A Flexible Heterogeneous Video Processing Architecture for Powerful HQ Television Applications "Finding the optimal balance between HW and SW"

Peter H.N. de With[1] and **Egbert G.T. Jaspers**[2]

[1]University Mannheim, Fac. Computer Engineering, B6-26, 68131 Mannheim, Germany.
[2]Philips Research Labs, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands.

Abstract – *A new architectural concept for high-quality television is presented, featuring multiple video window display with high flexibility and programmability. Key issue of the proposal is to create SW programmability for video signal manipulation at the upper levels and have cost-effective hardware support for pixel-based video processing.*

1. INTRODUCTION

In the consumer electronics area, the display technologies of television and computers are gradually merging, as a result from upcoming new features such as internet, electronic program guide and shopping, help wizards, games and video conferencing. Consequently, video processing for displaying high-quality television applications clearly points towards the consumption of several information channels in parallel, instead of watching a conventional single-channel broadcast. A key feature of such a TV set is that two signals (or more) can be monitored in real time on a TV display, together with graphical information. An additional desirable feature of such a TV set is that the installed signal processing power can be redistributed over the amount signals to be processed. This flexibility can be used for improving overall picture quality, because signal enhancement is preferably used for applications where it is needed mostly.

The system requirements for a new architecture are influenced by occurring trends in TV set design and computer systems. Requirements are as follows.

- With the introduction of digital TV and digital image enhancement techniques, TV is becoming very diverse in terms of features, requiring a flexible and reconfigurable system.
- TV is influenced by PC technology with respect to computing techniques, SW architectures and applications.
- Since memory devices continuously become larger and cheaper, it is desirable to have one uniform shared background memory.
- New features and standards from international bodies have to be realized in a short time frame and require an extensible and scalable system. Parts of the necessary signal processing are frozen, whereas in other parts programmability is allowed or even exploited intensively.
- New applications in a TV set need a cost-effective implementation, since it is a highly competitive market.

Several architecture proposals for TV have been developed [1][2], but they lack the flexibility and programmability that is required for the aforementioned requirements.

This paper is divided as follows. Section 2 shows the computational requirements of typical TV functions. In Section 3, an architectural framework is presented for further development. Section 4 provides an example how the architecture is programmed for a typical TV setting. Section 5 further elaborates on this topic in terms of costs and memory requirements. Section 6 outlines an experimental chip design and Section 7 describes the conclusions.

2. COSTS OF VARIOUS TV FUNCTIONS

At first glance, it seems that the required architecture could be optimally implemented on a fully programmable multi-media processor. The question to be answered subsequently is how much computing power should be realized. For this reason, a number of typical TV signal-processing functions were studied and computational and memory requirements were evaluated.

Table 1. *Expenses of various TV functions.*

Function	Operations per Second	Memory / Cache
H,V zoom / compress	400 MOPS	samples-lines
Filters, Comb filters	200 MOPS	samples-field
Advanced Peaking	650 MOPS	lines
Color Transient Improvement	300 MOPS	samples
Dynamic Noise Reduction	500 MOPS	field
MC-100 Hz	2 – 4 GOPS	2 – 3 fields
Color Space	150 MOPS	None
Teletext conversion	10 MOPS	> field
Adaptive Luminance	60 MOPS	1 KByte

Table 1 shows the intrinsical processing power of several TV functions. With respect to operation counting, additions, multiplications, etc., are considered as single operations. The third column shows the amount of memory or cache required. Here it is assumed that information can be retrieved or stored by single reads and writes (which is optimistic). For a normal TV application with 50-to-100 Hz conversion, Picture-in-Picture (PiP), noise reduction and aspect-ratio conversion, the amount of operations already exceeds 6 GOPS (Giga operations per second). This is not readily implemented on a general-purpose proces-

sor cost-effectively. With respect to storage, it can be noticed that caching of the field memories for e.g. MC-100Hz conversion, dynamic noise reduction or comb filtering is too expensive for on-chip integration as embedded memory. Instead, these functions are integrated in a large unified memory in the background. However, the consumption of already scarce bandwidth to this memory should be monitored.

Summarizing, the set of TV functions cannot be implemented cost-effectively on a fully programmable multi-media processor at this moment. Therefore, a solution based on a set of parallel processors is pursued. Let us now discuss how the required parallelism can be obtained.

3. ARCHITECTURAL ASPECTS

Most multi-media processors use so-called custom operators or functional units to increase the concurrency of processing on a fine-grain instruction level. Although better than a superscalar RISC core, the amount of parallelism obtained in this way is still limited, typically a factor of 3-5. More cost-effective implementations can be realized by focussing the processor hardware on the target application domain. By adding application-specific coprocessors to the general-purpose processor which contain complete TV functions, the grain of parallel operations is raised to the level of complete functions [3]. This enables a larger parallelism in processing, provided that sufficient parallelism in data exchange between coprocessors is implemented.

Communication between the coprocessors and the CPU via a central bus and the memory is limited in terms of bus bandwidth and memory bandwidth. Therefore, a special communication network is applied to reduce the amount of required memory bandwidth and to enable direct communication between the coprocessors without using the background memory (if possible). The communication network is based on a full switch matrix to enable all possible connections between coprocessors, memory and input/output of video signals. Figure 1 shows the hardware architecture of the system. From this architecture, the following properties are defined.

- The architecture is open and allows that new modules can be added after first implementation.
- The coprocessors are connected together via a programmable communication structure.
- The coprocessors are partially programmable and each coprocessor is optimized for a particular TV function, or a set of comparable functions.
- The system has one large uniform background memory for reduction of total system costs. Large memory functions, such as field delays, are mapped into this memory.
- The signal flow in the system is programmable and partially, the amount of parallelism as well.
- Parallel computing power is obtained by running the system on a multiple of the video clock frequency. This basically enables the processing of several video signals simultaneously.

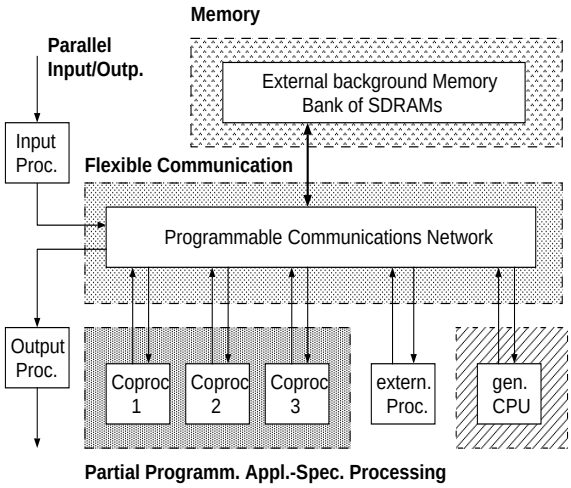

Figure 1. *Hardware architecture of the system.*

This novel concept has been used to implement a set of display processing functions of a high-end TV receiver. Display processing functions have been chosen, because the required parallelism for multiple signals becomes most apparent for DSP functions just prior to display. Examples of display functions are horizontal and vertical video scaling to any display format, noise reduction, sharpness improvement for both luminance and color, simple field-rate conversion, and graphics mixing and blending. The TV functions that are implemented in software are e.g. Teletext, On-Screen-Display (OSD), graphics generation, modem functionality and software applications like picture downloading, message reception, and smart user controls.

4. FLOWGRAPH OF TYPICAL TV SETTING

4.1 Concept of flow-graph system

Let us consider a full-featured application (see Fig. 2), executing on the proposed system architecture. The application shows two real-time live video sources, one expanded at the background and one compressed to a Picture-in-Picture (PiP) format (upper left win-

Figure 2. *Example of a multi-window application.*

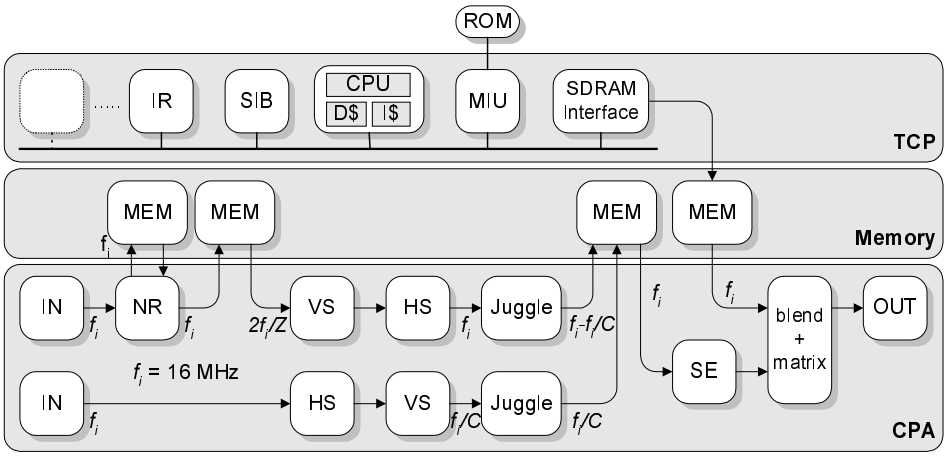

***Figure 3**. The signal flow-graph of the full-featured multi-window application.*

dow). The second window shows an Internet page which was generated by the general-purpose CPU. The CPU generates the internet page and the graphics borders of the windows and writes this into the shared background memory. This image can be read for each video field by a graphics blender which is available as a hardware coprocessor. The Internet page is received via a modem that is connected to the general-purpose CPU via a Serial Interconnect Bus (SIB). The browser application, which is located in a program ROM, is executed on the CPU and generates the graphics.

Figure 3 shows the complete signal flow-graph of this combined video-graphics application. The general-purpose CPU and other peripherals are connected to the SDRAM interface via a central bus. The peripherals used in this application are the Serial Interconnect Bus (SIB) to retrieve the input modem signal, the Infra-Red (IR) module for user remote control, a Memory Interface Unit (MIU) which interfaces with the program ROM, and the SDRAM interface to connect the uniform background memory. The aforementioned peripherals and the CPU are included in one component, called the Telecommunication and Control Processor (TCP). The additional coprocessor hardware performing the computational expensive regular processing, together with the reconfigurable communication network are called the coprocessor array (CPA). Since all registers in the CPA are memory mapped, communication between the CPU and the coprocessors is performed by memory I/O.

4.2 Motivation of the order of functions

Let us now discuss the example flow-graph in more detail. Prior to mixing the real-time video with the graphics in the blender, the two video sources are combined in the background memory. This means writing into the memory with the correct size and with correct positioning in the image plane. Up to this point, the video signals are processed using two separate video signal flow-graphs. The upper subgraph creates the zoomed-in (or expanded) background image and contains an input task (IN), a noise-reduction task (NR), a vertical and horizontal scaler (VS and HS), and a juggle task that performs the memory-address generation. The lower subgraph creates the PiP. Note that temporal noise reduction is applied onto the background image prior to expansion. Note furthermore, that this image is also written into the memory prior to vertical scaling. The reason for this memory access is threefold.

Firstly, it enables the vertical scaler to read only the part of the image that has to be expanded.

Secondly, if vertical scaling is applied to interlaced fields, the result may suffer from special aliasing artifacts. The aliasing that is inherently present in interlaced fields due to the vertical subsampling, is symmetrically distributed over the vertical frequency axis. However, if vertical scaling is applied to the separate fields, this symmetrical distribution may be distorted, leading to annoying visible vertical aliasing components. As a solution, the interlaced video is read progressively from the memory and is first de-interlaced by the scaler, by means of a three-tap median filter. Afterwards, vertical scaling is performed.

The third reason for the memory access prior to vertical scaling is that the output of the scaler is driven by the rate of the output, which is f_i. Due to the expansion, the number of input video lines is less than the number of output video lines. Because the video part that is read by the scaler is expanded to the original image size, the rate of the input video lines is exactly two times the expansion factor Z smaller than the output rate. The factor two results from the progressive reading of the video lines. Thus the input rate of the vertical scaler equals $2f_i/Z$, while the rate of the video source is f_i. This rate difference requires some buffering which is provided by the memory access. As was mentioned before, some memory resources and memory bandwidth could be saved when the scaling would provide intrafield processing, instead of interfield, at the cost of some picture quality. For creation of the PiP, the data rate at the output of the scaler is smaller than the input rate. Buffering to equalize this rate difference is combined with the mixing of the two video sources in the memory.

The order of horizontal and vertical scaling is relevant for memory usage. Since the vertical scaler needs some cache memory to store subsequent video lines

and the coprocessor should have the ability to process multiple video streams simultaneously (task switching), the division of the cache memory amongst the vertical scaler tasks is programmable. For PiP creation with optimal cache usage, the horizontal (down)scaling is therefore performed prior to vertical scaling and for expansion of the background picture, horizontal scaling is carried out after vertical scaling.

5. TASK AND MEMORY COMPLEXITY

From the previous section it can be derived that programming of an application is straightforward and is limited to programming of the signal flow-graph and the individual setting of the tasks. The real limitations are within the task resources of the individual coprocessors and the memory capacity and bandwidth. The task resources of the coprocessors used are shown in Table 2 (gross signal rates). For wide-screen standard-definition (SD) signals in the application of Figure 3 (f_i = 16 MHz), the task resources are more than sufficient.

Let us now consider the memory bandwidths. The memory device runs at a clock rate of 96 MHz and has a bus width of 32 bits. For 16-bit pixels, this means a total memory bandwidth of 192 Mpixels/s. Since all communication between the CPU and the coprocessors is performed via memory, part of the memory bandwidth cannot be used for video processing. Assuming 30 MByte/s of memory bandwidth for control and communication between the CPU and the coprocessors, a bandwidth of 177 Mpixels/s remains for video processing. Table 3 presents an overview of all memory accesses, including the necessary amount of memory and the number of inputs to and outputs from the memory. Firstly, the noise-reduction (NR) coprocessor accesses the memory to use a field memory for advanced adaptive temporal IIR filtering. For an SD image of 288 lines x 854 pixels with 2 Byte/pixel, the required amount of memory equals 0.49 MByte. For a pixel rate of 16 Mpixels/s, the total memory bandwidth for writing and reading is 64 MByte/s.

***Table 2**. Task resources of the coprocessors.*

Horizontal scaler	1 x < 64 Mpixels/s or 2 x < 32 Mpixels/s or 2 x < 16 + 1 x < 32 Mpixels/s or 1 x < 16 + 1 x < 48 Mpixels/s.
Vertical scaler	2 x < 32 Mpixels/s or 1 x < 16 + 1 x < 48 Mpixels/s
Sharpness enhancement	1 x < 32 Mpixels/s
Noise reduction	1 x < 16 Mpixels/s HQ or 2 x < 16 Mpixels/s MQ
Color conversion, GFX expansion, GFX blending	1 x < 64 Mpixels/s
Video inputs	3 x < 64 Mpixels/s
Video outputs	3 x < 64 Mpixels/s
Memory Inputs Memory Outputs	8 inputs + 12 outputs total < 192 Mpixels/s

***Table 3**. Mem. requirements for example application.*

	Connections to / from Mem	Memory (MByte)	Mem. bandwidth (MByte/s)
NR (expansion)	1/1	0.49	64
VS (expansion)	1/1	<0.98	<96
Juggling (write/ read, worst case)	2/1	0.98	64
GFX	0/1	0.49	32
Total (worst case)	**4/4**	**2.94**	**256(218)**

The memory requirements for the access prior to vertical scaling are different. The image is written with 16 Mpixels/s, but is read with $2 \bullet 16/Z$ Mpixels/s for interfield processing with Z being the expansion factor. Because $Z > 1$, the bandwidth to the memory is smaller than 96 MByte/s. If intrafield processing is used for vertical scaling, the data rate is even less than $16/Z$ Mpixels/s.

Computation of the required buffering is less straightforward. If interfield processing is used, a complete field of L_f lines has to be written in the memory and cannot be overwritten before it is read twice. Therefore, the amount of required memory in the progressive video part that is expanded equals:

$$Buf_{\text{inter}} = 2\frac{L_f}{Z}.$$

For intrafield scaling, buffering is only necessary to compensate for the rate difference between the writing to and reading from the memory. In this case, the time for writing one field is equal to the time for reading one field. Writing of the video lines to be expanded is done at a higher rate than reading. The maximum distance in the memory between the read and write pointer is equal to the memory space that has to be buffered for the rate difference. This maximum distance is reached when the write pointer has just finished the field. The part of the field that has been read at that time is $1/Z$. Therefore, the part not yet read is $1 - 1/Z$. As a result, the buffering that is required to deal with the rate difference equals:

$$Buf_{\text{intra}} = \frac{L_f}{Z}\left(1-\frac{1}{Z}\right).$$

Since it is desired to have a static memory allocation, the maximum buffering can be found as follows:

$$\frac{d}{dZ}\left(Buf_{\text{intra}}\right) = -\frac{L_f}{Z^2}\left(1-\frac{2}{Z}\right) = 0 \Rightarrow Z = 2$$

$$Buf_{\text{intra}} = \frac{L_f}{2}\left(1-\frac{1}{2}\right) = \frac{L_f}{4}$$

For L_f =288, the amount of required buffering is Buf_{intra}= 0.12 MByte.

Finally, the mixing or juggling of the video streams is discussed. Also this function is designed such that it requires a minimum amount of memory and memory bandwidth. In the background memory, one frame memory is allocated to construct the mixed image. This frame memory is filled with the background video image, except for the pixel positions where other video

windows are located. The second video window is written into the unfilled area that is created in the background picture. The total amount of data stored is thus equal to the data of one complete picture and similarly, the total required bandwidth equals the bandwidth for writing one complete video stream. Using a frame memory also synchronizes the two video streams.

For generation of the graphics in the background memory, a field or frame memory could be used depending on the desired quality. When a field memory is used and the content is read for both odd and even fields, the amount of memory is reduced at the cost of some vertical resolution. Since synthetically generated graphics may contain high spatial frequencies, the use of a frame memory may result in annoying line flicker when the memory is displayed in interlaced mode. For this reason, only a field memory is used.

Summarizing, the total amount of memory is less than 3 MByte and the maximum memory bandwidth is 256 MByte/s. This necessary bandwidth is only used for transfer of the active pixels. At the positions of the blanking, no data is transferred, thereby decreasing the bandwidth significantly. To be able to increase the available peak bandwidth, the data transfer can be equalized in time. To do this, the read and write tasks of the CPA have the ability to spread the transfer of an active video line over the time of a complete video line including the horizontal blanking. Common video signals contain 15 % horizontal line blanking, so that the total amount of bandwidth can be reduced by 15 %. This leads to a total memory bandwidth of 218 MByte/s.

The previously discussed application is an example from a large set of possible features [4]. The background image could also be a graphics wallpaper. On top of this wallpaper, two live video windows could be displayed together with some additional windows showing Teletext pages. It is obvious that applications such as PiP replay, split screen, aspect-ratio conversion, etc. are relative simple applications, which are a subset of the aforementioned application example. Furthermore, the system is able to generate a large range of resolutions and frame rates. It may even provide 50-100 Hz conversion making advanced use of the memory.

6. EXPERIMENTAL CHIP DESIGN

An experimental chip has been developed of which the design has been finished recently. Some key figures of the chip are shown in Table 4. The chip runs on 64 MHz, enabling sufficient parallelism for various coprocessors. For having sufficient bandwidth, the clock frequency of the memory was raised to 96 MHz.

A new aspect of the chip architecture is the notion of video *tasks*, rather than using video streams. Because of the variable size of video windows on the display and because the quality corresponding to each window is also variable, the assignment of a coprocessor to process a stream is not of constant length, but variable in time. Since pictures can be as small as the conventional Picture in Picture (PiP), the task can be small as well, leading to flexibility in assigning several small tasks, instead of single large tasks. The variable length of tasks asks for communication rules to prevent dead-lock in the system. Video is communicated as data tokens [5], where subsequent processing is only allowed under certain restrictions.

Table 4. *Summary specification of the experimental IC.*

Coprocessor Array IC for TV applications	
# of coprocessors	6
Input / Output Proc.	3 Inputs, 2 outputs
Video resolution	variable, up 848 x 600
Clock frequency	64 MHz
Memory external	96 MHz SDRAM
Memory bandwidth	max. 384 MByte/s

7. CONCLUSIONS

A new flexible architecture for high-end television is presented, featuring multiple video window display with high flexibility and programmability. The architecture is characterized by an ensemble of coprocessors, optimized for individual TV functions, which communicate in parallel via a full switching network. This network enables programmable signal flow-graphs. The architecture is not stream- but task-oriented, enabling flexibility for various picture formats and leaving computational power for remaining tasks when primary tasks have been carried out. The concept is more cost-effective than full programmable solutions, because application-specific coprocessors are used. Furthermore, the use of a shared background memory enhances this cost-effectiveness. A major benefit of the proposal is to create SW programmability for video signal manipulation at functional levels and to have cost-effective hardware support for pixel-based video processing.

8. REFERENCES

[1] H. Desor, “Single-Chip Video Processing System”, IEEE Trans. Consum. Electron., Aug. 1993.

[2] H. Miyaguchi *et.al*., “Digital TV with serial Video Processor”, IEEE Trans. Consum. Electron., Vol. 36, pp. 318-326, August 1990.

[3] J. Leijten, “Real-Time Constrained Reconfigurable Communication between Embedded Processors”, Eindhoven, Eindhoven Univerisity of Technology, Ph.D. thesis, November 1998.

[4] E.G.T. Jaspers, P.H.N. de With and J.G.W.M. Janssen, "A flexible heterogeneous Video processor system for Television applications", IEEE Trans. Consum. Electron., submitted for 1999.

[5] 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.