A VIDEO DISPLAY PROCESSING PLATFORM FOR FUTURE TV CONCEPTS

Peter H.N. de With [1], *Egbert G.T. Jaspers* [2], *Jef L. van Meerbergen* [2,3]
Adwin H. Timmer [2], *Marino T.J. Strik* [2]

[1]University of Mannheim, Fac. Computer Engineering, 68131 Mannheim, Germany.
[2] Philips Research Labs., Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands.
[3] Eindhoven University of Technology, Den Dolech 2, 5600 MB Eindhoven (NL).

Abstract— **This paper presents a generic multi-processor architecture for video processing[1], featuring an array of programmable application-specific coprocessors and a programmable switch-bar communication network. Key advantages of the platform are that the order of functions can be changed and that video tasks can be programmed at system and functional level using tasks of any length, by applying dynamic data-flow. This modular concept is cost-effective and suitable for a large range of applications and features in the video domain. The system can be well applied in combination with micro-controllers or media processors and it is used for implementing an experimental TV platform.**

Keywords—**Video processing architecture, HW/SW co-design, programmable hardware, dynamic data flow, multi-window, multi-processor, parallel processing, switch matrix, unified memory.**

I. INTRODUCTION

THE VLSI complexity and the corresponding software for implementing multimedia architectures including advanced video processing require an increasingly large design effort. Chips for consumer-electronics products are currently being designed containing up to 10 Million gates, having a computing power in the order of 10 GOPS (Giga Operations Per Second) [1]. Furthermore, the corresponding embedded software approaches 1 MByte for TVs. The architecture of conventional TV sets has been introduced more than half a century ago. Since that time, the functionality that was offered to the customer has been increased significantly, without reconsidering the architecture of the system. Feature extension boxes have been added to upgrade the conventional system to top-high-end TV sets currently available in the consumer market. Fig. 1 shows an example of a conventional top-high-end TV architecture. When standardization of the electronic television was carried out [2], the architecture already showed the traditional standard video path. It consisted of the demodulator, decoder and the picture control block with a separate audio path. Since then, an additional Teletext path was introduced, 50-to-100 Hz conversion, PALplus decoding, an additional Picture in Picture (PiP), dual-screen, new sound standards and more recently, digital transmission standards based on MPEG. All these new features are an extension of the conventional implementation, resulting in a suboptimal structure. Because each function uses its own memory device and has a dedicated communication path within the implementation, the solution becomes very cost inefficient and rigid. These restrictions are in conflict with occurring trends in TV applications. Requirements for the design of future TV systems should be as follows.

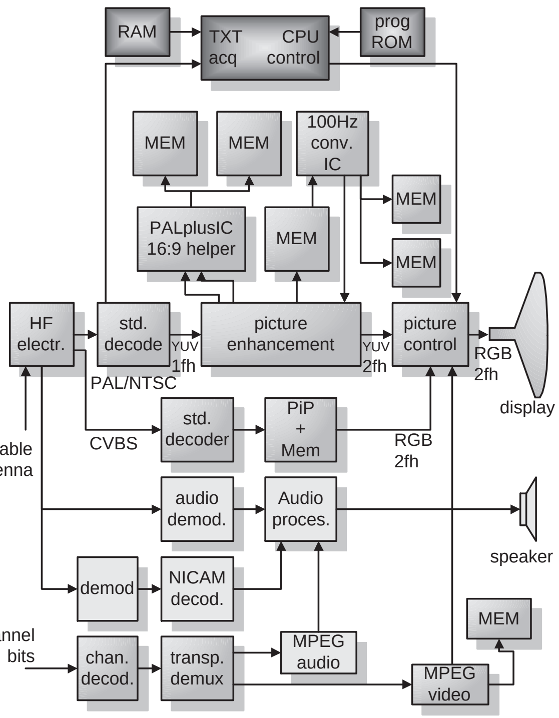

Fig. 1. The HW architecture of a conventional high-end TV set.

- With the introduction of digital TV and the increasing use of digital image enhancement, 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

[1]A paper is being presented at ICCE99 by E. Jaspers *et al.*, entitled ”Chip Set for Video Display of Multimedia Information”, describing a first realization of the architecture. One of the ICs, commercially available as SAA5810, is explained in more detail in a paper by O. Steinfatt *et al.* entitled ”TCP: A Next Generation for TV Control Processing” (Digest Techn. Papers IEEE ICCE99, Publ. No. 99CH36277, June 1999).

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-efficient implementation, since it is a highly competitive market.

Some attempt in the past were made to deal with these requirements. Full-programmable multimedia solutions like the Multimedia Video Processor (MVP) [3] can be very powerful, but are rather expensive for consumer products because they are more general than the TV application domain desires. Full programmable solutions that are more application specific like the Cheops processor module [4] and the so-called Video Signal Processor [5], contain video-specific operations but lack processing power. Both solutions are based on fine-grain functional units in a reconfigurable communication network. Since the functional resources are relatively limited, the amount of parallelism is not high enough. A recent trend is to use Very Long Instruction Word (VLIW) processors to boost processing power [6] [7]. However, this is still not sufficient to cover all processing requirements. Simply increasing the number of functional units would increase the present scheduling problem for such solutions.

More powerful implementations are possible by adapting the system more closely to the target application domain. As a result, the processing performance and efficiency can be increased by making use of the function-level parallelism that is expressed in a designers signal flow-graph (SFG) of the application. The grain of the operations within a functional unit is increased to the level of complete TV functions like a scaler, noise reduction, sharpness enhancement, field rate conversion, etc.

II. Problem statement

A. Computational effort

Let us now analyze why the aforementioned processor systems were not directly adopted in current TV systems, although it seems that the required applications could be implemented on a fully programmable multimedia processor. One of the questions to be answered 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 I shows the intrinsic processing power of several TV functions at standard definition (SD) resolution. With respect to operation counting, additions, multiplications, sample read and writes, etc., are considered as single operations. The fourth 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 50-100 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 processor cost-effectively. Consequently, the use of application-specific coprocessors to increase parallelism and local computing power, in combination with general-purpose processing is unavoidable to keep system costs low.

TABLE I
Expenses of various TV functions.

Function	**Operations per Second**	**Bandwidth MByte/sec.**	**Memory/ Cache**
H zoom / compress	400 MOPS	38–64	samples
V zoom / compress	400 MOPS	38–96	lines
Filters, Comb filters	200 MOPS	64–128	samples-field
Advanced peaking	650 MOPS	32	lines
Color transient improvement	300 MOPS	48	samples
Dynamic noise reduction	500 MOPS	64–128	field
MC-100 Hz	2–4 GOPS	192-256	2–3 fields
Color space	150 MOPS	80	None
Teletext conversion	10 MOPS	48	> field
Adaptive luminance	60 MOPS	32	1 KByte

B. Memory considerations

Because memory functions determine a significant part of the system costs, the access and the amount of memory are important optimization parameters in the system design. For the same reason, the distributed memory as shown in Fig. 1 is not acceptable in future products. To create a cost-efficient solution, one uniform background memory is used, based on a standard off-the-shelf RAM. To avoid excessive memory access, the processors have small local cache memories, e.g. a vertical-filter coprocessor may have line memories locally. The caching of field memories for e.g. MC-100 Hz conversion, dynamic noise reduction or comb filtering is too expensive for on-chip integration as embedded memory (1 field = 4 Mbit). Instead, these functions are integrated in the large unified memory in the background. However, the consumption of already scarce bandwidth to this memory should be monitored. Furthermore, the architecture should enable direct communication between processors without necessary access to external memory. This approach limits the memory access to functional usage. Generally, only systems that use large packets to communicate data (e.g. video field/frames) for high-throughput communication between multiple processors, need extreme fast communication networks with a high bandwidth and a considerable amount of additional memory. Examples of such systems are given in [6] and [8].

Although the memory requirements of the desired system are optimized, typical memory sizes would still be 4–16 M-

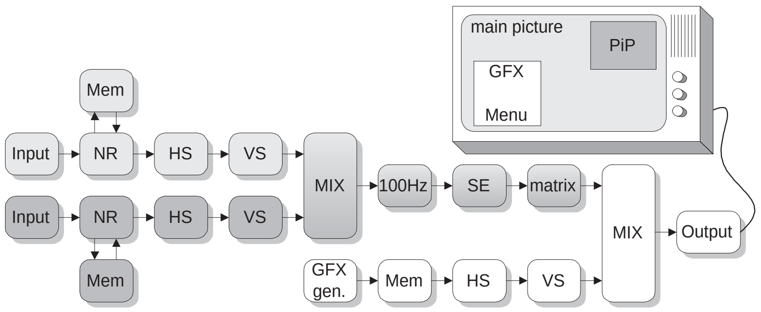

Fig. 2. An example of a signal flow-graph (SFG).

Byte with a bandwidth in the order of 350 MByte/s. For such memory devices, the synchronous DRAM (SDRAM) represent the mainstream market, offering the lowest price per Byte. However, for high-bandwidth applications, the newer but more expensive double-data-rate DRAM (DDR DRAM) and Direct Rambus (Direct RDRAM) devices are an alternative. However, for an efficient data transfer, these devices should be accessed for bursts of data instead of single data words, due to the overhead in the addressing. Since multimedia processing stores mostly neighbouring data of the actual sample in memory (spatial locality), multimedia memory caching can be used to reduce the address overhead. When performing signal processing on video samples, it is desired to have a high peak bandwidth with low latency. Therefore, the low-density static RAM (SRAM) is used at the on-chip cache level.

III. Analysis of TV applications

In this section, the TV applications are further analyzed with respect to the timing requirements in order to motivate the design of the new chip architecture. Prior to discussing the details, the concepts of graphs and tasks are introduced, as they will be applied extensively in the system analysis.

A *task* is a basic TV function, a set of closely coupled calculations, which are executed on a stream of data samples (a video signal). The set of TV functions is well defined for the high-end TV system. Tasks are *weakly* programmable, that is, they can be set to different video quality levels with different resource requirements, but the nature of the function itself cannot be modified. For example, a filter can be programmed to a smaller length resulting in a lower quality, thereby reducing local memory usage. This memory can be re-used for other filter tasks.

A *graph* consists of a set of TV functions (and thus tasks) which should be executed in parallel. It represents an application in the TV domain, e.g. a main video signal with a PiP in the upper corner.

The first step in coming to a new architecture is to classify the different tasks into three groups: *control*, *soft real-time* and *hard real-time* tasks. Examples of control tasks (CT) are user interactions such as channel switching and user menu generation, a modem function, etc. Control tasks are related to the interaction of the system with the environment. The second group contains soft real-time tasks (SRT). Key aspect is that such tasks have a ”soft” real-time deadline, meaning that a deadline can be missed and is not critical for the system performance. An example is the decoding of a Teletext page.

Hard real-time tasks (HRT) should not miss their deadline, since this would directly lead to system malfunctioning. Typical hard real-time tasks are most video processing operations, since output pictures have to be displayed at predetermined time intervals (field rate). Examples are horizontal and vertical sample-rate conversion, sharpness enhancement, noise reduction, etc.

Hard and soft real-time tasks can be represented in a signal flow-graph (SFG) as shown in Fig. 2, which portrays two input video streams (HRT) and one Teletext page (SRT). The results are displayed in three different video windows on the screen. The user can manipulate several features (size, position, shape) of the smaller windows.

The hard and soft real-time tasks that require a large computational effort are mapped onto more dedicated processor hardware to increase the amount of parallelism. This approach was adopted to come to a cost-effective system design, enabling a large programmable set of applications with a single chip. In [9], it was discussed that a fully programmable architecture for a broad range of TV applications is far too expensive for consumer television. Although the aforementioned application-specific (co)processors are weakly programmable and are able to process various tasks, the type of task is fixed, e.g. a sampling-rate converter can perform aspect-ratio conversion, scaling, geometric corrections or data compression, but cannot do noise reduction or sharpness enhancement. The system requires an architecture which provides multi-tasking capabilities, because several different tasks have to be processed simultaneously by the same coprocessor.

Let us now summarize the results of the analysis of the set of TV functions and the computational estimations from the previous section and find implications for the architectural design.

- The required computing power is in the order of 10 GOPS (see Table I).
- If all the communication between tasks in a typical flow-graph are accumulated, several GByte/sec required bandwidth is obtained. Each full-color video channel requires 32 MByte/s bandwidth (at 50-Hz field rate, see Table I).
- The constraints for hard real-time streams must be met under all conditions. This requires worst-case design of the architecture for this part of the system.
- One of the disadvantages of current rigid TV architectures is that signal-processing functions cannot be used in different orders (see Section 1). Such a feature would enable an optimal TV quality setting under all conditions within the limits of available resources. The new architecture should offer this flexibility, i.e. the feature to construct different flow-graphs with the same set of TV functions. Analysis has shown that about 100 different graphs exist for common high-end TV applications.
- The resulting quality of the performed video tasks should be scalable to adapt to the amount of available hardware resources (computing power and bandwidth). The quality depends therefore on the chosen flow-graph. For example, if only one video stream is being processed, all video functions can be switched to the highest quality level, requiring e.g. the best filters or most advanced processing (computationally intensive for each task). If two video streams have to be processed, the same video task can appear twice in the flow-graph, so that each TV function is executed at medium quality. This might be acceptable, e.g. a PiP application does not require the same video quality as an application with one main video plane. More precisely, the video streams are processed at the clock rate (64 MHz), but can have four different pixel rates: 16, 32, 48 and 64 MHz. Each coprocessor can process a number of streams in parallel, given by the following expression:

$$\frac{n_{16}}{4} + \frac{n_{32}}{2} + \frac{n_{48}}{3} + n_{64} \leq 1 \tag{1}$$

  where n_{16} represents the number of streams with 16-MHz pixel rate, n_{32} stands for the number of streams with 32-MHz pixel rate, and so on.
- The system should be capable of processing a minimum of two real-time input video streams, because our aim is a multi-window TV system. The clocks of the streams are not related (asynchronous), as they originate from different sources.

Up to this point, signal processing and its requirements were discussed primarily. Let us now concentrate on the memory which is associated with the signal processing. As was discussed in subsection II.B, the objective is to use one single unified memory. Summarizing the most important reasons for this:

- *memory re-use*: if functions are switched off, memory can be assigned to other tasks;
- *costs*: off-the-shelf large-scale memory components give low system costs.

An experimental system implementation is based on a memory bank of 96-MHz Synchronous DRAMs. For 32-bit words, this gives a theoretical bandwidth up to 384 MByte/s, which is lower than the maximum requirements mentioned in Section II. Therefore we make a distinction in the communication requirements, similarly to the distinction in tasks. Some communication channels do not need access to external memory and can be kept on chip. For other functions, especially those requiring complete field memories, access to external memory is inherent to the function, such as temporal noise reduction. Another example is the synchronization of two independent video streams, requiring a video frame memory (the mix task in Fig. 2). Since memory bandwidth is a scarce parameter, external memory accesses are indicated in the signal flow-graph: they are modeled as extra inputs and outputs to the graph. Furthermore, the graphs are cut at points where frame synchronization takes place. This leads to the graphs as shown in Fig. 3.

By cutting the flow-graph into pieces, we have implicitly introduced an additional concept, called subgraphs. A *subgraph* is a set of closely connected tasks, where all communication between the tasks belonging to the same subgraph takes place via on-chip communication. Inputs and outputs of subgraphs are either regular input/output streams or accesses to external memory. Communication between two different subgraphs always takes place via external memory.

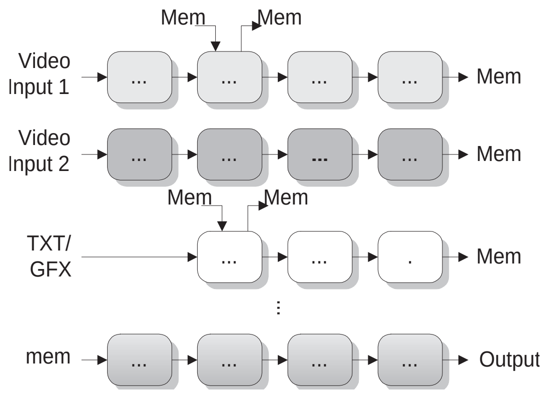

Fig. 3. Examples of subgraphs, corresponding to the flow-graph in Fig. 2.

IV. Architecture design

The application analysis of Section III and the computational requirements of Section II are starting points for designing the architecture. We have discussed sets of tasks being combined into graphs, and the associated memory communication. Consequently, the architecture contains principal modules and their components, and as such, it has a hierarchical structure. Let us therefore start at the top of the hierarchy.

A. Top-level architecture

Since the properties of control and the signal processing are totally different, we will define three different subsystems: a control subsystem, a signal-processing subsystem and a memory subsystem. The control subsystem is responsible for the execution of all control-oriented tasks and some of the soft real-time tasks. The signal-processing subsystem executes all hard real-time tasks and some of the soft real-time tasks. The memory subsystem performs the memory functions for both the system control and signal processing.

The nature of the processing is totally different. The control subsystem is based on *events*. It has to deal with unpredictable events coming from the environment. Therefore, it sends *random* requests to the memory subsystem. The signal-processing subsystem is characterized by the periodicity of the operations. It sends *periodic* requests to the memory subsystem. The top-level architecture is visualized in Fig. 4.

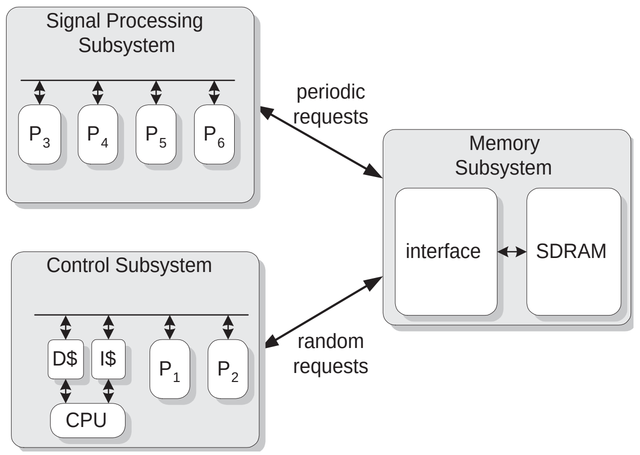

Fig. 4. The top-level architecture.

The principal design step at the top-level concentrates on the arbitration between the two types of requests. The difficulty is that different aspects have to be optimized. In the case of random requests, the *latency* should be minimized. In the case of periodic requests, the *throughput* of the system has to be guaranteed, because all hard real-time tasks are mapped onto the signal-processing subsystem which is periodic. Thus, in the case of periodic requests we want to design for the throughput that should be obtained. The latency is less relevant, since it can be hidden if sufficient buffering is provided.

Summarizing, an arbitration scheme is needed that ensures the throughput for the periodic requests, while minimizing the latency for the random requests. For our system, we used an arbitration scheme from [10], that provides a low latency and high-priority access for the microcontroller and ensures the required bandwidth for the video signal processors.

B. Signal-processing subsystem

The tasks to be performed for a complete SFG in a TV set can be classified globally into two groups: tasks requiring a similar type of signal processing and tasks which are both different from functional and signal-processing point of view. Examples of the first group are horizontal compression for PiP and aspect-ratio conversion. The required type of signal processing is sample-rate conversion in both cases. Examples of the second group are noise reduction and sharpness enhancement which are essentially different in most algorithmic aspects. Using the previous distinction, a number of choices can be made for the new architecture.

- After functional decomposition a task is mapped on a separate processor if they are different and is mapped on the same processor otherwise.
- Since the tasks are known in advance at design time, each processor can be optimized for a particular job.
- To allow the mapping of different graphs, a reconfigurable communication network is added. This means that signal-processing functions can be used in various orders and can be programmed.

The result of the architectural choices is that a heterogeneous multiprocessor architecture is obtained, which is capable of true task-level parallelism. The architecture is shown in Fig. 5. This will now be discussed in more detail.

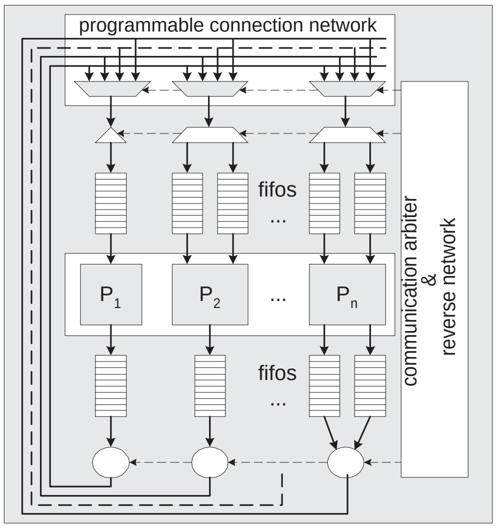

Fig. 5. The architecture of the signal processing subsystem.

First, it can be noticed that all processors are surrounded by FIFO buffers at their inputs and outputs. The aim is to separate signal processing from communication, so that the activity of the processors is de-coupled from the communication network. Second, the choice for FIFO buffers is motivated. They have been adopted because the arrows in the SFGs (called *edges*) represent streaming (video) signals, i.e. measurable physical quantities sampled at discrete points in time and binary encoded. The only identification of the different samples in the stream is given by the order of the samples. Samples are produced only once and cannot be lost on the communication channels. For streams with the aforementioned features, separation of communication and processing can be well be performed with FIFOs. Third, the architecture is based on a dynamic data-flow model (DDF), instead of synchronous data flow. The reasons are as follows.

- Analyzing the tasks in an SFG, some video signals cover the full display, while others cover a only a part of the screen, such as a PiP. In many stages, the amount of computations is proportional to the amount of samples processed. Therefore, a surplus of compute power can be used to make a processor available for alternative tasks. Such task switching is more difficult with static synchronous systems.
- The field blanking, i.e. the non-active vertical part of a video signal, can be used for soft real-time tasks. The detection whether a stream is in the blanking or not is a run-time decision, since input streams are asynchronous with respect to each other. This re-use for soft real-time tasks is easily implemented with a dynamic stream based model.
- A stream-based computing model is often simpler to implement in comparison with a static synchronous system, because it can perform run-time scheduling based on local availability of data.
- The concept is better scalable with respect to the addition or removal of processors. It is expected that processors will become increasingly dynamic. A good example is a variable-length decoder, which produces and consumes a data-dependent number of tokens.

Let us now discuss briefly how the signals will flow in the DDF model. The communication behaviour is such that the signal channels can be blocked locally (blocking semantic), according to the DDF model. A local processor is stopped when the output FIFOs are full or the input FIFOs are empty. The decision is performed locally in the shells surrounding the processors. Furthermore, the output FIFOs must be blocked when the input FIFOs are full. This is done by implementing a *reverse* network. So far, resource sharing, that is, the re-use of a processor for another task was only mentioned but not yet taken into account. For the model, a one-to-one correspondence between tasks in the graph and processors in the architecture has been assumed. In the following subsection, resource sharing will be discussed.

Resource sharing is desirable in order to obtain a flexible and a scalable architecture. Three different forms of resource sharing are distinguished. First, different tasks can be executed on the same processor. This leads to the processor model discussed below. Second, FIFOs can be shared, but this is avoided, because deadlock may occur in the multiprocessor system. Third, resources in the communication network can be shared. This option is also discussed below.

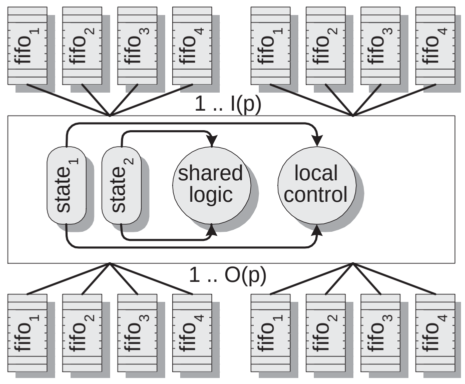

Fig. 6. The model of a signal processor.

B.1 Processor model

The same function (e.g. horizontal sampling-rate conversion, HS) can appear more than once in an SFG. For cost reasons, these conversions will all be executed on the same HSRC processor. However, there are some limitations that will be explained now.

The processor model portrayed by Fig. 6 has two input ports *I(p)* and two output *O(p)* ports. This could for example be a temporal noise reduction coprocessor that needs an input video signal and the filtered result from the temporal loop as inputs and has a video output and a backward channel to memory as outputs. Each input and output port is connected to four FIFOs, labeled 1-4. When a task is started, one or more FIFOs are assigned to this task at each input and output port. The number of assigned FIFOs to a task is dependent on the required amount of bandwidth and is explained in more detail in the next subsection. As long as the task is active, the assigned FIFOs are only processing data corresponding to this task. Associated with each task, a local memory state exists, called *state space*. Thus, according to the depicted model, two different state spaces can emerge, where each state space is assigned to a particular function. As a result, this model can start up to maximal two tasks in parallel (the experimental chip discussed in [9] can perform one to three parallel tasks, dependent on the coprocessor).

B.2 Communication network

The task of the communication network is to provide sufficient bandwidth for the data streams between the output and the input FIFOs. For every connection in the SFG, a path is

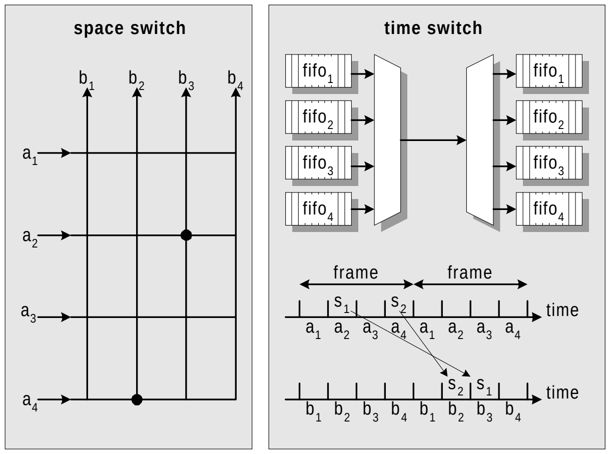

Fig. 7. A space switch (left) and a time switch (right).

created in the programmable connection network in Fig. 5 for transport of the data. The path is created using circuit switching.

The network is a so-called TST network with space and time switches. The reason to build such a network is to guarantee non-blocking connections between output FIFOs of processors and input FIFOs of succeeding processors, with a predetermined amount of bandwidth for each connection. Fig. 7 shows a space switch at the left-hand side making a connection between a_2 and b_3 and another connection between a_4 and b_2. At the right-hand side of Fig. 7, the same connections are shown using a time (T-)switch. At the input, a multiplexer is added and a demultiplexer at the output. Using four time slots, the total bandwidth equals the sum of bandwidths of the individual channels. This leads to the TST network as shown in Fig. 8. As an example, two paths through the network are indicated. Each path is programmed by putting the correct code for the three different parts of the network. A table with four different time slots is used. For example, the x connection is programmed in phase two via the correct code at the positions labeled with an x. This way, bandwidth is allocated corresponding to one phase. The connection labeled y has twice the bandwidth of one channel, because it is programmed during two phases. The programming flexibility ensures that sufficient bandwidth can be provided for particular stages or signals in the SFG.

B.3 Interaction between controller and processor

Also interactive communication is necessary to adapt the TV function to the content of the video signal, e.g. the noise-reduction processor measures a significant noise increase, communicates this property to the CPU, which is subsequently programmed by the CPU to perform more noise reduction. For this type of event-driven communication, one interrupt (irq) line to the CPU is used. The signal-processing subsystem contains a large range of interrupts to

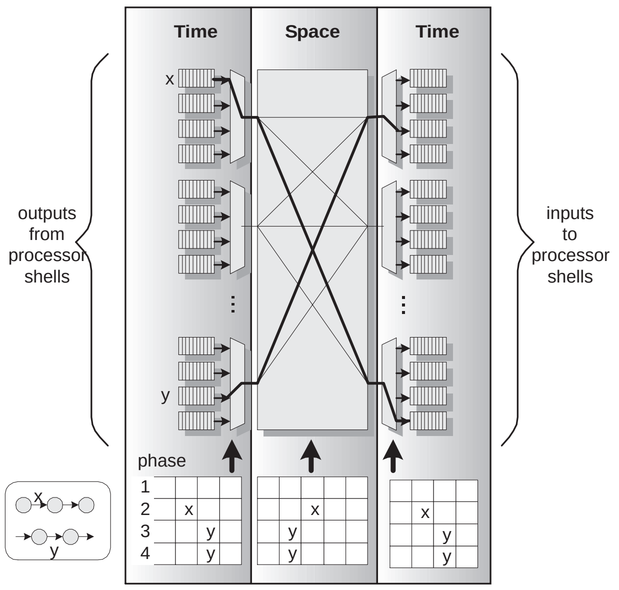

Fig. 8. A Time-Space-Time (TST) network, ensuring a guaranteed bandwidth for hard real-time tasks.

cover all possible events. Each interrupt is represented by one bit in a memory-mapped interrupt register. The CPU receives an interrupt if one of these bits indicates an ”irq” by means of a hardware OR function. Subsequently, the CPU may read the interrupt registers via the interrupt service routine in order to identify which event has occurred. The ”irq” is then acknowledged by resetting the corresponding irq-bit in the memory. This general communication protocol is visualized in Fig. 9. The interrupt mechanism is also used when processing generates erroneous behaviour or when signals are not in accordance with specified video formats. Since all interrupts generated by the signal-processing subsystem operate at video field rate and can be disabled individually, the interrupt rate to the microcontroller system can be kept sufficiently low. This optimization is advantageous, because interrupts increase the communication overhead (context switching). Adaptation of the processing is sufficient at field frequency, since it provides sufficient response-time of the system. This also gives ample time for the microcontroller system to perform other tasks, such as e.g. modem communication.

V. Memory

A. Subsystem communication

As was mentioned in section II, a cost-effective solution requires one uniform background memory that is based on a standard off-the-shelf RAM. The RAM can be accessed by both the CPU and the coprocessors. An attractive property of the system architecture is that the control-subsystem and the signal-processing subsystem can be used as stand-alone devices, because they provide there own control for operation. Both systems are favorably combined into a powerful full-featured system by a connection via the memory bus, as

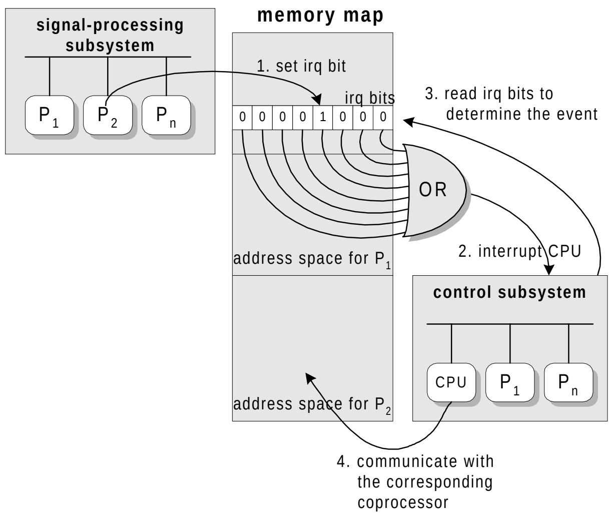

Fig. 9. Communication protocol for interactive processing.

depicted in Fig. 10. This enables communication between the CPU and the signal-processing subsystem by means of memory-mapped I/O. One unified memory map is applied in which all parameter and operation-mode registers of the coprocessors are located. The CPU can access all the parameter registers to program the coprocessor settings and derive information from the processors and/or the signals being processed.

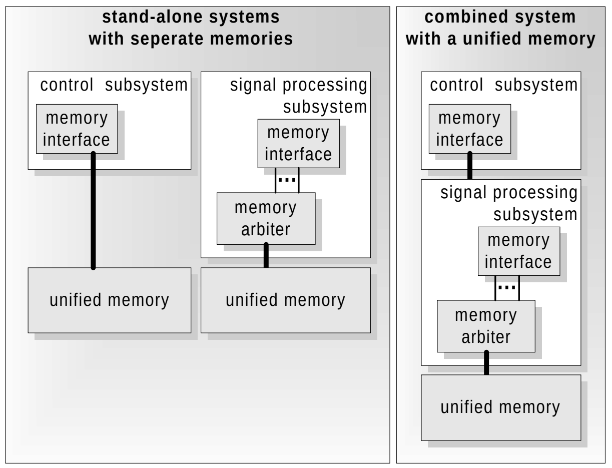

Fig. 10. Stand-alone systems and a combined system with unified memory.

Since in the architecture proposal a single memory is used, bandwidth limitations are likely to appear. Some measures have been taken to relax this so-called von-Neumann bottleneck [11]. As an example, the frequency of the memory bus is twice the frequency of the CPU, whereas the bus width is equal to the word width of the CPU (32 bits). Furthermore, coprocessors can be programmed such that memory access can be reduced at the cost of some quality. These aspects are discussed in the following subsection.

B. Memory resources versus quality

New applications for the system are mainly limited by the computing power of the resources, the bandwidth of the coprocessors, and the bandwidth of on-chip and external memory. Consequently, special attention should be paid to the architectural design of the communication network. As was mentioned in Section II.B, communication for video signals via the memory should be limited to large memory functions only, e.g. the field delays for temporal noise reduction and field-rate conversion. Additional key issues in the design of the communication architecture that have a direct impact on the memory resources are listed below.

Local FIFO buffers, which are relatively small, located at the input and output stage of all coprocessors (see Fig. 5), enable date exchange between coprocessors without access to the external background memory. This significantly reduces the bandwidth requirements of the external memory.

Mixing or juggling of video streams is designed such that it requires a minimum amount of memory bandwidth. In the background memory, two field blocks (for interlaced video) are allocated to construct the composed video frame. These memory blocks are filled with the odd and even fields of the picture, except for the pixel positions where another overlapping video window will be positioned. This unused memory area in the memory is used by a second video path to write the overlapping video window. Consequently, the total amount of data stored is equal to the data of one complete picture instead of the sum of individual pictures. Similarly, the total required bandwidth approximately equals the bandwidth for writing one complete stream instead of the sum of the bandwidths of the individual video streams.

Bandwidth equalization is used to decrease the required peak bandwidth. Equalization in memory read/write tasks is obtained by spreading the data 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, leading to a similar reduction of the peak bandwidth.

An additional system aspect besides the aforementioned key issues, is the desire to realize *scalable* processing power, in order to enable the exchange of bandwidth with picture quality. Some examples are given below.

Vertical scaling at high quality requires that the interlaced input video is converted to progressive video prior to sampling rate conversion. As a result, a necessary background memory function is to write interlaced video fields and to read the odd and even video lines progressively. Subsequently, de-interlacing should be applied prior to the scaling. After scaling the progressive frames, they are interlaced again, by removing the odd or even lines before the final result is communicated to the rest of the system. The memory requirements for such a vertical scaling task are significant. At least one field memory is necessary and a memory bandwidth of three times the bandwidth of one single interlaced video stream is required to write interlaced and to read progressive video. To save some memory resources, it is possible to scale the interlaced fields, thus to perform intra-field processing instead of inter-field processing. This would decrease the picture quality at the gain of memory resources.

Graphics generation in the background memory requires a field or frame memory, 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 loss in 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. Moreover, it is also expensive in terms of memory size. For 50-60 Hz interlaced video, a field memory is most attractive, whereas for field rates higher than 70 Hz, a frame memory could be used for high-resolution graphics.

VI. Conclusions & Applications

This paper presents a video processing architecture with high parallelism, based on a programmable array of coprocessors and a communication network. Function-specific hardware coprocessors perform pixel-based processing and are programmable at functional and system level. A switch matrix enables parallel processing of tasks and enables high communication bandwidth. Since the matrix is programmable, the order of signal processing tasks can be modified (programmable signal flow-graph) to realize new TV applications. Individual coprocessor processing power and performance can be adapted and optimized for each TV application flow-graph. The architecture uses an external background memory which combines all large memory functions. This memory is shared with a microcontroller. Each coprocessor has some local video memory, in correspondence with the function carried out, to avoid severe memory-bandwidth problems.

A novelty of the architecture is the use of dynamic dataflow for video processing, in combination with application-specific processors. In the concept, processors perform independent processing and data streams are exchanged via small local buffers. These buffers can start and stop the processors dependent on whether a buffer is full or empty while additional communication rules avoid deadlock problems. The advantage of this system is that video processing functions can be handled as tasks of any length. Thus video frames of various sizes can be applied in the system without low-level reprogramming of the hardware. If a task is terminated, processor hardware becomes available for alternative tasks.

The proposed system can be applied in a large set of application domains containing component video processing. For television, the architecture has been implemented in two experimental chips, a microcontroller and a coprocessor array. The microcontroller performs set control, Teletext and graphics, modem and low-speed data communication. The coprocessor array contains functional units for horizontal and vertical scaling, sharpness enhancement, motion-adaptive temporal noise reduction, graphics blending, mixing of video streams and 100-Hz up-conversion. Due to the flexibility of the coprocessors and the programmability of the signal flow-graph, the application range not only covers the intrinsic integrated functions [12], but also aspect-ratio conversions, PiP, a mosaic screens, pixel-based 2D graphics for PC-like applications, etc. In fact, the chip set enables all these features in a multi-window TV environment with high quality. As an example, Fig. 11 shows the resulting displayed picture of the the TV application visualized by the signal flow-graph in Fig. 2.

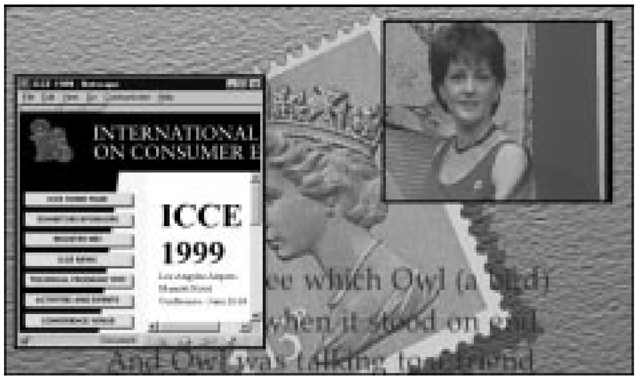

Fig. 11. Example of a multi-window application.

The modularity of the architecture concept allows that other advanced functions can be added easily in the form of new application-specific processors, e.g. MPEG decoding and 3-D graphics rendering. For consumer systems, it is possible that the architecture can be used for a number of years to come, due to its cost-efficiency. As technology proceeds and the demand for flexibility increases (e.g. MPEG-4, MPEG-7), the low-complexity and irregular processing functions will be implemented in software on advanced microcontrollers or media processors, so that corresponding coprocessors will disappear. Combining the proposed platform with such generic processors is attractive, because a hardware-software trade-off can be made. At the same time, it is expected that new computationally expensive functions, requiring specific hardware support, can be introduced.

REFERENCES

[1] B. de Loore *et al.*, “A video signal processor for motion-compensated field-rate upconversion in consumer electronics,” in *Proc. Int. Solid-State Conf.*, Febr. 1996, pp. 248–249.

[2] F.J. Bingley, “A half century of television reception,” in *Proc. of the IRE*, May 1992, vol. 50, pp. 799–805.

[3] R.J. Gove, “The multimedia video processor (MVP): an architecture for advanced dsp applications,” in *Proc. of the 5th int. conf. on Signal Processing Appl and Technology*, Oct. 1994, vol. 1, pp. 854–859.

[4] V.M. Bove and J.A. Watlington, “Cheops: A reconfigurable data-flow system for video processing,” *IEEE trans. on Circuits and Systems for Video Technology*, vol. 5, pp. 140–149, April 1995.

[5] H.J.M. Veendrick *et al.*, “A 1.5 GIPS signal processor (VSP),” in *Proc. of IEEE Custom Integrated Circuits Conf.*, May 1994, pp. 95–98.

[6] S. Rathnam and G. Slavenburg, “Processing the new world of interactive media,” *IEEE Signal processing Magazine*, vol. 15, no. 2, pp. 108–117, March 1998.

[7] G. Campbell V. Easwar and R. Natarajan, “Multimedia system on a chip for set-top box applications,” in *Int. Conf. Cons. Electr., Digest, no. 99CH36277*, June 1999, pp. 356–357.

[8] S. Purcell, “The impact of Mpact 2,” *IEEE Signal processing Magazine*, vol. 15, no. 2, pp. 102–107, March 1998.

[9] E.G.T. Jaspers and P.H.N. de With, “Chip-set for video display of multimedia information,” *IEEE Trans. Cons. Electr.*, vol. 45, no. 3, Sept. 1999.

[10] K.S. Hosseini and A.D. Bovoloupus, “A simple and efficient bus management scheme that supports continuous streams,” *ACM - Trans on Computer Systems*, vol. 13, pp. 122–140, May 1995.

[11] J. Backus, “Can programming be liberated from the von Neumann style? a functional style and its algebra of programs,” *Communication of the ACM*, vol. 21, no. 8, pp. 613–615, Aug. 1978.

[12] P.H.N. de With E.G.T. Jaspers and J.G.W.M. Janssen, “A flexible heterogeneous video processor system for TV applications,” *IEEE Trans. Cons. Electr.*, vol. 45, no. 1, pp. 1–12, Febr. 1999.

Jef van Meerbergen received the Electrical Engineering Degree and the Ph.D. degree from the Katholieke Universiteit Leuven, Belgium, in 1975 and 1980, respectively. In 1979 he joined the Philips Research Laboratories in Eindhoven, the Netherlands. He was engaged in the design of MOS digital circuits, domain-specific processors and general-purpose digital signal processors. In 1985 he started working on application-driven high-level synthesis. Initially this work was targeted towards audio and telecom DSP applications. Later the application domain shifted towards high-throughput applications (Phideo) which received the best paper award at the 1997 ED&TC conference. His current interests are in system level design methods, heterogeneous multiprocessor systems and reconfigurable architectures. Jef van Meerbergen is a Philips Research Fellow and Associate Editor of ”Design Automation for Embedded Systems”. He is also a part-time professor at the Eindhoven University of Technology.

Peter H.N. de With 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 projects on SDTV and HDTV recording. 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 was working on advanced programmable video processing architectures. 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 Training Centre and for other post-academic courses. 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, member of the program committee of the IEEE CES and board member of the Benelux working group for Information and Communication theory.

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 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.

Adwin H. Timmer was born in Apeldoorn, The Netherlands, in 1966. He received the electrical engineering and Ph.D. degrees from the Eindhoven University of Technology, The Netherlands, in 1990 and 1996, respectively. In 1995, he joined the Philips Research Laboratories, Eindhoven. In 1998, he was a visiting IC architect with the Philips Semiconductors WSG business line, Mountain View, CA. His currents interests are in IC architectures for high-performance signal processing applications, system-level design design methods, hardware/software codesign, and compilation techniques for embedded DSPs.

Marino Strik graduated in electrical engineering from the University of Technology in Eindhoven in 1992. In 1994 he completed the post graduation designers course of the University of Technology in Eindhoven. These two years he worked on high level synthesis and code generation for application domain specific processors. He joined Philips Research in 1995 in the digital VLSI group. Now he is involved in IC design methodology with a focus on digital system on a chip realization.