME BUFFER 1 IREG(2^1,2^0)= 1 := SELECT FRAME BUFFER 1 und 2	no output
3	WRITE DATA	j(ADRCNT <= ADRCNT + 1) with j = 1 ÷ WCNT + 1	ADRCNT =>RTA BUS ADR WDREG =>RTA BUS DATA control signals for WRITE

TABLE XI Description of the Read-Controller Instructions

No.	Instruction $\underline{X}^n$	State Transition Function $\underline{Z}^{n+1}$ <= $\underline{Z}^n$	Output Function $\underline{Y}^n$
1	SET ADDRESS COUNTER	ADRCNT <= IREG ($2^{15} \div 2^0$)	no output
2	SET WORD COUNTER	WCNT <= IREG ($2^{15} \div 2^0$)	no output
3	SET FRTO REGISTER	FRTOREG <= IREG ($2^{15} \div 2^0$)	no output
4	SET MM REG + SET $\overline{S}$/P FLIPFLOP + START READ	MMREG <= IREG (2^1,2^0) with IREG (2^1) = 1 : SELECT FRAME BUFFER 2 IREG (2^0) = 1 : SELECT FRAME BUFFER 1 $\overline{S}$/PFF <= IREG (2^2) with $\overline{S}$/PFF = Ø : SERIAL TRANSFER $\overline{S}$/PFF = 1 : PARALLEL TRANSFER j(ADRCNT <= ADRCNT + 1) with j = 1 ÷ WCNT + 1 RDREG <= RTA-BUS DATA	ADRCNT => RTA BUS ADR CONTROL SIGNALS FOR READ

TABLE XII Code of the Write-Controller Instructions

No.	Instruction	Operation Code	Operand Code
		AØ M E6 E5 E4 E3 E2 E1 EØ	D15 D14 ... D3 D2 D1 DØ
1	SET ADRCNT	- Ø - - - - - Ø 1	ADR FOR THE RTA BUS
2	SET MMREG	- 1 - - - - - - - -	- - MMREG
3	WRITE DATA	- Ø - - - - - Ø Ø	DATA TO BE WRITTEN

TABLE XIII Code of the Read-Controller Instructions

No.	Instruction	Operation Code	Operand Code
		M E6 E5 E4 E3 E2 E1 EØ	D15 D14 ... D3 D2 D1 DØ
1	SET ADRCNT	Ø - - - - - Ø 1	ADR FOR RTA BUS
2	SET WCNT	Ø - - - - - 1 Ø	WCNT
3	SET FRTOREG	Ø - - - - - 1 1	"FROM TO ADR" REG
4	CONTROL INSTRUCTION	1 - - - - - - -	$\overline{S}$/PFF MMREG

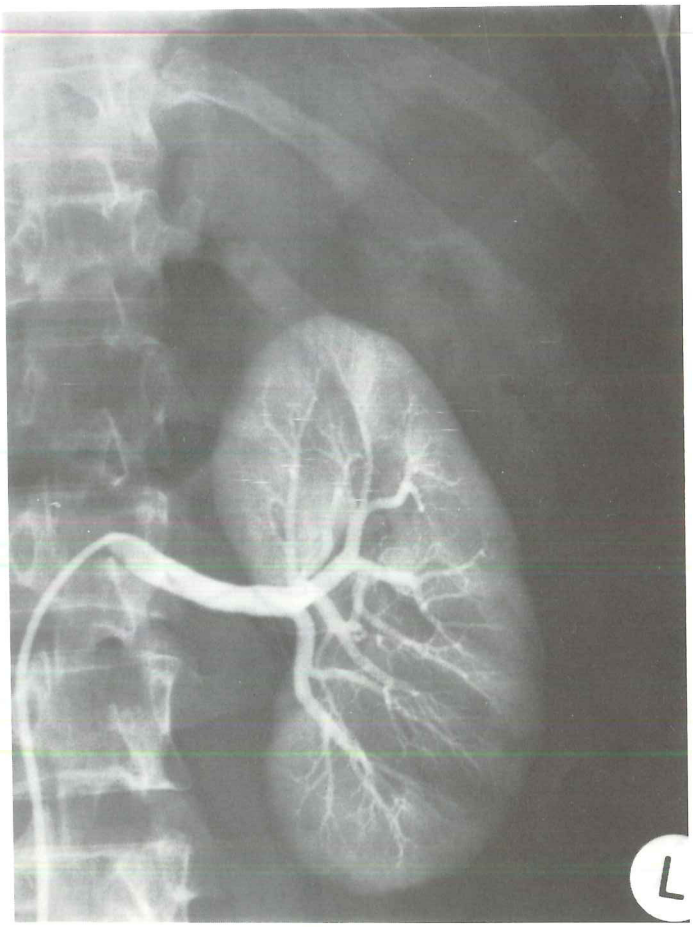


Fig. 1 A frame of an X-ray picture series showing a kidney

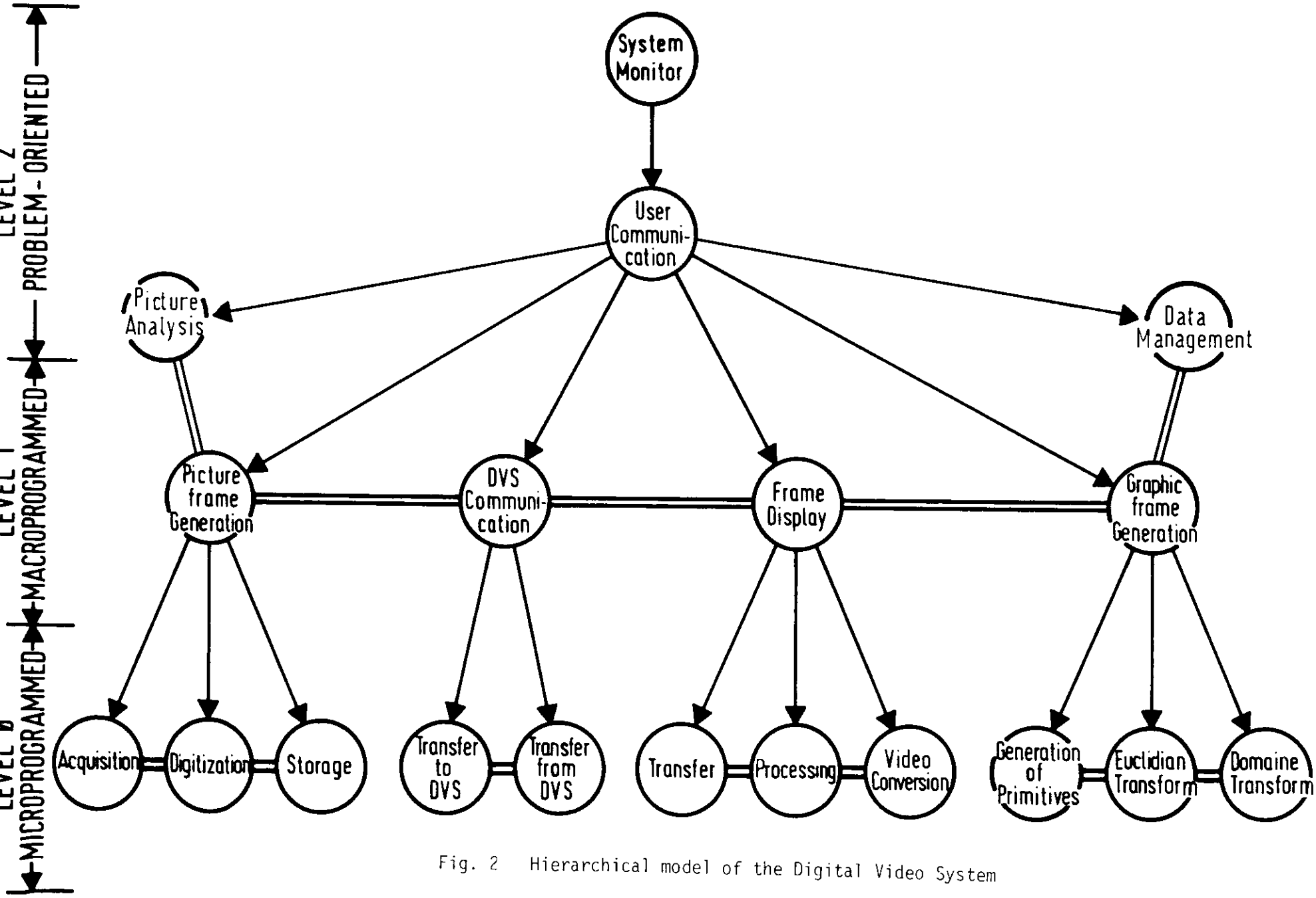


Fig. 2 Hierarchical model of the Digital Video System

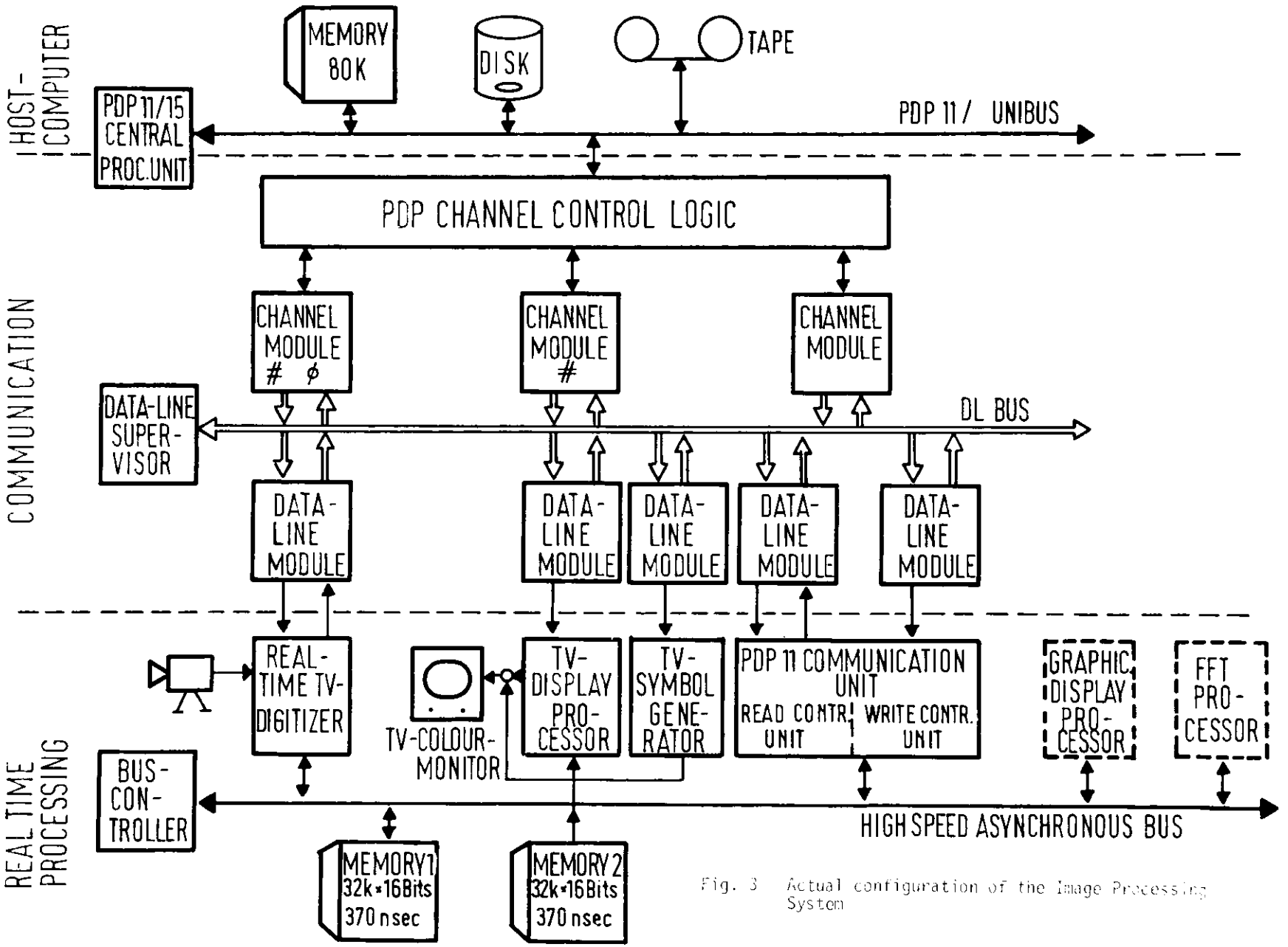


Fig. 3 Actual configuration of the Image Processing System

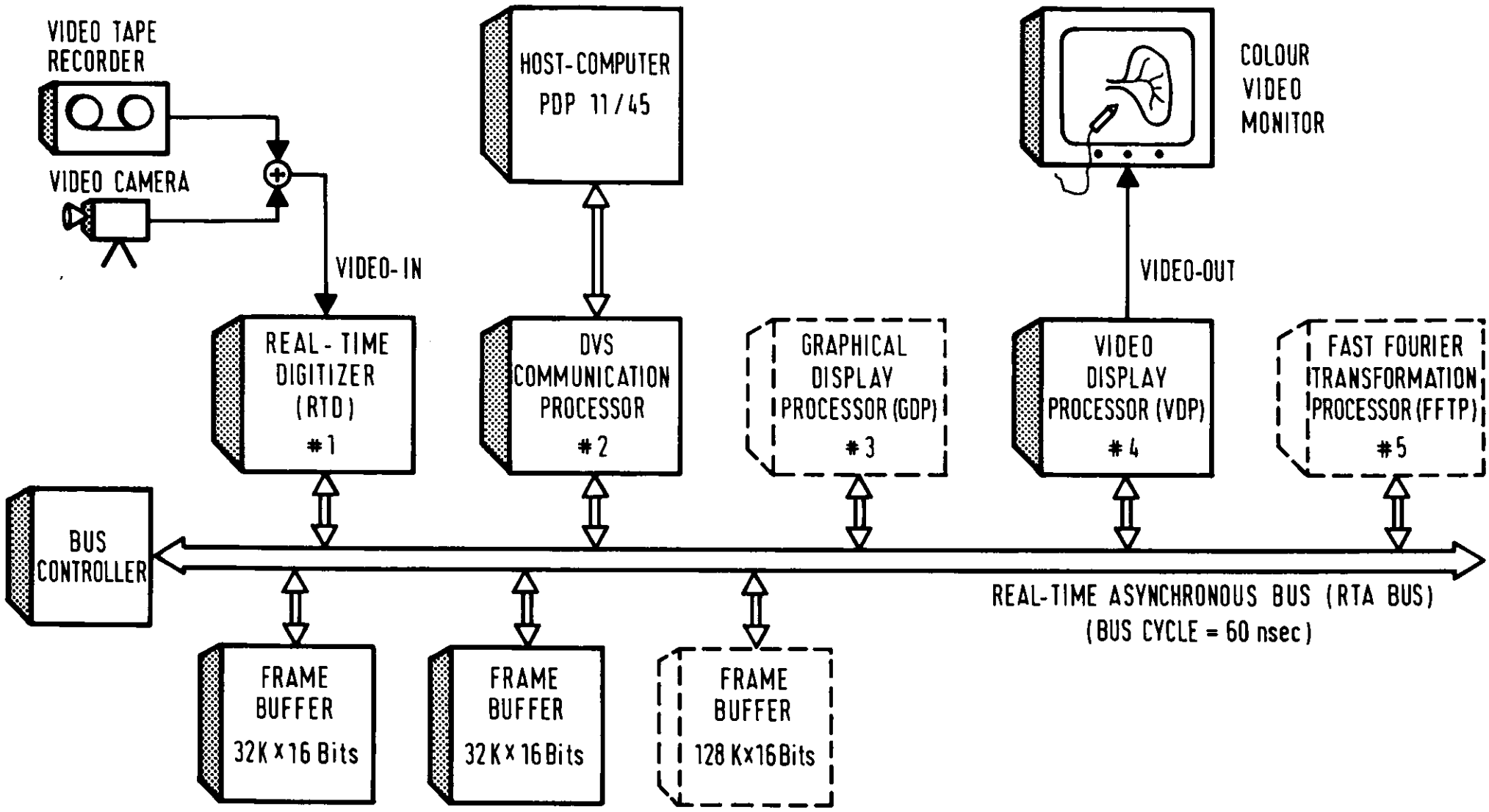


Fig. 4 Basic structure of the Digital Video System

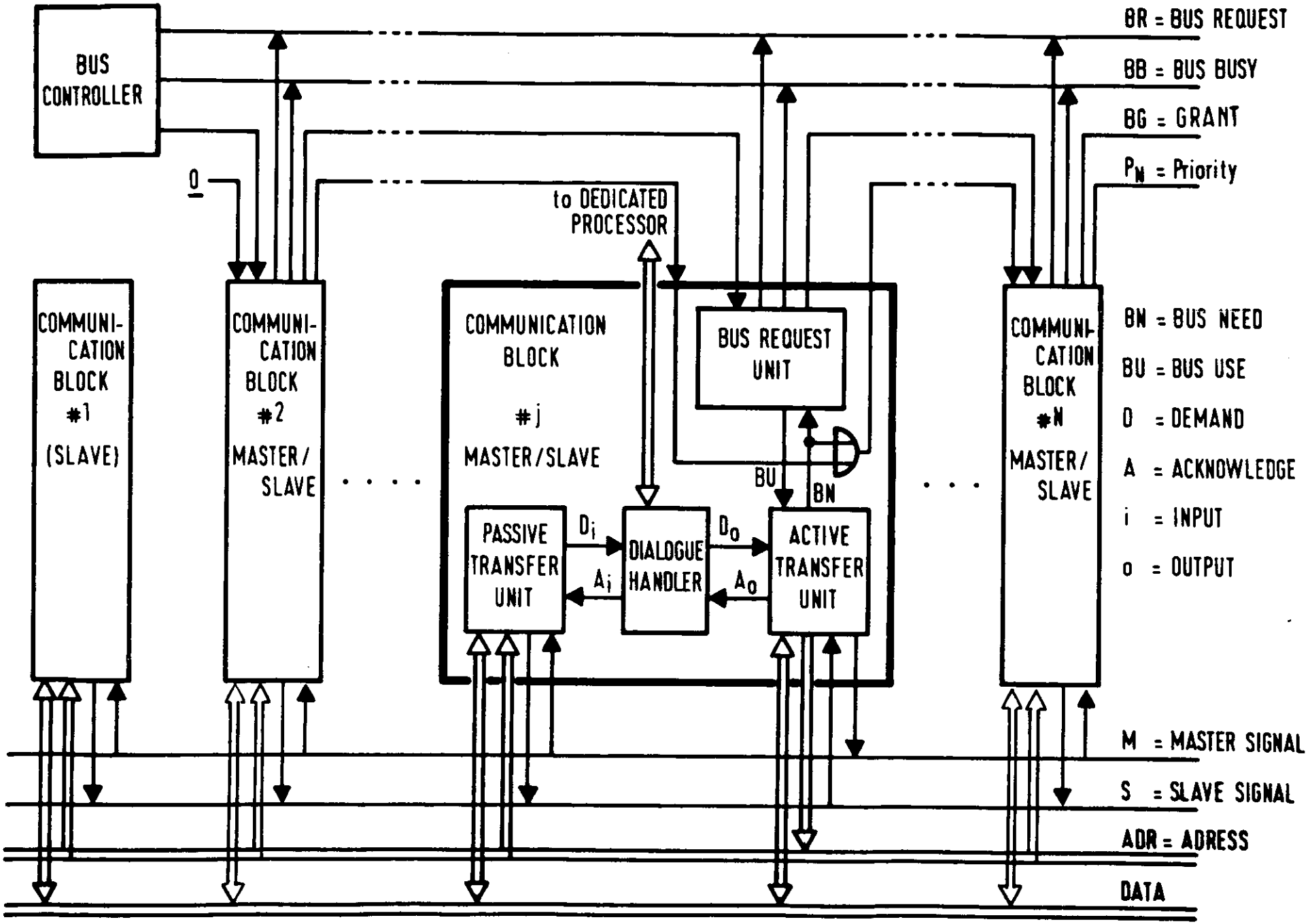


Fig. 5 Structure of the RTA-bus

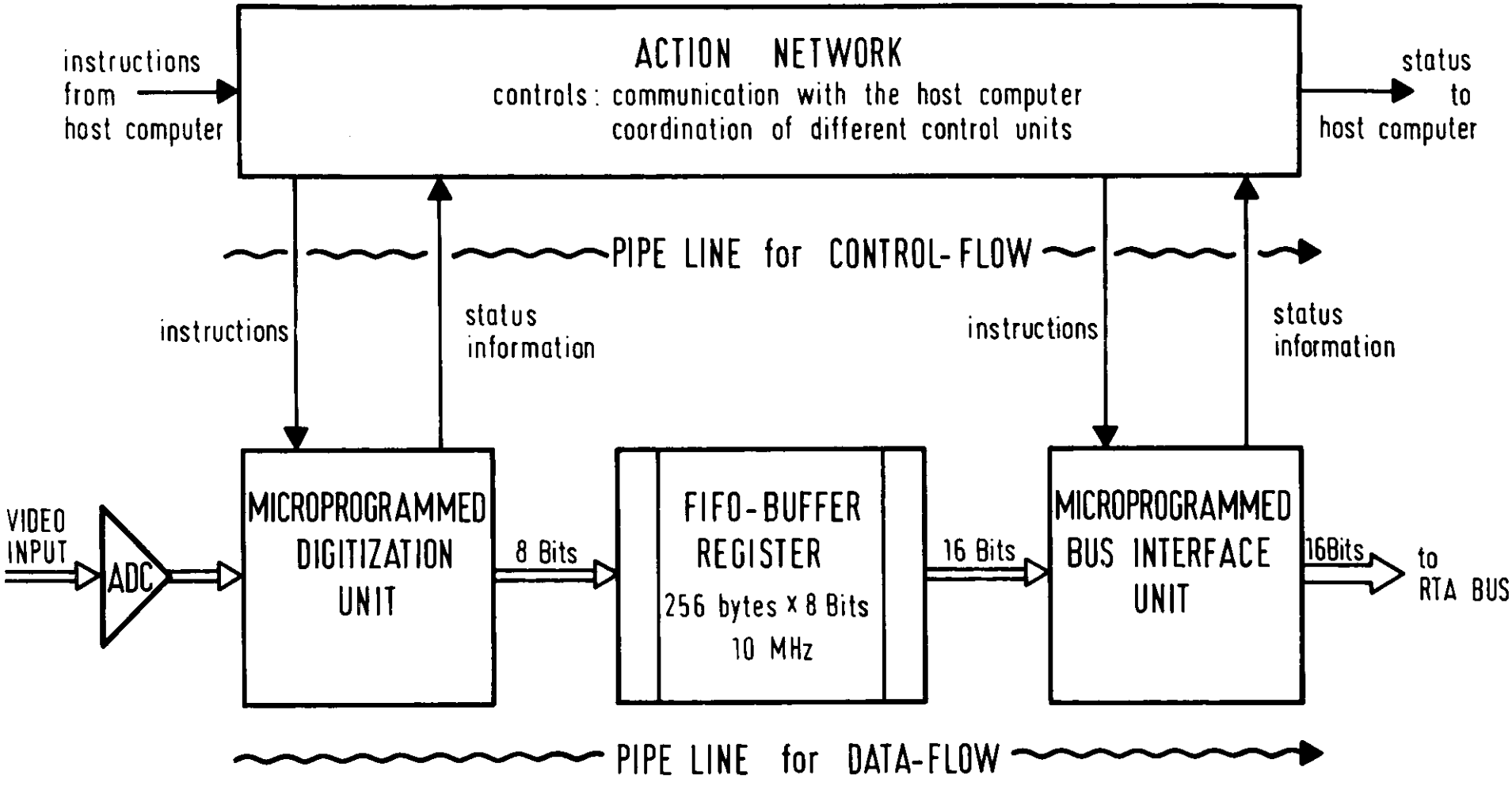
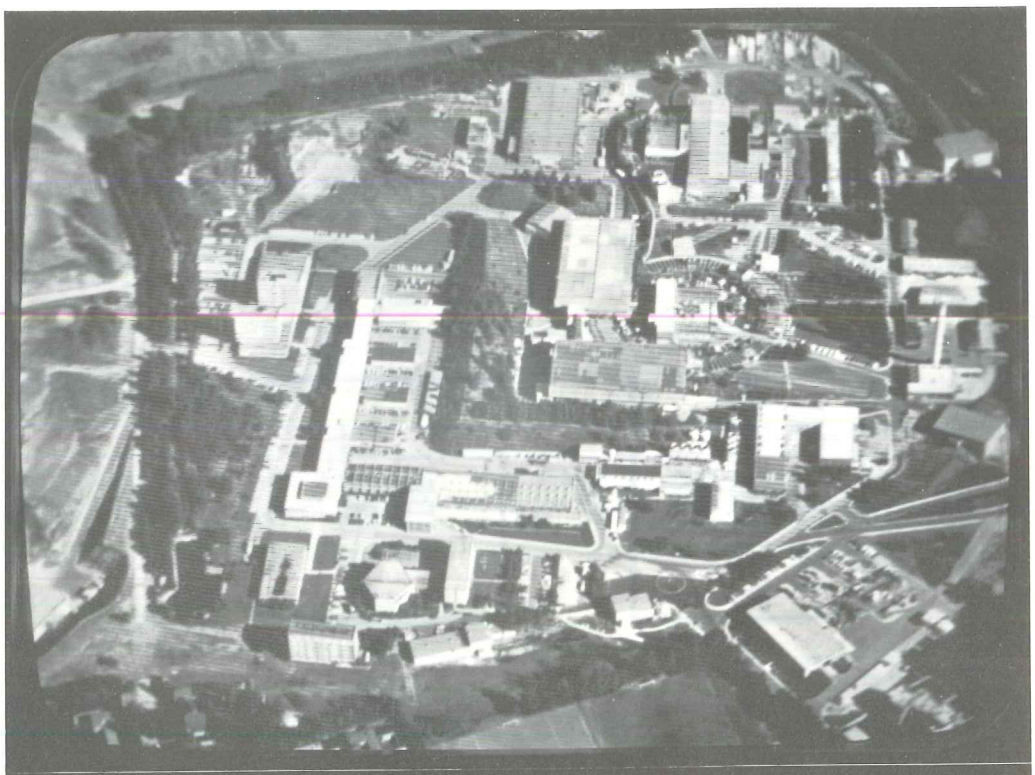


Fig. 6 Structure of the Real Time Digitizer



a

b

c

Fig. 7

Video presentation of a picture as

a) an analog image
b) a digitized image with 5 M samples/s
c) a digitized image with 10 M samples/s (section)

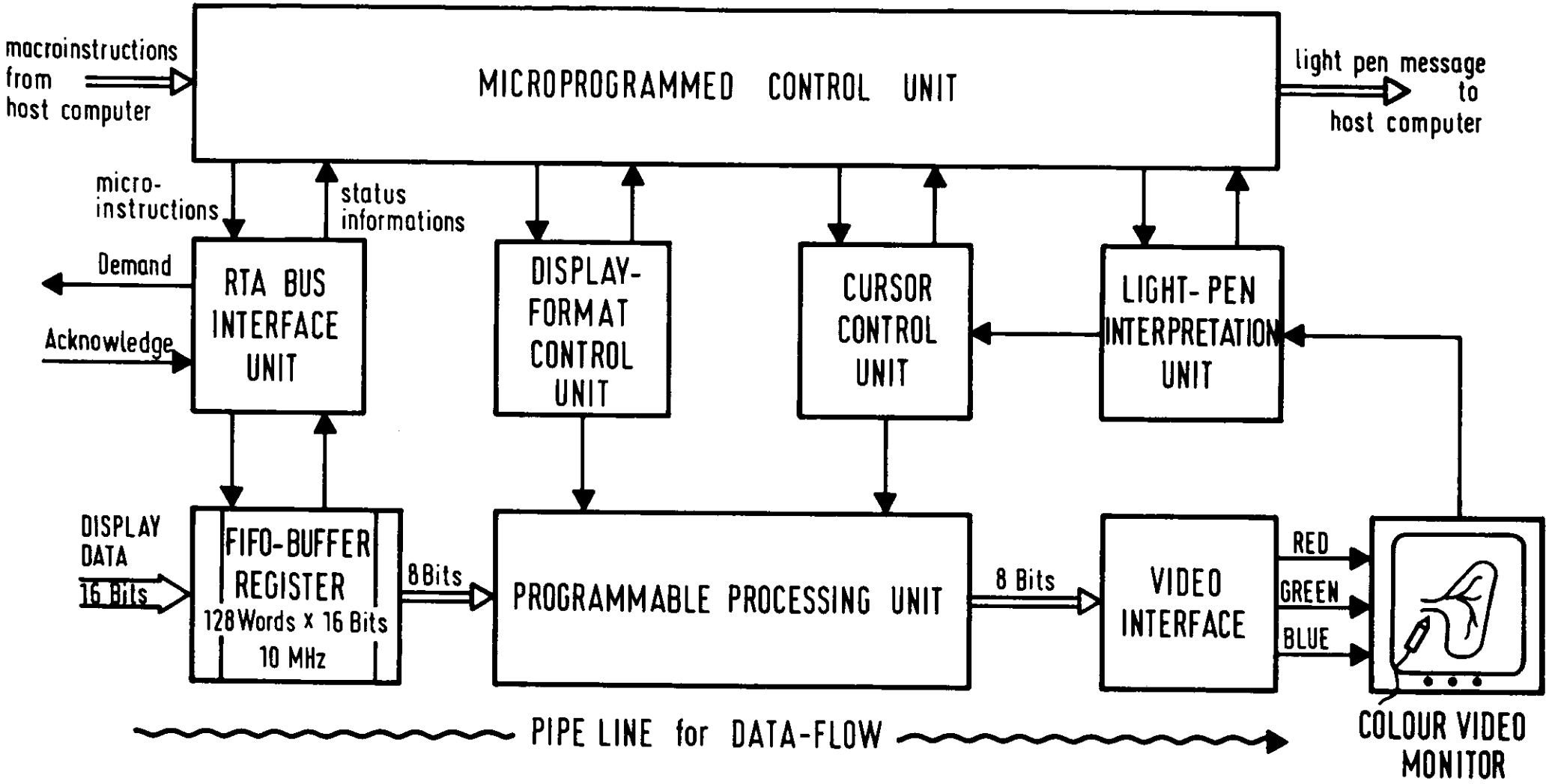


Fig. 8 Structure of the Video Display Processor

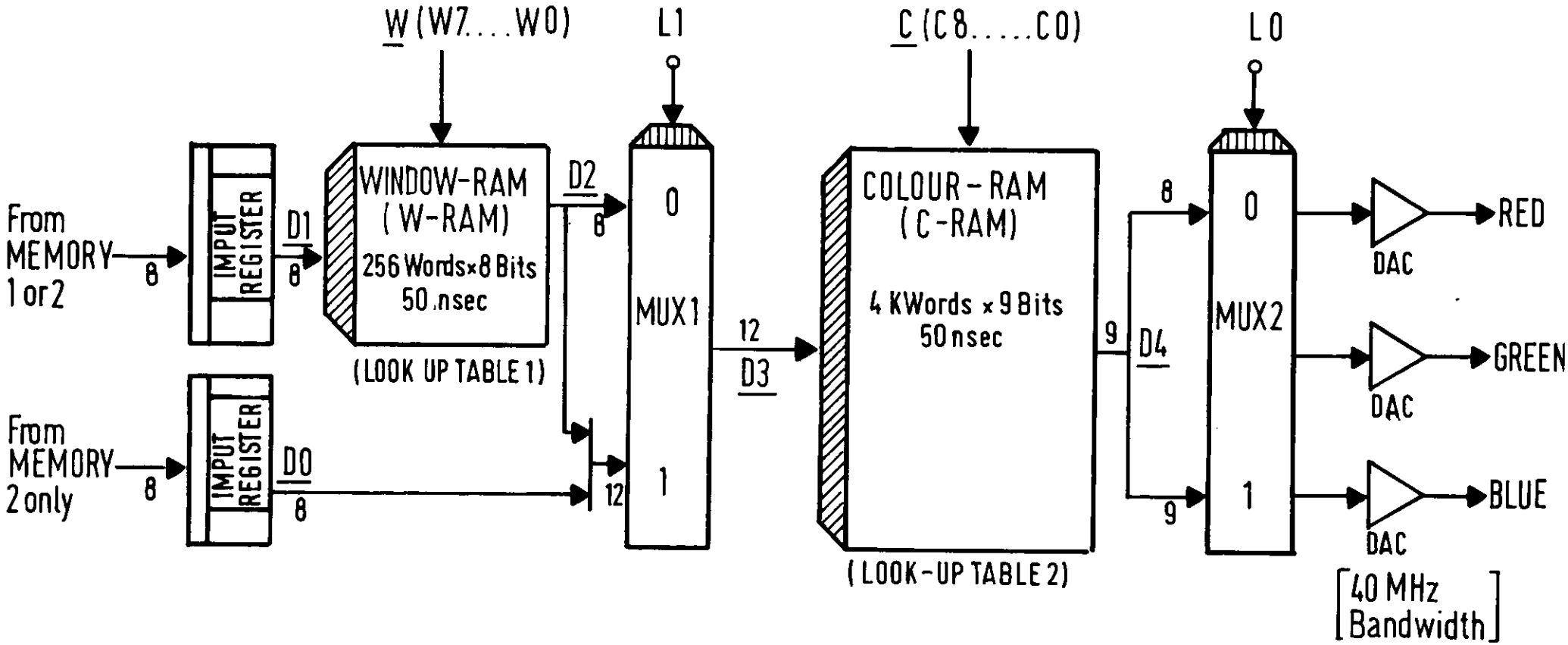
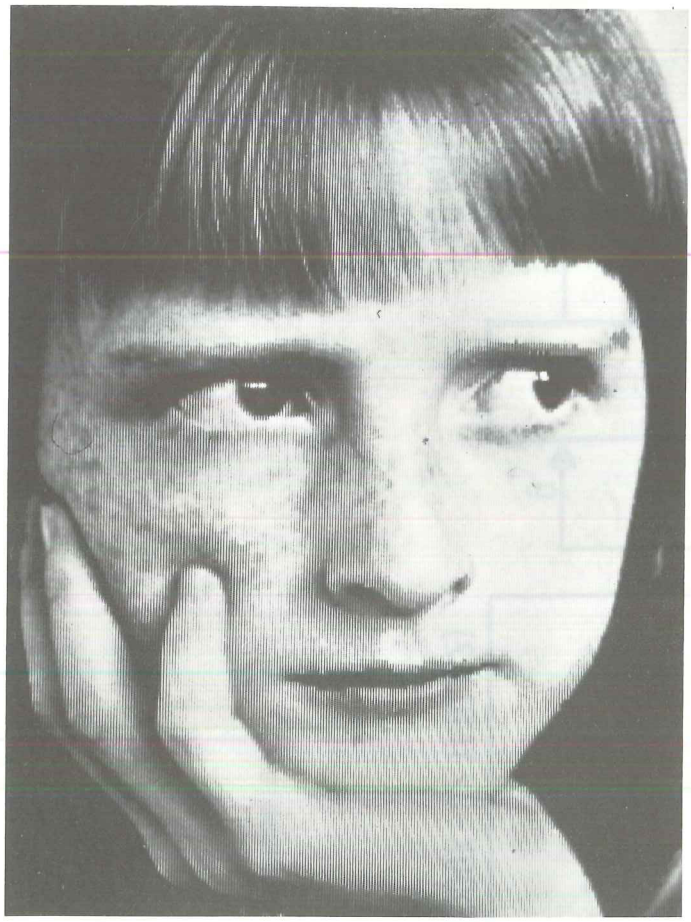
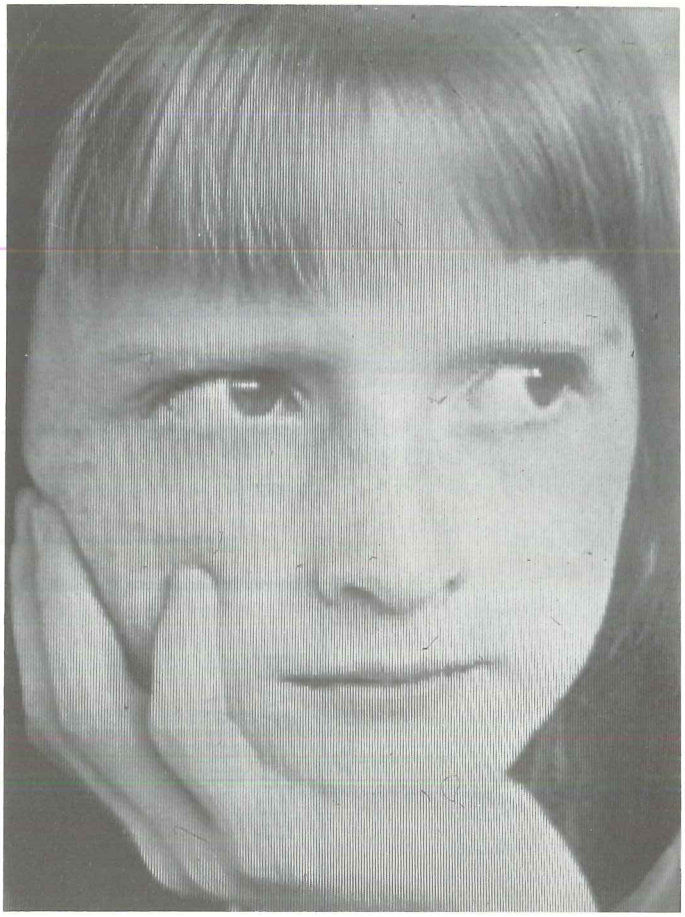


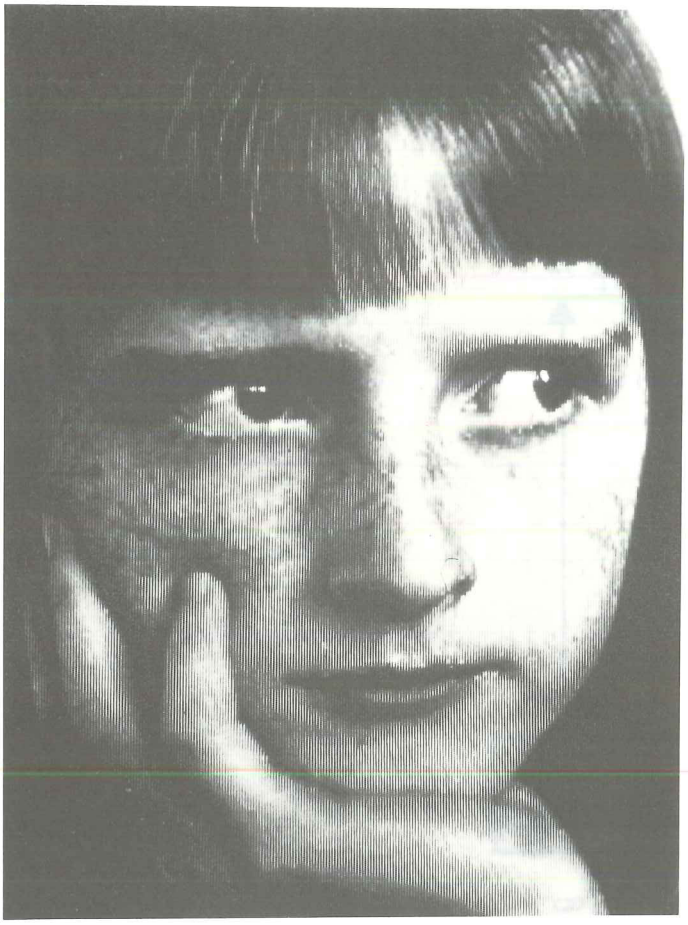
Fig. 9 Basic structure of the Programmable Processing Unit



a)



b)



c)

Fig. 10

Example of grey level transformations

a) linearly digitized image
b) logarithmic transformation of image a)
c) exponential transformation of image a)

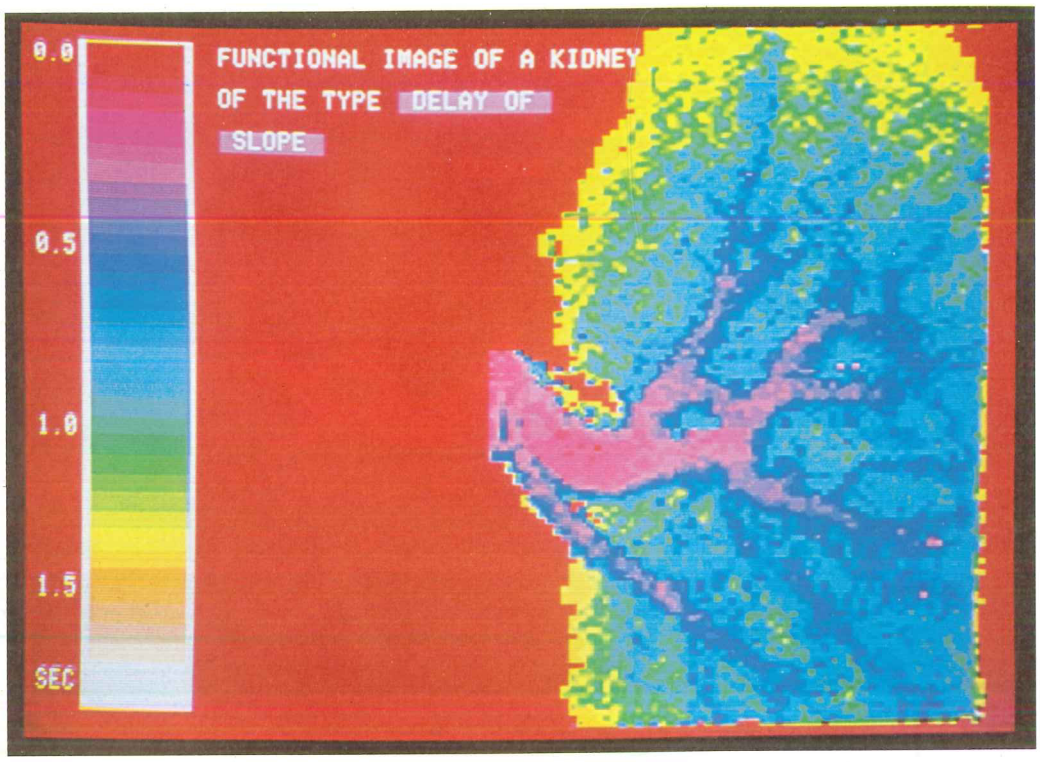


Fig. 11 Example of a pseudo-color image

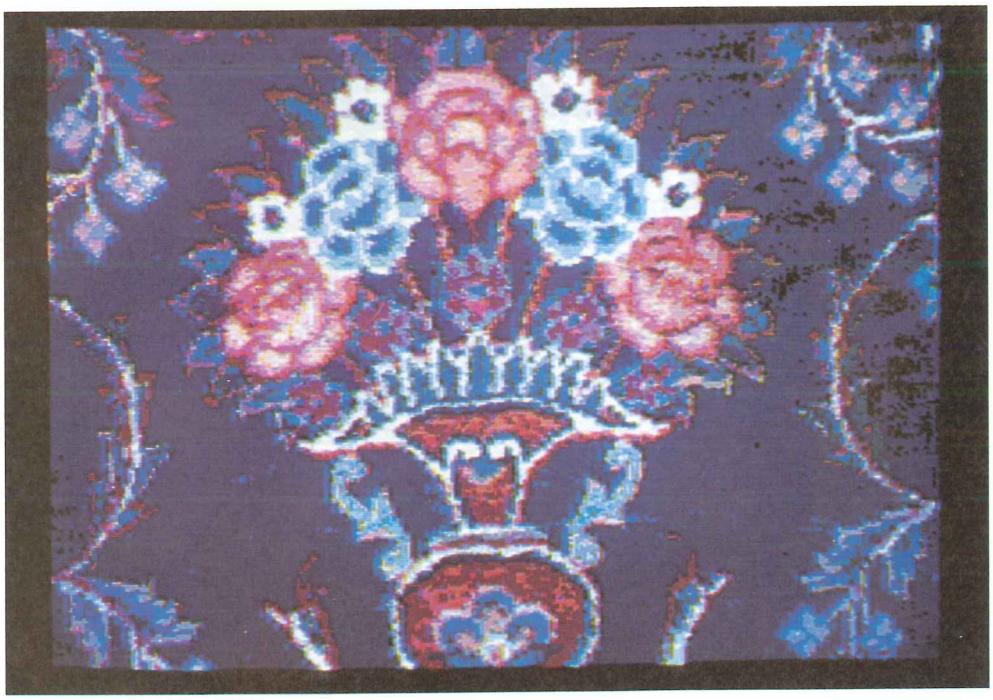
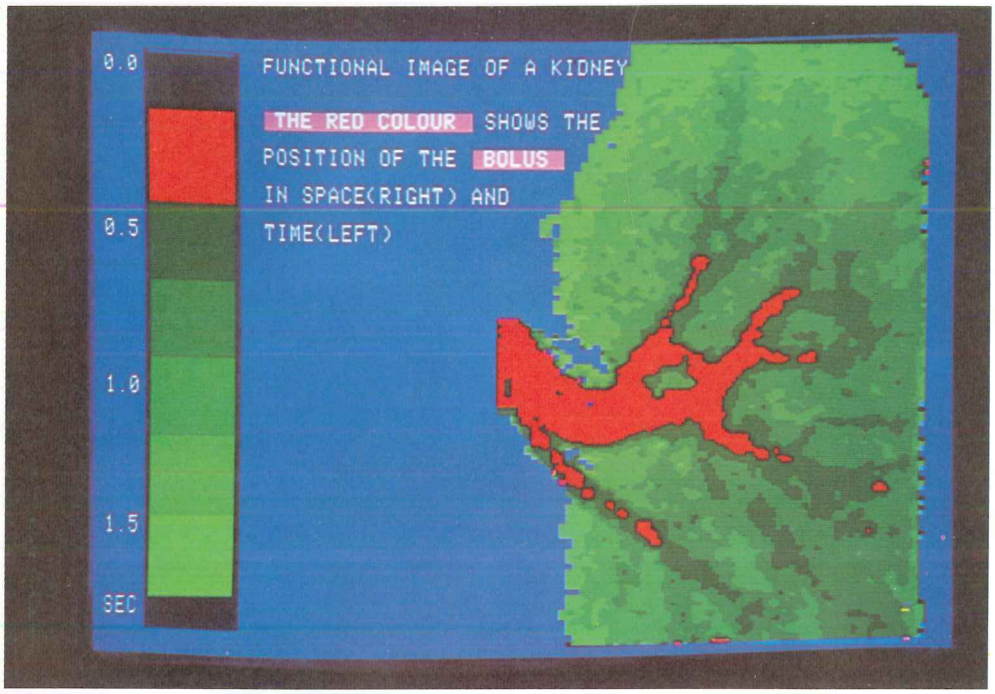


Fig. 12 Example of a true-color image



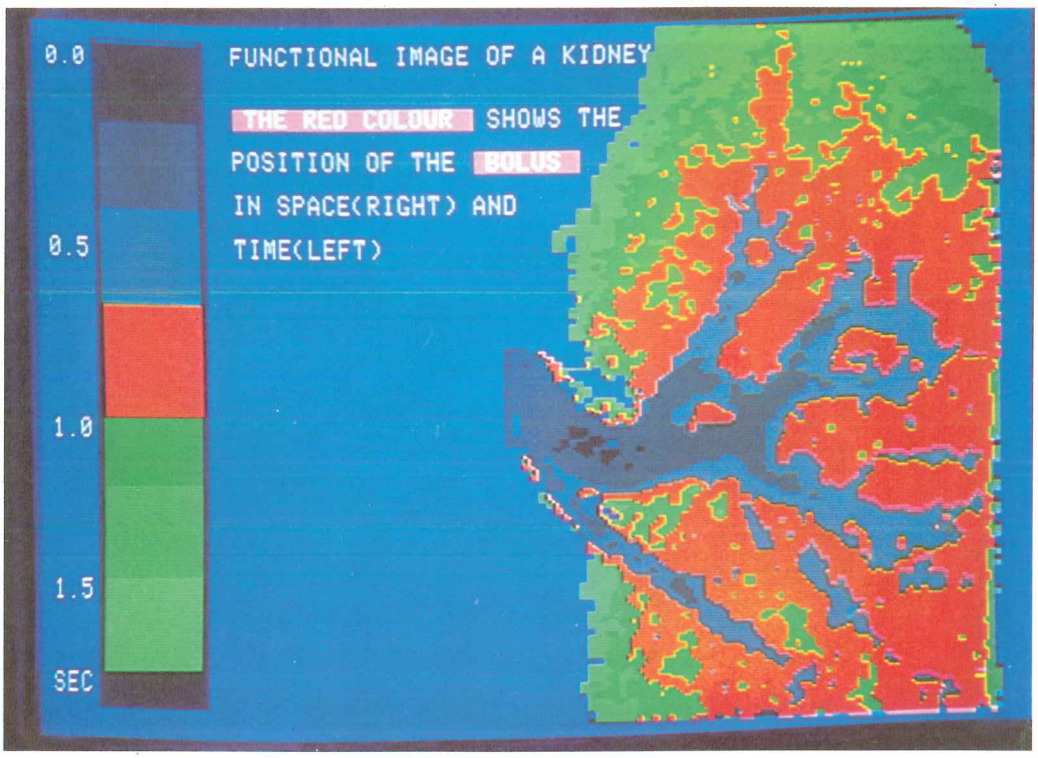


Fig. 13 Two phases of the real-time playback of the blood propagation in the kidney

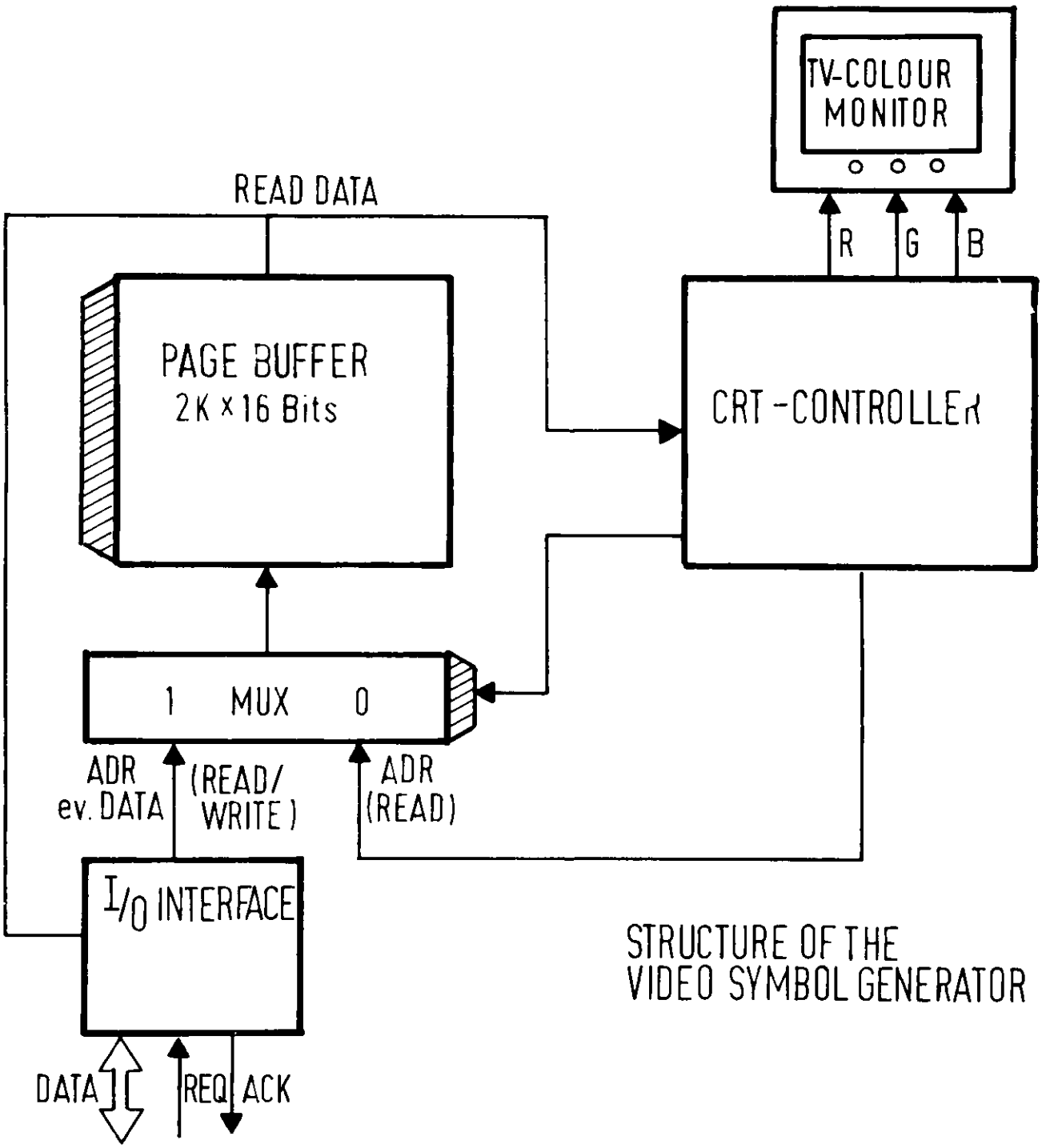


Fig. 14 Structure of the Video Symbol Generator

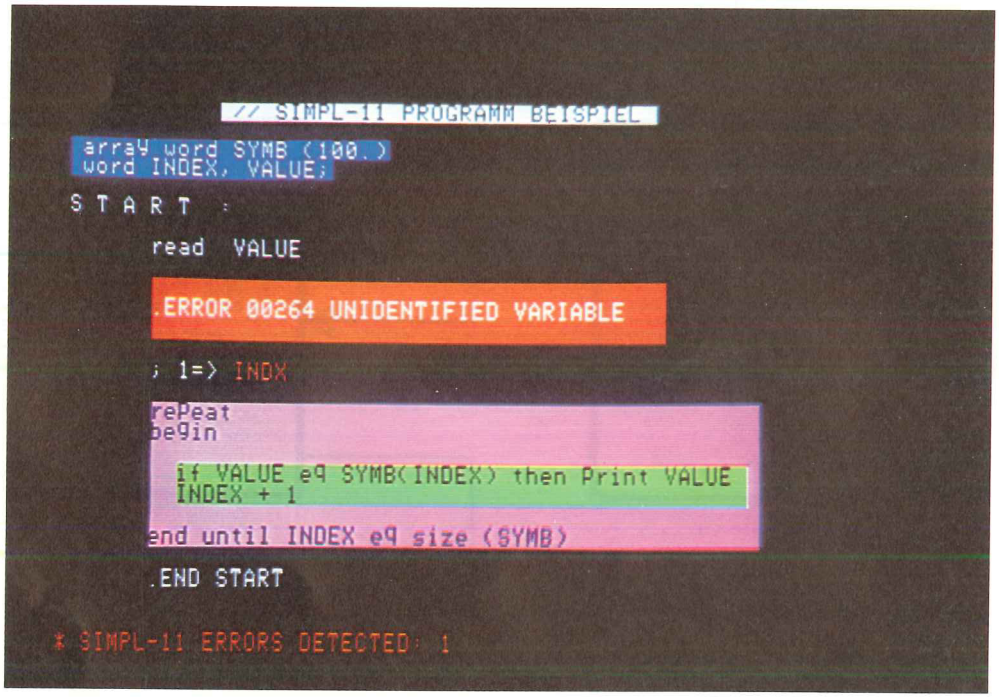


Fig. 15 Example of using optical attributes for a better transparency of program structure

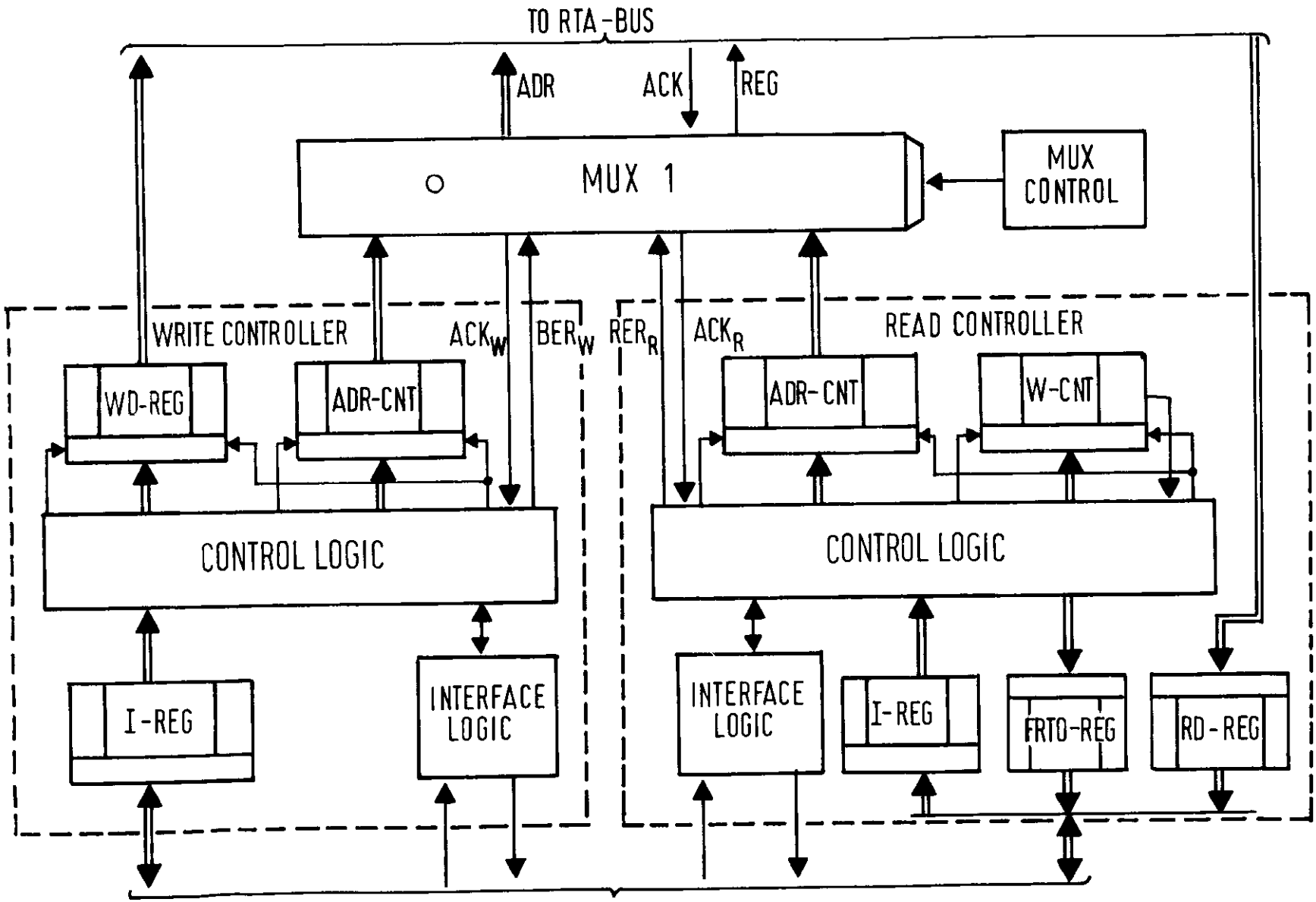
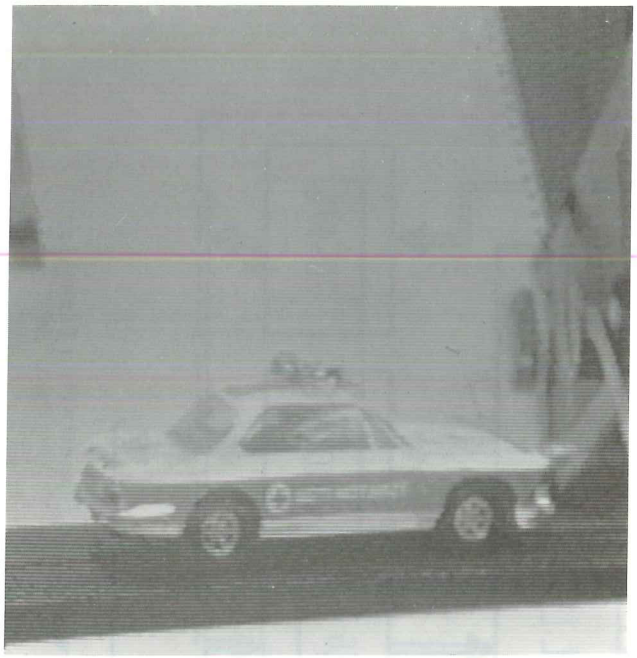
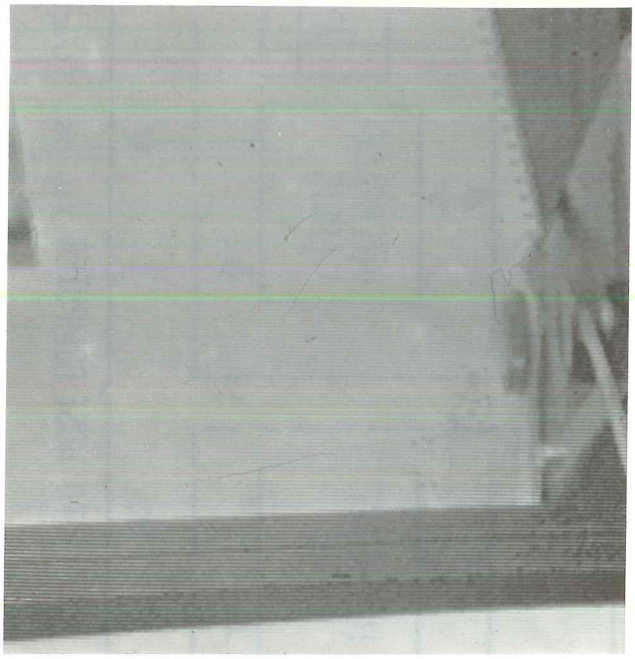


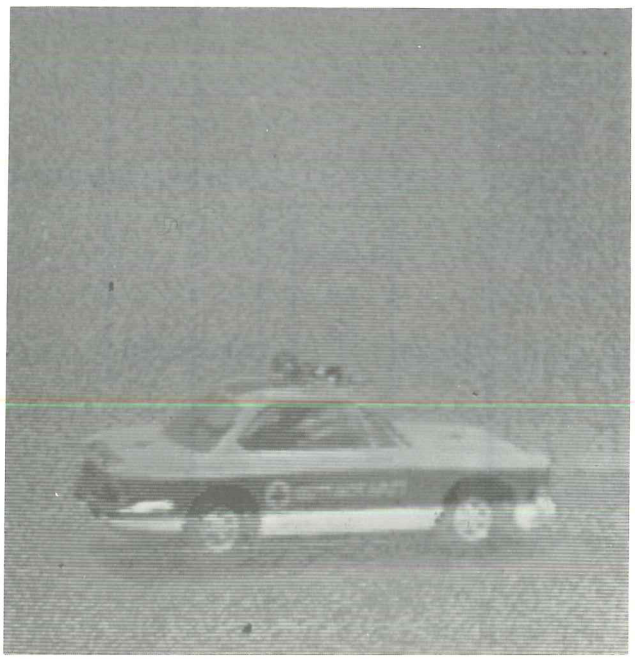
Fig. 16 Structure of the Communication Processor



a)



b)



c)

Fig. 17 Example of a real time arithmetic operation performed by the programmable processing unit.
a) Object with background
b) Background
c) Subtraction of the images a) and b)