Synchronization of Base-band Video for Multimedia Systems

E.G.T. Jaspers[a] and P.H.N. de With[b]

[a]Philips Research, Eindhoven, The Netherlands
[b]CMG Eindhoven / Univ. of Tech., Eindhoven, The Netherlands

ABSTRACT

A distributed multimedia computing system contains sources, processing units and presentation devices in which each component may operate autonomously. This independence can be exploited for optimization of individual component performance, using e.g. dedicated clock domains. This paper presents a Video I/O model for such a multimedia system enabling multiple video-signal processing and display (multi-window). This model provides an asynchronous communication interface for independent clock domains with the ability to synchronize a video display to one of the video sources. Robustness of the total system is improved by correcting the irregularities and signal deterioration at the input stage. Hence, the computing is always applied to a stable and regular signal. The communication model has been successfully implemented in I/O modules of an experimental multimedia system.

Keywords: multimedia computing, robust synchronization, data-driven processing, video interfacing, asynchronous communication, video I/O

1. INTRODUCTION

The MPEG standard has been developed to process multimedia information from various systems (TV, computer, telecom) in a more uniform way. In new extensions, such as MPEG-4 and MPEG-7, scenes are composed from a plurality of video objects. These new standards increasingly enable that source processing, transmission coding, composition and presentation are completely independent stages. However, the presentation can still be synchronized with one of the available sources.

The advantage of independent subsystems is readily understood from the broadcasting communication model, where a broadcaster (source) optimizes its performance, whereas the receiver is interested in low power and costs. Another advantage of independence is that it allows performance improvements of subsystems over time (*scalability*), such as more processing power due to a higher local clock frequency. Consequently, the processing of multimedia information is preferably independent of both the source(s) and the presentation device(s), and has its own separate processing and clock domains.

This paper is organized as follows. Section 2 lists the most important requirement for synchronization of video on a display. Section 3 presents a model for asynchronous communication between the different clock domains and discusses two extensions. The first one enables the use of bounded buffers instead of unlimited queues, whereas the second enables data-driven communication under all conditions. In Section 4, all synchronization aspects are described such as input signal corrections, the maintenance of frame synchronization throughout the whole system and the relation between line synchronization and frame synchronization. Features of the implemented experimental system can be found in Section 5.

2. PROBLEM STATEMENT

Base-band video communication between the various I/O and processing domains is originally stream-oriented, but video signals are intrinsically divided into frames and video lines, which can be regarded as data packets for synchronization. Synchronization markers indicate the start of the data packets (H- and V-signals for line and frame synchronization, respectively). From now on, the rising edges of H- and V-signals are referred to as pulses. Let us consider the following requirements for synchronization of video on a display, assuming that the video signals are adapted to the used display.

Email correspondence:
E.G.T. Jaspers: E-mail: egbert.jaspers@philips.com;
P.H.N de With: E-mail: peter.de.with@cmg.nl

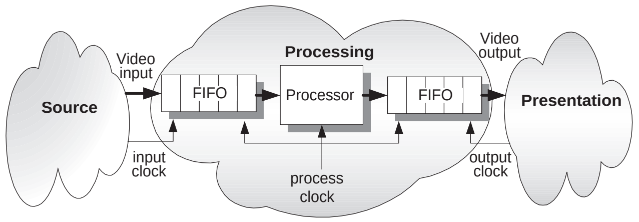


Figure 1. FIFOs for asynchronous communication.

- Frames of a source should be displayed at the correct time, although some jitter in the frame rate is allowed.
- The n-th pixel of each video line should form one straight vertical column on the screen.
- The number of pixels and lines of the video images should match with the display window which depends on the display type (CRT, LCD).

The aforementioned system aspects give direct requirements for the processing subsystem. Firstly, since independent clock domains are used, data exchange can only go via asynchronous communication. Consequently, means have to be provided to synchronization the display wiht one of the sources. Given the above-mentioned requirement, it was chosen to synchronize at video-frame level. The motivation for this is that synchronization at large packet sizes is beneficial for data-driven processing, which is used in our system and described in the next section. The second requirement for the processing subsystem is that the remaining processing bandwidth (if there is any) should become available for other tasks, or the system should switch in an idle mode. This enables scalability of resources and minimizes the power consumption per processing cycle. An additional requirement is that video streaming from input to output is achieved without extra communication to the bandwidth-limited external memory. This requirement is relevant, because most multimedia processing systems communicate input and output data via the external background memory [1].

Our objective in this paper is to define an input and output module for video communication that fits in the multimedia system and complies with the aforementioned requirements.

3. ASYNCHRONOUS COMMUNICATION

A data-driven communication model proves to be suitable for asynchronous communication between separate clock domains (An example is found in [2]). The Kahn process network model [3] is an attractive dynamic model for data-driven communication. The model describes communication between concurrent processes via unbounded queues. In hardware, this queuing scheme can be implemented with FIFO buffers (see Fig. 1). Reading from a queue for retrieving the input of a process can block this process if no data is available. Similarly, writing to a queue at the output stage of the process blocks the process if no space is available. The latter feature is not part of the original Kahn model, but is considered for our implementation because real FIFOs are bounded is size. This modification does not violate the validity of the Kahn model. Blocking read and write operations can be implemented by suspending the internal clock of the video task. For more details see [4].

The above-discussed communication model between processes does not hold when going from a stream-based domain to a data-driven domain and backwards. At these positions in the signal flow-graph, some additional constraints have to be satisfied. When going from a stream-based domain to a data-driven domain, a FIFO should never be full because a full FIFO would disrupt the continuity of the input stream and introduce data losses. Similarly, when going back from the data-driven to the stream-based domain, a FIFO should never be empty. Otherwise, undefined data would appear in the output signal. An extra complication in the data-driven domain is that timing information can be lost. Although the pixel position can be determined due to maintaining the order of data packets, timing information can only be recovered when additional timing information is communicated from the input module to the output module. In the sequel, we define a Video Input module (VIM) and Output Module (VOM) as special cases of the Kahn communication queues, which serve as interfaces between the different domains.

If the picture format is fixed (pixels $\times$ lines) and no pixels are lost during processing, the pixel position can be recovered by simply counting the pixels, provided that the start of a frame is known (or indicated). This fixed-sized picture format is established by explicitly capturing a fixed-sized window of the video signal for each frame under all circumstances. More details will be addressed in Section 4.

Summarizing, a processing component in the distributed system is based on asynchronous communication and has a fixed frame size, while synchronization is established at frame level. On a local time basis, pixels are communicated asynchronously.

Due to the real-time behaviour of practical I/O devices, a second modification of the Kahn communication model for the VIM and the VOM is required. This modification is further supported by the following constraints and system considerations.

- The timing information of a real-time source has to be recovered at the presentation side (for display).
- The sources and presentation devices can be unreliable. For example, the timing between successive frames in a video sequence can vary due to signal deterioration.
- The sources and presentation devices require different communication protocols (e.g. CCIR-656).

The second modification of the Kahn model for the VIM is that a possible overflow at the input FIFO of the VIM (due to too high input rate or a too low processing rate), is simply cut-off so that information loss cannot be avoided. In such a case, the VIM will loose the synchronization and completes the fixed-sized video frame with dummy samples. Similarly, the VOM generates dummy video samples for its display when the display consumes samples too fast (the display expects video samples and no samples are provided).

Finally, a separate synchronization V-pulse from the VIM to the VOM is communicated to recover the exact timing of the video frames. Thus, synchronization between the input and output is provided at frame level, whereas the pixel and line rates are derived from an externally supplied clock (see Fig. 2).

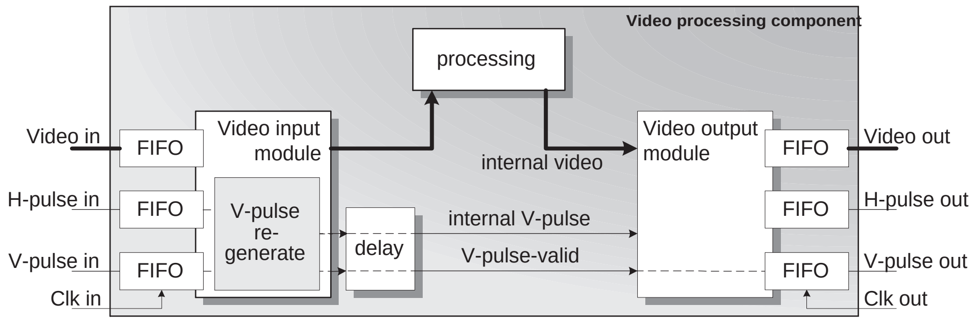


Figure 2. Block diagram of processing component.

4. SYNCHRONIZATION

This section explains how synchronization signals are used to divide a video stream into video frames and video lines. The timing relation between synchronization signals and a video stream needs to be maintained carefully to guarantee a stable video display without introducing artifacts. As we already explained in Section 3, timing information can be easily lost when a data stream crosses different communication domains. We propose an efficient method that maintains the timing information, when data streams cross from a real-time environment to a data-driven environment and backwards.

4.1. Creation of a fixed-sized format

Using a fixed-sized picture format (pixels × lines), is the most straightforward approach to recover a unique pixel position. In this case, counting pixels is sufficient to determine the exact position of a pixel. It is possible to extract a fixed-sized picture format from a video stream, because only part of a video signal is used for visible video display. This part is also called the active video part. Almost 20% of the pixels on a video line and 10% of the video lines in a video frame are not displayed. This is called the blanking time and is used by a CRT display to reposition the electron beam to the start of the next video line. Naturally, only the active part of a video stream is of interest for the processing units. It is even possible to select a smaller part of the active video for processing when the capture window is programmed to a smaller window size, e.g. to zoom out part of the video. Figure 3 shows a part of the active video signal that is captured for processing. The left-hand side of the figure shows the capturing in case of progressive video, whereas the right-hand side shows this process for interlaced video signals. Section 4.3.2 explains

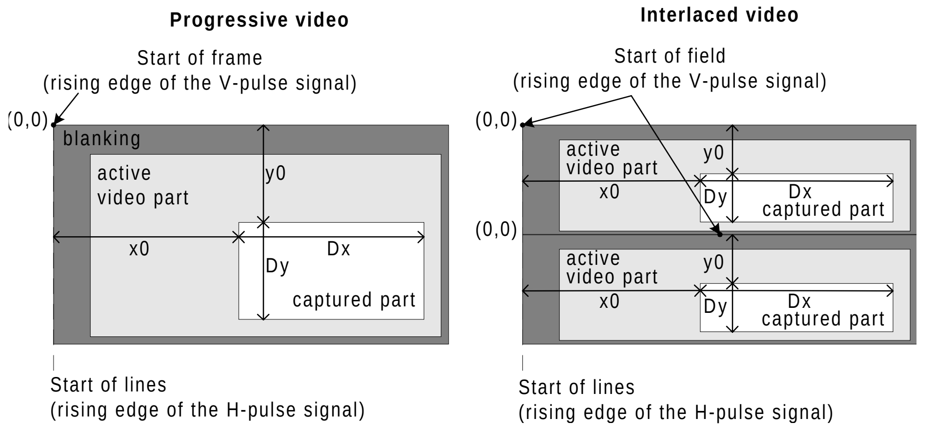


Figure 3. The active and captured part of a video signal.

in more detail the position of the H- and V-pulse for interlaced video. In the remainder of this paper, we assume that the complete active video part is captured. Thus, if in the sequel the active video part is indicated, it represents the captured window part. The blanking is referred to as the remaining part. Therefore, the active part clarified in Figure 3, has the dimension of $Dx \times Dy$. In the horizontal direction, the active part of a video line starts *x0* pixels after an H-pulse. In the vertical direction, the active part starts *y0* lines after a V-pulse. Because a fixed-sized picture format is required, *x0* + *Dx* and *y0* + *Dy* (see figure) are constant. In the case that the full active video is captured, the blanking interval represents a timing interval in which the V-pulse may vary. If a smaller capture window is used, the allowed variation interval increases with the non-captured active video part. In both cases, only the fixed-sized captured part is conveyed through the processing systems. This approach gives a fixed and stable signal under all conditions, thereby adding to the robustness of the total system.

Obviously, the Video Input Module (VIM) has to extract the fixed-sized active part ($Dx \times Dy$) from the video stream in all circumstances. Section 4.2 explains how this is accomplished. The task of the Video Output Module (VOM) is to extend the active part with the blanking part again, such that synchronization between the VIM and the VOM can be maintained. This is elaborated in Section 4.3. Section 4.4 explains how a composition of video coming from multiple sources can be displayed on the same presentation device.

4.2. The Video Input Module

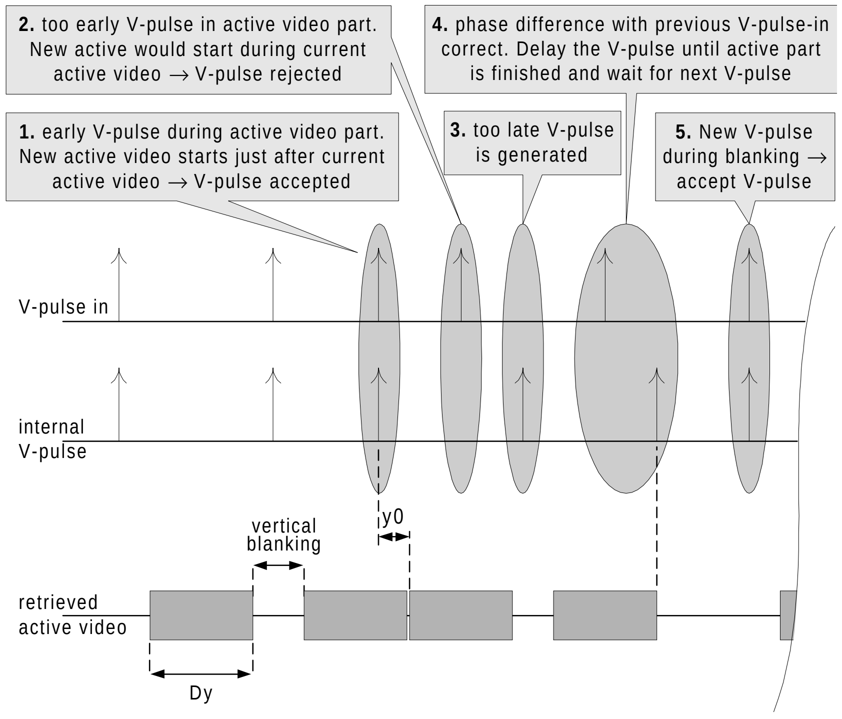
As mentioned before, the Video Input Module has to provide a fixed-sized video signal to the processing units, which can be regarded as standard signal. In practice, non-standard video signals, which have a variable number of pixels per line and variable number of lines per frame, occur regularly. Possible causes for the variability are:

- change of source by the broadcaster or the processing system, e.g. selection of another channel in a TV receiver;
- noise in a distribution channel, e.g. due to bad-weather conditions, H- and V-pulse detection may be distorted;
- sources with a "non-standard" signal, for example a video recorder in trick-play mode.

Once the VIM is programmed for a certain video format, it supplies the processing units with a fixed-sized video signal under all conditions. Consequently, to achieve this goal, the synchronization signals of the input signals may have to be adapted, while maintaining optimal video quality. For adaptation, the VIM uses a H- and V-pulse regenerator. The next subsection addresses this topic.

4.2.1. Frame and line synchronization

We first elaborate on frame synchronization. Line synchronization can be implemented in a similar way and is discussed at the end of this subsection. V-pulses indicate the start of a frame and may vary within the non-captured timing interval. Normally a V-pulse is accepted if it appears during the blanking at a constant rate. However, input V-pulses may occur during the active part or at varying rate. The VIM will try to correct such irregularities. Figure 4 illustrates this process. The following list summarizes all possible situations.



2. too early V-pulse in active video part. New active would start during current active video → V-pulse rejected
4. phase difference with previous V-pulse-in correct. Delay the V-pulse until active part is finished and wait for next V-pulse
1. early V-pulse during active video part. New active video starts just after current active video → V-pulse accepted
3. too late V-pulse is generated
5. New V-pulse during blanking → accept V-pulse
V-pulse in
internal V-pulse
vertical blanking
y0
retrieved active video
Dy

Figure 4. V-pulse regeneration performed by the VIM.

1. **Early pulses**. Pulses that would interrupt an active video part cannot be accepted, because retrieval of the fixed-sized video part has to be finished. The active video part starts *y0* number of lines after a V-pulse. Thus if this V-pulse occurs less than *y0* number of lines earlier than the end of the preceding active video frame, the V-pulse can still be accepted (see case 1 in Figure 4). Ultimately, the active video starts immediately after finishing the preceding active video frame.
2. **Too early pulses**. A V-pulse that occurs even more than *y0* number of lines earlier in the active video part cannot be accepted and is called a too early pulse.
3. **Missing or too late pulses**. When a V-pulse is not received, a predetermined time after it is expected, a V-pulse is generated (see case 3). When the input of the VIM does not receive any video signal, the V-pulse signal is repeatedly regenerated and the display can still be used to present some information (for example a warning message).
4. **Too large phase shifts**. Special measures are taken when an incoming V-pulse occurs during the active period and having a correct phase with respect to the previous input V-pulse. Suppose that a V-pulse is deleted because of a too early V-pulse (case 2) and the next V-pulse occurs at the correct phase. As shown in case 4 of the figure, this might occur during the active video part. Apparently, the too early V-pulse that was deleted in case 2, was caused by a phase shift in the input signal. Since the current active video part has to be finished, the regenerated internal V-pulse starts directly after the current active video. Immediately, conveying a new active video part would result in a too early V-pulse for the next V-pulse, because the remaining time is too small. Therefore, the VIM will not convey an active video part, but will go idle until a new input pulse occurs (case 5). Consequently, one field or frame is skipped at the input and has to be regenerated at the end of the processing chain (the VOM). Due to the fixed-sized format constraint, this is the only solution to correct the phase shift as fast as possible.
5. **Correct V-pulse**. This pulse occurs during the vertical blanking and at the correct phase.

H-pulses indicate the start of a video line. All situations as mentioned for the V-pulse may occur also for the H-pulse. However, for line synchronization we adopted a simplified solution to reduce system costs and because line synchronization is less critical. For example, all early H-pulses are considered as too early H-pulses. Consequently, they are rejected and line synchronization is lost.

4.3. The Video Output Module

The main task of the VIM is to convert a video signal from the real-time environment to a signal that is suitable for a data-driven communication domain. Similarly, the VOM is responsible for making a domain transition in the opposite direction. Timing information and synchronization signals for the presentation device have to be recovered and synchronization with an input video signal of a VIM should possibly be maintained if appropriate. The following subsections describe the solution for this process.

4.3.1. Frame synchronization

Frame synchronization for the Video Output Module (VOM) is different from the frame synchronization techniques used in the Video Input Module (VIM), because the VOM has to take additional constraints into account. Due to data-driven communication, the VOM has to read all active pixels from the processing units. No pixels should be skipped, for example, to start the next frame before the active pixels of the current frame are completely read.

A V-pulse can never be ignored as the VIM would do in case of "too early V-pulses". Deleting or ignoring a V-pulse would result in a picture that is continuously captured in system memory and may cause buffer overflow if the system does not contain sufficient memory. The conflicting frame can be flushed to solve this problem. Normally, any frame memory in the signal flow-graph of the multimedia application has read and write pointers with a fixed relation and cannot have collisions. In case of a "too early V-pulse", the read and write pointers in the memory are decoupled and can have collisions. Now the frame of the unacceptable V-pulse can be flushed, at the cost of limited visual artifacts. After receiving the next V-pulse, the FIFO mode of the frame memory is restored.

The VIM can also send a V-pulse to the VOM without capturing any video in case of "too large phase shifts". Consequently, the VOM cannot consume pixels from the processing units to reconstruct the frame; it would cause a buffer underflow. Instead, the VOM generates dummy pixels (e.g. black pixels) to fill the missing active part. The VIM informs the VOM about such an event through a separate synchronization channel. The V-pulse together with a valid signal for the V-pulse, is communicated over this channel. Figure 2 shows a high-level block diagram of the VIM, VOM, processing units and their synchronization signals.

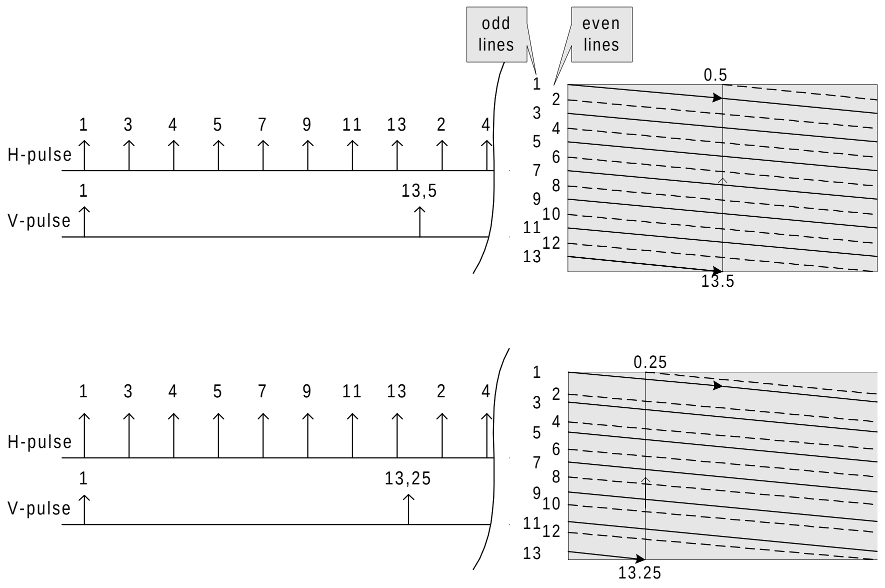


Figure 5. V-pulse position with respect to the H-pulse.

4.3.2. Interlaced video

With interlaced video, first the odd lines are displayed and subsequently the even lines. These odd and even lines should be interleaved on the display, such that the distance between all lines are equal. This is accomplished by a correct position of the V-pulse with respect to the H-pulse.

The top of Figure 5 shows that interleaving the V-pulse position at 0 (for the odd lines) and 0.5 (for the even lines) times the line time, results in a perfectly interlaced picture display. If the relation between the H- and V-pulse is not correct, such as in the bottom part of the figure, the V-pulse for the even lines starts at an irregular position, e.g. at 0.25 times the line time. In such a case, the interlacing results in a non-uniform distribution of the video lines on the display. Older video systems may have a independent H- and V-pulse synchronization systems for which exceptional rules apply. Since the V-pulse is synchronized with the input signal and the H-pulse with the external display, the timing position can be incorrect. When the VOM receives a V-pulse from the VIM, the current frame is processed until the correct position of the output V-pulse is reached. At that position, the pulse is regenerated, thereby starting the new field. The determination of the correct position is done at the input signal by the VIM and is communicated to the VOM.

4.4. Multiple video sources

The previous sections explained that the VOM synchronizes to a video signal retrieved by the VIM, but some applications require an asynchronous display of the video signal. Video PC cards displaying a 50 Hz PAL or 60 Hz NTSC signal onto a 72 Hz monitor are an example of such an application. Furthermore, when an output signal consists of several input sources, the output can only be synchronized with one of its sources. All other sources have to be displayed asynchronously. Figure 8 shows an application that consists of two video broadcasts, an internet connection and a graphics-based wallpaper background. The memory that is used to compose the video output signal can also be used to provide the asynchronous communication. Each source uses one write pointer, addressing the shared memory independently. One pointer is used for reading the composed output. These pointers can collide and overtake each other. Figure 6 shows how the pointers overtake each other for asynchronous display. When the pointers collide within the active part of a video stream, the resulting output picture consists of an upper and a lower part originating from different temporal moments. This is particularly visible when the video contains fast motion.

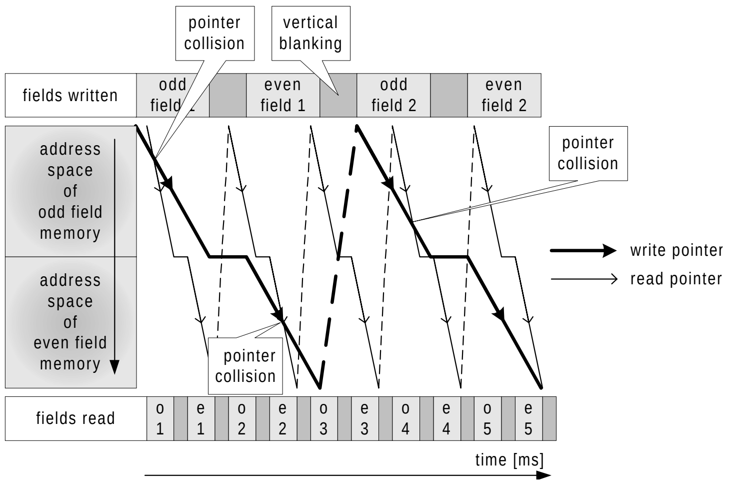


Figure 6. The read/write pointer behaviour for asynchronous output.

4.5. Latency of the processing system

The moment that the VOM receives a V-pulse from the VIM, determines when active video pixels from the processing system are used to reconstruct the frames. As a result, the amount of data that is buffered into the processing system depends on the time a V-pulse is received by the VOM. A small latency of the processing units or a large delay of the V-pulse from the VIM to the VOM will require a large data buffer, whereas a large latency or a small delay of the V-pulse requires little buffering. Obviously, the latency of the total processing system should be constant to provide a regular video stream for the display. Because the processing may vary over time, also the latency of the processing changes. For example, the user may activate a vertical sampling-rate converter to do aspect-ratio conversion in a widescreen TV set. This will increase the latency of the processing chain (the exact value depends on the scaling factor, the order of the filter and the position of the video window to be scaled). The variation in the latency has to be compensated by buffering. With a delay unit for the internal V-pulse signal (see Figure 2), the exact time the VOM receives the V-pulse can be set such that for a maximum latency application, the buffer requirement is minimal. The application with the minimum latency will then determine the maximum buffering required for compensation. This is illustrated in Figure 7.

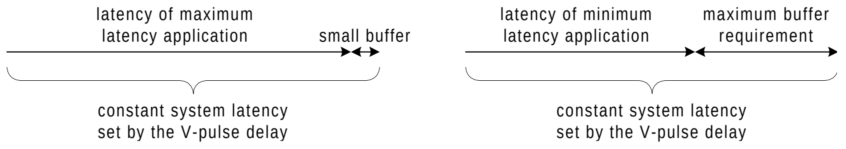

latency of maximum latency application
small buffer
constant system latency set by the V-pulse delay
latency of minimum latency application
maximum buffer requirement
constant system latency set by the V-pulse delay

Figure 7. Relation between the processing latency and the V-pulse delay.

5. CONCLUSION

We have proposed a Video I/O module featuring asynchronous communication of video, which enables synchronization of the display to one of multiple sources. The I/O modules split the internal and external clock domains and convert "non-standard" signals to frame-synchronous "standard" signals. The presented I/O model allows processing units to be individually optimized for speed, complexity, memory resources, etc. Robustness of the total system is improved by correcting the irregularities and signal deterioration at the input stage such that a fixed-sized video signal is obtained in all cases. Hence, the computing is always applied to a stable and regular signal. Simulations have shown that even randomly distributed synchronization signals can be handled correctly while the system maintains a continuous operation.

The proposed Video I/O modules have been evaluated in an experimental chip-set [2] [5], having the following capabilities. Several input formats are allowed (YUV and/or RGB with several multiplexing modes and CCIR-656). Similarly, several synchronization mechanisms are accepted. The H-, V-pulse regeneration such as discussed in Section 4, was applied. Multiple I/O modules within one system are possible, featuring variable window capturing. This enables multi-window applications as shown in Fig. 8. Moreover, the input modules can recognize automatically the picture size and the interlacing format of the input signals.

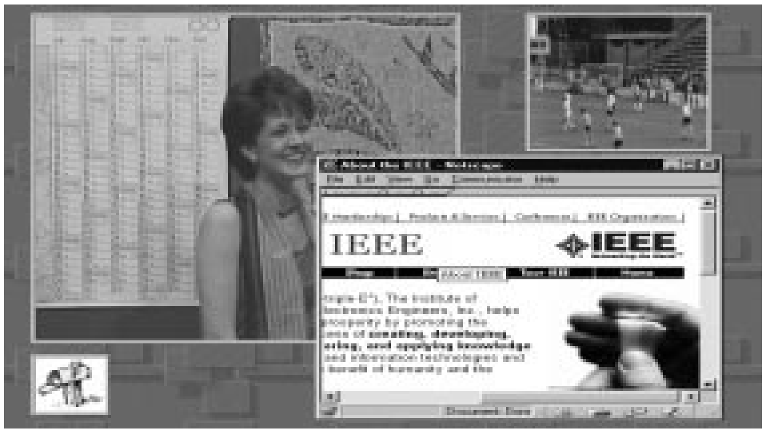


Figure 8. A video composition from four input sources.

REFERENCES

1. S. Rathnam and G. Slavenburg, "Processing the new world of interactive media," *IEEE Signal processing Magazine* **15**, pp. 108–117, March 1998.
2. P.H.N. de With, E.G.T. Jaspers *et al.*, "A video display processing platform for future TV concepts," *IEEE Trans. Cons. Electr.* **45**, pp. 1230–1240, Nov. 1999.
3. G. Kahn, "The semantics of a simple language for parallel programming," *Information Processing 74* , pp. 471–475, Aug. 1974.
4. J.A.J. Leijten, J.L. van Meerbergen, *et al.*, "Stream communication between real-time tasks in a high-performance multiprocessor," in *Proceedings of DATE '98*, pp. 125–131, February 1998.
5. E.G.T. Jaspers, P.H.N. de With and J.G.W.M. Janssen, "A flexible heterogeneous video processor system for TV applications," *IEEE Trans. Cons. Electr.* **45**, pp. 1–12, Febr. 1999.