"Digital Video Transmission through the Electrical Power Lines"

Alexandre Abreu $_{(i)}$***, Nuno Roma*** $_{(ii)}$***, José Gerald and Leonel Sousa***

DEEC - IST/INESC
Rua Alves Redol, nº 9, 1000 Lisboa, PORTUGAL
E-mails: l39775@alfa.ist.utl.pt $_{(i)}$, l39921@alfa.ist.utl.pt $_{(ii)}$

ABSTRACT:

This paper describes a digital video transmission system, which uses the electrical power lines as the transmission media. The system performs video acquisition and formatting, H.263 coding and decoding and power line signal transmission and reception. Multiple Digital Signal Processors (DSPs) make the core of the system, used for video coding and decoding.

1. INTRODUCTION

Over the last few years, we have faced an increasing growth of applications in the fields of general communications, multimedia, telemetry and remote control. As a consequence of this technological evolution, there has been a substantial increase in the complexity and dimension of the buildings' cabling structures. The project presented in this paper uses an advantageous form of intercommunication between several existing systems, in the way that it doesn't require extra cabling. The solution is to use, as a communication channel, the electrical power distribution lines. As an application for this communication channel, we have developed a system that codes, transmits and decodes digital video according to the H.263[1] recommendation using, for that purpose, a set of three DSPs TMS320C50 from Texas Instruments. This system can be used for a large range of applications including, for example, remote surveillance and video intercoms.

2. GENERAL SYSTEM OVERVIEW

The system is composed of three distinct stages: one implements the acquisition and coding of the video signal; consequently, another one performs the opposite process, decoding the digital data and converting it into an analog video signal; between these two stages, there is the one that performs the transmission and reception through and from the electrical power lines. This last stage can, therefore, be divided into a transmission and a reception sub-system (see **Figure 1**).
The coding stage is implemented by means of two Field Programmable Gate Arrays (FPGAs), which format the digital video signal generated by the A/D converter, in order to simplify the DSPs' processing task. Each sampled image is, then, divided into two sub-images. Each DSP processes one sub-image and sends the resulting bit stream to the corresponding video buffer (FIFO memory) in 16 bit words. Finally, a microcontroller converts the words from both buffers into a single bit stream and delivers it to the transmitter sub-system.
In the transmitter stage, the Viterbi algorithm is applied to the generated bit stream, in order to increase the protection of the digital data against transmission errors. The resulting stream modulates a 100 kHz carrier using a Gaussian Minimum Shift Keying (GMSK) modulator. The modulated signal is amplified and delivered to the power line through the power line interface module. In the reception sub-system, there is also a power line interface module that performs the

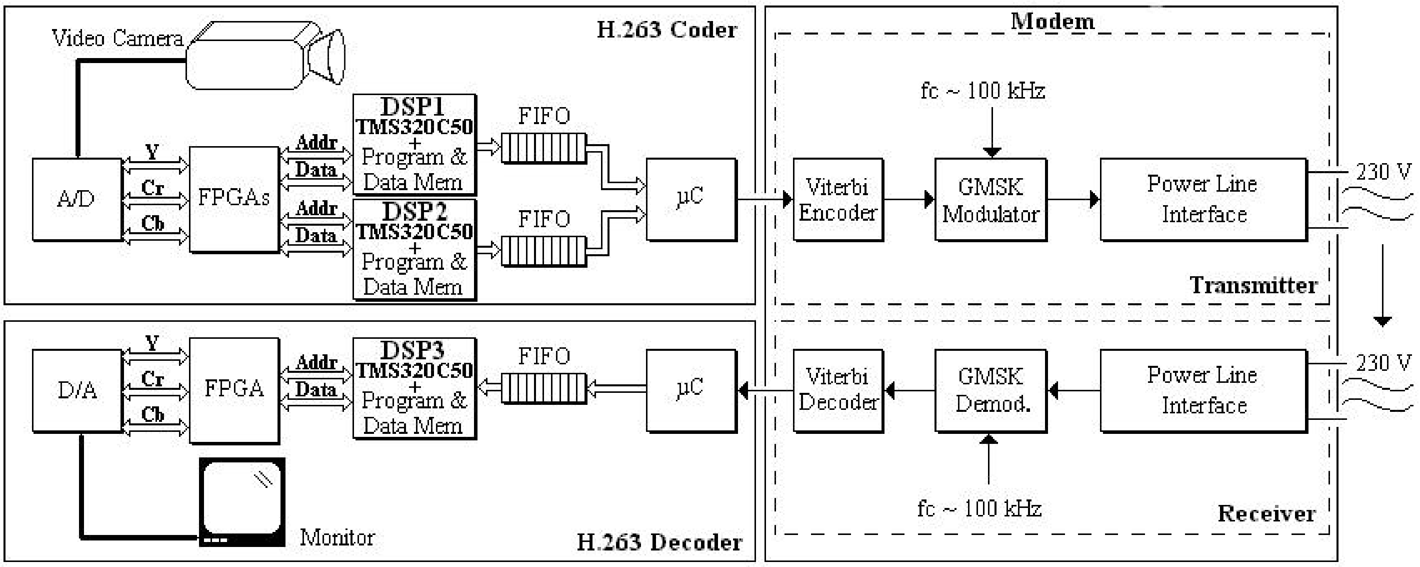

Figure 1 - Diagram with the main blocks of the system

filtering and amplification of the received signal. The resulting signal is demodulated and passed through a Viterbi decoder, obtaining the original H.263 bit stream.
Next, the received H.263 bit stream is processed by a microcontroller that performs the synchronisation and storage of the 16 bit words into a video buffer. Then, the DSP decodes them and sends the resulting digital video signal to a D/A converter by means of an FPGA that, using a similar method to that of the coding stage, performs a rearrangement of the digital video signal. These rearrangements will be described with more detail later.

3. ACQUISITION AND FORMATTING

As previously mentioned, an A/D module (based on Brooktree's Bt812 [5] chip) performs the sampling of the PAL or NTSC analog video signal, separating it into its Y, Cr and Cb components in the form of 8 bit samples. These samples are then formatted and rearranged in a matricial

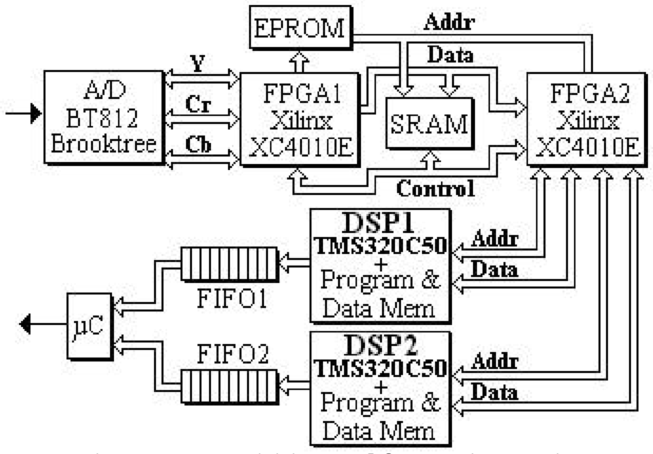

Figure 2 - Acquisition and formatting section

form by FPGA1, simplifying the DSP's processing task. The aim of this rearrangement is to group the samples in H.263 blocks (8x8 samples) and macroblocks (4 luminance and 2 chrominance blocks). The FPGA used was from Xilinx's XC4010E series [7]. The received samples are written into an SRAM (Static RAM) according to a sequence of addresses stored on the EPROM (see **Figure 2**). These stored samples are, then, processed by another FPGA (FPGA2) similar to the previous one. Circuits in FPGA2 can be functionally divided in two sections: the processing and the control. The control section implements a set of registers that are used to control the processing part performed by FPGA2, the functional behaviour of FPGA1 and the data exchange between the DSPs. The processing section reads the previously processed sample from the DSP's memory, subtracts it from the actual sample available in the SRAM and stores it back in the DSP's memory. This data exchange between the DSPs' memory and the FPGA is performed by means of a Direct Memory Access (DMA) scheme. It is also the task of these FPGAs to separate the image in two non-overlapped sub-images, assigning each one to a DSP.
It should be noted that the DSPs could have done all the processing performed by this stage. However, once that would lead to an intensive use of non-sequential external memory access, the DSPs' processing would become much slower and the algorithm's efficiency would be severely reduced.

4. IMAGE CODING

4.1 H.263 RECOMMENDATION

The image coding used in this project is based on the H.263 recommendation. According to this recommendation, each Quarter of Common Intermediate Format (QCIF) image is made up of blocks (matrixes of 8x8 pixels) with information

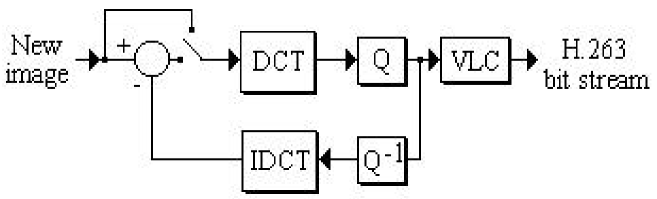

Figure 3 - The H.263 recommendation

concerning either luminance (Y) or chrominance (Cr or Cb). These blocks are grouped into larger cells (macroblocks) comprising of 4 Y blocks, 1 Cr block and 1 Cb block. In this system, we use QCIF images, which have a resolution of 176x144 pixels. These images are arranged in 9 lines of 11 macroblocks each (99 macroblocks). Each line of macroblocks is called a Group Of Blocks (GOB).
The algorithm has two distinct modes of operation, which are INTRA mode and INTER mode. With INTRA mode, the image processing is based solely on a single frame. With INTER mode, the temporal correlation between consecutive pictures is used to minimise the generated bit stream. To use this temporal correlation, the H.263 recommendation states that the image processing should be done on the information that results from subtracting the previously processed frame from the most recently acquired one. Therefore, the previously processed frame acts as a prediction of the most recent one. Moreover, the recommendation also defines the use of motion vectors corresponding to translations of similar macroblocks between frames. Therefore, depending on the similarity between the predicted macroblock and the current one, a decision is made on whether to use INTER or INTRA coding mode. During most of the time, and because of the above-mentioned correlation, INTER mode is chosen in order to achieve a smaller bit rate.
For both modes, each block is transformed using the Discrete Cosine Transform (DCT) that takes advantage of the spatial correlation of pixels in the same block. The resulting coefficients are quantized (Q) and entropically coded (VLC -

Variable Length Coding) together with the motion vectors and other control parameters.
After quantization, however, the coefficients are also used to build the prediction for the next frame. Therefore, they are dequantized and inverse transformed. This prediction also represents the image that will be present at the H.263 decoder, after processing the frame's bitstream. As mentioned previously, if INTRA mode is used, no prediction is used and the macroblocks are DCT transformed straightaway (see **Figure 3**)

4.2 H.263 IMPLEMENTATION

As previously mentioned, each DSP performs the coding of one of the two sub-images. The information corresponding to a sub-image is placed in the DSP's memory, through a DMA scheme, by the FPGA2.
After reading the previously processed image (which is stored from the memory address 2C00H) and the new sampled image from the A/D module, FPGA2 subtracts these two values and stores the result in the DSPs' memory according to the memory map described in **Figure 4**. This way, the coding algorithm works, by default, in INTER mode since the DSPs process the data resulting from the subtraction of the two images. However, if some macroblocks (or even the entire image) are to be coded in INTRA mode, the processor only has to fill all the necessary memory positions corresponding to the previously processed image with the zero value. These aspects of the coding algorithm are shown in **Figure 5**.

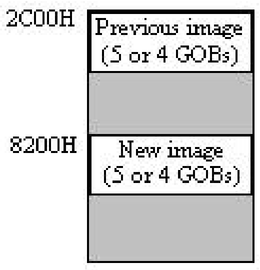

Figure 4 - Data memory map

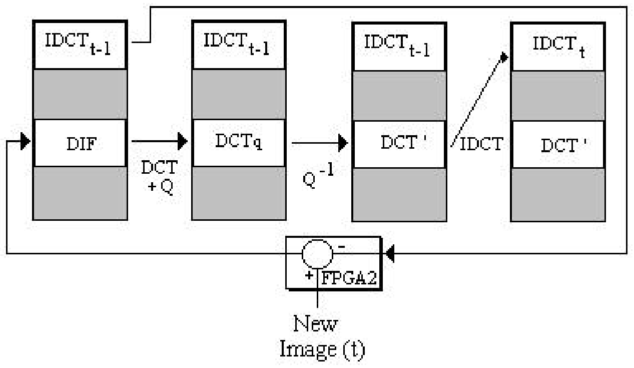

Figure 5 - Image coding sequence

4.2.1 Coding Architecture

The image coding is performed using both DSPs in a parallel-processing scheme according to the diagram shown in **Figure 6**. One of the main goals of this implementation is to make an independent processing in the two DSPs. In fact, since the data to process (given its size) cannot be kept in the DSPs' internal memory, a shared memory scheme was initially considered. However, that would degrade the system's performance because it requires that each processor,

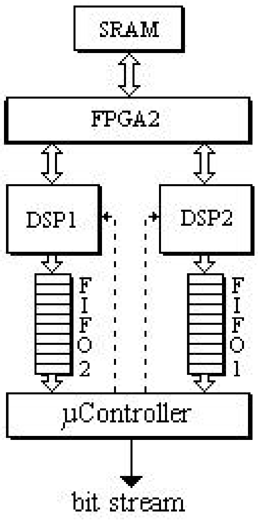

Figure 6 - Image coding implementation

at a given time, enter the HOLD state (placing its buses in high-Z), in order to exchange data between the two processors, interrupting the processing. Therefore, by following the independent parallel-processing scheme, the processing is only interrupted once per image, when FPGA2 is writing the data into the DSPs' memories.

4.2.2 Discrete Cosine Transform

There are several possible algorithms to perform the DCT calculation [3]. The algorithm based on the DCT definition computes it using the following equation:

$$F(u,v) = \frac{C(u).C(v)}{4}\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] \quad \textbf{(1)}$$

where C(0) is $1/\sqrt{2}$. For other values of *u* and *v*, this coefficient is 1. The variables *x* and *y* are the spatial coordinates in the spatial domain and variables *u* and *v* are the coordinates in the transform domain. In practice, this algorithm can be implemented as a multiplication of the block's pixels matrix (**A**) by the DCT coefficients matrix (**B**) according to **F**=**BAB**^{T}.
It is important to mention that there are some algorithms referred to in the literature, which are more efficient than this

last one, regarding to the number of arithmetic operations required. However, because of the non-sequential nature of the memory accesses and of the code itself, these algorithms would use an excessive number of branch instructions. Given the C50's pipeline structure, each jump instruction takes 4 cycles. Therefore, for these algorithms, the processor would spend most of its time indexing memory locations and executing control instructions. Therefore, we concluded that the matrix multiplication algorithm is the most suitable for DSP implementation, due to its architecture and programming characteristics.

The implemented DCT algorithm performs the matrix multiplication in two steps. In order to take advantage of the Multiply and Accumulate [2] (**`MAC)`** instructions (and since matrix multiplication is not commutative), care has to be taken to avoid the need to copy values among data and program memory, between the two steps of the algorithm. Each of the two mentioned steps performs $\mathbf{Y} = (\mathbf{M_D} \times \mathbf{M_P^T})^T$ where $\mathbf{M_D}$ is a matrix in data memory and $\mathbf{M_P}$ is a matrix in program memory. In the first step, $\mathbf{M_D}$=**A** and $\mathbf{M_P}$=**B**, with **B** being the DCT coefficients matrix. For the second step, $\mathbf{M_D}$ is the result matrix from the first step and $\mathbf{M_P}$=**B**, obtaining

$$\text{DCT} = \left[(\mathbf{AB}^T)^T \, \mathbf{B}^T\right]^T = \mathbf{B}\left[(\mathbf{AB}^T)^T\right]^T = \mathbf{BAB}^T \qquad \textbf{(2)}$$

Therefore, according to what has been stated, it's only necessary to consecutively apply two identical matrix multiplication routines to the image data. These algorithms consist of a repetitive sequence of vector multiplications (blocks' lines and columns). Each of these multiplications can be performed by code sequences like:

```
RPT 7
MAC coef, *+
```

However, this algorithm was still subjected to some improvements by avoiding branch instructions. This was achieved by using the *loop unrolling* method. In fact, only by applying this method, were we able to obtain a 20% improvement in the DCT final execution time (the DCT of a block takes a total time of 96µs).

4.2.3 Quantization

The following step involved in the H.263 coding algorithm is the quantization of the DCT coefficients. In order to avoid a new series of memory accesses to fetch each DCT coefficient and write back its quantized value, the quantizer is integrated with the DCT computation. Even though this aspect may seem irrelevant, the time to quantize an entire H.263 QCIF image can take about 3,8ms.

During quantization, a qualification of the coded macroblocks was also performed by saving information about the values of the resulting quantized DCT blocks for each macroblock. This information tells us, for example, if all blocks in a macroblock have zero valued coefficients or if the only non-zero value is their DC coefficient. As referred to in the following sub-sections, this information can be used to reduce the time spent in computing the inverse DCT.

4.2.4 Variable Length Coding

After quantization, an entropic coding of the obtained coefficients is performed. The algorithm used for this purpose takes advantage of the macroblock qualification mentioned before and determines, for each coefficient, a set of bits which represents it, according to a table included in [1]. Then, these sets of bits are concatenated and stored as 16 bit words in the DSPs' memories. A small intermediate buffer is, therefore, generated, which contains the bit stream corresponding to the coded image. By doing this, the serialization of this bit stream is simplified since each write access into the video buffer consists of 16 data bits to transmit

4.2.5 Dequantization

After writing the bit stream into the video buffer, it is still necessary to determine the image used for prediction in INTER coding mode. However, as mentioned before, if a macroblock is coded in INTRA, this and the following procedure can be replaced by just writing zeros to the corresponding memory locations.

To make the algorithm even faster, we have only used quantization steps of the form 2^i where *i* is an integer value. By doing this, this stage just has to perform a left shift of the quantized coefficients.

4.2.6 Inverse DCT

In this phase of the coding process, a recovery of the video image is made from the dequantized DCT coefficients. The algorithm used to implement the Inverse DCT (IDCT) is similar to the one used to compute the DCT. The main difference relies on the coefficient matrix, which, for the IDCT, is the transposed matrix of the DCT's coefficients (**B**).

However, the qualification of the macroblocks performed in the DCT's computation stage is used to reduce the computational load for IDCT calculation. That qualification allows for a distinction between three different kinds of blocks:

- if all the coefficients in a block are 0, the IDCT's result for this block will also be a set of 64 zero coefficients; therefore, no computation is made.
- if only the DC coefficient (1st line, 1st column) is non-zero, it can be proved that it is only necessary to divide the DC coefficient value by 8 (or left-shift it 3 positions) and 'spread' it (**Eq.3**) over the remaining 63 positions of the block.

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{16}\right]\cos\left[\frac{\pi(2y+1)v}{16}\right] =$$

$$= \frac{1}{4}C(0)C(0)F(0,0) = \frac{1}{4}\frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}F(0,0) \Leftrightarrow f(x,y) = \frac{1}{8}F(0,0)$$

(3)

- if the block contains non-zero coefficients other than the DC one, the complete IDCT calculation is performed.

The data obtained after this process is used as a prediction for the new image, being read by FPGA2 and subtracted from the new image.

5. DECODING

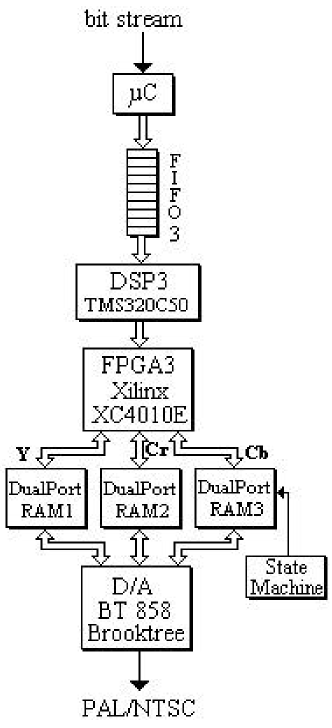

Figure 7 - Image decoding implementation

Unlike the coding stage, the decoding stage is only composed of one DSP. In fact, as the decoding algorithm only consists of decoding the received bit stream and calculating the IDCT of the coefficient blocks, the computational load doesn't justify the use of more DSPs. Furthermore, since a QCIF image consists of 144x176x1,5=38016 samples, the 64K memory map of one processor is enough to hold the information corresponding to a complete image. The architecture of this module is depicted in **Figure 7**.

To start with, the received bit stream is handled by a microcontroller, which performs the H.263 frame alignment into 16 bit words, placing them into the FIFO memory. Then, the DSP fetches these words and decodes the received H.263 stream using the VLC decoding algorithm, from which blocks of

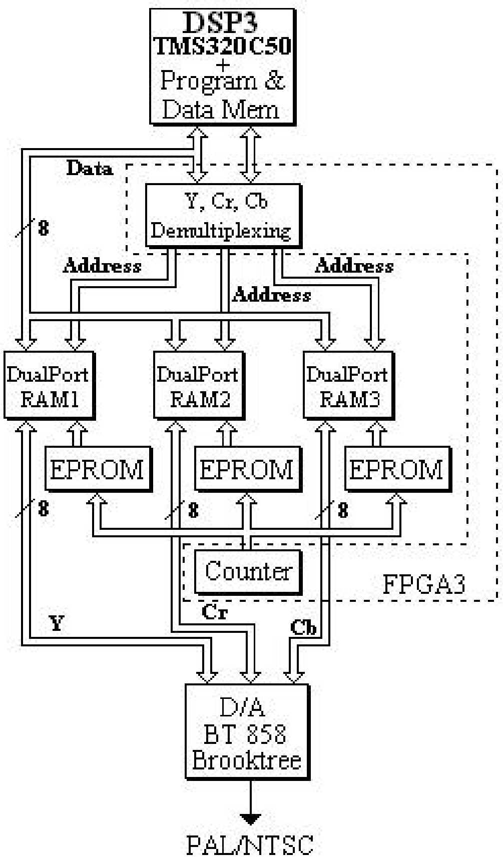

Figure 8 - Formatting functional section of FPGA3

quantized DCT coefficients are obtained. Then, these blocks are dequantized and inverse transformed using a similar algorithm to the one used in the coding stage. Even though the qualification information is not directly available as it was on the coding side, the coded macroblocks contain similar information. After the IDCT, the resulting pixels are transferred to the D/A module (based on Brooktree's Bt858 chip [6]), by means of one of the functional sections implemented in FPGA3. That functional block demultiplexes the Y, Cr and Cb blocks into three separate Dual-Port RAMs. The other functional section transfers the data in the three Dual-Port RAMs to the D/A module and formats the data with the help of three EPROMs (see **Figure 8**).

The D/A module converts the digital Y, Cr and Cb samples into an analog PAL or NTSC video signal.

6. TRANSMITTING THE VIDEO SIGNAL THROUGH THE POWER LINE

Two different sub-systems can be identified in the transmitting module: the transmission sub-system (depicted in **Figure 9**), which modulates the digital data and inserts the modulated signal into the electrical power lines, and the reception sub-system (depicted in **Figure 10**) that recovers the signal received from the electrical power lines and demodulates it.

6.1 TRANSMISSION SUB-SYSTEM

The transmission sub-system receives the bit stream corresponding to the coded image to be sent. In order to protect this bit stream from transmission errors (the electrical power lines are an extremely noisy channel), a Viterbi coding algorithm is applied. The implementation of this algorithm is performed by a Qualcomm's Q1900 integrated circuit [8]. The resulting bit stream modulates a carrier with a frequency around 100kHz. The signal obtained from this modulation is, then, filtered and amplified by a class AB amplifier, which allows for the lowest possible output impedance at the amplifier. The amplified signal is applied to a transformer to provide a galvanic isolation of the circuit. The possible frequencies and signal powers that can be used for signals to be transmitted over the power lines defined in [4] are respected.

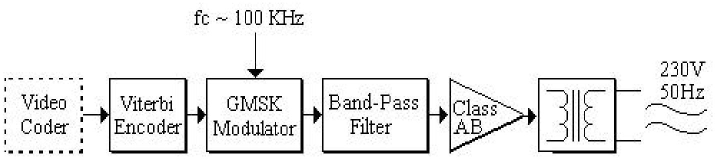

Figura 9 - Transmission sub-system

6.2 RECEPTION SUB-SYSTEM

The reception sub-system performs the opposite process to the one in the transmission sub-system. Therefore, it starts by applying the signal from the power lines to a transformer by means of a high-pass filter implemented with discrete components. The aim of this filter is to reduce the strong 50Hz component allowing the use of higher gain transformers, which has advantages in terms of the received signal power. This received signal is applied to a band-pass filter and an amplifier, in order to separate the signal containing the valid information (around 100kHz) from the residual 50Hz component and amplify it to allow for the demodulator to operate correctly. Afterwards, the signal is demodulated and Viterbi decoded using Qualcomm's Q1900 integrated circuit. The obtained bit stream is applied to the H.263 frame alignment microcontroller.

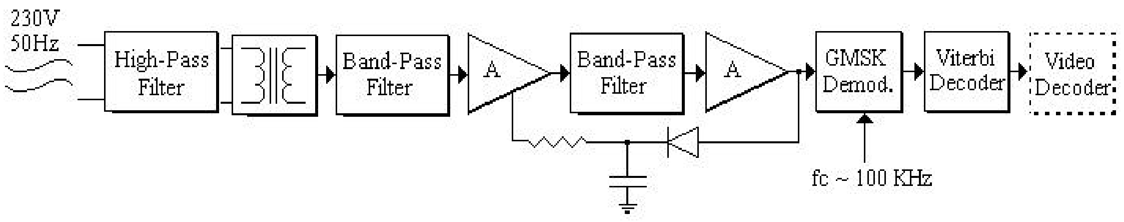

Figure 10 - Reception sub-system

7. CONCLUSION

This paper describes a digital video transmitting system, which uses the power lines as the communication channel. The video codec, based on multiple DSPs and FPGAs, follows the H.263 recommendation.
The system is now being tested and results will be available in the near future. It is expected that it will be possible to transmit video at a rate of about 10 frames per second, without requiring any extra cabling other than the electrical power distribution lines. To increase the frame rate, the coding stage can include more than two DSPs. For that, only some minor changes in the behavior of FPGA1 and FPGA2 have to be made.

REFERENCES

[1] Draft ITU-T Recommendation H.263 - *"Video Coding for Low Bitrate Communication"* - May 1996;

[2] *"TMS320C5x User's Guide"*, Texas Instruments, 1993;

[3] K.R. Rao and P. Yip, "*Discrete Cosine Transform*", Academic Press, 1990;

[4] *"Signalling on low-voltage electrical installations in the frequency range 3kHz to 148,5kHz - European Standard* ", CENELEC - January 1991;

[5] "*Bt812-NTSC/PAL to RGB/YCrCb Decoder*", Brooktree Corporation - February 1996;

[6] "*Bt858-RGB/YCrCb to NTSC/PAL Encoder*", Brooktree Corporation - February 1996;

[7] "*XC4000E and XC4000X Series Field Programmable Gate Arrays*", June 1997, Xilinx Inc.;

[8] "*Q1900 Viterbi/Trellis Decoder*", QUALCOMM Inc.