SMALL FORM FACTOR COGNITIVE RADIO, IMPLEMENTED VIA FPGA PARTIAL RECONFIGURATION, REPLACING A WIRED VIDEO TRANSMISSION SYSTEM

Raúl Torrego (Communications department: IK4-IKERLAN, Arrasate-Mondragón, Spain; rtorrego@ikerlan.es); Iñaki Val (Communications department: IK4-IKERLAN, Arrasate-Mondragón, Spain; ival@ikerlan.es); Eñaut Muxika (Communications and Signal Theory Department: University of Mondragon, Arrasate-Mondragón, Spain; emuxika@eps.mondragon.edu).

ABSTRACT

Nowadays there is a trend towards replacing wired communication systems by wireless ones. When this action has to be performed into industrial environments, special care has to be taken, considering the harsh conditions present in this type of environments. The use of Software Defined Radios or Cognitive Radios fulfils perfectly with both, the wireless characteristic and the securing of the reliable communications this kind of environments requires. Therefore, this paper presents a small form factor cognitive communication system that substitutes a wired video streaming system. The cognitive system senses the availability of the transmission channel and it is able to change its IF (intermediate frequency) working frequency if necessary. It has been implemented on an FPGA, taking advantage of partial reconfiguration to carry out the frequency change, whereas the signal processing algorithms that make up the system have been designed using System Generator: Xilinx's rapid prototyping tool.

1. INTRODUCTION

The reduction, or at least the no-inclusion, of new wires is a common trend nowadays both in the consumer and in the industrial environments. Particularly in industrial environments, or in places such as airplanes, trains or vertical transport devices, where the weight and the installation easiness are appreciated characteristics, the search of technologies that allow a reduction in the number of wires is an ongoing research topic. Software Defined Radios (SDR) [1] or Cognitive Radios [2] are good candidates to carry out this task as the ability they have to get adapted to their surrounding environment allows them to work into harsh environments and perform secure communications. SDRs are defined like communication systems that can change their characteristics "on-the-fly". If the decision of making this change of characteristics is made by the SDR itself, based on the knowledge of the environment status, they are known as Cognitive Radios.

On the other hand, due to the high performance and flexibility they provide, FPGAs are becoming one of the favorite devices when dealing with the implementation of signal processing algorithms. Moreover, dynamic partial reconfiguration, a characteristic that enables an FPGA to change the functionality of certain area while the rest of the device remains working, matches perfectly with the implementation of SDRs or Cognitive Radios.

Furthermore, dealing with the design and implementation of signal processing functions, the use of rapid prototyping tools is highly advisable. These tools enable model based design, allowing the designer, on the one hand, to program the required algorithms in a graphical way, generating synthesizable code automatically. On the other hand, functional simulations are feasible in the early design steps, which in turn leads to huge time savings.

Considering all of the above, this paper presents the implementation of a cognitive video streaming transmission system as a proof-of-concept of the presented design framework. The designed system removes the RS232 wire over which a video streaming is transmitted between two computers, and substitutes it with a cognitive wireless link. The system would emulate a real-world application such as closed-circuit television (CCTV) surveillance. With the digital part fully FPGA implemented, the transmission system is made up of a modulator, an RF section and a demodulator. It implements an OQPSK modulation scheme (used in the IEEE 802.15.4 wireless standard on which other industrial wireless standards such as WirelessHART [3] are based) and uses dynamic partial reconfiguration to perform a transmission frequency change if necessary. The signal processing algorithms involved in the processes of data modulation and demodulation have been designed using System Generator: Xilinx's rapid prototyping tool. Both the transmitter and the receiver are completed with an embedded MicroBlaze soft-processor in charge of managing

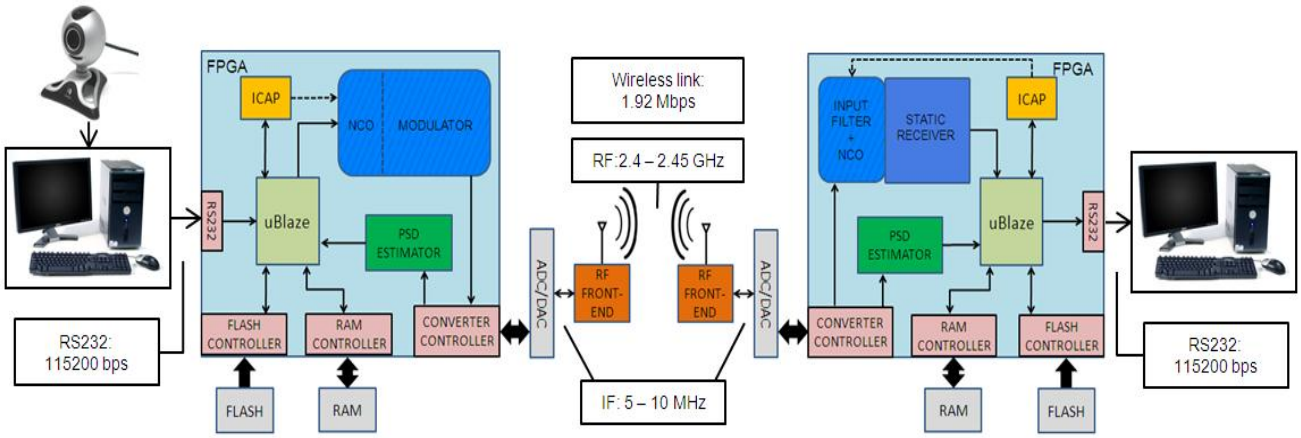

Figure 1: Cognitive video streaming transmission system

the end-point communications via the RS232 wire that has been replaced and carrying out the whole system's control.

The remainder of this paper is organized as follows. Section 2 gives an overview of the implemented system, analyzing the software and hardware resources that have been used. Section 3 introduces System Generator, Xilinx's rapid prototyping tool. In Sections 4 and 5 the different signal processing algorithms that make up the transmitter and the receiver and that have been implemented with System Generator are presented. Then, Section 6 introduces FPGA dynamic partial reconfiguration and describes the design implications that its use has. Numerical results about system performance are presented in Section 7. Finally, concluding remarks are drawn in Section 8.

2. ENVIRONMENT DESCRIPTION AND SYSTEM SETUP

As stated in the introduction, this paper presents a cognitive video streaming transmission system as a proof-of-concept of the feasibility of using small form factor Software Defined Radios or Cognitive Radios in replacement of traditionally wired communication systems. Specifically, the presented system replaces the RS232 wire that holds a video streaming generated by a webcam. Although the RS232 standard is neither the typical one, nor the most suitable one for a video transmission it has been chosen for this implementation due to its simplicity. As will be stated before within the conclusions, further research in this area will consider the replacement of more complex wired transmission systems such as Ethernet.

Figure 1 gives an overview of the implemented system. It is made up of:

- The end-point PCs that generate the video streaming from a Webcam, transmit and receive it through the RS232 interface and reproduce it on a screen.
- The cognitive transmitter. Fully FPGA implemented, receives the video streaming from the source PC, modulates the data with an OQPSK modulation scheme and up-converts it to a 5 or 10 MHz Intermediate Frequency (IF).The choice of this frequency is made based on the availability of the transmission channel. Besides, the implementation of this frequency change is carried out via FPGA dynamic partial reconfiguration.
- The RF front-end. Up/Down converts the modulated signal in IF to the 2.4 GHz RF band and transmits/receives it to/from the air.
- The cognitive receiver. Looks for the signal in the predefined frequencies (5 or 10 MHz), reconfigures itself to the target frequency, performs a frequency and phase synchronization, demodulates the input signal and outputs data over the RS232 interface.

A description of the hardware platforms and specific software programs used for this implementation is presented below.

2.1. Software for streaming video over RS232

Freeware software has been used in order to transmit the video stream generated by the webcam over the RS232 interface. The **AV RS232 Sender** developed by Olds [4] uses unreal media server, live server and player [5] on this purpose. These programs allow the configuration of certain parameters such as the video resolution and frame rate or RS232 baud rate.

The current configuration generates a raw capture of 208x170 pixels and 3 frames-per-second from the webcam (generating an overall throughput of 828 kbps) that is later compressed, with the VC1 (WMV9) codec to 90 kbps in order to fit the maximum available rate of 115200 bauds in the AV RS232 Sender.

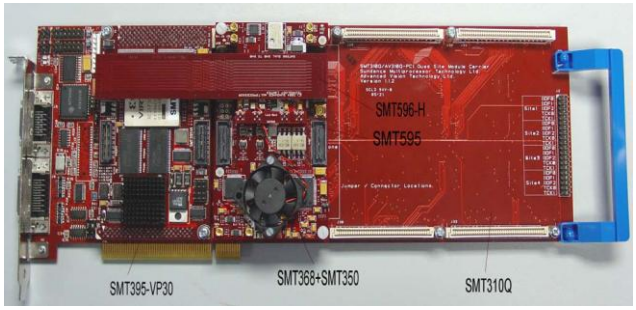

Figure 2: SMT8096

2.2. Hardware Platforms

Two SMT8096 boards [6] have been used to implement the digital part of the presented cognitive transmission system. This hardware platform is a PCI system, based on three main modules that can be seen on Figure 2. The SMT310Q PCI carrier board holds the SMT368 + SMT350 FPGA module with ADC/DAC converters and the SMT395 DSP module containing a TI C6416T DSP. The current implementation only uses the FPGA module and its associated ADC/DAC converters.

A Virtex 4-SX (XCV4SX35) FPGA is the core of the FPGA module. Complementing it, two 8 Mbyte banks of high speed ZBTRAM and a XCF32 Flash PROM are attached to the FPGA. This PROM is in charge of both, configuring the FPGA on start-up and storing the partial bitstreams for the dynamic partial reconfiguration.

Two 14-bit, 125 MHz sampling rate, TI ADS5500 analog-to-digital converters and a dual 16-bit, 500 MHz, DAC5686 digital-to-analog converter make up the SMT350 ADC/DAC expansion board. Additionally, a CDCM7005 programmable clock synchronizer is also present in this board providing low-jitter clock both to the ADC/DAC converters and to the FPGA.

Bearing in mind the IF and data requirements set up for the transmission system, a system clock of 61.44 MHz has been established. This clock is common to the ADC/DAC converters and to the signal processing algorithms implemented in the FPGA. Besides, a 100 MHz clock is also available in the FPGA for auxiliary tasks such as the microblaze system in charge of the RS232 interface and dynamic partial reconfiguration.

2.3. RF Front-end

A Radio Frequency (RF) front-end has also been designed and connected to each of the SMT8096 boards so as to achieve an over-the-air transmission. The front-end up/down converts the IF frequency output from the ADC/DAC converters to a 2.4 GHz RF frequency. It is completed with a pair of self-designed antennas that look for the best performance when working into metallic environments.

In terms of frequency reconfiguration, the design and use of reconfigurable antennas, and reconfigurable matching networks and front-ends has been widely covered in the literature as it is usually a necessary characteristic. However, it is out of the scope of this paper. Due to the small frequency hopping that the proposed transmissions system carries out (5 MHz wide) it can be performed by conventional elements.

3. SYSTEM GENERATOR

As indicated in the introduction, the use of rapid prototyping tools leads to a reduction in the overall design time of data processing systems [7]. These tools are situated at a higher level than the traditional code compilers/synthesizers and improve them by adding characteristics like simulations, graphic programming, automatic code generation or co-simulation. In particular, the digital implementation of both the transmitter and the receiver in the presented design has been carried out with Xilinx's rapid prototyping tool System Generator.

This tool, which works over the Matlab/Simulink design environment, offers all the aforementioned characteristics. It has a block library that is installed like a toolbox in Simulink and that is provided with a large number of blocks that represent and implement the typical functions and components used in the design of digital systems (i.e. logic, gates, relational operators, multiplexers, end even complex IPs like FFTs or microcontrollers). The system programming is carried out drag and dropping these blocks into the design and interconnecting them. Once the design is finished, the tool translates these blocks into synthesizable code for the FPGA (i.e. VHDL or Verilog) and is even able to synthesize and download it to the target FPGA. In addition, on early design steps, this is, prior to the translation into VHDL, System Generator makes it possible to perform functional simulations in order to analyze and adjust the design's behavior. Bearing in mind that this tool is based on Matlab/Simulink, resources in this design environment such as data sources, analysis tools or the mathematical environment are available on this purpose.

Traditionally, rapid prototyping tools have been used – if you will forgive the repetition -, for just prototyping purposes. In other words, the preliminary implementations and design tests to check the feasibility of a certain system were realized using these tools but the final implementations were manually programmed looking for a better performance. However, with the evolution these tools have suffered, nowadays it is possible to completely implement a design using these tools. The system presented in this paper is a proof-of-concept of this affirmation.

The following sections describe the implementation of the transmitter and receiver that has been carried out in System Generator.

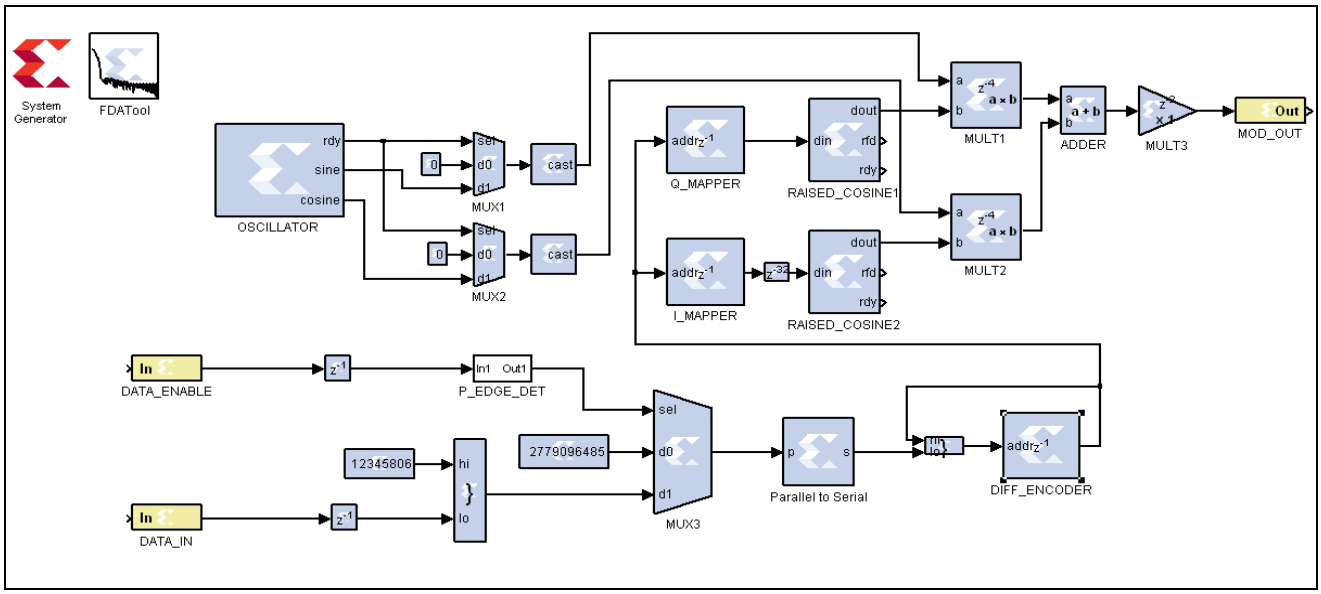

Figure 3: Data acquisition and OQPSK modulator

4. TRANSMITTER

The transmitter is divided into three main data processing tasks: the data acquisition task, the OQPSK modulator itself and the Power Spectral Density (PSD) estimator in charge of the analysis of the transmission channel's availability. A comprehensive description of this transmitter has already been presented in [8], consequently this section will only introduce its main characteristics.

Figure 3 shows the implementation in System Generator of the data acquisition task and OQPSK modulator. The system achieves an overall data rate of 1.92 Mega bits per second, high enough to fulfill with the 115200 bps required by the RS232 link. Comprising the bottom line of the graph, the data acquisition task receives the data from the MicroBlaze processor that monitors the whole system and prepares it for the modulation. In this step, first of all a 3 byte header is attached to the data in order to help its demodulation in the receiver. Subsequently, data passes through a differential encoder that suppresses the phase ambiguity present in OQPSK modulation schemes. In the top line of the figure the OQPSK modulator is implemented. It is mainly composed of the numerically controlled oscillator (NCO) that generates the sine and cosine signals needed for the modulation, the mapper and the 64-tap, 0.25 roll-off raised-cosine filter that generate the I and Q baseband signals and the final multipliers and adder that perform the frequency up-conversion.

As stated before, the system is able to work in two IF frequencies depending on channel's availability, 5 or 10 MHz. An ideal cognitive design would require the transmitter to work in any IF frequency within the limits of the DAC converter, however, looking for simplicity it has been limited to the aforementioned two frequencies. The frequency change is carried out changing the oscillator's settings. With the aim of reducing the amount of resources used by the design and as a demonstration of the feasibility of this technology, this change of settings is performed using dynamic partial reconfiguration.

The PSD estimator shown in Figure 4 is responsible of analyzing the transmission channel. It is present both in the transmitter and the receiver and offers them the necessary information about the signal power presence in the target frequencies. It is implemented with a 64-tap Fast Fourier Transform (FFT) attached to a power detector. The FFT carries out a real time analysis of the power presence in the spectrum. It delivers a 64 byte array that represents the spectral power distribution. Bearing in mind that the ADC converters work at 61.44 MHz, the FFT is able to discern a 1 MHz approximate bandwidth. Subsequently, this data, that is received every millisecond, is averaged to reduce the background noise and passed to the power detector. In the current implementation, this power detector only checks the state of the 5 and 10 MHz frequency bands, although the whole span is available. In case the power presence in any of the frequencies is higher than the maximum assumable by the communication, the corresponding signal is asserted.

5. RECEIVER

The cognitive OQPSK receiver is one of the main contributions of this work. It carries out a power search in the available spectrum and dynamically reconfigures itself to receive the transmitted signal in the corresponding frequency. Afterwards, fine grain frequency and phase

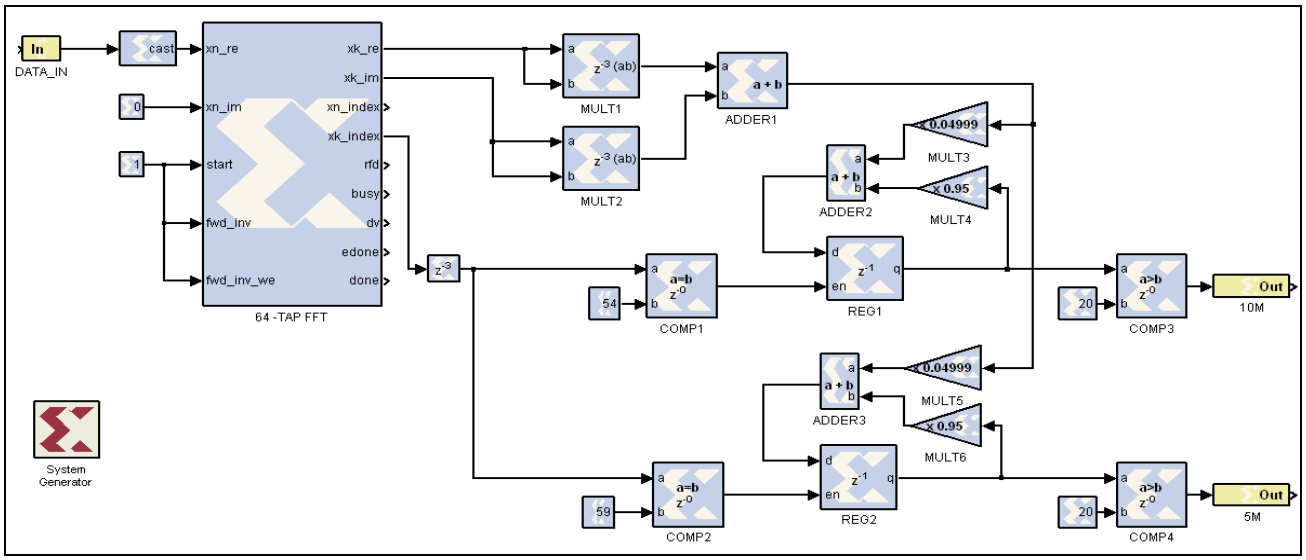

Figure 4: PSD Estimator

synchronization are performed and data is demodulated and submitted to the MicroBlaze in charge of the conversion to the final RS232 interface.

The power search is performed with the aforementioned PSD estimator, while the rest of signal processing functions are detailed below. An overview of the receiver is shown in Figure 5.

5.1. Costas Loop

The core of the receiver is a Costas Loop [9]. This algorithm is a digital PLL used for carrier phase recovery from suppressed-carrier modulation signals. It is made up of an oscillator, a complex multiplier, a phase detector and a loop filter. The algorithm tunes the oscillator until it is phase and frequency locked to the carrier signal. Besides, it also carries out the demodulation of the input signal giving out the baseband In-phase (I) and Quadrature (Q) data components.

Although the Costas Loop is able to track differences between the default frequency of the oscillator and the real incoming carrier frequency, the 5 MHz jump between the two possible IF frequencies is too large. Consequently, in order the Costas Loop to work properly it is necessary to reconfigure the default frequency of the oscillator and adjust certain filtering parameters in the complex multiplier. In the same way as the transmitter, this reconfiguration is carried out using FPGA dynamic partial reconfiguration.

5.2. Early-late gate timing recovery

Due to the structure of the receiver and to the relation between the ADC sampling rate and data rate, a 64x oversampling ratio is present in the demodulated data. Therefore, it is necessary to implement a timing recovery algorithm in order to choose the optimal sampling point. The implemented algorithm is the well known Early-late gate [10]. This algorithm tunes the clock that synchronizes the data to its optimal point with a loop filter. The error signal that drives this loop filter is generated using samples that are early and late compared to the ideal sampling point. When the sampling point is not optimal the early and late samples are at different amplitudes, hence the error signal is generated. Once the timing recovery loop converges, the early and late samples are at equal amplitudes and the sample to be used for later processing is the sample that lies in the middle of them.

5.3. Data recovery

Once the received data is optimally sampled, the source data has to be recovered. As stated in the description of the transmitter, in order to solve certain problems that OQPSK modulation introduces (i.e. phase ambiguity), some data transformations have been carried out. Consequently, the receiver has to undo these transformations. The “OQPSK decoder” and “Data extraction” blocks are in charge of this operation. The first block is a differential decoder that regenerates the original data analyzing, not only the current received symbol, but also the previous one. Later, the “Data extraction” block looks for the header bytes introduced in the transmitter, removes them and outputs the original data and a data enable signal.

5.4. Anti-aliasing filter

Data digitalized by the ADC converter at 61.44 MHz is firstly filtered by a 25-tap anti-aliasing filter when it arrives to the receiver. This filter, which rejects the out of band

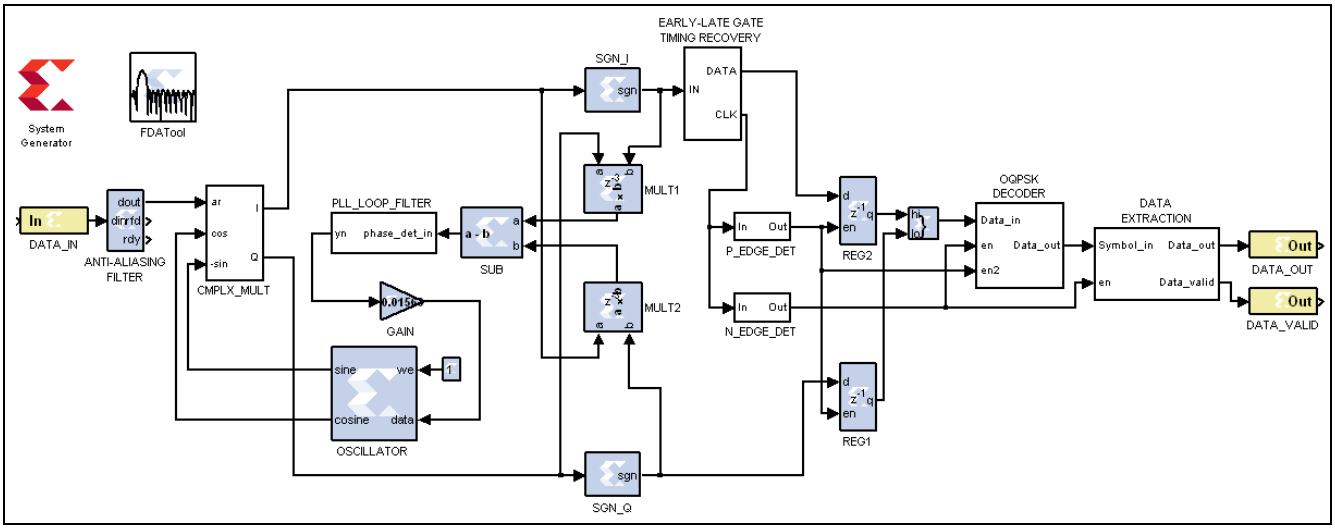

Figure 5: OQPSK demodulator

power in order to improve the performance of the Costas Loop, is quite simple but requites special attention in the proposed implementation. Due to the cognitive characteristic of the system, the pass and stop-band of the filter have to be reconfigured depending on the used IF frequency. Just in the same way as the rest of the reconfigurable components in the system, the reconfiguration of this filter is also carried out with dynamic partial reconfiguration.

6. FPGA DYNAMIC PARTIAL RECONFIGURATION: BASICS AND IMPLEMENTATION ARCHITECTURE

Partial reconfiguration, by definition, is the process in which part of the FPGA changes its configuration while the rest of it remains unaltered [11]. Usually restricted to SRAM-Based FPGAs, this technique is based on the particular configuration structure these devices have. The configuration of the diverse resources within the FPGA (i.e. Look-up-tables (LUT), Input-Output Blocks (IOB), Registers… and so on) is stored in an SRAM memory that can be accessed on runtime. Therefore, updating part of this memory, only the resources controlled by the rewritten part change their functionality.

When the "Dynamic" or "Active" adjective is attached to the definition of partial reconfiguration, the static part of the FPGA not only remains unaltered but also continues working without interruption. This way, the so called "Partial self-reconfiguration" is enabled. In this type of reconfiguration, the static part of the FPGA manages the reconfiguration process so there is no need of external devices such as PCs or processors, hence achieving small-form-factor designs. This is precisely the type of reconfiguration implemented in both the transmitter and the receiver.

Dynamic partial reconfiguration enables a FPGA to implement different functionalities in different time without penalizing the design size, just reusing certain resources. Therefore, it is a promising technology for the implementation of Software Defined Radios or Cognitive Radios. Unfortunately, this technique also has some disadvantages. The main one is the so called "Reconfiguration time". The area being reconfigured is not completely functional until the whole configuration data has been written to the configuration memory. This dead time in which the reconfigurable area is disabled is the "Reconfiguration time". Bearing in mind that the reconfiguration process is based on a memory write, the reconfiguration time depends on the writing speed [12] and on the size of the configuration data, which in turn is proportional to the size of the reconfigurable area. The next section will provide some measurements of this effect.

As stated before, in order to achieve a self-reconfigurable system, both the transmitter and the receiver include a MicroBlaze processor together with the above detailed signal processing functions (Figure 1). This processor is connected to an ICAP port (Internal Configuration Access Port) that enables access to the configuration memory. Besides, other peripherals like the RS232 controller, the memory controllers and the ADC/DAC controllers are also connected to this processor.

When the algorithm running in the MicroBlaze decides that a frequency change is needed, the new configuration data is downloaded from one of the memories of the system to the configuration memory of the FPGA via the ICAP port. The configuration data source can be 'any' of the memories in the system because the storage place of these files

Table 1: Transmitter resource utilization

Transmitter FPGA resource utilization					
	SLICE	Flip-Flop	LUT	BRAM	DSP48
OQPSK Modulator	548 (4%)	644 (2%)	813 (3%)	3 (2%)	70 (36%)
Oscillator	51 (<1%)	61 (<1%)	79 (<1%)	1 (<1%)	0 (0%)
PSD estimator	1547 (10%)	1955 (6%)	2089 (7%)	3 (2%)	32 (17%)
uBlaze system	3074 (20%)	3004 (10%)	3943 (13%)	33 (17%)	3 (2%)
Full design	5432 (35%)	5679 (18%)	6881 (22%)	40 (21%)	105 (55%)
NR design	5531 (36%)	5772 (19%)	6988 (23%)	41 (21%)	105 (55%)

Table 2: Receiver resource utilization

Receiver FPGA resource utilization					
	SLICE	Flip-Flop	LUT	BRAM	DSP48
Filter + oscillator	2183 (14%)	3158 (10%)	1928 (6%)	1 (<1%)	41 (21%)
Static receiver	1037 (7%)	1201 (4%)	982 (3%)	1 (<1%)	3 (2%)
PSD estimator	1547 (10%)	1955 (6%)	2089 (7%)	3 (2%)	32 (17%)
uBlaze system	3074 (20%)	3004 (10%)	3943 (13%)	33 (17%)	3 (2%)
Full design	8091 (53%)	9544 (31%)	9390 (31%)	38 (20%)	79 (41%)
NR design	10324 (67%)	13224 (43%)	11365 (37%)	39 (20%)	120 (63%)

Table 3: Reconfiguration times

Reconfiguration time			
	Partial bitstream size	Bitstream storage	
Design	-	RAM	BRAM
Transmitter (oscillator)	24 KBytes	4,5 ms	3 ms
Transmitter (whole)	267 KBytes	60 ms	-
Receiver (Filter and osc.)	763 Kbytes	171,5 ms	-

depends on their size. When working with small partial bitstream they are stored in the FPGA's internal BRAM memories. These limited resources provide very high speed data access so that the reconfiguration time is minimum. When dealing with bigger partial bitstreams they have to be stored in the external RAM memory that offers a poorer performance. Nevertheless, in either case all the bitstreams are stored in the non-volatile FLASH memory on startup and later copied by MicroBlaze to their final location.

To conclude, a summary of the tasks carried out by the MicroBlaze processors will be presented. Bearing in mind that the programs executed in the transmitter and the receiver are quite similar they will both be presented together. Once the FPGA's initial configuration is automatically loaded on power-up and the MicroBlaze wakes up, the code execution starts. The initialization task first initializes all device drivers (ICAP, memory controllers, RS232, ACD/DAC controllers… and so on) and later copies the partial bitstreams from the FLASH memory to their final storage place. Then, the execution loop starts. This loop is made up of two tasks: the RS232 receive/send task (depending on whether it is the transmitter or the receiver) and the channel monitoring task. The first task is in charge of the communication between the RS232 controller and the signal processing algorithms, sending/receiving data to/from them. The channel monitoring task in turn, receives information from the PSD estimator and decides whether an IF change is necessary or not. In case it is necessary, the relevant actions in order to perform the dynamic partial reconfiguration of the device are carried out.

7. MEASUREMENTS

This section will provide an overview of the most relevant measurements made on the proposed implementation.

As already mentioned, regarding the performance of the communication link, a 1.92 Mbps wireless link has been achieved. This data rate is high enough so as to properly transmit the video streaming at 115.2 Kbps generated by the webcam and the AV RS232 sender.

Table 1 and Table 2 sum up the FPGA resource utilization of the implemented transmitter and receiver respectively. At a first glance the transmitter occupies 5432 SLICES, while the receiver occupies 8091. These values correspond to a 35 % and 53% of the XC4VSX35 Virtex-4 FPGAs in which they have been implemented. Taking into account that this FPGA could be considered as "small", (the biggest FPGA in the Virtex-4 family is nearly six times bigger) the small form factor and simplicity of the presented system is demonstrated. Analyzing the different parts that make up the transmission system, the MicroBlaze constitutes the biggest part of it (20% of the FPGA). This processor, mainly in charge of the management of the dynamic partial reconfiguration, could be considered as an important overhead in relation to the size of the signal processing functions. However, this overhead can be reduced or played down. On the one hand, in a more complex design the presence of a powerful processor like MicroBlaze could be exploited to execute complex software implemented tasks. On the other hand, and targeting small size designs, the management of partial reconfiguration could be ported to a more simple element like a state machine or a PicoBlaze processor hence reducing this overhead.

The bottom rows of the tables present the resource utilization of a Non-Reconfigurable (NR) implementation of the designs. In the case of the transmitter, due to its simplicity and to the only reconfiguration of the oscillator, the use of dynamic partial reconfiguration only reduces the design size by a 2%. In turn, in the receiver, this reduction reaches a 21%, what justifies its use. The above mentioned simplification of the reconfiguration management system would also improve these ratios.

In Table 3 the reconfiguration times of the different designs are shown. As can be observed, two different reconfiguration granularities have been tested for the transmitter in order to evaluate the relation between the size of the reconfigurable area, the size of the partial bitstreams and the reconfiguration time. On the one hand (named, "Transmitter (oscillator)"), only the oscillator is

reconfigured in order to carry out the IF frequency change. On the other hand (named "Transmitter (whole)"), although it is not necessary, the whole OQPSK modulator is erased and reconfigured to achieve the IF change. The measurements obtained from this configuration provide a reference on how long it would take to perform a bigger reconfiguration (i.e. a constellation change or a full communication standard change). The minimum reconfiguration time (3 ms) is achieved when only the oscillator is reconfigured and when the partial bitstream is stored in the internal BRAMs. As stated before, this bitstream location is only possible for small partial bitstreams. With this bitstream stored in a RAM memory, the reconfiguration time rises to 4.5 ms. The reconfiguration of the whole transmitter takes 60 ms and the reconfiguration of the receiver, the longest one, reaches the 171.5 ms. These values are good enough for a video streaming application but may not be sufficient for many other applications.

Calculating the reconfiguration speed from the above data, 8 MBps are achieved when the partial bitstream is stored in BRAM and 4.5 MBps when it is in RAM. This performance is very poor compared to the theoretical maximum writing speed reachable by the ICAP port in Virtex-4 devices of 400 MBps. The responsible of this performance loss is the standard ICAP controller provided with the MicroBlaze processor, which is not as optimized as possible. An enhanced implementation of this controller would offer reconfiguration times around the millisecond.

8. CONCLUSIONS

This paper has presented a cognitive video streaming transmission system that replaces the RS232 wire that holds a video transmission between two computers. Fully FPGA implemented, using dynamic partial reconfiguration to perform the frequency change and being designed with Xilinx's System Generator, a small form factor design has been achieved. The algorithmic complexity of the system may seem basic; however, this paper is mainly focused on demonstrating the feasibility of using cognitive radios implemented via FPGA partial reconfiguration and rapid prototyping tools in the replacement of wired systems in industrial environments and not in the radio features itself. Nevertheless, the achieved performance is good enough for the presented application.

In has been shown that the reconfiguration time is one of the main disadvantages of dynamic partial reconfiguration when used in communication systems like cognitive radios. Consequently, future work will improve the ICAP bandwidth utilization hence enabling the use of the proposed implementation in the replacement of communication systems with hard timing requirements like Ethernet.

ACKNOWLEDGEMENTS

Work partly supported by the Strategic Research Project program of the Government of the Basque County (Spain), through the project MOVITIC, (Etortek program).

9. REFERENCES

[1] Mitola, J.: 'Software radios-survey, critical evaluation and future directions'. Proc. National Telesystems Conference, 1992. NTC-92., pp. 13/15-13/23, 1992

[2]http://www.wirelessinnovation.org/page/Defining_CR_and_DSA, accessed 2012

[3] 'HART Field Communication Protocol Specification, Revision 7.0', 2007

[4] http://homepages.paradise.net.nz/peterfr2/avrs232sender.htm, accessed 2012

[5] http://www.umediaserver.net/umediaserver/download.html, accessed 2012

[6] (SUNDANCE): 'SMT8096 User Manual Version 1.2', 2005

[7] Haessig, D., Hwang, J., Gallagher, S., and Uhm, M.: 'A case study of Xilinx System Generator design flow for rapid development of SDR waveforms'. Proc. SDR Forum Technical Conference, Orange County, 2005

[8] Torrego, R., Val, I., and Muxika, E.: 'OQPSK cognitive modulator fully FPGA-implemented via dynamic partial reconfiguration and rapid prototyping tools'. Proc. 2011 Wireless Innovation Forum European Conference on Communications Technologies and Software Defined Radio (SDR'11 – WInnComm – Europe), Brussels, pp. 142-147, 2011

[9] Shah, S., and Sinha, V.: 'GMSK Demodulator Using Costas Loop for Software-Defined Radio'. Proc. ICACC '09. International Conference on Advanced Computer Control., pp. 757-761, 2009

[10] Zicari, P., Corsonello, P., and Perri, S.: 'A high flexible Early-Late Gate bit synchronizer in FPGA-based software defined radios'. Proc. Circuits and Systems for Communications, 2008. ECCSC 2008. 4th European Conference on, pp. 252-255, 2008

[11] (XILINX): 'UG702 - Partial Reconfiguration User Guide',(2011, 13.3 edn.)

[12] Liu, M., Kuehn, W., Lu, Z., and Jantsch, A.: 'Run-time partial reconfiguration speed investigation and architectural design space exploration'. Proc. FPL 09: 19th International Conference on Field Programmable Logic and Applications, Prague, pp. 498-502, 2009