A Flexible Heterogeneous Video Processor System for Television Applications.

Egbert G.T. Jaspers [1], Peter H.N. de With [2], Johan G.W.M. Janssen [1]

[1] Philips Research Labs., Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands.
[2] University of Mannheim, Dept. Circ. & Simulation, B6-26, 68131 Mannheim, Germany.

ABSTRACT - A new video processing architecture for high-end TV applications is presented, featuring a flexible heterogeneous multi-processor architecture, executing video tasks in parallel and independently. The signal flow graph and the processors are programmable, enabling an optimal picture quality for different TV display modes. The concept is verified by an experimental chip design. The architecture allows several video streams to be processed and displayed in parallel and in a programmable way, with an individual signal quality.
Keywords: TV, multi-window, display processing, co-processor, PiP, scaling, processor architecture.

1. Introduction

The development for high-quality television and the strong emerge of Internet for information search and its commercial applications, has lead to the design of a new TV architecture with much more flexibility and openness to new TV features, than was usual in the past. Application analysis showed that Internet in the TV environment implies that several video windows should be displayed simultaneously, rather than looking to a single broadcast stream. For example, one window could be used for TV signal watching, where in the second channel, Internet (TV) could be displayed for additional information. The usage of several channels should be programmable in size and quality and preferably be set by the consumer.

A second argument for creating more flexibility is the advent of digital television, which offers a wider variety of picture characteristics. The new TV technology will inevitably push the quality of the conventional TV to a higher level. Including the new digital processing, such as decoding of MPEG-2 streams, will also lead to performance reconsiderations of the conventional video enhancement algorithms, like noise reduction and sharpness improvement.

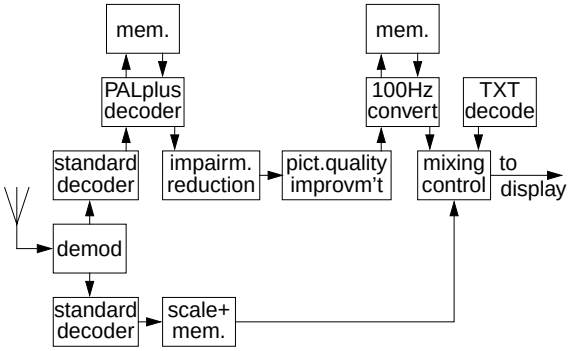


Figure 1: Block diagram of signal enhancement path in conventional high-end television.

The aforementioned system aspects can be converted into a number of requirements, concerning a new TV architecture.

1. Several video signals can be monitored on a TV display with flexible windows, together with graphical information.
2. If several signals have to be displayed, appropriate video scaling is required, suitable for both high-quality video and graphics signals.
3. Conventional algorithms for TV video enhancement need to be reconsidered and reprogrammed for digital TV, MPEG-decoded applications.

Up till now, a secondary video signal on the display is mostly realized with additional dedicated hardware [1], see Fig. 1. This hardware is usually minimized to limit costs, thereby accepting some quality loss. Fig. 1 shows that alternative processing, such as PALplus in Europe, (or equivalently EDTV in Japan) and a 100 Hz up-conversion for large area flicker reduction, may be added. An expensive part of this extra signal enhancement processing is the memory usage: it is distributed among the applications, and if the application is switched off, the memory cannot be re-used. Another point of concern in the diagram is the optimal

Table 1: Computing requirements for a set of typical TV functions (MOPS = 10^6 operations per second).

Function	Operations	Computations	Memory, Bandwidth
H,V zoom	sample-rate conv.	300–500 MOPS	lines, 40-70 MB/s
Filters, Comb filter	multiply-accumulate	200 MOPS	lines, 40-70 MB/s
Mot.Comp. 100 Hz	block subtract, filter	2–4 GOPS	2–3 fields, 256 MB/s
Colour-space conv.	matrix	150 MOPS	samples, 40-70 MB/s
Teletext decoding	various	10 MOPS	1 MB, 50 MB/s

insertion point for digital TV signals. Depending on the signal quality, it is sometimes required to insert a signal at different stages of the signal enhancement processing chain. More generally, it is desirable that functions can be operated on signals at arbitrary positions in a signal flow graph. The new architecture should enable more flexibility in the aforementioned system aspects.

The paper is divided as follows. The system aspects and architectural requirements are further elaborated in Section 2. Section 3 provides two more detailed examples of typical TV functions: peaking for sharpness improvement and scaling for video signal resizing. Section 4 addresses two applications to show the flexibility and programmability of the new concept. In Section 5, the hardware design and key parameters of a commercial IC are discussed. Section 6 concludes the paper.

2. System and Architecture

The most flexible architecture is obtained, when all signal processing algorithms are fully described in software, and executed in real-time by a powerful programmable processor. However, this solution would lead to an expensive ensemble of general-purpose processors. For the TV environment, strict cost constraints have to be satisfied, so that a more heterogeneous approach should be adopted. Let us first analyse how much computing power and memory is involved in typical TV signal-processing applications.

Table 1 shows typical video functions of a high-end TV system. The numbers in the table show the order of magnitude, not exact values. The exact values depend on the applied sampling standard and resolution. In this case, the assumptions are 4:2:2 sampling and 832 samples per video line resolution (suited for widescreen TV). From the table, the following conclusions are drawn.

- All types of signal processing appear. Regular processing, such as filters, but also irregular, like Teletext decoding or motion control.

- There are fundamental operations for different TV functions. An example is sample-rate conversion, which can be re-used for widescreen stretch, zoom, time-base correction, and so on.

- Pixel-based processing is generally 'expensive' in terms of computational power, because of the high sampling rates (up to 64 MHz) for high-end video signals.

- Several functions can be readily implemented in software.

The table shows that, if a number of TV functions such as filters have to be implemented, a plurality of 200-300 MOPS functions would quickly lead to the design of GOPS (Giga) engines and thus large chip sizes. A simpler and more cost-effective approach is created by mapping the mostly used functions on application-specific coprocessors, which are supported by one or more general-purpose processors.

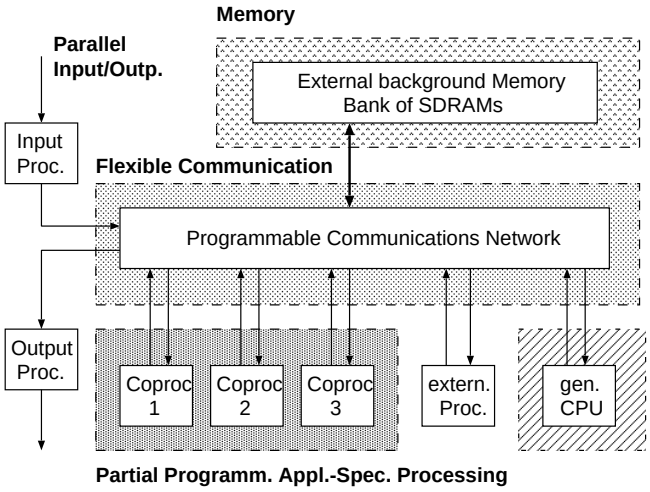


Figure 2: Proposal for a new flexible architecture for high-end TV processing.

This brief analysis reveals the contours of a new flexible architecture for TV applications, which is heterogenous from nature, as the signal processing. Regular and expensive video processing is executed in reprogrammable application-specific processors, whereas low-cost or low-speed functions are implemented in

SW. All processors can be reprogrammed for a secondary signal (or more). Large memory functions are concentrated in one external background memory, thereby enabling re-usage if a particular memory function is switched off. A block diagram of such a system is shown in Fig. 2.

In the following sections, two examples of TV functions are described, and the corresponding coprocessors are defined. In Section 4, these functions will be applied in different TV display modes.

3. Examples of key functions

3.1 Sharpness improvement

3.1.1 Principles and requirements

The subjective attribute sharpness is one of the most important factors in the perception of image quality. The ever increasing demand for high quality –with future digital TV in mind– justifies the presence of a sharpness-enhancement coprocessor as an integral part of the total video processing system. Since both natural video and graphical information will be processed and displayed, the generic sharpness improvement discussed below can be applied for TV images containing both types of signals.

We have confined ourselves to a model for sharpness improvement, called peaking, in which an overshoot is added to the edges around objects in the image [3][4]. This technique has been extended with an advanced adaptive control which uses the local image content, for combating various signal deteriorations such as transmission impairments and aliasing artifacts, thereby enabling the application in a much wider range of video signals.

In peaking, the added overshoot is determined by 2-D high-pass filtering of the input signal. The result can be formulated as:

$$G(j,k) = F(j,k) + \sum_{x=-w}^{w} \sum_{y=-w}^{w} a_{x,y} \cdot F(j+x, k+y), \quad (1)$$

where $F(j,k)$ denotes the input signal and the weight factors $a_{x,y}$ represent the coefficients of the high-pass filter with order w. In this model, the filtered signal acts as an approximation of the second derivative of the original image. With the aforementioned model, a generic sharpness improvement algorithm can be found by locally controlling the amount of overshoot added, i.e. making the addition adaptive to the local picture contents and the transmission impairments measured in the same area. We have concentrated on the following properties for adaptive control of the sharpness enhancement:

– local intensity level and related noise visibility;
– noise level contained by the signal;
– local sharpness of the input signal;
– aliasing prevention, where alias results from non-linear processing such as clipping.

The above-cited points are discussed briefly. Details can be found in [2].

3.1.2 Sharpness control parameters

A *Local intensity and related noise visibility*
The Human Visual System (HVS) is capable of perceiving a large range of intensities, but not simultaneously within an image [5]. This means that the perceived local image intensity depends on the average brightness level (accommodation). This phenomenon has been exploited by compensating the amount of overshoot added in high and low brightness areas (proportional to intensity level). As a bonus, this function reduced the noise sensitivity of the peaking.

B *Local sharpness of the input signal*
For steep luminance transitions, the output of the high-pass filter has generally a high amplitude. This results in a large overshoot at steep edges. To prevent this, the local steepness of luminance transitions are measured. The steepness measurement is based on an approximation of the maximum derivative of the signal in all spatial directions. The approximation is based on determining the dynamic range of the signal within a small surrounding of the desired position. The output of this steepness measurement controls a suppression gain, to reduce the sharpness enhancement at the concerning positions.

C *Noise contained by the signal (adaptive coring)*
Sharpness enhancement amplifies the noise level and thus decreases the signal-to-noise ratio (SNR). Since the output of the enhancement filter is proportional to the local detail in the image, and the noise is distributed over the image, the SNR decreases in low-detail areas, where noise is mostly annoying. By suppressing the sharpness enhancement for low-detail areas, the subjective image quality can be improved. In addition, the amount of suppression is regulated as a function of the average noise level.

D *Aliasing prevention from non-linear processing*
Clipping of the signal for preventing pixel range excursions causes artifacts. Suppression of the sharpness enhancement to prevent clipping should not be performed on a sample basis either (would also lead to aliasing). For this reason, the suppression factor is varied smoothly and is controlled on an area basis (e.g. rectangular blocks).

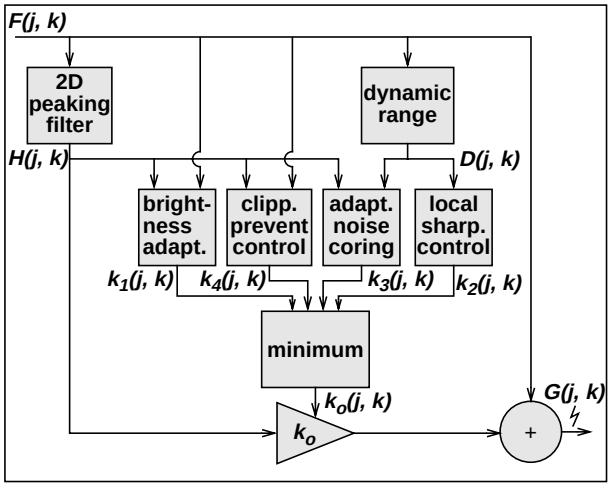


Figure 3: Block diagram of the generic 2D sharpness enhancement coprocessor.

3.1.3 Sharpness improvement diagram

The block diagram of the complete algorithm is depicted in Fig. 3. In the overall sharpness enhancement model, the overshoot at edges in the peaking processing is suppressed by an overall control parameter, which is determined by all individual control parameters discussed in Section 3.1.2. In the complete system, the formulation of the peaking becomes as follows:

$$G(j,k) = F(j,k) + k_o(j,k) \cdot H(j,k), \text{with} \qquad (2)$$

$$H(j,k) = \sum_{x=-w}^{w} \sum_{y=-w}^{w} a_{x,y} \cdot F(j+x, k+y),$$

with $0 \leq k_o(j,k) \leq 1$. When one of the individual contributions of the control blocks $k_i(j,k)$ for $i = 1 \cdots 4$, portrays a large occurrence of the corresponding artifact, the correction of the added overshoot will be large too, leading to a small $k_o(j,k)$. In this model, the smallest correction factor represents the most annoying artifact. When the gain of the smallest artifact measure is applied to suppress the total enhancement, the remaining artifacts will be suppressed as well. For normal TV scenes, it was found experimentally that this decision criterion yields a good performance, although it is a highly non-linear operation.

3.2 Video Scaling

3.2.1 Introduction and requirements

Multi-window TV applications as discussed in Section 1, combined with superimposed graphics and menus,

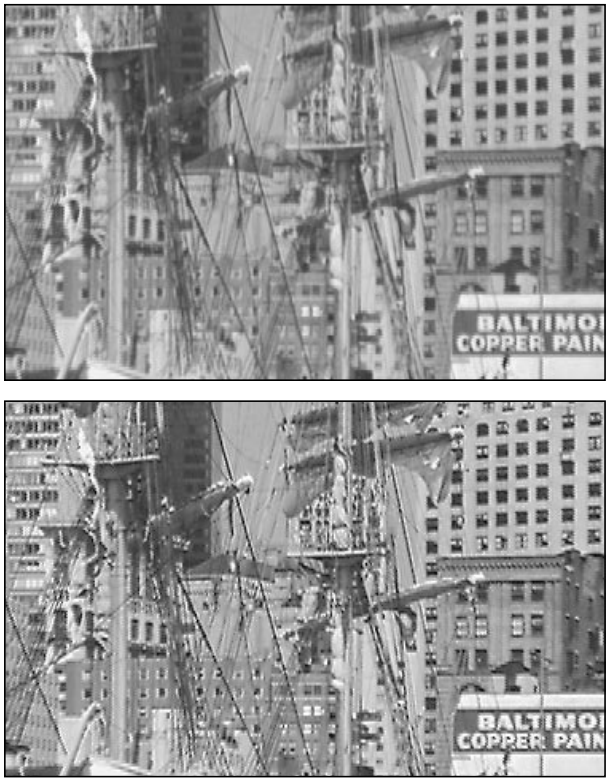


Figure 4: The original 'Baltimore' image (upper picture) and the sharpness enhanced version of this image (lower picture).

rely on sampling-rate conversion (SRC). Instead of using several dedicated SRCs, our aim is to design a more flexible, high-quality 2-D scaler, which can be used for a broad range of TV functions and graphics applications. It is obvious that flexible advanced scalers will become an integral part of consumer electronics and multimedia products. System requirements of the desired high-quality SRC for TV applications are severe and listed below.

Large dynamic scaling range. In multi-window applications, any size of a video window may be used with good quality (not true with current PiP, dual-window TV).
Suitable for graphics and video. The visibility of aliasing for graphics signals differs substantially from that of video signals.
Size matching to the display. The video signal processing with scaling should be applicable to various displays (CRT, LCD, plasma).
Compression and expansion. The converter can be used for compression (e.g. PiP) with the same level of picture quality as for expansion.

3.2.2 Architecture of SRC

The theory involved with digital SRC can be found in [6][7]. Let us assume that the input and output sampling frequencies (f) are related by the ratio K/L. The corresponding conversion scheme is depicted in Fig. 5. Upsampling of the input with a factor K is achieved

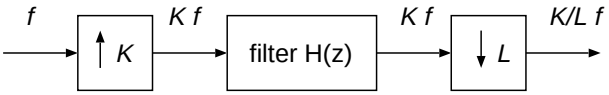


Figure 5: Principle of SRC using digital filtering.

by simply adding zeros between every incoming sample. As a result, the intermediate sampling frequency operates at a sampling rate of Kf. Subsequently, the function $H(z)$ filters out the undesired parts of the video spectrum. Finally, the sequence is decimated by a factor L. Implementations of this scheme typically have a fixed filter $H(z)$, a fixed upsampling factor K, and a variable downsampling factor L. Consequently, the implementation is only suited for e.g. expansion and limited compression.

Generally, digital SRC can be divided into three categories [9], which all can be modelled according to the block scheme depicted in Fig. 5. These categories are (1) curve fitting using (e.g. linear, quadratic) interpolation functions, (2) digital filters such as *polyphase* filters, and (3) hybrid filters such as Variable Phase Delay filters [10], which are often used in TV sets. From these categories, the SRC based on a polyphase filter combines the upsampling, low-pass filtering and decimation into a single function.

Since compression and expansion are both desired key features of the converter, and the previously mentioned techniques give a disappointing performance for compression, a new architecture was investigated [8], satisfying this constraint with a high performance. Whereas conventional SRC implementations for either compression or expansion lead mostly to different architectures, the objective here is to map both properties onto a single architecture. This is achieved by using the concept of *transposition*, which is briefly discussed now.

3.2.3 Transposition of a SRC

The transposed version of a *discrete linear time-invariant network* has the same function as the original network [9]. For transposition, the directions in the network are reversed and all branch nodes in the original network become summation nodes and vice versa. Generalized rules for transposition of *time-varying systems*, such as SRCs, were derived in [11]. These generalized rules state that the main element pairs transpose mutually, so that for example, upsamplers become

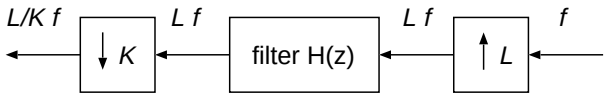


Figure 6: SRC after transposition.

downsamplers and vice versa, an addition becomes a signal split and vice versa, and constant multipliers and delays remain equal.

Applying these transposition rules to Fig. 5 leads to the SRC depicted in Fig. 6. When considering compression, $H(z)$ results in a decimating filter for $L \leq K$ (note that for the non-transposed case in Fig. 5, this filter is an interpolation filter). Since L is variable, $H(z)$ operates on a *varying* sampling rate in the transposed case. As a result, depending on L, a different part of the baseband spectrum is filtered out. In other words, an important property of a transposed SRC is that it inherently prefilters the input signal according to the compression ratio.

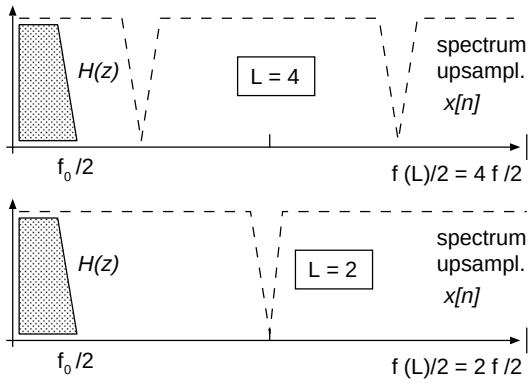


Figure 7: SRC spectral analysis for $L = 4$ and 2.

Example. Assume a decimation factor $K = 8$. In Fig. 7, the spectrum of a signal $x[n]$ is shown after upsampling with factor $L = 2$ and $L = 4$. The upsampled signals are filtered with $H(z)$. After filtering, the signals are decimated by a factor 8. It can be seen that a different part of the baseband spectrum is filtered out, as a result of the choice of the upsampling factor L, thereby showing the main advantage of a transposed filter structure. From Fig. 7 we can conclude that a filter can be designed such that for all $L \leq K$, it is ensured that the filtered signal is bandwidth-limited to $f_0/2$ prior to decimation.

3.2.4 Experiments with polyphase filters

A normal and a transposed implementation of an SRC, based on a 6-tap, 64-phase polyphase filter, have been implemented in software. Fig. 8 portrays a possible

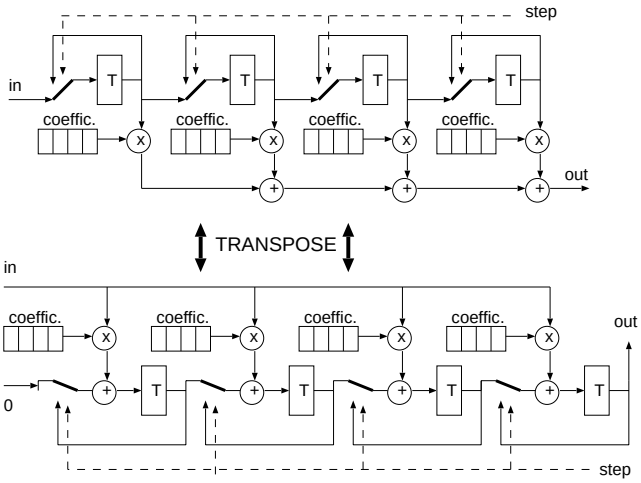


Figure 8: Architecture of a 4-tap polyphase filter.

architecture of the normal and transposed polyphase filter [12]. Filter characteristics have been optimized for both the regular and the transposed mode. The results have been obtained by computer simulations. For more severe testing, the regular and transposed polyphase filter have been implemented on a real-time video system. We have found that the results of the transposed algorithm outperforms clearly the existing implementations in current TV receivers. By switching between transposed and regular polyphase operation, high-quality compression and expansion is possible without adding an adjustable prefilter.

Fig. 9 shows the visual results for a compression factor of 2.6. Fig. 9a portrays that a normal filter leads to substantial aliasing in the multiburst (also the crosshatch pattern). It can be seen that the performance of the transposed filter (Fig. 9b) is much better due to the proper bandwidth limitation. As a result of the scaling of the filter passband, the obtained picture quality remains high for a large range of compression and expansion ratios (experiments with factors between 1 and 16). Furthermore, since the regular and transposed implementation rely on the same resources, implementations of both operation modes can be mapped on the same hardware architecture.

Finally, the suitability for sampling rate conversion of graphics material has been examined. The spectral energy distribution of graphics source material differs significantly from that of video. Without taking extra care, conversion of graphics material may result in visible ringing around steep edges. Therefore, new filters have been designed which reduce considerably the ringing artifacts for graphical images. Also for graphics, the transposition concept has proven to be suitable for realizing high-quality compression.

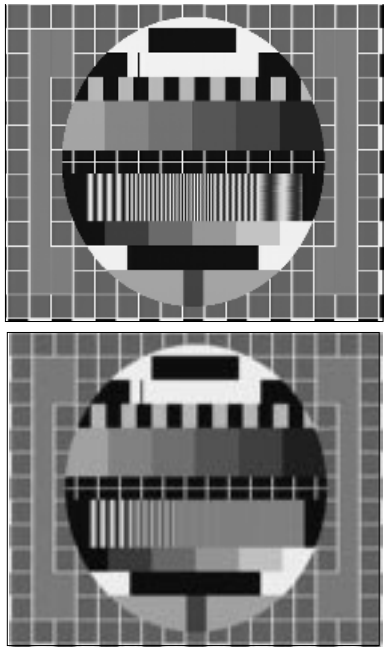


Figure 9: Frequency burst performance of a normal (top) and a transposed filter (bottom) for compression.

4. Examples of TV applications

4.1 Flexibility and programmability

To show the flexibility of the new architecture, two examples are discussed to explain the programmability of the system. The system allows two forms of programmability.

Firstly, each function has been made programmable in performance by using *parameterizable* signal processing functions. The parameter setting of functions can be optimized depending on other TV functions in the flow graph, e.g. more sharpness enhancement is applied when zooming is used. This property allows optimal performance for the total chain of signal-processing functions.

The second form of programmability is that a function can be *re-used for more than one signal stream simultaneously*, for different applications and with individual parameter settings. This property enables a large variety of features. The result is that the individual display functions can be used in an arbitrary order and at different positions in the signal flow graph of the TV processor: a new phenomenon in a TV receiver.

Both forms of programmability will be illustrated. The flexibility inside a coprocessor is discussed first. The flexibility in using a set of various coprocessors is shown afterwards in Sections 4.2 and 4.3.

Let us now briefly discuss the first form of programmability and consider the sharpness enhancement

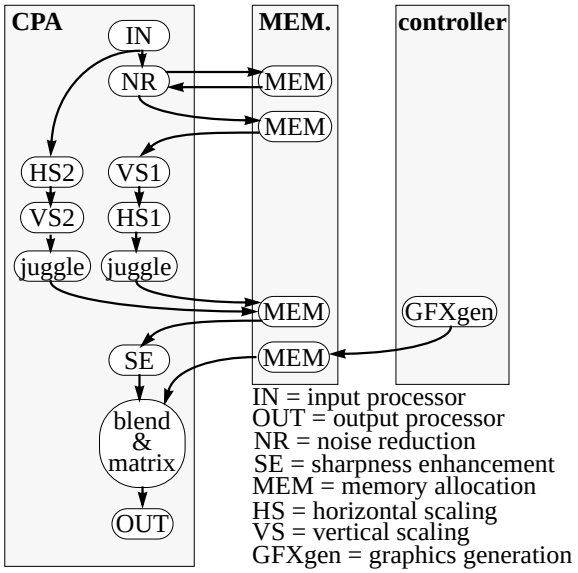


Figure 10: The signal flow graph for zoom and PiP of the same source (CPA is the processor system).

of Section 3.1. By disabling all adaptive control blocks in the sharpness enhancement functionality and by re-programming the flexible two-dimensional filter into a 15-tap horizontal filter, the coprocessor could also be used as a post-processing filter for PALplus decoding. Furthermore, the vertical upsampling of the letterbox and the helper lines in a PALplus signal can be covered with the vertical sampling-rate converter, which was described in Section 3.2. Other steps of the PALplus decoder can be carried out in the remaining coprocessors (It is indicated here, that the experimental IC of Section 5 does not contain all processing for a full-featured PALplus decoder). In this way, the sharpness enhancement is part of completely different processing, and has a different function. Alternative examples of flexibility are provided by (1) the two-dimensional peaking filter having programmable filter coefficients, (2) programmable control on the amount of adaptivity and (3) the possibility to add new adaptive control blocks implemented in software. Let us now discuss two applications, where the same coprocessors are used in different configurations.

4.2 Example 1: Magnifying glass

Besides multi-window applications, like dual-screen and PiP, which can be performed by the system, an interesting example is presented which shows the flexibility of the architecture, and the ease to program very diverse features. Fig. 10 shows the signal flow graph of a special zoom application. In this application, the background picture is a zoomed-in part of the foreground PiP picture (see Fig. 11). To be able to apply two different kind of processing steps to one input source, the video input processors contain the operation to copy its output onto two successor coprocessor inputs. Consequently, the video signal is treated as two independent sources which are processed separately. The first source is expanded (zoomed-in) in both horizontal and vertical directions, whereas the second source is compressed in both horizontal and vertical directions (PiP size). The part of the image that is zoomed-in, is selected or interactively controlled by the customer by means of a window which can be moved around in the PiP image. This window is generated by a microprocessor control program and is subsequently mixed with the video signal using a "blend & matrix" coprocessor (see Fig. 10). This coprocessor can mix or blend two video signals.

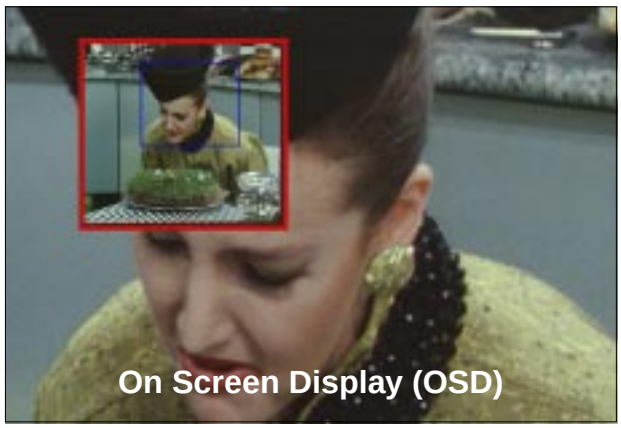


Figure 11: Picture of the "magnifying glass" feature.

Prior to vertical conversion for zooming, the data is stored in memory. This has two reasons. Firstly, the data rates at the input and output of the background memory are generally different and depend on the chosen scaling factor. This implies that the data rate at the input of the vertical scaler (VS) f_i equals $f_i = f_o \cdot S_v$, with f_o the output rate and S_v the vertical scaling factor. For expansion (zooming) $0 < S_v < 1$, whereas $S_v > 1$ enables compression. Since we want to limit the maximum video data-rate in the system, this relation requires that in the case of expansion, f_i decreases for a constant f_o, whereas for compression, the output rate f_o decreases, given a constant f_i. Consequently, buffering to do compression is necessary after scaling, while for expansion (zooming), buffering is needed prior to the scaling operation. A second reason to store the data in memory prior to conversion, is that a high-quality vertical conversion needs a field delay to de-interlace the signal prior to scaling. It is indicated that flexibility in the processing order of horizontal and vertical scaling can further optimize the intermediate memory cache use (e.g. horizontal zooming after vertical zooming).

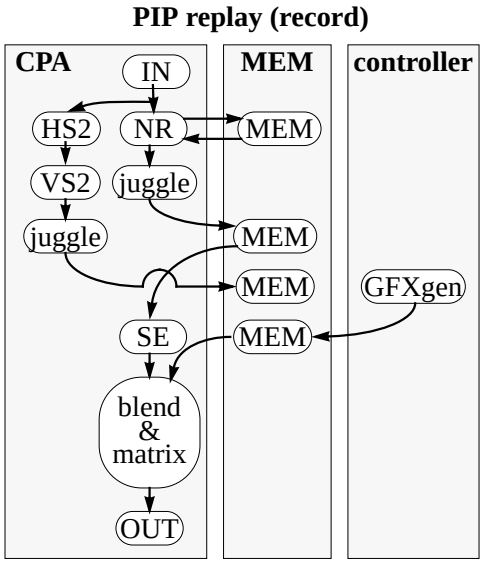


Figure 12: Signal flow graph for recording a PiP signal.

After high-quality scaling of the video streams, the data is written into memory by the video jugglers. Such a 'juggle' task in the signal flow graph contains programmable memory-address generation to mix the PiP and the background image at the correct positions in the memory. For further processing of the combined video streams, the video data is read from memory again. In the signal flow graph of Fig. 10, the sharpness enhancement coprocessor as discussed in Section 3.1, reads the image from the memory and increases the sharpness. The result of this final image processing step is transferred to the 'blend & matrix' coprocessor. The second input source of this coprocessor is used to insert graphics, which are off-line generated in the memory by the controller. Prior to blending of the two input signals, the YUV 4:2:2 video format of the composed "magnified" picture is converted to an RGB format similar to the format of the graphics. The final result is shown in Fig. 11.

4.3 Example 2: PiP replay

An alternative example to show the flexibility of signal flow graph programming is shown in Fig. 12 and Fig. 14. PiP replay is a feature in which the last x seconds of a video stream is "recorded" into the background memory, and after recording replayed in a so-called shuttle or repeat mode. In order to use the background memory efficiently, the recorded source is scaled to PiP format. An additional compression coprocessor and temporal subsampling would enable a further reduction of the necessary amount of memory and its bandwidth. Because recording and playback

Figure 13: Picture of a recorded sequence and the replayed PiP in the upper right corner.

are different functionalities at different times, two different signal flow graphs are used for writing into and reading from the memory.

4.3.1 PiP record

When the PiP record mode is switched on, the system starts recording the current program, until the "user" stops the recording and switches to playback. When during record the background memory becomes full, the memory will be cyclically overwritten until PiP record is stopped.

Since the user is still watching the current program during recording, two different types of processing are performed on the same input source, similar to the first example in Section 4.2. This requires the already discussed "signal-copy" functionality of the input processor. One of the signals is processed and displayed, while the other signal is scaled with a programmable scaling factor in both horizontal and vertical directions and subsequently recorded. Fig. 12 shows the signal flow graph for PiP recording.

4.3.2 PiP playback

The second part of PiP replay is playback. The signal flow graph for this part of the PiP replay feature is shown in Fig. 14. To mix the scaled PiP frames with the main video stream, the recorded frames are read from the memory and written back at the correct positions. During playback, the inserted PiP can either be shuttled, repeated or displayed in a trick mode (slow-motion, fast forward, fast backward, still, etc.). All these playback modes require their own specific memory control. Note that the PiP source may optionally be zoomed-out, before it is combined with the main source signal. The result of the PiP replay is shown in Fig. 13.

Comparing example 1 and 2, completely different features were build using the same set of coprocessors in a different signal flow graph configuration. For each of the flow graphs, an optimal parameter setting could

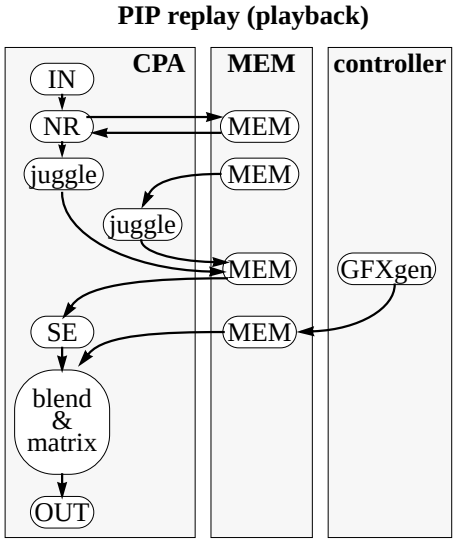


Figure 14: The signal flow graph for replay of a PiP.

be adopted. Because several inputs and outputs are available in the system, the same hardware can also be configured to perform multi-window applications.

4.4 Memory bandwidth limitation

The combination of a group of coprocessors and an external background memory has a limitation with respect to the amount of memory access that is enabled. The upper limit is defined by the memory bus width and the operational clock frequency. The required bandwidth of an application can be found by adding the bandwidths that correspond with each of the arrows to and from the memory indicated in the flow graphs (see Examples 1 and 2).

A second hardware limitation is the parallelism and the bandwidth of each individual coprocessor, i.e. the number of video sources that can be processed in parallel and its total throughput rate. Examples of such limitations are the number of streams that can be handled by the juggler or the number of video sources that can be scaled simultaneously.

Let us start with the last contraint. Table 2 summarizes for some of the coprocessors the throughput rate and parallelism implemented. It can be seen that the conditions for each coprocessor are rather different; this is caused by analyzing the desired useful video applications in the TV environment. For example, sharpness enhancement (SE) is merely used as a final post-processing step of the complete image, so that it is limited to handle only one video stream. In contrast with this, the horizontal scaler, which is used for many tasks, is able to process up to three video sources simultaneously, with a total bandwidth of 128 MB/s.

Table 2: Overview of the parallelism and maximum bandwidths of each coprocessor.

coprocessor	parallism	Bandwidth
HS	1 full color	<128 MB/s, or
	2 full color	<64 MB/s, or
	2 full color	one <32 MB/s + one <96 MB/s, or
	3 full color	two <32 MB/s + one <64 MB/s
VS	1 full color	<128 MB/s, or
	2 full color	<64 MB/s, or one <32 MB/s + one <96 MB/s
SE	1 full color	<64 MB/s
juggler	8 full color	total bandwidth <256 MB/s

The external memory bandwidth is a major limitation of the complexity of the complete TV application to be executed. To further elaborate on this issue, example 1 of the "Magnifying glass" feature is used to calculate the requirements. The derivation starts with formulating some general assumptions.

- Sample frequency of all sources is assumed to be $f_i = 16$ MHz;
- Internal video format is assumed to be YUV 4:2:2;
- Only active video data is processed and stored. The size of a field memory therefore becomes $832 \cdot 288 \cdot 2 = 0.48$MB;
- Clock frequency of the background memory is 96 MHz which implies a maximum available memory bandwidth of 384 MB/s (32-bit bus);
- Graphics generation is not taken into account.

In Table 3, the requirements for the amount of memory and the memory bandwidth are calculated. The calculations are performed for the worst case in which the horizontal and vertical conversion factors (S_v and S_h) equal unity. The following system aspects are emphasized.

- Mixing of the two video sources requires two field memories to be able to synchronize the interlaced video.
- Since only the visible parts of the video streams have to be stored in the memory, the bandwidth is equal to writing and reading 1 video stream.
- The VS requires less memory resources, due to the horizontal pre-processing with the HS.

Summarizing the memory requirement calculations of the example, only 1.92 MB of the background memory

Table 3: Memory capacity and bandwidth for the "Magnifying glass" feature.

	Coproc's connection to/from memory	Mem. (MB)	Memory bandwidth (MB/s)
Jugglers (write/read)	2/1	0.96	96
VS (Zoom)	1/1	<0.48	<96
NR (Zoom)	1/1	0.48	64
Total	**4/3**	**1.92**	**256(213)**[1]

[1]For the number between brackets, part of the horizontal blanking period is used for memory transfer. This reduces the net required bandwidth with about 20%.

is used with less than 60% of the total memory bandwidth, to create the "Magnifying glass" feature with high picture quality.

Besides memory bandwidth limitations, the set of possible flow graphs is limited by the variety of functions that can be derived from the set of coprocessors integrated.

5. Hardware design and IC

5.1 Hardware architecture requirements

The system requirements from Section 1 have been solved by sharing the hardware and memory for similar signal processing and by enabling a higher clock frequency. Bearing this strategy in mind, a new chip set for high-end TV applications is designed. The key features of the chip set are that:

- comparable TV functions are mapped on the *same* hardware processing unit;
- large field memories and other similar storage functions are shared in the *same background memory*;
- the processing hardware runs on a *multiple* of the intrinsical video clock rate to enable sufficient parallelism;
- a communication structure with sufficient parallelism provides simultaneous transfer of multiple data streams.

This novel concept has been used to implement a set of enhancement and signal mixing functions of a high-end TV receiver. Examples of such functions have been provided in Sections 3.1 and 3.2. It is interesting to implement the baseband signal-processing part of the TV receiver with this new architecture, because in the enhancement part, sharing of memory and signal-processing hardware is most beneficial. The main TV signal-processing functions implemented are:

– horizontal and vertical scaling of video to any display format (see e.g. [8]);
– adaptive noise reduction;
– sharpness improvement [2] for both luminance and colour;
– low-cost field-rate conversion ("digital scan").

The Fig. 2 from Section 2 is representative for the architecture of the IC implementation. Each of the above-cited functions are covered by a single coprocessor. Such a coprocessor can perform a video function for a few signals simultaneously. The way the processing units are connected is programmable, so that the sequential order of TV processing functions is flexible.

Each coprocessor is an autonomous unit, independent of all other processors in the system and it is optimized for executing its own functionality. The data communication between processors is defined by simple *data-driven* communication rules. These rules are according to Kahn's language for parallel programming [14], in which one axiom is that data is processed only when video data is received and available at the input. It has been shown that large multi-processor systems build upon such processing units can execute programmable signal flow graphs (dynamic data flow). This is enabled by using synchronous (deterministic) processing functions and run-time scheduling [15].

5.2 Experimental IC

The IC is running on a clock frequency of 64 MHz, which allows up to four TV signals of 16 MHz clock rate to be processed in parallel. The interconnections are implemented by a fully programmable switch matrix, which supports any connection between coprocessors to transfer video streams. This matrix concept was adopted earlier for programmable video processing applications [13]. As a result, virtually any type of signal flow graph can be programmed into the IC, which makes the application area highly versatile. Examples of typical flow graphs have been presented in Sections 4.2 and 4.3. Table 4 gives a summary of the key parameters of the experimental IC, which will be tested at the end of 1998.

6. Conclusions

A new flexible heterogeneous processor system for TV applications has been presented, featuring multi-signal display and programmable video functionality. The architecture of the system is based on an array of

Table 4: Key parameters of experimental IC.

Coprocessors	6 types
Input/output proc.	3 / 2
Hor. resolution	$\leq$ 848 samples
Ver. resolution	$\leq$ 600 lines
Clock frequency	64 MHz
External memory	96 MHz SDRAM
Memory bus bandw.	$\leq$ 384 MByte/s

application-specific coprocessors, each of which being programmable in performance and quality. A key feature of the system is that it enables the programming of signal flow graphs, which define the routing of one or more video signals in the architecture. The concatenation of subfunctions results in various TV applications. Examples provided are a "magnifying glass" and PiP recording and playback. Because of this flexible routing of signals, many types of signal flow graphs can be programmed into the IC, which makes the application area highly versatile. Furthermore, it can be concluded that this coprocessor-array architecture is well scalable and extensible to the future by adding new processing units. As a bonus, in any new configuration, the picture quality setting and the order of signal processing tasks can be re-optimized for the new set of coprocessors and the corresponding applications. The flexibility of the system allows that the system may be applied in combination with fully programmable processors, such as VLIW processors to perform a more extensive set of processing functions derived from a complete TV system.

References

[1] J. Veerhoek and A. Stuivenwold, "Economy Picture-In-Picture Processor", Proc. IEEE Cons. Electr. Conf., Digest Techn. Papers ICCE98, IEEE Publ. 98CH36160, pp 350–351, June 1998.

[2] E.G.T. Jaspers and P.H.N. de With, "A Generic 2D Sharpness Enhancement Algorithm for Luminance Signals", Sixth Conf. Image Proc. & its Applic. IPA97, Dublin, 1997, IEE Publ.No. 443, Vol. 1, pp. 269–273, July 1997.

[3] W.K. Pratt, *Digital Image Processing*, 2nd. ed., John Wiley & Sons, Inc., New York, 1991.

[4] R.H. Wallis, "An Approach for the Variant Restoration", Proc. Symp. Current Math. Problems in Image Science, Monterey, USA (CA), November 1976.

[5] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Addison-Wesley Publishing Company Inc., June 1992, Chapter 2.

[6] A. Papoulis, *"Systems and Transforms with Application in Optics"*, McGraw-Hill, New York, 1968.

[7] B. Wendland, H. Schröder, *Fernsehtechnik II*, Hüthig Verlag, Heidelberg, 1991.

[8] J.G.W.M. Janssen, J. Stessen and P.H.N. de With, "An Advanced Sampling Rate Conversion technique for Video and Graphics Signals", Sixth Conf. Image Proc. & its Applic. IPA97, Dublin, 1997, IEE Publ.No. 443, Vol. 2, pp. 771–775, July 1997.

[9] L.B. Jackson, *Digital Filters and Signal Processing*, Kluwer, 1986.

[10] A.H. Nillesen, "Non-Integral Delay Circuit", United States Patent, US 5,349,548, September 1994.

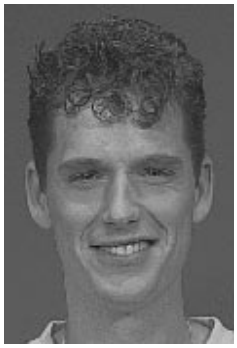
[11] T.A.C.M. Claasen, W.F.G. Mecklenbräuker, "On the transposition of linear time-variant discrete-time networks and its application to multirate digital systems", Philips Journ. Research, Vol. 33, pp. 78–102, 1978.

[12] A.J. van Dalfsen, J.H.C.J. Stessen, J.G.W.M. Janssen, "Sample rate conversion", European Patent Application, EP-A 96203035.9, October 1996.

[13] H.J.M. Veendrick, O. Popp, G. Postuma, M. Lecoutere, "A 1.5 Gips Video Signal Processor (VSP)", Proc. CICC, 6.2, San Diego CA, May 1994.

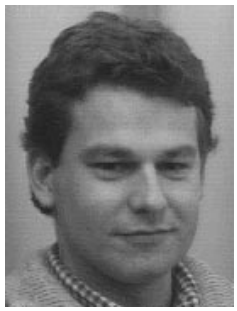
[14] G. Kahn, "The Semantics of a Simple Langue for Parallel Programming", Information Processing 74 (Proc. of IFIP congress 1974), pp. 471–475, August 1974.

[15] J.T. Buck and E.A. Lee, " Scheduling Dynamic Dataflow Graphs with Bounded Memory using the Token Model", Proc. of ICASSP '93, Vol 1, pp. 429–432, April 1993.



Egbert Jaspers was born in Nijmegen, The Netherlands, in 1969. He graduated in electrical engineering from the Venlo Polytechnical College in 1992 and subsequently, he joined Philips Research Laboratories in Eindhoven. For one year, he worked on video compression for digital HDTV recording. In 1993, he continued his education at the Eindhoven University of Technology, from which he graduated in electrical engineering (ir. degree) in 1996. In the same year, he joined Philips Research Laboratories Eindhoven, where he became a member of the TV Systems Department. He is currently involved in the research of programmable architectures and their implementation for TV and computer systems.

Peter H.N. de With was born in Lexmond, The Netherlands, in 1958. He graduated in electrical engineering from the University of Technology in Eindhoven. In 1992, he received the Ph.D. degree from the University of Technology Delft, The Netherlands, for his work on video bit-rate reduction for recording applications. He joined Philips Research Laboratories Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Department. From 1985 to 1993 he was involved in several European RACE projects (a.o. 1001 and 2026) in which digital SDTV, HDTV and data recording was studied. In the early nineties he contributed as a video coding expert to the DV standardization committee. Since 1994 he became a member of the TV System group where he is working on advanced video processing architectures for various enhancements and compression systems. In 1996 he became senior TV systems architect and in October 1997 he was appointed as full professor at the University of Mannheim, Germany. Regularly, he is a teacher of the Philips Centre for Technical Training and for other public post-academic courses on digital video compression and recording and video processing issues. In 1995 he co-authored the paper that received the IEEE CES Transactions Paper award. In 1996, he received a company Invention Award. In 1997, Philips received the ITVA award for its contributions to the DV standard. Mr. de With is a senior member of the IEEE, and board member of the IEEE Benelux Chapter on Consumer Electronics and the Benelux working group for Information and Communication theory.



Johan G.W.M. Janssen was born in Roggel, The Netherlands, in 1968. He graduated in electrical engineering (ir. degree) at the Eindhoven University of Technology in 1993. In 1993, he joined Philips Research Laboraties Eindhoven and became a member of the TV-systems department. He was involved in programmable solutions for high-speed TV-related algorithms, and studied the required computing power for mapping such algorithms onto a CPU. Afterwards, he joined a project on programmable, scalable architectures for TV systems. He was responsible for defining a number of high-end TV signal processing functions and their architectural requirements. Recently, he joined Philips Research Laboratories Briarcliff (NY), where he is working on Digital TV systems.