Proc. 2nd IEEE Benelux Signal Processing Symposium (SPS-2000), Hilvarenbeek, The Netherlands, March 23–24, 2000

SYNCHRONIZATION OF VIDEO IN DISTRIBUTED COMPUTING SYSTEMS

E.G.T. Jaspers[a] B.S. Visser[b] P.H.N. de With[c]

[a]Philips Research, Eindhoven, The Netherlands
[b]University of Twente, Enschede, The Netherlands
[c]University of Mannheim, Mannheim, Germany

ABSTRACT

A distributed multimedia computing system consists of sources, processing units and presentation devices in which each component operates autonomously (see Fig. 1). This independency 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. 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. The communication interface has been used for the design of I/O modules in a multimedia system, which are briefly outlined.

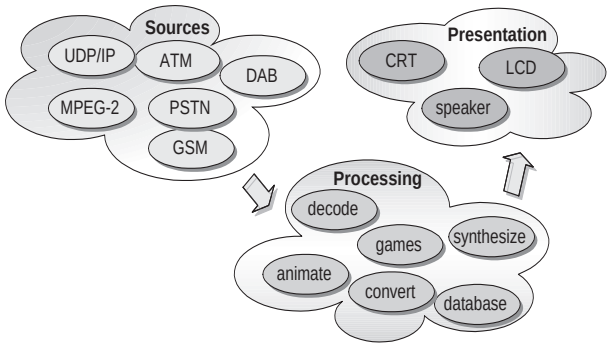

Figure 1. Independent domains in a distributed system.

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 autonomous subsystems is readily understood from the broadcasting communication model, where a broadcaster (source) optimizes its performance, whereas the receiver (player) is interested in low power and costs. Another advantage of independency is that it allows performance improvements of subsystems over time (scalability, e.g. more processing power due to a higher local clock frequency), without changing the overall system. Let us now consider the consequences of the described approach.

If the video display (presentation device) is independent from the source signals, the display is completely asynchronous. Similarly, the processing of the multimedia information is preferably independent of both the source(s) and the presentation device(s), and has its own separate processing and clock domains.

2. PROBLEM STATEMENT

Fig. 1 suggests that at least three different domains are used with communication in between. Video communication between the domains is originally stream-oriented, but video signals are intrinsically divided into frames and video lines, which can be regarded as data packets. These packets can be used to synchronize video communication between the various domains. The level of synchronization depends on the granularity of these data packets. For example, synchronization on video sample level requires accurate information about the start of a video sample, whereas for synchronization on video-line level (or frame), accurate time information about start of a video line (or frame) is required by means of H- (or V-) synchronization signals. 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. These requirements are constrained by the display, hence we have assumed that the video signals are adapted to the used display.

- 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, e.g. for a CRT, the line rate and scan speed has to be known, for an LCD the resolution.

The aforementioned system aspects give direct requirements for the processing subsystem. Firstly, since independent clock domains are used also for communication, synchronization between the display and (one of) the sources

should be provided at frame level. Secondly, a 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.

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

3. ASYNCHRONOUS COMMUNICATION

In this section we discuss in more detail how communication between processing subsystems, having different clock domains, is established. For this purpose, we apply the Kahn [1] network model and explain rules for data-driven communication.

For our distributed multimedia system, the following aspects of real-time video processing applications have to be considered.

- Video applications require high-speed computation on vast amounts of data. Thus, high parallelism in computing and communication is essential.
- The rate of making control decisions is much lower than the pixel processing rate. This means that a large amount of data is communicated and processed before the next control decision is made.
- The individual components in Fig. 1, can have its own clock domain and its own latency. As a result, parallel operations cannot be scheduled at compile-time, because the timing relation is *a priori* unknown.

A data-driven communication model proves to be suitable for these aspects. An example of a video processing application using the data-driven model for asynchronous communication between separate clock domains can be found in [2].

The Kahn process network model [1] is an attractive dynamic model for data-driven communication. The model describes a unidirectional asynchronous communication network that is modeled as a directed graph. Each process is concurrent and is modeled as a node in the directed graph. These concurrent processes communicate asynchronously through unidirectional *edges*, i.e. queues of unlimited capacity (writes are never blocked). However, reads on input queues can block the corresponding process if no data is available.

A video application can be easily mapped onto the Kahn communication model. The data-flow consisting of a set of independent video processing tasks is mapped onto the nodes of a directed graph. Video communication is asynchronous and is based on packets. These video packets are queued on the edges and are received in the same order as they were sent. In hardware, this queuing scheme can be implemented with FIFO buffers (see Fig. 2). Alternative models for data-flow communication have been reported [3], but they do not support the system aspects as discussed earlier.

Implementation of the Kahn model in a real video system requires some adaptation of the communication rules. Read operations, which empty a FIFO, are no problem. However, unconstrained write operations (called non-blocking), which fill a FIFO, must be restricted, because FIFOs of unlimited capacity do not exist. Therefore, in hardware only so-called blocking write operations on a FIFO of limited capacity can be implemented. Blocking write operations can be implemented by suspending the internal clock of the video task. This modification does not violate the validity of the Kahn model. For more details see [4].

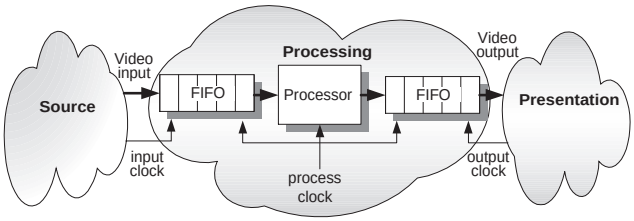

Figure 2. FIFO's for asynchronous communication.

In the sequel, we define a Video Input module (VIM) and Output Module (VOM) that can be applied in the modified Kahn model. Both VIM and VOM provide the interface between the data-driven processing and the real-time environment (e.g. camera or a display). When going from one domain to the other, timing information can be lost when no special precautions are taken, so that there is no relation between the pixel position and time. 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 of the pixels, provided that the start of a frame is known (or indicated). This will be addressed in the next section. A picture format of fixed size is also referred to as "standard" signals.

Summarizing, a processing component in the distributed system based on asynchronous communication, has a fixed frame size and 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.

- A real-time video source will continuously produce samples and a video display will continuously consume samples.

- 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).

We have adopted the following measures to make the VIM and VOM robust for the presented constraints. Let us first address the design of the VIM.

The tasks successive to the VIM expect a fixed frame size. An 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. The VIM will complete the video frame with dummy samples (see next section).

For the VOM similar measures are taken. If the display consumes samples too fast (the display expects video samples and no samples are provided), the VOM generates dummy video samples for its display.

Finally, a separate synchronization V-pulse from the VIM to the VOM is communicated to recover the exact start time of the 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. 5).

4. SYNCHRONIZATION

In practice, ”non-standard” video signals having variable number of pixels per line and number of lines per frame, occur regularly (e.g. the picture output of video recorder in trick-play mode). As indicated in the previous section, the VIM supplies the processing units with ”standard” signals and additionally, the synchronization signals of the input signal have to be adapted accordingly. This is accomplished by means of a H- and V-pulse re-generator. Only the active (visible) part of a video signal is captured and conveyed to the processing units (see Fig. 3). In the blanking (non-visible) part, the V-pulse may vary. Consequently, the following synchronization pulses have to be corrected.

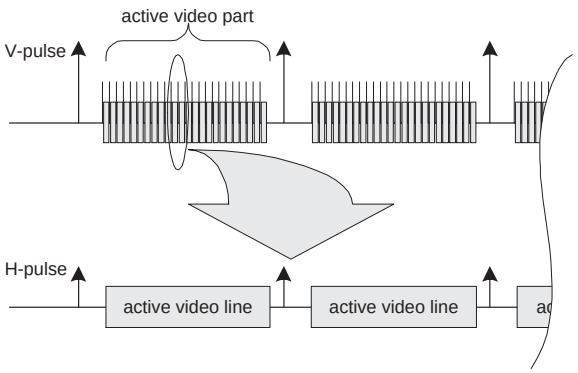

Figure 3. A standard video signal with H- and V-pulses.

- **Too early pulses** : pulses that would interrupt an active video part cannot be accepted, because the size of this video part is fixed. For example, this happens when an active video part starts a number of lines after a V-pulse and this V-pulse occurs more than the same number of lines earlier than the end of the preceding active video frame. Such V-pulses are deleted. A V-pulse that occurs later in the active video part is accepted and called an early pulse.
- **Early pulses** : Ultimately, the active video starts immediately after finishing the preceding active video frame.
- **Missing or too late pulses** : when a V-pulse is not received, a predetermined time after it is expected, a V-pulse is generated. When the input video signal is removed from a VIM, this V-pulse signal is therefore continued and the display can still be used to present graphical information (warning message).
- **Too large phase shifts** : when a deleted and generated pulse is incorrect (see case 1 and 2 in the Fig. 4), but was caused by a phase shift, the current active video part is finished (case 3) and the VIM is set in idle mode until a new input pulse occurs (case 4). Due to the standard-signal constraint, this is the only solution to correct the phase shift as fast as possible.

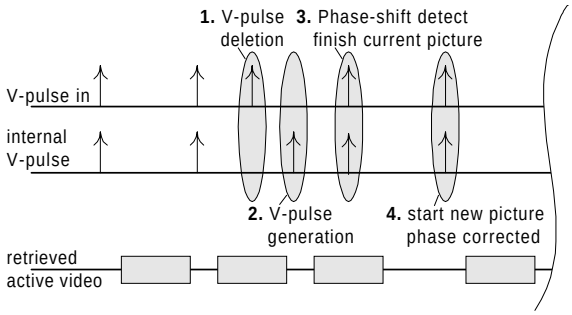

Figure 4. VIM handling of a too large phase shift.

For the VOM, V-pulse modification should be considered carefully, because the consequences work out differently than for the VIM. A V-pulse from the VIM received by the VOM cannot be accepted under all circumstances. The main problem is that if the VOM reads the active pixels of a picture in the system, it has to read all pixels before it can start the next picture (no loss of pixels). As a result, the input data of the VOM can only be blocked for a moment, but not skipped. This means that the solution in the VIM for ”too early” V-pulses cannot be applied for the VOM. Deleting or ignoring a V-pulse would result in a picture that is caught in the system and may cause buffer overflow if the system does not contain sufficient memory. To solve this problem, a picture memory in the system is used to flush the frame of the V-pulse that cannot be accepted. Normally, this picture memory is used as FIFO where read and write pointers are synchronized and cannot have collisions. In the case of a ”too

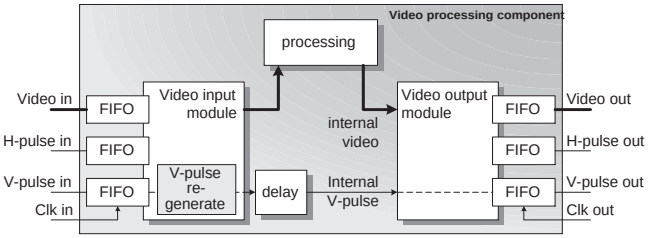

Figure 5. Block diagram of processing component.

early" V-pulse, the read and write pointers are decoupled and may show collisions. This causes the picture with the unacceptable V-pulse to be flushed at the cost of limited visual artifacts. After receiving the subsequent V-pulse, the FIFO mode of the picture memory can be restored. To prevent the use of an additional picture memory, any frame buffer in the signal flow-graph can be used for this purpose.

Finally, we emphasize that the timing position of the output V-pulse with respect to the H-pulse is important, because it determines the vertical phase of a field picture for interlaced sequences. Since the V-pulse is synchronized with the input signal and the H-pulse with the external display, the timing position might not be correct. When a V-pulse from the VIM is received, the current picture is processed until the correct position of the output V-pulse is reached. At that position, the pulse is regenerated, thereby starting the new picture. The determination of the correct position is measured at the VIM.

5. FEATURES

The combination of video input module (VIM) and video output module (VOM) has a powerful synchronization mechanism that copes with signal irregularities of the external environment. Furthermore, the modules support several formats and are highly parameterizable. This is illustrated by the following features.

- **Format.** The VIM accepts YUV (chrominance / luminance) video streams and enables communication of both 8-bit multiplexed (U-Y-V-Y) and 16-bit (UY-VY) video streams. The output formats of the VOM are similar to the VIM formats, but it supports also RGB-based video streams.
- **Synchronization.** The VIM accepts 3 different synchronization mechanisms: according to CCIR-656, synchronization on H-/V-pulses and synchronization on blanking identifier signals.
- **H-, V-pulse regeneration.** See Section 4.
- **Window capture.** Defines the video area to be processed. The VIM and VOM are able to handle any image resolution up to High Definition (HD). Sizes of the input and output format can be modified internally by e.g. a scaler.
- **Resolution Recognition.** The VIM recognizes automatically the picture size of the incoming video stream and the interlace factor.

Figure 6. A video composition from four input sources.

6. 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. The concept of such a Video I/O module has been evaluated in an experimental chip-set [2] [5], where it was generalized with interface protocols such as CCIR-656. Fig. 6 shows an application produced with the chip-set. Several distributed sources such as two video broadcasts, internet and a graphics-based wallpaper background are processed, composed and subsequently presented on a display device.

REFERENCES

1. G. Kahn, "The semantics of a simple language for parallel programming," *Information Processing 74* , pp. 471–475, Aug. 1974.
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. D.D. Gajski *et al.*, "A second opinion on data flow machines and languages," *IEEE Computer* **15**, pp. 58–69, Feb. 1982.
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.