Design of a synchronous FFHSS modulator on a FPGA with System Generator

SANTIAGO T. PÉREZ, JESÚS B. ALONSO, CARLOS M. TRAVIESO, MIGUEL A. FERRER, JOSÉ F. CRUZ
Signals and Communications Department
University of Las Palmas de Gran Canaria
Campus University of Tafira, 35017, Las Palmas de Gran Canaria, Las Palmas
SPAIN
{sperez, jalonso, ctravieso, mferrer}@dsc.ulpgc.es, jcruz@pas.ulpgc.es

Abstract: -This work is based on a previous Fast Frequency Hopping Spread Spectrum (FFHSS) transceiver designed for wireless optical communications. The core of the transmitter was a discrete Direct Digital Synthesizer (DDS). In the first prototype the DDS control and the digital synchronization signal were generated using a Programmable Logic Device (PLD), besides a discrete external filter was necessary to eliminate high frequency components of digital synchronization signal and generate analog synchronization signal. The FFHSS and analog synchronization signals were emitted by two separated Light Emitting Diodes (LED) to avoid adding them with discrete analog circuits. The peak emission wavelength of the LED was 650 nanometers. The objective of this work is redesign the modulator on a Field Programmable Gate Array (FPGA) using System Generator from Xilinx and analyse the performances of this design methodology. System Generator is one design tool in Simulink of Matlab. It allows fast design of digital systems using block diagrams. The compilation generates the files necessary for the Integrated System Environment (ISE) for FPGA of Xilinx, where the description of the circuit is obtained in a standard hardware description language. In ISE it is possible to compile the hardware description language files, simulate the system, assign pins, obtain the program file and program the FPGA.

Key-Words: - spread spectrum, modulator, DDS, FPGA, System Generator, Xilinx, Simulink, HDL, floating point, fixed point.

1 Introduction

A Field Programmable Gate Array (FPGA) [1] [2] [3] is a digital integrated circuit that is configured by the designer from a personal computer, without sending it to the manufacturer, it can be reprogrammed even if it has been placed in the printing circuit board. The FPGA is programmed by downloading a file called bitstream to an internal or external memory in the system. A FPGA consists in a two dimension array of digital configurable resources which can perform arithmetic and logic functions. Data dimensions in FPGA are flexible and highly parallel architectures can be built, for these reasons FPGA are suitable for Digital Signal Processing (DSP).

The Xilinx company is one of the most extended manufacturer of FPGA. Xilinx offers System Generator [4] that is one design tool in Simulink of Matlab. System Generator allows the fast design of systems using block diagrams, and its simulation even before the compilation. The compilation generates the files necessary for the Integrated System Environment (ISE) [5] of Xilinx for FPGA, where the description of the circuit is obtained in a standard hardware description language (HDL). These languages are Verilog [6] and Very High Speed Integrated Circuit Hardware Description Language (VHDL) [7]. In ISE it is possible to compile the hardware description language files, and simulate the system behavioral or timing analysis. Afterwards the program file can be generated for the chosen device, this file can be downloaded from the computer to the board where the FPGA is included. Finally, the performance of the design system must be checked with electronic measure equipment.

This work is based on a previous designed transceiver [8] for Fast Frequency Hopping Spread Spectrum (FFHSS) [9] [10] for wireless optical communications [11]. The core of the transmitter was a discrete Direct Digital Synthesizer (DDS) [12] from Analog Devices [13]. The objective of this work is redesign the FFHSS modulator with System Generator and analyse the performances of

this design methodology: the integration and flexibility are improved, the design time is minimizing and the system has less power consumption.

2 Design methodology

System Generator version 10.1 was used in this design. When System Generator is installed some Blocksets (see Fig. 1) are included in Simulink of Matlab (version R2007a). Each block is configured opening its dialog window, this permits fast and flexible designs.

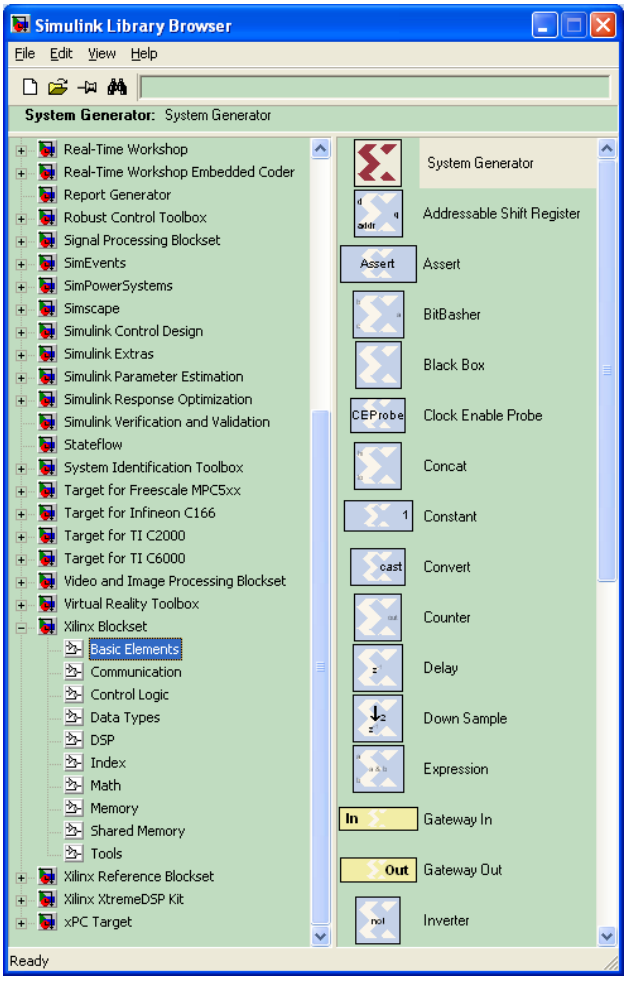

Fig. 1. System Generator Blocksets in Simulink.

The FPGA boundary from the Simulink simulation model is defined by *Gateway In* and *Gateway Out* blocks. The *Gateway In* block converts the floating point input to fixed point format, saturation and rounding modes can be defined by the designer. The *Gateway Out* block converts the FPGA fixed point format to Simulink double numerical precision.

In System Generator the designer does not perceive the signals as bits, instead the bits are grouped in signed or unsigned fixed point format. The operators force signals to change automatically to the appropriate format in the outputs. A block is not a hardware circuit necessarily, it relates with others blocks to generate the appropriate hardware. The designer can include blocks described in a hardware description language, finite state machine flow diagram, Matlab files, etc. The System Generator simulations are bit and cycle accurate, this means results seen in simulation exactly match the results seen in hardware. The Simulink signals are shown as floating point values, which makes easier to interpret them. The System Generator simulations are faster than traditional hardware description language simulators, and the results are easier to analyze. Otherwise the VHDL and Verilog code are not portable to others FPGA manufacturers, the reason of this is that System Generator uses Xilinx primitives which take advantages of the device characteristics.

System Generator can be used for algorithm exploration or design prototyping, for estimating the hardware cost and performance of the design. Other possibility is using System Generator for designing a portion of a big system and join with the rest of the design. Finally, System Generator can implement a complete design in a hardware description language. Designs in System Generator are discrete time systems, the signals and blocks generate automatically the sample rate, however a few blocks set the sample rate implicitly or explicitly. System Generator supports multirates circuits and some blocks can be used for changing the sample rate.

Often an executable specification file is created using the standard Simulink Blocksets (see Fig. 2). The specification file can be designed using floating point numerical precision and not hardware detail. Once the functionality and basic dataflow have been defined, System Generator can be used to specify the hardware implementation details for the Xilinx devices. System Generator uses the Xilinx DSP Blockset from Simulink and will automatically invoke Xilinx Core Generator to generate highly optimized netlists for the building blocks. System Generator can execute all the downstream implementation tools to get a bitstream file for programming the FPGA device. An optional testbench can be created using test vectors extracted from the Simulink environment for using with ISE simulators.

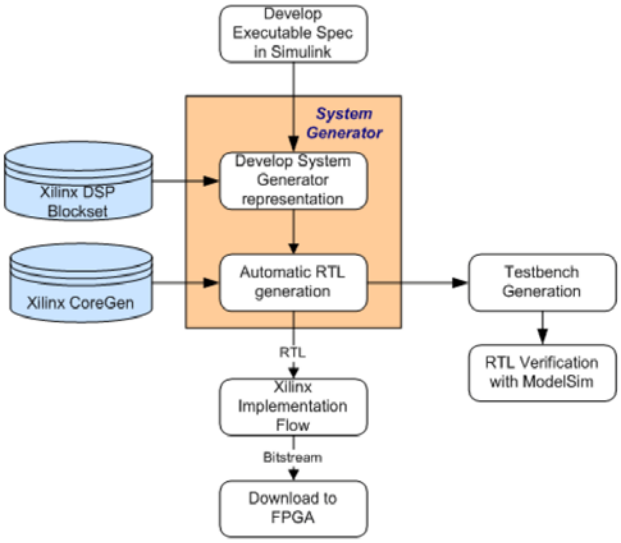

Fig. 2. System Generator design flow.

Every system designed with System Generator must contain a *System Generator block* (see Fig. 3) and it specifies how simulation and code generator can be used.

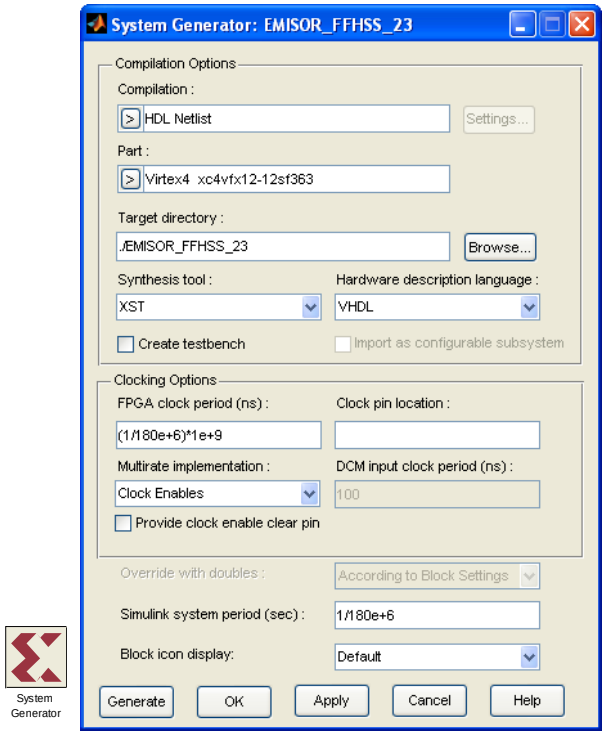

Fig. 3. *System Generator block* and its dialog window.

Firstly, in *System Generator block* the type of compilation can be specified to obtain: HDL netlist, Bitstream for programming, etc. Secondly, the FPGA type can be chosen. The *Target directory* defines where the compilation writes the files of ISE project. The *Synthesis tool* specifies which tool is chosen for synthesizing the circuit: Synplify, Synplify Pro [14] or Xilinx Synthesis Tool (XST). In *Hardware description language* the designer can choose between VHDL or Verilog. Finally, *Clock Options* defines the period of the clock, its input pin location, the mode of multirate implementation and the Simulink system period which is the greatest common divisor of the sample periods that appear in the system. In *Block icon display* is specified the type of information to be displayed on the block icon.

When the designer clicks on *Generate* in dialog window of *System Generator block* the structural description files in a hardware description language are obtained, and a project is created for ISE version 10.1. In ISE, it is possible to check the syntax of the hardware description language files (see Fig. 4). The first step in the compilation process is synthesizing the system. The synthesis tool used is XST, it is an application that synthesizes hardware description language designs to create Xilinx specific netlist files called NGC files. The NGC file is a netlist that contains both logical design data and constraints. The NGC file takes the place of both Electronic Data Interchange Format (EDIF) and Netlist Constraints File (NCF) files. In synthesis options optimization goal for area or speed can be fixed; by default, this property is set to speed optimization. Similarly, optimization effort can be established as normal or high effort, in last case additional optimizations are performed to get best result for the target FPGA device. Synthesis report can be analyzed by the designer; moreover, the designer can view Register Transfer Level (RTL) schematic or technology schematic. After synthesizing the system the design is implemented in three stages: translate, map and place and route. The translation process merges all the input netlists and design constraint information and outputs a Xilinx Native Generic Database (NGD) file. Then the output NGD file can be mapped to the targeted FPGA device family. The map process takes the NGD file, runs a design rule checker and maps the logic design to a Xilinx FPGA device. The results appears in a Native Circuit Design (NCD) file, which is used for placing and routing. The place and route process takes a NCD file and produces a new NCD file to be used by the programming file generator. The generate programming file process runs the Xilinx bitstream generation program BitGen to produce a bit file for Xilinx device configuration. Finally, the configure target device process uses the bit file to configure the FPGA target device.

Behavioral simulations are possible in the design before synthesis with the simulate behavioral model process. This first pass simulation is typically

performed to verify RTL or behavioral code and to confirm the designed function. Otherwise, after placed and routed the design on the chip, timing simulations are possible. This process uses the post place and route simulation model and a Standard Delay Format (SDF) file. The SDF file contains true timing delay information of the design.

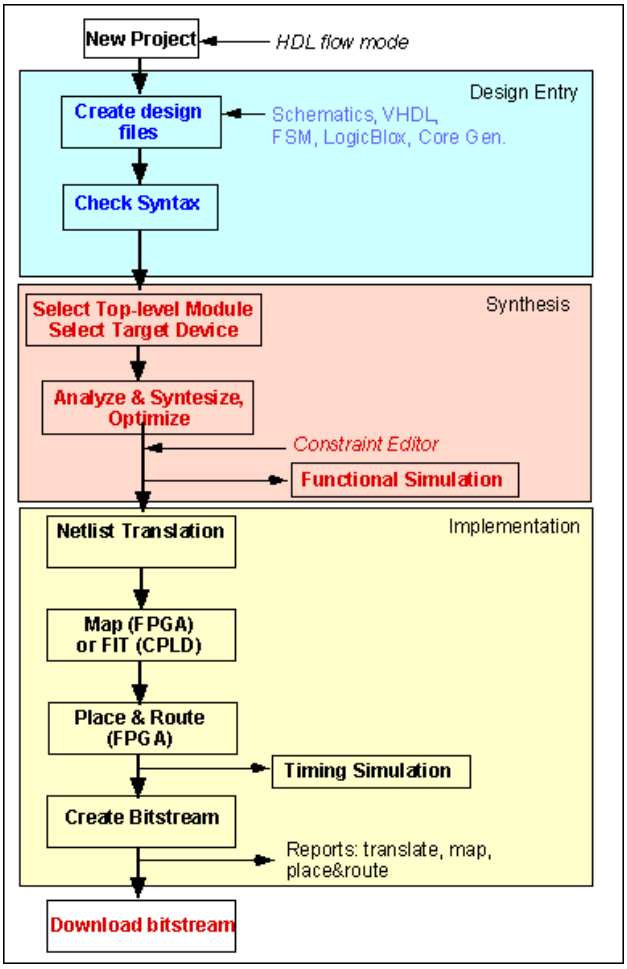

Fig. 4. Overview of design flow of Integrated System Environment.

3 Previous designed FFHSS transceiver

In this section the previous FFHSS transceiver (see Fig. 5 and 6) for wireless optical communications is described and its block diagram is shown.

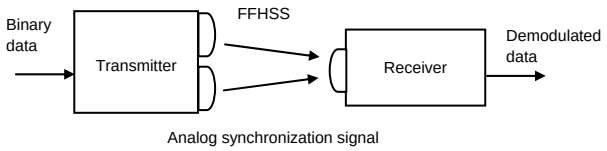

Fig. 5. Block diagram of previous designed FFHSS transceiver.

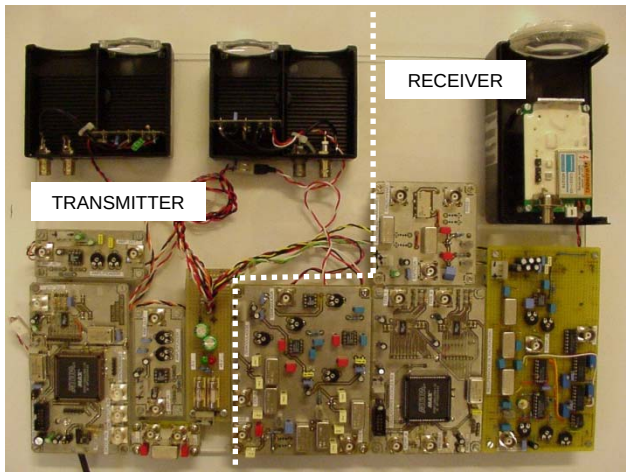

Fig. 6. Hardware of previous designed FFHSS transceiver.

3.1 Designed transmitter

In the first prototype (see Fig. 7) the DDS control signals and digital synchronization signal were generated using a Programmable Logic Device (PLD). Besides an external discrete filter was necessary to eliminate the high frequency components of digital synchronization signal and get an analog synchronization signal. The FFHSS and synchronization signals were emitted by two separated optical emitters to avoid summing them with discrete analog circuits.

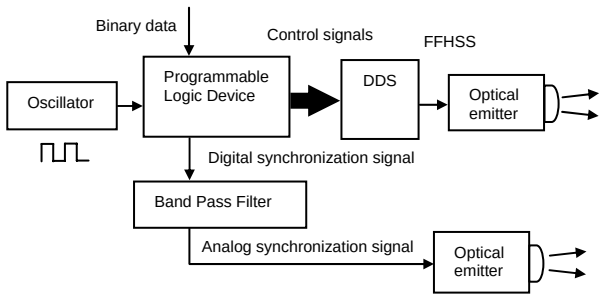

Fig. 7. Block diagram of previous designed transmitter.

The discrete DDS (see Fig. 8) is a digital system excepting the final Digital to Analog Conversor (DAC), its output signal is a sampled signal at 180 MHz, the emitted FFHSS signal is smoothed by the 100 MHz bandwidth of the optical emitter. In the DDS used the output frequency is fixed by the expression (1), where f_{DDS_CLK} is the frequency of the DDS clock (180 MHz), N is the number of bits of the tuning word (32 bits) and Word is a decimal value of 32 bit frequency tuning word.

$$f_{out}=(Word \cdot f_{DDS\_CLK}) / 2^{N} \quad (1)$$

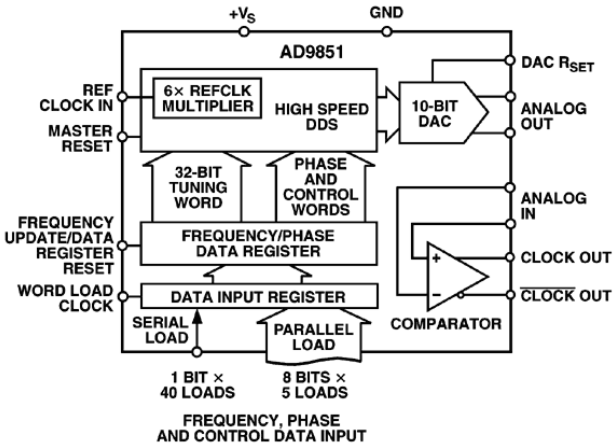

Fig. 8. Block diagram of DDS AD9851 from Analog Devices.

3.2 Designed receiver

In the receiver (see Fig. 9) the signals were separated by discrete filters, the synchronization is reached after adjusting a discrete circuits chain. The synchronization of the receiver makes possible to demodulate the FFHSS signal with two discrete DDS as local oscillators. These two DDS are the same type as the transmitter. In the receiver the control of these DDS is made with the same PLD type as the transmitter. The demodulator is a double branch structure of discrete circuits, each branch is formed by a mixed and an envelope detector, and its intermediate frequency is the standar value of 10,7 MHz.

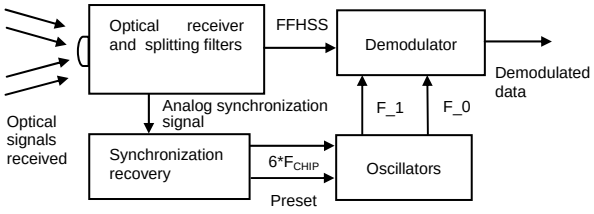

Fig. 9. Block diagram of previous designed receiver.

4 New modulator design

In this section the new modulator design with the new methodology is described, it coincides with the previous designed transmitter (see Fig. 7) except the optical emitters and the output DAC of the discrete DDS.

4.1 Design and simulation with System Generator

The present work consists in redesign the FFHSS modulator and the generation of analog synchronization signal with System Generator. The block diagram of the designed system is in Fig. 10. It is formed by an internal data generator, a code generator, and two DDS to generate the FFHSS and synchronization signals. An external clock of 180 MHz is needed for the system.

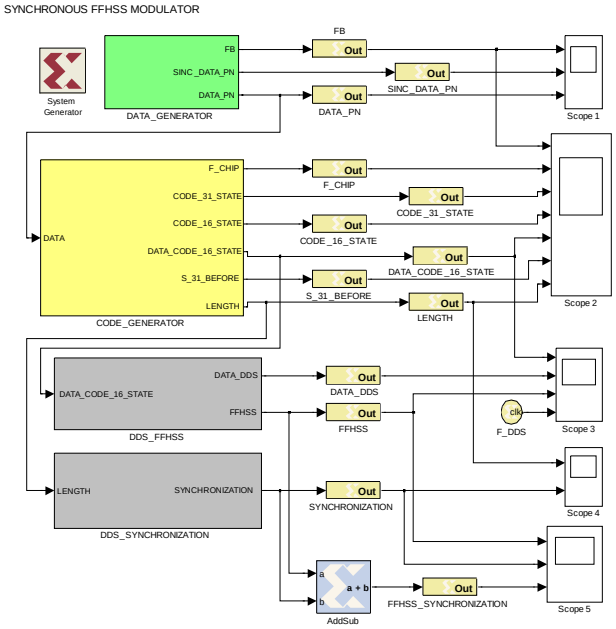

Fig. 10. Block diagram of synchronous FFHSS modulator designed with System Generator.

4.1.1 Pseudorandom data generator

The internal data generator (see Fig. 11) avoids using an external data source, it was designed using a Linear Feedback Shift Register block (LFSR) as pseudorandom generator of 15 bits long at 500 kilobits per second. In the pseudorandom data generator a pulse is generated each time the sequence begins, this provides a high quality periodic signal to synchronize the oscilloscope. In Fig. 12 the clock, the data synchronization pulse and the pseudorandom data are shown after Simulink simulation.

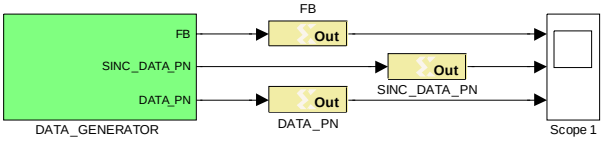

Fig. 11. Internal data pseudorandom generator.

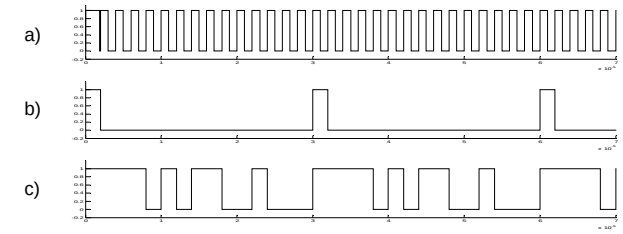

Fig. 12. Pseudorandom data generator signals: a) clock at bit rate, b) the data synchronization pulse, c) the pseudorandom binary data.

4.1.2 Pseudorandom code generator

The pseudorandom code generator is shown in Fig. 13 and its Simulink simulation is on Fig. 14. The code rate is called chip frequency (signal a in Fig. 14), its value is 1,5 Megachips per second, consequently three codes are generated by each data bit. The code generator is based on a LFSR of 31 states 5 bits width (signal b in Fig. 14). In the pseudorandom code generator a pulse is generated each time the sequence begins (signal e in Fig. 14). A five bits code (signal d in Fig. 14) is obtained with the four most significant bits of the pseudorandom code generator (signal c in Fig. 14) and the data bit as most significant bit.

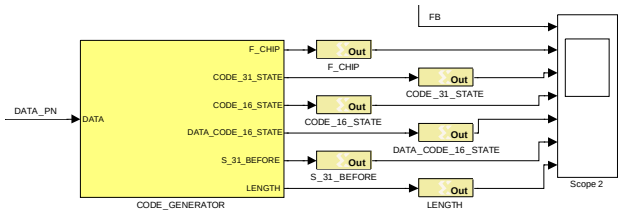

Fig. 13. Pseudorandom code generator.

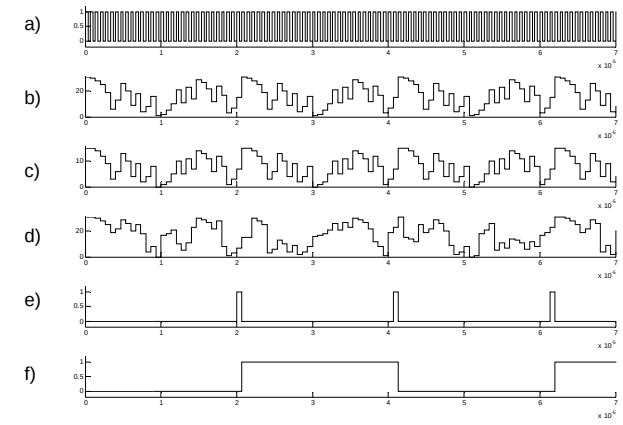

Fig. 14. Pseudorandom code generator signals: a) chip frequency, b) pseudorandom code 5 bits width, c) 4 most signicants bits of pseudorandom code 5 bits width, d) data joined with 4 most signicants bits, e) the previous stage to "11111", f) square signal which marks the code length.

4.1.3 FFHSS signal generation

For each group of five bits (signal d in Fig. 14) a sampled sinusoidal signal is generated according with Table 1.

Code	Frequency (MHZ)	Code	Frequency (MHZ)
00000	24.384	10000	48.960
00001	25.920	10001	50.496
00010	27.456	10010	52.032
00011	28.992	10011	53.568
00100	30.528	10100	55.104
00101	32.064	10101	56.640
00110	33.600	10110	58.176
00111	35.136	10111	59.712
01000	36.672	11000	61.248
01001	38.208	11001	62.784
01010	39.744	11010	64.320
01011	41.280	11011	65.856
01100	42.816	11100	67.392
01101	44.352	11101	68.928
01110	45.888	11110	70.464
01111	47.424	11111	72.000

Table 1. Transmitted frequencies for the FFHSS signal.

In Fig. 15 the DDS that generates the FFHSS signal is shown. The DDS clock is the system clock (180 MHz), therefore with an external filter a pure senoidal signal can be synthesized until a bit less than 90 MHZ.

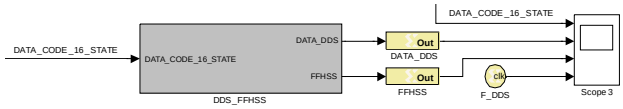

Fig. 15. DDS that generates FFHSS signal.

The input for the Xilinx DDS block must be the division between the synthesized frequency and the DDS clock. The equation (2) shows the meaning of this relation. Consequently, the DDS block fixes the number of bits N according with the rest of the DDS parameters: spurious free dynamic range, resolution, implementation mode, etc.

$$f_{out} / f_{DDS\_CLK} = Word / 2^N \qquad (2)$$

The five bits (signal a in Fig. 16) are transformed to the format of the input of Xilinx DDS block (signal b in Fig. 16). This last operation is an unsigned fixed point integer to unsigned fixed point decimal conversion. In signal c of Fig. 16 five chip times of FFHSS signal are shown. In Fig. 17 there is one chip time, and the DDS clock can be analyzed.

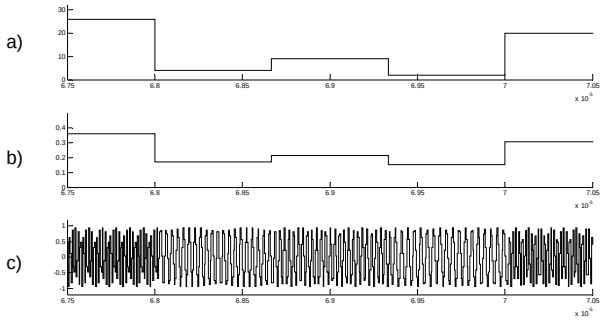

Fig. 16. Signals in DDS that generates the FFHSS signal (five chip times): a) five bits DDS input, b) input for Xilinx DDS block, c) FFHSS signal.

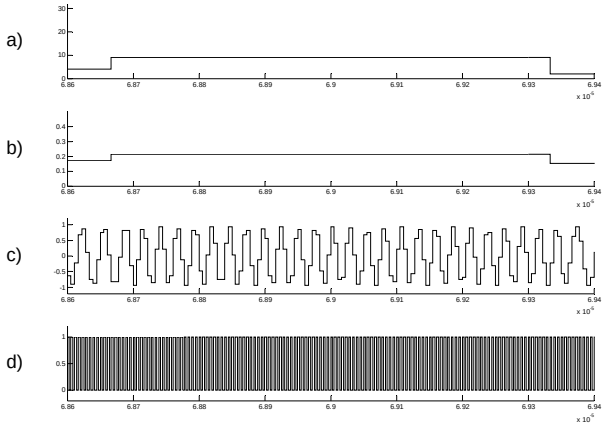

Fig. 17. Signals in DDS that generates the FFHSS signal (one chip time): a) five bits DDS input, b) input for Xilinx DDS block , c) FFHSS signal, d) 180 MHz DDS clock.

4.1.4 Synchronization signal generation

In the pseudorandom code generator a square signal is generated with a 50% duty cycle (signal f in Fig. 14). This square signal has a semiperiod with the same duration as the pseudorandom code length. The square signal is the DDS input that generates the synchronization signal (see Fig. 18), it modulates in phase to a 9 MHz carrier (see Fig. 19). The phase modulated signal carries information about the beginning of the pseudorandom code and its chip frequency because its carrier is a multiple of 1.5 MHz. Finally, the FFHSS and the synchronization signals are added with an *AddSub block* in Fig. 18.

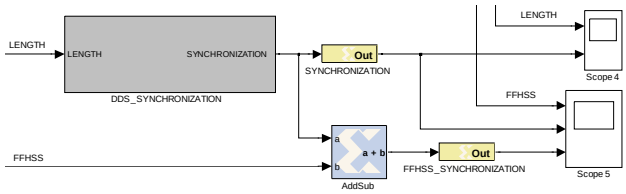

Fig. 18. DDS for synchronization generation and final adder.

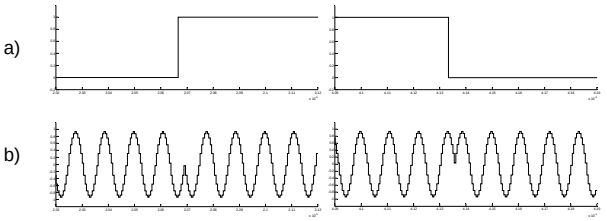

Fig. 19. Signals in DDS that generates the synchronization signal: a) square input signal, b) synchronization signal.

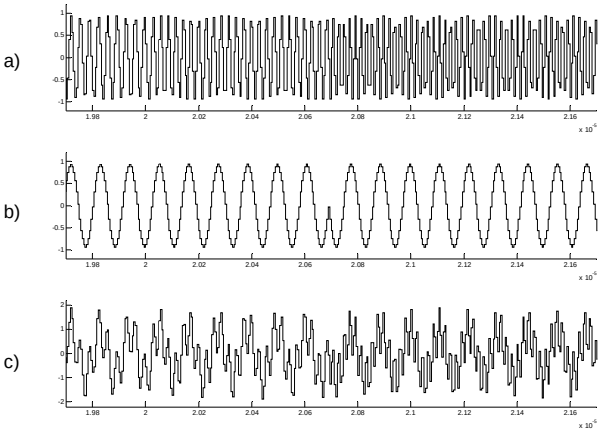

Fig. 20. Inputs and output of final adder: a) FFHSS signal, b) synchronization signal, c) the above signals added together.

It must be noted that data and chip rates were changed softness from the initial prototype values, where the clock of the discrete DDS and the PLD clock were independents, and one frequency was not multiple of the other one. For the chosen FPGA device a master clock must exist, the rest of the clocks are generated by frequency division, the data and chip rates were changed softness to reach this condition. Others FPGA devices allow in its clocks generator block an integer multiplication factor, which joined with the division factor forms a rational number as multiplication factor.

4.2 Simulation and compilation with ISE

After the system has been simulated with Simulink it can be compiled with System Generator. The chosen device is a Virtex4 FPGA, and the hardware description language is VHDL. A project is then generated for ISE, which include the structural description of the system using several files. The syntax of the VHDL files can be checked, and the synthesis and behavioral simulation of the system can be done (see Fig. 21 and 22). After that, the implementation of the design allows the timing simulation of the modulator (see Fig. 23). Lastly, the programming file is generated for the chosen FPGA.

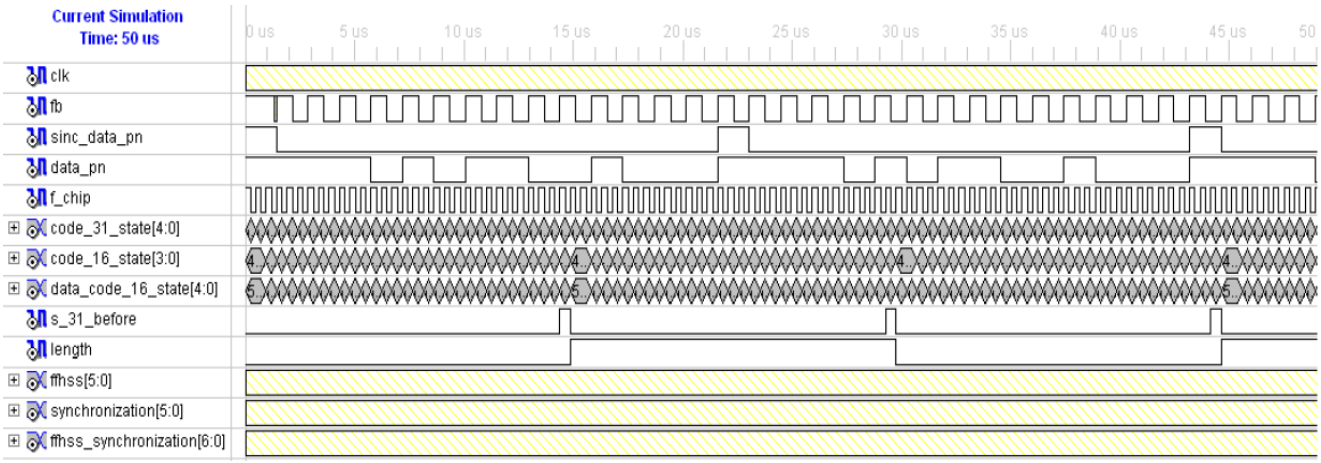

Fig. 21. A long behavioral simulation of synchronous FFHSS modulator using ISE (49 microseconds).

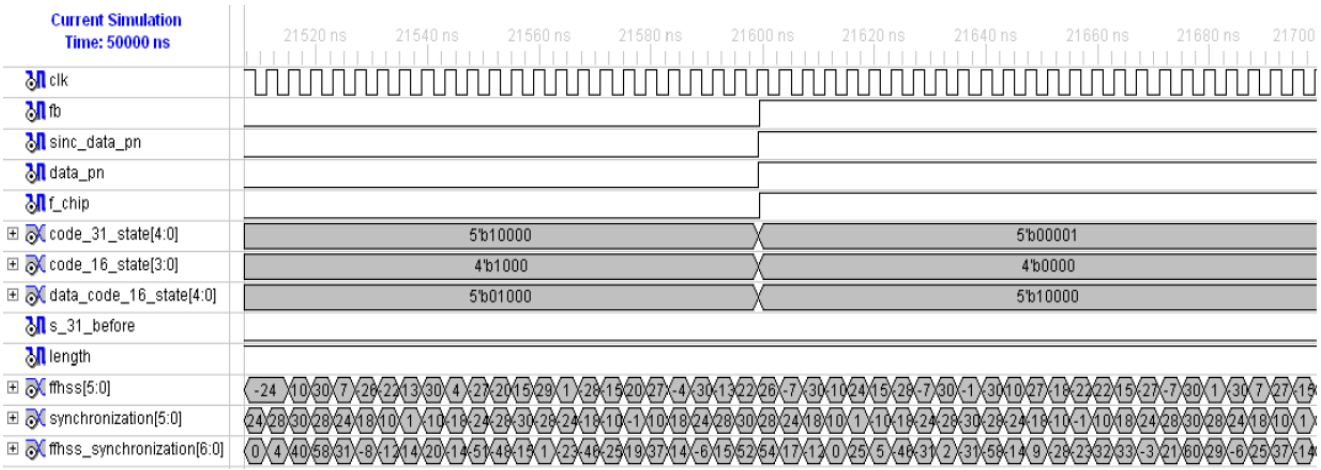

Fig. 22. A short behavioral simulation of synchronous FFHSS modulator using ISE (192.5 nanoseconds).

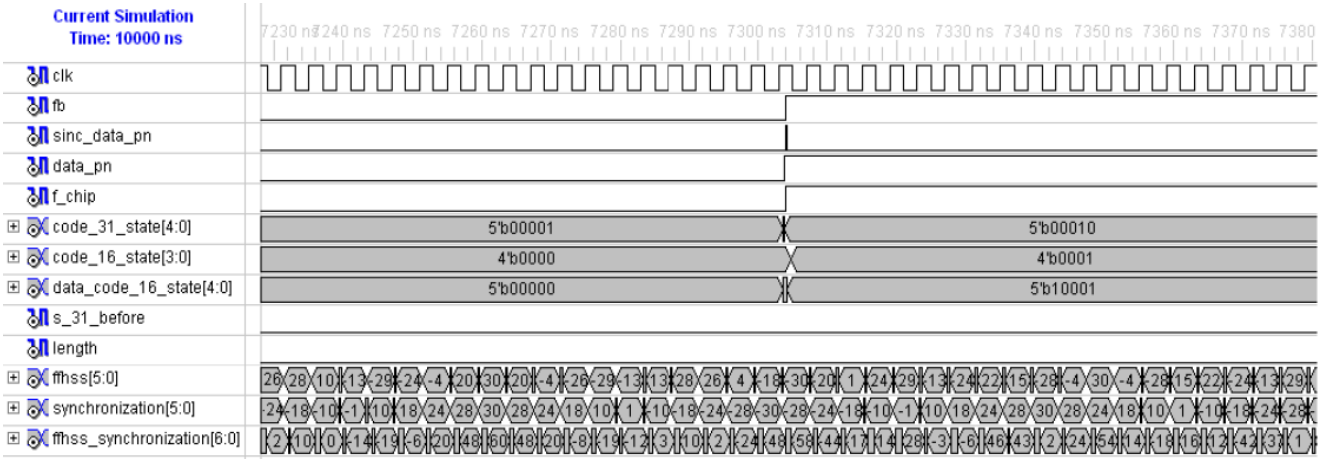

Fig. 23. Timing simulation of synchronous FFHSS modulator using ISE (152 nanoseconds).

The files obtained in this design occupy about 5 thousand lines of VHDL code, and 7 thousand in Verilog. The ISE software provides a power estimator that indicates a dissipation of 0.34 watts in the FPGA, and an estimated temperature of 29.8 degrees centigrade. This represents a power savings of 86% with respect the initial prototype. The FPGA core is supplied with 1.2 volts and the input-outputs pins support the LVCMOS 2.5 volts standard (Low Voltage Complementary Metal Oxide Semiconductor). The occupation rate of hardware in the FPGA is about 1% for logical resources and 20% for input-output pins, however the occupation rate for pins can be reduced until 10% if internal signals are not checked. The timing simulation demonstrates that 250 MSPS can be reached.

4.3 The chosen board

In the moment of writing this document Xilinx does not offer boards with enough performance of Digital to Analog Conversor (DAC). The design needs at least a 7 bits DAC, its rate conversion must reach 180 megasamples per second (MSPS), which is the sample rate of the transmitted signal. It can be verified that several boards exist for the proposed system with enough features, these boards have been developed for companies which are specialized in FPGA kits. These kits include the necessary FPGA or even better and several 14 bits DAC at 480 MSPS. Three DAC are necessary is besides the FFHSS and synchronization signal are monitored. Another solution is to connect two boards, one of them with the FPGA and the other with the DAC stage.

Finally, the proposed board (see Fig. 24) is the ICS-8560A-100 of GE Fanuc [15]. Its block diagram is in Fig. 25, it includes two 16 bits DAC at 400 MSPS. At the moment of writing this work has not been possible to buy this board. Programming the FPGA and capturing results in the laboratory is proposed as future work.

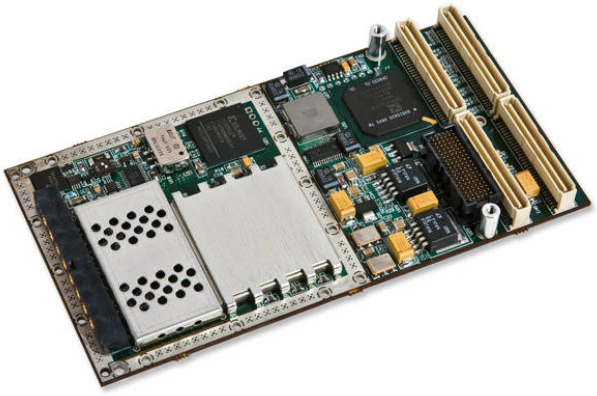

Fig. 24. The proposed board for the synchronous FFHSS modulator: ICS-8560A-100 of GE Fanuc.

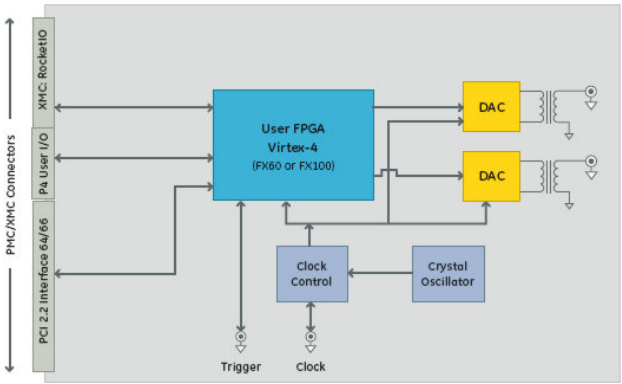

Fig. 25. Block diagram of ICS-8560A-100 board.

5 Conclusions

With this design methodology the typical advantageous features of using programmable digital devices are reached. Repeating a design consists in reprogramming the FPGA in the chosen board, without designing printing circuit boards with discrete circuits. The alternative prototype reduces the number of external discrete components, the integration is improved and the adjusting of analog circuits is avoided.

The design and simulation times are decreased, consequently the time to market is minimizing. It is important to note that electronic equipments are short-lived, and reaching late to the market involves economic losses. The used tool permits great flexibility; in others words, the design parameters can be changed and new features can be checked in several minutes. The Simulink simulations are easy to run, and the signals are shown in floating point format which make easier its analysis. These simulations are possible even before the compilation of the System Generator blocks to obtain the hardware description language files. The possible hardware description languages are Verilog and VHDL.

In the system designed with System Generator the FFHSS and synchronization signals are added in fixed point format, even a weighted adder can be used. The flexibility allows to change the DDS parameters and check its performance. It can be included also an inverse sync filter for compensating variations in amplitude of the sinusoidal signal generated in the DDS output, caused by the sample and retention effect.

The design of the receiver is proposed as future work, all its blocks can be included in a FPGA: splitting filters, synchronization recovery and two branches demodulator. With System Generator is possible to simulate the transceiver, its performance can be tested in presence of additive white gaussian noise, which is available in a System Generator block. Moreover, it is possible to simulate the transmission through a channel with interference, distortion and others spread spectrum signals using different codes.

References:

[1] Scott Hauck, André DeHon, *Reconfigurable Computing*, Elsevier, 2008.

[2] Rosula S.J. Reyes, Carlos M. Oppus, Jose Claro N. Monje, Noel S. Patron, Reynaldo C. Guerrero, Jovilyn Therese B. Fajardo, FPGA Implementation of a Telecommunications Trainer System, *International Journal of*

Circuits, Systems and Signal Processing, Volume 2, Issue 2, 2008, pp. 87-94.

[3] I. Jivet, B. Dragoi, FPGA Implementation of the Curve Generator Algorithm for HW Acceleration Applications, *WSEAS Transactions on Circuits and Systems*, Volume 7, Issue 1, 2008, pp. 7-12.

[4] http://www.xilinx.com/tools/sysgen.htm

[5] http://www.xilinx.com/tools/webpack.htm

[6] Pong P. Chu**,** *FPGA Prototyping by Verilog Examples*, Wiley, 2008.

[7] Volnei A. Pedroni, *Circuit Design with VHDL*, The MIT Press, 2004.

[8] Santiago T. Pérez, José A. Rabadán, Francisco A. Delgado, José R. Velázquez, Rafael Pérez, Design of a synchronous Fast Frequency Hopping Spread Spectrum transceiver for indoor Wireless Optical Communications based on Programmable Logic Devices and Direct Digital Synthesizers, *XVIII Conference on Design of Circuits and Integrated Systems*, 2003, pp. 737-742.

[9] Marvin K. Simon, Jim K. Omura, Robert A. Scholtz, Barry K. Levitt, *Spread Spectrum Communications Handbook*, McGraw-Hill Professional, 1994.

[10] Paul Bechet, Mircea Virgil Popa, Radu Mitran, Iulian Bouleanu, Mircea Bora, Some aspects regarding themeasurement of the Adjacent Channel Interference for Frequency Hopping Radio Systems, *WSEAS Transactions on Signal Processing*, vol. 2, 2006, pp. 991-996.

[11] Joseph M. Kahn, John R. Barry, Wireless Infrared Communications, *Proceedings of the IEEE*, Volume 85, no. 2, 1997, pp. 265-298.

[12] I. Jivet, B. Dragoi, Performance Analysis of Direct Digital Synthesizer Architecture with Amplitude Sequencing, *WSEAS Transactions on Circuits and Systems*, Volume 7, Issue 1, 2008, pp. 1-6.

[13] http://www.analog.com/static/imported-files/data_sheets/AD9851.pdf

[14] http://www.synplicity.com

[15] http://www.gefanuc.com/products/2211