Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 53(67), Fascicola 2, 2008

Alternative Subsystems for the Simulink Implementation of the IEEE 802.15.4 Transmitter

Ana-Maria Popescu[1], Ion Gabriel Tudorache[2], Mircea Mihaiu[3], Carlos Valderrama[4], Papy Ndungidi[5]

Abstract – This paper presents a part of a Simulink toolset, which is currently being developed. More precisely, it describes different subsystems for several functions present in the digital base band part of an IEEE 802.15.4 transmitter. The subsytems' speed as well as their complexity is taken into consideration for comparison. The entire model of the transmitter is developed under MATLAB/Simulink using the 7.3.0.367(R2006b) version.

Keywords: Simulink, IEEE 802.15.4, transmitter, toolset

I. INTRODUCTION

Wireless communication technology is in its infancy and is undergoing rapid development. Innovative standards attempt to respond effectively to the high demand for low cost and low power consumption wireless networking protocols. As the demand for such wireless system increases, so does the need to accurately model these designs.

Behavioral modeling is a very popular approach to simulate and verify complex, mixed-signal systems which consist of RF, analogical and digital parts. The making of Virtual Prototypes allows users to view and analyze the virtual reality which represents the simulated objects with as much accuracy as possible. This facilitates the improvement of the communication systems during the designing stage, without unnecessary costs [1].

The development of a Simulink toolset to serve such purposes consists of multiple stages. The first one implies the simplest step, meaning the making of the ideal elements and chains. The following stages grow in complexity due to the fact that the models are improved with realistic blocks and added imperfections.

This paper proposes to describe a part of a Simulink toolset which is currently being developed and which can be used to simulate the ideal transmitter, receiver and channel of an IEEE 802.15.4 "ZigBee" physical layer. The implementation of such a chain (Fig.1) can be done in multiple ways because of the options Simulink offers for behavioral modeling.

The IEEE 802.15.4 standard [2] was designed to support three frequency bands with three different data rates (250 kbps for the 2.4 GHz ISM World Wide band, 40 kbps for the 915 MHz ISM USA band and 20kbps for the 868 MHz ISM European band). This paper will focus only on the transmitter which employs Binary Phase Shifting (BPSK) modulation (for the 868MHz band and the 968MHz band) and Offset-Quadrature Phase Shift Keying (O-QPSK) modulation (for the 2.4GHz band).

The chosen architecture for the developed models is the Homodyne architecture [3] because it down-converts the RF signal to baseband and minimizes the number of necessary chain components. This architecture offers a lot of advantages like high processing accuracy, high flexibility, low-power consumption and low-cost components. Its DC offset and the 1/f noise problems can be overcome through careful design.

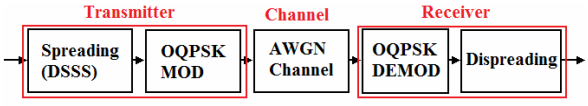

Fig.1. Digital chain for the Zigbee physical layer

Each function that the transmitter performs can be implemented with Simulink blocks in more than one way. The alternative subsystems propose the variation of the number of blocks from the Simulink Library or the use of S-functions which can reduce the

[1]Student at the Faculty of Automation, Computers and Electronics in Craiova, Romania
e-mail: amidamia2003@yahoo.com
[2]Student at the Faculty of Automation, Computers and Electronics in Craiova, Romania
e-mail: tudgabriel@yahoo.com
[3]Professor at the Faculty of Automation, Computers and Electronics in Craiova, Romania
e-mail: mihaium@electronics.ucv.ro
[4]Professor at the Polytechniq Faculty in Mons, Belgium
e-mail: carlos.valderrama@fpms.ac.be
[5]Phd Student and Assistant at the Polytechnic Faculty in Mons, Belgium
e-mail: papy.ndungidi@fpms.ac.be

dimension of the model. Some of the subsystems prove to be simpler and faster, while others are flexible and complex. It is up to the user to choose whichever suits the application best.
The paper is divided into four sections which describe the modeled IEEE 802.15.4 transmitter. The first section presents 3 alternatives for the spreading. The second section presents two alternatives for the BPSK modulation. The third section presents two alternatives for the OQPSK modulation and the fourth section presents the general conclusion. The encountered difficulties as well as their solutions are explained in each section of the paper.

II. THE SPREADING

The transmitted digital data stream is a string of bits that have to be conveyed into symbols and then into chips. This is what the spreading block does. The encoding takes place before the modulation process and is used to replace the transmitted data symbols with codes with the purpose of counteracting the effects of noise [4].
Direct Sequence Spread Spectrum (DSSS) replaces the symbols with a pseudo-noise sequence (PN sequence). For the IEEE 802.15.4 system with BPSK modulation, the PN encoding consists of 2 sequences of 15 bits[2], one to encode the input bit 0 and one to encode input bit 1.
For the IEEE 802.15.4 system which uses OQPSK modulation, the PN coding consists of 16 sequences of 32 bits [2], each sequence coding one of the input symbols, from 0000 to 1111 (representing the numbers from 0 to 15).
The spreading block for the BPSK transmitter is presented in Fig.2. For the OQPSK, the block is changed to have 16 cases with the corresponding codes, but due to dimensions, it is not presented here.

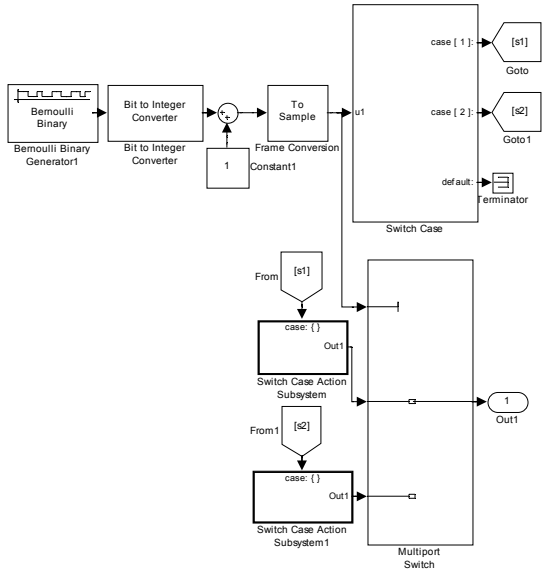

Fig.2. Spreading block for BPSK – alternative no. 1

The spreading block was implemented using:

- a Bernoulli Binary Generator which outputs frames of 1 bit for the BPSK transmitter and frames of 4 bits for the O-QPSK
- a *Bit to Integer Converter*, with a selected number of 1 bit per integer for BPSK and a number of 4 bits per integer for OQPSK,
- a *Sum block* which adds a *Constant* of 1 to the integer outputted by the previous block (the purpose is to shift the integers and have a sequence from 1 to 16 instead of 0 to 15; the reason is that the Multiport Switch accepts only positive integers for the control input),
- a *Frame Conversion block*, which outputs a sample based signal according to the input demands of the following blocks,
- a *Switch Case subsystem* with 16 *Switch Case Action subsystems* which replace each 4 bits symbol with a sequence of 32 bits; in the BPSK case only 2 *Switch Case Action subsystems* are necessary because each bit (0,1) is replaced by a 15 sequence of bits,
- 16 *Go to* labels and 16 *From* labels in the OQPSK case; for BPSK only 2 *Go to* labels are necessary and 2 *From* labels (these labels are used to simplify the model),
- and a *Multiport Switch* block which outputs a sequence of 32 bits.

Another option to implement the spreading block would be to avoid the use of “Go To” and “From” labels. The architecture of the subsystem contains fewer blocks, but it has a more complex design, especially for the O-QPSK transmitter which contains 16 PN codes.
The spreading can also be performed with a *Direct Lookup Table* and the dimensions of the block can be greatly reduced (Fig.3). The Direct Lookup table has to be configured to extract a column from a matrix. The columns consist in the PN code. For the BPSK transmitter, there are 2 columns, each with 15 bits. For the OQPSK transmitter, there are 16 columns each with 32 bits. In the first case the table is a 15x2 matrix and, in the second case, the table is a 16x32 matrix.

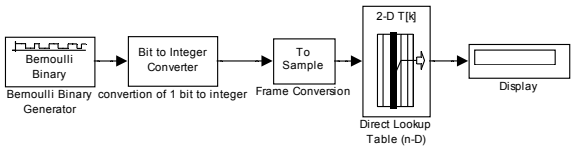

Fig.3. Spreading block for BPSK - alternative no. 3

Simulink offers another alternative for the spreading through the S-function block. An S-function block can contain a user-defined function written in any of these languages: MATLAB, C, C++, FORTRAN or ADA. The S-functions act as general purpose functions with adjustable parameters which can serve more than once in an application. By using the S-function template [3] from MATLAB and by adjusting it with the necessary simple algorithm, the spreading can take place without any additional blocks.

III. BPSK MODULATION

The BPSK modulation is the simplest form of PSK modulation [5]. It uses two opposite signal phases to encode 0 and 1 bits. The state of the wave is changed according to the inputted bits. If consecutive bits are the same, then the wave phase remains the same until a bit change ocurrs. If the bits change so does the phase of the wave.

The Simulink library offers a first option for the BPSK modulation (Fig.4). It contains a BPSK modulation block which can be used in the transmitter. It is a recommended option because it is fast, but it is not a flexible.

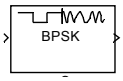

Fig.4. BPSK modulation block from Simulink - alternative no. 1

The speed for this block was analyzed by using the MATLAB Profiler function. After the models for the BPSK transmitter were completed and after running the profiler function for each of them, one of the models had a total simulation time of aproximately 44 seconds, while the other had a simulation time of aproximately 2 seconds. The difference between these models consisted in the BPSK modulation block.

The model which used the BPSK block from the library was the fastest, while the one which used the other BPSK modulation subsytem (implemented through other blocks like in Fig.5) had a longer total simulation time. The reason for this is simple: the Library blocks are implemented through C functions and have Simulink masks, therefore they are faster to compile. Implementing a subsytem with the help of other blocks implies using more C functions which take more time to be compiled.

While the advantage of the library blocks is speed, their disadvantage is the lack of flexibility. The parameters of this blocks can not be adjusted (for example the frequency of the carrier). These blocks also accept only a certain type of input and output.

However, the implementation of alternative blocks, like the one in Fig.5, solves this problem and offers flexibility in the detriment of speed.

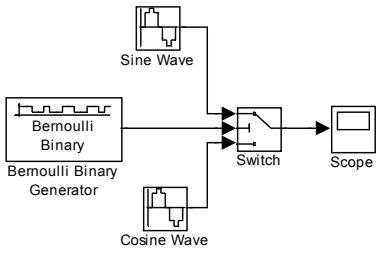

Fig.5. BPSK modulator implemented with a switch block – alternative no. 2

The BPSK modulation in Fig.5 is made by using a switch and two sine waves (a sine and a cosine). The output is clearly the modulated signal.

If used in a transmitter chain after the spreading block, the BPSK modulation in Fig.5 needs to be added a few blocks. The role of these additional blocks is to adjust the output from the spreading block.

The output of the spreading is a sequence (a vector of 15 bits). The modulator needs to receive 1 bit at a time. The blocks which need to be added are 15 Variable Selectors connected to a Multiport Switch which is controled by a Counter Limited block.

An alternative to these additional blocks is a Demultiplexor with 15 outputs and a Multiport Switch controled through a Counter Limited block. Because the blocks are numerous, they won't be shown here.

IV. OQPSK MODULATION

O-QPSK is the modulation used for the 2.4 GHz transmission [6]. In O-QPSK modulation, the inputted bit stream is divided into two parts. The first bit stream corresponds to the even positioned bits of the initial stream and the second bit stream corresponds to the odd positioned bits. The even bit stream is called the in-phase component (I) and odd bit stream is called the quadrature-phase component (Q). The bits in the in-phase and quadrature-phase bit streams are transmitted with an offset of half a symbol period, not in the same instants as in the QPSK modulation. Therefore, the O-QPSK modulation takes place for 2 bits at a time and the two components are then separately modulated onto two orthogonal sinusoidal waves.

As in the BPSK case, MATLAB offers users the possibility to implement communication systems using O-QPSK modulation through an O-QPSK block found in the Simulink library (Fig.6). To implement this block in a transmitter model, its input must be adjusted according to its limitations.

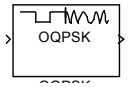

Fig.6. O-QPSK modulation block from Simulink - alternative no. 1

A subsystem, the *Two bits at a time* subsystem, which receives the 32 bit output from the spreading block, is placed before the Library's modulation block (Fig.7). It outputs two bits at a time as expected by the O-QPSK modulator. The Two bits at a time subsystem consists of 16 Variable Selector blocks and a Multiport Switch controlled through a Limited Counter.

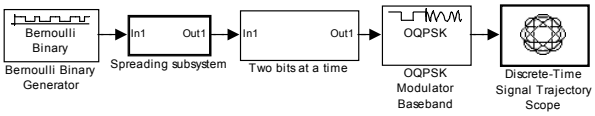

Fig.7. Transmitter chain which uses the Simulink O-QPSK modulation block

The O-QPSK modulator block in Fig.6 is advantageous because of the speed and reduced complexity and, disadvantageous due to flexibility issues. In fact, because of the lack of flexibility, the complexity of the transmitter's chain is increased through the adding of the Two bits at a time block.

After applying the Profiler, the measured speed for a model using the Library's O-QPSK modulation block was approximately 2 seconds. For the other model, with the modulator from Fig.8, it was approximately 48 seconds. In conclusion, when choosing the modulation block, the user has to make a compromise between flexibility and speed.

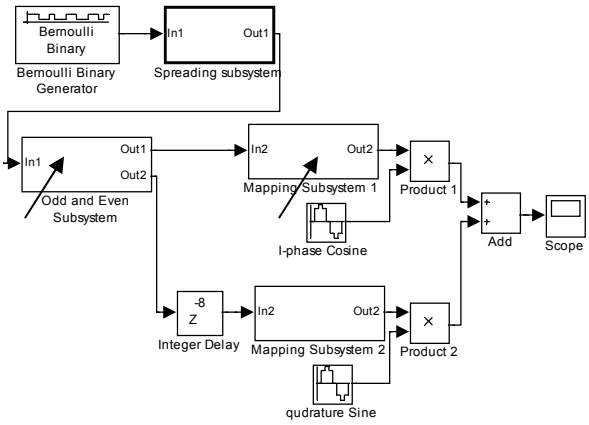

Fig.8. O-QPSK modulation block from Simulink - alternative no. 2

Implementing the O-QPSK modulation with other blocks from the Library offers the user a more complex transmitter, but a more flexible one (Fig.8).
This solution to implement the O-QPSK modulation uses:

- an Odd and Even subsystem which splits the 32 bit sequence into two sequences of odd and even positioned bits
- a mapping subsystem which outputs -1 for the bit 0 and 1 for the bit 1; these values represent the voltage values -1 and 1
- a sine and cosine wave which are modulated by the transmitted information;

The result of this block is the modulated signal.
The arrows in Fig.8 indicate the blocks which can be implemented in different ways.
The Odd and Even subsystem can be made with the help of two S-functions (odevI and odevQ) like in Fig.9. The algorithm and the code for the two functions is the same with one exception - the output: sys=I or sys=Q.

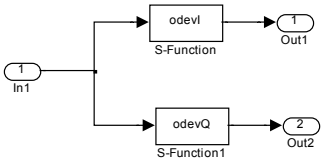

Fig.9. Odd and Even subsystem - alternative no. 1

If the S-function solution is not desired, then another alternative can be implemented successfully. This alternative implies the use of a *Deinterlacer* block and of additional blocks which adjust the format of the input for this block (Fig.10). The output of the spreading block is a 32 bit sequence and the input of the Deinterlacer is limited to a certain format. By introducing the *Two bits at a time* block, which is described in the previous section of the article, the Deinterlacer receives the correct 2 bit input.

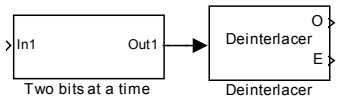

Fig.10. Odd and Even subsystem - alternative no. 2

However, this alternative was not as easily made. It was the result of a long analysis and consecutive implementations. The initial ideas for this subsystem, which did not function, were the following:

- a subsystem which contains an *If Case block* and which is commanded by a *Pulse Generator*, like in Fig.11. The subsystem's input is a double input, one for the 32 bits and one for the generator. The output is double as well, one for each string of 16 bits. The result of this block should be the splitting of the initial sequence into 2 sequences of odd and even bits, but unfortunatelly the simulation does not function as expected. Both outputs are the same with the input in the Odd and Even subsytem. So, this version can not be used.

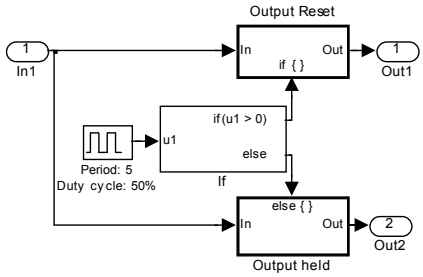

Fig.11. Odd and Even subsystem - alternative no. 3

- a subsytem which consists of 2 *Enabled Subsystems*, a *Pulse Generator* and a *Not* block, like in Fig.12. The input data sequence of 32 bits is sent to both Enabled Subsystems, but one is enabled when the generator sends a 1, while the other is enabled when it sends a zero. The output should be a string of odd bits and one of even bits, but this subsytem does not function as expected either. One of the Enabled Subsytems outputs the exact input of this Odd and Even subsytem and the other Enabled Subsytem outputs a continuous string of 0 bits.

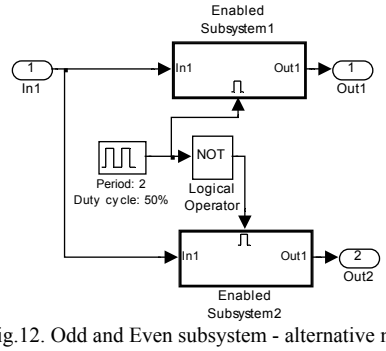

Fig.12. Odd and Even subsystem - alternative no. 4

The mapping subsystem can also be implemented in two ways: by using a *Unipolar to Bipolar* block or by using a *Switch* with two positions, for -1 and 1(Fig.13

and Fig.14). These blocks have to be placed on each branch (meaning the same operations have to be applied for the in-phase and for the quadrature-phase component).

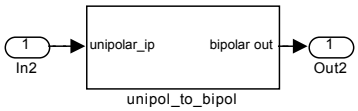

Fig.13. Mapping subsystem - alternative no. 1

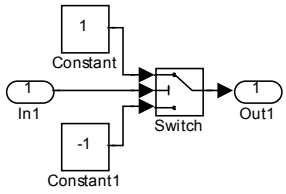

Fig.14. Mapping subsystem - alternative no. 2

With these alternatives for different parts of the transmitter, the user can implement an entire model. As the toolset is not completed, the RF part has to be added to obtain a real model of the transmitter [7].

V. CONCLUSION

The overall goal of this paper is to contribute to the creation of a toolset to help dimension and improve wireless communication applications. System designers can adjust the already existent devices to improve their functionality by simulating their models in Simulink and using toolsets like the one being developed right now.

For example, the reliability of systems which use the IEEE 802.15.4 technology can be evaluated correctly because of more realistic models. The blocks implemented in this paper can be reused and their parameters can be changed according to the data rate and frequency of the transmitter for numerous cases and simulations.

The alternative subsystems can be further improved to a level which implies not only flexibility, but speed as well. So, additional work will be done to offer these subsystems the same speed as the blocks from the Simulink Library.

While implementing the functions of the transmitter did not create difficulties, adjusting the output from certain blocks to become the proper input for other blocks proved to be time consuming. The result however was satisfactory, especially after creating the entire chain (meaning the transmitter, the receiver and the channel as well) and after calculating the Bit Error Rate. The transmitter's efficiency was only afterwards analyzed with accuracy.

REFERENCES

[1]The MathWorks, Inc., "Communications Blockset. For Use with Simulink", Version 2,

[2] The Institute of Electrical and Electronics Engineers, Inc., **"**Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)", 2006, pg.48 and pg.28, pg.51

[3] S. Zouari, M. Loulou, A. Fakhfakh et N. Masmoudi, Etude "Comparative des Architectures de Réception pour un Système Radiomobile de Troisième Génération", pg.4, http://www.csgroup.tunet.tn/publications_equipe_c&s/zouari/setit_2003/SETIT-R290RN.pdf

[3]The MathWorks, Inc., "Simulink. Simulation and Model-Based Design. Writing S-Functions", Version 6

[4]Chinmoy Gavini , "Quantifying Tradeoffs in the IEEE 802.15.4 protocol through simulation", 2007 http://users.ece.utexas.edu/~bevans/courses/ee464/ChinmoyGavini FinalReport2007.pdf

[5]Johanna S. Ruque, David I. Ruiz, Carlos E. Carrión "Simulation and implementation of the BPSK modulation on a FPGA Xilinx Spartan 3 xcs200-4ftp256, using Simulink and the System Generator blockset for DSP/FPGA.", www.eece.unm.edu/xup/docs/collaboration/BPSK_Modulator.pdf

[6]John Allgeyer, Jamie Jenshak, „Quadrature Amplitude Modulation using Simulink", 2003 http://cegt201.bradley.edu/projects/proj2004/dispjjja/Microsoft%20Word%20-%20Project%20Proposal2.pdf

[7]Eloi Ramon, Jordi Carrabina "Using FPGAs for Software-Defined Radio Systems: a PHY layer for an 802.15.4 transceiver" http://cephis.uab.es/resources/pdf/papers/JCRA_2005_SDR.pdf