TVP - Introduction and application requirements


High-end television requires a flexible architecture that supports the programmable display of multiple video windows. This was addressed in the TVP project. This article describes the signal flow graph concept as model for architecture and application and discusses the application requirements in terms of amount of compute power, memory size and memory bandwidth. It is shown that these figures are beyond the capabilities of cost-effective programmable general-purpose devices. More processing power with low external memory bandwidths can be gained with a more application-specific multi-processor architecture.

1. Introduction

In the consumer electronics area, the display technologies used for television and computers are gradually merging, as a result of emerging features such as Internet access, electronic program guides, electronic shopping, help wizards, games and video conferencing. Consequently, provision of video processing for displaying high-quality television applications clearly need to envision the consumption of multiple simultaneous information channels, as opposed to the watching of a conventional single-channel broadcast. A key feature of a TV set that supports multiple information channels is that two (or more) signals together with graphical information can be monitored in real time on its display together with graphical information. An additional desirable feature of such a TV set is that the installed signal processing power can be reallocated as required to process the various signals to be displayed. This flexibility can be used for improving overall picture quality, because signal enhancement can be preferentially applied to the signals that need it more.

The system requirements for a new display processing architecture are influenced by the following trends in TV set design and computer systems:

- TV is influenced by PC technology with respect to computing techniques, SW architectures and applications,
- since memory devices continuously become larger and cheaper, it is desirable to have one uniform shared background memory,
- new features stemming from standards created by international bodies have to be realized quickly and therefore 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 features in a TV set call for cost-effective implementation, since it is a highly competitive market.

Several proposals for TV architectures have been developed,[1][2] but the described architectures lack the flexibility and programmability to satisfy the aforementioned requirements.

This paper is divided as follows. Section 2 gives the computational requirements of typical TV functions. Section 3 describes a typical TV application, in terms of a signal flow graph model of the architecture. Section 4 further elaborates on the topic of Section 3 in terms of costs and memory requirements.

2. Costs of various TV functions

At first glance, it seems that the required architecture could be optimally implemented on a fully programmable multi-media processor. The question that

<table>
<thead>
<tr>
<th>Function</th>
<th>Operations per Second</th>
<th>Memory / Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>H,V zoom / compress</td>
<td>400 MOPS</td>
<td>samples-lines</td>
</tr>
<tr>
<td>Filters, Comb filters</td>
<td>200 MOPS</td>
<td>samples-field</td>
</tr>
<tr>
<td>Advanced Peaking</td>
<td>650 MOPS</td>
<td></td>
</tr>
<tr>
<td>Colour Transient Improvement</td>
<td>300 MOPS</td>
<td></td>
</tr>
<tr>
<td>Dynamic Noise Reduction</td>
<td>500 MOPS</td>
<td>field</td>
</tr>
<tr>
<td>MC-100 Hz</td>
<td>2 – 4 GOPS</td>
<td>2 – 3 fields</td>
</tr>
<tr>
<td>Colour Space</td>
<td>150 MOPS</td>
<td></td>
</tr>
<tr>
<td>Teletext conversion</td>
<td>10 MOPS</td>
<td>field</td>
</tr>
<tr>
<td>Adaptive Luminance</td>
<td>60 MOPS</td>
<td>1 KByte</td>
</tr>
</tbody>
</table>
will be answered subsequently is: how much computing power should be provided? For this reason, a number of typical TV signal-processing functions were studied and computational and memory requirements were calculated.

Fig. 1: Example of a multi-window application.

... gives the intrinsic computational and memory requirements of several TV functions. With respect to operation counting, additions, multiplications, etc., are considered as single operations. The third column shows the amount of memory or cache required. Here it is optimistically assumed that information can be retrieved or stored by single reads and writes. For a typical TV application with 50-to-100 Hz conversion, Picture-in-Picture (PiP), noise reduction and aspect-ratio conversion, the amount of operations per second already exceeds 6 GOPS (Giga operations per second). This cannot be cost-effectively implemented on a general-purpose processor. With respect to storage, it can be observed that caching of the field memories for e.g. MC-100Hz (MC stands for Motion Compensated) conversion, dynamic noise reduction or comb filtering is too expensive for on-chip integration using embedded memory. Instead, these functions can share the use of a large unified memory in the background. However, the consumption of the already scarce bandwidth of this memory should be monitored.

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

3. Flow graph of a typical TV application

3.1 Signal Flow Graph concept

Let us consider a full-featured application (see Fig.1). The application displays two real-time live video sources, one expanded in the background and one compressed to a Picture-in-Picture (PiP) format (left window).

The second window displays an Internet page which can be generated by the TV’s general-purpose CPU. The CPU generates the Internet page and the graphics borders of the windows and writes this into the shared background memory. Fig.2 shows the

Fig. 2: The signal flow graph of the full-featured multi-window application.
complete signal flow graph of this combined video-graphics application.

3.2 Motivation for the ordering of the functions

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

- first, it enables the vertical scaler to read only the part of the image that has to be expanded;
- second, if vertical scaling is applied to interlaced fields, the result may suffer from vertical aliasing artifacts. This is caused by the fact that vertical filtering of interlaced field is equal to filtering of progressive video with zero-valued coefficients at the positions of the missing field lines. This results in a repeat spectra of the filter with a band pass at half the sampling rate. These high frequencies cause the aliasing when scaling is applied. As a solution, the interlaced video is read progressively from the memory and is first de-interlaced by the scaler, by means of a three-tap median filter. Afterwards, vertical scaling is performed;
- the third reason for the memory access prior to vertical scaling is that the output of the scaler is driven by the rate of the output, which is \( f_i \). Due to the expansion, the number of input video lines is less than the number of output video lines. Because the video part that is read by the scaler is expanded to the original image size, the rate of the input video lines is exactly two times the expansion factor \( Z \) smaller than the output rate. The factor of two results from the progressive reading of the video lines. Thus the input rate of the vertical scaler equals \( 2/f_i \), while the rate of the video source is \( f_i \). This rate difference requires some buffering, which is provided by the memory access.

The order of horizontal and vertical scaling is relevant for memory and cache usage. Since cache memory is generally shared with other tasks and because it is scarce in most systems due to silicon area, the usage should be optimized. For PIP creation with optimal cache usage, the horizontal (down)scaling is therefore performed prior to vertical scaling and for expansion of the background picture, horizontal scaling is carried out after vertical scaling.

4. Memory and memory bandwidth requirements

Apart from the required raw compute power as listed in Table 1, the requirements for memory size and memory bandwidth between an SDRAM background memory and the signal processing chip are of major concern. Let us address these memory issues. The system memory device runs at a clock rate of 96 MHz and has a bus width of 32 bits. For 16-bit pixels, this means a total memory bandwidth of 192 Mpix/s. The communication between the CPU and the coprocessors is performed via memory. Therefore, part of the memory bandwidth cannot be used for video processing. Assuming 30 MByte/s of memory bandwidth for control and communication between the CPU and the coprocessors, a bandwidth of 177 Mpix/s remains for video processing. Table 2 presents an overview of all memory accesses, including the necessary amount of memory and the number of inputs to and outputs from the memory. First, the noise-reduction (NR) coprocessor accesses the memory to use a field memory for advanced adaptive temporal filtering. For an SD image of 288 lines x 854 pixels with 2 Byte/pixel, the required amount of memory equals 0.49 MByte. For a pixel rate of 16 Mpix/s, the total memory bandwidth for writing and reading is 64 MByte/s.

<table>
<thead>
<tr>
<th>Connections to/from Mem</th>
<th>Memory (MByte)</th>
<th>Mem. bandwidth (MByte/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR (expansion)</td>
<td>1/1</td>
<td>0.49</td>
</tr>
<tr>
<td>VS (expansion)</td>
<td>1/1</td>
<td>&lt;0.98</td>
</tr>
<tr>
<td>Juggling (write/Read, worst case)</td>
<td>2/1</td>
<td>0.98</td>
</tr>
<tr>
<td>Graphics (GFX)</td>
<td>0/1</td>
<td>0.49</td>
</tr>
<tr>
<td><strong>Total (worst case)</strong></td>
<td><strong>4/4</strong></td>
<td><strong>2.94</strong></td>
</tr>
</tbody>
</table>

The memory requirements for the access prior to vertical scaling are different. The image is written at 16 Mpix/s, but is read at 2 x 16/Z Mpix/s for inter field processing, with Z being the expansion factor. Because \( Z > 1 \), the bandwidth to the memory is smaller than 96 MByte/s. If intra field processing is used for vertical scaling, the data rate is even less than 16/Z Mpix/s.

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

$$\text{Buf}_{\text{total}} = 2 L_0 B_l / Z,$$

with $B_l$ the number of bytes per video line. For intra field scaling, buffering is only necessary to compensate for the rate difference between the writing to and reading from the memory. In this case the calculation is less straightforward, but can be shown to result in:

$$\text{Buf}_{\text{intra}} = L_df / 4.$$

For $L_f = 288$, the amount of required buffering is $\text{Buf}_{\text{intra}} = 0.12 \text{ MByte}$.

Finally, the mixing or juggling of the video streams is discussed. This function can also be optimized with respect to memory usage and memory bandwidth. In the background memory, one frame memory is allocated to the construction of the mixed image. This frame memory is filled with the background video image, except for the pixel positions where other video windows are located. The second video window is written into the unfilled area that is created in the background picture. The total amount of data stored is thus equal to the data of one complete picture and similarly, the total required bandwidth equals the bandwidth for writing one complete video stream. Using a frame memory also synchronizes the two video streams.

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

To summarize, the total amount of memory is less than 3 MByte and the maximum memory bandwidth is 256 MByte/s. This bandwidth is only required during transfer of the active pixels. At the blanking times, no data transfer is necessary, thereby decreasing the required bandwidth significantly. To be able to decrease the required peak bandwidth, the data transfer rate can be equalized over time. To do this, the read and write accesses to the memory have to spread the transfer of an active video line over the time of a complete video line including the horizontal blanking. Typical video signals contain 15% horizontal line blanking time, so that the total amount of bandwidth can be reduced by 15%. This leads to a memory bandwidth requirement of 218 MByte/s.

5. Other applications

The previously discussed application is an example from a large set of possible features. The background image could also be "graphic wallpaper". Two live video windows could be displayed on top of this wallpaper, together with some additional windows displaying teletext pages. It is obvious that applications such as PIP replay, split screen, aspect-ratio conversion, etc. are relatively simple applications, and therefore can each be supported by a fraction of the processing required for the aforementioned example application. Furthermore, the specified system will be able to support a large range of resolutions and frame rates. It should even be able to support 50–100 Hz conversion, by making intelligent use of the memory.

Acknowledgement

The concepts described in this paper have been developed under coordination of an architecture team with representatives from the sub-projects Applications, Silicon Design and Software. The contributions of the following people to the application studies are gratefully acknowledged: Erwin Bellers, Laurens Doornheim, Gerard de Haan, David La Hei, Hans de Jong, Paul Hovens, Harry Kuipers, Jef van Meerbergen, Ton Nillesen, Erik Riddersma, Marino Strik, Adwin Timmer, Teun Poot, Kees Vissers, Kees van Zon.