Multiprocessor system for the real-time digital processing of video-image series

Ein Multiprozessorsystem für die digitale Verarbeitung von Fernsehbildserien in Echtzeit

von G. C. NICOLAE,
Deutsches Elektronen-Synchrotron DESY, Hamburg
und K. H. HÖHNE,
Institut für Mathematik und Datenverarbeitung in der Medizin, Universitäts-Krankenhaus Eppendorf, Hamburg

Elektron. Rechenanl. 21 (1979), H. 4, S. 171–183
Manuskripteingang: 15. Dezember 1978

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

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 Multiprozessorsystems. Die mikroprogrammierten Spezialprozessoren kommunizieren über einen asynchronen Hochgeschwindigkeitsbus mit einer Transferrate von 15 MWorten/s. 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. Diese Arbeit beschreibt den Entwurf, die Struktur und die Anwendung des Systems.

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 rising interest by physicians concerning 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 the blood. This scene (see Fig. 1) is viewed by the image-intensifier video

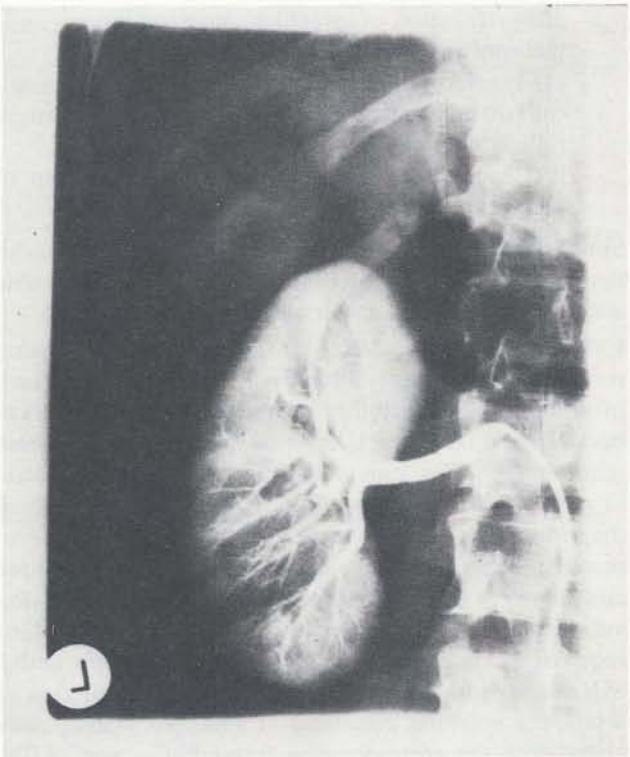


Fig. 1 A frame of a X-ray picture series showing a kidney.

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 in the x-ray picture field (functional images).

Our objective was 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 an 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 in a time while the patient is still present.

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

- Picture data acquisiton, storage, display and transfer of pictures or regions of them 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 levels, 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 destructive material testing.

Earlier approaches to this problem have used analog techniques [3]. Despite of 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 at which 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 and 5 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 existing digitisation 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 the 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 grouped around a processor called "data-interchange", in which the whole *intelligence* for the real-time processing is *concentrated*. It connects pairs of data channels and performs the data processing required for each configuration. 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 intelligence*. The arguments for this decision are given in the following section.

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 and picture display etc. Table I contains the list of the processes which occur in the DVS, with their semantics and a set of associate attributes. Looking at Table I in view of our objective of achieving a throughput compatible with the real-time requirements and a maximum of flexibility we make the following observations:

1. A high throughput can be achieved by *parallelisation* of 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 are running. They can be characterized by their level of

Table 1. Set of Processes.

Process Name	Semantics	Priority Level	Flow Speed	Realization	No. of Process Levels
Picture-frame generation	Picture – acquisition digitizing transfer to frame-buffer storage	1	real-time	hardware + firmware	2
Frame display	Frame data (i.e. picture and/or graphic) – transfer from frame buffer – display in raster-scan format	4	real-time	hardware + firmware	2
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	2
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 for interactivity	hardware +	2
DVS Communication with environment	Exchange of data and control information with the environment	2	real-time	hardware + firmware	2
User Communication with DVS	Exchange of data and control information with DVS	2	real-time	software + hardware	1
Picture analysis	User programmable processing – functional images – blood velocity – blood flow	2	interactivity	software	
Data management	Retrieval of pictures	2	interactivity	software	1

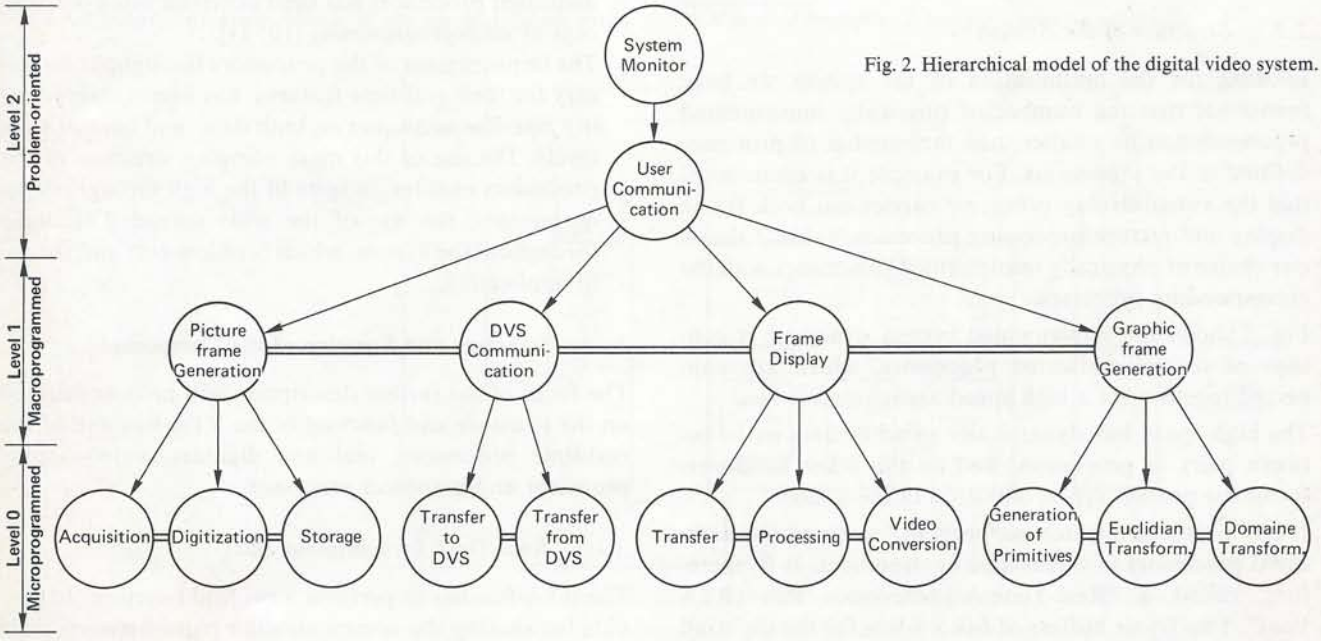


Fig. 2. Hierarchical model of the digital video system.

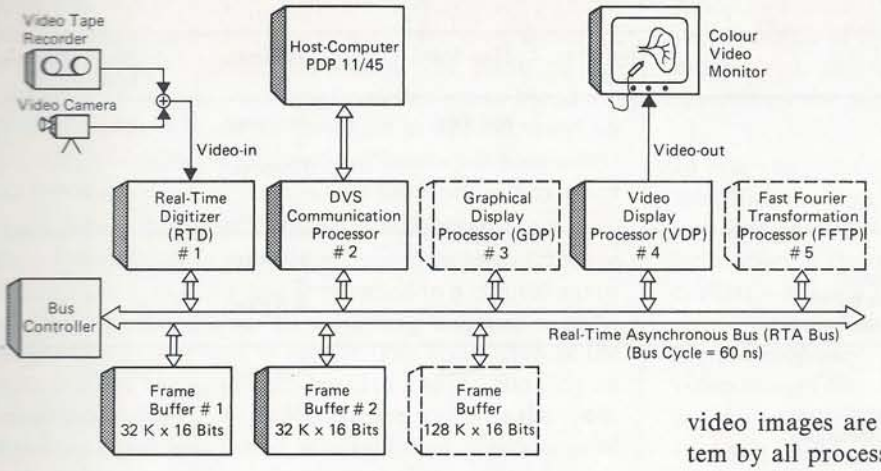


Fig. 3. Basic structure of the digital video system.

programming: problem oriented, macro- and microprogramming.

2. There is a wide spectrum of *flow speeds*. The means of realising 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 both 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 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*

Looking for the optimisation of the system we have found out 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 that the video display processor carries out both frame display and picture processing processes. Table 2 shows our choice of physically implemented processors with the corresponding processes.

Fig. 3 shows the implemented system structure. It consists of several dedicated processors, which are connected together via a high speed asynchronous bus.

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 64k × 8 Bits for the digitized video images 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 interaction between the concurrent processes and subprocesses in the system are very complex, a formal description of their communication rules was necessary. A suitable description of the interactions between the parallel prodesses can be given by the *Petri-Net model* [6]. This model is considered as a further development of the state-transition graph which is normally used for description of sequential processes [8]. Here the communication rules are physically represented by the edge transitions of the communication signals. Petri-net-representation of the communication rules allows a relatively easy and reliable check for deadlock and unsafe conditions [7]. 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* [10; 11].
- The improvement of the processors throughput, necessary for their real-time features, has been achieved using *pipe-line techniques* on both data- and control-flow levels. The use of this more complex structure of the processors enables, in spite of the high throughput requirements, the use of the wide spread TTL logic throughout the system, which is of low cost and simple in application.

3. Structure and Function of the Components

The focus of the further description will be concentrated on the structure and function of the RTA-Bus and of the real-time processors, real-time digitizer, video-display processor and graphical processor.

3.1 *Real-Time Asynchronous Bus*

The RTA-Bus has to perform a twofold function: to provide for sharing the communication paths between pairs

Table 2. Process Set Versus Processor Set.

Processor	Process							
	Picture-Frame Generation	Frame Display	Picture processing	Graphic-Frame Generation	User Communication	DVS Communication	Picture Analysis	Data Management
Real-Time Digitizer	×							
Video-Display Processor		×	×					
Graphic Display Processor				×				
Communication Processor						×		
General Purpose Computer			×		×		×	×
Special Processor							×	

of processors, and to manage the process synchronization 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. 4 shows the structure of the implemented bus. As in conventional bus structures [12], 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 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 the control over the bus paths. The request from a master block with a higher priority than the actually active master block will interrupt the communication over the bus within one bus cycle. A communication block contains dedicated units such as bus request unit, active transfer unit and dialog-handler, which simultaneously carry out communication (sub)processes with their environment. 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. Its structure shows the following particular properties:

- By the *parallelization* of the (sub)processes needed for the bus communication, the administration times were minimized (60 ns bus cycle).
- 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 M words/s.
- The data transfer over the bus occurs wordwise or blockwise. As a property decisive for the real-time application the choice about the transfer mode is done

*) A detailed description is being prepared for publication.

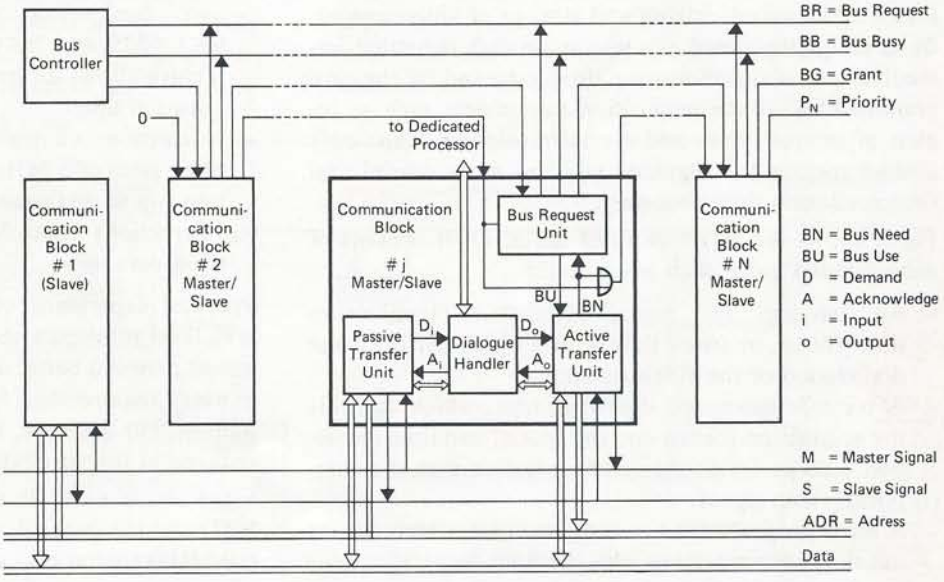


Fig. 4. Structure of the RTA-Bus.

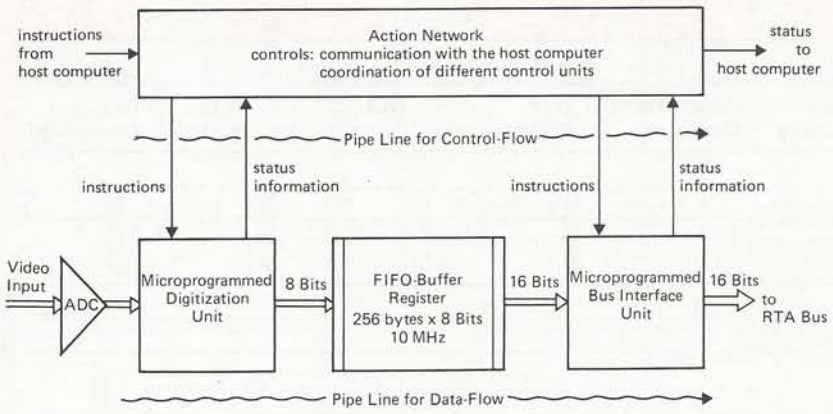


Fig. 5. Structure of the real-time digitizer.

automatically depending on the bandwidth of the communicating processors themselves. Processors with a high bandwidth, such as the real-time digitizer, transfer data in blocks, whereas processors with a low bandwidth, such as the general purpose computer, transfer data in words.

- The time constants necessary to grant or release the bus for one of the dedicated processors are 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 a *minimum delay time* for bus grant.

For the compatibility with the host computer, the width of the data vector in the RTA-bus is actually 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 with the consequence of a bandwidth diminuation 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.

Fig. 5 shows the architecture of the RTD. It consists of subordinated units, such as

- An analog to digital converter (conversion time = 66 ns, aperture time = 1 ns) which performs the digitization of 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 microprogrammed bus interface unit, which carries out the communication with the RTA bus.
- A FIFO buffer register, which compensates for the fluctuation of the incoming data from the digitizing unit and outgoing data to the RTA bus.

These subordinate units are connected together in a pipe-lined structure on the data-flow level which enables to overlapp the different phases of the picture acquisition. Their effective control is performed by a second pipe-lined structure, called *action network*, which works on the control-flow level and enables the overlapping of the control phases corresponding to those on the data-flow level [3]. 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.

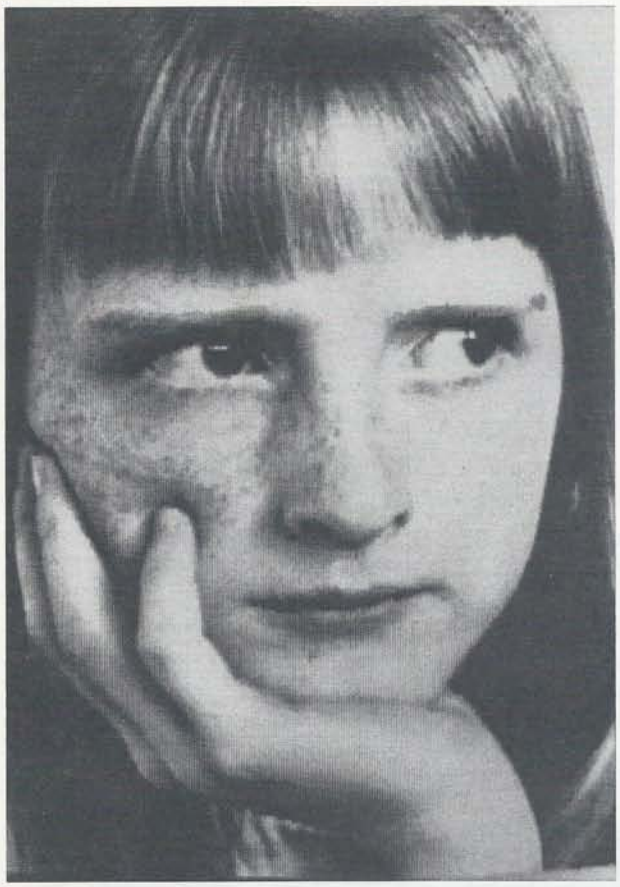
This pipe-lined structure on both the control-flow and data-flow levels allows the uncritical satisfaction of the high throughput requirements of the RTD.

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 an irrelevant data reduction at the 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.

Practical experience with the DVS has shown that the grey level resolution should also 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 could be reduced in favour of the time resolution.

Fig. 6 shows as an illustration of the performance of the RTD a video image digitized with low (5 MHz) and high (10 MHz) spatial resolution.



a) digitized video image (5 M samples/s).

b) window of the digitized video image (10 M Samples/s).

Fig. 6. Example of digitized video image.

3.3 *Video Display Processor (VDP)*

The Video Display Processor (VDP) is used for the display of the data contained in 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 3.3). To give an easily interpretable optical presentation for the different contexts associated with the data a transformation 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*.

Fig. 7 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 Bus-interface control unit, which manages the communication with the RTA-bus.

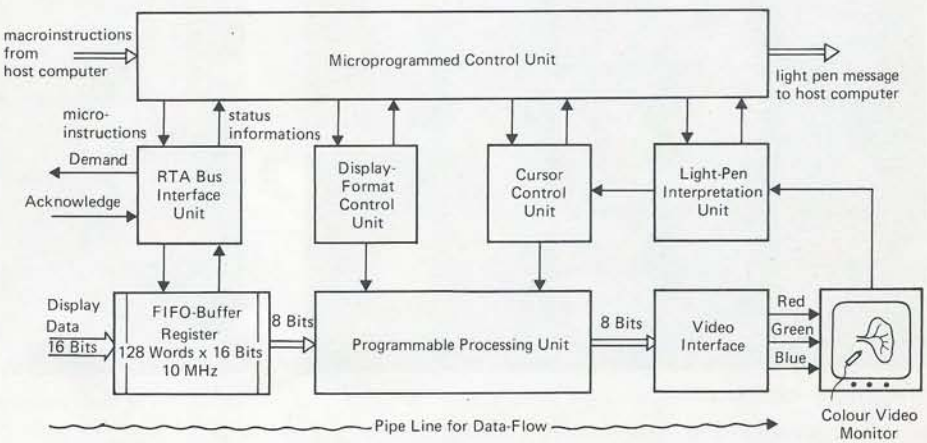


Fig. 7.
Structure of the video-display processor.
(siehe Seite 184) interactivity

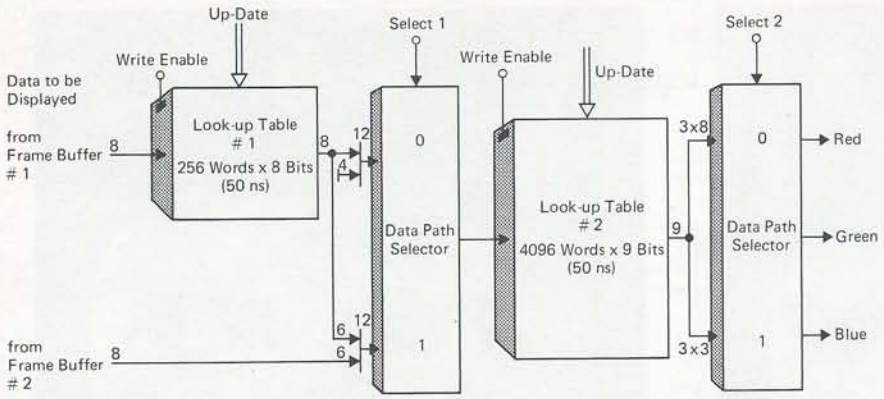
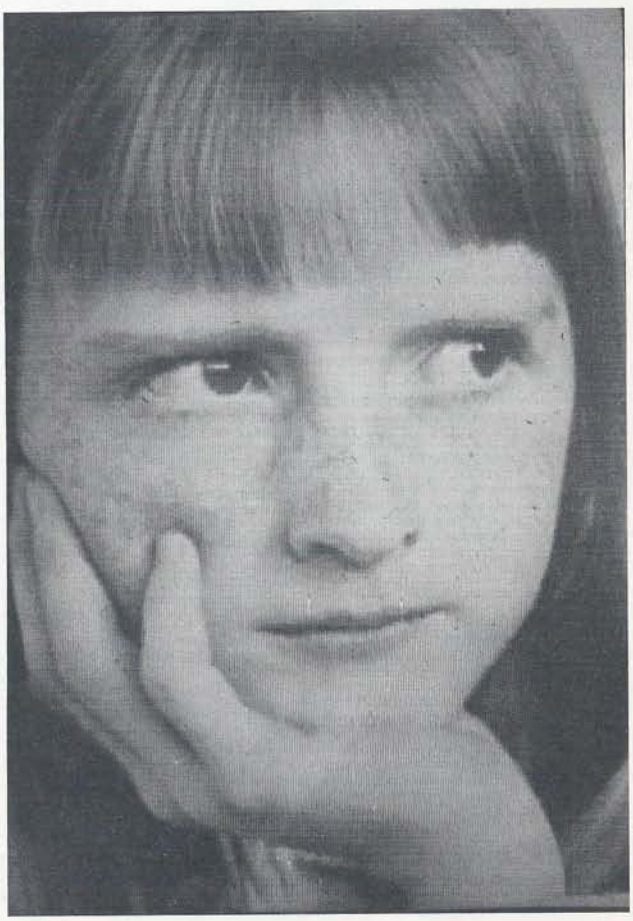


Fig. 8. Basic structure of the programmable processing unit.

- 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 (PPU), 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 interpretation and cursor control unit, which supports the hardware part for the 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 occurring 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. 8.

a) logarithmic transformation of image 6a.



b) exponential transformation of image 6a.

Fig. 9. Example of gray level transformations.

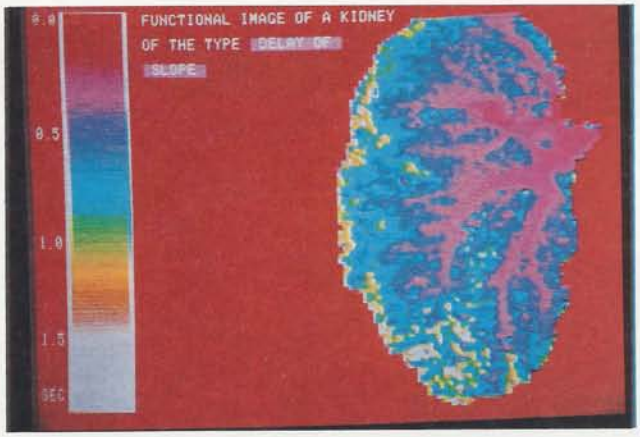


Fig. 10. Example of a pseudo-color image.

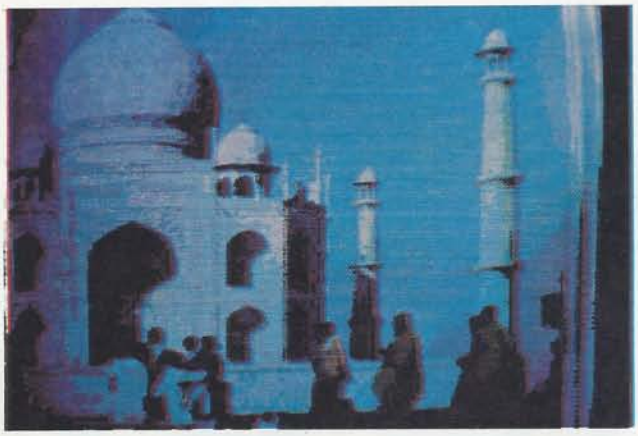


Fig. 11. Example of a true-color image.

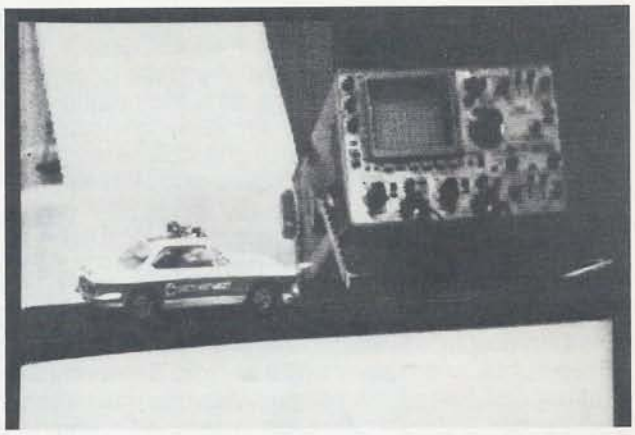
For the realization of the processes we chose the technique of look up tables. Here the transformation function from frame buffer contents to gray levels or colours is written into a RAM in tabular form. As the transformation function is often composed of two functions (e.g. contrast enhancement and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

OPERATION (BUFFER N) ⇒ PICTURE
OPERATION (BUFFER 1 ∘ BUFFER 2) ⇒ PICTURE

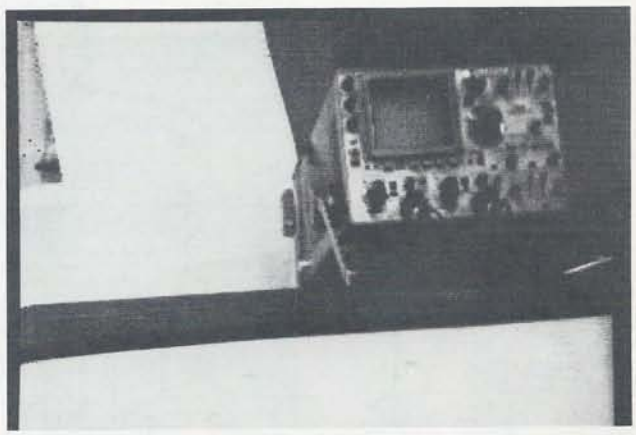
A typical monadic operation on one buffer is a grey-level transformation. As an example Fig. 9 shows the lograrithmic transformation (a) of the linearly digitized picture from Fig. 6a in comparison with its logarithmic transformation (b). Obviously the logarithmic transformation enhances the low grey 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. 10 shows a functional image [13] 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.

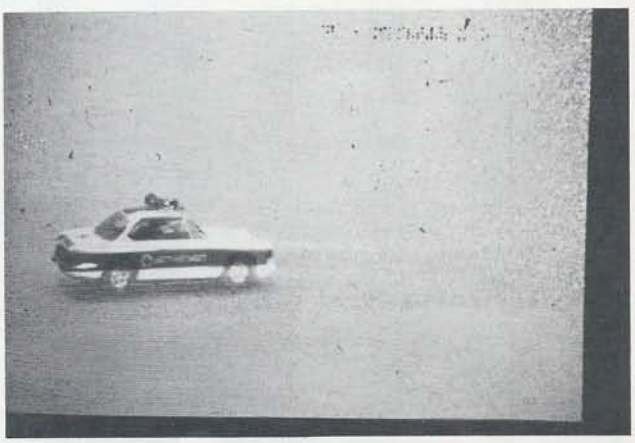
As an example of a monadic operation on the concatenation of both buffers one can write the picture intensity into one buffer and the associated colour into the other.



a) object with background.



b) background.



c) subtraction of the images a) and b).

Fig. 12. Example of a real time arithmetic operation performed by the programmable precessing unit.

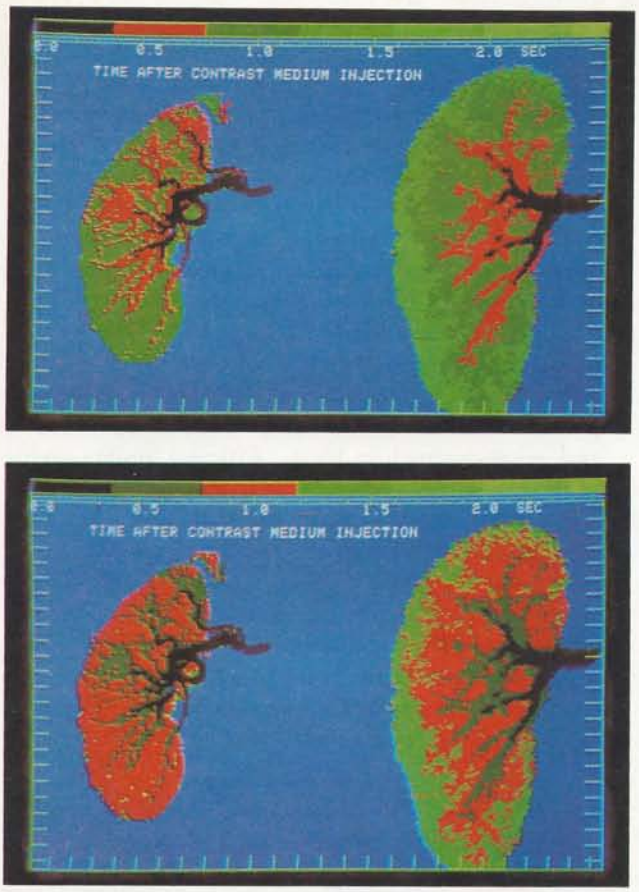


Fig. 13. Two phases of the real-time playback of the blood propagation in the kidney.

The appropriate function in the look-up tables would provide the true-colour coded picture on the TV screen. Fig. 11 shows a *true-colour* picture of a post-card picture.

The dyadic operations work on the two frame-buffers.

(BUFFER 1) OPERATION (BUFFER 2) ⇒ PICTURE

As an example one may fill the RAM with a function which performs a subtraction of the two pictures, as shown in Fig. 12.

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.

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.

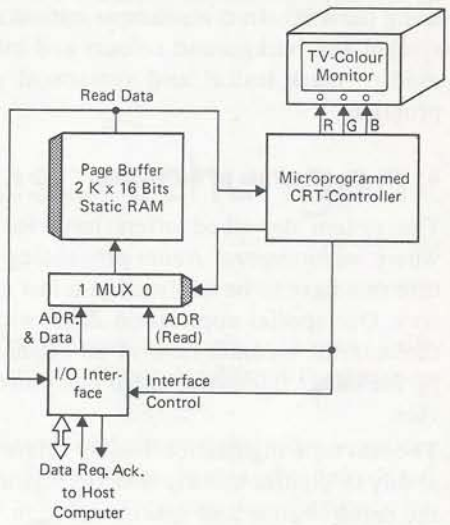


Fig. 14. Structure of the Video Symbol Generator.

Fig. 14 shows the structure of the VSG. The alphanumerical information to be displayed is contained in the page buffer (2K Words × 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/0 interface, from the host computer which updates the alphanumerical

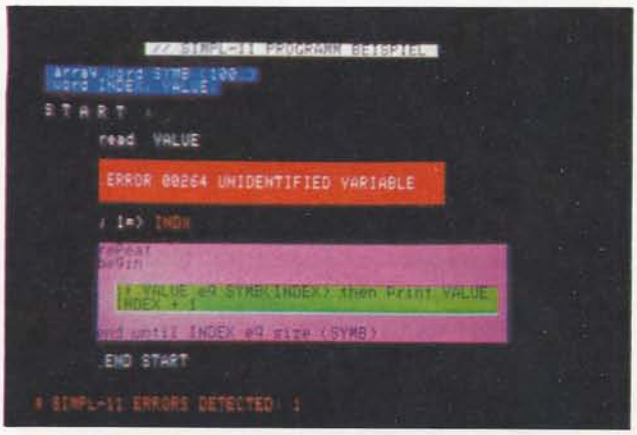


Fig. 15. Example of using optical attributes for a better transparency of program structure.

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.

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 [14] 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.

4. Application of the System

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 intensifier video equipment of an x-ray device.

The real time digitization facility offers the physician the ability to digitize quickly 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 for the first time 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 processors 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-Transformations 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/0 communication part with its environment (communication processor).
- Using *structured design methods* throughout in the system we have achieved improved transparency of the design, better documentation, shorter implementation time 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 data vector of RTA-bus, should favourably influence the performance/price 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 complete 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.

Acknowledgement

The authors wish to thank Prof. *H. Schopper* and Prof. *G. Weber* (DESY) for their continuous support of the project, and 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. Especially we are gratefully to Prof. *S. Wendt* (University of Kaiserslautern) for many stimulating discussions concerning the design philosophy.

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 Procesing, Informatik-Fachberichte Vol. 8, pp. 232–243, Berlin–Heidelberg–New York: Springer 1977.

[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 Instrumen., 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 Management, 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 Bideo Images. IEEE Trans. Comput. Vol. C 25, pp. 1089–1100, Nov. 1976.

[7] *C. A. Petri:* Kommunikation mit Automaten. Ph. D. Thesis, Darmstadt, 1962.

[8] *J. L. Peterson:* Petri-Nets. ACM Computing Surveys Vol. 9, No. 3, September 1977.

[9] *S. Wendt:* Petri-Nets and Asynchronous Sequential Circuits. Elektronische Rechenanlagen, Vol. 16, pp. 208–216, 1974.

[10] *S. S. Husson:* Microprogramming: Principles and Practices. Englewood Cliffs: Prentice Hall, 1970.

[11] *S. Wendt:* On Structures of Microprogramm Control Units. Elektronische Rechenanlagen, Vol. 13, pp. 22–26, 1971.

[12] *K. J. Thurber, E. P. Jensen, L. A. Hack:* A Systematic Approach to the Design of Digital Bussing Structure. In 1972 Fall Joint Computer Conference, AFIPS Conf. Proc. Vol. 41, part II, pp. 719–740.

[13] *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.

[14] *G. Pfeiffer:* Simpl 11, eine einfache Implementierungssprache für PDP11-Rechner. DESY Report Nr. DESY DV-76/02, 1976.