IMPLEMENTATION OF A 2-FSK CONTINUOUS WAVEFORM USING A SOFTWARE DEFINED RADIO PLATFORM

Jesús García Lledó, Javier Bermejo Parra, Ramón García Gómez
GTSC - ETSIT Universidad Politécnica de Madrid, Spain,
{jesus.garcial, javier.bermejo, ramon.garcia}@upm.es

ABSTRACT

In this paper we describe a "software-based" implementation of a 2-FSK continuous waveform. The base band processing is done in the DSP and the interpolation / decimation, and frequency translation is implemented in the FPGA. The AD and DA interfaces are situated in an intermediate frequency of 70 MHz. For the AD conversion a band pass scheme is used. The DA conversion is a low pass filtering although the signal is digitally modulated at the frequency of 70 MHz. Sliding Goertzel algorithm is used in the 2-FSK receiver as a pass band filter. Symbol Synchronization is based on an Early – Late algorithm. Viterbi Coprocessor available in the DSP is used to implement a convolutional decoder. The architecture and most significant blocks are explained in this paper. Finally some results of the implementation are given.
Keywords: Software Defined Radio, FSK Modulation, DSP, FPGA, Goertzel, ADC, DAC, NCO, DDS.

1. INTRODUCTION

Nowadays the challenge of Software Defined Radio concept is to digitize the signal as early as possible in the receiver while keeping the signal in the digital domain and converting to the analog domain as late as possible in the transmitter, to achieve a high degree of reconfigurability. In some practical designs this solution is not feasible or is very expensive. In these cases an alternative can be making a frequency translation of the signal to an intermediate frequency. Other alternatives are also possible [1]. Therefore, we have worked in the digital domain from baseband to the intermediate frequency of 70 MHz which is also a standard in communications.

The main electronic devices we have used are: a fixed-point DSP of Texas Instruments (TMS320C6416T), a XC4VSX35 Virtex-4 FPGA of Xilinx, a Digital-to-Analog Converter of Texas Instruments (DAC5687) and an Analog-to-Digital Converter of Texas Instruments (ADS5500).

The processing capabilities of these chips exceed the necessary amount for the implementation of the FSK Modem, however we have selected them because other more complex waveforms and signal processing algorithms are under consideration.

In this work we present the design and implementation of a "software based" 2-FSK continuous waveform. The baseband processing is implemented in the DSP. This processing includes functions such as: Scrambler / Descrambler, convolutional encoder / decoder and 2-FSK digital modulator / demodulator. Some remarkable algorithms we have used in the baseband demodulator are the sliding Goertzel algorithm, which works as a band-pass filter, and an early-late timing recovery block. On the other hand, the FPGA is used to implement the up and down conversion. In this process we can distinguish three blocks: the interpolation / decimation stages, mixers and local oscillators.

We have implemented the interpolation / decimation stages with CIC (Cascaded Integrator-Comb) filters. For large rate changes, this kind of filter has a significant advantage over a FIR filter with respect to architectural and computational efficiency, because they don't need to use multipliers. Regarding to the mixer stage, both transmitter and receiver, use a complex multiplier and a NCO (Numerically Controlled Oscillator) as local oscillator. The data conversion module is equipped with a 125MSPS 14-bit single channel ADC and a 500 MSPS 16-bit dual channel interpolating DAC. The DAC also includes programmable gains at the input and output, which can be used to implement a transmission power control.

In the paper we present the design and implementation of each one of the blocks of the previously commented architecture. Different considerations about reconfigurable parameters, the computational burden and the FPGA utilization are discussed.

Finally, some of the implementation results, both simulation and real-time execution, are presented, and for future work we describe some possibilities to achieve a software defined radio with a higher degree of reconfigurability.

2. ARCHITECTURE

In figures 1 and 2, block diagrams of the transmitter and receiver of the 2-FSK modem are shown. As can be seen, it was decided to use the FPGA to perform the stages of channelization (interpolation in case of transmitter and decimation in the receiver one), and frequency modulation

(from Baseband to Intermediate Frequency in transmitter and the inverse operation in the receiver). Therefore, DSP is responsible for implementing the blocks corresponding to the binary processing and digital baseband modulator – demodulator.

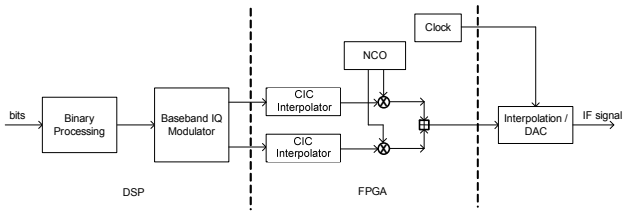

Fig. 1. Transmitter diagram.

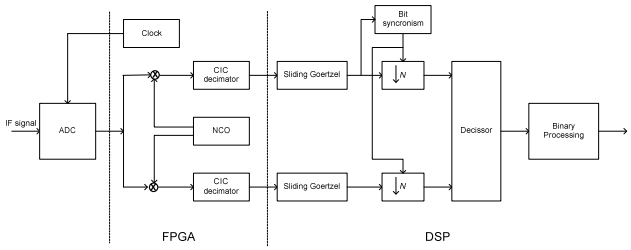

Fig. 2. Receiver diagram.

2.1. Binary Processing

This block would include operations such as source coding, encryption, scrambling, channel coding… In the case of this modem it has been selected an auto-synchronized randomizer and a convolutional encoder (2, 1, 7), with 2 as the number of output bits, 1 as the number of input bits and 7 as the number of cells in the shift register (*constraint length*). The connection vectors in octal code are 131 and 171.

To perform the convolutional decoder, the Viterbi algorithm is implemented using the Viterbi Coprocessor available in the DSP.

2.2. Digital Baseband Modulator

Digital 2-FSK modulator [2] consists of a module that matches each bit with its corresponding digital waveform. As this is a 2-FSK modulation, a two-symbol alphabet is used and only one bit is needed to code each symbol. Bit '0' is matched with waveform $\tilde{g}_0(t)$, and waveform $\tilde{g}_1(t)$ is assigned to bit '1'. Being $\tilde{g}_0(t)$, $\tilde{g}_1(t)$, two windowed sinusoids of frequencies w_0, w_1 respectively.

In this way, concatenating previous waveforms, the 2-FSK Band Pass signal is formed. As the modulator is implemented in digital, it is only necessary to store in memory the samples corresponding to the waveforms of each symbol, i.e. $\tilde{g}_0(t)$ and $\tilde{g}_1(t)$. The problem of generating the 2-FSK signal by this procedure is that the number of samples to be stored may be quite huge if the frequencies w_0, w_1 are very high.

Due to the mentioned problem, it was chosen to implement a 2-FSK Baseband Modulator in the DSP and then the FPGA will perform an interpolation and a frequency translation from baseband to the frequency band desired.

Baseband pulses for each symbol have now the following expressions:

$$g_0(t) = W(t) \cdot \exp\left(j \cdot \left((w_0 - w_c)t - \pi/2\right)\right) \quad (1)$$

$$g_1(t) = W(t) \cdot \exp\left(j \cdot \left((w_1 - w_c)t - \pi/2\right)\right) \quad (2)$$

It should be kept in mind that the pulses $g_0(t)$, $g_1(t)$ are complex signals, and therefore, two arrays for each waveform are required to store their samples, one for the real part and another one for the imaginary part. The main advantage is that the number of samples of each array N_{samp_BB} is much less than in the case of the band-pass modulator. It was used 16 samples per symbol.

In Figure 3 seven symbols of a 2-FSK baseband signal are shown.

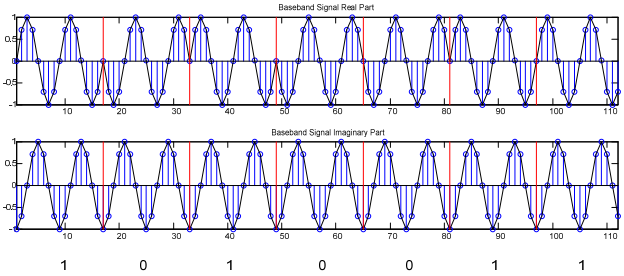

Fig. 3. Baseband 2-FSK signal.

2.3. Up Conversion

2.3.1. CIC interpolator

Once generated the 2-FSK baseband signal, the next stage is interpolating the signal to increase their sampling rate. Interpolation is performed in the FPGA using two CIC Filters (Cascaded Integrator and Comb) [3], one for the real part of the signal and the other for the imaginary part. Scheme of the used CIC Interpolator filter is shown in Fig.4.

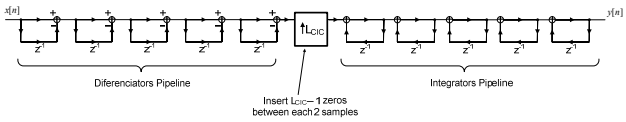

Fig. 4. CIC Interpolator Filter.

The main advantage of CIC filters is that they do not use multipliers, so they are very efficient in terms of computational cost.

Due to instability of the integrator filter, it is necessary to increase the number of bits of the datapath. The minimum number of bits required at the output of the integrator stage is given by following expression [3]:

$$B_{OUT} = \lceil B_{IN} + N \cdot \log_2(R) \rceil \quad (3)$$

being B_{IN}, B_{OUT} the number of input and output bits respectively, N the number of integrator filters and R the interpolation factor. In our case $N=5$ and $R=24$. In the comb stage it is also required one additional bit per differentiator. Consequently it has been chosen a 64 bits width datapath.

As mentioned above, the system has been implemented and developed in a XC4VSX35-10 Xilinx FPGA.

Synthesize and implementation results are shown in the following Tables:

Device Utilization Summary		
Slices	2293 out of 15360	14%
Flip Flops	2070 out of 30720	6%
4 Input LUTs	3140 out of 30720	10%
IOBs	140 out of 448	31%
GCLKs	1 out of 32	3%

Table 1. Device resource utilization summary.

Timing Summary	
Minimum period	6.435 ns
Maximum frequency	155.412 MHz
Minimum input arrival time before clock	7.350 ns
Maximum output required time after clock	7.358 ns
Maximum combinational path delay	7.906 ns

Table 2. Timing summary and maximum operating frequency.

In the next figure it is represented the output of the CIC interpolator filter, obtained with the Real-Time platform, using the Texas Instruments tool Code Composer Studio.

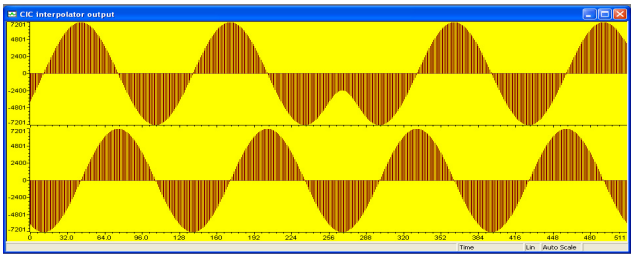

Fig. 5. CIC Interpolator Filter Output Signal (Real and Imaginary parts).

Finally, baseband signal is translated to the intermediate frequency of 70 MHz, simply being multiplied by a complex exponential and then the real part is taken.

2.3.2. Mixers and NCOs

Mixers are used to modulate or demodulate the signal with the carrier frequency. In the transmitter, mixer modulates the interpolated baseband signal with the carrier generated by an NCO (Numerically Controlled Oscillator) programmed with the value of the desired carrier frequency.

On the other hand, in the receiver, mixer demodulates the signal provided by the ADC with the carrier provided by another NCO configured with the specific value of frequency.

The transmitter clock is not known exactly at the receiver, in spite of that, we have enough precision (10^{-6} ppm) to do the bit synchronization with a first order PLL. In our application the carrier frequency is considered to be low enough that does not increase the error rate of the receiver. Consequently a carrier frequency tracking loop was not implemented.

The block denoted as MIXER is composed by two IP cores, one of them implements a NCO based on DDS and the other performs the complex multiplier.

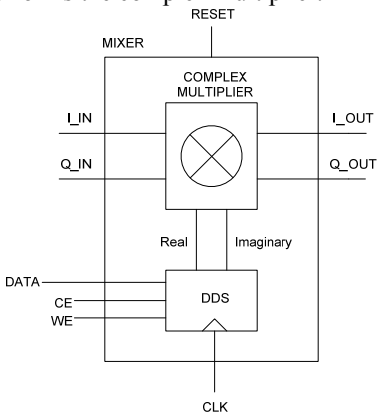

Fig. 6. Block Diagram of the Mixers.

As mentioned above, NCOs have been implemented with a core of a Direct Digital Synthesizer based on DDS techniques. This core has been created and configured using the Xilinx tool Coregen. Besides, the Complex Multipliers have been performed with another core that implements a multiplier of complex data. The block has been programmed using Coregen. Synthesize and implementation results are shown in the following Tables:

Device Utilization Summary		
Slices	167 out of 15360	1%
Flip Flops	266 out of 30720	0%
4 Input LUTs	193 out of 30720	0%
IOBs	95 out of 448	21%
FIFO16 / RAMB16	8 out of 192	4%
GCLKs	1 out of 32	3%
DSP48s	4 out of 192	2%

Table 3. Device resource utilization summary.

Timing Summary	
Minimum period	4.214 ns
Maximum frequency	237.290 MHz
Minimum input arrival time before clock	5.541 ns
Maximum output required time after clock	5.280 ns
Maximum combinational path delay	No path found

Table 4. Timing summary and maximum operating frequency.

Figure 7 shows the 2-FSK signal that would result after the whole process discussed above. It can be observed that band pass signal has continuous phase despite of 2-FSK baseband signal having phase jumps of 180º in the real part. The FFT magnitude of 2-FSK modulated signal is represented in figure 8.

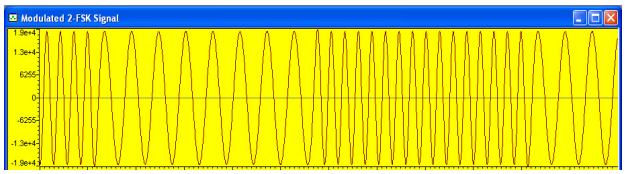

Fig. 7. 2-FSK modulated signal.

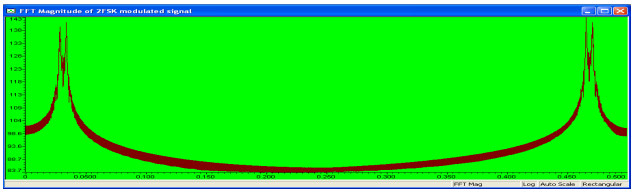

Fig. 8. FFT magnitude of 2-FSK modulated signal.

2.4. Down Conversion

The first stage of 2-FSK demodulator consists of translating the 2-FSK signal from intermediate frequency to baseband. For achieving this, the digital band pass signal is multiplied by a complex exponential of the same frequency (but opposite sign) as the one used in the transmitter for the frequency modulation.

This process is implemented in the FPGA in the same way as the Up Conversion.

Afterwards, it would be necessary to low pass filter the resulting signal, but it would be taken advantage of the fact that CIC filter is a low pass filter to avoid that filtering stage. Therefore, decimation CIC filter will perform the low pass filtering, while reducing the sampling rate of the signal to get only L=16 samples per symbol. The CIC filter designed is shown in Figure 9:

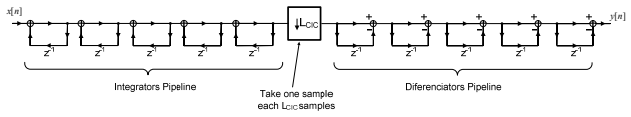

Fig. 9. CIC Decimator Filter.

According to the expression (3), it has been used a 64 bits width datapath for the whole CIC Decimator filter.
Synthesize and implementation results for this specific FPGA are shown in Tables 5 and 6:

Device Utilization Summary		
Slices	2988 out of 15360	19%
Flip Flops	3198 out of 30720	10%
4 Input LUTs	3552 out of 30720	11%
IOBs	140 out of 448	31%
GCLKs	2 out of 32	6%

Table 5. Device resource utilization summary.

Timing Summary	
Minimum period	6.393 ns
Maximum frequency	156.431 MHz
Minimum input arrival time before clock	5.769 ns
Maximum output required time after clock	7.653 ns
Maximum combinational path delay	7.906 ns

Table 6. Timing summary and maximum operating frequency.

In the next figure it is represented the output of the CIC decimator filter, obtained with the Real-Time platform, using the Texas Instruments tool Code Composer Studio.

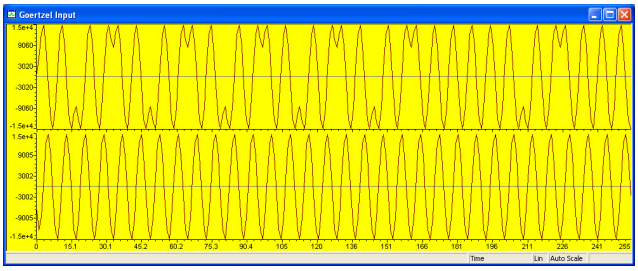

Fig. 10. CIC Decimator Filter Output Signal (Real and Imaginary parts).

2.5. Digital Baseband Demodulator

The first block of the digital baseband demodulator is the Goertzel algorithm module [4] [5] [6]. This algorithm is used to calculate efficiently the Discrete Fourier Transform (DFT) at one specific frequency.

It basically helps us to detect the tones corresponding to the symbols of the 2-FSK signal, i.e. it performs a band pass filtering.

Figure 11 presents a scheme to describe the Goertzel algorithm [4]. N is the number of samples of the DFT and $W_N = e^{-j\frac{2\pi}{N}}$.

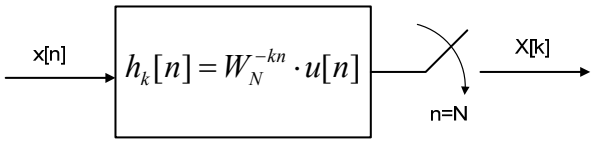

Fig. 11. Scheme corresponding to Goertzel Algorithm.

We have implemented the Sliding Goertzel Algorithm, which is a variant of the Goertzel Algorithm that allows computing efficiently the DFT of a signal at a specific frequency while samples arrive at the block. Instead of calculating the Goertzel algorithm of Figure 11 to blocks of N samples, which differ only in one sample from one block to the next, new algorithm calculates the Goertzel algorithm of a group of a N samples using the Goertzel value obtained in the previous block.

Figure 12 displays two consecutive time windows of the signal x[n], which will be the input of the Goertzel module.

The sequence $S_{t-1}[n]$ corresponds to the window that includes the samples n=0,1,…N-1 of the signal x[n]. Sequence $S_t[n]$ corresponds to the window which includes the samples n=1,2,…N of the original signal x[n], i.e. it represents the situation when a new sample of x[n] has just arrived for processing.

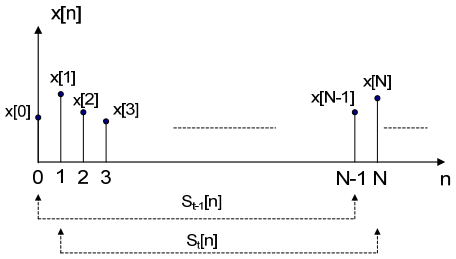

Fig. 12. Sequence x[n] with two consecutive time windows.

The relationship between the DFT of a window and the next one is given by equation 4, which is finally programmed in the DSP.

$$S_t[K] = e^{j\frac{2\pi}{N}K} \cdot \left(S_{t-1}[K] + x[N] - x[0]\right) \qquad (4)$$

In figure 13 the programmed flow graph is shown:

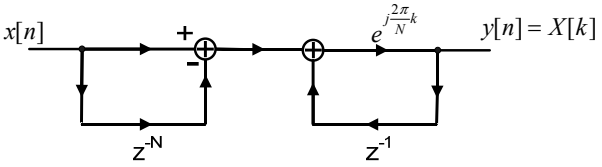

Fig. 13. Flow graph corresponding to Sliding Goertzel Algorithm.

A filter as the one represented in Figure 13 has been implemented to compute the Discrete Fourier Transform (DFT) at frequency f_0 corresponding to symbol defined as '0', and another analogous filter, to calculate the DFT at frequency f_1 corresponding to symbol defined as '1' was also implemented.

Figure 14 presents the output provided by the sliding Goertzel algorithm for the case of f_0. At the bottom of the figure appears the 2-FSK signal after having been converted to baseband and decimated, i.e. the signal the sliding Goertzel algorithm is applied to. Notice that the output of the Goertzel Algorithm is maximum when the input symbol is '0', as it corresponds to the case when the algorithm seeks the frequency f_0, and minimum when the input symbol is'1'.

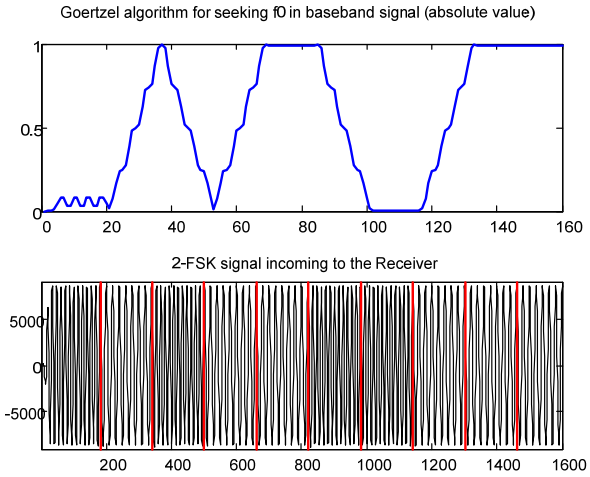

Fig. 14. Digital 2-FSK Demodulator.

In Figure 15 the outputs provided by the sliding Goertzel algorithm for the case of f_0 and f_1 are displayed. Those are complementary curves, namely when one is maximum the other is minimum and vice versa.

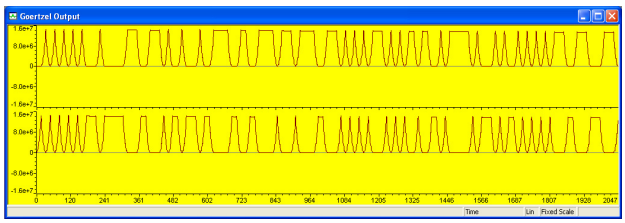

Fig. 15. Comparative time diagram of Goertzel at frequencies f_0 y f_1.

Then, after the Goertzel algorithm, a timing recovery algorithm or symbol synchronism is implemented to select only one sample per symbol and also to look for the best sampling instant. To achieve this, the timing recovery algorithm is based on an Early – Late mechanism.

Sampling phase instant correction is done dynamically, as samples reach the receiver. The target would be sampling at the top of the curve generated by the Goertzel algorithm, where outputs of the filters tuned at f_0 and f_1 differ more, and so it will be easier to take a correct decision. The decision process simply compares the output samples of the Goertzel module at f_0 with output samples at f_1 and decides the received symbol based on whichever is greater.

2.6. FIFOs

Two FIFOs, one for the transmitter and another for the receiver, have been implemented in the FPGA using a block RAM IP Core. These FIFOs interface between the DSP and the FPGA. The parameters of the FIFOs are: 2048 positions depth, 32 bits width.

2.7. Clock Domains

The clock signal is generated in the FPGA with a DCM (Digital Clock Manager) from a crystal oscillator of 16.384 MHz. The value of the synthesized frequency is 98.304 MHz. This signal is used in the following blocks: DAC, AD Mixers, and CIC filters.

FIFOs previously commented work with a lower sample rate clocks which are generated in the CIC filters and depend on the interpolation / decimation factors.

2.8. Data Converters

The real-time platform used, provides two data converters of Texas Instruments, the Digital to Analog Converter DAC5687 and the Analog to Digital Converter ADS5500.

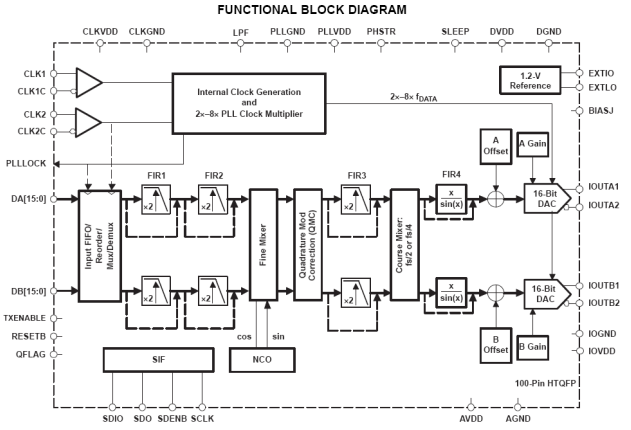

Fig. 16. Functional Block Diagram of the DAC5687.

The chip DAC5687 is a 16-bit, 500 MSPS (2x-8x) interpolating dual-channel DAC. Besides, it includes a complex mixer with 32-Bit NCO. This DAC provides two digital data channels of 16 bits and two analog output channels. The highest data rate is 500 MSPS. It includes a digital quadrature modulation correction (QMC), sinc correction and gain control. It has several memory registers inside for configuring the different operation modes. The interface for programming the registers is the serial interface SPI. The ADS5500 is a 14-bit, 125MSPS single-channel ADC. It can work in two data formats (Straight Binary and 2's Complement), and with two active edges (rising and falling). The interface for programming the registers is the serial interface SPI.

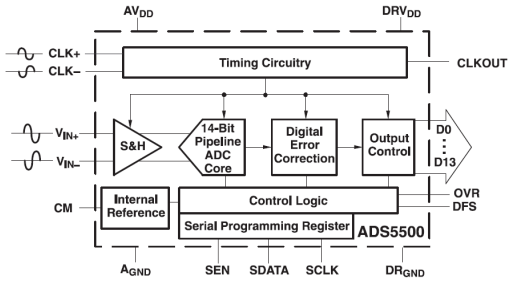

Fig. 17. Functional Block Diagram of the ADS5500.

3. CONCLUSIONS

In this paper we present the design and implementation of a 2-FSK continuous waveform based on what nowadays is known as Software Defined Radio design philosophy. We explain the chosen architecture, firstly mentioning the main electronics devices we have used and afterwards describing the different blocks that have been implemented. Apart from the description of each block we also discuss about the computational burden of the algorithm and we present some results that we have obtained, mainly using the Code-Composer Tool for the DSP device.

4. ACKNOWLEDGEMENTS

Authors would like to thank Signal Processing on Communications Group (GTSC) team members for their valuable help and feedback.

5. REFERENCES

[1] J. H. Reed, Software Radio: a modern approach to radio engineering.Prentice Hall. Communications Engineering and Emerging Technologies Series. 2002.

[2] Edward A. Lee, David G. Messerschmitt, *Digital Communication* 2ª Ed. Kluwer Academic Publishers 1994.

[3] Frederic Harris, *Multirate Signal Processing for communication systems*, Prentice Hall 2004.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall Signal Processing Series, Einglew ood Cliffs, New Jersey: Prentice Hall, 1983.

[5] K. Banks, "The Goertzel algorithm" , Embedded Syst. Programming Mag., pp. 34-42, Sept. 2002.

[6] Jacobsen, E.Lyons, Signal Processing Magazine, IEEE Volume 20, Issue 2, Mar 2003 Pages 74 – 80.

[7] J.García, J.Bermejo, "Design and Implementation of a 2-FSK Software Radio Modem", URSI Madrid 2008