# Direct Sequence Spread Spectrum Transmitter using FPGAs

**Abhijit S. Pandya, Ralph D'Souza** and **Gyoo-Yong Chae,** *Member, KIMICS*

***Abstract***—The DS-SS (Direct Sequence Spread Spectrum) transmitter is part of a low data rate ( ~150 kbps - burst rate and 64 bps - average data rate) wireless communication system. It is traditionally implemented using Digital Signal processing chip (DSP). However, with rapid increase in variety of services through cell phones, such as, web access, video transfer, online games etc. demand for higher rate is increasing steadily. Since the chip rate and thereby the sampling rate requirements of the system are fairly high, the transmitter should implemented using Field programmable Gate Arrays FPGAs instead of a DSP. This paper shows the steps taken to get a working prototype of the transmitter unit on a FPGA based platform.

***Index Terms***—DS-SS (Direct Sequence Spread Spectrum), DSP, FPGA

## I. INTRODUCTION

Spread spectrum was first developed for use by the military because it uses wideband signals that are difficult to detect and that resist attempts at jamming. In recent years, researchers have turned their attention to applying spread spectrum processes for commercial purposes, especially in local area wireless networks.

Direct sequence spread spectrum [1], also known as direct sequence code division multiple access (DS-CDMA), is one of two approaches to sequence spread modulation for digital signal transmission over the airwaves. In direct sequence spread spectrum, the stream of information to be transmitted is divided into small pieces, each of which is allocated across to a frequency channel across the spectrum. A data signal at the point of transmission is combined with a higher data-rate bit sequence (also known as a *chipping code*) that divides the data according to a spreading ratio. The redundant chipping code helps the signal resist interference and also enables the original data to be recovered if data bits are damaged during transmission.

With the emergence over the past few years of larger gate count FPGA devices, the implementation of wideband communication sub-systems on FPGA devices is becoming more prevalent. Although the processing speed of DSP devices is increasing every year, it lags behind the demand for processing speed by emerging wideband wireless communication standards such as 3G, 4G, 802.11 and Bluetooth [2]. When a DSP is used in the implementation of these standards, the high rate signal processing is done outside the DSP, either in an ASIC or an FPGA. In some cases the entire processing is performed in an ASIC or an FPGA. There are several criteria that can be examined when evaluating the processor platform appropriate for the application. Also, algorithms for wireless communications systems that have traditionally been implemented on DSP's by DSP engineers must now be implemented on devices that have a completely different development paradigm. In this paper we discuss Direct Sequence Spread Spectrum (DS-SS) transmitter design using FPGAs

## II. SYSTEM REQUIREMENT

With rapid increase in variety of services through cell phones, such as, web access, video transfer, online games etc. demand for higher rate is increasing steadily. DS-SS transmitters are integral components of wireless systems and their ability to handle higher data rates is crucial for high-speed wireless communication. DS-SS is probably the most widely recognized form of spread spectrum. The DS-SS process is performed by effectively multiplying an RF carrier and a pseudo-noise (PN) digital signal.

First the PN code is modulated onto the information signal using one of several modulation techniques (eg. BPSK, QPSK, etc ). Then, a doubly balanced mixer is used to multiply the RF carrier and PN modulated information signal. This process causes the RF signal to be replaced with a very wide bandwidth signal with the spectral equivalent of a noise signal. The demodulation process (for the BPSK case) is then simply the mixing/multiplying of the same PN modulated carrier with the incoming RF signal. The output is a signal that is a maximum when the two signals exactly equal one another or are "correlated". The correlated signal is then filtered and sent to a BPSK demodulator.

The table 1 below lists some of the system parameters on which the design and implementation of the DS-SS transmitter system are based.

The data payload of 128 bits is made up of preamble, frame synchronization sequence and bits representing the transmitter unit ID, transmission sequence number and

Manuscript received May 25, 2004.

A. S. Pandya is a Professor at Dept of Comp. Sci. and Engg., Florida Atlantic Univ., Boca Raton, Florida-33431,USA

R. D'Souza is with Motorola Inc., Plantation, FL, USA

G. Y. Chae is with the IT design research center, silla University, Busan, Korea,(e-mail:cgy1234@hanmail.net)

(corresponding author to provide phone: +82-051-999-5735)

data obtained from sensing devices attached to the transmitter.

Table 1 parameters for DS-SS system

| System Parameter | Value |
|---|---|
| Chip Rate | 10 MHz |
| Bits per Packet | 128 |
| Chips per Bit | 127 |
| Samples per Chip | 4 |
| Channel Access / Modulation | DS-SS / DOQPSK |
| Frequency Band | 2.4 – 2.483 GHz (ISM band) |
| Transmit Power | 500 mW – 1W |

## III. FUNCTIONAL DESCRIPTION

Fig. 1 shows a functional block diagram of the transmitter. After the 128 bit data frame has been assembled, the odd and even bits in the frame are separated and sent down two separate paths, I and Q. The odd bits sent down the I path, are first differentially encoded and then spread using a 127 chip $PN_I$ sequence. The even bits sent down the Q path, are also differentially encoded first and then spread using a 127 chip $PN_Q$ sequence. On the Q path, there is an extra block where a delay of one-half of a chip is applied to the shaped bit stream.

This is needed to implement the offset in the modulation scheme. After the bits have been spread they are shaped using a half-sine pulse shaping filter. The shaped bits from the I and Q paths are then fed to two D/A converters. The output of the D/A converters are then fed to baseband and then to RF converters where the I and Q signals are up-converted, combined and the resultant signal modulated.

Fig. 1 also represents the transmitter as implemented using off-the-shelf hardware components. The system consists of two major blocks – a Xilinx FPGA based platform which takes care of all baseband processing and sensor interface, and a RF platform which takes care of Baseband→IF and IF→RF conversion and modulation. The functions implemented within each block are listed below the corresponding block.

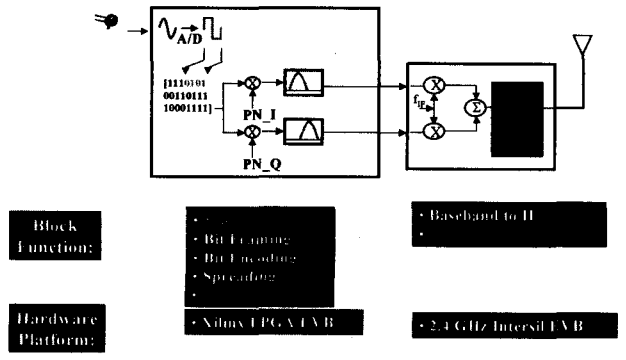



Fig. 1 Prototype DS-SS Transmitter Block Diagram

## IV. PROCEDURE FOR DESIGN

The selected platform, on which the transmitter was implemented, had two FPGAs, with each one of the FPGAs having access to one of the two A/D's and D/A's on board. The board also had an EEPROM, which when programmed and enabled would load the FPGAs with their corresponding configuration files on powerup. Given the board configuration, the functional tasks were split among the two FPGAs - $FPGA_I$ and $FPGA_Q$, as follows:

$FPGA_I$

- A circuit for the A/D interface is implemented on $FPGA_I$. The A/D is connected to a sensor device and is read by $FPGA_I$ once every time period T. The 12 bits representing the signal at the sampling instant are provided as the output by this circuit.
- A circuit to assemble the A/D data bits with the rest of the transmission frame is implemented on $FPGA_I$. The A/D data is combined with the rest of the data and then appended to the preamble and frame synchronization bits to create the entire frame. The frame is then broken up into odd and even bit streams. This circuit then shifts the odd and even bit streams out one bit at a time.
- A circuit to differentially encode the I bit stream is implemented on $FPGA_I$. Fig. 2 shows a functional block diagram of the differential encoder while Table 1 shows the relationship between the input and output bits. The logical "0" represents the negation of the input signal.
- A circuit to differentially encode the Q bit stream is implemented on $FPGA_I$. The operation of this circuit is similar to the one created for the I bit stream. The output bits are read by a circuit on $FPGA_Q$.
- A circuit to spread and shape the differentially encoded I bit stream is implemented on $FPGA_I$. Spreading is accomplished by using the pre-defined $PN_I$ sequence which is 127 chips long. The spread sequence is then processed by a pulse shaping filter where each chip is represented by 4 samples. The output of this circuit is fed to one of the D/A's for conversion to an analog signal and further routing to the IF/RF board.
- Circuits to realize clock rates from the main on board clock are implemented on $FPGA_I$

A circuit to spread and shape the differentially encoded Q bit stream is implemented on $FPGA_Q$. Spreading is accomplished by using the pre-defined $PN_Q$ sequence which is 127 chips long. The spread sequence is then processed by a pulse shaping filter where each chip is represented by 4 samples. The output of this circuit is fed to one of the D/A's for conversion to an analog signal and further routing to the IF/RF board. Circuits to realize clock rates from the main on board clock are implemented on $FPGA_Q$.

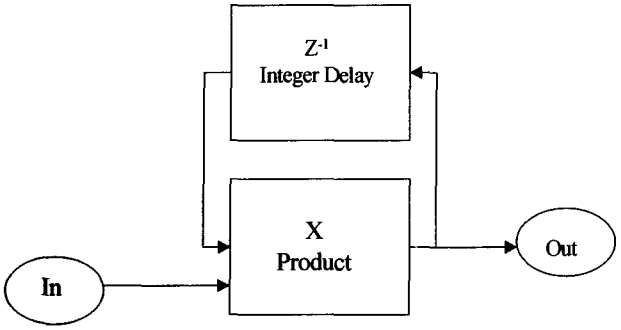



Fig. 2 Functional Block Diagram of Differential Encoder

Table 1 - Differential encoder truth table

| Input bit | Previous Output | Present Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The delay of 1/2 a chip or 2 samples between the I and Q channels, required by OQPSK, is implemented by having $FPGA_I$ generate a control signal for $FPGA_Q$ signaling when the spreading and shaping of the Q bit stream can begin.

The steps necessary to synthesize and simulate the baseband transmitter, described above, using the FPGA design tools Express package are as follows:

1. Develop the Verilog Hardware Design Language HDL [3,4] code to implement the blocks that make up the transmitter
2. Synthesis phase. This is the process of compiling the HDL code into an EDIF net-list of gates.
3. Implementation phase. This includes:

Translation - all input net-lists are merged.

Map - This step optimizes the gates and trims unused logic in the merged net-list. The design's logic resources are also mapped to resources on the silicon and a physical rule check is performed.

Place and route - All logic blocks are assigned to specific locations on the die. If timing constraints have been placed on particular logic components, the placer tries to meet those constraints by moving the corresponding logic blocks closer together. In the routing stage, the logic blocks are assigned specific inter-connect elements on the die. If timing constraints have been placed on particular logic components, the router tries to meet those constraints by choosing a faster interconnect.

Configure - The physical implementation is translated into a bit configuration file that is used to program the FPGA.

Testing and verification of the baseband transmitter functions are performed at two places within the implementation phase. After the synthesis phase functional simulation can be performed to verify that the logic created is correct. For the lack of access to a more standardized tool set, the logic simulator within Foundation Express (Xilinx design package) [5] is used for functional simulation. At the end of the Implementation phase, timing simulation can be performed. Timing simulation uses the block and routing delay information from the routed design to give a more accurate assessment of the behavior of the circuit. Timing simulation will be done with the same tools as the functional simulation. The only difference will be that the design loaded into the simulator for timing simulation will contain routing delays based on the actual placed and routed design.

Functional Simulation - When the logic simulator is invoked, the project netlist created by the synthesis phase is loaded into the simulator. There are three basic steps to simulating the design:

1. Adding signals - the inputs and outputs that are to be seen in the simulator.
2. Adding stimulus - inputs on some of the signals.
3. Running the simulation - and viewing the output waveforms.

Timing Simulation - The timing simulation follows the same process as the functional simulation. The difference will be that the design loaded into the simulator contains the route delays based on the actual placed and routed design.

After the two designs were verified for function and timing, a PROM file was created using the PROM File Formatter utility in Foundation Express. The PROM file consists of a data stream made up of two bit configuration files, one for each FPGA. The bit configuration files contain data to configure the two FPGAs connected in daisy chain fashion on the breadboard. Within the PROM File Formatter utility, in the PROM description area, the bit files are displayed in the order in which they are to be loaded into the FPGA devices on the chain. The first bit file is for $FPGA_I$ and the second bit file is for $FPGA_Q$. After the PROM file is created, it is programmed onto the breadboard using the Download utility within Foundation Express. Once the two FPGA devices were programmed, the reset circuitry on the board asserts the reset signal and the FPGA devices were able to operate based on the design programmed into them.

On the breadboard, several test points were probed to verify the presence of valid signals [6]. The inverted 40 MHz clock was generated within the FPGA, off the on board 80 MHz clock and this signal was fed to the D/A chip. The inverted clock was captured on the scope and the period/frequency verified. Whenever the I or Q transmitter entered the transmission state, the LED corresponding to the I or Q FPGA would light up. This visual indication also verified correct operation of the design. Finally, the data output by the I and Q D/A's were captured on the oscilloscope as well as written to files using the floppy drive on the oscilloscope. The raw I and Q data were plotted using Matlab. In addition a constellation diagram and the eye diagrams for the I and Q channels were plotted from the data. The constellation diagram showed that the offset in samples required for OQPSK modulation was being achieved.

Figure 3 below shows three plots which are based off I and Q data captured from the breadboard.

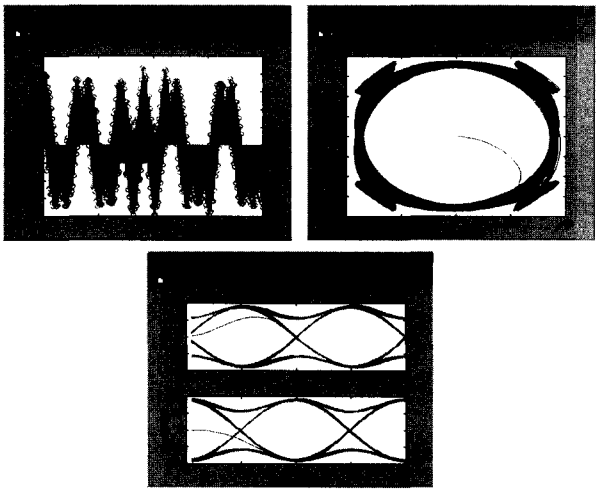


Fig. 3 Plots obtained from I and Q data output by the FPGA's on two D/A channels

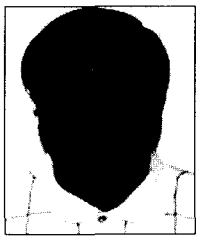
## V. CONCLUSIONS

Applications like web access, video transfer, online games etc. require much higher data transfer rates. With increasing demand for higher communication rate in wireless communications there is a need to explore other ways of designing DS-SS transmitters. We have shown that FPGAs can be an alternative to traditional designs using DSPs which tend to have rate limitations.

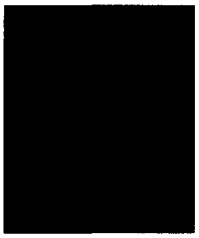
In this paper we have described the design of the baseband processing portion of a DS-SS transmitter and its implementation on a Xilinx based platform. The various steps involved in arriving at a successful implementation have been outlined. There are other tool sets, considered more mainstream, that were available to synthesize and simulate the design. The Foundation series package had a more affordable cost structure and it took on the order of a week to get familiarized with the operation of the package. The design could have been accomplished on FPGA's of smaller sizes (since each bit configuration file occupies less than 20% of the respective FPGA), but the prototype board had the right match of components required to implement our design.

## REFERENCES

[1] R. C. Dixon, spread spectrum systems, John willey and sons, Inc., New York, 1984.
[2] S. LMiller," An adaptive direct-sequence code–division multiple-access receiver for multiuser interference rejection ," *IEEE Trans Commun.*, vol. 43, pp. 1746-1755, Feb./Mar./Apr. 1995.
[3] Samir Palnitkar, Verilog HDL A Guide to Digital Design and Synthesis, SunSoft Press, Prentice Hall Title, 1996.
[4] Douglas Smith, HDL Chip Design, Doone Publications, 1996.
[5] Xilinx, Foundation Series 3.1I Users Guide.
[6] GV & Associates, Inc., GVA-200A Hardware Accelerator User's Manual.



**Dr. A.S. Pandya**
Dr. A.S. Pandya is a professor at the Computer Science and Engineering Department, Florida Atlantic University. He has published over 100 papers and book chapters, and a number of books in the areas of neural networks and ATM networks. This includes a text published by CRC Press and IEEE Press entitled "Pattern Recognition using Neural Networks in C++". He consults for several industries including IBM, Motorola, Coulter industries and the U.S. Patent Office. He received his undergraduate education at the Indian Institute of Technology, Bombay. He earned his M. S. and Ph. D. in Computer Science from the Syracuse University, New York. He has worked as a visiting Professor in various countries including Japan, Korea, India, etc. His areas of research include VLSI implementable algorithms, Applications of AI and Image analysis in Medicine, Financial Forecasting using Neural Networks.



**Gyoo-Yong Chae**
He received the M. S. and the Ph.D. degrees from Department of Industrial Engineering of KonKuk University, Seoul, Korea, in 1993 and 1999, respectively. At present he is a member of research laboratory committee of Silla university IT design research laboratory. His research interests include Fuzzy Neural network, Data mining, Image Processing, Pattern Recognition, Bioinfomatics, DB, EC, etc.