

# **CONTRIBUTIONS TO FPGA-BASED DIGITAL MODULATION/DEMODULATION TECHNIQUES**

Teză destinată obținerii  
titlului științific de doctor inginer  
la  
Universitatea "Politehnica" din Timișoara  
în domeniul Inginerie Electrică și Telecomunicații  
de către

**ing. Silvana-Oana POPESCU**

Conducător științific:	prof.dr.ing. Aurel GONTEAN
Referenți științifici:	prof.dr.ing. Paul SVASTA
	prof.dr.ing. Dan PITICA
	prof.dr.ing. Alimpie IGNEA

Ziua susținerii tezei: 19 Martie 2012

Seriile Teze de doctorat ale UPT sunt:

- |                        |   |
|------------------------|---|
| 1. Automatică          | 7. Inginerie Electronică și Telecomunicații |
| 2. Chimie              | 8. Inginerie Industrială                    |
| 3. Energetică          | 9. Inginerie Mecanică                       |
| 4. Ingineria Chimică   | 10. Știința Calculatoarelor                 |
| 5. Inginerie Civilă    | 11. Știința și Ingineria Materialelor       |
| 6. Inginerie Electrică |   |

Universitatea „Politehnica” din Timișoara a inițiat seriile de mai sus în scopul diseminării expertizei, cunoștințelor și rezultatelor cercetărilor întreprinse în cadrul școlii doctorale a universității. Seriile conțin, potrivit H.B.Ex.S Nr. 14 / 14.07.2006, tezele de doctorat susținute în universitate începând cu 1 octombrie 2006.

Copyright © Editura Politehnica – Timișoara, 2012

Această publicație este supusă prevederilor legii dreptului de autor. Multiplicarea acestei publicații, în mod integral sau în parte, traducerea, tipărirea, reutilizarea ilustrațiilor, expunerea, radiodifuzarea, reproducerea pe microfilme sau în orice altă formă este permisă numai cu respectarea prevederilor Legii române a dreptului de autor în vigoare și permisiunea pentru utilizare obținută în scris din partea Universității „Politehnica” din Timișoara. Toate încălcările acestor drepturi vor fi penalizate potrivit Legii române a drepturilor de autor.

România, 300159 Timișoara, Bd. Republicii 9,  
tel. 0256 403823, fax. 0256 403221  
e-mail: editura@edipol.upt.ro

## Acknowledgements

This doctoral thesis was supported in part by POSDRU/6/1.5/S/13 strategic grant, ID6998, financed from European Social Fund "Investing in people" in the Human Resources Development Operational Programme 2007-2013.

This thesis is the result of a sustained effort over the last 3 years in the Applied Electronics Department from the University Politehnica Timisoara.

My sincere thanks go to all people that have contributed in various ways to the finalization of this work.

First, I would like to thank prof. PhD eng. Aurel Gontean for being a great advisor. His ideas, guidance, encouragement and patience were absolutely invaluable.

I am extremely grateful to conf. PhD eng. Georgeta Budura for her permanent support and endless discussions about my research and life, in general. I also want to thank her for reading this thesis and being so kind about it.

Thanks to conf. PhD eng. Ioan Jivet for reading this thesis and for his last minutes advices.

Special thanks to the members of the thesis committee, prof. PhD eng. Marius Ottesteanu - University Politehnica Timisoara, prof. PhD eng. Paul Svasta - University Politehnica Bucurest, prof. PhD eng. Dan Pitica - Technical University Cluj-Napoca and prof. PhD eng. Alimpie Ignea - University Politehnica Timisoara for accepting to be part of my thesis jury, for their evaluations and suggestions regarding the thesis.

I would also like my undergratuated students with whom I worked with.

Many thanks go to all my friends for their support and care.

Above all, I would like to thank my parents who have supported me constantly. This thesis is for you! Thanks Mom and Dad for letting me persue my dreams.

Timișoara, March 2012

Silvana-Oana POPESCU

## ***To my parents***

Popescu, Silvana-Oana

### **Contributions to the FPGA-Based Digital Modulation/Demodulation Techniques**

Teze de doctorat ale UPT, Seria 7, Nr. 45, Editura Politehnica, 2012,  
148 pagini, 146 figuri, 17 tabele.

ISSN: 1842-7014

ISBN: 978-606-554-466-6

Keywords: BPSK, digital, demodulator, detector, FPGA, modulator, QPSK,  
simulation, System Generator, VHDL.

#### **Abstract:**

In the field of digital communications, a remarkable development had happened in the last years as new technologies and applications had emerged daily and FPGAs became an essential part in implementing DSP systems.

This work presents the use of reconfigurable computing devices (FPGAs) in digital communication systems implementation due to the fact that FPGAs have the ability to realize high-speed parallel operations, being ideal for high-performance digital signal processing.

Designs of the Binary Phase Shift Keying and Quadrature Phase Shift Keying systems were the subject of simulation, testing and experimental implementations on FPGAs. The results lead to the conclusion that such systems are suitable, both in common area applications for data transmission and in more specific fields like underwater communication.

# CONTENT

Table of content .....	5
List of figures .....	7
List of tables .....	12
List of Acronyms .....	13
I. MOTIVATION .....	15
1.1. Introduction .....	15
1.2. Thesis Outline .....	16
II. RESOURCES. METHODS. WAYS .....	17
2.1. Reconfigurable Computing .....	17
2.1.1. A Brief Overview of Reconfigurable Computing .....	17
2.1.2. Field Programmable Gate Array .....	18
2.1.2.1. FPGA Architecture .....	19
2.1.2.2. FPGA Families and Models .....	21
2.1.2.3. FPGA Design Flow .....	22
2.2. Simulation Capabilities on Nexys2 and Virtex-II boards. Case Study .....	26
2.2.1 Simulations on Nexys2 board .....	27
2.2.2 Simulations on Virtex-II board .....	31
2.3. Reconfigurable Computings' Suitability for Digital Signal Processing .....	34
2.3.1. Digital Signal Processing .....	34
2.3.2. System Level Tools for DSP in FPGAs .....	35
2.4. Application in Matlab/Simulink and System Generator .....	38
2.4.1. Adder in Simulink .....	39
2.4.2. Adder in System Generator .....	39
2.5. Conclusions and Contributions .....	43
III. MODULATION IN DIGITAL AGE PROSPECTION .....	45
3.1. Digital Communication System .....	45
3.1.1. General Assumption for the Digital Communication System .....	45
3.1.2. Criteria of Choosing Modulation Schemes .....	49
3.1.2.1. Power Efficiency .....	49
3.1.2.2. Bandwidth Efficiency .....	50
3.1.2.3. System Complexity .....	50
3.2. Basics of Digital Modulation Techniques .....	50
3.2.1. Introduction in Digital Modulation .....	50
3.2.2. Overview of Digital Modulation Techniques .....	51
3.2.3. M-ary Phase Shift Keying .....	53
3.2.3.1. Binary Phase Shift Keying .....	53
3.2.3.2. Quadrature Phase Shift Keying .....	55
3.3. A Literature Survey .....	62
3.4. Conclusions and Contributions .....	69
IV. ARGUMENTED BPSK MODULATION FROM FPGA PERSPECTIVE .....	71
4.1. BPSK Detector .....	72
4.2. BPSK Modulator .....	76
4.2.1. BPSK Modulator in Simulink .....	76

6 Content

---

4.2.2.	BPSK Modulator in System Generator .....	78
4.2.3.	BPSK Modulator on FPGA via VGA .....	82
4.2.4.	BPSK Modulator on FPGA via oscilloscope .....	84
4.3.	BPSK System .....	86
4.3.1.	BPSK System in Simulink .....	87
4.3.2.	BPSK System in System Generator .....	89
	BPSK System on FPGA .....	90
4.4.	Conclusions and Contributions .....	92
V.	ARGUMENTED QPSK MODULATION FROM FPGA PERSPECTIVE .....	96
5.1.	QPSK Modulator .....	97
5.1.1.	QPSK Modulator in Simulink .....	97
5.1.2.	QPSK Modulator in System Generator .....	99
5.1.3.	QPSK Modulator on FPGA .....	102
5.2.	QPSK System .....	104
5.2.1.	QPSK System in Simulink .....	104
5.2.2.	QPSK System in System Generator .....	107
5.2.3.	QPSK System on FPGA .....	109
5.3.	Conclusions and Contributions .....	111
VI.	HARDWARE CO-SIMULATION OF THE BPSK AND QPSK SYSTEMS .....	114
6.1.	BPSK Modulator .....	116
6.2.	BPSK System .....	117
6.3.	QPSK Modulator .....	120
6.4.	QPSK System .....	122
6.5.	Conclusions and Contributions .....	125
VII.	CONCLUSIONS AND CONTRIBUTIONS .....	126
7.1.	Conclusions .....	126
7.2.	Contributions .....	131
7.3.	Future Directions .....	133
7.4.	Publication List .....	134
	BIBLIOGRAPHY .....	135
	Appendix .....	146

## LIST OF FIGURES

Fig. 2.1	Generic FPGA architecture .....	19
Fig. 2.2	FPGA using (a) the direct interconnection model and (b) the segmented interconnection model .....	20
Fig. 2.3	Boards of the Spartan 3E FPGA family .....	21
	(a) Nexys2	
	(b) Spartan 3E Starter Kit	
Fig. 2.4	Virtex-II Board .....	22
Fig. 2.5	FPGA Design Flow .....	23
Fig. 2.6	FPGA Design Flow with files .....	23
Fig. 2.7	The overall view of the described design flow .....	25
Fig. 2.8	A n bits ring counter .....	26
Fig. 2.9	Behavioral Simulation of the six-phase generator .....	27
Fig. 2.10	Zoomed aria of the Behavioral Simulation .....	27
Fig. 2.11	The post-route simulation of the six-phase generator .....	28
Fig. 2.12	Zoomed delayed detail between out[3] and out[4] .....	28
Fig. 2.13	The measured delay between the third and the fourth signal .....	29
Fig. 2.14	Design Summary of the six-phase generator .....	30
Fig. 2.15	Behavioral Simulation of the eight-phase generator .....	31
Fig. 2.16	Zoomed aria of the Behavioral Simulation .....	31
Fig. 2.17	The Post-Route Simulation of the eight-phase generator .....	32
Fig. 2.18	Zoomed delayed detail between out[3] and out[4] .....	32
Fig. 2.19	The measured delay between the third and the fourth signal.....	33
Fig. 2.20	Design Summary of the eight-phase generator .....	34
Fig. 2.21	System Generator design flow .....	36
Fig. 2.22	FPGA development process based on System Generator .....	37
Fig. 2.23	The Simulink model of the adder .....	39
Fig. 2.24	Corresponding signals: 1-constant waveform 2-sinus waveform 3- output waveform .....	39
Fig. 2.25	First implementation of the System Generator adder .....	39
Fig. 2.26	The corresponding signals from the scope block: a – constant waveform b – sine waveform c – output signal .....	40
Fig. 2.27	The corresponding signals from the waveform block: a – constant waveform b – sine waveform c – output signal .....	40
Fig. 2.28	Second implementation of the System Generator adder .....	41
Fig. 2.29	The corresponding signals: a – constant waveform b – sine waveform c – output signal .....	41
Fig. 2.30	The third implementation of the System Generator adder .....	41
Fig. 2.31	The corresponding signals: a – constant waveform b – sine waveform c – output signal .....	42
Fig. 2.32	The fourth implementation of the System Generator adder .....	42
Fig. 2.33	The corresponding signals: a – constant waveform b – sine waveform c – output signal .....	43
Fig. 3.1	Block diagram of a typical digital communication system .....	45

Fig. 3.2	The modulator scheme .....	46
Fig. 3.3	The block diagram of a binary digital communication system .....	47
Fig. 3.4	Digital modulation tree .....	52
Fig. 3.5	PSK modulation .....	53
Fig. 3.6	Signal space plot of BPSK .....	54
Fig. 3.7	BPSK Modulator .....	54
Fig. 3.8	BPSK Demodulator .....	54
Fig. 3.9	Application of the BPSK Modulation .....	55
Fig. 3.10	QPSK waveforms .....	58
Fig. 3.11	Representing QPSK signals as phases of the sinusoidal carrier.....	59
Fig. 3.12	QPSK Modulator .....	59
Fig. 3.13	QPSK Demodulator .....	60
Fig. 3.14	Application of the QPSK Modulation .....	61
Fig. 3.15	Error performance of the BPSK and QPSK modulation schemes .....	61
Fig. 3.16	(a) Arhitecture of digital implementation and simulation system (b) Logic diagram of the PN generator .....	64
Fig. 3.17	BPSK carrier signal and PN source data .....	65
Fig. 3.18	Architecture of computational section of digital BPSK detector .....	66
Fig. 3.19	BPSK Transmitter using FPGA .....	66
Fig. 3.20	An underwater acoustic experiment platform .....	67
Fig. 4.1	First step in implementing the BPSK modulation technique .....	71
Fig. 4.2	Second step in implementing the BPSK modulation technique .....	71
Fig. 4.3	Third step of implementing the BPSK modulation technique .....	71
Fig. 4.4	The diagram of the BPSK detector .....	72
Fig. 4.5	BPSK carrier signal and PN source data .....	73
Fig. 4.6	Principle of the BPSK Detector .....	73
Fig. 4.7	Test bench lab .....	74
Fig. 4.8	The transmitted signal if the input is 1 .....	74
Fig. 4.9	The transmitted signal if the input is 0 .....	74
Fig. 4.10	Design Summary of the BPSK Detector .....	75
Fig. 4.11	The resources map of the BPSK Detector on the Nexys2 board ..	75
Fig. 4.12	BPSK Modulator in the Simulink environment .....	76
Fig. 4.13	The waveforms on the scope: (a) Sine (b) -Sine (c) Modulating signal (d) Modulated signal .....	77
Fig.4.14	Second implementation of the BPSK Modulator in Simulink .....	77
Fig. 4.15	BPSK Modulator in System Generator .....	78
Fig. 4.16	The modulating and modulated signals .....	78
Fig. 4.17	A second implementation of the BPSK Modulator in System Generator .....	79
Fig. 4.18	The waveforms: (a) Sine (b) -Sine (c) Modulating signal generated by the LFSR (d) Modulated signal .....	79
Fig. 4.19	The third model of the modulator .....	80
Fig. 4.20	The waveforms: (a) The modulated signal; (b) The modulating signal; (c) Sine; (d) -Sine; (e) The output of the LFSR (f) The signal obtained external, from a function generator .....	81
Fig. 4.21	The fourth implementation of the BPSK Modulator in System Generator .....	81
Fig. 4.22	The waveforms: (a) Sine (b) -Sine (c) The modulating signal (d) The modulated signal .....	82



Fig. 4.23	Test bench lab of the BPSK modulator on VGA .....	82
Fig. 4.24	The transmitted signal if the input is 1 .....	83
Fig. 4.25	The transmitted signal if the input is 0 .....	83
Fig. 4.26	Design Summary of the first implementation of the BPSK Modulator on FPGA .....	83
Fig. 4.27	The resources map of the BPSK Modulator on the Spartan 3E board .....	84
Fig. 4.28	BPSK Modulator board .....	84
Fig. 4.29	The principle of the BPSK modulator on the FPGA .....	85
Fig. 4.30	The waveforms: (a) The modulating signal (b) Sine (c) The modulated signal .....	85
Fig. 4.31	The Design Summary of the second implementation of the BPSK Modulator .....	86
Fig. 4.32	The resources map of the BPSK modulator on the Spartan 3E Board .....	86
Fig. 4.33	BPSK System .....	87
Fig.4.34	Band Limited Noisy Channel .....	87
Fig. 4.35	The waveforms shown on the scope: (a) the modulated signal (b) the noise (c) the noise added to the modulated signal .....	87
Fig. 4.36	BPSK demodulator .....	88
Fig. 4.37	The signal at the output of the saturation block .....	88
Fig. 4.38	The signal at the output of the transfer block .....	88
Fig. 4.39	(a) The modulating signal (b) The demodulated signal .....	88
Fig. 4.40	BPSK Demodulator in System Generator .....	89
Fig. 4.41	(a) The modulating signal (b) The demodulated signal .....	89
Fig. 4.42	BPSK System .....	90
Fig. 4.43	BPSK System – experimental setup .....	90
Fig. 4.44	BPSK Demodulator board .....	90
Fig. 4.45	The principle of the BPSK demodulator on the FPGA .....	91
Fig. 4.46	The waveforms: (a) The modulated signal (b) The modulating signal (c) The demodulated signal .....	91
Fig. 4.47	Design Summary of the BPSK Demodulator .....	91
Fig. 4.48	The resources map of the BPSK demodulator on the Spartan 3E Board .....	92
Fig. 4.49	Comparison of the logic resources.....	93
Fig. 5.1	First step in implementing the QPSK modulation technique .....	96
Fig. 5.2	Second step of implementing the QPSK modulation technique .....	96
Fig. 5.3	QPSK Modulator in the Simulink environment .....	97
Fig. 5.4	The waveforms on the scope: (a) The modulating signal (b) $I(t)$ (c) $I(t) \cos 2\pi f_c t$ (d) $Q(t)$ (e) $Q(t) \sin 2\pi f_c t$ (f) The QPSK modulated signal .....	98
Fig. 5.5	QPSK Modulator in System Generator .....	99
Fig. 5.6	The waveforms: (a) the modulating signal; (b) $I(t)$ ; (c) $Q(t)$ .....	100
Fig. 5.7	The waveforms in the modulator: (a) The modulating signal; (b) $I(t)$ ; (c) $I(t) \cos 2\pi f_c t$ ; (d) $Q(t)$ ; (e) $Q(t) \sin 2\pi f_c t$ ; (f) The modulated signal .....	101
Fig. 5.8	The principle of the QPSK modulator on the FPGA .....	102
Fig. 5.9	The waveforms: (a) the modulating signal (LFSR) (b) the modulated signal .....	103

10 List of Figures

Fig. 5.10	The waveforms: (a) I-channel (b) Q-channel (c) the modulated signal .....	103
Fig. 5.11	The waveforms: (a) the modulating signal (LFSR); (b) I-channel; (c) Q-channel; (d) the modulated signal .....	103
Fig. 5.12	Design summary of the QPSK modulator .....	104
Fig. 5.13	The resources map of the QPSK modulator on the Spartan 3E Board .....	104
Fig. 5.14	QPSK System .....	105
Fig. 5.15	The waveforms shown on the scope: (a) the modulated signal; (b) the noise; (c) the noise added to the modulated signal .....	105
Fig. 5.16	QPSK demodulator in Simulink .....	105
Fig. 5.17	The waveforms in the demodulator: (a) The modulated signal with noise; (b) The I-channel; (c) The Q-channel; (d) The demodulated signal .....	105
Fig. 5.18	QPSK Demodulator in System Generator .....	107
Fig. 5.19	The waveforms: (a) the modulated signal with noise; (b) the demodulated I-channel; (c) the demodulated Q-channel .....	108
Fig. 5.20	The waveforms: (a) the modulating signal (b) the demodulated signal .....	108
Fig. 5.21	QPSK System – experimental setup .....	109
Fig. 5.22	The principle of the QPSK demodulator on the FPGA .....	110
Fig. 5.23	The waveforms: (a) The modulating signal; (b) The demodulated signal .....	110
Fig. 5.24	Design Summary of the QPSK Demodulator .....	111
Fig. 5.25	The resources map of the QPSK demodulator .....	111
Fig. 5.26	Comparison of the logic resources of the QPSK modulator and demodulator .....	112
Fig. 6.1	System Generator design flow for the hardware co-simulation technique .....	114
Fig. 6.2	Organization of the chapter .....	115
Fig. 6.3	BPSK hwcosim modulator .....	116
Fig. 6.4	The modulating and the modulated signals .....	116
Fig. 6.5	The design summary of the BPSK modulator .....	117
Fig. 6.6	Parallel between the resources used in implementing the BPSK modulator .....	117
Fig. 6.7	The BPSK hwcosim system .....	118
Fig. 6.8	The modulating signal, the BPSK modulated signal and the demodulated signal .....	119
Fig. 6.9	The design summary of the BPSK system .....	119
Fig. 6.10	Parallel between the resources used in implementing the BPSK system .....	120
Fig. 6.11	The QPSK hwcosim modulator .....	120
Fig. 6.12	I-channel, Q-channel and the QPSK modulated signals .....	121
Fig. 6.13	The design summary of the QPSK modulator .....	121
Fig. 6.14	Parallel between the resources used in implementing the QPSK modulator .....	122
Fig. 6.15	The QPSK hwcosim system .....	123
Fig. 6.16	The modulating signal, the QPSK modulated signal and the demodulated signal .....	122

Fig. 6.17	Design summary of the QPSK system .....	124
Fig. 6.18	Parallel between the resources used in implementing the QPSK system .....	124

## LIST OF TABLES

Table 2.1	Comparison of representative computing architecture .....	17
Table 2.2	Qualitative comparison of several DSP implementation technologies .....	18
Table 2.3	FPGA families, models, boards and reference manual .....	21
Table 2.4	Delays between consecutive signals of the six-phase generator ...	28
Table 2.5	Delays at different 6-pin connectors .....	29
Table 2.6	Real Rise Time for the six-phase generator .....	30
Table 2.7	Delays between consecutive signals for the eight-phase generator	32
Table 2.8	Delays at the two 40-pin connectors .....	33
Table 2.9	Real Rise Time for the eight-phase generator .....	33
Table 2.10	Advantages and disadvantages of the System Generator .....	38
Table 3.1	Abbreviations and names of the digital modulation techniques ....	51
Table 3.2	QPSK signals and a mapping to the messages .....	56
Table 3.3	QPSK signal coordinates .....	57
Table 3.4	Main previous work and practice in phase modulation .....	70
Table 4.1	Matlab code of the embedded function .....	77
Table 4.2	Comparison of previous work and practice in BPSK modulation on FPGA .....	93
Table 5.1	Comparison of previous work and practice in QPSK modulation on FPGA .....	113

## LIST OF ACRONYMS

A/D	Analog to Digital Converter
ASIC	Application-Specific Integrated Circuits
ASK	Amplitude Shift Keying
AWGN	Additive White Gaussian Noise Channel
BER	Bit Error Rate
BFSK	Binary Frequency Shift Keying
BPSK	Binary Phase Shift Keying
CLB	Configurable Logic Blocks
CPFSK	Continuous Phase Frequency Shift Keying
CPLD	Complex Programmable Logic Devices
DCS	Digital Communication System
DDS	Direct Digital Synthesis
DSP	Digital Signal Processing
DSPs	Digital Signal Processor
DVB-S	Digital Video Broadcasting-Satellite
D/A	Digital to Analog Converter
EDIF	Electronic Data Interchange Format
FPGA	Field Programmable Gate Array
GMSK	Gaussian Minimum Shift Keying
GPP	General-Purpose microProcessors
GPS	Global Positioning System
HDL	Hardware Description Language
HW	HardWare
hwcosim	Hardware Co-Simulation
I	Inphase Channel
IJF-OQPSK	Inter symbol-Interference/Jitter-Free Offset Quadrature Phase Shift Keying
IOB	Input-Output blocks
ISI	Inter Symbol Interference
IS-95	Interim Standard 95
I/O	Input/ Output
JTAG	Joint Test Action Group
LFSR	Linear Feedback Shift Register
LRC	Raised Cosine Pulse of Length L
LREC	Rectangular Pulse of Length L
LSRC	Spectrally Raised Cosine Pulse of Length L
LUT	Look-Up Tables
MASK	M-ary Amplitude Shift Keying
MFSK	M-ary Frequency Shift Keying
MHPM	Multi-h Phase Modulation
MPSK	M-ary Phase Shift Keying
MSK	Minimum Shift Keying
NCD	Native Circuit Description
NGC	Native Generic Circuit
NGD	Native Generic Database

## 14 List of Acronyms

---

NRE	Non-Recurring Engineering
OOK	Binary On-Off Keying
OQPSK	Offset Quadrature Phase Shift Keying
PLD	Programmable Logic Devices
PSK	Phase Shift Keying
Q	Quadrature Channel
QAM	Quadrature Amplitude Modulation
QORC	Quadrature Overlapped Raised Cosine Modulation
QOSRC	Quadrature Overlapped Squared Raised Cosine Modulation
QPSK	Quadrature Phase Shift Keying
Q <sup>2</sup> PSK	Quadrature Quadrature Phase Shift Keying
RC	Reconfigurable Computing
RFU	Reconfigurable IP-Cores
RHW	Reconfigurable HardWare
ROM	Read-Only Memory
SDARS	Satellite Digital Audio Radio Service
SHPM	Singhle-h Phase Modulation
SMSK	Serial Minimum Shift Keying
SNR	Signal-to-Noise Ratio
SQAM	Superposed Quadrature Amplitude Modulation
SQORC	Staggered Quadrature Overlapped Raised Cosine Modulation
SW	SoftWare
TFM	Tamed Frequency Modulation
TSI-OQPSK	Two-Symbol-Interval Offset Quadrature Phase Shift Keying
UCF	User Constraints File
UMTS	Universal Mobile Telecommunications System
VGA	Video Graphic Array
VHDL	VHSIC-Very High Speed Integrated Circuits Hardware Description Language
XPSK	Cross correlated Quadrature Phase Shift Keying
$\pi/4$ -QPSK	$\pi/4$ Quadrature Phase Shift Keying

# 1. MOTIVATION

## 1.1. Introduction

We live in the era of communication. Everything is audio or video and any information is transformed into analog or digital signal. The design of a communication system depends on the type of the used signal. The major advantage of using digital modulation techniques is that, the use of digital signals reduces the hardware utilization of any system, reduces the noise in any equipment and also reduces the interferences problems as compared to the analog signals.

In the object of digital communications, a remarkable development had happened in the last 20 years. Due to this grown, newer technologies and applications had emerged daily. The existing modulation/demodulation techniques need to be modified according to the new technologies.

Being a new field, the first FPGA implementations of the BPSK and QPSK techniques were quite recent and also made the subject of this thesis.

In the last years, FPGAs became an essential part in implementing DSP systems, especially in areas such as digital communications, since FPGAs have the ability to realize high-speed parallel operations, ideal for high-performance digital signal processing device. Since Xilinx has introduced System Generator, a system level tool for FPGA programming based on Matlab/Simulink environment, the gap in the DSP community had been reduced considerable since System Generator can be used in many ways: to explore an algorithm without translating the design into hardware, to simulate a design which is a part of something bigger or to implement any design in hardware. After reviewing the FPGA architecture and its functionality for DSP implementation, typical DSP applications that can be mapped to FPGA devices are: image processing [93], [107], video processing [93], [108], [139], audio and speech processing [76], [109], coding and wireless communications [23], [29], [45], [60], [95], [96], [111], [130], [144], and more recently in biomedical applications (wireless implants) [39], [182].

Advances in technology and new demands on the existing system have now led to use Matlab/ Simulink environment in simulating a GPS (Global Positioning System) system [15], [21], [25], [26], [45], [69] from a satellite transmitter to a receiver [138]. This kind of application was also done on FPGAs [25], [90]. Recent studies also show that Phase Shift Keying, especially BPSK achieves a better signal-to-noise ratio (SNR) and thus it is a frequently applied technique for underwater communications [7], [38], [87] [112], which also make the subject of my future work. The capabilities offered by the reconfigurable FPGA platform were promising for a robust communication system.

In the last few years, more studies about implementation of the BPSK technique on FPGA had appeared, but also of the QPSK technique. More details can be found later in this work.

A new technique, called Hardware Co-Simulation using FPGA appeared in the last two years. Studies using this new technique can be found in [50], [86], [98], [107], [137], [139]. The objective of the Hardware Co-Simulation technique is to accelerate an application. Therefore, an application is distributed in two or more

hardware and software parts. Those parts of the application which are not critical for the process are kept in software, while the critical parts are implemented in hardware [107].

The results for the compiled area are generated in hardware. This allows the compilation area to be tested in actual hardware and can speed up the simulation dramatically. This method enables building a hardware version of the model and several tests can be performed in order to verify the functionality of the system in hardware. The results can be verified by comparing the simulation output to the expected output.

As FPGA devices have become larger and faster, verifying functionality of costly ASIC designs in FPGAs has become an effective and economical method of verification. However, some ASIC structures cannot be directly implemented in an FPGA efficiently. At the same time, the development cost of an ASIC is increasing rapidly, making it less feasible to use ASIC devices for many cost-sensitive applications without extensive testing and simulation. Precision Synthesis helps ease the transition from ASIC to FPGA design by allowing the same HDL code and constraint syntax to be used. To obtain optimal performance, automatic conversions of ASIC design structures are utilized.

## 1.2. Thesis Outline

This thesis is organized as follows:

- **Chapter 2** presents the resources, the methods and the ways used in this research. In this chapter is also presented an overview of reconfigurable computing and the reasons why digital signal processing is well suited for reconfigurable computing.
- **Chapter 3** presents the backgrounds of a digital communication system, including its components, the communication channels and the criteria of choosing a modulation technique. Also, the backgrounds of the digital modulation techniques, underlining the BPSK and QPSK modulation schemes, are also discussed.
- **Chapter 4** presents the proposed BPSK modulator and system. Here are presented, in detail, all implementation steps of the designs. The BPSK modulator and system were simulated, tested and finally implemented on FPGA.
- **Chapter 5** presents the proposed QPSK modulator and system, which were the subject of simulation, testing and implementation on FPGA.
- **Chapter 6** describes the hardware implementation of the BPSK modulator and system, and also of the QPSK modulator and system using the hardware co-simulation technique.
- **Chapter 7** presents the contributions and conclusions of this work.



## 2. RESOURCES. METHODS. WAYS.

### 2.1. Reconfigurable Computing

#### 2.1.1. A Brief Overview of Reconfigurable Computing

A central preoccupation has always been the research in architecture of computer science. The investigation goals vary according to the target applications, the programmability of the system, the environment in which processors will be deployed and many others. The multiplicity of goals has led to the development of several processing architectures, each optimized according to a given goal.

Traditionally, computing was classified into general-purpose computing performed by a General- Purpose Processor (GPP) and application-specific computing performed by an Application-Specific Integrated Circuit (ASIC). Reconfigurable computing (RC) has combined the advantages of both. A comparison of the characteristics of the three different architectures is given in table 2.1 [68], [162], [164].

Table 2.1 Comparison of representative computing architecture [68]

Architecture	Resources	Algorithms	Performance	Cost	Power	Flexibility	Computing Model	NRE Cost
GPP	Fixed	Software	Low	Low	Medium	High	Mature	Low
ASIC	Fixed	Fixed	High	High	Low	Low	Mature	High
RC	Configware	Flowware	Medium	Medium	Medium	High	Immature	Medium

According to table 2.1, reconfigurable computing has the combined advantages of configurable computing resources, called configware, as well as configurable algorithms, called flowware. Further, the performance of reconfigurable systems is better than general-purpose systems and the cost is smaller than that of ASICs. Only recently the power consumption of reconfigurable systems has been improved such that it is now either comparable with ASICs or even smaller. The main advantage of the reconfigurable system is its high flexibility, while its main disadvantage is the lack of a standard computing model. The design effort in terms of non-recurring engineering (NRE) cost is between that of general-purpose processors and ASICs [43], [68]. At a fundamental level, reconfigurable computing is the process of best exploiting the potential of reconfigurable hardware [65].

In the past years, studies were made in order to demonstrate that reconfigurable computing can be well suited for DSP. In [162] survey, the reconfigurable computing platforms offer functional efficiency of hardware and programmability of software. A good hardware platform is when the balance between performance and flexibility is almost equal. According to [93], a good hardware platform should provide high computation throughput, low power consumption and small design area. It should also be easy to program and flexible

to changes [113]. These tradeoffs resulted in a four implementation solution that includes: ASIC, DSP, GPP and RC [93], [162]. Each of them has its place in the spectrum of DSP systems and applications and is represented in table 2.2.

Table 2.2 Qualitative comparison of several DSP implementation technologies [46]

Technology	Performance	System Design Cost	Cost per Chip	Power	Flexibility	Memory Bandwidth	I/O Bandwidth
GPP	Low	Low	Low-Medium	High	High	Low	Low
Micro-controller	Low	Low	Low	Medium	High	Low	Low
Programmable DSP	Medium	Medium	Low-Medium	Medium	Medium	Medium	Low
ASIC	High	High	Low	Low	Low	High	High
Reconfigurable Hardware	Medium-High	Medium	Medium-High	Medium-High	High	High	High

Reconfigurable computing became popular in the mid 1980s due to the availability of FPGA, although the concept of reconfigurable computing has existed since the 1960s, when Gerald Estrin proposed a new concept of a computer made of a standard processor and an array of "reconfigurable" hardware [40]. This gap was due to ignorance of the new technology and studies were made almost two decades later [14], [61] - [64], [153], [154], [165], [166]. So, at the end of 1990s and beginning of 2000s, the reconfigurable computing field was at its beginning [28], [162], [163].

The application fields in which reconfigurable computing had been applied are embedded systems, network security applications, multimedia applications, scientific computing. More detailed applications are presented in [46].

### 2.1.2. Field Programmable Gate Array (FPGA)

Field-programmable gate arrays (FPGAs) are truly revolutionary devices that blend the benefits of both hardware and software. They provide huge power, area, and performance benefits over software and can be easily reprogrammed to implement a wide range of tasks. Just like computer hardware, FPGAs implement tasks simultaneously, computing millions of operations in resources distributed across a silicon chip and can be hundreds of times faster than microprocessor-based designs [65]. The innovative development of the FPGAs is that it can be reprogrammed an unlimited number of times [46]. FPGAs provide nearly all of the

benefits of software flexibility and development models, and nearly all of the benefits of hardware efficiency. Compared to a microprocessor, these devices are typically several orders of magnitude faster and more power efficient, but creating efficient programs for them is more complex. Typically, FPGAs are useful only for operations that process large streams of data, such as signal processing, networking [65]. Compared to ASICs, they may be 5 to 25 times worse in terms of area, delay, and performance. However, while an ASIC design may take months to years to develop and have a multimillion-dollar price tag, an FPGA design might only take days to create and cost tens to hundreds of dollars.

In most FPGAs, the logic components can be programmed to duplicate the functionality of basic logic gates or functional intellectual properties. FPGAs also include memory elements composed of simple flip-flops or more complete blocks of memories [14]. Hence, FPGA has made possible the dynamic execution and configuration of both, hardware and software, on a single chip [68], [163] which made the FPGA the computational unit for reconfigurable computing systems [46].

FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. A hardware description language is any language from a class of computer languages used for description of digital logic. The HDL describes the circuits' operation, its design and behavior by simulating.

The most common HDLs are VHDL and Verilog [152]. Other programming languages used to program the FPGA are LabVIEW, MATLAB/Simulink [18], [52], [53], [79], [80]. A lot of work has been done in targeting existing programming languages like C, C++ and Java as the next hardware description language [167]. But, VHDL, Verilog, MATLAB/Simulink remain the popular ones.

### 2.1.2.1. FPGA Architecture

The architecture of an FPGA is based on the three main building blocks: configurable logic blocks (CLB), input-output blocks (IOB) and communication resources, illustrated in fig. 2.1. The FPGAs used throughout this work are all manufactured by Xilinx, so the following architectural details apply to the chips produced by this company [68].

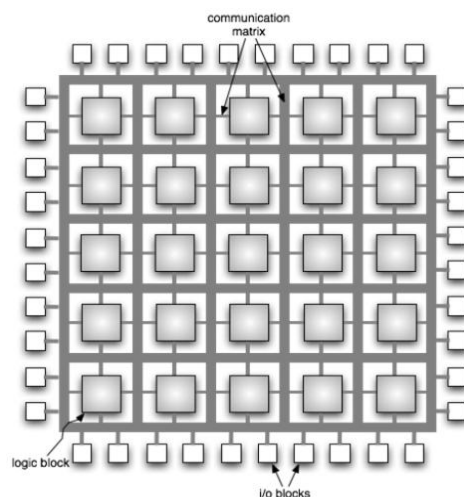


Fig. 2.1 Generic FPGA architecture [68]

CLBs are the main components of an FPGA. They can have one or more function generators realized with look-up tables (LUT) and can implement an arbitrary logic function according to their configuration. The result of the function is stored for every possible combination of the inputs, such that a 4-bit LUT will require 16 memory cells to store the function, no matter its complexity. Around the LUT there is the interconnect logic that routes signals to and from the LUT, implemented using standard logic gates, multiplexers, and latches. Therefore, during the configuration process of an FPGA, the memory inside the look-up tables is written to implement a required function, and the logic around it is configured to route the signals correctly in order to build a more complex system around this basic building block. In Xilinx FPGAs a single CLB contains a set of four slices that in turn contain two look-up tables and the necessary interconnect hardware [68].

The input output blocks (IOBs) have the function of interconnecting the signals of the internal logic to an output pin of the FPGA package. There is one and only one IOB for every I/O pin of the chip package. The IOBs have their own configuration memory, storing the voltage standards to which the pin must comply and configuring the direction of the communication on it, making it possible to establish mono-directional links in either way or also bidirectional ones [68].

The interconnection resources within an FPGA allow the arbitrary connection of CLBs and IOBs. The main modes of interconnections are direct and segmented and are illustrated in fig. 2.2. Direct interconnection (fig. 2.2a) is made of groups of connections that cross the device in all its dimensions. Logic blocks put data on the most suitable channel according to data destination. Segmented interconnection (fig. 2.2b) is based on lines that can be interconnected using programmable switch matrices.

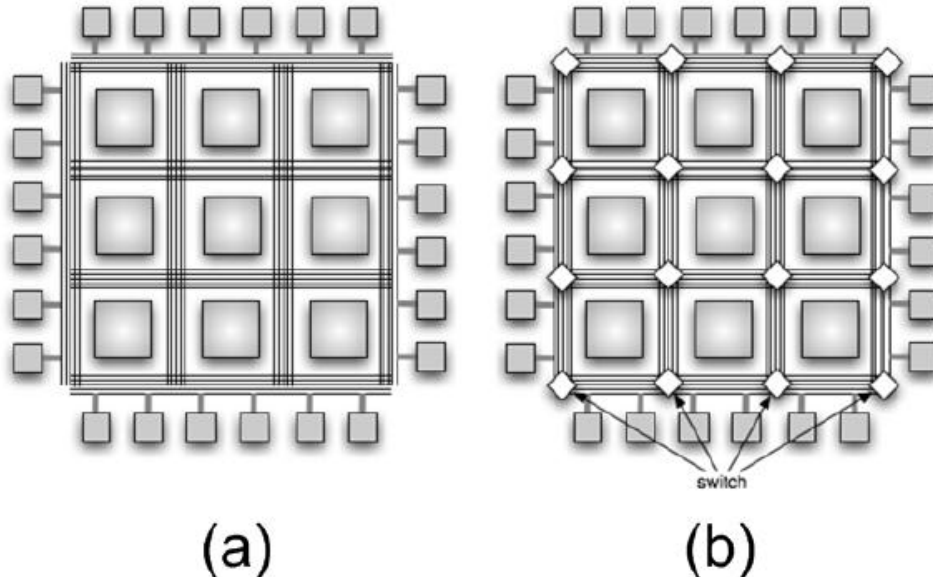


Fig. 2.2 FPGA using (a) the direct interconnection model and (b) the segmented interconnection model. [68]

Also in this kind of interconnection there are lines that cross the entire device, in order to maximize the speed of communication and limit signal skew. The main advantage of direct interconnection is that parasite resistance and capacitance

are almost constant, resulting in an improved predictability of the propagation times of the signals. Segmented interconnections offer reduced power dissipation because resistance and capacity of the interconnection lines are only those of the interconnection length between the blocks [68].

### 2.1.2.2. FPGA Families and Models

The FPGAs used in this research belong to two distinct families with different architectures and features. Table 2.3 lists the FPGA models used in my research.

Table 2.3 FPGA families, models, boards and reference manual used in my research.

Family	Board	Model
Spartan 3E	Nexys 2	XC3S500E
	Spartan 3E Starter Kit	
Virtex II Pro	Virtex II Pro	XC2VP20 XC2VP30

The two boards from the Spartan 3E FPGA family used in this research are Nexys2 [33] and Spartan 3E Starter Kit [176] and are illustrated in fig. 2.3.

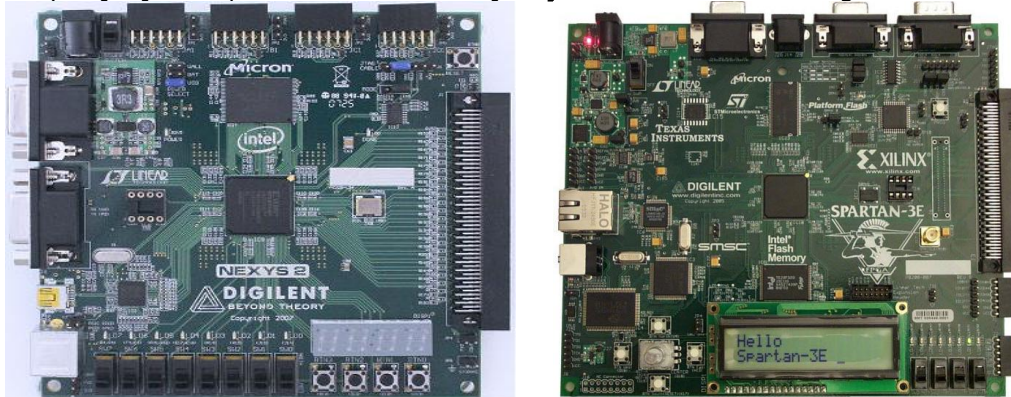


Fig. 2.3 Boards of the Spartan 3E FPGA family

a) Nexys2

b) Spartan 3E Starter Kit

The Spartan-3E family is built on the earlier success of Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. The Spartan-3E family of FPGAs is specifically designed to meet the needs of high volume, cost-sensitive consumer electronic applications. Because of their exceptionally low cost, Spartan-3E FPGAs are ideally suited to a wide range of consumer electronics applications, including broadband access, home networking, display/projection, and digital television equipment [178].

The XUP Virtex-II Pro Development System provides an advanced hardware platform that consists of a high performance Virtex-II Pro Platform FPGA surrounded by a comprehensive collection of peripheral components that can be used to create a complex system and to demonstrate the capability of the Virtex-II Pro Platform FPGA [174], [175]. The board can be seen in fig. 2.4.

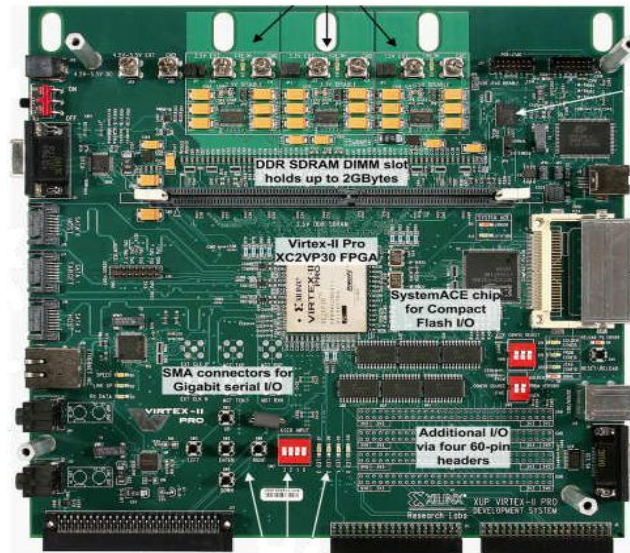


Fig. 2.4 Virtex-II Board

### 2.1.2.3. FPGA Design Flow

FPGAs are programmed using a logic circuit diagram or a source code in a HDL to specify how the chip will work. A hardware description language is any language from a class of computer languages used for description of digital logic. HDL design is supported with simulation, as with circuit schematic design entry, and with logic synthesis (normally referred to simply as synthesis), which converts (synthesizes) the HDL design description into a circuit netlist consisting of the required logic gates and interconnection wiring [48]. The HDL describes the circuit's operation, its design and behavior by simulating. The most common hardware language is VHDL (VHSIC-Very High Speed Integrated Circuits Hardware Description Language).

The key advantage of VHDL when it is used for systems design is that it allows the behavior of the required system to be described (modeled) and verified (simulated) before synthesis tools translate the design into real hardware (gates and wires). The gates and wires are mapped into a programmable logic device such as a CPLD (Complex Programmable Logic Devices) or FPGA [117]. Another benefit is that VHDL allows the description of a concurrent (parallel) system and this is the reason why VHDL is usually referred to as a code rather than a program [75], [89], [115].

A fundamental motivation to use VHDL is that VHDL is a standard and therefore it is portable and reusable. The two main applications of VHDL are in the field of PLD (Programmable Logic Devices) and ASIC [115].

In order to program an FPGA device in VHDL code, the ISE Web Pack software from Xilinx is used in all experiments from this work.

The steps followed during a project are illustrated in fig. 2.5.

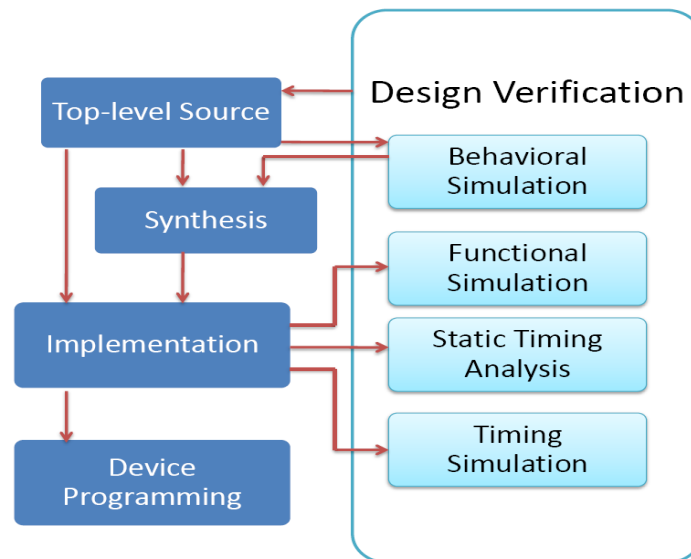


Fig. 2.5 FPGA Design Flow

**Top-level Source:** can be Schematic, Hardware Description Language and combination of both. The selection of a method depends on the design and designer. In this work, we are going to deal with the HDL based design entry. Simulations are performed when the HDL code is completed and the designer is ready to simulate the design [89].

**Synthesis:** is the process which translates VHDL code into a device netlist format. The resulting netlist is saved to a \*.NGC (Native Generic Circuit) or \*.EDIF (Electronic Data Interchange Format) file. The synthesis process checks the code syntax and analyzes the hierarchy of the design which ensures that the design is optimized for the design architecture.

**Implementation:** consists of a sequence of three steps

1. Translate
2. Map
3. Place and Route

1. Translate process combines all the input netlists and constraints to a logic design file. This information is saved as a \*.NGD (Native Generic Database) file. Defining constraints involves assigning ports in the design in order to be physical elements (e.g. pins, switches, buttons) of the targeted device and specifying time requirements of the design.

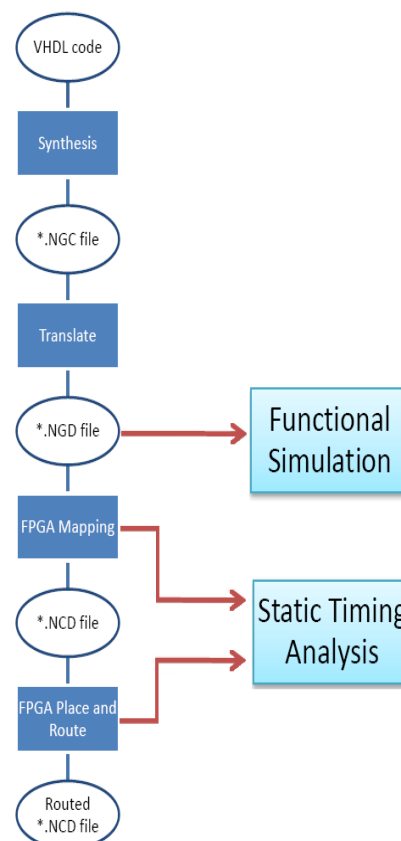


Fig. 2.6 FPGA Design Flow with files

The constraints information is stored in a file named \*.UCF (User Constraints File).

2. Map process divides the whole circuit into logical sub blocks so that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (CLB, IOB) and generates a \*.NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA.

3. Place and Route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Following this process, the input mapped \*.NCD file becomes a completely routed \*.NCD output file.

**Device Programming:** after all the above steps, the design can almost be loaded on the FPGA. In order to program the FPGA, the design must be converted into a format so that the FPGA can accept it. The BITGEN program deals with this conversion: the routed \*.NCD file is given to the BITGEN program to generate a bit stream, a \*.BIT file, which can be used to configure the target FPGA device.

**Design Verification:** can be done at different stages of the process steps.

1. Behavioral Simulation is the first of all simulation steps. This simulation is performed before the synthesis process to verify the RTL behavioral code and to confirm that the design is fully functional as it was intended. Behavioral simulation can be performed on either VHDL or Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set, but without timing information. The behavioral simulation is the fastest simulation but provides the least design information. It also allows the designer to change the HDL code if the required functionality is not met within a short time period.

2. Functional simulation (Post Translate Simulation) gives information about the logic operation of the circuit. The functionality of the design can be verified using the functional simulation process after the translate process. If the functionality is not as desired, then changes have to be made in the code and again follow the design flow steps.

3. Static Timing Analysis can be done after the map and post-and-route processes. The post-map-timing report lists signal path delays of the design derived from the design logic and the post-place-and-route timing report incorporates timing delay information to provide a comprehensive timing summary of the design.

If the system is supposed to be composed of three parallel parts: the hardware (HW), the reconfigurable (RHW) and the software (SW) sides, the design flow is illustrated in fig. 2.7.

The HDL Core Design and the IP-Core Generation are the first steps that have to be performed in the hardware and reconfigurable hardware sides. Here, the core functionalities of the original specification are translated in a hardware description language and extended with a communication infrastructure that makes it possible to interface them with a bus-based communication channel. After implementing these steps, the design of the static architecture is realized with the static components of the system, while the reconfigurable components are handled as reconfigurable IP-Cores (RFU). In order to implement the correct communication infrastructure between the static component and the reconfigurable ones, the component placement constraints need to be executed. Although the system description and the design synthesis and placement constraints assignment phases are shown separate, they are extremely interrelated; the system description phase is needed to generate the HDL architectural solution while the design synthesis and placement constraints assignment is needed in the identification and in the



corresponding assignment of the physical placement constraints of the final architectural solution.

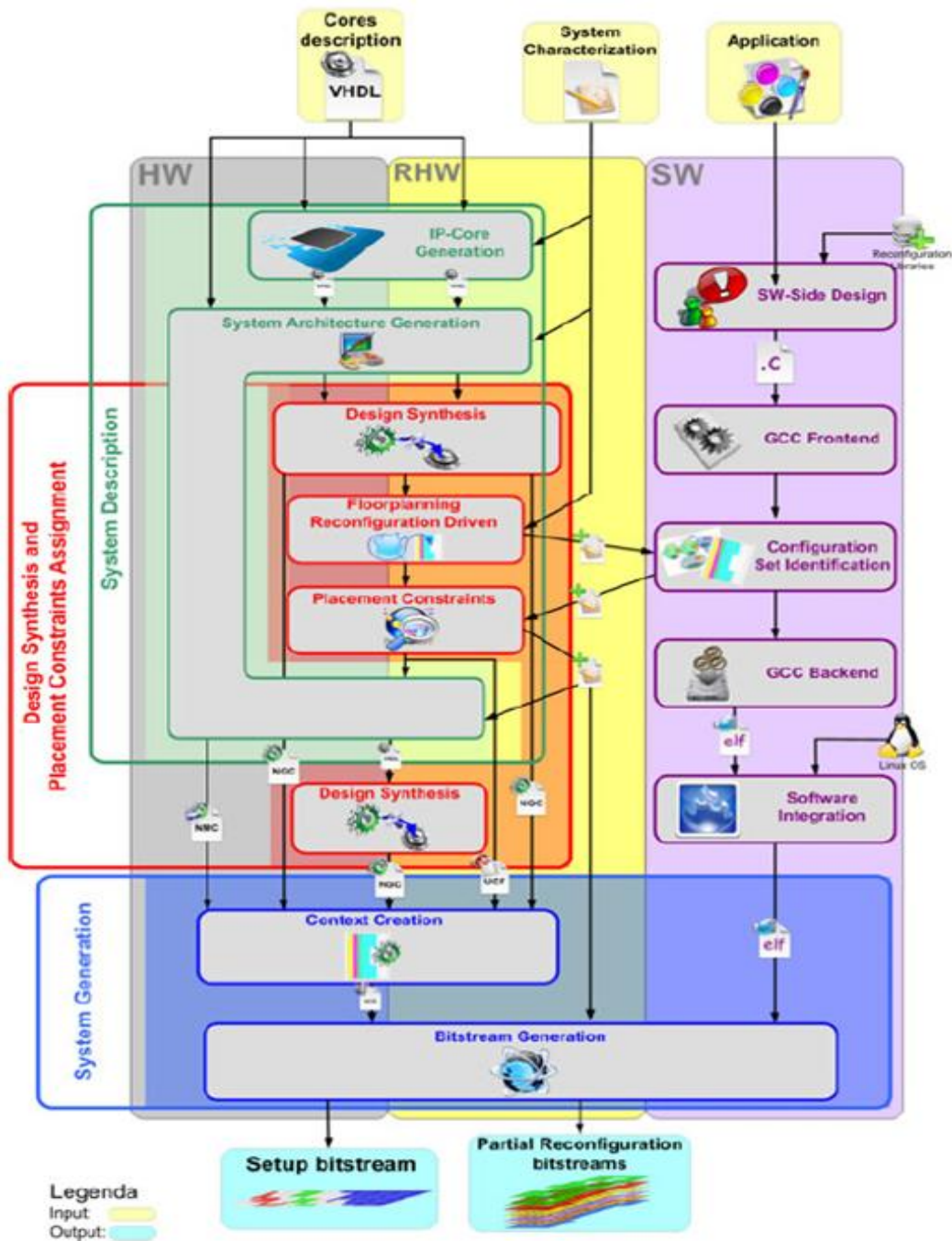


Fig. 2.7 The overall view of the described design flow [68]

In the reconfiguration management software part, there is the need to develop a set of drivers to handle both the reconfigurable and the static components of the system, in addition to a control application that is able to manage the reconfiguration tasks. All these software applications are compiled for the processor of the target system. Then, the compiled software is integrated in the Software Integration phase, with the bootloader, with the Linux OS and with the Linux Reconfiguration Support, which extends a standard operating system, making it possible for both to perform reconfigurations of the reprogrammable device and to manage the reconfigurable hardware as well as the static hardware, in order to allow component runtime plug-in. The following step is the Bitstreams Generation. This step is necessary to obtain the bitstreams that will be used to configure and to partially reconfigure the reprogrammable device. Finally, the last step of the design flow is the Deployment Design phase. Its goal is to create the final solution that consists of the initial configuration bitstream, the partial bitstreams, the software part and the deployment information that will be used to physically configure the target device [68].

## 2.2. Simulation Capabilities on Nexys2 and Virtex-II Boards. Case Study.

The goal of this study was to investigate the simulation capabilities of modern FPGA post-layout simulator (ISE from Xilinx) compared to real life behavior on two boards from Digilent: Nexys2 and Virtex-II. For this study, the test circuit was a simple ring counter as a phase generator and the delays between the outputs have been evaluated. The test circuit was a six-phase generator implemented for the Nexys2 board and an eight-phase generator implemented on the Virtex-II board [117], [118].

In order to verify the limits of the ISE Web Pack Suite, a sample design has been simulated, implemented and tested. ISE WebPACK from Xilinx is a software for FPGA design solution for Linux and Windows. ISE WebPACK is used for FPGA and CPLD design offering HDL synthesis and simulation, implementation, device fitting, and JTAG programming. The sample design is the six-phase generator for the Nexys2 board [117] and an eight-phase generator for the Virtex-II board [118].

The six/eight-phase generator is an application of shift registers in ring counters. A shift register is a group of flip flops set up in a linear fashion which have their inputs and outputs connected together in such a way that the data is shifted down the line when the circuit is activated. The ring counter illustrated in figure 2.8 is a type of counter composed of a circular shift register. The output of the last shift register is fed to the input of the first register. The VHDL programs of the six/eight phase generator are presented in [117] and [118].

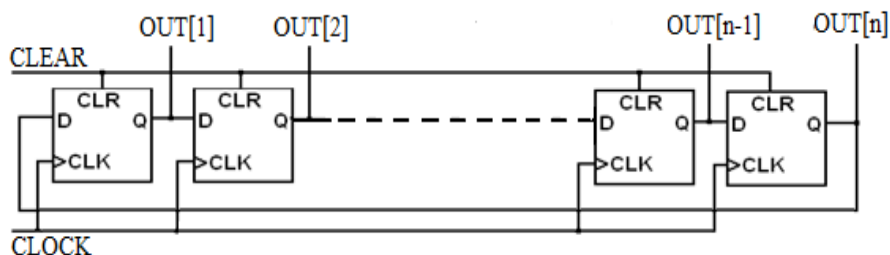


Fig. 2.8 A n bits ring counter

An internal active-high signal vector, R\_W, is used for reading and writing what eventually becomes the circuit’s output; this internal signal is conveniently inverted in the last statement to obtain the required active-low output signal vector [168]. Aux is an auxiliary state bit to keep track of the two states within each phase. The OUTi and Aux signals are all outputs from the flip-flops clocked by the same clock CLK. OUT2 goes low before Aux goes high.

This design is simulated, first behavioral, then post-route.

### 2.2.1. Simulations on Nexys2 board

To test the design functionality, the behavioral simulation is used. The signals obtained in the behavioral simulation are shown in figure 2.9. Signal CLK is the clock and the signals: Reset, Start and Restart are control inputs. The OUT signals are active-low phase outputs [176].

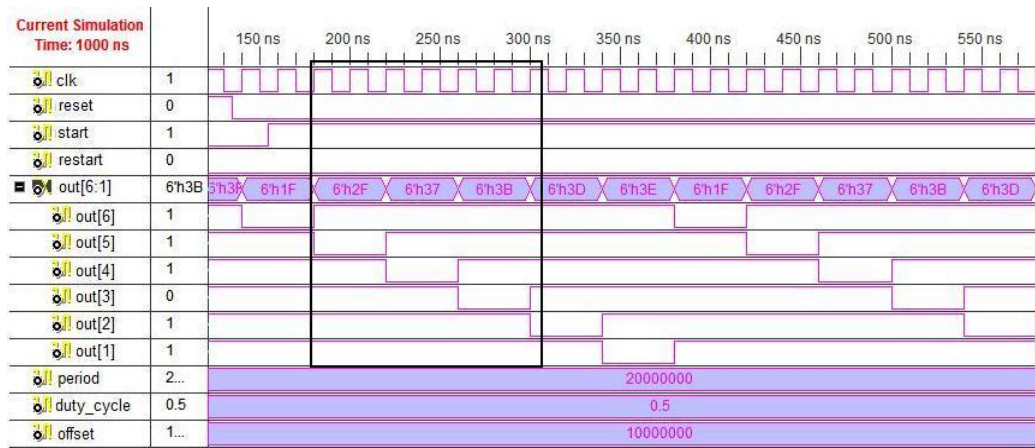


Fig. 2.9 Behavioral Simulation of the six-phase generator [117]

The signals in the rectangle are zoomed and show below, in figure 2.10.



Fig. 2.10 Zoomed area of the Behavioral Simulation [117]

As expected, the signals from figure 2.10 show that the behavioral simulation does not emphasize any delays.

In the post-route simulation, unlike the behavioral simulation, the delays became visible. Fig. 2.11 illustrates the post-route simulation and fig. 2.12 shows the delay between two known signals.

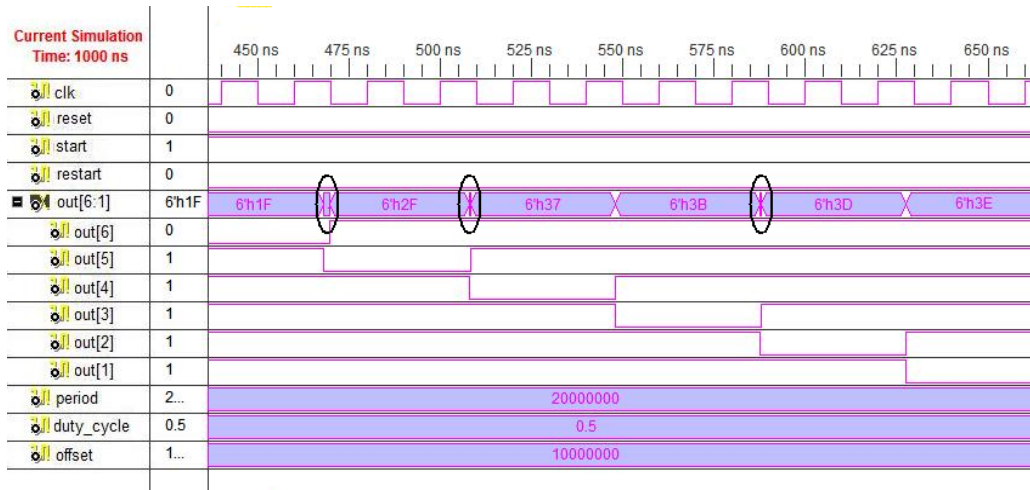


Fig. 2.11 The post-route simulation of the six-phase generator [117]

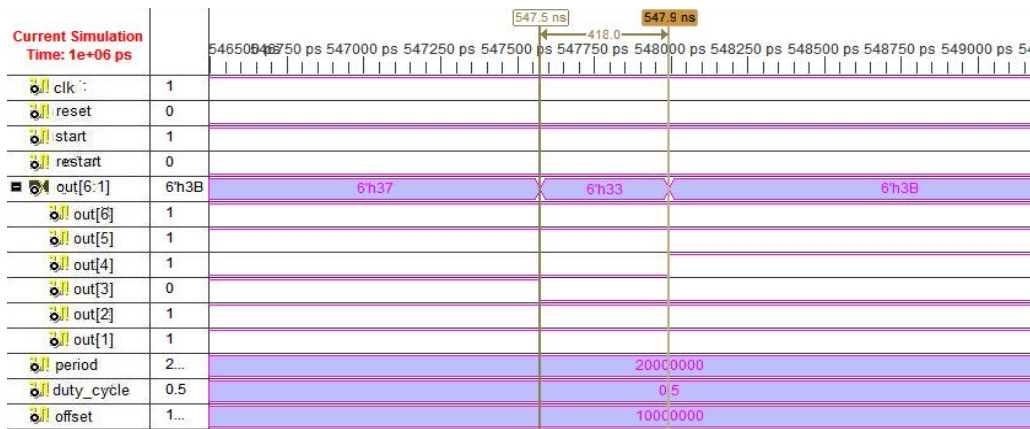


Fig. 2.12 Zoomed delayed detail between out[3] and out[4] [117]

The delays between every two consecutive signals are listed in Table 2.4.

Table 2.4 Delays between consecutive signals of the six-phase generator [117].

Signals	1_2	2_3	3_4	4_5	5_6	6_1
Delays[ps]	32	254	418	645	56	459

After computing the delays from the post-route simulation, the purpose was to estimate these delays with a LeCroy WaveSurfer Xs Series Oscilloscope, a high

performance digital oscilloscope [88]. For this, the output signals were routed to one of the four 6-pin header connectors from de Nexys2 board. Each connector provides  $V_{dd}$ , GND and four unique FPGA signals. All four 6-pin header circuits have short circuit protection resistors and ESD protection diodes. The first measurement was, with the signals routed to the first 6-pin connector. For example, fig. 2.13 illustrates the delay between the third and the fourth output signal. The delay is calculated with the function skew of the digital oscilloscope. The skew function is defined as: time of clock1 edge minus time of the nearest clock2 edge which represents exactly the delay between the two signals. To make this possible, one of the signals is inverted.

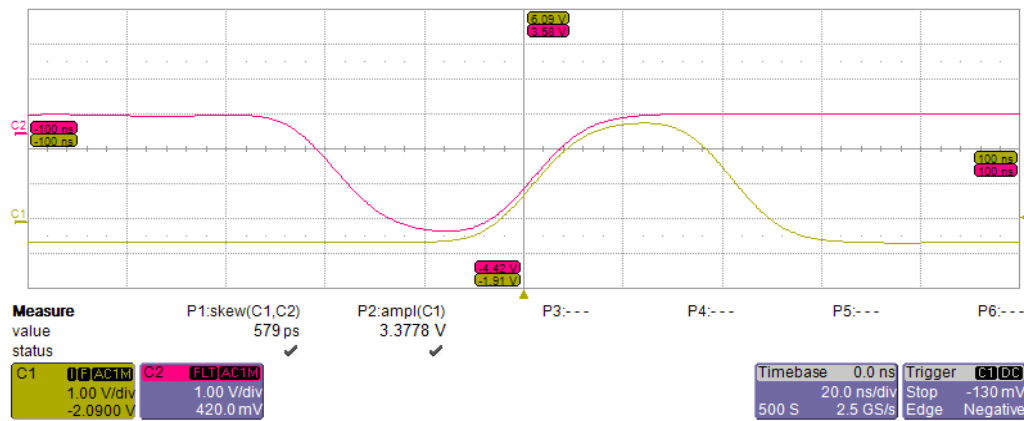


Fig. 2.13 The measured delay between the third and the fourth signal [117]

After measuring different delays between consecutive signals and after routing the signals to different 6-pin connectors, in table 2.5 are illustrated some of the most effective delays.

Table 2.5 Delays at different 6-pin connectors [117]

Delay Signals	Post-Route Simulation [ps]	First 6-pin connector [ps]	Different first 6-pin connector [ps]	Second 6-pin connector [ps]
3_4	418	579	405	405
4_5	645	670	516	1040

The measured rise time is commonly related to the system rise time and the signal rise time using the formula [1], [2], [161]:

$$t_{measured}^2 = t_{signal}^2 + t_r^2 \tag{2.1}$$

The risetime is a measured parameter that provides considerable insight into the potential pitfalls in performing a measurement or designing a circuit. Risetime is defined as the time it takes for a signal to rise (or fall for falltime) from

10% to 90% of its final value. The relationship between the rise time and the bandwidth of an oscilloscope is given by:

- for an analog oscilloscope:

$$t_r [\mu s] = \frac{0.35}{B [MHz]} \quad (2.2)$$

- for a digital oscilloscope:

$$t_r [\mu s] = \frac{N}{B [MHz]}, \quad N = 0.4 \quad \text{to} \quad 0.5 \quad (2.3)$$

The relationship (2.3) is useble only if  $t_{measured} \geq t_r$ . Otherwise, the

errors make the correction useless. Sometimes this relationship is used to estimate the actual signal rise time when the oscilloscope's system rise time is not sufficiently faster than the signal's rise time to make an accurate measurement [1], [2], [161].

From the technical manual of the LeCroy digital oscilloscope [88] that we used, N is 0.4, which means that the rise time is 400ps. The real rise time is shown in table 2.6.

Table 2.6 Real Rise Time for the six-phase generator [117].

Delay [ps] Signals	Measured Rise Time [ps]	Oscilloscope Rise Time [ps]	Real Rise Time [ps]
3_4	579	400	418
4_5	670	400	537

After simulating the six-phase waveform generator and implementing it on the Nexys2 board, the summary is illustrated in fig. 2.14 [117].

Device Utilization Summary				<a href="#">H</a>
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	7	9,312	1%	
Number of 4 input LUTs	6	9,312	1%	
<b>Logic Distribution</b>				
Number of occupied Slices	9	4,656	1%	
Number of Slices containing only related logic	9	9	100%	
Number of Slices containing unrelated logic	0	9	0%	
<b>Total Number of 4 input LUTs</b>	<b>6</b>	<b>9,312</b>	<b>1%</b>	
Number of bonded IOBs	10	232	4%	
Number of BUFGMUXs	1	24	4%	

Fig.2.14 Design Summary of the six-phase generator [117]

**2.2.2. Simulations on Virtex-II Board**

The same experiments were made on the Virtex-II board, but for an eight-phase generator.

The simulation for the behavioral test is illustrated in fig. 2.15, in order to test the design functionality. The signals in the selected area are zoomed and shown in fig. 2.16.

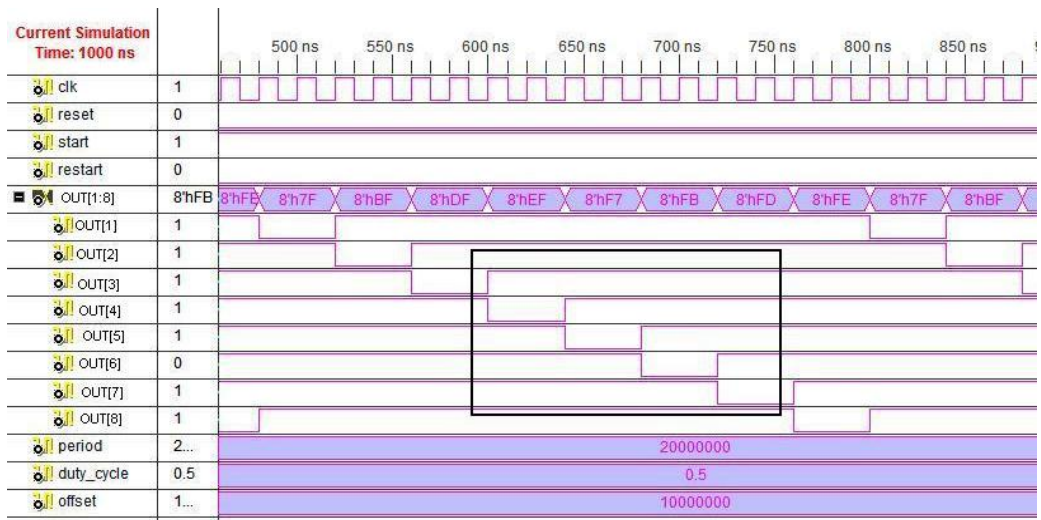


Fig. 2.15 Behavioral Simulation of the eight-phase generator [118]



Fig. 2.16 Zoomed area of the Behavioral Simulation [118]

As demonstrated in the six-phase generator, the behavioral simulation of the eight-phase generator does not point out any delays.

Unlike the behavioral simulation, in the post-route simulation, the delays became noticeable. The encirclements from figure 2.17 are the delays introduced by the post-route simulation. One of these delays is shown in figure 2.18, namely the delay between the third and the fourth signal.

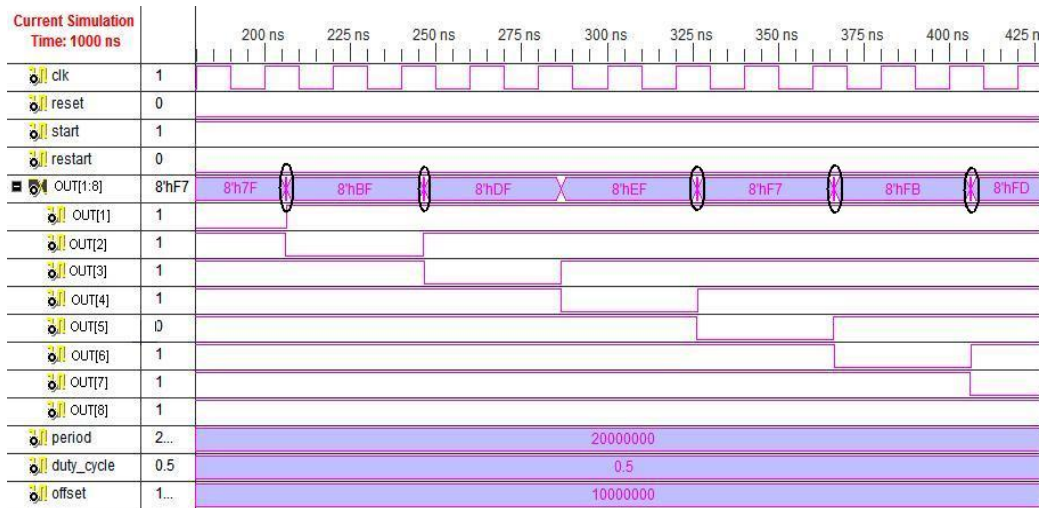


Fig. 2.17 The Post-Route Simulation of the eight-phase generator [118]

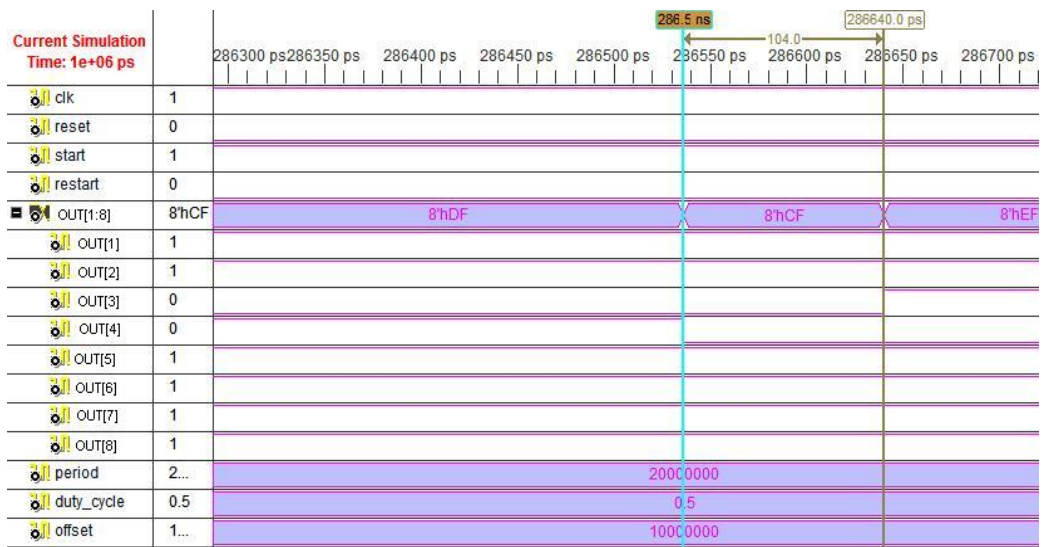


Fig.2.18 Zoomed delayed detail between out[3] and out[4] [118]

The delays between every two consecutive signals are listed in table 2.7.

Table 2.7 Delays between consecutive signals for the eight-phase generator [118].

Signals	1_2	2_3	3_4	4_5	5_6	6_7	7_8	8_1
Delays[ps]	238	527	104	376	117	208	106	176



In order to obtain results, the eight signals were routed to the two 40-pin right angle connectors and measured with a LeCroy oscilloscope. The first measurement was with the eight signals routed to the left expansion connector and the second measurement was with the signals routed to the right connector. Fig. 2.19 illustrates the delay between the third and the fourth signal. The delay is estimated with the skew function of the digital oscilloscope.

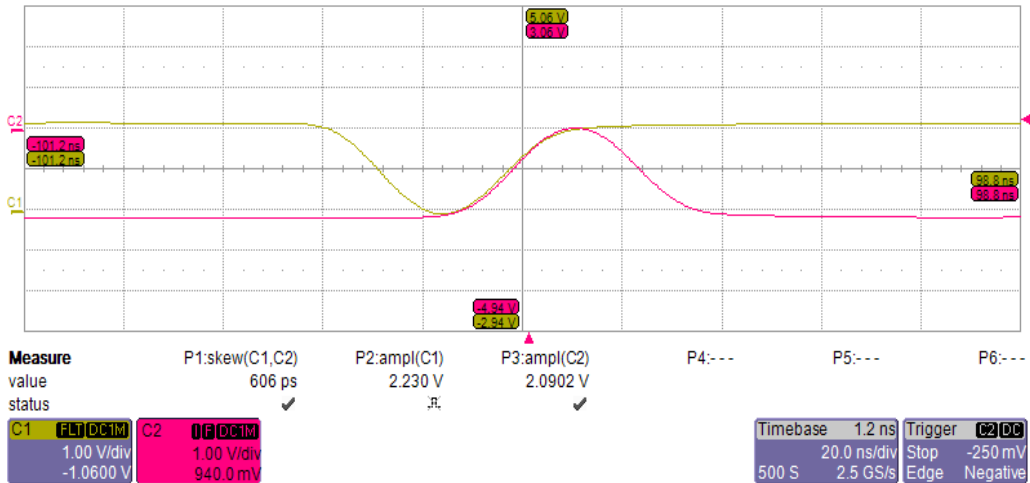


Fig. 2.19 The measured delay between the third and the fourth signal [118]

Computing all the delays between every two consecutive signals, table 2.8 illustrates some of the most effective delays. And after using formula (2.3), the real rise time is shown in table 2.9.

Table 2.8 Delays at the two 40-pin connectors [118]

Delay [ps] / Signals	Post-Route Simulation [ps]	First 40-pin connector [ps]	Second 40-pin connector [ps]
3_4	104	606	451
5_6	117	438	1051

Table 2.9 Real Rise Time for the eight-phase generator [118]

Delay [ps] / Signals	Measured Rise Time [ps]	Oscilloscope Rise Time [ps]	Real Rise Time [ps]
3_4	606	400	456
5_6	438	400	178

The simulation report of the eight-phase generator is illustrated in figure 2.20 [118].


Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	9	27,392	1%	
Number of 4 input LUTs	5	27,392	1%	
<b>Logic Distribution</b>				
Number of occupied Slices	9	13,696	1%	
Number of Slices containing only related logic	9	9	100%	
Number of Slices containing unrelated logic	0	9	0%	
<b>Total Number of 4 input LUTs</b>	<b>5</b>	<b>27,392</b>	<b>1%</b>	
Number of bonded <a href="#">IOBs</a>	12	556	2%	
Number of BUFGMUXs	1	16	6%	

Fig.2.20 Design Summary of the eight-phase generator

In managing short times, extra care should be taken. The oscilloscope's bandwidth should be at least five times higher than the fastest clock rate in the design. Otherwise, in order to make accurate edge-speed measurements on the signals, you will need to determine the maximum practical frequency present in the signal [118].

## 2.3. Reconfigurable Computing's Suitability for Digital Signal Processing

### 2.3.1. Digital Signal Processing

Nowadays, there is an increasing need to process, interpret and understand information (e.g. speech, music, image, video). In a lot of cases, the aim is to process data in order to obtain parts of a signal.

In the early days of electronics, signals were transmitted in their typical form, analog. The signals created from an analog source, were converted into electrical signals and then transmitted across a channel [19]. For a long time, analog techniques were the methods of choice when processing signals, but as computers and digital hardware have advanced, the use of DSP techniques replaced the analog ones [46]. The algorithms used in image and signal processing, multimedia, telecommunications, cryptography, networking and computation domains in general were first used on Digital Signal Processors (DSPs) or General Purpose Processors (GPPs) [132]. Digital signal processing is all about mathematics, algorithms and techniques, used to manipulate all signals after they have been converted into digital form.

The digital techniques replaced the analogs' for a many of reasons, such as [46], [105]:

- the increased immunity to changes in external parameters such as age and temperature as well as to variations in circuit components;
- the ease of reproducing DSP systems;

- an internal active-high signal vector,  $R\_W$ , is used for reading and writing what the flexibility in precision through changing word lengths and/ or numeric representation;
- the ability to use a single processing element to process multiple incoming signals through multiplexing;
- the ease with which signals digital approaches can adjust their processing parameters, such as with adaptive signal processing;
- the ideal characteristics that are only possible through digital techniques.

Also, digital hardware is more reliable and superior than its' analog counterpart which is prone to ageing and give uncertain performance in production [19].

In contrast, the disadvantages of using digital techniques over analog techniques include: system complexity, power consumption and frequency range limitation. For many applications, the advantages of DSP make up for the disadvantages [46]. The substitution of the analog techniques by the digital ones has been driven by the cheap hardware which can be interfaced with computers or even implemented on computers [19].

DSP functions are commonly implemented on two types of programmable platforms: DSPs and FPGAs. While DSPs are a specialized form of microprocessor, FPGAs are a form of highly configurable hardware [8], [183]. Specialized DSPs can implement many applications, such as: 3G wireless, multimedia systems, medical systems radar and satellite systems, consumer electronics, image-processing applications. Although DSPs are programmable through software, their hardware architecture is not flexible. The DSPs's fixed hardware architecture is not suitable for some applications that require customized DSP function implementations. On the other hand, FPGAs provide a reconfigurable solution for implementing DSP applications, higher DSP throughput, and more raw data processing power than DSPs. FPGAs are configured in hardware, therefore they offer complete hardware customization while implementing various DSP applications [9].

Some of the common operations used on digital or analog techniques are [46], [83]:

- elementary time-domain operations: amplification, integration, differentiation, addition of signals, multiplication of signals;
- filtering;
- transforms;
- convolution;
- modulation and demodulation;
- signal generation.

Also, some combinations of these operations can be used to remove noise from signals, prepare signals for wireless transmission, extract signals from wireless transmissions or select signals of interest from a larger collection of signals [46].

### 2.3.2. System Level Tools for DSP in FPGAs

In the last years, FPGAs became an essential part in implementing DSP systems, especially in areas such as digital communications, since FPGAs have the ability to realize high-speed parallel operations, ideal for high-performance digital signal processing device. The VHDL programming language is different from serial programming on microprocessor, so the developers often use the MATLAB language for system modeling and simulation. Xilinx has introduced a system level FPGA development tool based on Matlab/Simulink, which is System Generator [5], [31], [32], [181].

Simulink is a graphical environment, integrated with MATLAB [70], [85], which provides immediate access to a large range of tools that develop algorithms, analyze and visualize simulations, create batch processing scripts, customize the modeling environment, and define signal, parameter, and test data. Signals are time-varying quantities represented by the lines connecting blocks and parameters are coefficients that help define the dynamics and behavior of any system. Simulink provides tools for hierarchical modeling, data management and subsystem customization, making it easy to create concise, accurate representations, regardless of the system's complexity. Every system in Simulink can be simulated in order to see its dynamic behavior and to view the results. A variety of time-varying systems can be designed, simulated, implemented and tested, in many fields, including communications, controls, signal processing, video processing and image processing.

System Generator is a DSP tool from Xilinx based on the Matlab/Simulink environment and is used for FPGA design. System Generator is actually a library in Simulink which translates a Simulink model into a hardware realization of the same model. It also maps the system parameters defined in Simulink into entities and architectures, ports and signals in a hardware realization. In addition, System Generator produces command files for FPGA synthesis, HDL simulation and implementation tools in order to obtain the hardware model. Only the subsystems and blocks from the Xilinx Blockset are translated by System Generator into hardware realization. All implementation steps, including synthesis, place and route are automatically performed to generate the FPGA programming file [129]. System Generator supports bit and cycle true modeling, and automatically generates an FPGA implementation from a system model. In addition, it provides access to auxiliary tools for filter design, data analysis, visualization, and testbench creation [12], [71], [86], [146], [179].

The System Generator design flow is shown in the fig.2.21.

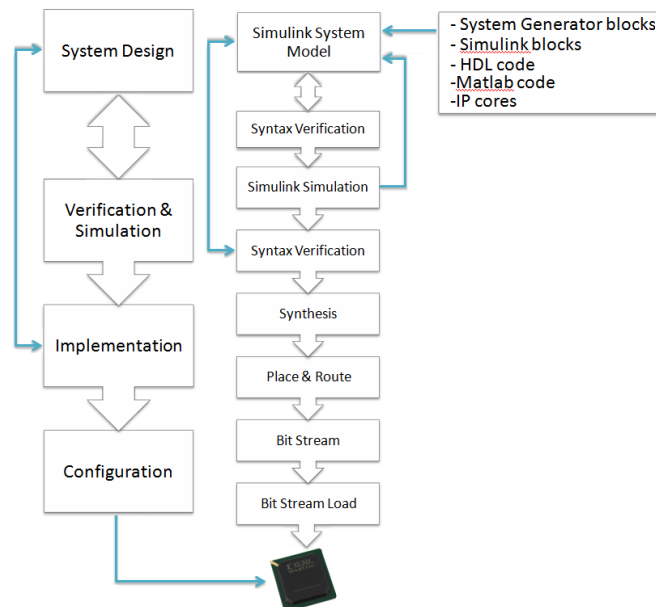


Fig.2.21 System Generator design flow

In order to program the FPGA, two distinct software packages are used: Matlab and Xilinx ISE. Matlab/Simulink is the software where the system functionality is verified and where the programming takes place and ISE is where the program will be configured to run on the FPGA. The main bridge between the two packages is System Generator, the part of Matlab, which converts the Simulink math code into VHDL code that is recognized by the ISE software [129]. System Generator models the algorithm in the environment of Matlab/Simulink and generates the corresponding ISE project. The generated project is simulated and synthesized in ISE environment. Finally a bit stream file is generated for FPGA programming. The details of the development process are shown in fig. 2.22.

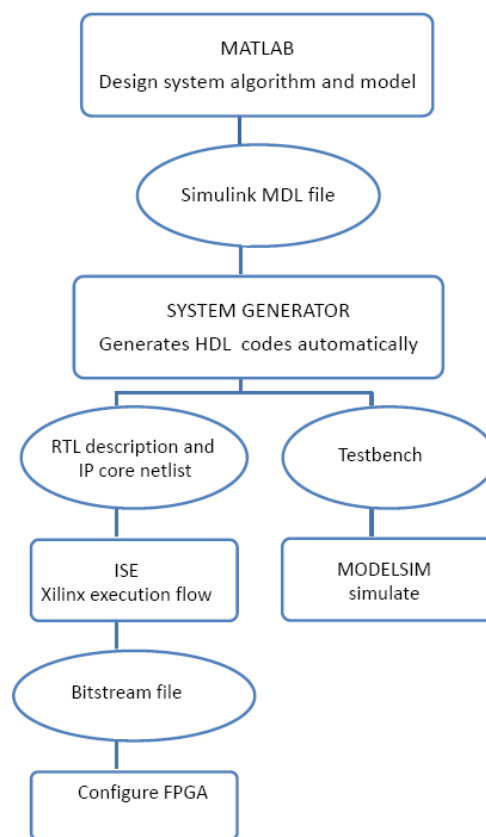


Fig.2.22 FPGA development process based on System Generator

Working with System Generator, some of advantages and disadvantages of this environment were established and are presented in table 2.10.

## Advantages

Any circuit or system developed in System Generator can be simulated in Matlab.

Being a graphical environment based on Simulink and having predefined blocks of Xilinx core, all components of a design can be verified in hardware.

The capability of the bit-true and cycle-accurate simulation for DSP, due to which the design can be validate before implementing it on hardware.

The synthesis process can be made using HDL code generated by System Generator; also testbench and test vectors can be made in order to verify a design.

Due to its generic to any Xilinx FPGA device, by simply changing one single parameter, the results of a board can be compared with the results of another one.

System Generator provides a unique hardware in the loop co-simulation which greatly accelerates simulation while simultaneously verifying the design in hardware.

System Generator provides access to underlying FPGA resources through low-level abstractions, along with the high level abstraction, allowing efficient FPGA designs.

## Disadvantages

Not all Simulink blocks are available in Xilinx Blockset and there is no way to generate hardware from designs which use the basic Simulink Blockset.

The Xilinx blocks do not provide as much flexibility as their counterparts in Simulink Blockset.

System Generator requires a lot of the computer resources and the generation step can be a problem for larger models.

The number of blocks and functions of the blocks are also limited today.

The time needed in order to simulate design with Xilinx components is longer than the time needed to simulate in Simulink. The simulation with Simulink blocks is made in one simulation step, while the simulation with Xilinx blocks can only process one piece of data in one simulation step.

There is a significant learning curve in using tools such as XSG which is rather undesired for many situations such as the undergraduate system-level design experience in universities [Martinez-Torres06].

## 2.4. Application in Matlab/Simulink and System Generator

The following application was achieved in order to introduce and understand the basic concepts of creating a design using System Generator within the model based design flow provided through the Matlab/Simulink environment, in order to understand the future applications.

The tested design is a simple circuit which adds a constant and a sinus generator.

The input signals are:

- constant:  $s_{i1}(t) = 1$  (2.5)
- sinus:  $s_{i2}(t) = A \sin(\omega t + \varphi)$

where:  $A = 1$  - amplitude of the signal  
 $\varphi = 0^\circ$  - initial phase of the signal  
 $\omega = 2\pi f$ ;  $f = 1 \text{ rad/s}$  - frequency of the signal  
 $s_{i2}(t) = \sin(2\pi t)$  (2.6)

The output signal adds the sinus and the constant:

$$s_o(t) = s_{i1}(t) + s_{i2}(t) = 1 + \sin(2\pi t) \quad (2.7)$$

First, the application is designed, simulated, implemented and tested in the Matlab/Simulink environment with only Simulink blocks. Afterwards, the same

application is modified in the same Matlab/Simulink environment, but with Xilinx blocks which makes the model implemented in System Generator.

### 2.4.1. Adder in Simulink

This design is an executable specification creating in Simulink using the standard blockset. It is a simple adder circuit, but serves to demonstrate many of the key concepts of model based design.

Fig. 2.23 illustrates the Simulink model of the adder and fig. 2.24, the corresponding signals. The constant waveform  $s_{i1}(t)$  from equation (2.5) and shown in fig. 2.24 is added with the  $s_{i2}(t)$  sinus waveform, exactly as the one from the equation (2.6) and shown in fig. 2.24, obtaining the  $s_o(t)$  output signal represented in fig. 2.24.

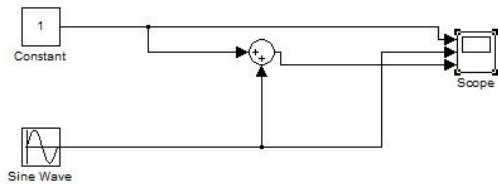


Fig.2.23 The Simulink model of the adder

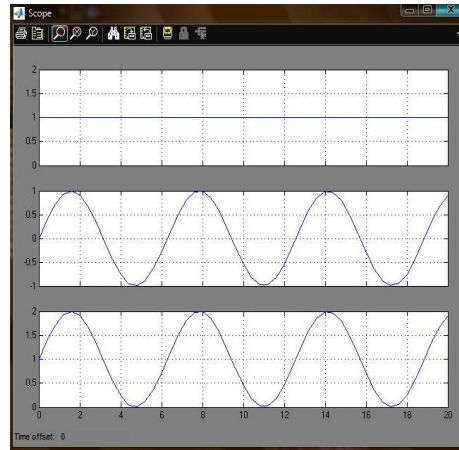


Fig. 2.24 Corresponding signals  
1-constant waveform 2-sinus waveform 3-output waveform

### 2.4.2. Adder in System Generator

The same principle of the adder is applied in System Generator, but with Xilinx blocks. Four implementations of the design were made in order to demonstrate the system modeling concept.

The first implementation of the adder, fig. 2.25, is made out of both Simulink and System Generator blocks. Both, the constant waveform and the sine waveform are generated external, outside the FPGA, and then passed through the Gateway Out ports in order to be seen on the scope.

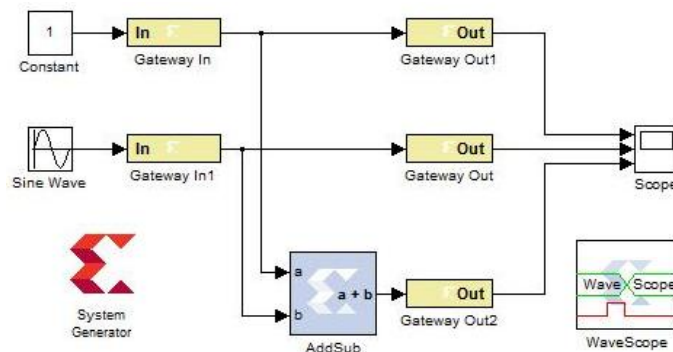


Fig. 2.25 First implementation of the System Generator adder

The Simulink blockset contains:

- the *Constant* block – used to define a real or complex real value. The output of the block has the same dimension and elements as the constant value parameter, which here is 1.
- the *Sine Wave* block – generates a sine waveform.
- the *Scope* – displays its input with respect to simulation time. The scope can adjust the amount of time and the range of input values displayed.

The System Generator blockset contains:

- the *Gateway In* blocks: the inputs into the Xilinx portion of the Simulink design;
- the *Gateway Out* blocks: the outputs from the Xilinx portion of the Simulink design;
- the *AddSub* block: implements an adder;
- the *Wavescope*: allows the signals to be viewed without the use of the Gateway Out ports.

Fig. 2.26 and 2.27 illustrate the corresponding signals. Fig. 2.26 represents the signals from the scope block, while fig. 2.27 represents the signals from the wavescope block and both are the same.

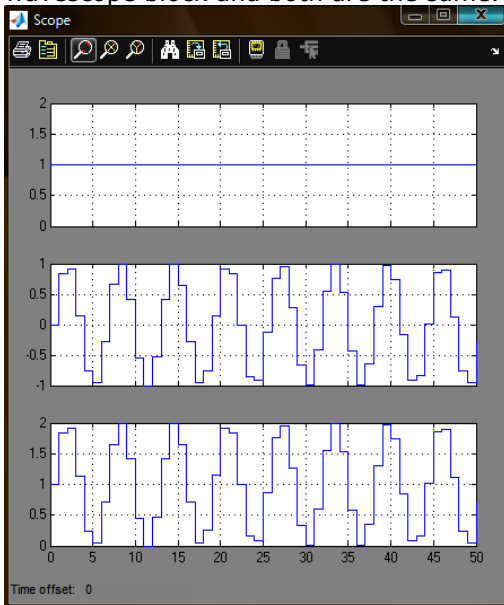


Fig. 2.26 The corresponding signals from the scope block:  
 a – constant waveform    b – sine waveform  
 c – output signal

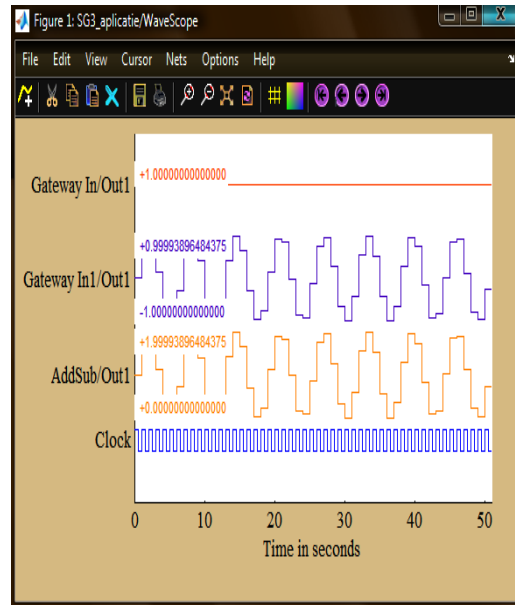


Fig. 2.27 The corresponding signals from the waveform block:  
 a – constant waveform    b – sine waveform  
 c – output signal

The sine waveforms have square edges. This is because the System Generator blockset forces a discrete sampling of the input signals which represents the behavior of the actual hardware which operates on synchronous clock edges.

In the second implementation of the adder, fig. 2.28, the Simulink blockset is made only of the *Sine Wave* and the *Scope* block. The System Generator contains the blocks from the first implementation and the *Constant* block. The *Constant* block is similar to the Simulink constant block, but can be used to directly drive the inputs



on Xilinx blocks. Fig. 2.29 illustrates the corresponding signals of the design from the wavescope block.

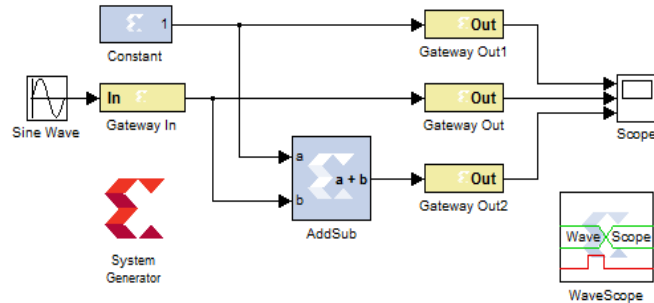


Fig. 2.28 Second implementation of the System Generator adder

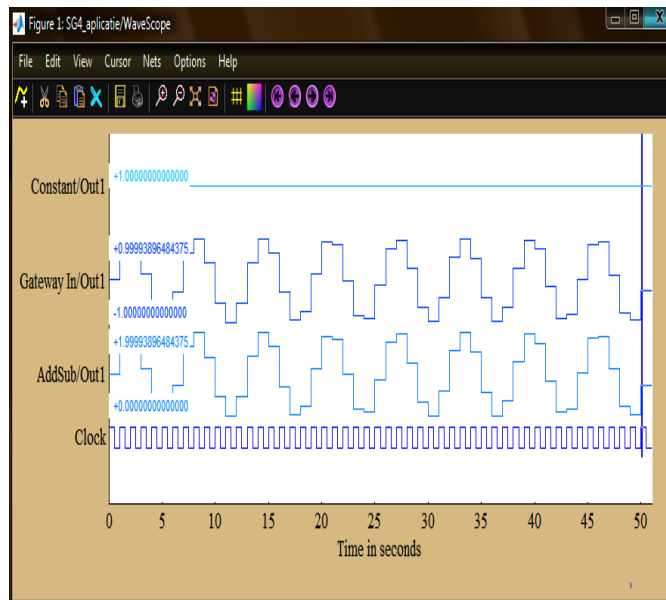


Fig. 2.29 The corresponding signals: a – constant waveform b – sine waveform c – output signal

The third implementation of the adder is illustrated in fig. 2.30.

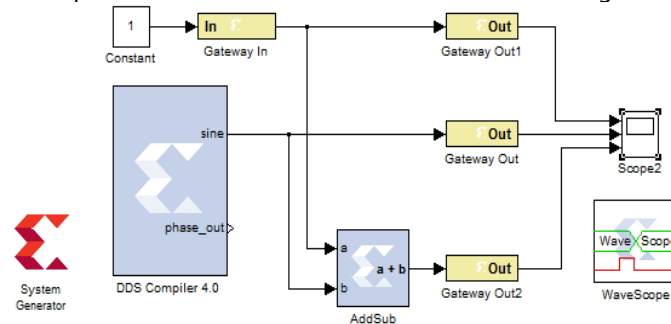


Fig. 2.30 The third implementation of the System Generator adder

The third implementation is made in the reverse order of the blocks: the constant block is in the Simulink environment, while the sine wave block is made in System Generator with the help of the DDS block. The DDS Compiler Block is a direct digital synthesizer and it uses a lookup table scheme to generate sinusoids.

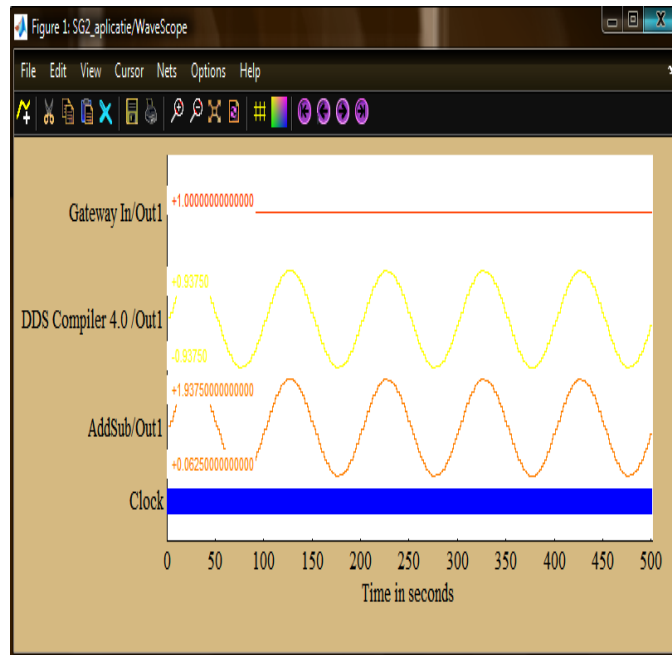


Fig. 2.31 The corresponding signals:  
 a – constant waveform    b – sine waveform    c – output signal

The sine signals do not have anymore the square edges because the signals are generated internal, in the FPGA.

The fourth implementation of the adder is illustrated in fig. 2.32. All blocks are generated internal in System Generator, which means that in the FPGA.

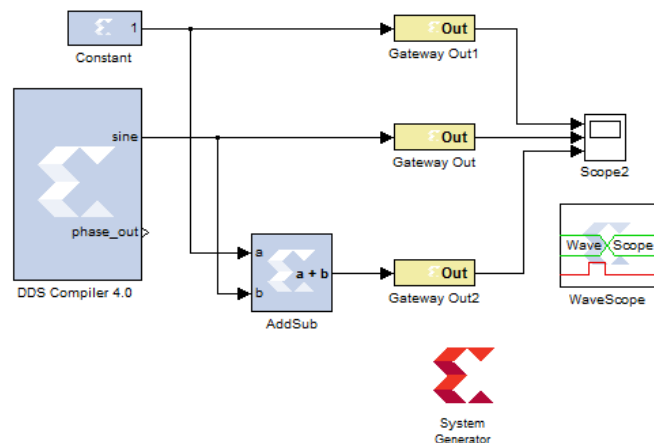


Fig. 2.32 The fourth implementation of the System Generator adder

The corresponding signals are also illustrated, in fig. 2.33.

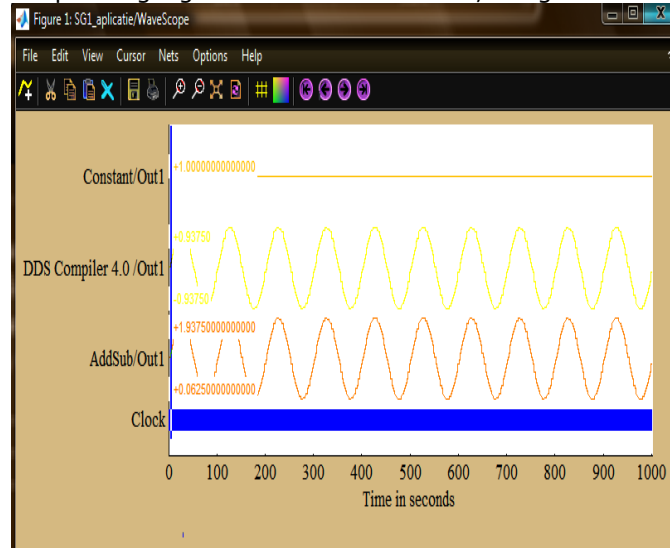


Fig. 2.33 The corresponding signals:

a – constant waveform    b – sine waveform    c – output signal

## 2.5. Conclusions and Contributions

Chapter 2 presents the resources, the methods and the ways used in the research.

In the first part of chapter 2, an overview of reconfigurable computing was presented. Reconfigurable computing became popular due to the availability of field programmable gate arrays. The main architectures of the FPGAs were investigated; also the software tools used in implementing and simulating a design and programming the FPGAs were presented. These tradeoffs led to the decision/conclusion to use the Xilinx FPGA boards (Nexys2 and Spartan 3E) and the software tool (ISE Webpack) provided by the same corporation.

The simulation capabilities of the FPGA simulator were investigated and also compared to real life behavior. The exploitation and dissemination of the experiments were further investigated in two scientific papers: **[117]** and **[118]**.

In the second part of chapter 2, the reasons why DSP is suitable for reconfigurable computing were brought up. The most useful DSP tool used for FPGA design is System Generator, a software tool from Xilinx based on the Matlab/Simulink environment. System Generator produces command files for FPGA synthesis, HDL simulation and implementation tool, all with the purpose of obtaining a hardware model. Also, the advantages and disadvantages of System Generator were outlined.

For understanding the implementation steps, a simple design, an adder made up of a constant and a sine generator, was simulated. First, only in the Simulink environment, and then, the design was implemented in System Generator, in four different ways:

- the constant and the sine generator were both Simulink blocks and pass through the GatewayIn blocks into the FPGA boundary which represents the System Generator environment;
- The constant in System Generator and the sine generator in Simulink;
- The constant in Simulink and the sine generator (DDS compiler) in System Generator;
- Both, the constant and the sine generator implemented in System Generator.

The signals of the simulated blocks were more accurate if Xilinx blocks had been used. Otherwise, the waveforms had square edges.

### **Contributions:**

- An overview of reconfigurable computing was presented, outlining the strengths and the advantages of FPGAs in contrast to ASIC and GPP.
- **A common but accurate methodology on an FPGA, simulations checked by measurements, was followed.**
- **The capabilities of the FPGA simulator, the Xilinx ISE, were analysed and compared with practical results:**
  - **The delays of the post-route simulation were investigated and also compared with measured delays;**
  - **Corrections had been made to results according to the type of their use.**
- **The advantages and disadvantages of the System Generator were outlined.**
- A simple design was simulated in Simulink and System Generator in order to introduce and understand the basic concepts of the MATLAB simulator:
  - To test the design functionality, the adder was simulated in Simulink;
  - To test the functionality as an FPGA design, the Simulink blocks were translated in dedicated Xilinx blocks from System Generator, obtaining four different designs of the adder.
- Different implementations of the same model were compared: the signals of the simulated blocks were more accurate if Xilinx blocks were used.
- Experimental results proved the precision of the two simulators: the Xilinx ISE and MATLAB/Simulink and System Generator.

## 3. MODULATION IN DIGITAL AGE PROSPECTION

### 3.1. Digital Communication System

As it was mentioned in the previous chapter, two of the most common operations used in digital signal processing were modulation and demodulation. These operations make the topic of the chapter.

Communication represents information transfer between different points in space or time and requires a sender, a message and a receiver. In digital communication, the information is represented in digital form, as binary digits or bits. Today, most communication systems used for transferring information are either digital or are being converted from analog to digital.

#### 3.1.1. General Assumptions for the Digital Communication System

Fig. 3.1 illustrates the block diagram of a typical digital communication system.

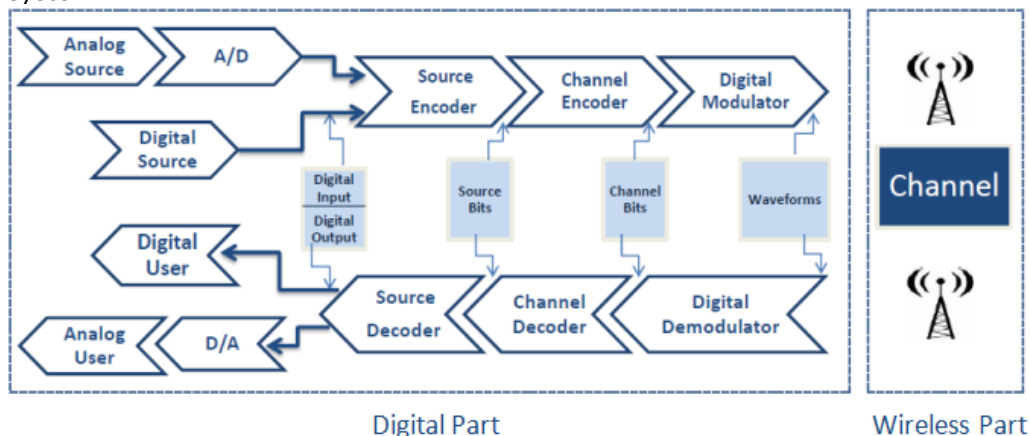


Fig. 3.1 Block diagram of a typical digital communication system

#### Functionalities of the Transmitter

Typically, the message to be sent is from an analog source (e.g. voice) or digital (e.g. computer data). Since analog messages are characterized by data whose values vary over a continuous range, the output of an analog communication system can assume an infinite number of possible waveforms [110]. In digital communication, the information is represented in a digital form, as binary digits or simply bits. Today, most of the communication systems are either digital, or are converted from analog to digital [94], [158].

The analog to digital converter A/D is a device which converts the analog signals to a digital form, a sequence of binary digits. Ideal, is to represent the message into a shorter digital signal.

The process of efficiently converting the output of either an analog source or digital source into a sequence of binary digits is called source encoding or data compression [131]. The task of the source encoder is to reduce the redundancy in the original information in a manner that takes into account the end user's requirements [94]. While the source encoder eliminates unwanted redundancy in the information to be sent, the channel encoder introduces redundancy in order to combat errors that may arise from channel imperfections and noise, so that some of the errors caused by noise or interference through the channel can be corrected at the receiver [94], [131], [180]. The binary information obtained at the output of the channel encoder is then passed to a digital modulator which serves as interface with the communication channel. The main purpose of the modulator is to translate the discrete symbols into an analog waveform that can be transmitted over the channel [41], [74], [94], [131], [170].

Fig. 3.2 represents the modulator scheme with the signals related, where:

$m(t)$  is the message/ information signal;

- $c(t)$  is the carrier signal;
- $s(t)$  is the modulated/ transmitted signal.

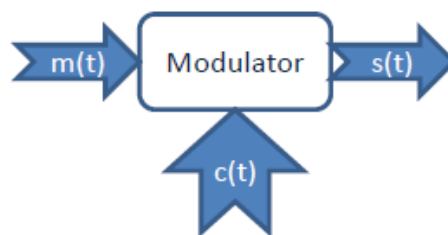


Fig. 3.2 The modulator scheme

The carrier is a waveform, a sinusoidal signal, which is modulated with an input signal for the purpose of conveying information, given by [10]:

$$c(t) = A_c \cos(2\pi f_c t + \theta_c(t)) \quad (3.1)$$

where:  $A_c$  represents the amplitude of the carrier;  
 $f_c$  represents the frequency of the carrier;  
 $\theta_c$  represents the phase of the carrier.

After examining the carrier  $c(t)$ , the three parameters which can vary are: the amplitude, the frequency and the phase. These parameters can be varied either in analog form or in digital form. When varying in digital form, it is referred to as "Shifting and Keying".

Fig. 3.2 is part of a more detailed scheme, fig. 3.3, which is the block diagram of a binary digital communication system. The binary digit  $b_k$  (0 or 1) is represented by one of two electrical waveforms  $s_1(t)$  or  $s_2(t)$ . These waveforms are transmitted through the channel and perturbed by noise. At the receiving side, the receiver needs to make a decision,  $\hat{b}_k$  on the transmitted bit based on the received signal  $r(t)$ . The performance of the receiver is measured in terms of error probability.

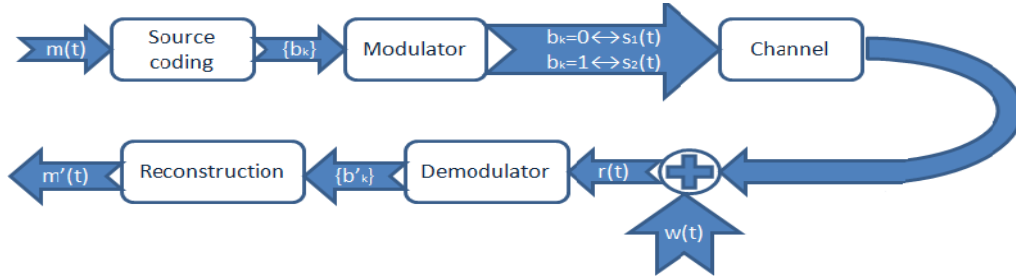


Fig. 3.3 The block diagram of a binary digital communication system

A channel model of infinite bandwidth with AWGN is assumed. Other assumptions and the notation which pertain to the discussion of the communication system in fig. 3.3 are as follows [110]:

- The bit duration of  $b_k$  is  $T_b$  second, or the bit rate is  $r_b = \frac{1}{T_b}$   $\left[ \frac{\text{bits}}{\text{second}} \right]$ .
- Bits in two different time slots are statistically independent.
- The probabilities of  $b_k$  are as follows:

$$P[b_k = 0] = P_1 \tag{3.2}$$

$$P[b_k = 1] = P_2 \tag{3.3}$$

where  $P_1 + P_2 = 1$ . Most often, we will assume that the bits are equally likely,  $P_1 = P_2 = \frac{1}{2}$ .

- The bit  $b_k$  is mapped by the modulator into one of the two signals  $s_1(t)$  and  $s_2(t)$ . Each signal is of duration  $T_b$  seconds and has a finite energy:

$$E_1 = \int_0^{T_b} s_1^2(t) dt \quad [\text{Joules}], E_1 < \infty \tag{3.4}$$

$$E_2 = \int_0^{T_b} s_2^2(t) dt \quad [\text{Joules}], E_2 < \infty \tag{3.5}$$

- The channel is sufficiently wideband that the signals  $s_1(t)$  and  $s_2(t)$  pass through without any distortion. Essentially, this means that there is no inter symbol interference (ISI) between successive bits.
- The noise  $w(t)$  is stationary Gaussian, zero-mean, white noise, which means that:

$$E\{w(t)\} = 0, \quad E\{w(t)w(t+\tau)\} = \frac{N_0}{2} \delta(\tau) \tag{3.6}$$

The two electrical signals  $s_1(t)$  and  $s_2(t)$  must be represented as linear combinations of two orthonormal basis functions  $\phi_1(t)$  and  $\phi_2(t)$  used to represent the signals exactly. The two functions  $\phi_1(t)$  and  $\phi_2(t)$  are said to be orthonormal if the following two conditions are satisfied [110]:

1. *Orthogonality:*

$$\int_0^{T_b} \phi_1(t)\phi_2(t)dt = 0 \quad (3.7)$$

2. *Normalize:*

$$\int_0^{T_b} \phi_1^2(t)dt = \int_0^{T_b} \phi_2^2(t)dt = 1 \quad (3.8)$$

After computing these data, [110] gives the formula for these orthonormal functions:

$$\phi_1(t) = \frac{s_1(t)}{\sqrt{E_1}} \quad (3.9)$$

$$\phi_2(t) = \frac{1}{\sqrt{1-\rho^2}} \left[ \frac{s_2(t)}{\sqrt{E_2}} - \frac{\rho s_1(t)}{\sqrt{E_1}} \right] \quad (3.10)$$

where  $\rho$  is named correlation coefficient and is defined by the formula below:

$$\rho = \int_0^{T_b} \frac{s_2(t)}{\sqrt{E_2}} \phi_1(t)dt = \frac{1}{\sqrt{E_1 E_2}} \int_0^{T_b} s_1(t)s_2(t)dt \quad (3.11)$$

The orthonormal set  $\{\phi_1(t), \phi_2(t)\}$  is selected to represent the two signals exactly. This representation is expressed geometrically in a signal space plot.

In any bit interval we receive the noise-corrupted signal. The received signal is [110]:

$$\begin{aligned} r(t) &= s_i(t) + w(t), \quad 0 \leq t \leq T_b \\ &= \begin{cases} s_1(t) + w(t), & \text{if a 0 is transmitted} \\ s_2(t) + w(t), & \text{if a 1 is transmitted} \end{cases} \end{aligned} \quad (3.12)$$

The probability of making an error is given by [110]:

$$\begin{aligned} P[\text{error}] &= P[(0 \text{ transmitted and } 1 \text{ decided}) \text{ or } (1 \text{ transmitted and } 0 \text{ decided})] \\ &= P[(0_T, 1_D), \text{ or } (1_T, 0_D)] \end{aligned} \quad (3.13)$$

### Channel

To transmit the signal (message) from the source to the receiver, a communication channel is used. Modulation techniques are chosen or designed according to channel characteristics in order to optimize their performance. The characteristics of a communication channel can vary widely and good channels are critical to the design of efficient communication systems [131]. Whatever the medium used for transmission is, the signal is corrupted with thermal noise. The channel also has a limited frequency bandwidth so that it can be viewed as a filter [74], [180].



In studying, choosing and designing modulation schemes an important factor is the channel characteristic. An Additive White Gaussian Noise (AWGN) channel model is a basic assumption made about the corrupting noise in most communication systems performance analysis and is assumed in the present work. This assumption corresponds to the fact that the channel does nothing but add a white gaussian noise to the signal passing through the channel. It was further assumed that the modulated signals passing through the channel does not suffer for amplitude loss and phase distortion.

The AWGN channel does not exist since no channel can have an infinite bandwidth. Nevertheless, when the signal bandwidth is smaller than the channel bandwidth, many practical channels are approximately with an AWGN channel.

### Functionalities in the Receiver

In the receiver, the reverse signal processing happens. First, the weak signal is amplified and demodulated [180]. One of its key tasks is synchronization: the demodulator must account for the fact that the channel can produce phase, frequency, and time shifts, and that the clocks and oscillators at the transmitter and receiver are not synchronized a priori. Another task may be channel equalization, or compensation of the inter-symbol interference induced by a dispersive channel. The ultimate goal of the demodulator is to produce tentative decisions on the transmitted symbols to be fed to the channel decoder [41], [94]. Then, the added redundancy is taken away by the channel decoder. Next, the source decoder recovers the signal to its original form and sends it to the user. For an analog source, a digital to analog converter D/A is used [180].

In order to transmit a signal over long distances, carrier modulation is used. This modulation uses a sequence of digital symbols to alter the parameters of a high-frequency sinusoidal signal called carrier.

## 3.1.2. Criteria of Choosing Modulation Schemes

The essence of digital modem design is to transmit digital bits from the transmitter to the receiver through a channel and recover these bits from corruptions from noise and other imperfections of the channel. There are three primary criteria of choosing modulation schemes: power efficiency, bandwidth efficiency and system complexity, all documented from [180].

### 3.1.2.1. Power Efficiency

The Power efficiency of a modulation scheme is defined straightforwardly as the required  $\frac{E_b}{N_0}$  for a certain bit error probability ( $P_b$ ) over an AWGN (Additive White Gaussian Noise) channel, where  $E_b$  is the average bit energy,  $N_0$  is the noise power spectral density.  $P_b = 10^{-5}$  is usually used as the reference bit error probability. In other words, power efficiency is a measure of how much received power is needed to achieve a specified bit error rate (BER). As the bit error rate increases, the power efficiency decreases since transmitted power is wasted on more bad data.

### 3.1.2.2. Bandwidth Efficiency

The bandwidth efficiency of a communication system is typically a measure of how well the system is using the bandwidth resource [41].

The bandwidth efficiency is defined as the number of bits per second that can be transmitted in one hertz of system bandwidth.

$$\eta_B = \frac{R}{B} \left[ \frac{\text{bits/s}}{\text{Hz}} \right] \quad (3.14)$$

where  $R$  is the ratio of the bit rate and  $B$  is the signal bandwidth.

Bandwidth efficiency depends on the requirement of system bandwidth for a certain modulated signal.

Three bandwidth efficiencies are listed in literature [180] as follows:

(a) *Nyquist Bandwidth Efficiency*: ideal case.

Assuming the system uses Nyquist filtering at baseband which has the minimum bandwidth required for inter symbol interference – free transmission of the digital signals, the bandwidth efficiency for a M-ary modulation is:

$$\frac{R_b}{B} = \log_2 M \quad (3.15)$$

where  $R_b$  represents the bit rate and  $B$  the bandwidth of the signal.

(b) *Null-to-Null Bandwidth Efficiency*: to define the bandwidth as the main spectral lobe.

(c) *Percentage Bandwidth Efficiency*: In the case that the modulated signal does not have nulls, energy percentage bandwidth may be used. Usually, 99% is used.

### 3.1.2.3. System Complexity

System complexity refers to the amount of circuits involved and the technical difficulty of the system. Associated with the system complexity is the cost of manufacturing, which is of course a major concern in choosing a modulation technique. Usually the demodulator is more complex than the modulator. Coherent demodulator is much more complex than non coherent demodulator since carrier recovery is required.

## 3.2. Basics of Digital Modulation Techniques

### 3.2.1. Introduction in Digital Modulation

All modulation techniques are known as M-ary modulations. In a M-ary modulation, two or more bits are grouped together to form symbols and signals, and one of the possible signals obtained is transmitted. Normally, the number of possible signals is  $M = 2^n$ , where  $n$  is an integer [10].

The basic modulation techniques have  $M = 2$ , so in another words, they are called binary modulations. M-ary modulation techniques are attractive for use in band limited channels, because these techniques achieve better bandwidth efficiency at the expense of power efficiency. M-ary signaling results in poorer error performance because of smaller distances between signals in the constellation diagram. In another words, M-ary schemes increase the bandwidth efficiency but

require higher transmission power to keep the same bit error rate. Several commonly used M-ary signaling schemes are discussed in the present work, but before of doing this, an overview of the digital modulation schemes is presented.

### 3.2.2. Overview of Digital Modulation Techniques

In order to describe the digital modulation techniques, [180] listed the abbreviations and names of these techniques in table 3.1 and then, arranged them in a tree diagram listed in fig. 3.4.

The modulation schemes listed in the table 3.1 and in fig. 3.4 are classified into two large categories: constant envelope and non constant envelope.

Table 3.1 Abbreviations and names of the digital modulation techniques [180]

Abbreviation	Alternative Abbreviation	Descriptive name
<b>CONSTANT ENVELOPE MODULATIONS</b>		
<b>Frequency Shift Keying (FSK)</b>		
<b>BFSK</b>	<i>FSK</i>	<b>Binary Frequency Shift Keying</b>
<b>MFSK</b>		<b>M-ary Frequency Shift Keying</b>
<b>Phase Shift Keying (PSK)</b>		
<b>BPSK</b>	<i>PSK</i>	<b>Binary Phase Shift Keying</b>
<b>QPSK</b>	<i>4PSK</i>	<b>Quadrature Phase Shift Keying</b>
<b>OQPSK</b>	<i>SQPSK</i>	<b>Offset QPSK, Staggered QPSK</b>
<b><math>\pi/4</math>-QPSK</b>		<b><math>\pi/4</math> Quadrature Phase Shift Keying</b>
<b>MPSK</b>		<b>M-ary Phase Shift Keying</b>
<b>Continuous Phase Modulations (CPM)</b>		
<b>SHPM</b>		<b>Single-h (modulation index) Phase Modulation</b>
<b>MHPM</b>		<b>Multi-h Phase Modulation</b>
<b>LREC</b>		<b>Rectangular Pulse of Length L</b>
<b>CPFSK</b>		<b>Continuous Phase Frequency Shift Keying</b>
<b>MSK</b>	<i>FFSK</i>	<b>Minimum Shift Keying, Fast FSK</b>
<b>SMSK</b>		<b>Serial Minimum Shift Keying</b>
<b>LRC</b>		<b>Raised Cosine Pulse of Length L</b>
<b>LSRC</b>		<b>Spectrally Raised Cosine Pulse of Length L</b>
<b>GMSK</b>		<b>Gaussian Minimum Shift Keying</b>
<b>TFM</b>		<b>Tamed Frequency Modulation</b>
<b>NON CONSTANT ENVELOPE MODULATIONS</b>		
<b>Amplitude and Amplitude/Phase Modulations</b>		
<b>ASK</b>		<b>Amplitude Shift Keying (generic name)</b>
<b>OOK</b>	<i>ASK</i>	<b>Binary ON-Off Keying</b>
<b>MASK</b>	<i>MAM</i>	<b>M-ary ASK, M-ary Amplitude Modulation</b>
<b>QAM</b>		<b>Quadrature Amplitude Modulation</b>
<b>Other Non constant Envelope Modulations</b>		
<b>QORC</b>		<b>Quadrature Overlapped Raised Cosine Modulation</b>
<b>SQORC</b>		<b>Staggered QORC</b>
<b>QOSRC</b>		<b>Quadrature Overlapped Squared Raised Cosine</b>

		Modulation
<b>Q<sup>2</sup>PSK</b>		<b>Q</b> uadrature <b>Q</b> uadrature <b>P</b> hase <b>S</b> hift <b>K</b> eying
<b>IJF-OQPSK</b>		<b>I</b> nter symbol- <b>I</b> nterference/ <b>J</b> itter- <b>F</b> ree <b>OQPSK</b>
<b>TSI-OQPSK</b>		<b>T</b> wo- <b>S</b> ymbol- <b>I</b> nterval <b>OQPSK</b>
<b>SQAM</b>		<b>S</b> uperposed- <b>QAM</b>
<b>XPSK</b>		<b>C</b> ross correlated <b>QPSK</b>

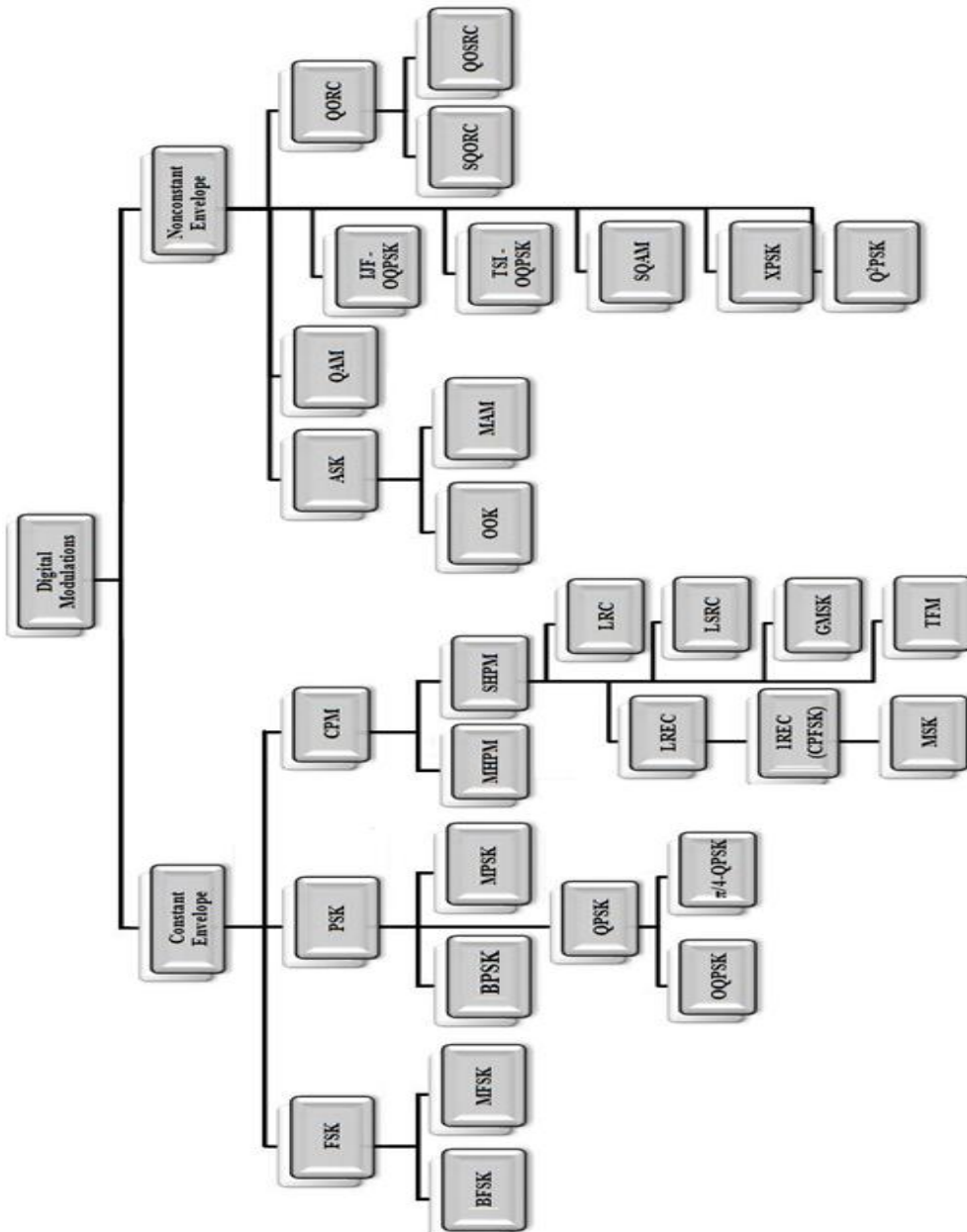


Fig. 3.4 Digital Modulation Tree

### 3.2.3. M-ary Phase Shift Keying (MPSK)

Phase Shift Keying is an efficient, in terms of signal power, digital modulation method. It is widely used in modern communication systems, such as satellite links, wideband microwave radio relay systems [110]. The digital information is coded in the phase function of a constant – amplitude carrier signal in which the phase of the transmitted signal is varied to convey information.

The motivation behind MPSK is to increase the bandwidth efficiency of the PSK modulation schemes. In BPSK, a symbol represents a data bit. In MPSK, a symbol is represented by  $n = \log_2 M$  data bits so the bandwidth efficiency is increased to  $n$  times [159]. Among all MPSK schemes, QPSK is the most used since it does not suffer from BER degradation while the bandwidth efficiency is increased [180]. The advantage of QPSK over BPSK is evident: QPSK transmits twice the data rate in a given bandwidth compared to BPSK, at the same BER.

#### 3.2.3.1. Binary Phase Shift Keying (BPSK)

BPSK is the simplest form of phase shift keying and it uses two phases which are separated by  $180^\circ$ . This modulation is the most robust of all the PSKs since it takes the highest level of noise or distortion to make the demodulator reach an incorrect decision. It is, however, only able to modulate at 1 bit/symbol and this is the reason why is unsuitable for high data-rate applications.

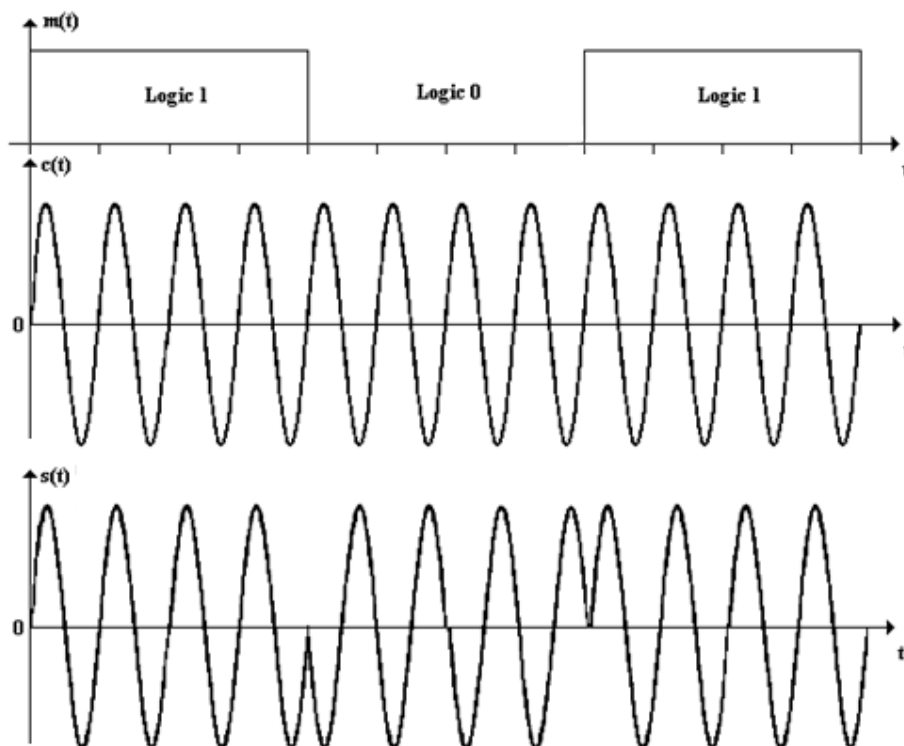


Fig. 3.5 PSK modulation [180]

A BPSK signal is generated by modulating the amplitude of the sinusoidal carrier with a digital signal [22]. The transmitted/modulated BPSK signal is  $s(t) = m(t)c(t)$ , so the signal set is given by equation (3.16) with a resultant phase of either 0 or  $\pi$  radians:

$$s(t) = \begin{cases} s_1(t) = A \sin(2\pi f_c t + \pi) = -A \sin(2\pi f_c t), & \text{if } 0_T \\ s_2(t) = A \sin(2\pi f_c t + 0) = +A \sin(2\pi f_c t), & \text{if } 1_T \end{cases}, \quad 0 < t \leq T_b \quad (3.16)$$

In other words, in BPSK, a symbol "1" is transmitted as a burst of carrier with a  $0^\circ$  initial phase, while a symbol "0" is transmitted as a burst of carrier with  $180^\circ$  initial phase shown in fig. 3.7.

The signals constellation is illustrated in fig. 3.6.

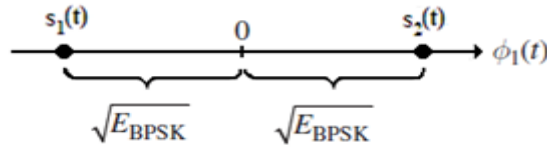


Fig. 3.6 Signal space plot of BPSK [34], [110]

The BPSK modulator is quite simple and is illustrated in fig. 3.7. The binary sequence  $m(t)$  or modulating signal is multiplied with a sinusoidal carrier and the BPSK modulated signal  $s(t)$  is obtained.

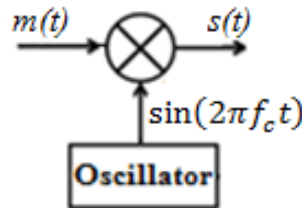


Fig.3.7 BPSK Modulator [180]

To demodulate the signal, it is necessary to reconstitute the carrier. This process is made in the Carrier Recovery Circuit. Next, the BPSK modulated signal is multiplied with the carrier, pass through an integrator and a decision circuit to obtain in the end the modulating signal.

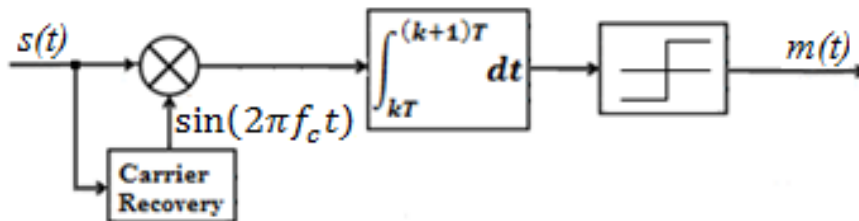


Fig.3.8 BPSK Demodulator [180]

Each signal is of duration  $T_b$  and has the same energy [110]:

$$E_{BPSK} = A^2 \frac{T_b}{2} \quad [J] \quad (3.17)$$

Only one of the two orthonormal basis function is needed in representing the two signals [110]:

$$\phi_1(t) = \frac{s_2(t)}{\sqrt{E_{BPSK}}} = \sqrt{\frac{2}{T_b}} \sin(2\pi f_c t) \quad (3.18)$$

For  $P_1 = P_2$ , the error probability is [110]:

$$P[\text{bit error}]_{BPSK} = Q\left(\sqrt{\frac{2E_{BPSK}}{N_0}}\right) = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \quad (3.19)$$

A simple application of the BPSK modulation has been made in the MATLAB environment. The modulating and modulated signals are shown in the figure below:

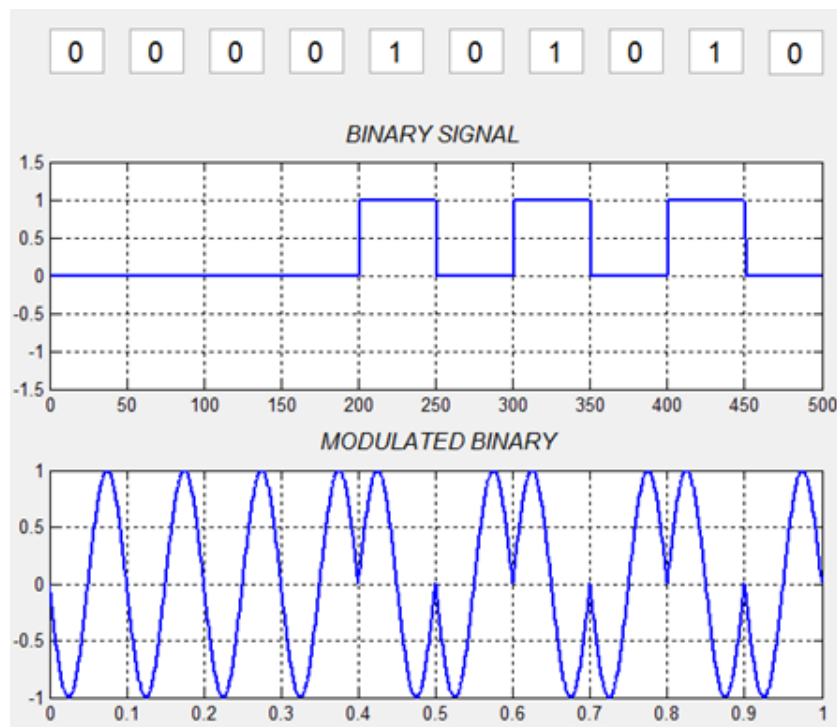


Fig.3.9 Application of the BPSK Modulation

### 3.2.3.2. Quadrature Phase Shift Keying (QPSK)

The basic idea behind QPSK exploits the fact that  $\cos(2\pi f_c t)$  and  $\sin(2\pi f_c t)$  are orthogonal over the  $[0, T_b]$  interval, where  $f_c = \frac{k}{T_b}$  and  $k$  is an integer. This leads to the transmission of two different messages over the same frequency band. In order

to accomplish this, the bit stream is taken two bits at a time and mapped into signals as shown in table 3.2.

Table 3.2 QPSK signals and a mapping to the messages

Bit pattern	Message	Signal transmitted	
00	$m_1$	$s_1(t) = A \cos(2\pi f_c t),$	$0 \leq t \leq T_s = 2T_b$
01	$m_2$	$s_2(t) = A \sin(2\pi f_c t),$	$0 \leq t \leq T_s = 2T_b$
11	$m_3$	$s_3(t) = -A \cos(2\pi f_c t),$	$0 \leq t \leq T_s = 2T_b$
10	$m_4$	$s_4(t) = -A \sin(2\pi f_c t),$	$0 \leq t \leq T_s = 2T_b$

Since each bit occupies  $T_b$  seconds, the signals corresponding to the 'dibits' or symbols: 00, 01, 11, 10, last for a symbol duration of  $T_s = 2T_b$  seconds. The symbol signaling rate is therefore halved:

$$r_s = \frac{1}{T_s} = \frac{1}{2T_b} = \frac{r_b}{2} \text{ [symbols/second]} \quad (3.20)$$

Since the bandwidth requirement is proportional to  $r_s$ , it can also be reduced by half for a given bit rate,  $r_b$ . In opposites, for a fixed bandwidth the bit rate  $r_b$  can be doubled [110].

Since QPSK is a special case of MPSK [150], the signals of the QPSK modulation are defined in equation (3.21):

$$s_i(t) = A \cos[2\pi f_c t + \theta_i], \quad i = 1, 2, 3, 4 \quad (3.21)$$

and  $\theta_i = \frac{(2i-1)\pi}{4}$

The initial phases of the signals are  $\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, \frac{7\pi}{4}$ .

Equation (3.21) can be rewritten as:

$$s_i(t) = A \cos \theta_i \cos(2\pi f_c t) - A \sin \theta_i \sin(2\pi f_c t) \quad (3.22)$$

$$= s_{i1}(t)\phi_1(t) + s_{i2}(t)\phi_2(t)$$

where:

$$\phi_1(t) = \sqrt{\frac{2}{T_s}} \cos(2\pi f_c t) \quad (3.23)$$

$$\phi_2(t) = \sqrt{\frac{2}{T_s}} \sin(2\pi f_c t) \quad (3.24)$$

$$s_{i1} = \sqrt{E} \cos \theta_i \quad (3.25)$$

$$s_{i2} = \sqrt{E} \sin \theta_i \quad (3.26)$$



and  $E_s = \frac{A^2 T_s}{2}$  is the symbol energy and  $\theta_i = \tan^{-1} \frac{s_{i2}}{s_{i1}}$ .

In a M-ary modulation, two or more bits are grouped together to form symbols and signals, and one of the possible signals obtained is transmitted. Normally, the number of possible signals is  $M = 2^n$ , where n is an integer. For QPSK, M is 4, which means that n is 2 bits per symbol. There are four possible cases of dibits: 00, 01, 10 and 11. Each of the four QPSK signals is used to represent one of them. The coordinates of signal points are illustrated in table 3.3. In the table, the logic '1' is mapped into  $+\sqrt{\frac{E}{2}}$  and the logic '0' into  $-\sqrt{\frac{E}{2}}$  [180].

Table 3.3 QPSK signal coordinates

Dibit	Phase $\theta_i$	$s_{i1} = \sqrt{E} \cos \theta_i$	$s_{i2} = \sqrt{E} \sin \theta_i$
11	$\frac{\pi}{4}$	$+\sqrt{\frac{E}{2}}$	$+\sqrt{\frac{E}{2}}$
01	$\frac{3}{4}$	$-\sqrt{\frac{E}{2}}$	$+\sqrt{\frac{E}{2}}$
00	$-\frac{3}{4}$	$-\sqrt{\frac{E}{2}}$	$-\sqrt{\frac{E}{2}}$
10	$-\frac{\pi}{4}$	$+\sqrt{\frac{E}{2}}$	$-\sqrt{\frac{E}{2}}$

Also, the odd bits were mapped to  $a_i(t)$  and the even bits into  $a_q(t)$  and the original information is  $a(t)$ , where the I and Q are mnemonics for inphase and quadrature [149]. The individual bits in each stream occupy  $T_s = 2T_b$  seconds and modulate the inphase carrier,  $A \cos(2\pi f_c t)$ , and respectively the quadrature carrier  $A \sin(2\pi f_c t)$ .

The transmitted signal from equation (3.22) can be written as:

$$s(t) = a_i(t)A \cos(2\pi f_c t) + a_q(t)A \sin(2\pi f_c t) \tag{3.27}$$

$$= \sqrt{a_i^2(t) + a_q^2(t)} A \cos\left(2\pi f_c t - \tan^{-1}\left(\frac{a_q(t)}{a_i(t)}\right)\right)$$

$$= \sqrt{2} A \cos[2\pi f_c t - \theta(t)] \tag{3.28}$$

where the phase  $\theta(t)$  is determined as follows:

$$\theta(t) = \begin{cases} \frac{\pi}{4}, & \text{if } a_I = +1, a_Q = +1 \text{ (bits are 11)} \\ -\frac{\pi}{4}, & \text{if } a_I = +1, a_Q = -1 \text{ (bits are 10)} \\ \frac{3\pi}{4}, & \text{if } a_I = -1, a_Q = +1 \text{ (bits are 01)} \\ -\frac{3\pi}{4}, & \text{if } a_I = -1, a_Q = -1 \text{ (bits are 00)} \end{cases} \quad (3.29)$$

Fig. 3.10 shows how the QPSK transmitted waveform is generated from its inphase and quadrature components from the bit sequence.

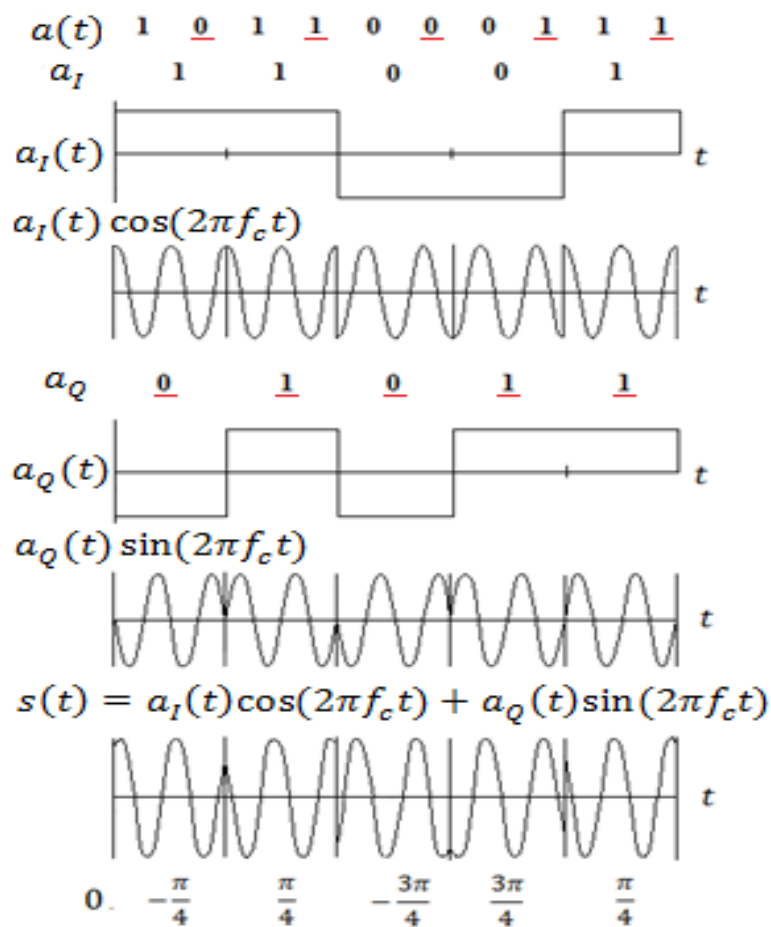


Fig.3.10 QPSK waveforms [180]

The four signals can be represented in terms of the two orthonormal basis functions,  $\phi_1(t)$  and  $\phi_2(t)$  as shown in fig. 3.11.

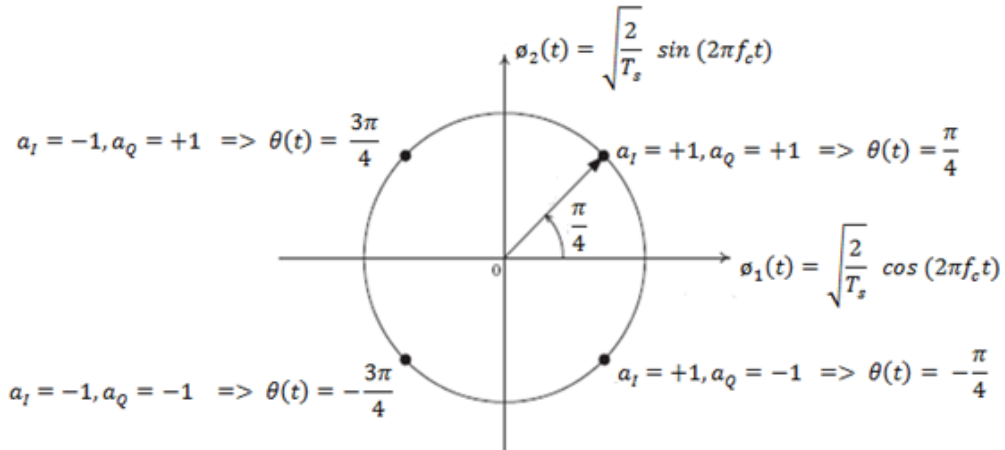


Fig.3.11 Representing QPSK signals as phases of the sinusoidal carrier [110]

The QPSK modulator illustrated in fig. 3.12 is based on equation (3.27). The binary sequence  $a(t)$  is separated by the serial-to-parallel converter into odd-bit-sequence for the  $I$  channel and even-bit-sequence for the  $Q$  channel. The channel modulated with cosine is called inphase ( $I$ ) and the channel modulated with sine is called quadrature ( $Q$ ). The odd-bit-sequence is multiplied with  $\cos 2\pi f_c t$  and the even-bit-sequence with  $\sin 2\pi f_c t$ . It is obvious that the  $I$ -channel and  $Q$ -channel signals are BPSK signals with a symbol duration of  $2T_b$ . Finally, an adder brings the two waveforms together producing the modulated QPSK signal, all seen in fig. 3.10.

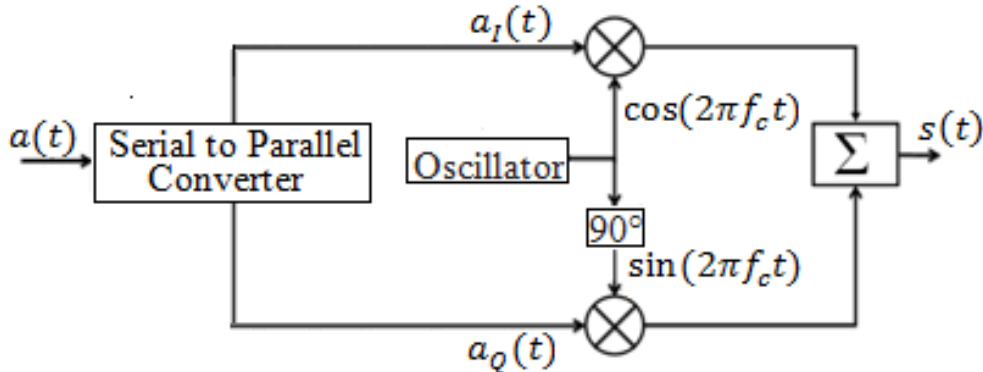


Fig.3.12 QPSK Modulator

In the demodulator, the reverse operation happens. The  $I$ -channel and the  $Q$ -channel are demodulated separately as two different BPSK signals. The  $I$ -channel is demodulated with a cosine waveform and the  $Q$ -channel, with a sine waveform. A parallel-to-serial-converter is used to combine the two sequences into a single one which represents the modulating signal.

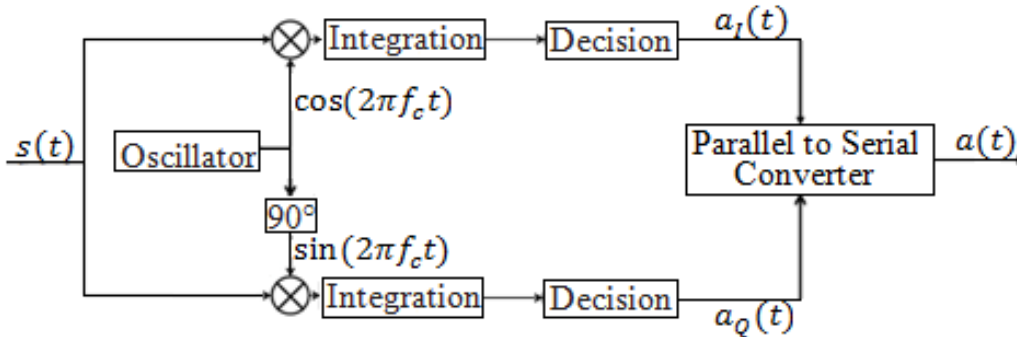


Fig.3.13 QPSK Demodulator

The phase of each signal point is relative to the  $\phi_1(t)$  axis and is proportional to  $A \cos(2\pi f_c t)$ . The energy of each QPSK signal is:

$$E_s = A^2 T_s \quad (3.30)$$

To obtain the minimum-error-probability, the even and odd bit sequences must be treated separately since  $a_I(t)$  does not have a component along  $\phi_2(t)$  and  $a_Q(t)$  does not have a component along  $\phi_1(t)$ . Thus the QPSK signal can be considered to consist of two separate BPSK signals. For equally likely signals, the bit error rate probability is the same with that of the BPSK modulation and is given by [110]:

$$P[\text{bit error}] = Q \left( \sqrt{\frac{A^2 T_s}{N_0}} \right) \quad (3.31)$$

The above expression is identical to that of (3.19) when expressed in terms of  $E_b$ , the energy bit.

The energy of each QPSK signal is given by (3.30), and hence:

$$E_b = \frac{E_s}{2} = \frac{A^2 T_s}{2} \Rightarrow A^2 T_s = 2E_b \quad (3.32)$$

Substituting the expression (3.32) into (3.31), the expression of the bit error probability of the QPSK modulation becomes:

$$P[\text{bit error}] = Q \left( \sqrt{\frac{2E_b}{N_0}} \right) \quad (3.33)$$

A simple application of the QPSK modulation has been made in the MATLAB environment. The binary sequence and the QPSK modulated signals are shown in the figure below:

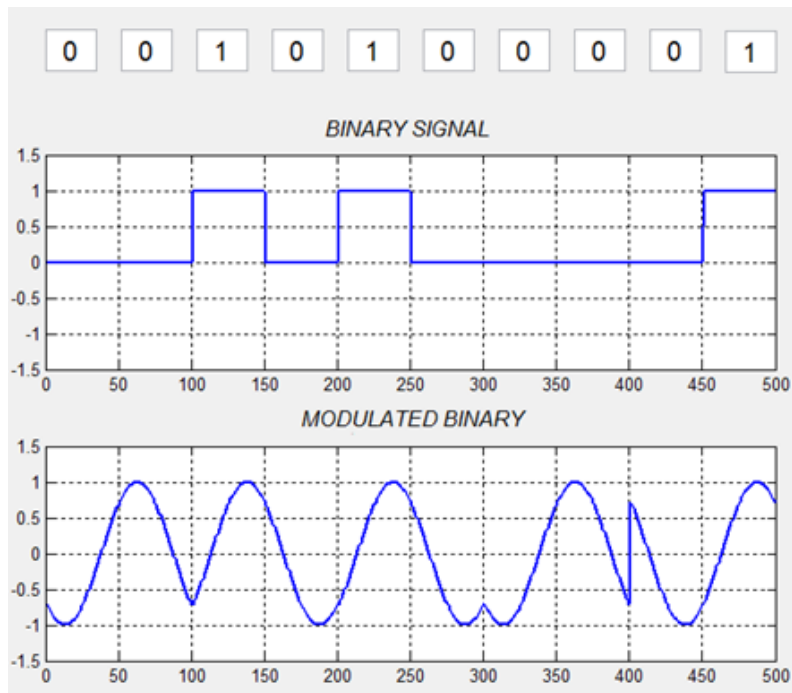


Fig. 3.14 Application of the QPSK Modulation

The error performances of the BPSK and QPSK modulation schemes are illustrated in fig.3.15:

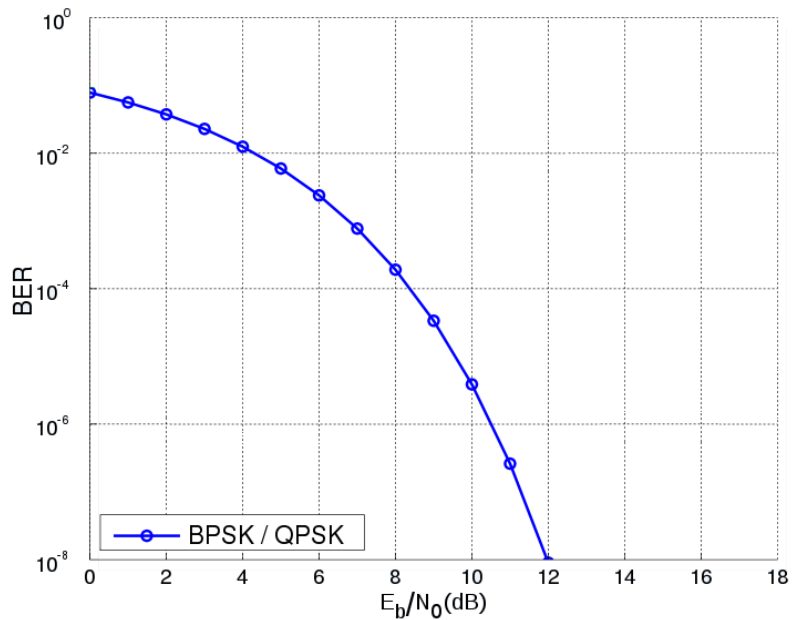


Fig. 3.15 Error performance of the BPSK and QPSK modulation schemes

### 3.3. A Literature Survey

We live in the era of communication. Everything is audio or video and any information is transformed into analog or digital signal. The design of a communication system depends on the type of the used signal. The choice of digital communication technique over its analog counterpart was discussed in detail in chapter 2. The major advantage of using digital modulation techniques is that, the use of digital signals reduces the hardware utilization of any system, reduces the noise in any equipment and also reduces the interferences problems as compared to the analog signals.

The key feature of a digital communications system is that it sends only a finite set of messages, in contrast to an analog communications system, which can send an infinite set of messages. In a digital communications system, the objective at the receiver is not to reproduce a waveform with precision, but to determine a finite set of waveforms that had been sent by the transmitter [151].

In the object of digital communications, a remarkable development has happened in the last 20 years. Due to this grown, newer technologies and applications have emerged daily. The existing modulation techniques need to be modified according to the new technologies.

The two major digital modulation techniques used in this research are the Binary Phase Shift Keying (BPSK) and the Quadrature Phase Shift Keying (QPSK).

Early works in the field of digital modulation have been published in the literature. Some of the most papers are remembered, starting with [169], where frequency domain parameters were defined and used in order to distinguish between six modulation types. One of the early papers treating digital modulation types was [91], where results based on a statistical analysis of various signals parameters to discriminate between Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and PSK were presented. Other early works of techniques using signal parameters were reported in [6], [24], [56], [66], [82] and recently in [20], [57], [58], [59].

A combination of techniques including pattern recognition is used in [35], [77] and more recently in [111].

In [116], a new method for automatic modulation recognition of MPSK was used, with  $M=2, 4$  and  $8$ . In [147], coherent and noncoherent performance in terms of error rate and false-alarm rate for PSK/QAM modulations were evaluated. Comparing [116] with [147], the performance obtained with this new method is comparable with coherent maximum-likelihood approach and is better than the noncoherent one [120]. In [13], more techniques for modulation identification were explained. Using the wavelet transform, different modulation techniques were identified in [67], [100] and [114].

The authors from [81] used a truncated series approximation of the likelihood ratio function to distinguish a BPSK from a QPSK only in low SNR cases. A PSK classification technique with improved sensitivity to parametric degradation is presented in [142].

Different reviews of digital modulation techniques are presented in [34], [145], [151], but the PSK modulation technique is pointed out in [73], [120], [121], [141].

Innovative approaches in the design and simulation of digital communication systems were made recently and can be found in [49] and [158], in which basic components of a digital communication system were simulated complete with channel and noise models.

Different papers, [103] and [104], written years later, provide estimations of different digital modulation techniques via computer simulation using Matlab software, and using the Matlab/Simulink environment, studies have been made [3] for complex communication systems, such as BFSK, BPSK, QPSK and QAM in universities, integrating these studies within the laboratory exercises made by students.

Although reconfigurable computing became popular in the mid 1980s due to the availability of FPGA, studies were made almost two decades later [14], [61] - [64], [153], [154], [165], [166]. So, at the end of 1990s and beginning of 2000s, the reconfigurable computing field was at its beginning [28], [85], [162], [163].

Paper [85] had been concentrated on the analysis of the Matlab/Simulink environment and its similarity to the VHDL language. It was the first time when a parallel between the two was made and pointed out that Matlab/Simulink could convert a structure into VHDL code which could lead to great effort and time savings in the design cycle. The disadvantage of that version of the program was that it was able to perform only direct mapping of the Simulink structures to VHDL.

In the middle of the 2000s, Matlab has successfully bridged the gap between the system developer and the researcher [52], [53]. In [18], an efficient design flow from Matlab to FPGA is presented. The proposed design flow was well suited for FPGA implementation and reduced the time for algorithm development. In the same period, System Generator appeared and restrictions as inaccuracy, slowness, and/or high complexity of any design, disappeared [146].

Thanks to the advances in FPGAs and many other factors suggested that students preparing to enter industry specializing in the fields of signal processing or computer architecture need to have a minimum knowledge of hardware digital signal processing implementations, different courses for teaching FPGA and DSP concepts, including HDL language and Matlab toolbox, had been introduced in universities curricula [54], [55], [78], [101], [133], [167].

After reviewing the FPGA architecture and its functionality for DSP implementation, typical DSP applications that can be mapped to FPGA devices are: image processing [93], [107], video processing [93], [108], [139], audio and speech processing [76], [109], coding and wireless communications [23], [29], [45], [60], [95], [96], [111], [130], [144], and more recently in biomedical applications [39], [182].

In the last years, FPGAs became an essential part in implementing DSP systems, especially in areas such as digital communications, since FPGAs have the ability to realize high-speed parallel operations, ideal for high-performance digital signal processing device. Since Xilinx has introduced System Generator, a system level tool for FPGA programming based on Matlab/Simulink environment, the gap in the DSP community had been reduced considerable since System Generator can be used in many ways: to explore an algorithm without translating the design into hardware, to simulate a design which is a part of something bigger or to implement any design in hardware. Considerable work about System Generator can be found in [51], [71], [95], [106], [134], [181].

Advances in technology and new demands on the existing system have now led to use Matlab/ Simulink environment in simulating a GPS (Global Positioning System) system [15], [21], [25], [26], [45], [69] from a satellite transmitter to a receiver [138]. This kind of application was also done on FPGAs [25], [90]. The GPS system has been widely used recently in life, such as automobile and mobile phone. The GPS is a space-based satellite navigation system that provides location and time information in all weather, anywhere on or near the Earth, where there is an

unobstructed line of sight to four or more GPS satellites. To verify the received function of GPS receiver, the authors from [90] proposed a simplified four-channel GPS Digital Satellite Signal baseband system using a low-cost FPGA prototyping which included the design of C/A code generator, the navigation messages processing and the BPSK baseband modulation. Experiments had shown that their proposed baseband system used only 10% percent of the FPGA resources [90]. Another GPS experiment was done using the QPSK modulation technique in [60], since several communication standards use this modulation as a main modulation scheme for its robustness against noise and very high data rate. The proposed design was capable of operating at a maximum data rate of 77 Mbps on Xilinx Virtex II-Pro FPGA board.

Since only the BPSK and QPSK modulation make the subject of this work, I will talk about only related work about these two modulation techniques.

Being a new field, the first FPGA implementations of the BPSK technique were quite recent and can be found in papers [11], [4] and [42]. All three works are based on implementing a BPSK detector, but I will speak only about [4] since it was my starting point in this direction.

The simulation system from fig. 3.16a contains two read-only memories. ROMA is used as a data source of the reference signal, equivalent to  $\phi_1(t)$ , define in equation (3.18). ROMB is used as a data source of the transmitted signal and can be either  $s_1(t)$  or  $s_2(t)$ , from equation (3.16). The signal from ROMB is further processed by inverting the phase conforming to the simulated data signal from the PN generator. The PN generator illustrated in fig.3.16b is a Linear Feedback Shift Register (LFSR) [172], [173].

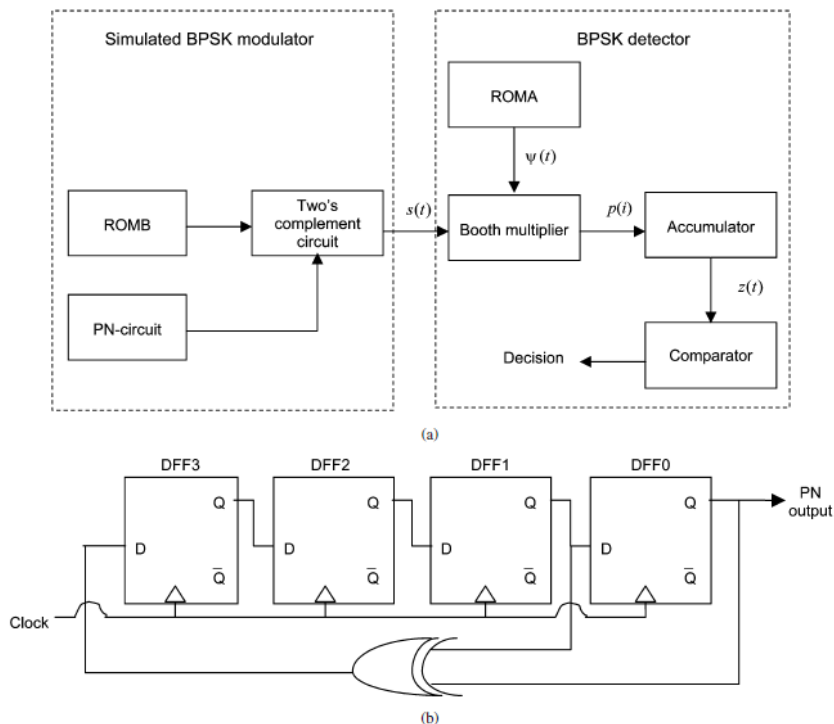


Fig. 3.16 (a) Architecture of digital implementation and simulation system  
(b) Logic diagram of the PN generator



The LFSR has as the input bit, a linear function depending on its previous states. Because of the finite number of possible states, the register will enter in a repeating cycle. So, the PN generator is used to generate a random sequence of bits that would be available at the receiver.

The sample carrier signal from ROMB and the random sequence of bits from the PN generator are fed to a two's complement circuit which generates sampled data which represents the BPSK modulated carrier. The two's complement circuit creates the BPSK carrier sampled data phase inversion when the PN generator signal indicates phase inversion.

Fig. 3.17 illustrates the BPSK carrier signal (dotted line) and the PN source data (solid line). For this experiment only eight samples have been taken. If the input data is "1", the transmitted signal is unchanged, but if the input data is "0", the transmitted signal is yielded with 180° phase shift.

The BPSK detector (fig. 3.16a) includes a Booth multiplier, an accumulator and a comparator. The Booth multiplier is a serial, signed multiplier. At the input of the multiplier, the data have 16 bits length, but because the multiplier requires 32 clock times to obtain the result, the output has 32 bits. The accumulator stores the 32 product samples from the multiplier. If the input data is "1", the reference signal and the transmitted signal have the same phase which means that the product value  $p(i)$  is positive and the accumulator value is positive. If the input data is "0", the two signals are out of phase with each other which means that the product value  $p(i)$  is negative and so, the accumulator value is negative.

The accumulator stores the sum of the product value. The accumulator represents the integral:

$$S \approx \int_0^{T_b} \phi(t)s_i(t)dt \quad (3.34)$$

where S is the accumulator value.

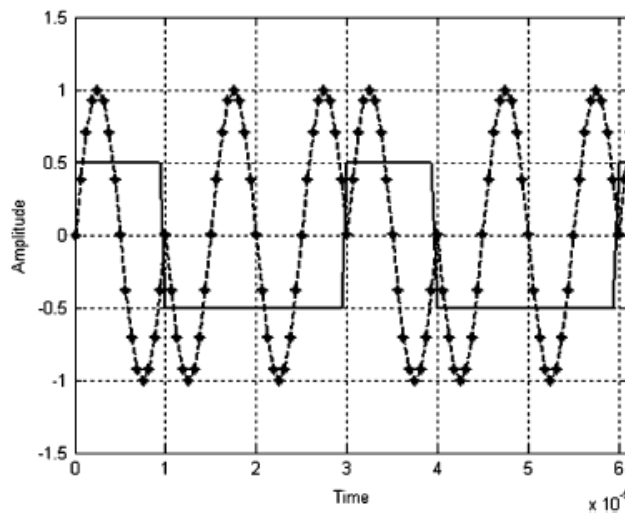


Fig. 3.17 BPSK carrier signal and PN source data

If the accumulator value is positive, "1" is transmitted. Otherwise, "0" is transmitted. In the real world, the poor signal quality indicates a low confidence in the data coming from the detector.

The process of deciding which symbol is transmitted, is called a detection process, illustrated in fig.3.18.

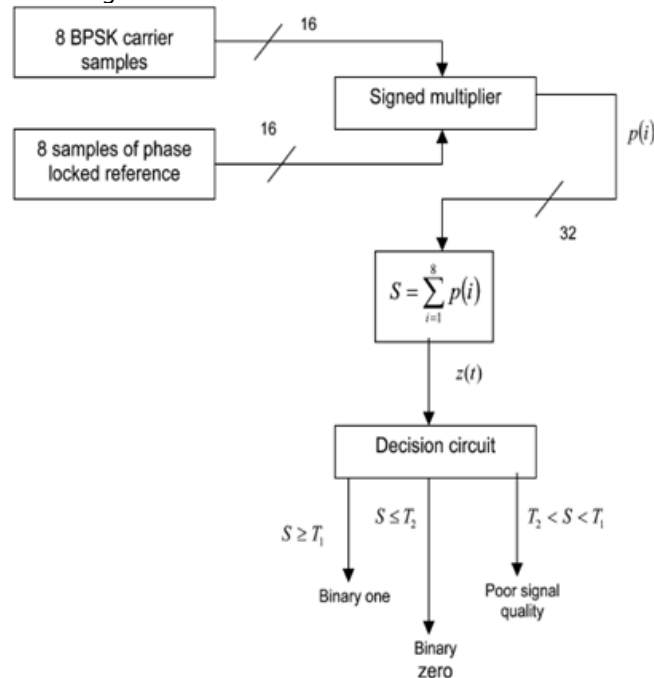


Fig. 3.18 Architecture of computational section of digital BPSK detector

In the last 5 years, more studies about implementation of the BPSK technique on had appeared [23], [27], [30], [37], [73], [84], [92], [97], [143], [155], [160].

In [23], an all digital wireless transceiver was presented, with a proposed technique for data recovery. A modified BPSK was used, with longer periods of phase change to enable data recovery in the circuit and without clock recovery. The transceiver was implemented and tested on FPGA and was connected to coils to perform actual short range wireless communication.

The design and implementation of DSP-based BPSK transmitter using FPGA was recommended in [27], and also HDL configurations of the expansion module (DAC) and of the clock synthesizer that controlled the FPGA system clock frequency. The DSP-based design of BPSK transmitter was developed in Xilinx System Generator and the verification of real-time result (DAC output signal) was done via observation from oscilloscope.

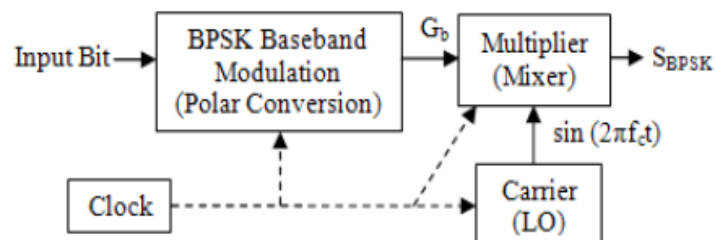


Fig. 3.19 BPSK Transmitter using FPGA [27]

The motivation for the research in [84] was to develop a customized digital communication system, based on the BPSK modulation principle, which determine relatively low BER under low SNR conditions. BPSK modulation is widely used approach in industrial telemetry systems for digital communication over noisy channels due to inherent high noise immunity and low BER.

In order to design a BPSK modulator and demodulator on FPGA, the method presented in [155] had used DSP Builder to substitute the VHDL programming and had been implemented on the Cyclone II DSP development board using QuartusII software and also used a new technology of frequency synthesis named DDS (Direct Digital Synthesis) [47].

In [160], a hybrid DSP/FPGA architecture of a BPSK transceiver was discussed with implementation is System Generator.

Recent studies also show that Phase Shift Keying, especially BPSK achieves a better signal-to-noise ratio (SNR) and thus it is a frequently applied technique for underwater communications [7], [38], [87] [112], which also make the subject of my future work. Among these work, paper [112] examined in detail the design of a new underwater acoustic modem targeted for high frequency communications. The FPGA modem was designed to modulate and demodulate a BPSK signal, which had been shown to be suitable for underwater communications. The capabilities offered by the reconfigurable FPGA platform were promising for a robust communication system.



Fig. 3.20 An underwater acoustic experiment platform [87]

Since not only the BPSK modulation scheme make the subject of this work, but also the QPSK technique, important studies were made in this field recently and can be found in [16], [17], [30], [36], [39], [50], [60], [92], [102], [135], [136], [140], [148], [156].

In [39], a new simple direct QPSK digital modulator model in MATLAB/Simulink environment was proposed. The modulator was successfully designed with VHDL programming code by Altera development kit. The experimentally measurements were obtained at 12.50 MHz carrier frequency and data rate 2Mbps, which also presented better performance with high data rate.

Paper [60] was based on the design and development of a programmable baseband modulator that perform the QPSK modulation schemes and as well as its other three commonly used variants to satisfy the requirement of several established 2G and 3G wireless communication standards: IS-95 (Interim Standard 95), UMTS (Universal Mobile Telecommunications System), GPS, DVB-S (Digital Video Broadcasting-Satellite) and SDARS (Satellite Digital Audio Radio Service).

In [136], a software-defined radio with QPSK modulation scheme has been successfully designed using Xilinx tools. Software simulation and hardware implementation procedures had shown that this type of hardware/software design methodology was well suited to a broad range of applications in communication and signal processing, including the digital wireless communication industry.

In [140], a new simple direct QPSK digital modulator model in MATLAB software was simulated and successfully designed using VHDL programming code. The modulator generated QPSK signal directly from binary digital data. The carrier frequency had been 5MHz and the input frequency, 500 KHz. The results were verified by testbench generated by the FPGA.

The method presented in [156], in order to design a QPSK modem on FPGA, used the DSP Builder to substitute the VHDL programming and was implemented on the Cyclone II DSP development board using QuartusII software. A new technology of frequency synthesis named DDS (Direct Digital Synthesis) [47] was also introduced.

A new technique, called Hardware Co-Simulation using FPGA appeared in the last two years. Studies using this new technique can be found in [50], [86], [98], [107], [137], [139]. The objective of the Hardware Co-Simulation technique is to accelerate an application. Therefore, an application is distributed in two or more hardware and software parts. Those parts of the application which are not critical for the process are kept in software, while the critical parts are implemented in hardware [107].

The Hardware Co-Simulation is provided by System Generator, based on the Matlab/Simulink environment, and makes possible the incorporating of a running design in FPGA directly into a Simulink simulation. The Simulink environment is used in order to verify the system functionality. When the design is made in Simulink, the results for the compiled portion are generated in hardware. This allows the compilation portion to be tested in actual hardware and can speed up the simulation dramatically. This method enables building a hardware version of the model and, using Simulink environment, several tests can be performed in order to verify the functionality of the system in hardware. Hardware co-simulation supports FPGAs from Xilinx on boards that support JTAG or Ethernet connectivity. When the compilation is complete, a new library, formed by a single block, encapsulates the hardware implementation of the DSP system. The block has inputs and outputs according with the number of the GatewayIn and GatewayOut ports. This new block includes all the functionality required for the system to be implemented on FPGA and is linked to a bitstream that will be downloaded into the FPGA during co-simulation. After starting the co-simulation, the System Generator will first download the associated bitstream. When the download is completed, System Generator reads the inputs from Simulation environment and sends the bitstream to the board using the JTAG connection. System Generator reads the output back from

JTAG and sends it to Simulink in order to be displayed. The results can be verified by comparing the simulation output to the expected output. VHDL code can also be generated from System Generator. System Generator not only generates the HDL and netlist files for any model during the compilation process, but it also runs the downstream tools necessary to produce an FPGA configuration file [129]. The Hardware Co-Simulation of the BPSK and QPSK systems will be also discussed in this work, in chapter 6.

### 3.4. Conclusions and Contributions

Chapter 3 begins with basics about digital communication systems. The components of these systems are presented, but also the functionalities in the receiver and the transmitter. Besides the components of a DCS, the three main criteria of choosing a modulation schemes make the subject of the first part of this chapter

The second part presents the backgrounds of the digital modulation techniques, underlining the BPSK and QPSK modulation schemes. Among all MPSK schemes, QPSK is the most used since it does not suffer from BER degradation while the bandwidth efficiency is increased. The advantage of QPSK over BPSK is obvious: QPSK transmits twice the data rate in a given bandwidth compared to BPSK, at the same BER. Two applications of the BPSK and QPSK modulation techniques were developed in the Matlab software.

In the third and last part of chapter 3, a survey about the BPSK and QPSK was presented. The survey begins with early work in the field of digital modulation. Although FPGA technology appeared in the mid 1980s, because of the gap between the existing software tools and the develop circuits, at the end of 1990s and first part of the 2000s, this filed was at its beginning. Matlab was the software environment who reduced the gap and made possible the FPGA implementation using System Generator. The study continued with different papers where the FPGA technology can be found, like image, video, audio and signal processing, coding and wireless communications, biomedical applications, including FPGA implementations of the BPSK and QPSK modulation techniques. The Hardware Co-Simulation technique was also mentioned and it would be later presented in this work.

#### Contributions:

- A review of the BPSK and QPSK, two very important modulation techniques, was made.
- MATLAB is generally used in research and development. In order to thoroughly analyse the efficiency of the simulator, two applications were developed, studied and compared.
- **A literature survey regarding the main previous work and general practice in phase modulation, of the BPSK and QPSK modulations, on FPGA had been conducted and summarized in table 3.4.**

**Table 3.4 Main previous work and practice in phase modulation**

Modulation scheme	Reference	Board	Resource Utilization		
			Flip-Flops	LUTs	Slices
B P S K	[27]	Virtex-4	1%	1%	1%
	[30]	Virtex-4	6%	27%	25%
	[97]	Spartan 3E	13%	13%	16%
	[143]	Virtex-II	1%	9%	10%
	[160]	Virtex-4	11%	11%	15%
Q P S K	[30]	Virtex-4	1%	16%	17%
	[60]	Virtex-II	1%	2%	2%
	[135]	Virtex-II	1%	13%	13%

## 4. ARGUMENTED BPSK MODULATION FROM FPGA PRESPECTIVE

To simulate, test and implement the actual process of the BPSK modulation technique is the subject of this chapter. Matlab/Simulink and System Generator are used for the theoretical study and two FPGA boards and Xilinx ISE software tool for the hardware implementation of the BPSK modulation technique.

The chapter is divided in three parts:

- The study, analysis and implementation of a BPSK system, made of a modulator and demodulator, on the same FPGA;



Fig. 4.1 First step in implementing the BPSK modulation technique

- The simulation, testing and implementation of a BPSK modulator on a single FPGA;

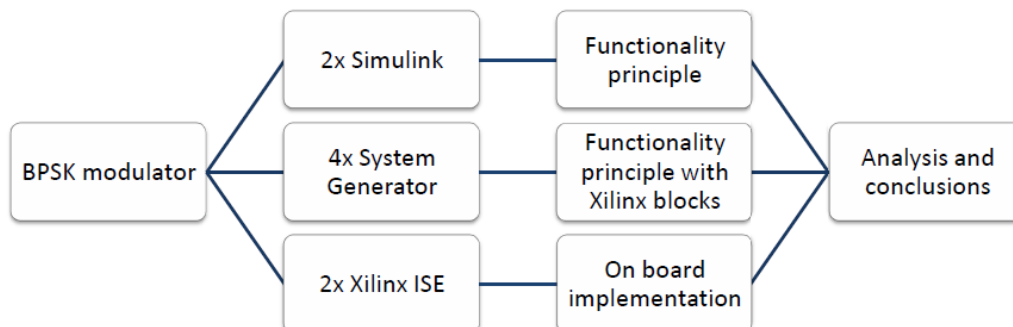


Fig. 4.2 Second step in implementing the BPSK modulation technique

- The third step in simulating, testing and implementation of a BPSK digital system on two FPGAs.

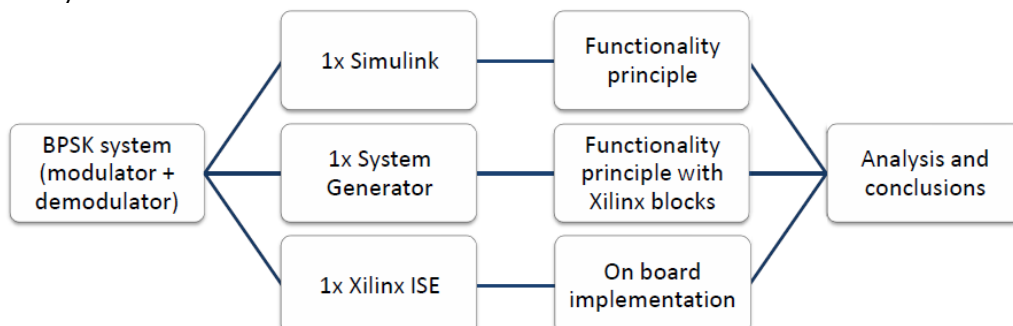


Fig. 4.3 Third step of implementing the BPSK modulation technique

#### 4.1. BPSK Detector

The first step taken in designing a BPSK system was the implementation of both the modulator and demodulator on the same FPGA board. The starting point was [4], which was presented in detail in chapter 3, and an improvement of the detector was made and materialized into [122].

The newness of the detector, illustrated in fig. 4.4, was made by modifying the LFSR which had improved the randomness sequence of the signal. Analysis and comparison were made on a Nexys2 board from Digilent [33]. The Spartan-3E FPGA [178] from the Nexys2 board has dedicated 18x18 bits hardware multiplier and so, there was no need for a relatively slow Booth multiplier.

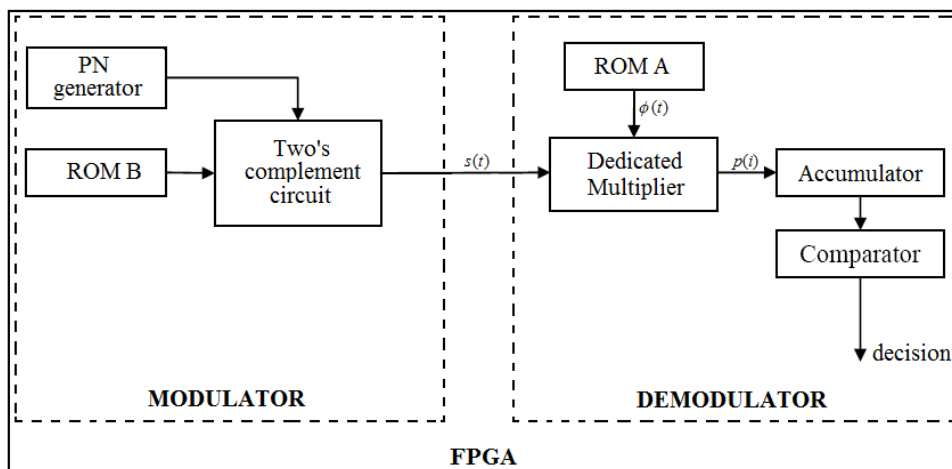


Fig. 4.4 The diagram of the BPSK detector

The main blocks of the improved BPSK Detector are: the two ROMs which generate the two sine signals, the PN Generator, the two's complement circuit, the dedicated multiplier, the accumulator, the comparator and the decision circuit.

The original architecture of the detector, fig.3.18, was kept but two changes were made in it. The modifications were resumed in modifying the LFSR and thanks to the dedicated multiplier on the Nexys2 board, the Booth multiplier was not necessary anymore.

The simulation system from fig. 4.4 contains two read-only memories. ROMA is used as a data source of the reference signal, equivalent to  $\phi_1(t)$  given by equation (3.18) and would be replaced, in real life, by a carrier-recovery circuit. ROMB is used as a data source of the transmitted signal and can be either  $s_1(t)$  or  $s_2(t)$ , which also were defined in equation (3.16). The ROMA and ROMB are sinusoids, but in this work are represented discontinuous, by instantaneous samples of 16 different values. The signal from ROMB is further processed by inverting the phase conforming to the simulated data signal from the PN generator. The PN generator is a LFSR and it is used to generate a random sequence of bits that would be available at the receiver. The sample carrier signal from ROMB and the random sequence of bits from the PN generator are fed to a two's complement circuit which generates sampled data which represents the BPSK modulated carrier. The two's



complement circuit creates the BPSK carrier sampled data phase inversion when the PN generator [173] signal indicates phase inversion [4]. The PN circuit is used to generate a random data pattern according to fig. 4.5.

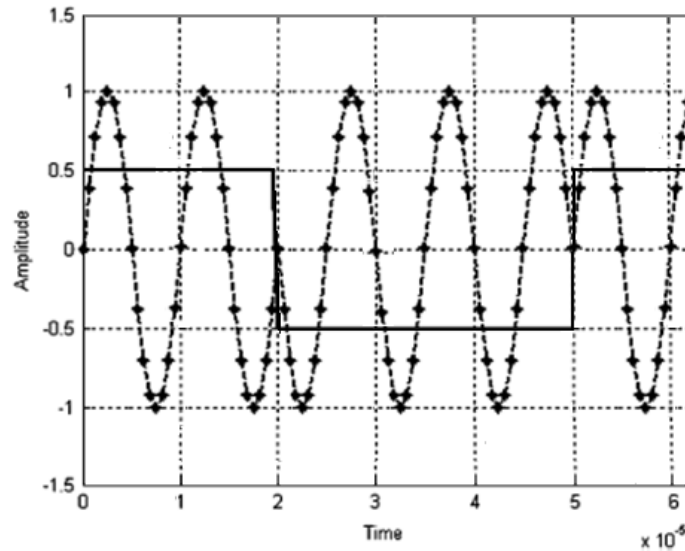


Fig. 4.5 BPSK carrier signal and PN source data

If the input data is "1", the transmitted signal is unchanged, same as the carrier waveform, but if the input data is "0", the transmitted signal is yielded with 180° phase shift.

The detector also includes an accumulator that stores the product samples from the multiplier, but the final decision is made depending upon the final numerical value of the accumulator. The accumulator is represented by the integral defined in (3.34). According to the computational architecture of the BPSK Detector, if the input data is "1", the reference signal and the transmitted signal have the same phase which means that the product value  $p(i)$  is positive and the accumulator value is positive. If the input data is "0", the two signals are out of phase with each other which means that the product value  $p(i)$  is negative and so, the accumulator value is negative.

The principle of the experiment is shown in fig. 4.6. The Setup Lab measurement used for realizing the BPSK Detector is illustrated in fig. 4.7

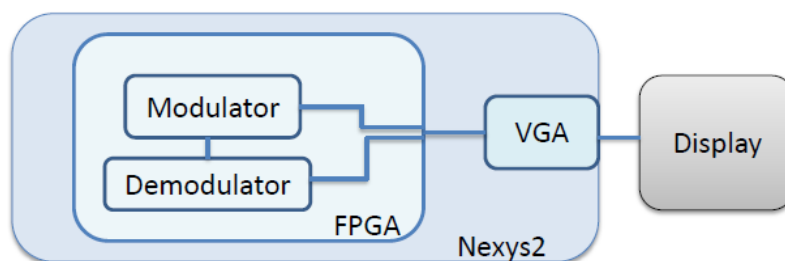


Fig. 4.6 Principle of the BPSK Detector

The resources used are the Nexys2 board, a computer with the Xilinx WebPack ISE on it and a monitor. The ISE WebPack from Xilinx is a fully featured front-to-back FPGA design solution and it offers HDL synthesis and simulation, implementation, device fitting and JTAG programming. The hardware language used from the ISE software was VHDL. The modulated signal obtained by programming the board, was routed to the VGA port. The monitor was attached to the VGA port of the board and the modulated signal could be seen on it.



Fig. 4.7 Test bench lab [122]

After implementing the BPSK Detector on the Nexys2 board, the signals were routed to a monitor. Fig. 4.8 and 4.9 are pictures taken with a photo after routing the signals to the monitor. The BPSK modulation can be seen in them. If the input data is "1", the transmitted signal to the monitor is unchanged and has a green border on the right of the figure, but if the input data is "0", the transmitted signal is yielded with 180° phase shift and has a red border on the right of the figure.



Fig. 4.8 The transmitted signal if the input is 1



Fig. 4.9 The transmitted signal if the input is 0

A module selection optimization of the resources map as an indicator of the FPGA architecture, in which, balancing FPGA resource utilities is treated as a constraint, is illustrated. Using fewer FPGA resources such as FPGA slices, block RAMs, and block multipliers is a desirable process in many FPGA applications.

Fig. 4.10 represents the design summary which represents the utilization of flip-flops, LUTs, slices used from the capabilities of the FPGA from the Nexys2 board while fig. 4.11 represents the resources map of the BPSK detector on the Nexys2 board.

Device Utilization Summary				[-]
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	693	9,312	7%	
Number of 4 input LUTs	940	9,312	10%	
Number of occupied Slices	755	4,656	16%	
Total Number of 4 input LUTs	1,059	9,312	11%	
Number of bonded IOBs	78	232	33%	
Number of BUFGMUXs	3	24	12%	
Number of MULT18X18SIOs	1	20	5%	
Average Fanout of Non-Clock Nets	3.18			

Fig. 4.10 Design Summary of the BPSK Detector

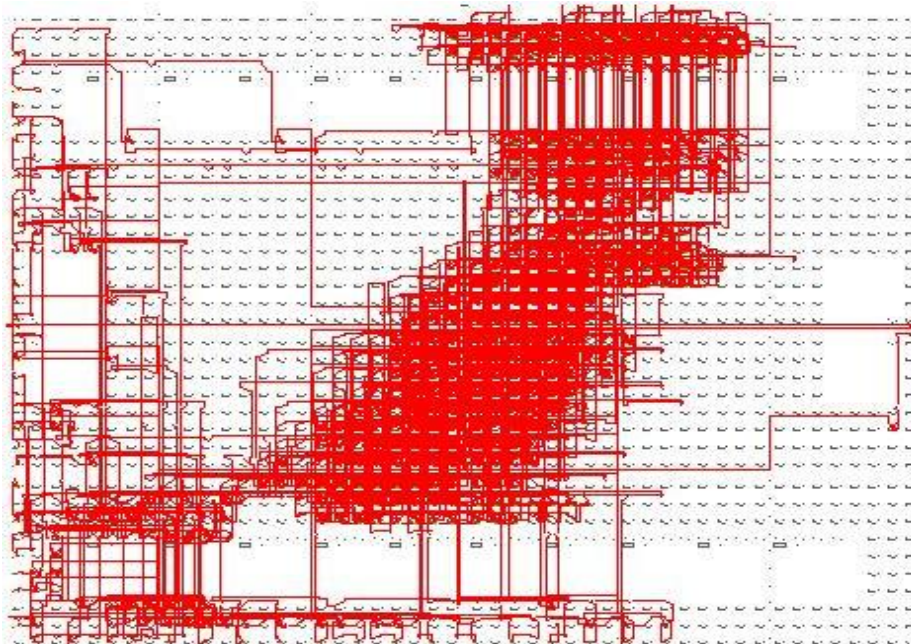


Fig. 4.11 The resources map of the BPSK Detector on the Nexys2 board

## 4.2. BPSK Modulator

The second experiment was to implement a digital communication system based on the BPSK modulation technique on two identical boards, the first board comports as a modulator and the second one, as a demodulator. The information was sent from the transmitter to the receiver through a communication channel affected by AWGN noise.

In order to implement the BPSK system, first the modulator [123], [184] was simulated, tested and implemented on a Spartan 3E board. To test the modulator functionality, Matlab/Simulink environment was used. Afterwards, the design was tested using FPGA blocks in System Generator [177], [179]. System Generator is a software tool for modeling and designing FPGA-based DSP systems in Simulink. It extends Simulink in many ways to provide a modeling environment that is well suited to hardware design and it produces command files for FPGA synthesis and HDL simulation. The final step was to implement the BPSK modulator on the Spartan 3E board using Xilinx ISE 12.3 software tool.

### 4.2.1. BPSK Modulator in Simulink

Fig. 4.12 shows an implementation of a BPSK modulator in the Simulink environment and fig. 4.13 the waveforms generated by the corresponding blocks. By convention, the figures from top to bottom are numbered starting with a.

The Simulink block set contains:

- the sine wave block (fig.4.12) which generates a sine waveform (fig. 4.13a);
- the gain block (fig.4.12) which introduce a 180° phase difference of the sine waveform (fig.4.13b);
- the random source block (fig.4.12);
- the rounding function block (fig.4.12);

The random source block and the rounding function block generate the binary sequence or the modulating signal (fig. 4.13c).

- the switch (fig. 4.12) which will choose between the first or third output depending on the value of the second input. If the second input is "1", the output value will be sinus, but if the second input is "0", the output will be -sinus (fig. 4.13d).

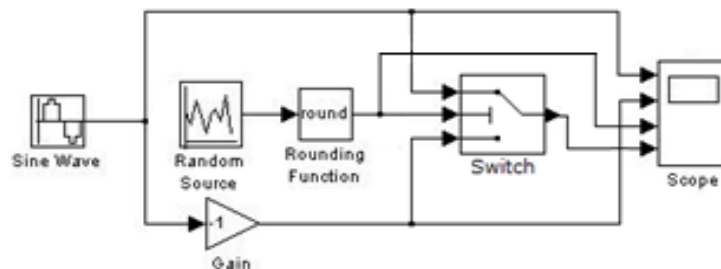


Fig. 4.12 BPSK Modulator in the Simulink environment



Fig.4.13 The waveforms on the scope:  
 (a) Sine; (b) -Sine; (c) Modulating signal; (d) Modulated signal

Fig. 4.14 represents a second implementation of the modulator in the same Simulink environment. The Simulink block set contains approximately the same blocks, the main difference been the embedded Matlab function. This block contains a Matlab language function in a Simulink model. The block accepts multiple inputs and produces multiple outputs [79]. The Matlab code of the embedded Matlab function is shown in table 4.1 [123], [184]. The waveforms of the second simulation are the same as the ones in fig. 4.13.

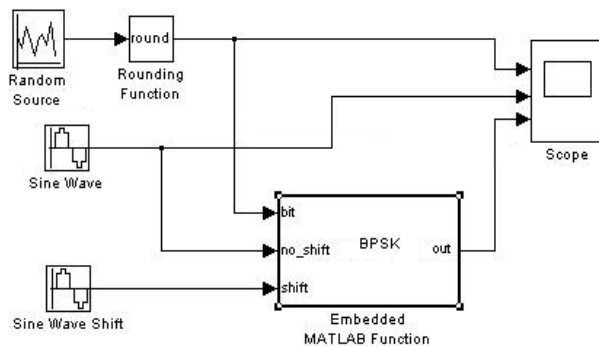


Table 4.1 Matlab code of the embedded function

```
function out = BPSK(bit,
no_shift, shift)
if bit==1
    out=no_shift;
else out=shift;
end
```

Fig. 4.14 Second implementation of the BPSK Modulator in Simulink

### 4.2.2. BPSK Modulator in System Generator

Fig. 4.15 illustrates an implementation of the BPSK modulator in System Generator.

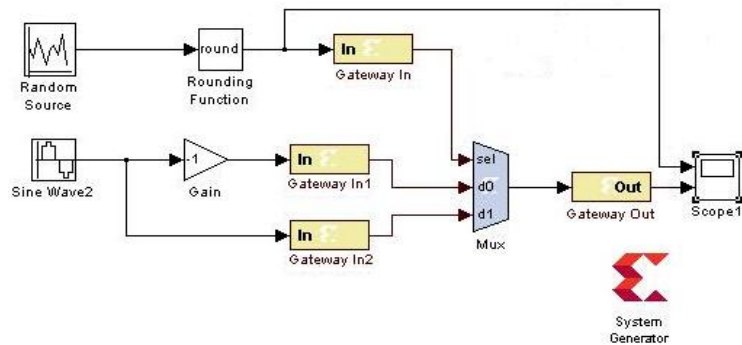


Fig. 4.15 BPSK Modulator in System Generator

The Simulink Blockset contains:

- the sine wave block;
- the gain block;
- the random source block;
- the rounding function block;
- the scope.

The System Generator Blockset contains:

- the gateway in blocks which are the inputs into the Xilinx area of the Simulink design;
- the mux which implements a multiplexer;
- the gateway out block which is the output from the Xilinx area of the Simulink design.

The carrier as well as the modulating signal is generated external and the modulated signal is created inside the board and then routed to be seen on the scope as indicated in fig. 4.16.



Fig. 4.16 The modulating and modulated signals

In fig. 4.17, the carrier is generated external, but the modulating signal is generated internal by a LFSR. Fig. 4.18 illustrates the carrier signal, the sine and -sine waveforms, as well as the modulating signal from the LFSR and the modulated signal obtained inside the FPGA.

The Simulink Blockset contains:

- the sine wave blocks;
- the scopes.

The System Generator Blockset contains:

- the gateway in blocks;
- the mux;
- the gateway out blocks;
- a LFSR.

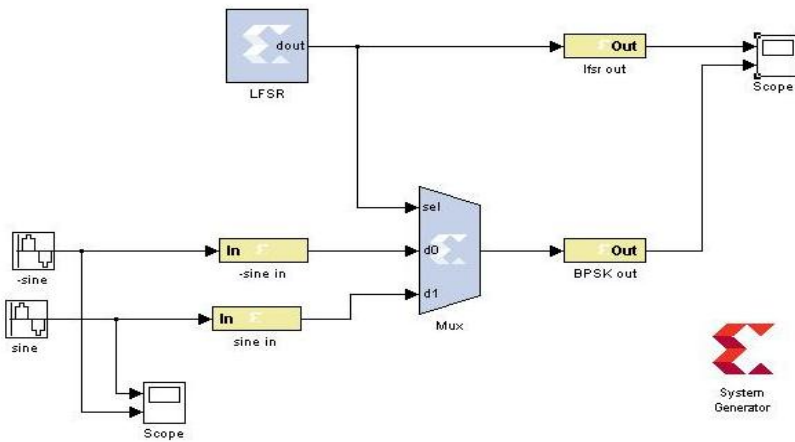


Fig.4.17 A second implementation of the BPSK Modulator in System Generator

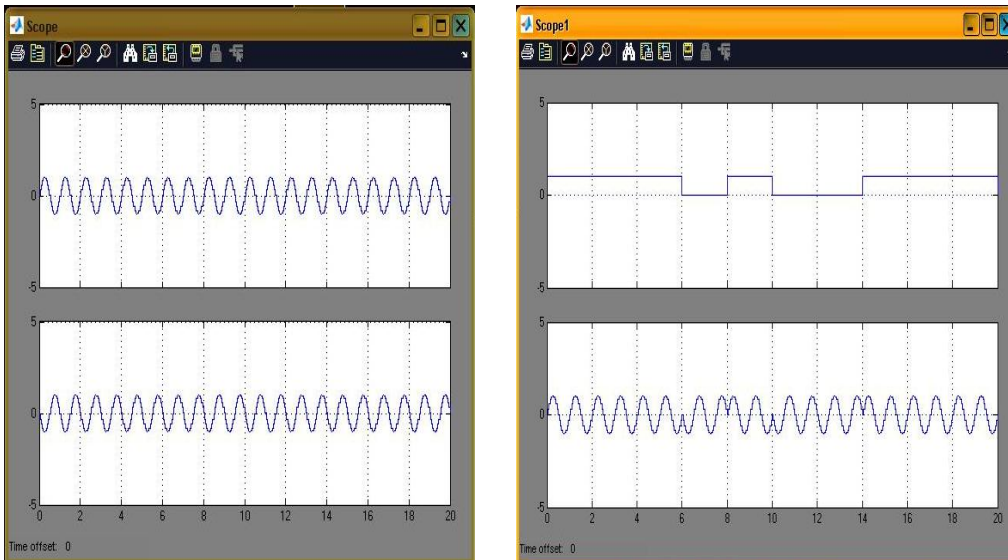


Fig. 4.18 The waveforms: (a) Sine; (b) -Sine; (c) Modulating signal generated by the LFSR; (d) Modulated signal

The third implementation of the BPSK Modulator, illustrated in fig. 4.19 consists of signals generated internal. The carrier is generated internal by DDS blocks from System Generator and the modulating signal can be generated internal by the LFSR or external. Fig.4.20 illustrates the signals obtained after testing the modulator.

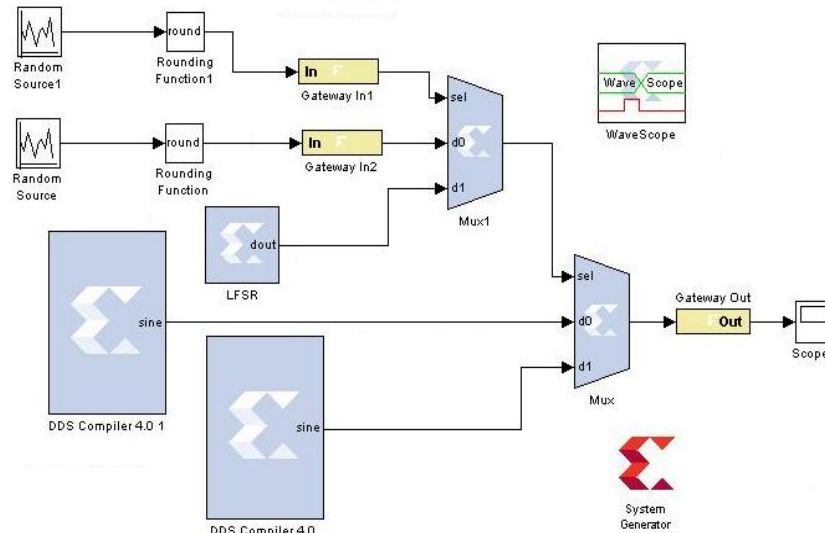


Fig.4.19 The third model of the modulator

The Simulink Blockset contains:

- the random source;
- rounding function;
- the scope.

The System Generator Blockset contains:

- the gateway in blocks;
- the mux blocks;
- the gateway out block
- the LFSR block;
- two DDS compiler blocks.

The DDS Compiler Block is a direct digital synthesizer and it uses a lookup table scheme to generate sinusoids. The principle of the DDS block is described in detailed in [155]. A digital integrator generates a phase that is mapped by the lookup table into the output waveform [177]. The sinusoids can be seen in fig. 4.20 (c) and (d).

The mux block implements a multiplexer. It has one select input and a configurable number of data inputs that can be defined by the user [177]. The d0 and d1 inputs of mux1 represent the modulating signal. The sel input of mux1 selects between the d0 and d1 inputs. Because in the System Generator environment, a switch cannot be represented, it was replaced with a random sequence of bits. The same thing happened in mux2, where depending on the output of mux1, either sinus or  $-\text{sinus}$  was chosen.

With the System Generator WaveScope, the user can view the waveforms generated in a design. It is well suited for analyzing and debugging System



Generator designs. The wavescope allows the user to observe the time-changing values of any wires in the design after the conclusion of the simulation [177].

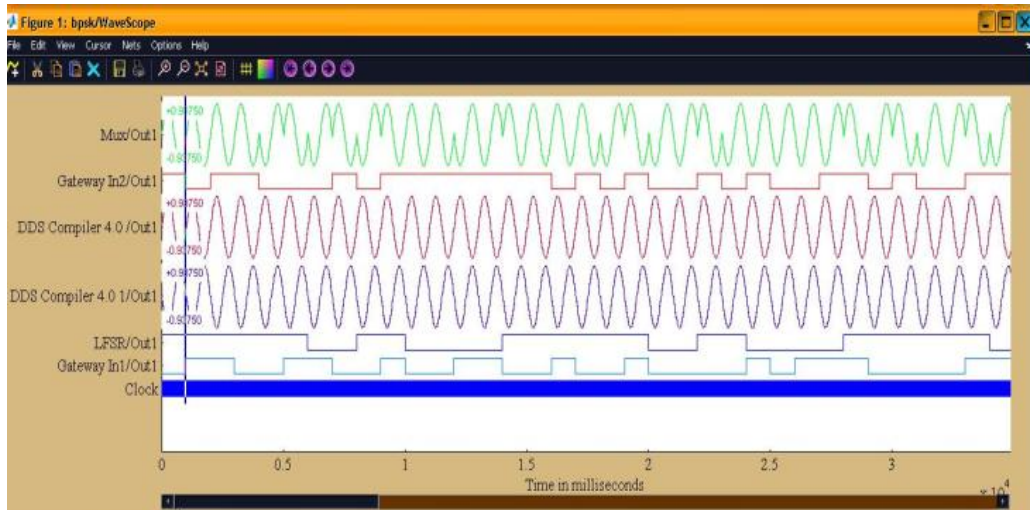


Fig. 4.20 The waveforms: (a) The modulated signal; (b) The modulating signal; (c) Sine; (d) -Sine; (e) The output of the LFSR; (f) The signal obtained external, from a function generator

A fourth implementation of the BPSK modulator is illustrated in fig.4.21. The modulating signal (fig. 4.22c) is generated internal by the LFSR from fig.4.21. The carrier is also generated internal by DDS blocks from System Generator (fig. 4.21). The DDS Compiler Block is a direct digital synthesizer and it uses a lookup table scheme to generate sinusoids. A digital integrator generates a phase that is mapped by the lookup table into the output waveform [79], [80]. The sine waveforms can be seen in fig.4.22 (a) and (b). The mux block implements a multiplexer. It has one select input and a configurable number of data inputs that can be defined by the user. The d0 and d1 inputs of mux represent the sine waves. The sel input of mux represents the modulating signal and selects between the d0 and d1 inputs. If LFSR generates a '1', the modulated signal remained same as the carrier, but if '0' was transmitted, the yielded carrier is transmitted.

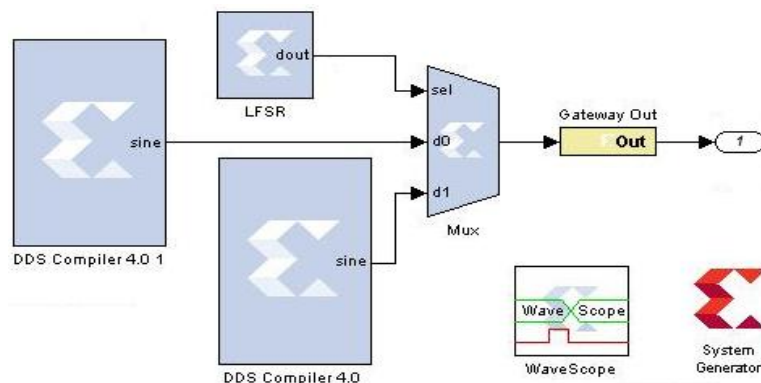


Fig. 4.21 The fourth implementation of the BPSK Modulator in System Generator

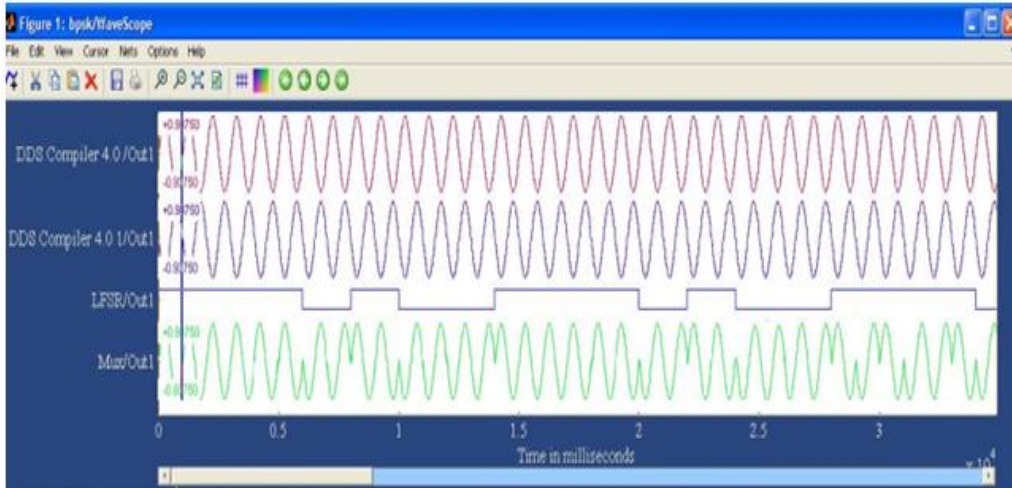


Fig. 4.22 The waveforms: (a) Sine; (b) -Sine; (c) The modulating signal; (d) The modulated signal

#### 4.2.3. BPSK Modulator on FPGA via VGA

The first implementation of the BPSK modulator has, as a model, the third implementation in System Generator with the signals routed to a VGA monitor. The carrier is generated internal, but in a ROM and that is the reason of which the sinus signal is represented discontinuous, by instantaneous samples of 16 different values [4], [121]. The only thing that is different is that a switch has been used that replaced the mux1 block. The switch behaves as a random sequence of bits which introduces either "1" or "0" depending on its position.

Fig. 4.23 represents the test bench lab used in implementing the BPSK Modulator on the Spartan 3E Starter Kit Board. The experimental setup consists of a computer, a monitor, an oscilloscope, a pulse generator and the Spartan 3E board. The ISE Web Pack runs on the computer and it programs the Spartan 3E board.



Fig. 4.23 Test bench lab of the BPSK modulator on VGA

The pulses of the generator are fed to an entry of a connector on the board. Depending on the position of a slide switch, the modulating signal is acquired either external, from the pulse generator or internal, from the LFSR. Opposite to System Generator, the switch can be represented or can be configured in VHDL language. The modulated signal obtained is routed to the VGA port of the board, in order to be seen on the monitor.

The waveforms of the BPSK modulator can be seen in fig.4.24 and fig.4.25. If the input data is "1", the transmitted signal to the monitor is unchanged and has a green border on the right of the figure, but if the input data is "0", the transmitted signal is yielded with 180° phase shift and has a red border on the right of the figure.

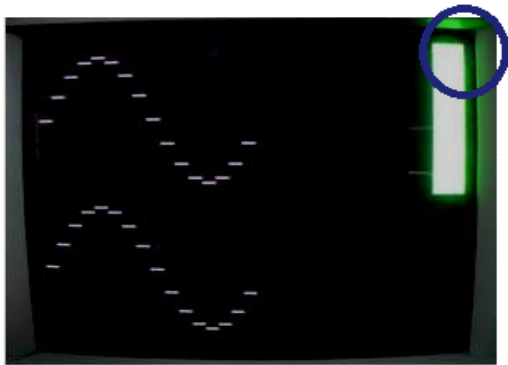


Fig. 4.24 The transmitted signal if the input is 1

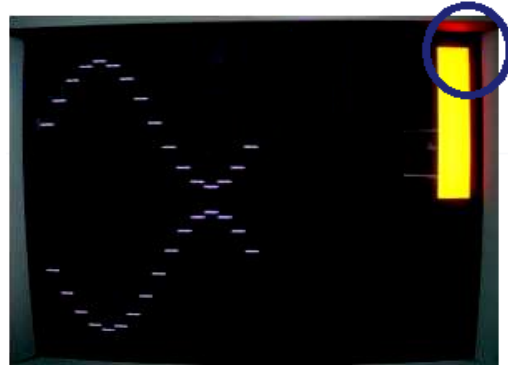


Fig. 4.25 The transmitted signal if the input is 0

Fig. 4.26 represents the design summary which represents the utilization of flip-flops, LUTs, slices used from the capabilities of the FPGA from the Spartan 3E board, while fig. 4.27 shows the resources map inside the board.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	679	9,312	7%	
Number of 4 input LUTs	952	9,312	10%	
Number of occupied Slices	765	4,656	16%	
Total Number of 4 input LUTs	1,101	9,312	11%	
Number of bonded IOBs	39	232	16%	
Number of BUFGMUXs	3	24	12%	
Average Fanout of Non-Clock Nets	3.18			

Fig. 4.26 Design Summary of the first implementation of the BPSK Modulator on FPGA

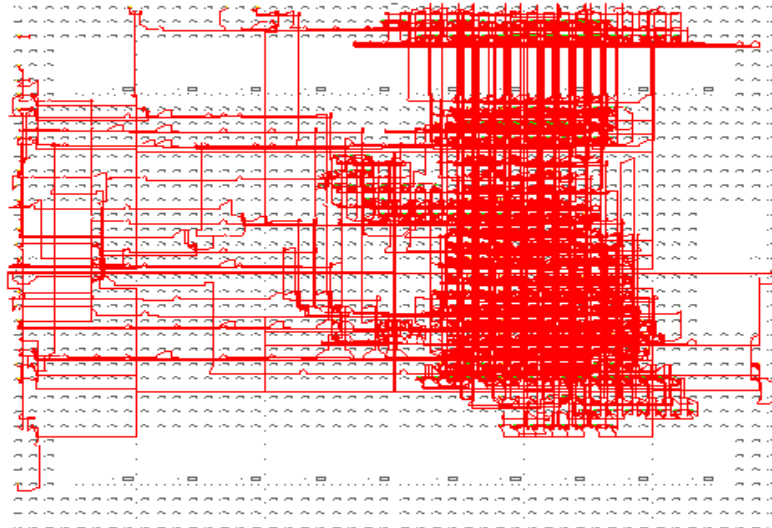


Fig. 4.27 The resources map of the BPSK Detector on the Spartan 3E board

#### 4.2.4. BPSK Modulator on FPGA via oscilloscope

The second implementation of the BPSK modulator has, as a model, the fourth implementation in System Generator, illustrated in fig.4.21. The experimental setup consisted of a computer, a Spartan 3E board and an oscilloscope. The Matlab/Simulink, System Generator and Xilinx ISE packages run on the computer, and the Xilinx ISE programs the Spartan 3E board.

Fig.4.28 illustrates the Spartan 3E board which comports as a modulator.

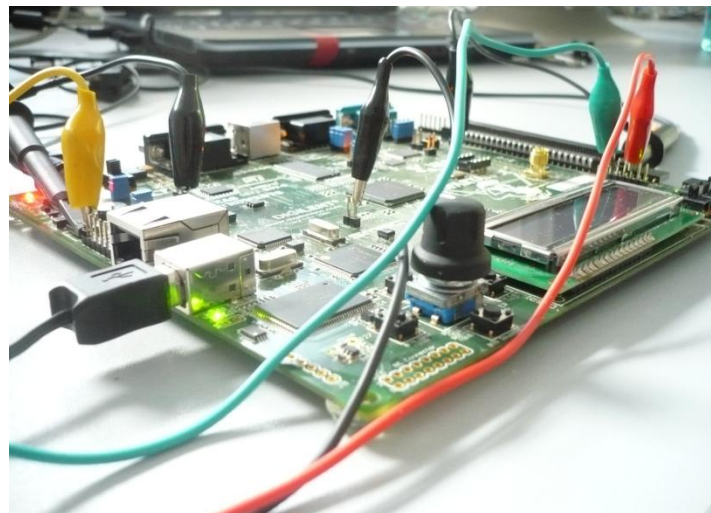


Fig. 4.28 BPSK Modulator board

The modulating signal is generated internal, in the modulator, by a LFSR. The carrier is also generated internal, and is made of 16 different values kept in a ROM memory [4]. The yielded carrier with 180° phase shift is obtained by reading the ROM memory later with 8 samples. If LFSR was '1', the modulated signal remained same as the carrier, but if '0' was transmitted, the modulated signal became the yielded carrier. The modulated signal is then sent to the DAC on the board in order to be sent through a channel. The principle of the BPSK modulator implemented on the FPGA [127], [128], [184] is illustrated in fig.4.29.

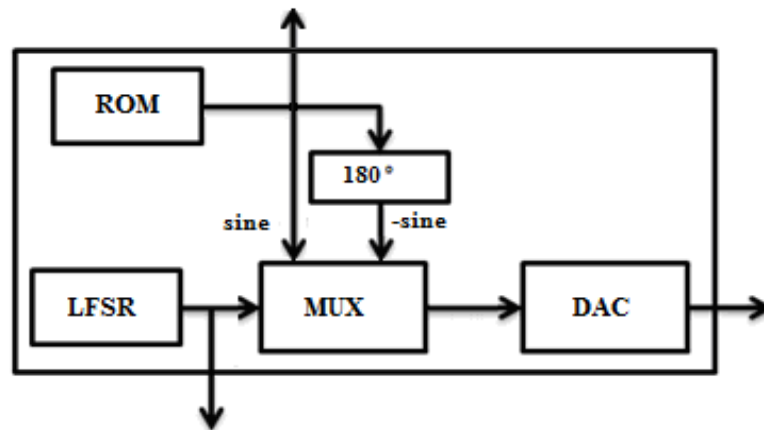


Fig. 4.29 The principle of the BPSK modulator on the FPGA

In the validation and data collecting part of the implementation, the signals were routed to a LeCroy WaveSurfer Xs Series oscilloscope. Fig. 4.30 illustrates the signals from the modulator. The first one represents the modulating signal generated by the LFSR, the second one, the carrier and the third, the modulated signal.

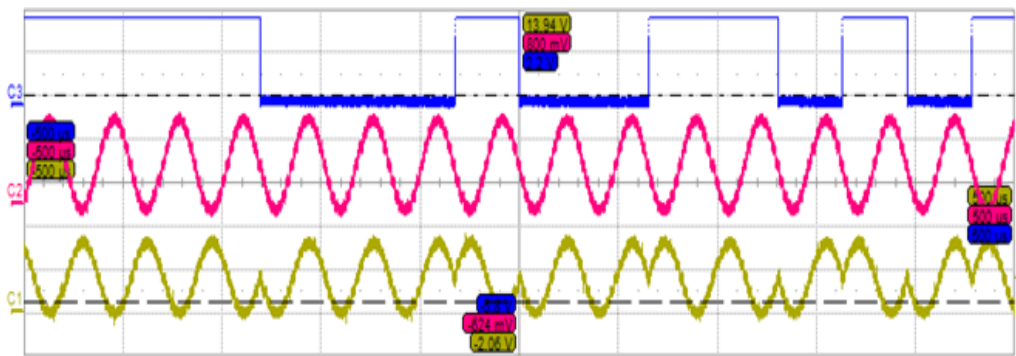


Fig. 4.30 The waveforms: (a) The modulating signal; (b) Sine; (c) The modulated signal

Due to the limitations of the serial DAC on the board, relatively slow data rates of the carrier and modulated signals were obtained. The carrier was generated at 31,25 KHz and the output frequency of BPSK modulated signal was obtained at the same frequency. In order to obtain higher frequencies, the slow DAC must be replaced with external devices for a complete wireless/underwater system.

Fig. 4.31 illustrates the design summary of the modulator board. The design summary shows the various synthesizer options that were enabled and some device utilization and timing statistics for the synthesized design, while fig. 4.32 represents the resources map of the BPSK modulator on the Spartan 3E board.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	97	9,312	1%	
Number of 4 input LUTs	111	9,312	1%	
Number of occupied Slices	94	4,656	2%	
Total Number of 4 input LUTs	114	9,312	1%	
Number of bonded IOBs	20	232	8%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	3.34			

Fig. 4.31 The Design Summary of the second implementation of the BPSK Modulator.

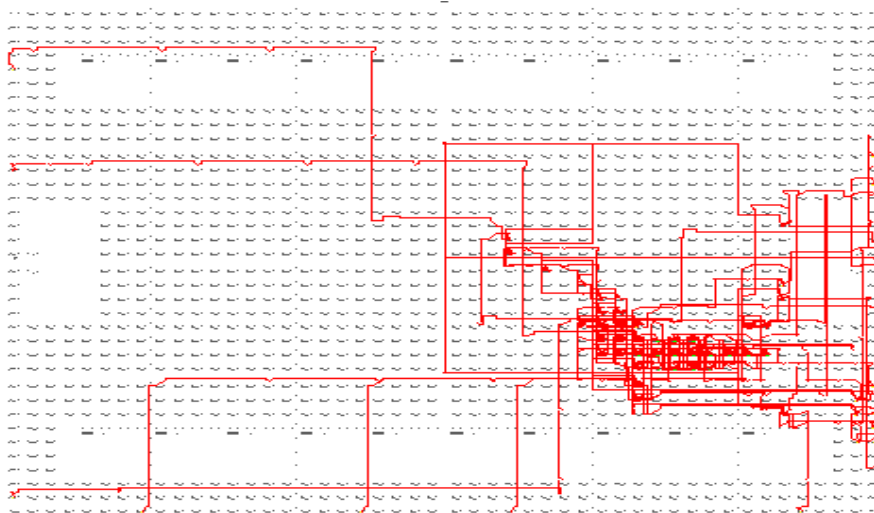


Fig. 4.32 The resources map of the BPSK modulator on the Spartan 3E Board

### 4.3. BPSK System

The final step in implementing a digital communication system based on the BPSK modulation technique was to simulate and test the demodulator part. In order to test the demodulator, in every design the modulator had to appeared. The modulator part will not be the subject of this topic and it will be represented as a block. Its functionalities were described in the previous topic and remained the same in this topic also.

The demodulator functionality was simulated using Matlab/Simulink environment. Afterwards, the design was tested using FPGA blocks in System Generator. The final step was to implement the BPSK system [128], the modulator and demodulator parts, on two Spartan 3E boards using Xilinx ISE 12.3 software tool.

**4.3.1. BPSK System in Simulink**

Fig. 4.33 represents a communication system implemented in the Matlab/Simulink environment that uses the BPSK modulation technique. The system [128] is composed of the binary data source, a modulator, a channel and a demodulator. The previous chapter made the subject of the binary data source and the modulator part.

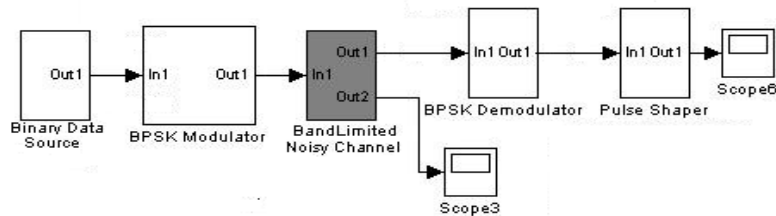


Fig. 4.33 BPSK System

The modulated signal from fig. 4.13 is then pass through a channel where noise is added. The channel also has a limited frequency bandwidth so that it can be viewed as a filter (fig.4.34). The corresponding signals are shown in fig.4.35.

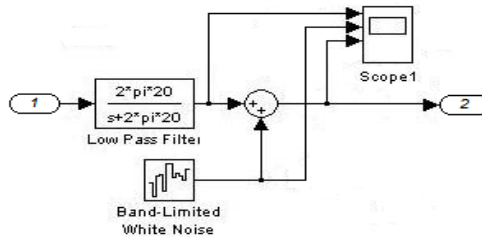


Fig.4.34 Band Limited Noisy Channel

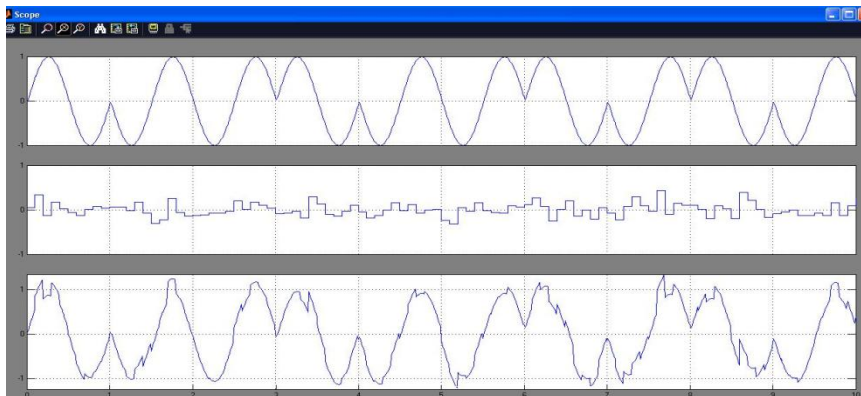


Fig. 4.35 The waveforms shown on the scope: (a) the modulated signal; (b) the noise; (c) the noise added to the modulated signal

The modulated signal added with noise arrives at the input of the demodulator. The first block in the demodulator is saturation (fig.4.36). The saturation block [79] establishes upper and lower bounds for an input signal. If the input signal is within the limits of upper and lower bounds, the input signal passes through unchanged, otherwise the signal is clipped to the upper and/ or lower bounds (fig.4.37).

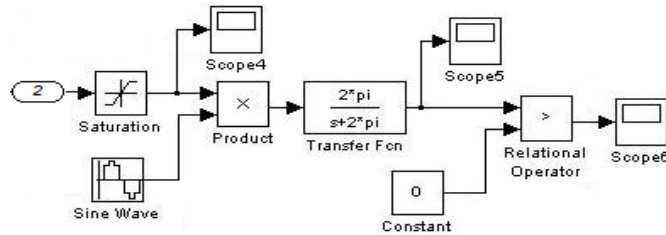


Fig. 4.36 BPSK demodulator



Fig. 4.37 The signal at the output of the saturation block

After limiting the signal to the upper and lower bounds, it is multiply by a sine waveform, which is the carrier obtained in theory from the carrier recovery circuit and then passed through the transfer function block which implements a transfer function between the input and the output. The signal obtained is illustrated in fig.4.38.

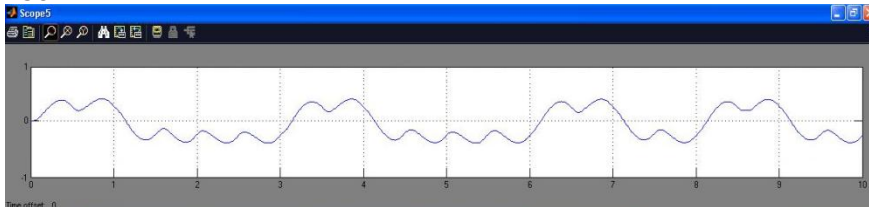


Fig. 4.38 The signal at the output of the transfer block

The obtained signal enters the pulse shaper block and at its output the demodulated signal is found (fig.4.39).

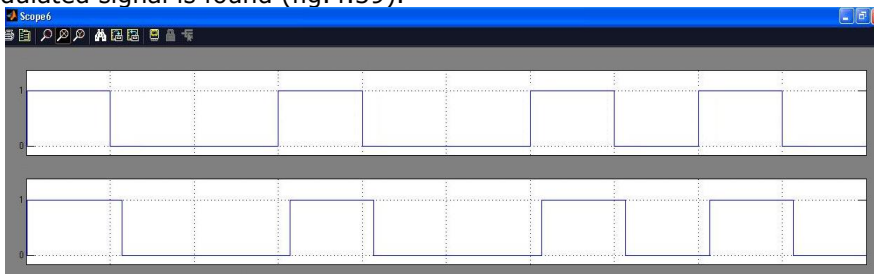


Fig. 4.39 (a) The modulating signal; (b) The demodulated signal



### 4.3.2. BPSK System in System Generator

The modulator part is the same as the fourth implementation illustrated in fig. 4.21. The modulating signal (fig.4.22c) is then pass through the same channel (fig. 4.34) where noise is added and arrives at the input of the demodulator (fig.4.40).

The carrier is recovered due to the DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal is then added with all the samples, multiplied, from a period. This operation takes place in the accumulator. Once we have a result, it is compared with a threshold. If the compared signal is positive, the demodulator take the decision that '1' was transmitted, otherwise, '0'.

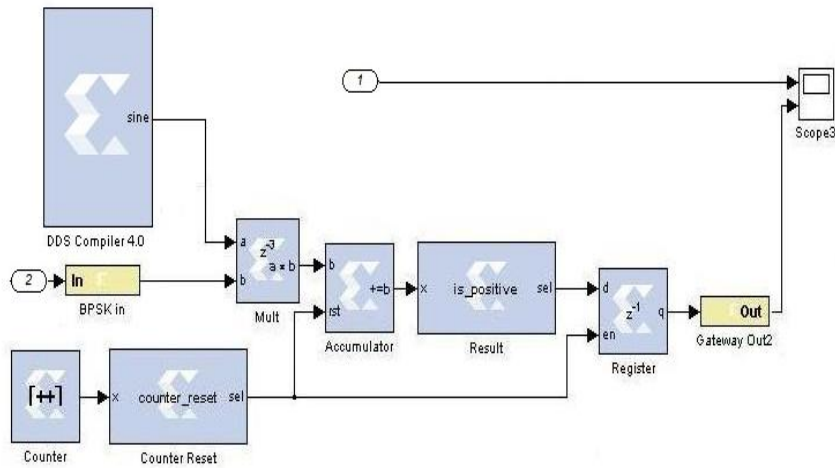


Fig. 4.40 BPSK Demodulator in System Generator

Fig.4.41 illustrates the modulating signal generated in the modulator and the demodulated signal obtained after the demodulation operation.

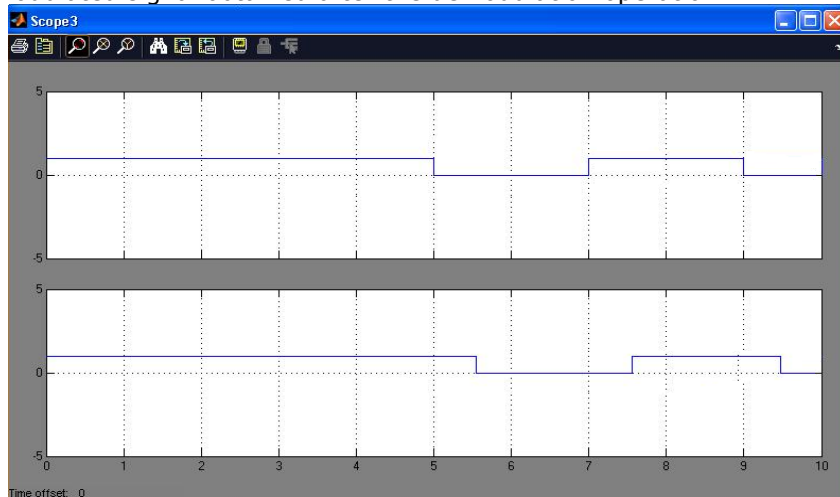


Fig. 4.41 (a) The modulating signal; (b) The demodulated signal

### 4.3.3. BPSK System on FPGA

The BPSK System (fig.4.42) = Modulator (fig.4.28) and Demodulator (fig.4.44) implemented on the Spartan 3E board is, exactly, the implementation in System Generator. The carrier is generated internal, in a ROM [128].

The BPSK system (fig.4.42) consists of two Spartan 3E boards, first behaves as a modulator and the second one, as a demodulator. The connections between the two boards are made of three wires: first comports as a communication channel, the second as an asynchronous reset signal and the last one for the synchronization of the two boards.

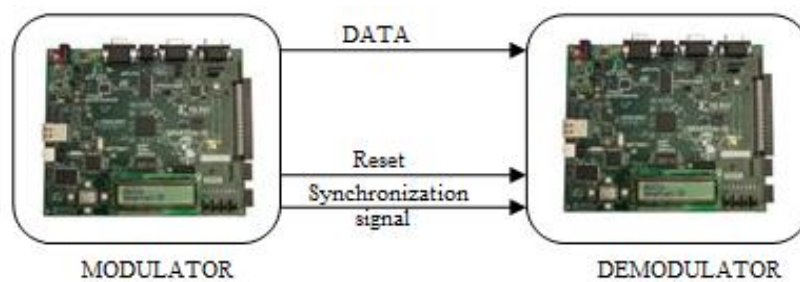


Fig. 4.42 BPSK System

The yellow cable represents the communication channel, the red one is the reset and the green one makes possible the synchronization between the two boards. The experimental setup of the BPSK system is shown in fig. 4.43 and in fig. 4.44 the demodulator board.

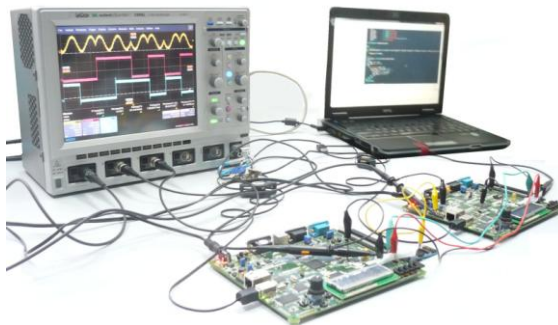


Fig. 4.43 BPSK System – experimental setup



Fig. 4.44 BPSK Demodulator board

The principle of the BPSK demodulator implemented on the FPGA is illustrated in fig.4.45. The modulated signal affected with noise arrives at the second board which behaves as a demodulator. The signal enters the demodulator with the help of a pmod AD1 which transforms the analog signal into a digital one. This digital signal is then multiply with the recovered carrier, generated internal in a ROM memory, practically an integration was achieved. The result is kept in an accumulator and compared with a decision threshold and so, the demodulated signal is obtained (fig. 4.45).

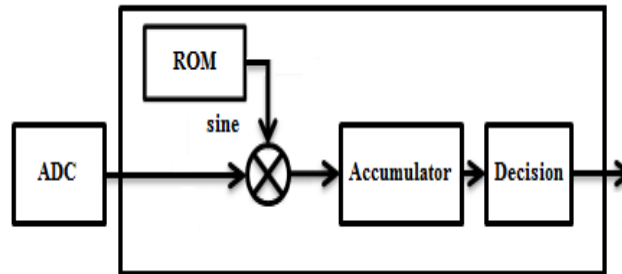


Fig. 4.45 The principle of the BPSK demodulator on the FPGA

After implementing the BPSK System made of a modulator and demodulator on the two Spartan 3E Starter Kit boards, the signals were routed to the LeCroy WaveSurfer Xs Series Oscilloscope. Fig. 4.46 illustrates the signal on the demodulator board: the modulated, the modulating and the demodulated signals.

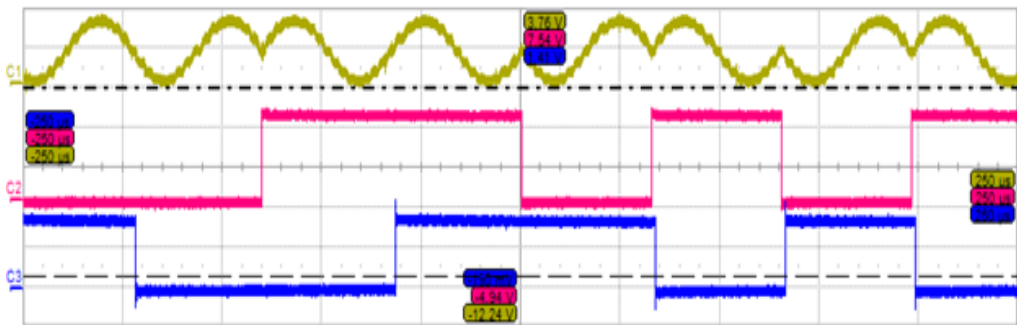


Fig. 4.46 The waveforms: (a) The modulated signal; (b) The modulating signal; (c) The demodulated signal.

Due to the limitations of the ADC, relatively slow data rates of the signals were obtained, of same order as in the modulator part. In order to obtain higher frequencies, the slow ADC must be replaced with external devices for a complete wireless/underwater system.

Fig. 4.47 illustrates the design summary of the demodulator board, while fig. 4.48 shows the schematic representation of the board.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	169	9,312	1%	
Number of 4 input LUTs	199	9,312	2%	
Number of occupied Slices	159	4,656	3%	
Total Number of 4 input LUTs	214	9,312	2%	
Number of bonded IOBs	13	232	5%	
Number of BUFGMUXs	3	24	12%	
Number of MULT18X18SIOs	1	20	5%	
Average Fanout of Non-Clock Nets	2.88			

Fig. 4.47 Design Summary of the BPSK Demodulator

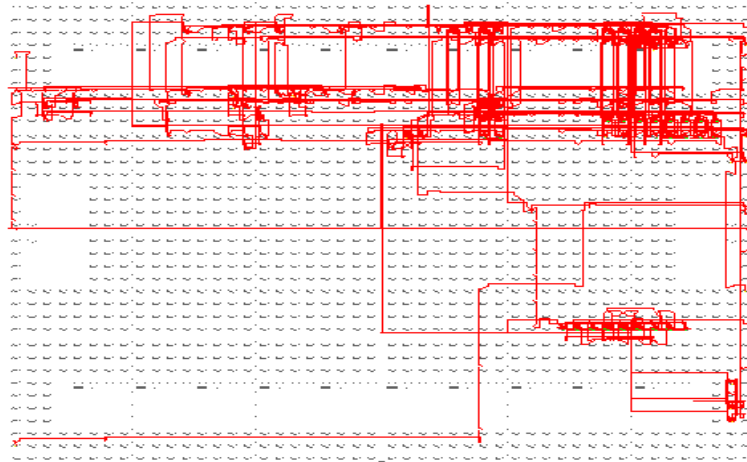


Fig. 4.48 The resouces map of the BPSK demodulator on the Spartan 3E Board

#### 4.4. Conclusions and Contributions

Chapter 4 describes the actual process of the BPSK modulation technique, as well as the implementation of the hardware itself. The Simulink and System Generator, included in the Matlab software, were used for the theoretical study and simulation, and for implementation the Nexys2 and Spartan 3E development boards and the Xilinx ISE software tool.

In the first part of the chapter, a BPSK Detector, all digital, was implemented, both the modulator and demodulator on the same FPGA board, and successfully designed with VHDL programming code. The BPSK detector had as a starting point paper [4] and an improvement of the detector was made. The newness of the detector was made by modifying the LFSR which had improved the randomness sequence of the signal. Analysis and comparison were made on a Nexys2 board. The Spartan-3E FPGA from the Nexys2 board has dedicated 18x18 bits hardware multiplier and so, there was no need for a relatively slow Booth multiplier.

In the second part of the chapter, a BPSK digital modulator was simulated, tested and implemented. In terms of the simulation process, two implementations of the BPSK Modulator in the Matlab/Simulink environment have been proposed, the first with simple blocks and the second, with a block in which we wrote Matlab code. In the testing part, a proposal of four implementations of a BPSK modulator in System Generator was conducted. In the first, the three signals: the carrier and the modulating signal were generated external, and the modulated signal was obtained internal. In the second scheme, the carrier is generated external, and the modulating signal is generated internal by a LFSR. In the third scheme, all three signals were generated internal with the exception of the modulating signal which can be obtained either internal by the LFSR, or external by a pulse generator and in the fourth implementation, all signals were obtained internal. Two implementation of the BPSK modulator on the Spartan 3E were designed, in the implementation part. The first one was based on the third proposal of the modulator made in System Generator and the signals were routed to a VGA monitor. If "1" was transmitted, the modulated signal remained same as the carrier, but if "0" was transmitted, the

modulated signal was yielded with a  $180^\circ$  phase. The second implementation had as a model the fourth proposal of the modulator in System Generator and the signals were routed to an oscilloscope. Comparing the designs summary obtained with the summary of the BPSK detector, the logic utilization of the board was lower in terms of the slice flip-flops and LUTs used. All of these make the design suitable in terms of propagation, implementation and logic utilization of the Spartan 3E boards used in this work.

A comparison in terms of the logic utilization between the implementation of the BPSK modulator via VGA and the implementation via oscilloscope is illustrated in fig. 4.49 and table 4.2 represents a comparison between previous works in the field of BPSK modulation on FPGA and this work.

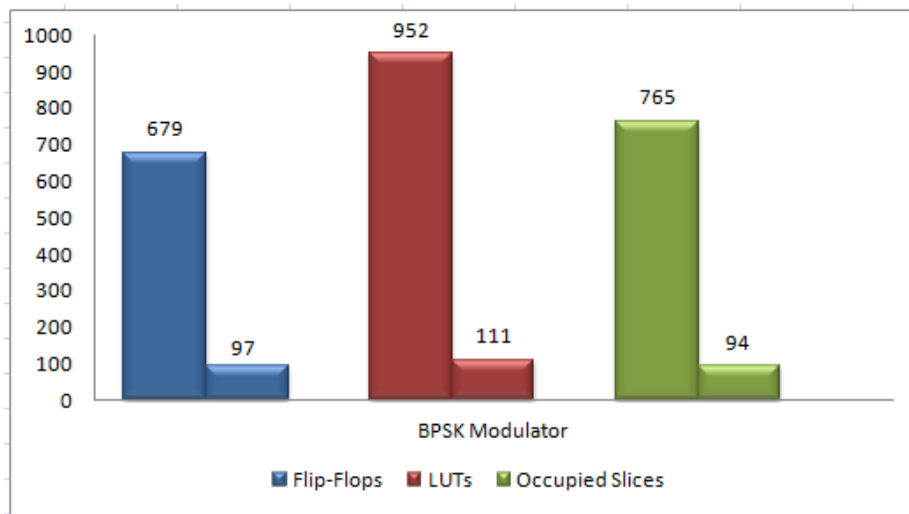


Fig. 4.49 Comparison of the logic resources

**Table 4.2 Comparison of previous work and practice in BPSK modulation on FPGA**

Modulation scheme	Reference	Board	Resource Utilization		
			Flip-Flops	LUTs	Slices
<b>B P S K</b>	[27]	Virtex-4	1%	1%	1%
	[30]	Virtex-4	6%	27%	25%
	[97]	Spartan 3E	13%	13%	16%
	[143]	Virtex-II	1%	9%	10%
	[160]	Virtex-4	11%	11%	15%
	<b>[122]</b>	<b>Spartan 3E</b>	<b>7%</b>	<b>10%</b>	<b>16%</b>
	<b>[123]</b>	<b>Spartan 3E (modulator)</b>	<b>1%</b>	<b>1%</b>	<b>2%</b>
	<b>[128]</b>	<b>Spartan 3E (demodulator)</b>	<b>1%</b>	<b>2%</b>	<b>3%</b>

In the third part of the chapter, a BPSK System, made of a modulator and demodulator, was the subject of simulation, testing and implementation. The simulation of the BPSK System was made in the Matlab/Simulink environment. The testing part of the system was made in System Generator. The modulating signal

and the carrier were generated internal, the modulating signal by a LFSR and the carrier by a DDS Compiler. The modulated signal was obtained at the output of a mux block and, then, passed through a communication channel where noise is added. In the demodulator, the carrier was recovered due to another DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal was then added with all the multiplied samples from the carrier in a period. The operation took place in the accumulator. Once the result had been obtained, it was compared with a decision threshold. If the compared signal was positive, the demodulator took the decision that '1' was transmitted, otherwise, '0'. The BPSK System implemented on the Spartan 3E board had the same principle as the implementation in System Generator. Although System Generator has an option to generate the VHDL code, for this design the code was made from the beginning because the generated code was hard to read. The only difference was the carrier which was indeed generated internal, in a ROM memory, but made of 16 different values. The yielded carrier with  $180^\circ$  phase shift was obtained by reading the ROM memory later with 8 samples. Comparing the design summary obtained with other works in this field [12], [27], [37], [49], [84], [155], [160], the logic utilization of the board was lower in terms of the slice flip-flops and LUTs used. All of these make the design suitable in terms of propagation, implementation and logic utilization of the Spartan 3E boards used in this work.

### **Contributions:**

- **An improvement of a BPSK detector was made: the LFSR inside the detector was modified which improved the randomness sequence of the signal. Also, it was no need for a relatively slow Booth multiplier, since the board had a dedicated 18x18 bits hardware multiplier.**
- Two implementations of a BPSK Modulator in the Matlab/Simulink environment had been proposed, first with simple Simulink blocks and the second with an embedded Matlab function.
- Four implementations of a BPSK modulator in System Generator were also proposed:
  - the carrier and the modulating were generated external and the modulated signal was obtained internal;
  - the carrier is generated external and the modulating signal is generated internal by a LFSR;
  - all three signals were generated internal with the exception of the modulating signal which can be obtained either internal by the LFSR, or external by a generator;
  - all signals were generated internal.
- **Two new implementation of the BPSK modulator were made:**
  - **The first method was a visual one, in which two different colored rectangular were displayed on a VGA monitor;**
  - **In the second implementation, the signals were routed to an oscilloscope in order to be measured.**
- **A comparison in terms of the logic utilization between the implementation of the BPSK modulator via VGA and the implementation via oscilloscope was made and presented in fig.4.49.**

- In order to test the functionality principle of the BPSK System, simulations were made in the Matlab/Simulink environment.
- Simulations were made in System Generator to test the hardware part of the BPSK System: the modulating signal was generated internal by a LFSR and the carrier, also internal, by a DDS Compiler. The modulated signal obtained was passed through a communication channel where noise was added. In the demodulator, the carrier was recovered due to another DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal was then added with all the multiplied samples from the carrier in a period. The operation took place in the accumulator. Once the result had been obtained, it was compared with a decision threshold. If the compared signal was positive, the demodulator took the decision that '1' was transmitted, otherwise, '0'.
- **A new architecture proposal of a digital communication system based on the BPSK modulation technique had been introduced and optimized as a pair.**
- **An optimization chart emphasizing the quality measures of the resources map was conducted for all FPGA designs.**
- **A comparison between previous works in the field of BPSK modulation on FPGA and this work in terms of the logic utilization is illustrated in table 4.2.**

## 5. ARGUMENTED QPSK MODULATION FROM FPGA PRESPECTIVE

To simulate, test and implement the actual process of the QPSK modulation technique is the subject of this chapter. Matlab/Simulink and System Generator are used for the theoretical study and two FPGA boards and Xilinx ISE software tool for the hardware implementation of the QPSK modulation technique.

The chapter is divided in two main parts:

- The simulation, testing and implementation of a QPSK modulator on a single FPGA;

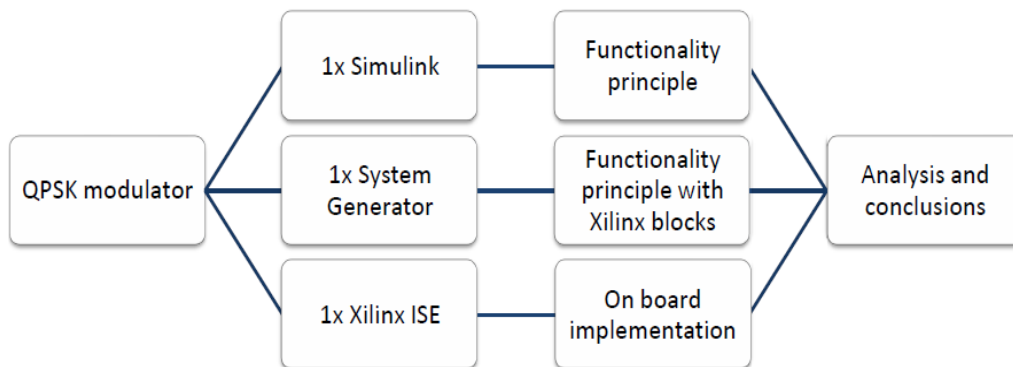


Fig. 5.1 First step in implementing the QPSK modulation technique

- The second step in simulating, testing and implementation of a QPSK digital system on two FPGAs.

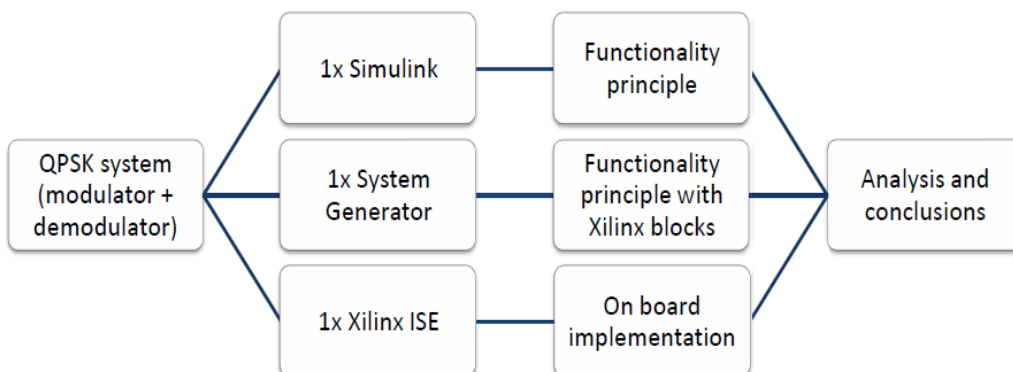


Fig. 5.2 Second step of implementing the QPSK modulation technique



## 5.1. QPSK Detector

The main purpose was to implement a digital communication system based on the QPSK modulation technique on two identical boards, the first board comporting as a modulator and the second one, as a demodulator.

For implementing a QPSK system [125], first the modulator [124] was simulated, tested and implemented on a Spartan 3E board. To test the modulator functionality, Matlab/Simulink environment was used. Afterwards, the design was tested using FPGA blocks in System Generator [177], [179]. The final step was to implement the QPSK modulator on the Spartan 3E board using Xilinx ISE 12.3 software tool.

### 5.1.1. QPSK Modulator in Simulink

Fig. 5.3 shows an implementation of a QPSK modulator in the Simulink environment and fig. 5.4 the waveform generated by the corresponding blocks. By convention, the figures from top to bottom are numbered starting with a.

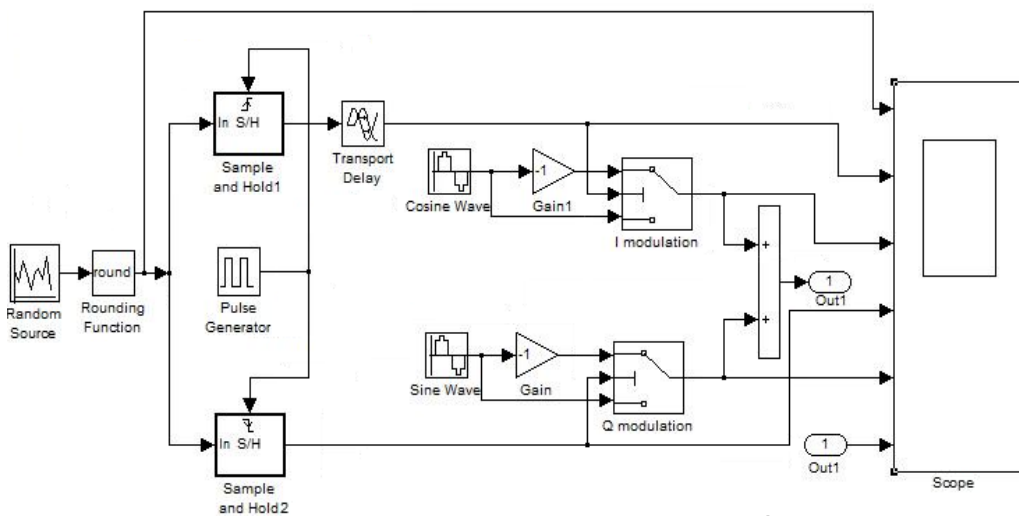


Fig. 5.3 QPSK Modulator in the Simulink environment

The Simulink block set contains:

- the random source block (fig. 5.3);
- the rounding function blocks (fig. 5.3) which generate the binary sequence or the modulating signal (fig.5.4a);
- the sample-and-hold block which separates the binary sequence into an odd-bit-sequence I (from the sample-and-hold1 block) and an even-bit-sequence (from the sample-and-hold2 block), illustrated in fig. 5.3. The Sample and Hold block acquires the input when it receives a trigger event at the trigger port (rising, falling or either edge). The block then holds the output at the acquired input value until the next triggering event occurs.

The odd-bits are acquired at the rising edge triggers: when the trigger input rises from a negative value or zero to a positive value and the even-bits are acquired at the falling edge triggers: when the trigger input falls from a positive value or zero to a negative value [79]. Because the I sequence (fig. 5.4b) is read first, it has been delayed with the Transport Delay block (fig. 5.3) in order to be synchronized with the Q sequence (fig. 5.4d);

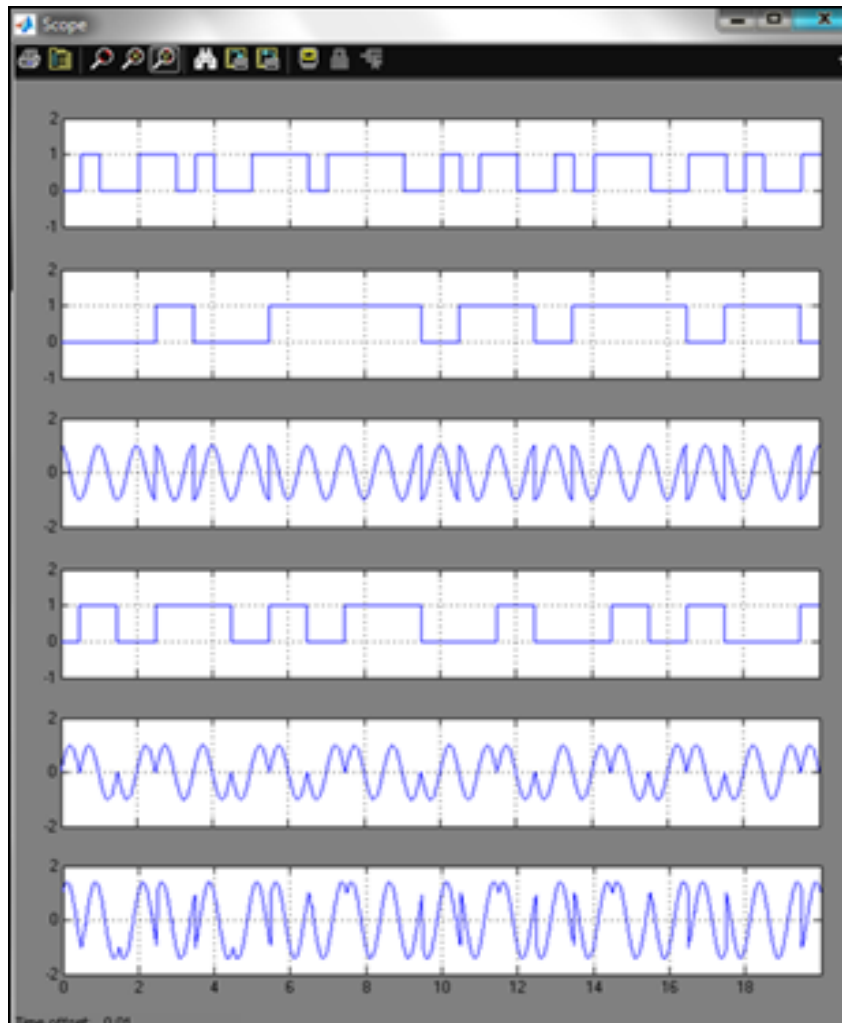


Fig. 5.4 The waveforms on the scope: (a) The modulating signal; (b)  $I(t)$ ; (c)  $I(t) \cos 2\pi f_c t$ ; (d)  $Q(t)$ ; (e)  $Q(t) \sin 2\pi f_c t$ ; (f) The QPSK modulated signal

- the cosine wave block (fig. 5.3) which generates a cosine waveform for the I-channel;
- the gain block (fig. 5.3), which determines a cosine waveform with 180° phase difference also for the I-channel;
- the sine wave block (fig. 5.3) which generates a sinus waveform for the Q-channel;

- the gain block, a sinus with phase difference of  $180^\circ$  also for the Q-channel;
- the switch blocks (fig. 5.3) which will choose between the first or third output depending on the value of the second input. In the case of the I-channel, if the second input is "1", the output value will be cosine, but if the second input is "0", the output will be  $-\cos$  (fig. 5.4c); and for the Q-channel, if the second input is "1", the output value will be sine, but if the second input is "0", the output will be  $-\sin$  (fig. 5.4e);
- the sum block (fig. 5.1) which combines the two modulated signals into the QPSK modulated signal (fig. 5.4f).

### 5.1.2. QPSK Modulator in System Generator

Fig. 5.5 illustrates the implementation of a QPSK Modulator using System Generator tools in Simulink.

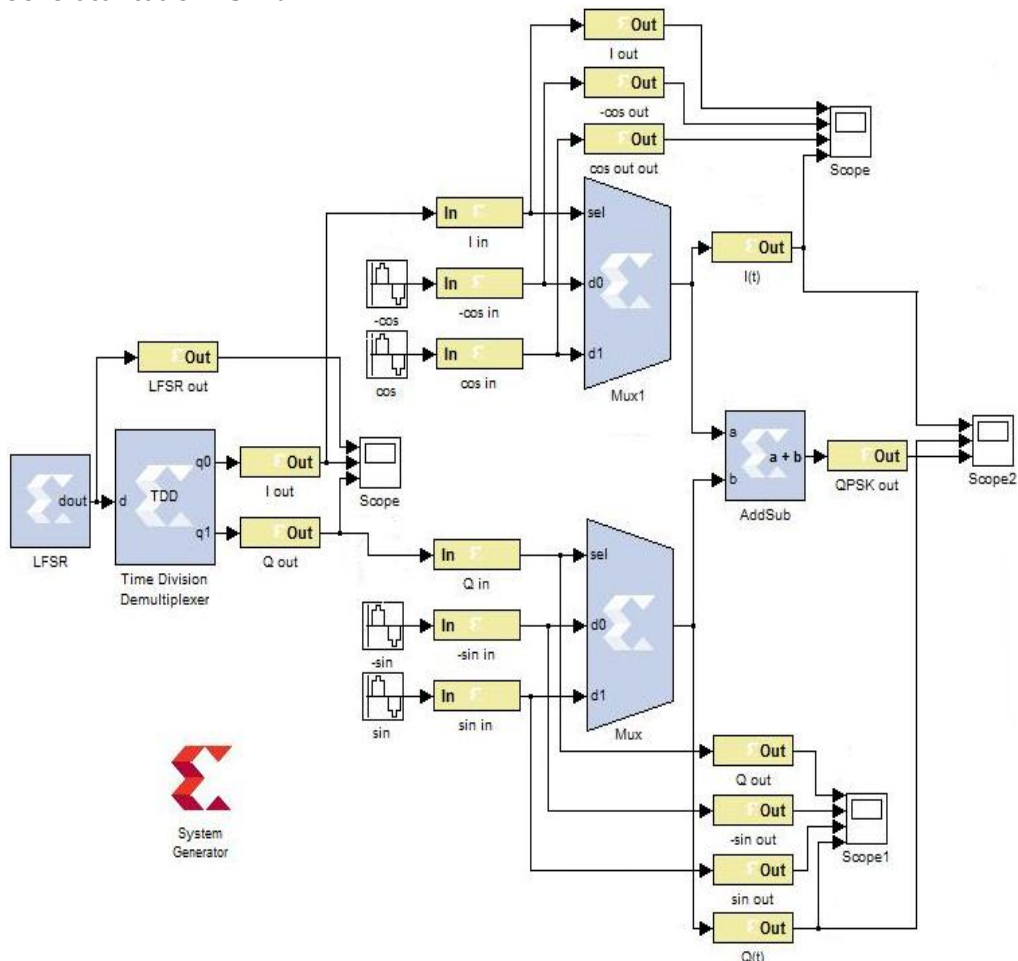


Fig. 5.5 QPSK Modulator in System Generator

The Simulink Blockset contains:

- the sine wave blocks;
- the cosine wave blocks;
- the scope block.

The System Generator Blockset contains:

- the gateway in blocks: the inputs into the Xilinx area of the Simulink design;
- the gateway out blocks: the outputs from the Xilinx area of the Simulink design;
- the LFSR block: implements a Linear Feedback Shift Register and it was used to create the binary sequence or modulating signal internal;
- the Time Division Demultiplexer block: accepts input serially and presents it to multiple outputs to a slower rate. In another word, this block divides the modulating signal into the odd-sequence I and even-sequence Q, illustrated in fig. 5.6.

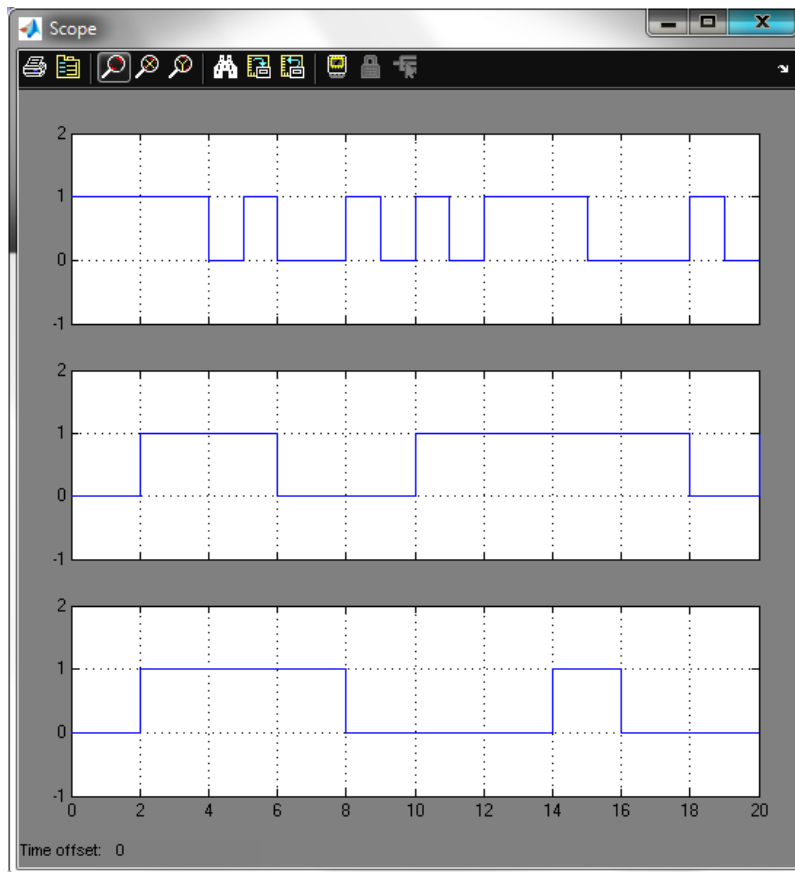


Fig. 5.6 The waveforms: (a) the modulating signal; (b)  $I(t)$ ; (c)  $Q(t)$ .

- the Mux block: implements a multiplexer. It has one select input and a configurable number of data inputs that can be defined by the user. The d0 and d1 inputs of the mux represent cosine and  $-\text{cosine}$  for the I sequence and sine and  $-\text{sine}$  for the Q sequence. The sel input selects between d0 and

d1 depending on the odd-sequence for mux1 and the even-sequence for mux;

- the AddSub: implements an adder.

In order to see any signal on the scope, the signal must be passed through a gateway out.

Fig 5.7 shows the inphase and quadrature channels; the inphase channel was modulated with a cosine waveform, while the quadrature channel was modulated with a sine waveform, as well as the modulated QPSK signal.

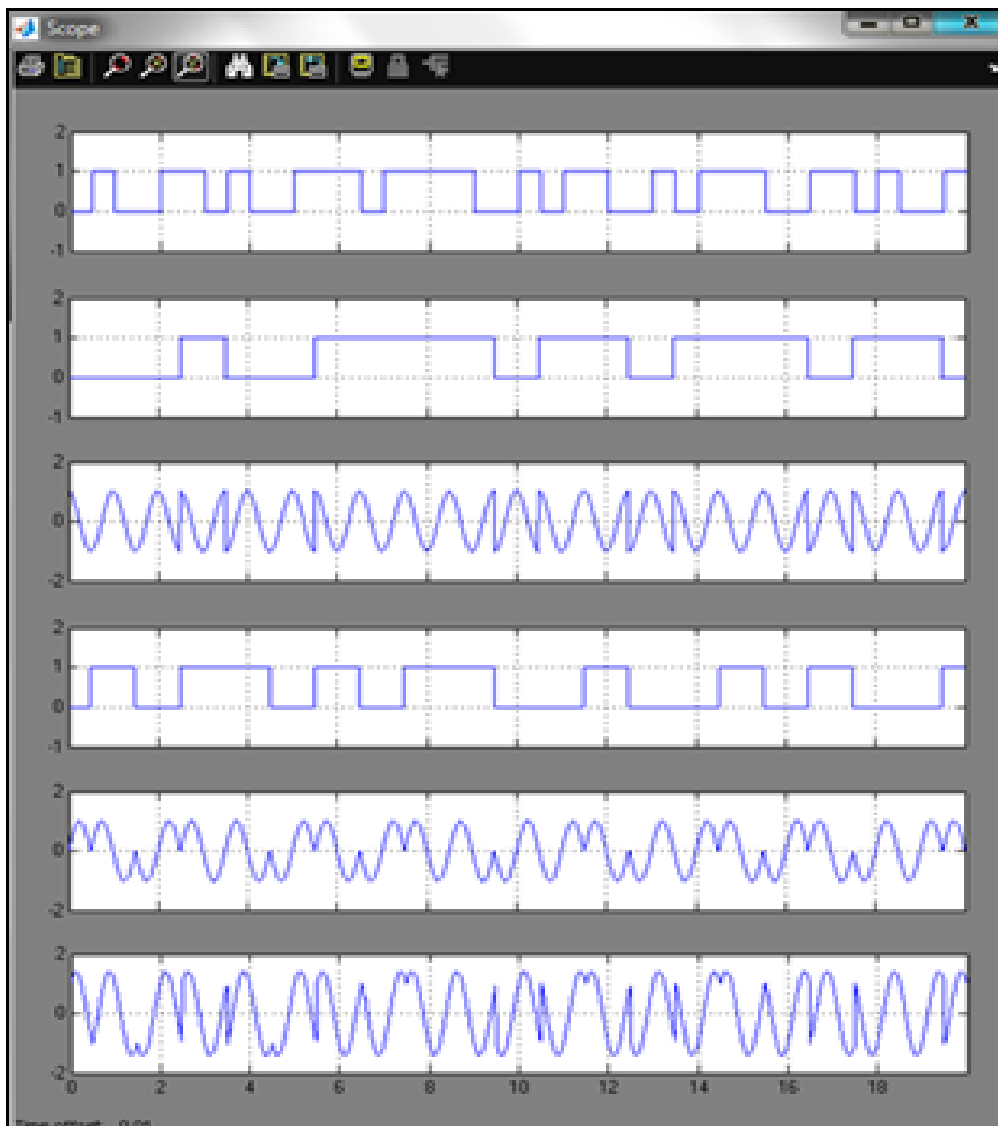


Fig. 5.7 The waveforms in the modulator

- (a) The modulating signal; (b)  $I(t)$ ; (c)  $I(t) \cos 2\pi f_c t$ ;  
 (d)  $Q(t)$ ; (e)  $Q(t) \sin 2\pi f_c t$ ; (f) The modulated signal

### 5.1.3. QPSK Modulator on FPGA

The QPSK Modulator implemented on the Spartan 3E board has, as a model, the implementation in System Generator. The only difference is that the sine and cosine signals are generated internal, in a ROM. The experimental setup consists of a computer, a Spartan 3E board and an oscilloscope. The ISE Web Pack runs on the computer and it programs the Spartan 3E board.

The principle of the QPSK modulator implemented on the FPGA is illustrated in fig. 5.8 [124].

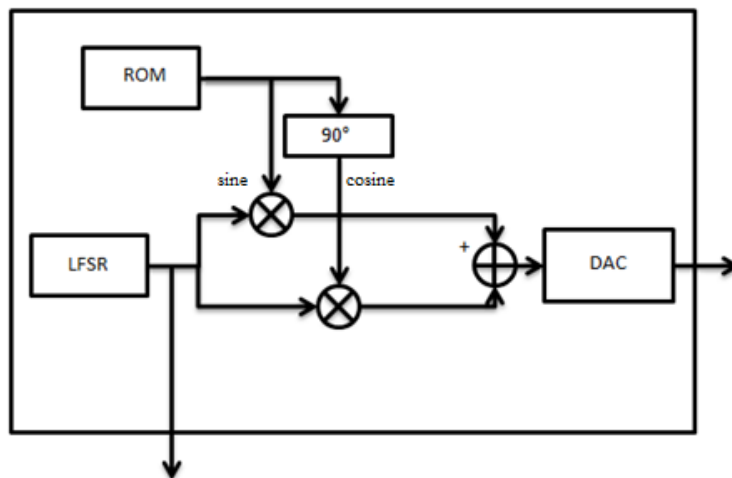


Fig. 5.8 The principle of the QPSK modulator on the FPGA

The binary sequence or the modulating signal is generated internal by a LFSR and is divided into two sequences: the odd-bit-sequence for the I-channel and the even-bit-sequence for the Q-channel. The sine signal is also generated inside the board and is made of 16 different values kept in a ROM memory. Knowing that the phase differences between the cosine and sine signal is  $90^\circ$ , we obtain the cosine from the sine reading it later with 4 samples. The I-channel is modulated with the cosine waveform and the Q-channel with the sine waveform which is a BPSK modulation. Adding the two modulated signals, the QPSK modulated signal is obtained. In order to transmit the QPSK modulated signal, it is routed to the DAC (Digital-to-Analog Converter) on the board.

In the validation and data collecting part of the implementation, the signals were routed to a LeCroy WaveSurfer Xs Series oscilloscope. Fig. 5.9, fig. 5.10 and fig. 5.11 show different scenarios of the modulating signal, the I-channel, the Q-channel and the modulated signal.

Due to the limitations of the serial DAC on the board, relatively slow data rates of the carrier and modulated signals were obtained. The frequency of the QPSK carrier is 31,250 kHz and the output QPSK frequency is 62,50 kbps. In order to obtain higher frequencies, the slow DAC must be replaced with external devices for a complete wireless/underwater system.

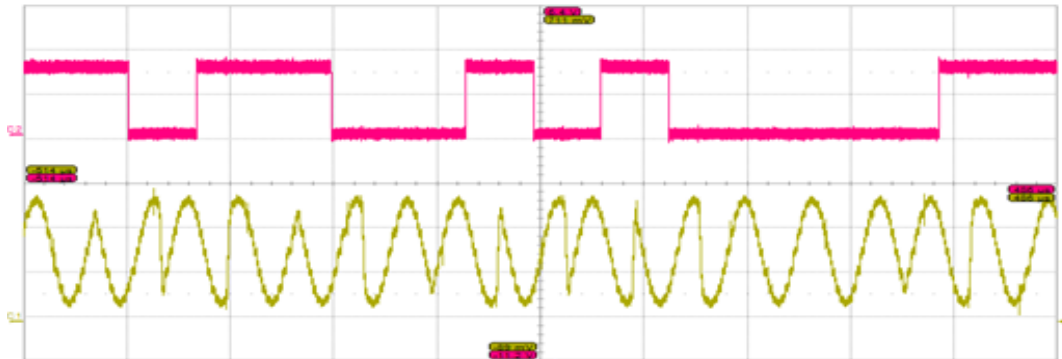


Fig. 5.9 The waveforms: (a) the modulating signal (LFSR) ; (b) the modulated signal

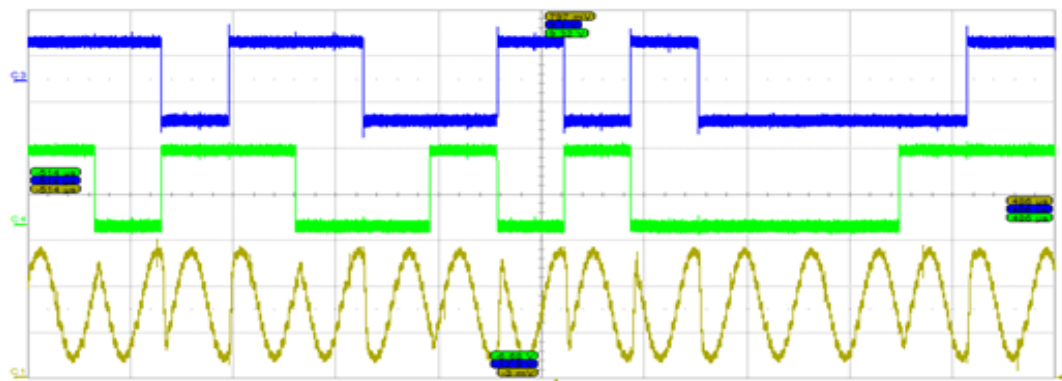


Fig. 5.10 The waveforms: (a) I-channel; (b) Q-channel; (c) the modulated signal

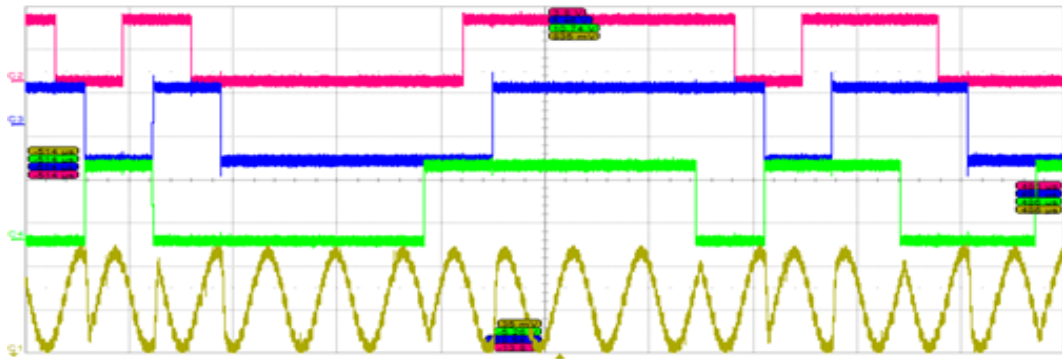


Fig. 5.11 The waveforms: (a) the modulating signal (LFSR); (b) I-channel; (c) Q-channel; (d) the modulated signal.

Fig. 5.12 represents the design summary which represents the utilization of flip-flops, LUTs, slices used from the capabilities of the FPGA from the Spartan 3E board and fig. 5.13 illustrates the resources map of the FPGA from the same board.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	95	9,312	1%	
Number of 4 input LUTs	157	9,312	1%	
Number of occupied Slices	113	4,656	2%	
Total Number of 4 input LUTs	164	9,312	1%	
Number of bonded IOBs	14	232	6%	
Number of BUFGMUXs	2	24	8%	
Average Fanout of Non-Clock Nets	3.29			

Fig. 5.12 Design summary of the QPSK modulator

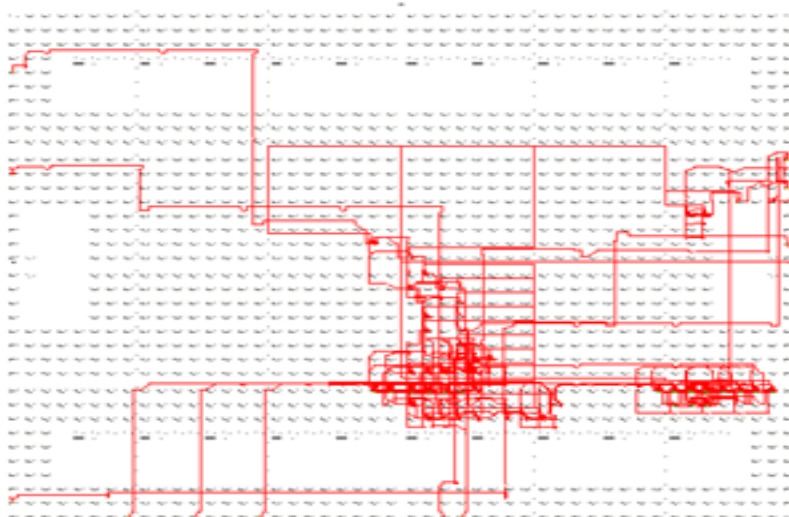


Fig. 5.13 The resources map of the QPSK modulator on the Spartan 3E Board

## 5.2. QPSK System

The second and final step in implementing a digital communication system based on the QPSK modulation technique was to simulate and test the demodulator part. The demodulator functionality was simulated using Matlab/Simulink environment. Afterwards, the design was tested using FPGA blocks in System Generator. The final step was to implement the QPSK system [125], the modulator and demodulator parts, on two Spartan 3E boards using Xilinx ISE 12.3 software tool.

### 5.2.1. QPSK System in Simulink

Fig. 5.14 represents a communication system implemented in the Matlab/Simulink environment that uses the QPSK modulation technique. The system [128]



is composed of the binary data source, a modulator, a channel and a demodulator. The previous chapter made the subject of the binary data source and the modulator part.

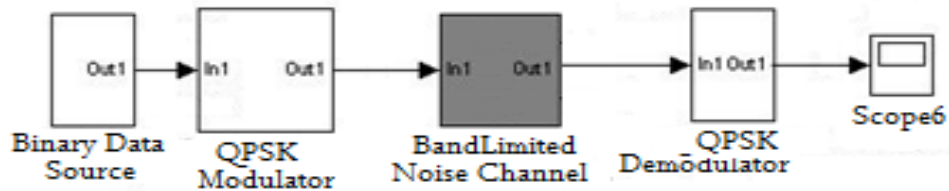


Fig. 5.14 QPSK System

The modulated signal from fig. 5.4f is then pass through the same channel from fig. 4.34 where noise is added. The channel also has a limited frequency bandwidth so that it can be viewed as a filter. The corresponding signals are shown in fig.5.15.

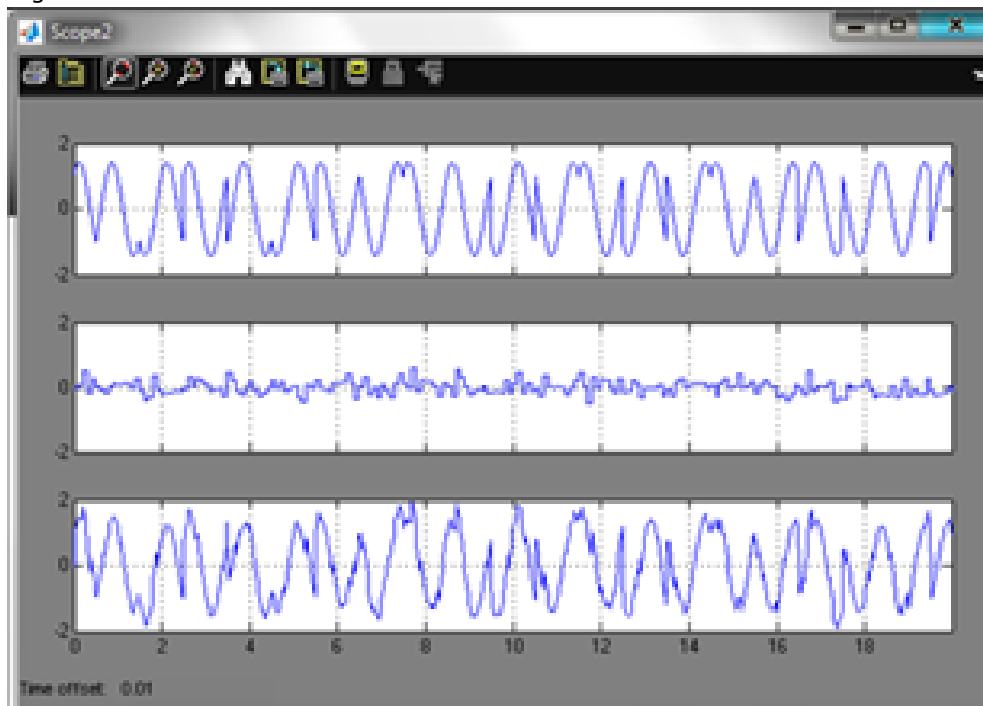


Fig. 5.15 The waveforms shown on the scope: (a) the modulated signal; (b) the noise; (c) the noise added to the modulated signal

The modulated signal added with noise arrives at the demodulator (fig. 5.16) and is divided into two channels. The first channel is modulated with a cosine waveform and the second with a sine as well as in theory. At the outputs, the I and Q channels are found and put together to obtain the initial binary sequence (fig. 5.17).

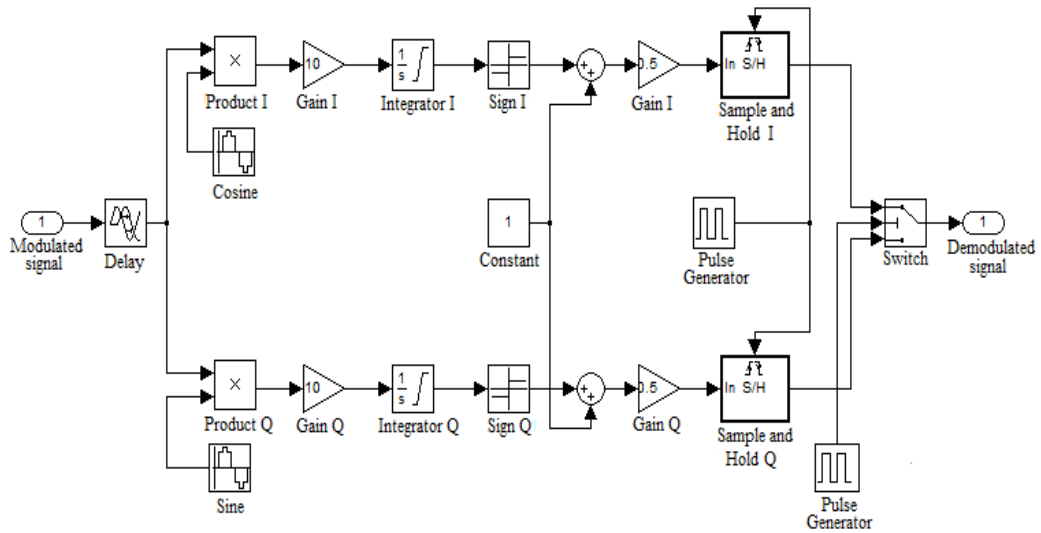


Fig. 5.16 QPSK demodulator in Simulink

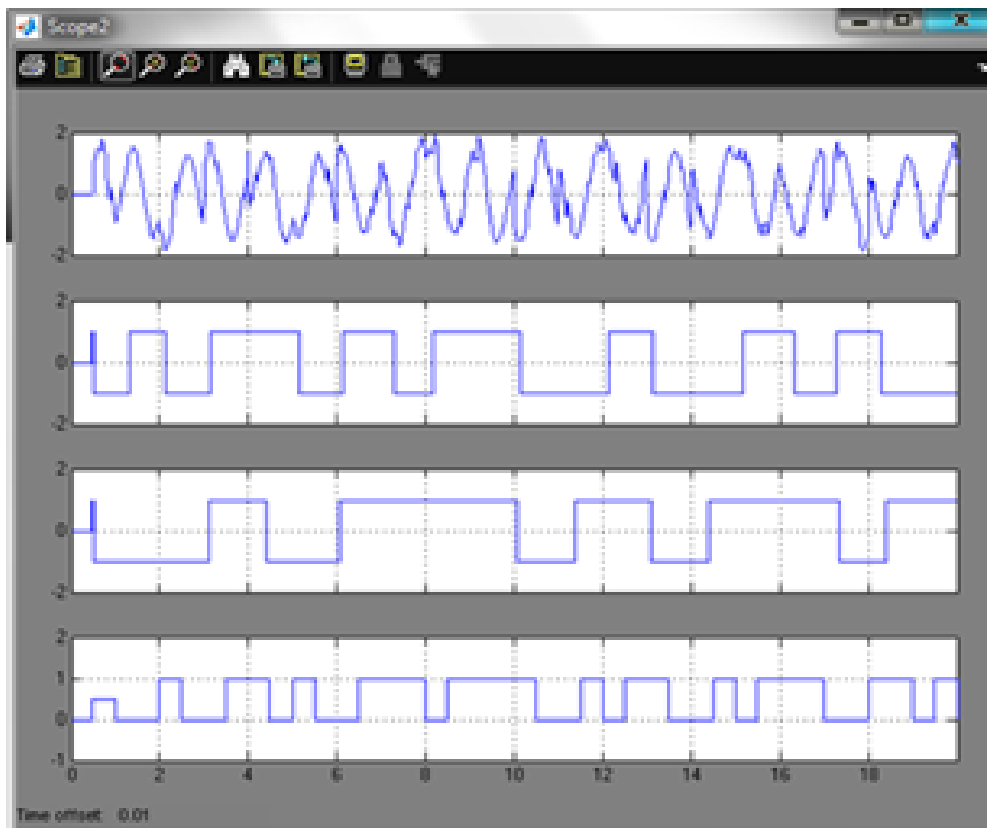


Fig. 5.17 The waveforms in the demodulator:  
 (a) The modulated signal with noise; (b) The I-channel; (c) The Q-channel; (d) The demodulated signal

### 5.2.2. QPSK System in System Generator

The modulator part is the same as the one in fig. 5.5. The modulating signal (fig.5.7f) is then pass through the same channel (fig. 4.34) where noise is added and arrives at the input of the demodulator (fig. 5.18).

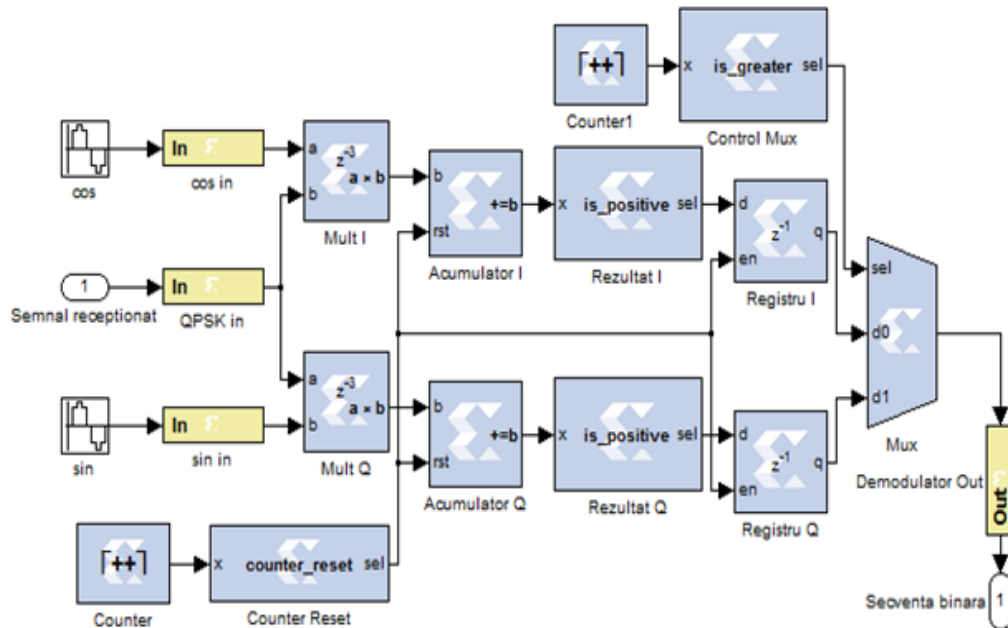


Fig. 5.18 QPSK Demodulator in System Generator

In the demodulator, the modulated signal with noise is multiplied once with sine, and the second time with cosine. The sine and cosine waveforms are recovered in Simulink with the sine and cosine blocks.

The two signals obtained are kept in two accumulators and then compared with a threshold. If the compared signal is positive, the demodulator takes the decision that '1' was transmitted, otherwise, '0', obtaining the demodulated I and Q channels (fig. 5.19b and fig. 5.19c). The multiplexer is the one who puts the I and Q channels together obtaining the demodulated signal which is the initial modulating signal (fig. 5.20).

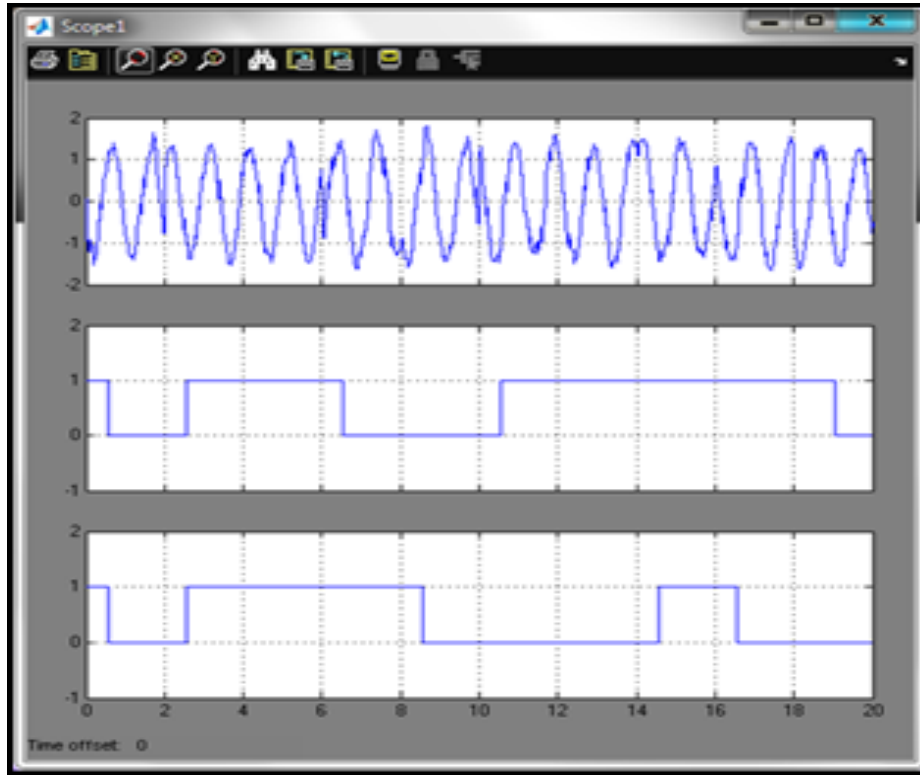


Fig. 5.19 The waveforms: (a) the modulated signal with noise; (b) the demodulated I-channel; (c) the demodulated Q-channel

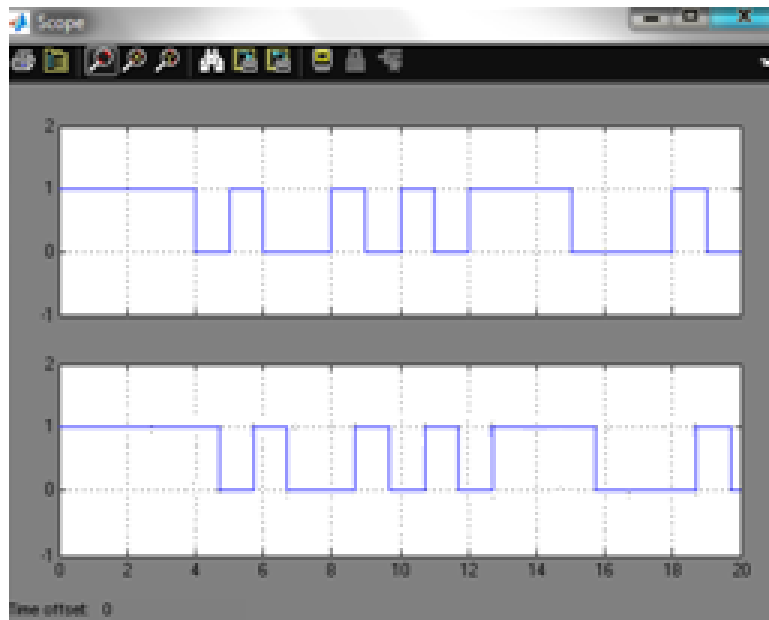


Fig. 5.20 The waveforms: (a) the modulating signal; (b) the demodulated signal.

### 5.2.3. QPSK System on FPGA

The QPSK System implemented on the Spartan 3E board has as a model the implementation in System Generator with the only difference that the sine and cosine signals are generated internal, in a ROM.

The digital communication system based on the QPSK modulation technique consists of two Spartan 3E boards, the first board behaves as a modulator and the second one, as a demodulator.

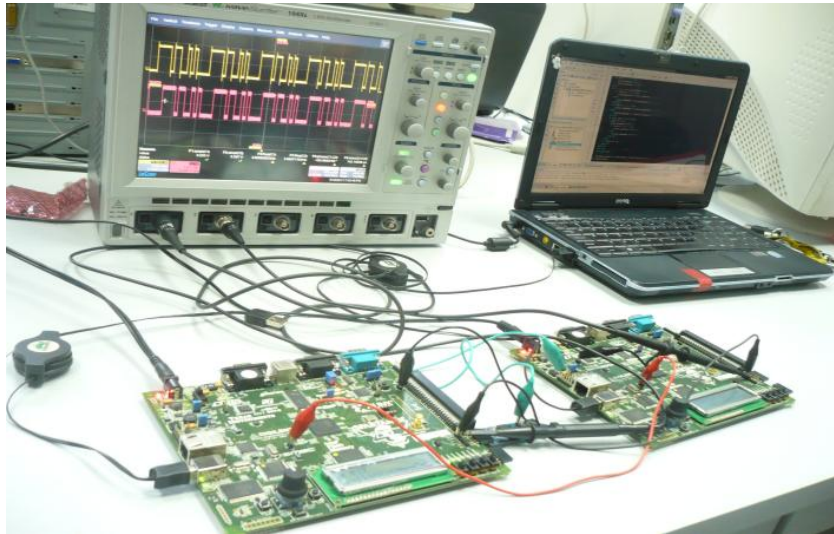


Fig. 5.21 QPSK System – experimental setup

The connections between the two boards are made of three wires: the green wire comports as a communication channel, the red wire represents the synchronization between the two boards and the black one comports as an asynchronous reset signal.

The principle of the QPSK modulator as the one described in topic 5.1.3: the modulating signal is generated internal, in the modulator, by a LFSR and is divided into two sequences: the-odd-sequence or the I-channel and the-even-sequence or the Q-channel. The I-channel is modulated with cosine and the Q-channel with sine. The sine and cosine are also generated internal, and are made of 16 different values kept in a ROM memory [4]. The cosine is obtained from sine by reading the values later with 4 samples [125]. By adding the two modulated signals, the QPSK modulated signal is obtained. The modulated signal is then sent to the DAC (Digital-to-Analog Converter) on the board in order to be sent through the channel.

The principle of the QPSK demodulator implemented on the FPGA is shown in fig. 5.22.

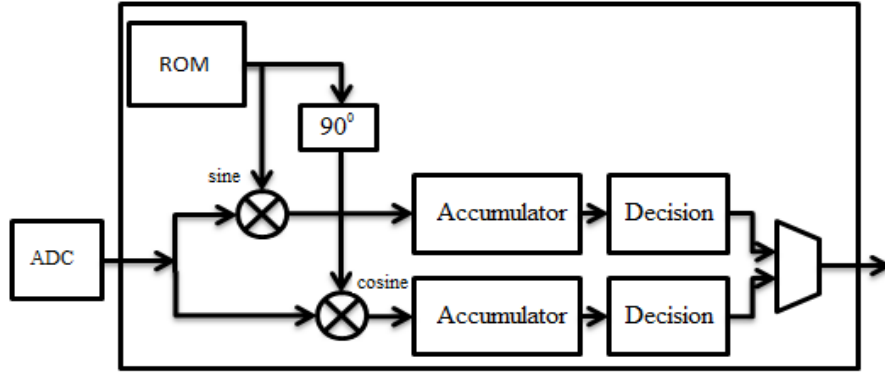


Fig. 5.22 The principle of the QPSK demodulator on the FPGA

The modulated signal affected with noise arrives at the second board which behaves as a demodulator. The signal is converted into a digital form with the help of a pmod AD1. This digital signal is then multiply with both signals, the sine and cosine generated internal in a ROM memory. The results are kept in an accumulator and compared with a decision threshold and so, the demodulated I-channel and Q-channel are obtained and in the end the demodulated signal.

After implementing the QPSK System made of a modulator and demodulator on the two Spartan 3E boards, the signals were routed to an oscilloscope. Fig. 5.23 illustrates two signals: the modulating signal from the LFSR and the demodulated QPSK signal.



Fig. 5.23 The waveforms: (a) The modulating signal; (b) The demodulated signal

Fig. 5.24 illustrates the design summary of the demodulator board and fig. 5.26 shows the post and route implementation of the QPSK demodulator the Spartan 3E board.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	233	9,312	2%	
Number of 4 input LUTs	277	9,312	2%	
Number of occupied Slices	221	4,656	4%	
Total Number of 4 input LUTs	304	9,312	3%	
Number of bonded IOBs	15	232	6%	
Number of BUFGMUXs	3	24	12%	
Number of MULT18X18SIOs	2	20	10%	
Average Fanout of Non-Clock Nets	2.62			

Fig. 5.24 Design Summary of the QPSK Demodulator

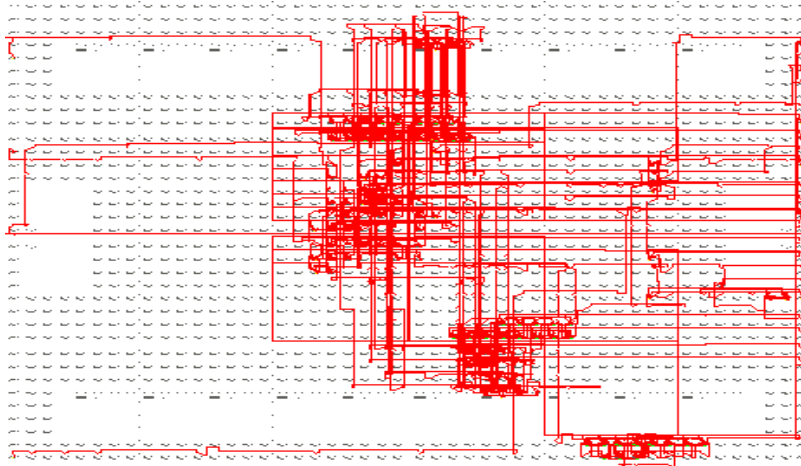


Fig. 5.25 The resources map of the QPSK demodulator

### 5.3. Conclusions and Contributions

Chapter 5 describes the actual process of the QPSK modulation technique, as well as the implementation of the hardware itself. The Simulink and System Generator, included in the Matlab software, were used for the theoretical study and simulation, and for implementation the Nexys2 and Spartan 3E development boards and the Xilinx ISE software tool.

In the first part of the chapter, an implementation of the QPSK Modulator in the Matlab/Simulink environment as well as a proposal of a QPSK modulator in System Generator was presented. The modulating signal is generated internal by a LFSR and is divided into two sequences: the odd-sequence for the I-channel and the even-sequence for the Q-channel. The sine and cosine waveform are generated outside the FPGA. The odd-sequence is modulated with the cosine waveform and the even-sequence with the sine waveform which is a BPSK modulation. The QPSK modulated signal is obtained by adding the two modulated signals. The same principle is applied in implementing the modulator on the board, with the difference that the sine and cosine signals were generated inside the board. The sine signal was made of 16 different values kept in a ROM memory. The phase differences

between the cosine and sine signal is  $90^\circ$ , and so the cosine was obtained from the sine by reading it later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK modulated signal was obtained. The design has been written in the VHDL programming code by Xilinx software.

In the second part of the chapter, an implementation of a QPSK System (Modulator and Demodulator) in the Matlab/Simulink environment as well as a proposal of a QPSK System in System Generator is studied. The binary sequence is generated internal by a LFSR and is divided into two sequences: the odd-sequence (I-channel) and the even-sequence (Q-channel). The odd-sequence is modulated with the cosine waveform and the even-sequence with the sine waveform, waveforms obtained outside the board. The QPSK modulated signal is obtained by adding the two modulated signals. The QPSK modulated signal is then passed through the same channel used in Simulink where noise is added. In the demodulator, the modulated signal with noise is multiplied twice, once with sine, and the second time with cosine. The two signals are kept in two accumulators and then compared with a threshold. If the compared signal is greater than the threshold, the demodulator takes the decision that '1' was transmitted, otherwise, '0', obtaining the demodulated I and Q channels. The multiplexer is the one who puts the I and Q channels together obtaining the demodulated signal which is the initial modulating signal. The same principle is applied in implementing the modulator and the demodulator on the two boards, with the difference that the sine and cosine signals were generated inside the board. The sine signal was made of 16 different values kept in a ROM memory and the cosine was obtained reading the ROM memory later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK signal was obtained. The modulated signal is then sent to the DAC on the board in order to be sent through the channel. The modulated signal affected with noise arrives at the demodulator. The signal is converted into a digital form with the help of an AD board. This digital signal is then multiply with both signals, the sine and cosine generated internal in the ROM memory from the second board. The results are kept in an accumulator and compared with a decision threshold and so, the demodulated I-channel and Q-channel are obtained and in the end the demodulated signal.

Fig. 5.26 summarizes the resources of the two boards used in implementing the QPSK modulator and demodulator.

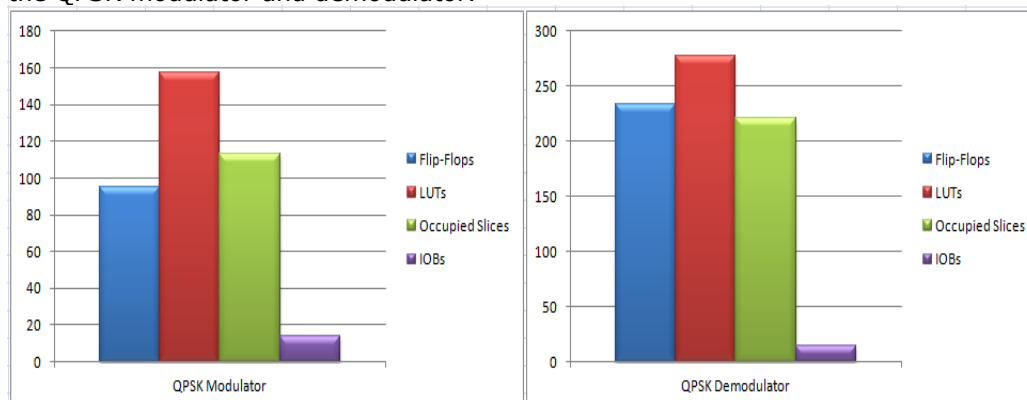


Fig. 5.26 Comparison of the logic resources of the QPSK modulator and demodulator



Table 5.1 illustrates a comparison between previous works in the field of QPSK modulation on FPGA and this work.

**Table 5.1 Main Comparison of previous work and practice in QPSK modulation on FPGA**

Modulation scheme	Reference	Board	Resource Utilization		
			Flip-Flops	LUTs	Slices
<b>Q P S K</b>	[30]	Virtex-4	1%	16%	17%
	[60]	Virtex-II	1%	2%	2%
	[135]	Virtex-II	1%	13%	13%
	<b>[124]</b>	<b>Spartan 3E (modulator)</b>	<b>1%</b>	<b>1%</b>	<b>2%</b>
	<b>[125]</b>	<b>Spartan 3E (demodulator)</b>	<b>2%</b>	<b>2%</b>	<b>2%</b>

#### Contributions:

- The design of a QPSK Modulator in the Matlab/Simulink environment had been proposed.
- A QPSK modulator in System Generator, based on the Simulink design, was also proposed: the modulating signals was generated internal by a LFSR and divided in an odd and an even sequence. The odd sequence was modulated with sine and the even one with cosine. The sine and cosine signals were generated inside the board. The sine signal was made of 16 different values kept in a ROM memory and the cosine was obtained from the sine by reading it later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK modulated signal was obtained.
- The QPSK modulator was implemented on FPGA with the signals routed to an oscilloscope in order to be measured.
- Simulations of the QPSK System were made in the Matlab/Simulink environment for verifying the functionality principle;
- The hardware part of the QPSK system was tested in System Generator: the modulated signal affected with noise arrives at the demodulator. The signal is converted into a digital form with the help of an AD board. This digital signal is then multiply with both signals, the sine and cosine generated internal in the ROM memory from the second board. The results are kept in an accumulator and compared with a decision threshold and so, the demodulated I-channel and Q-channel are obtained and in the end the demodulated signal.
- **A new architecture proposal of a digital communication system based on the QPSK modulation technique had been introduced and optimized as a pair.**
- **An optimization chart emphasizing the quality measures of the resources map was conducted for all FPGA designs.**
- **A comparison between previous works in the field of QPSK modulation on FPGA and this work in terms of the logic utilization is illustrated in table 5.1.**

## 6. HARDWARE CO-SIMULATION OF THE BPSK AND QPSK SYSTEM

A new technique, called Hardware Co-Simulation using FPGA appeared in the last few years. Studies using this new technique can be found in [50], [86], [98], [107], [137], [139]. The objective of the Hardware Co-Simulation technique is to accelerate an application. Therefore, an application is distributed in two or more hardware and software parts. Those parts of the application which are not critical for the process are kept in software, while the critical parts are implemented in hardware [107].

System Generator provides hardware co-simulation which makes possible to incorporate a design running in a FPGA directly into a Simulink simulation. The Simulink environment is used in order to verify the system functionality. When the design is made in System Generator, the results for the compiled area are generated in hardware. System Generator converts the Simulink math code into VHDL code that is recognized by the ISE software. The method is called Hardware/Software Co-Simulation and allows the compilation area to be tested in actual hardware and can speed up the simulation dramatically. The System Generator design flow for the hardware co-simulation technique is shown in fig. 6.1.

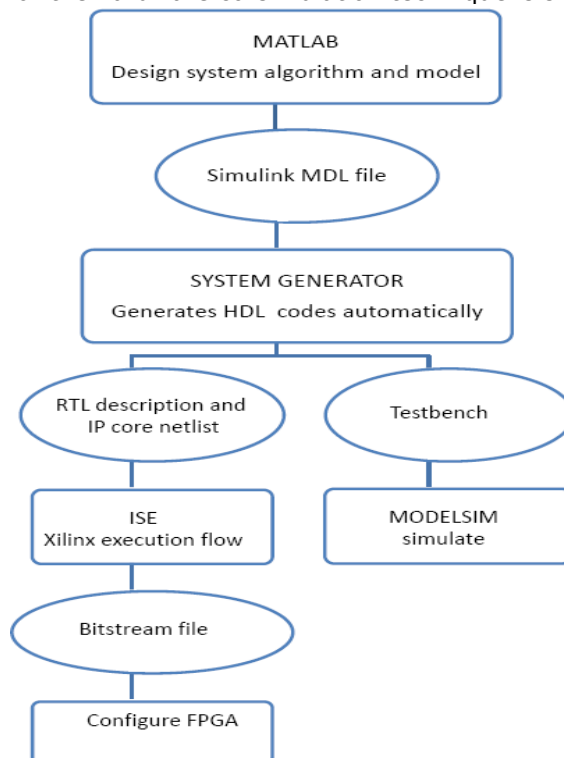


Fig. 6.1 System Generator design flow for the hardware co-simulation technique

This method enables building a hardware version of the model and, using Simulink environment, several tests can be performed in order to verify the functionality of the system in hardware. Hardware co-simulation supports FPGAs from Xilinx on boards that support JTAG or Ethernet connectivity. Several boards are predefined on System Generator for co-simulation, but not the Spartan 3E board, so in order to use it, it must be define.

When the compilation is complete, a new library, formed by a single block, encapsulates the hardware implementation of the DSP system. The block has inputs and outputs according with the number of the GatewayIn and GatewayOut ports. This new block includes all the functionality required for the system to be implemented on FPGA and is linked to a bitstream that will be downloaded into the FPGA during co-simulation. After starting the co-simulation, the System Generator will first download the associated bitstream. When the download is completed, System Generator reads the inputs from Simulation environment and sends the bitstream to the board using the JTAG connection. System Generator reads the output back from JTAG and sends it to Simulink in order to be displayed. The results can be verified by comparing the simulation output to the expected output. VHDL code can also be generated from System Generator. System Generator not only generates the HDL and netlist files for any model during the compilation process, but it also runs the downstream tools necessary to produce an FPGA configuration file [129].

The hardware co-simulation technique is used to validate the designs used in previous chapters. The designs are the BPSK modulator and system, respectively, the QPSK modulator and system.

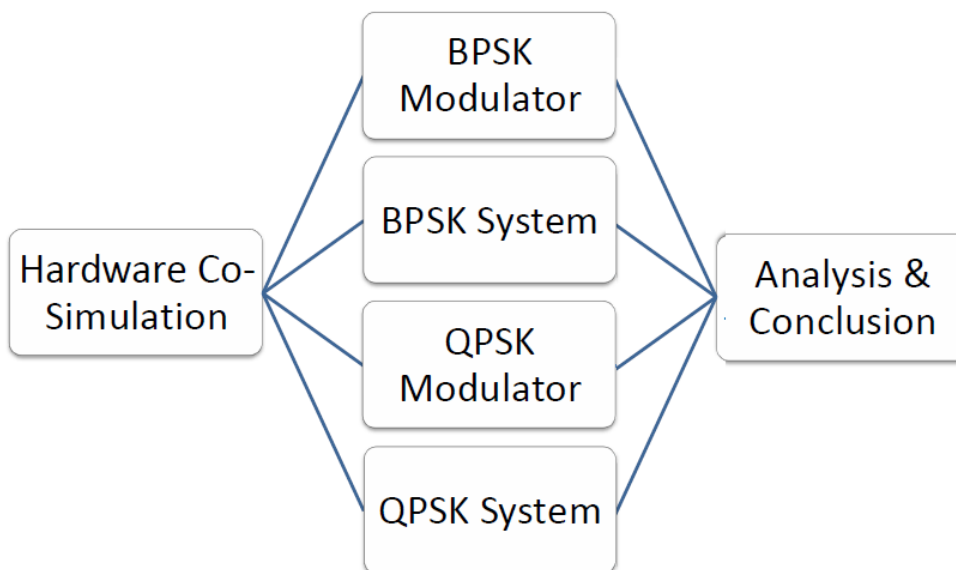


Fig. 6.2 Organization of the chapter

The experimental setup used for the hardware co-simulation technique consists of a computer, a Spartan 3E board and an oscilloscope. The Matlab/Simulink, System Generator and Xilinx ISE packages run on the computer, and the Xilinx ISE programs the Spartan 3E board.

### 6.1. BPSK Modulator

The purposed design is illustrated in fig. 6.3. The design is the same as the one in fig. 4.17 [128], [184], completed with the new block, the BPSK modulator hwcosim obtained after the hardware co-simulation.

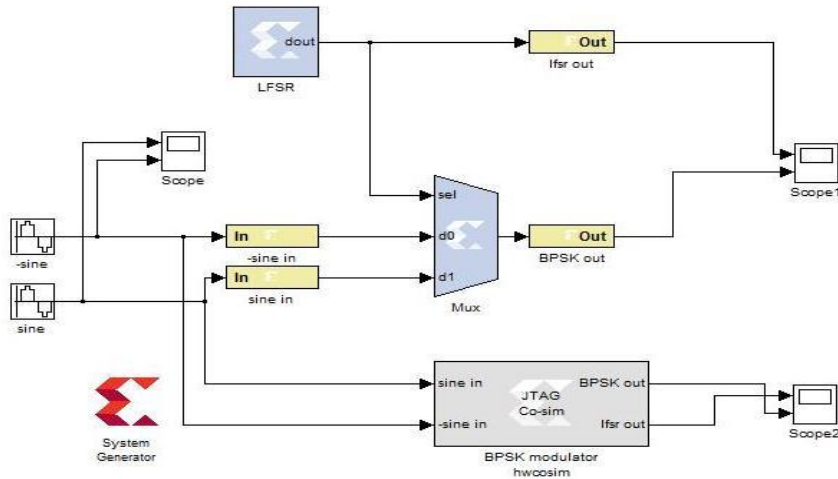


Fig.6.3 BPSK hwcosim modulator

The new BPSK modulator (hwcosim block) has two inputs and two outputs according to the number of the GatewayIn and GatewayOut ports. The block includes all the functionality required for the design to be implemented on the FPGA and is linked to a bitstream that will be downloaded into the FPGA during the co-simulation.

After the simulation is completed, the results are displayed as shown in fig. 6.4. The results can be verified with the results from fig. 4.18.

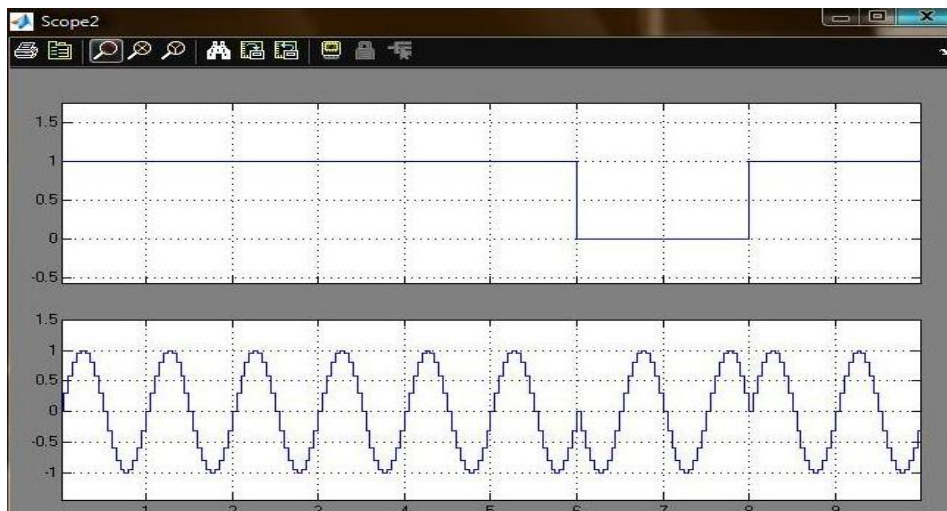


Fig. 6.4 The modulating and the modulated signals

After computing the VHDL code generated by System Generator, the summary of our design is illustrated as in fig. 6.5.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	21	9,312	1%	
Number of 4 input LUTs	25	9,312	1%	
Number of occupied Slices	31	4,656	1%	
Total Number of 4 input LUTs	25	9,312	1%	
Number of bonded IOBs	50	232	21%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	1.65			

Fig. 6.5 The design summary of the BPSK modulator

Comparing the design summary obtained by generating the VHDL code directly from System Generator and the design summary in fig. 4.31, the logic utilization of the board was lower in terms of the slice flip-flops, LUTs and occupied slices used. Fig. 6.6 illustrates a parallel between the resources used in implementing the BPSK modulator in the two versions (the first described in topic 4.2.4, and the second using the hardware co-simulation technique).

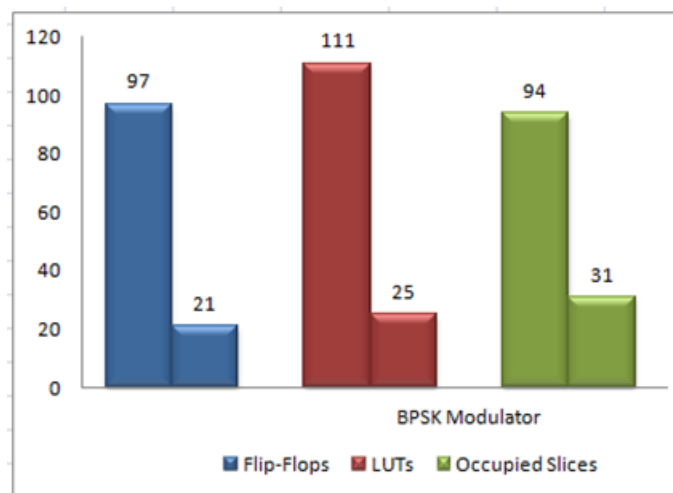


Fig. 6.6 Parallel between the resources used in implementing the BPSK modulator

## 6.2. BPSK System

The purposed design is illustrated in fig. 6.7 [128]. The design was completed with the new block, named `bpsk_system_hwcosim`, obtained after the hardware co-simulation.

The bpsk\_system hwcosim block has four inputs and three outputs according to the number of the GatewayIn and GatewayOut ports. The hwcosim block includes all the functionality required for the design to be implemented on the FPGA.

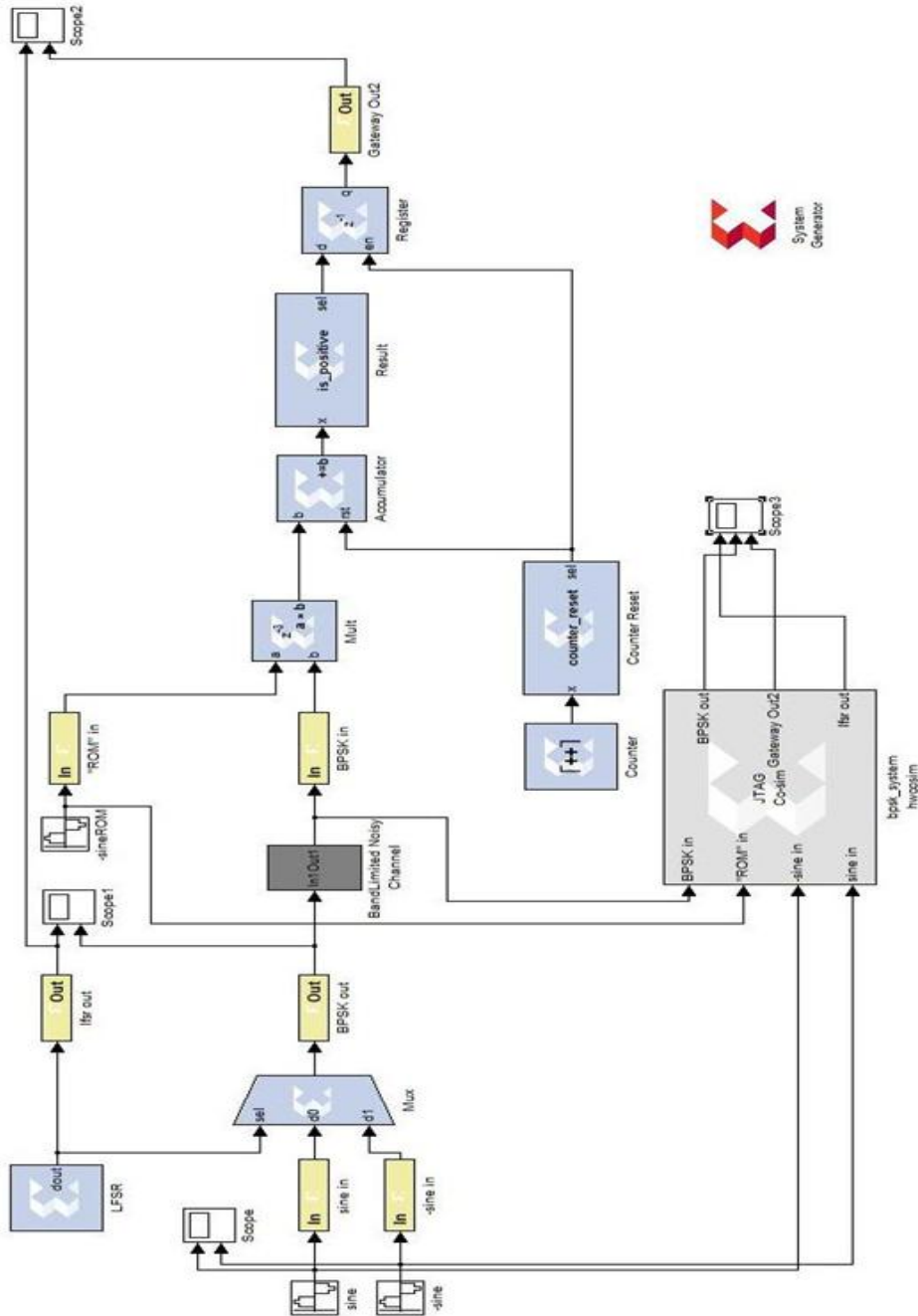


Fig. 6.7 The BPSK hwcosim system

After the simulation is completed, the results are displayed in fig. 6.8. The results obtained after the hardware co-simulation can be verified with the results from fig. 4.41. The design summary is illustrated in fig. 6.9.

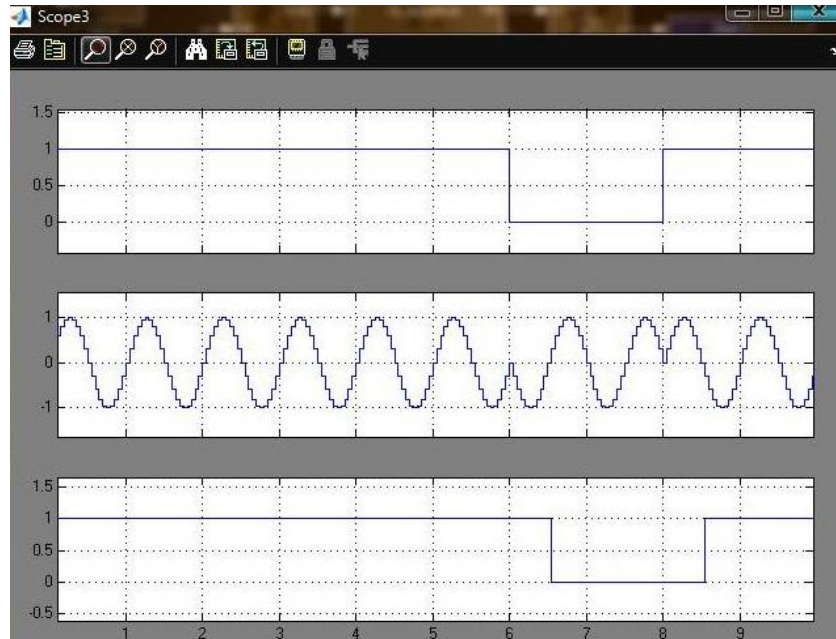


Fig. 6.8 The modulating signal, the BPSK modulated signal and the demodulated signal

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	92	9,312	1%	
Number of 4 input LUTs	60	9,312	1%	
Number of occupied Slices	76	4,656	1%	
Total Number of 4 input LUTs	63	9,312	1%	
Number of bonded IOBs	83	232	35%	
Number of BUFMUXs	1	24	4%	
Number of MULT18X18SIOs	1	20	5%	
Average Fanout of Non-Clock Nets	1.41			

Fig.6.9 The design summary of the BPSK system

Comparing the design summary from fig. 6.9 with the results from fig. 4.31 and fig. 4.47, the logic utilization is lower and is illustrated in fig. 6.10. The first column represents a mean between the resources from fig. 4.31 and fig. 4.47, and the second column represents the resources used in the hardware co-simulation technique. Also, by generating automatically VHDL code, the system is implemented on a single board, not on two boards like it was shown in chapter 4.

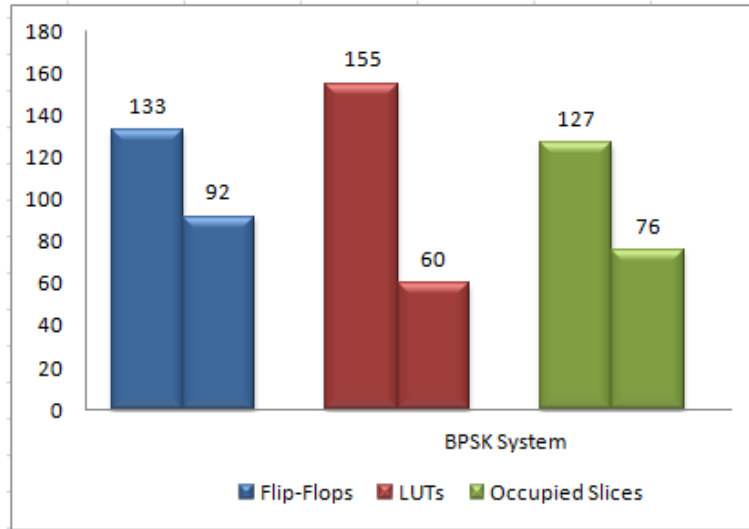


Fig. 6.10 Parallel between the resources used in implementing the BPSK system

### 6.3. QPSK Modulator

The proposed design, a QPSK modulator is illustrated in fig. 6.11 [124]. The design is the same as in fig.5.5, but completed with the QPSK modulator hwcosim generated by the hardware co-simulation. The block has four inputs and three outputs like in the original design.

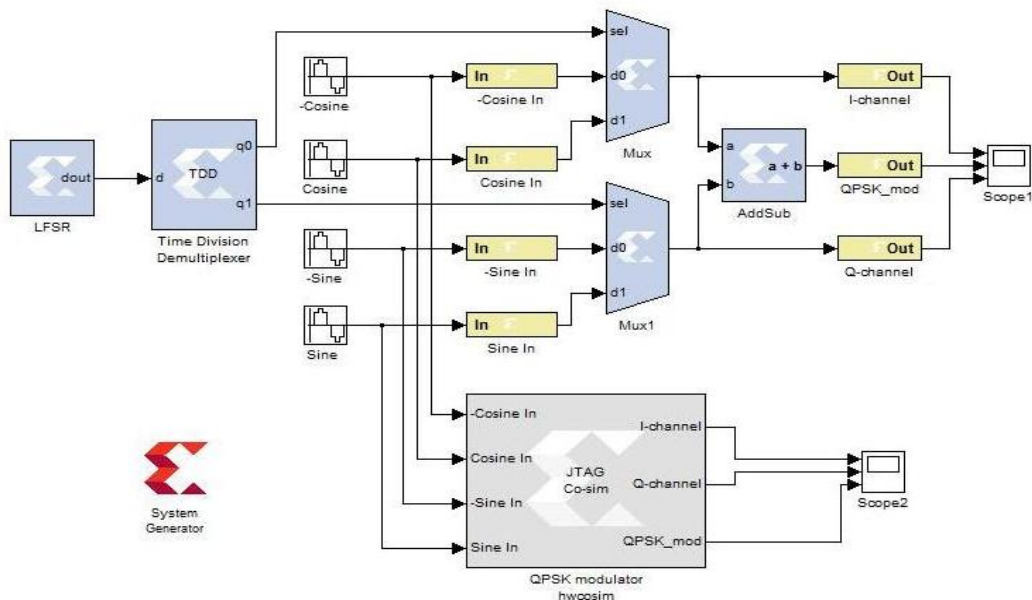


Fig. 6.11 The QPSK hwcosim modulator



The simulation results are shown in fig. 6.12 and are the same like in fig. 5.7. The signals shown in fig. 6.12 represent the I-channel which is modulated with a cosine waveform, the Q-channel which is modulated with a sine waveform and the QPSK modulated signal.

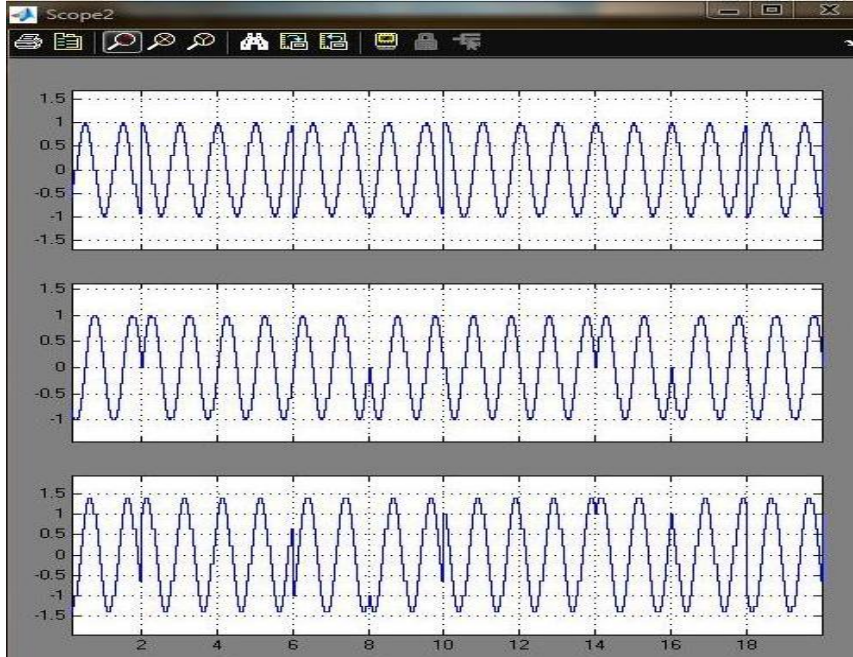


Fig. 6.12 I-channel, Q-channel and the QPSK modulated signals

The design summary is illustrated in fig. 6.13. Same as before, the utilization resources are lower if we generate VHDL code from System Generator, in terms of the slice flip-flops, LUTs and occupied slices used. If in this case, the number of the slice flip-flops was 41 and the number of LUTs was 67 from a total of 9312, in fig. 5.12, these numbers are higher: 95 for flip-flops and 157 for LUTs. A comparison between the two implementations is illustrated in fig. 6.14.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	41	9,312	1%	
Number of 4 input LUTs	67	9,312	1%	
Number of occupied Slices	70	4,656	1%	
Total Number of 4 input LUTs	67	9,312	1%	
Number of bonded IOBs	119	232	51%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	1.96			

Fig.6.13 The design summary of the QPSK modulator

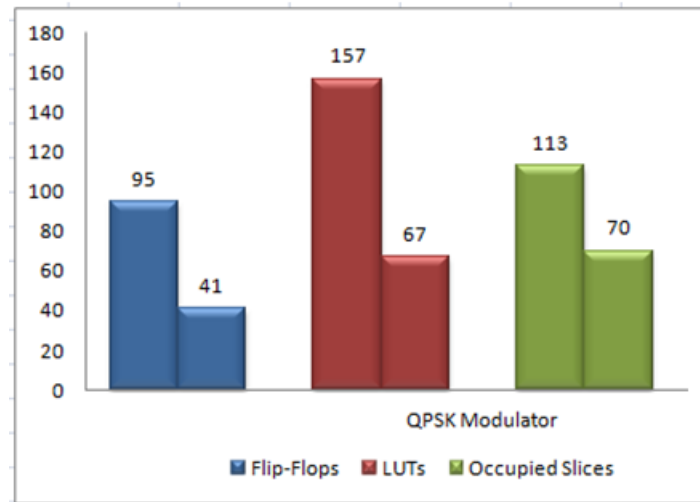


Fig. 6.14 Parallel between the resources used in implementing the QPSK modulator

#### 6.4. QPSK System

The proposed QPSK system is illustrated in fig. 6.15 [125]. The hardware co-simulation was run in order to obtain the `sist_qpsk hwcosim` block, necessary in implementing the system on FPGA. The new block has seven inputs and three outputs. By generating automatically VHDL code, the system is implemented on a single board, not on two boards as we did in our previous work.

After the hardware co-simulation, the results are illustrated in fig. 6.16.

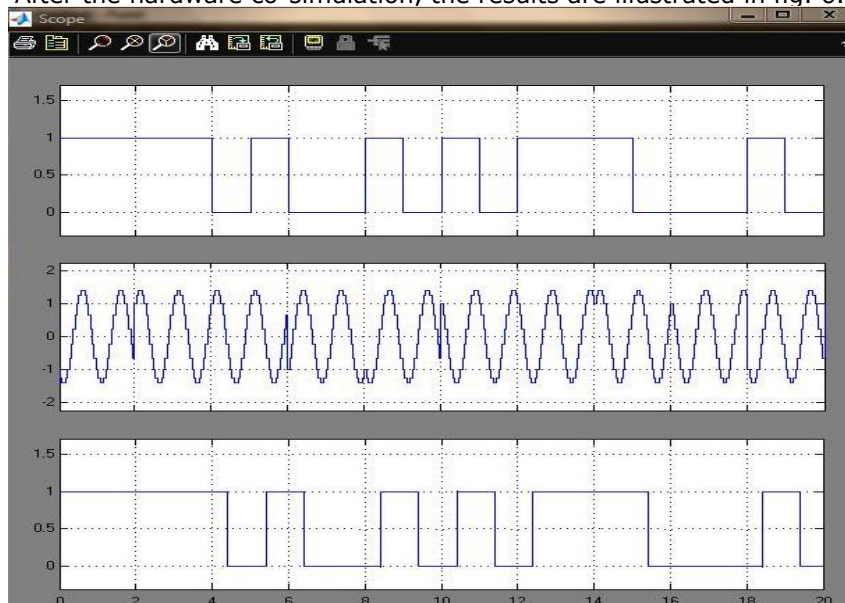


Fig. 6.16 The modulating signal, the QPSK modulated signal and the demodulated signal



In fig. 6.17, the design summary is shown. Same as in the other designs, the logic utilization of the Spartan3E board was lower in comparison with previous work. And also, the system is implemented on a single board, not on two like in chapter 5.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	111	9,312	1%	
Number of 4 input LUTs	102	9,312	1%	
Number of occupied Slices	115	4,656	2%	
Total Number of 4 input LUTs	105	9,312	1%	
Number of bonded IOBs	152	232	65%	
Number of BUFGMUXs	1	24	4%	
Number of MULT18X18SIOs	1	20	5%	
Average Fanout of Non-Clock Nets	1.65			

Fig. 6.17 Design summary of the QPSK system

Comparing the design summary from fig. 6.17 with the results from fig. 5.12 and fig. 5.25, the logic utilization is lower and is illustrated in fig. 6.18. The first column represents a mean between the resources from fig. 5.12 and fig. 5.25, and the second column represents the resources used in the hardware co-simulation technique. Also, by generating automatically VHDL code, the system is implemented on a single board, not on two boards like it was shown in chapter 5.

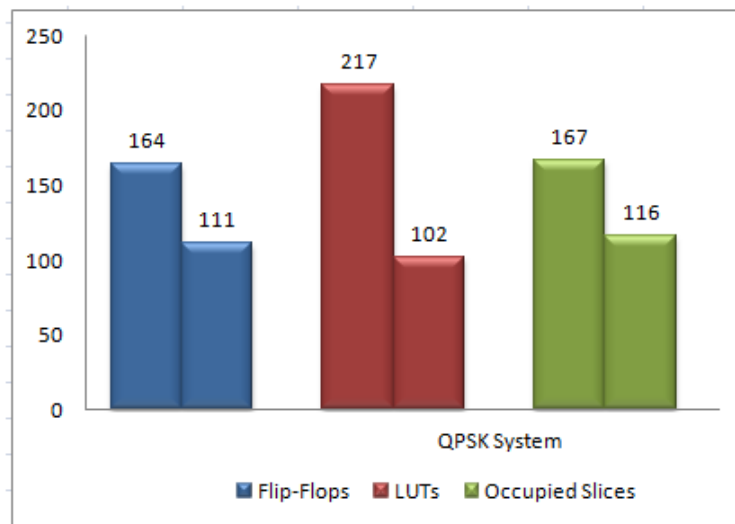


Fig. 6.18 Parallel between the resources used in implementing the QPSK system

## 6.5. Conclusions and Contributions

Chapter 6 describes the hardware co-simulation technique which allows the compilation portion to be tested in actual hardware.

Original hardware digital designs, the BPSK modulator and BPSK system as well as the QPSK modulator and QPSK system were validated in this chapter. Each design made in Simulink/ System Generator was implemented into hardware using two software packages: Matlab/Simulink and Xilinx ISE. The bridge between the two packages is represented by the Xilinx System Generator which converts the Simulink math code into VHDL code, recognized by the ISE software. The time used to simulate the designs with components from the Xilinx Blockset is longer than the time taken to simulate in Simulink. Still, using System Generator inside Simulink for bit and cycle time simulations is an order of magnitude faster than running the same simulation through an HDL simulator. Also, the time for designing a DSP system in the System Generator environment is smaller than the time used for writing VHDL code. A main advantage for a design made in System Generator is that it can be validated through simulations before implementing it in hardware.

The logic utilization of the board used in the hardware co-simulation technique is smaller if the generated VHDL code from System Generator is used, not writing the code from the beginning.

### Contributions:

- **Four unique implementations (BPSK modulator and system, QPSK modulator and system) were proposed and tested in actual hardware, using the hardware co-simulation technique.**
- **A comparison between the resources used in implementing the designs in the two versions was underlined (the first described in chapter 4 and 5, and the second using the hardware co-simulation technique).**

## 7. CONCLUSIONS AND CONTRIBUTIONS

### 7.1. Conclusions

The purpose of this thesis was to implement two digital communication systems. The two digital communication systems were based on the BPSK and QPSK modulation/demodulation techniques and were tested and implemented on FPGAs.

Chapter 2 presents the resources, the methods and the ways used in the research.

In the first part of chapter 2, an overview of reconfigurable computing was presented. Reconfigurable computing became popular due to the availability of field programmable gate arrays. The main architectures of the FPGAs were investigated; also the software tools used in implementing and simulating a design and programming the FPGAs were presented. These tradeoffs led to the decision/conclusion to use the Xilinx FPGA boards (Nexys2 and Spartan 3E) and the software tool (ISE Webpack) provided by the same corporation.

The simulation capabilities of the FPGA simulator were investigated and also compared to real life behavior. The exploitation and dissemination of the experiments were further investigated in two scientific papers: [117] and [118].

In the second part of chapter 2, the reasons why DSP is suitable for reconfigurable computing were brought up. The most useful DSP tool used for FPGA design is System Generator, a software tool from Xilinx based on the Matlab/Simulink environment. System Generator produces command files for FPGA synthesis, HDL simulation and implementation tool, all with the purpose of obtaining a hardware model. Also, the advantages and disadvantages of System Generator were outlined.

For understanding the implementation steps, a simple design, an adder made up of a constant and a sine generator, was simulated. First, only in the Simulink environment, and then, the design was implemented in System Generator, in four different ways:

- The constant and the sine generator were both Simulink blocks and pass through the GatewayIn blocks into the FPGA boundary which represents the System Generator environment;
- The constant in System Generator and the sine generator in Simulink;
- The constant in Simulink and the sine generator (DDS compiler) in System Generator;
- Both, the constant and the sine generator implemented in System Generator.

The signals of the simulated blocks were more accurate if Xilinx blocks had been used. Otherwise, the waveforms had square edges.

Chapter 3 began with basics about digital communication systems. The components of these systems were presented, but also the functionalities in the receiver and the transmitter. Besides the components of a digital communication system, the three main criteria of choosing a modulation schemes made the subject of the first part of this chapter.

The second part presented the theoretical backgrounds of the digital modulation techniques, underlining the BPSK and QPSK modulation schemes. Among all MPSK schemes, QPSK is the most used since it does not suffer from BER degradation while the bandwidth efficiency is increased. The advantage of QPSK over BPSK is obvious: QPSK transmits twice the data rate in a given bandwidth compared to BPSK, at the same BER. Two simple applications of the BPSK and QPSK modulation techniques were developed in the Matlab software.

In the third and last part of chapter 3, a survey about the BPSK and QPSK was presented. The survey began with early work in the field of digital modulation. Although FPGA technology appeared in the mid 1980s, because of the gap between the existing software tools and the develop circuits, at the end of 1990s and first part of the 2000s, this filed was at its beginning. Matlab was the software environment who reduced the gap and made possible the FPGA implementation using System Generator. The study continued with different papers where the FPGA technology can be found, like image, video, audio and signal processing, coding and wireless communications, biomedical applications, including FPGA implementations of the BPSK and QPSK modulation techniques [120]. The Hardware Co-Simulation technique was also mentioned and it would be later presented in this work.

Chapter 4 described the actual process of the BPSK modulation technique, as well as the implementation of the hardware itself. The Simulink and System Generator, included in the Matlab software, were used for the theoretical study and simulation, and for implementation the Nexys2 and Spartan 3E development boards and the Xilinx ISE software tool.

In the first part of the chapter, a BPSK Detector, all digital, was implemented, both the modulator and demodulator on the same FPGA board, and successfully designed with VHDL programming code. The BPSK detector had as a starting point paper [4] and an improvement of the detector was made. The newness of the detector was made by modifying the LFSR which had improved the randomness sequence of the signal. Analysis and comparison were made on a Nexys2 board. The Spartan-3E FPGA from the Nexys2 board has dedicated 18x18 bits hardware multiplier and so, there was no need for a relatively slow Booth multiplier.

In the second part of the chapter, a BPSK digital modulator was simulated, tested and implemented. In terms of the simulation process, two implementations of the BPSK Modulator in the Matlab/Simulink environment were proposed, the first with simple blocks and the second, with a block in which we wrote Matlab code. In the testing part, a proposal of four implementations of a BPSK modulator in System Generator was conducted. In the first, the three signals: the carrier and the modulating signal were generated external, and the modulated signal was obtained internal. In the second scheme, the carrier was generated external, and the modulating signal was generated internal by a LFSR. In the third scheme, all three signals were generated internal with the exception of the modulating signal which can be obtained either internal by the LFSR, or external by a pulse generator and in the fourth implementation, all signals were obtained internal. Two implementation of the BPSK modulator on the Spartan 3E board were designed, in the implementation part. The first one was based on the third proposal of the modulator made in System Generator and the signals were routed to a VGA monitor. If "1" was transmitted, the modulated signal remained same as the carrier, but if "0" was transmitted, the modulated signal was yielded with a 180° phase. The second implementation had as a model the fourth proposal of the modulator in System Generator and the signals were routed to an oscilloscope. Comparing the designs

summary obtained with the summary of the BPSK detector, the logic utilization of the board was lower in terms of the slice flip-flops and LUTs used. All of these make the design suitable in terms of propagation, implementation and logic utilization of the Spartan 3E boards used in this work.

A comparison in terms of the logic utilization between the implementation of the BPSK modulator via VGA and the implementation via oscilloscope was illustrated in fig. 4.49 and table 4.2 represented a comparison between previous works in the field of BPSK modulation on FPGA and this work.

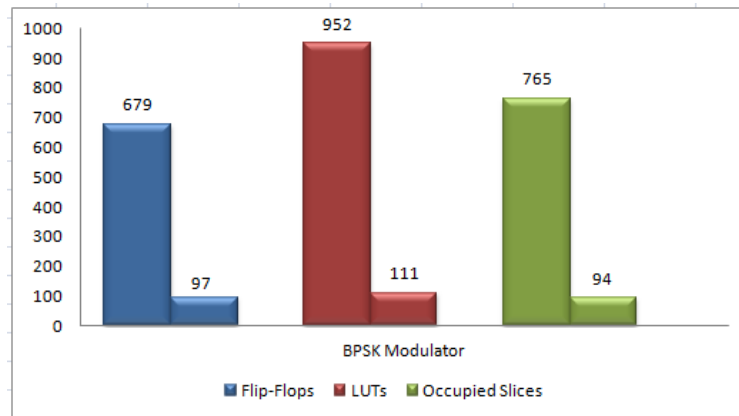


Fig. 4.49 Comparison of the logic resources

**Table 4.2 Comparison of previous work and practice in BPSK modulation on FPGA**

Modulation scheme	Reference	Board	Resource Utilization		
			Flip-Flops	LUTs	Slices
<b>B P S K</b>	[27]	Virtex-4	1%	1%	1%
	[30]	Virtex-4	6%	27%	25%
	[97]	Spartan 3E	13%	13%	16%
	[143]	Virtex-II	1%	9%	10%
	[160]	Virtex-4	11%	11%	15%
	<b>[122]</b>	<b>Spartan 3E</b>	<b>7%</b>	<b>10%</b>	<b>16%</b>
	<b>[123]</b>	<b>Spartan 3E (modulator)</b>	<b>1%</b>	<b>1%</b>	<b>2%</b>
	<b>[128]</b>	<b>Spartan 3E (demodulator)</b>	<b>1%</b>	<b>2%</b>	<b>3%</b>

In the third part of the chapter, a BPSK System, made of a modulator and demodulator, was the subject of simulation, testing and implementation. The simulation of the BPSK System was made in the Matlab/Simulink environment. The testing part of the system was made in System Generator. The modulating signal and the carrier were generated internal, the modulating signal by a LFSR and the carrier by a DDS Compiler. The modulated signal was obtained at the output of a



mux block and, then, passed through a communication channel where noise was added. In the demodulator, the carrier was recovered due to another DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal was then added with all the multiplied samples from the carrier in a period. The operation took place in the accumulator. Once the result had been obtained, it was compared with a decision threshold. If the compared signal was positive, the demodulator took the decision that '1' was transmitted, otherwise, '0'. The BPSK System implemented on the Spartan 3E boards had the same principle as the implementation in System Generator. Although System Generator has an option to generate the VHDL code, for this design the code was made from the beginning because the generated code was hard to read. The only difference was the carrier which was indeed generated internal, in a ROM memory, but made of 16 different values. The yielded carrier with  $180^\circ$  phase shift was obtained by reading the ROM memory later with 8 samples.

Chapter 5 described the QPSK modulation technique, as well as the implementation of the hardware itself.

In the first part of the chapter, an implementation of the QPSK Modulator in the Matlab/Simulink environment as well as a proposal of a QPSK modulator in System Generator was presented. The modulating signal was generated internal by a LFSR and is divided into two sequences: the odd-sequence for the I-channel and the even-sequence for the Q-channel. The sine and cosine waveform were generated outside the FPGA. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform which is a BPSK modulation. The QPSK modulated signal was obtained by adding the two modulated signals. The same principle was applied in implementing the modulator on the board, with the difference that the sine and cosine signals were generated inside the board. The sine signal was made of 16 different values kept in a ROM memory. The phase differences between the cosine and sine signal is  $90^\circ$ , and so the cosine was obtained from the sine by reading it later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK modulated signal was obtained. The design has been written in the VHDL programming code by Xilinx software.

In the second part of the chapter, an implementation of a QPSK System (Modulator and Demodulator) in the Matlab/Simulink environment as well as a proposal of a QPSK System in System Generator were studied. The binary sequence was generated internal by a LFSR and was divided into two sequences: the odd-sequence (I-channel) and the even-sequence (Q-channel). The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform, waveforms obtained outside the board. The QPSK modulated signal was obtained by adding the two modulated signals. The QPSK modulated signal was then passed through the same channel used in Simulink where noise was added. In the demodulator, the modulated signal with noise was multiplied twice, once with sine, and the second time with cosine. The two signals were kept in two accumulators and then compared with a threshold. If the compared signal was greater than the threshold, the demodulator took the decision that '1' was transmitted, otherwise, '0', obtaining the demodulated I and Q channels. The multiplexer was the one who puts the I and Q channels together obtaining the demodulated signal which made the initial modulating signal. The same principle was applied in implementing the modulator and the demodulator on the two boards, with the difference that the sine and cosine signals were generated inside the board. The sine signal was made of 16

different values kept in a ROM memory and the cosine was obtained reading the ROM memory later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK signal was obtained. The modulated signal was then sent to the DAC on the board in order to be sent through the channel. The modulated signal affected with noise arrives at the demodulator. The signal was converted into a digital form with the help of an AD board. This digital signal was then multiply with both signals, the sine and cosine generated internal in the ROM memory from the second board. The results were kept in an accumulator and compared with a decision threshold and so, the demodulated I-channel and Q-channel were obtained and in the end the demodulated signal.

Fig. 5.26 summarized the resources of the two boards used in implementing the QPSK modulator and demodulator.

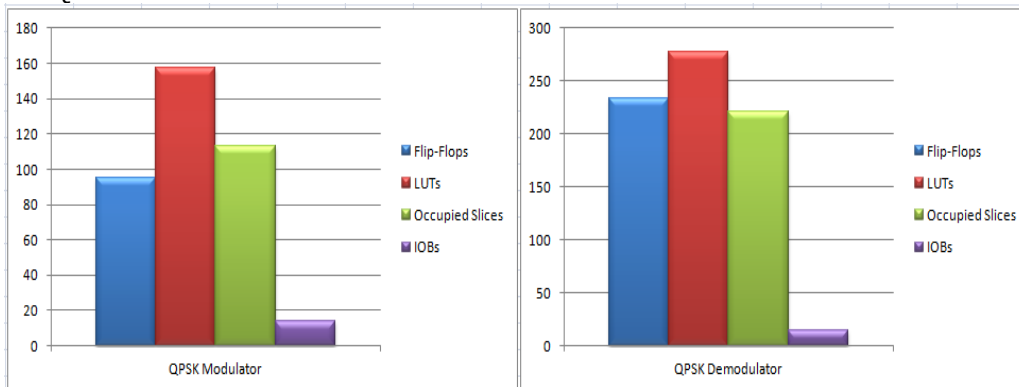


Fig. 5.26 Comparison of the logic resources of the QPSK modulator and demodulator

Table 5.1 illustrated a comparison between previous works in the field of QPSK modulation on FPGA and this work.

**Table 5.1 Main Comparison of previous work and practice in QPSK modulation on FPGA**

Modulation scheme	Reference	Board	Resource Utilization		
			Flip-Flops	LUTs	Slices
QPSK	[30]	Virtex-4	1%	16%	17%
	[60]	Virtex-II	1%	2%	2%
	[135]	Virtex-II	1%	13%	13%
	[124]	<b>Spartan 3E (modulator)</b>	<b>1%</b>	<b>1%</b>	<b>2%</b>
	[125]	<b>Spartan 3E (demodulator)</b>	<b>2%</b>	<b>2%</b>	<b>2%</b>

Chapter 6 described the hardware co-simulation technique which allowed the compilation FPGA area to be tested in actual hardware. Original hardware digital designs, the BPSK modulator and BPSK system as well as the QPSK modulator and

QPSK system were validated in this chapter. Each design made in Simulink/ System Generator was implemented into hardware using two software packages: Matlab/Simulink and Xilinx ISE. The bridge between the two packages was represented by the Xilinx System Generator which converted the Simulink math code into VHDL code, recognized by the ISE software. The time used to simulate the designs with components from the Xilinx Blockset was longer than the time taken to simulate in Simulink. Still, using System Generator inside Simulink for bit and cycle time simulations was an order of magnitude faster than running the same simulation through an HDL simulator. Also, the time for designing a DSP system in the System Generator environment was smaller than the time used for writing VHDL code. A main advantage for a design made in System Generator was that it can be validated through simulations before implementing it in hardware. The logic utilization of the board used in the hardware co-simulation technique is smaller if the generated VHDL code from System Generator is used, not writing the code from the beginning.

## 7.2. Contributions

- An overview of reconfigurable computing was presented, outlining the strengths and the advantages of FPGAs in contrast to ASIC and GPP.
- **A common but accurate methodology on an FPGA, simulations checked by measurements, was followed.**
- **The capabilities of the FPGA simulator, the Xilinx ISE, were analysed and compared with practical results:**
  - **The delays of the post-route simulation were investigated and also compared with measured delays;**
  - **Corrections had been made to results according to the type of their use.**
- **The advantages and disadvantages of the System Generator were outlined.**
- A simple design was simulated in Simulink and System Generator in order to introduce and understand the basic concepts of the MATLAB simulator:
  - To test the design functionality, the adder was simulated in Simulink;
  - To test the functionality as an FPGA design, the Simulink blocks were translated in dedicated Xilinx blocks from System Generator, obtaining four different designs of the adder.
- Different implementations of the same model were compared: the signals of the simulated blocks were more accurate if Xilinx blocks were used.
- Experimental results proved the precision of the two simulators: the Xilinx ISE and MATLAB/Simulink and System Generator.
- A review of the BPSK and QPSK, two very important modulation techniques, was made.
- MATLAB is generally used in research and development. In order to thoroughly analyse the efficiency of the simulator, two applications were developed, studied and compared.
- **A literature survey regarding the main previous work and general practice in phase modulation, of the BPSK and QPSK modulations, on FPGA had been conducted and summarized in table 3.4.**
- **An improvement of a BPSK detector was made: the LFSR inside the detector was modified which improved the randomness sequence of**

- the signal. Also, it was no need for a relatively slow Booth multiplier, since the board had a dedicated 18x18 bits hardware multiplier.**
- Two implementations of a BPSK Modulator in the Matlab/Simulink environment had been proposed, first with simple Simulink blocks and the second with an embedded Matlab function.
  - Four implementations of a BPSK modulator in System Generator were also proposed:
    - the carrier and the modulating were generated external and the modulated signal was obtained internal;
    - the carrier is generated external and the modulating signal is generated internal by a LFSR;
    - all three signals were generated internal with the exception of the modulating signal which can be obtained either internal by the LFSR, or external by a generator;
    - all signals were generated internal.
  - **Two new implementation of the BPSK modulator were made:**
    - **The first method was a visual one, in which two different colored rectangular were displayed on a VGA monitor;**
    - **In the second implementation, the signals were routed to an oscilloscope in order to be measured.**
  - **A comparison in terms of the logic utilization between the implementation of the BPSK modulator via VGA and the implementation via oscilloscope was made and presented in fig.4.49.**
  - In order to test the functionality principle of the BPSK System, simulations were made in the Matlab/Simulink environment.
  - Simulations were made in System Generator to test the hardware part of the BPSK System: the modulating signal was generated internal by a LFSR and the carrier, also internal, by a DDS Compiler. The modulated signal obtained was passed through a communication channel where noise was added. In the demodulator, the carrier was recovered due to another DDS compiler and then multiplied with the modulated signal affected by noise. The obtained signal was then added with all the multiplied samples from the carrier in a period. The operation took place in the accumulator. Once the result had been obtained, it was compared with a decision threshold. If the compared signal was positive, the demodulator took the decision that '1' was transmitted, otherwise, '0'.
  - **A new architecture proposal of a digital communication system based on the BPSK modulation technique had been introduced and optimized as a pair.**
  - **An optimization chart emphasizing the quality measures of the resources map was conducted for all FPGA designs.**
  - **A comparison between previous works in the field of BPSK modulation on FPGA and this work in terms of the logic utilization is illustrated in table 4.2.**
  - The design of a QPSK Modulator in the Matlab/Simulink environment had been proposed.
  - A QPSK modulator in System Generator, based on the Simulink design, was also proposed: the modulating signals was generated internal by a LFSR and divided in an odd and an even sequence. The odd sequence was modulated

with sine and the even one with cosine. The sine and cosine signals were generated inside the board. The sine signal was made of 16 different values kept in a ROM memory and the cosine was obtained from the sine by reading it later with 4 samples. The odd-sequence was modulated with the cosine waveform and the even-sequence with the sine waveform. Adding the two modulated signals, the QPSK modulated signal was obtained.

- The QPSK modulator was implemented on FPGA with the signals routed to an oscilloscope in order to be measured.
- Simulations of the QPSK System were made in the Matlab/Simulink environment for verifying the functionality principle;
- The hardware part of the QPSK system was tested in System Generator: the modulated signal affected with noise arrives at the demodulator. The signal is converted into a digital form with the help of an AD board. This digital signal is then multiply with both signals, the sine and cosine generated internal in the ROM memory from the second board. The results are kept in an accumulator and compared with a decision threshold and so, the demodulated I-channel and Q-channel are obtained and in the end the demodulated signal.
- **A new architecture proposal of a digital communication system based on the QPSK modulation technique had been introduced and optimized as a pair.**
- **An optimization chart emphasizing the quality measures of the resources map was conducted for all FPGA designs.**
- **A comparison between previous works in the field of QPSK modulation on FPGA and this work in terms of the logic utilization is illustrated in table 5.1.**
- **Four unique implementations (BPSK modulator and system, QPSK modulator and system) were proposed and tested in actual hardware, using the hardware co-simulation technique.**
- **A comparison between the resources used in implementing the designs in the two versions was underlined (the first described in chapter 4 and 5, and the second using the hardware co-simulation technique).**

### 7.3. Future Work

Since implementation of different systems on FPGAs became popular quite recently, a few years ago, there are a lot of other modulation techniques on FPGA which can be the subject of future work.

In the case of BPSK and QPSK systems on FPGA implemented in this thesis, applications in the field of wireless and underwater communication can be developed.

The similarity of empirical real-time and simulated results shows the success of FPGA and DAC implementations that would be further processed by external analog RF devices for complete wireless SDR system or underwater communications. On the other hand, timing issues such as sample rate, constraints, and precoding like source and channel coding can also be added in the BPSK and QPSK systems for error checking and control.

#### 7.4. List of Publications

- [1] **S.O.Popescu**, A.S. Gontean, I. Lie, *Comparing FPGA Behavioral Simulation, Post – Route Simulation with real – life experiments*, Proceedings of the 15<sup>th</sup> International Symposium for Design and Technology of Electronics Packages, Hungary, 2009, pp. 75 – 80.
- [2] **S.O.Popescu**, A.S. Gontean, *Virtex – II Timing Simulation vs. Reality*, Telecommunications Forum Telfor, Serbia, 2009, pp. 755 – 758
- [3] **S.O.Popescu**, *Current Stage of Basic Modulation Implemented on the FPGA*, 1<sup>st</sup> Scientific Report, Romania, 2010.
- [4] **S.O.Popescu**, G. Budura, A.S. Gontean, *Review of PSK and QAM – Digital Modulation Techniques on FPGA*, Joint Conferences on Computational Cybernetics and Technical Informatics CONTI, Romania, 2010, pp. 327 – 332
- [5] **S.O.Popescu**, *Implementation of the BPSK modulation/ demodulation on FPGA*, 2<sup>nd</sup> Scientific Report, Romania, 2010.
- [6] **S.O.Popescu**, A.S.Gontean, F.Alexa, *Improved FPGA-based Detector*, Proceedings of the 6<sup>th</sup> IEEE International Symposium on Applied Computational Intelligence and Informatics, Romania, 2011, pp. 455-458.
- [7] **S.O.Popescu**, A.S.Gontean, G.Budura, *Simulation and Implementation of a BPSK modulator on FPGA*, Proceedings of the 6<sup>th</sup> IEEE International Symposium on Applied Computational Intelligence and Informatics, Romania, 2011, pp. 459-463.
- [8] **S.O.Popescu**, A.S.Gontean, D.Ianchis, *QPSK Modulator on FPGA*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp. 359-364.
- [9] **S.O.Popescu**, A.S.Gontean, D.Ianchis, *Implementation of a QPSK System on FPGA*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp. 365-370.
- [10] D. Ianchis, V. Tiponut, **S. Popescu**, Z. Haraszy, *Improved Collision Detection System Inspired from the Neural Network of the Locust*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp.211-215.
- [11] **S.O.Popescu**, A.S. Gontean, *Performance comparison of the BPSK and QPSK Modulation Techniques on FPGA*, Proceedings of the 17<sup>th</sup> International Symposium for Design and Technology of Electronics Packages, Romania, 2011, pp. 257 – 260.
- [12] **S.O.Popescu**, A.S. Gontean, G.Budura, *BPSK System on FPGA*, Proceedings of the 10<sup>th</sup> Jubilee International Symposium on Applied Machine Intelligence and Informatics, Slovakia, 2012, pp. 301-306.
- [13] **S.O.Popescu**, A.S. Gontean, G.Budura, *Hardware Co-Simulation of the BPSK and QPSK Systems on FPGA*, Proceedings of the 11<sup>st</sup> IFAC/IEEE International Conference on Programmable Devices and Embedded Systems, Czech Republic, *unpublished*.
- [14] **S.O.Popescu**, A.S. Gontean, G.Budura, *Modern Implementation of a BPSK Modulator on FPGA*, submitted at the International Journal of Advanced Computer Science, *unpublished*.

## BIBLIOGRAPHY

- [1] Agilent, *Understanding Oscilloscope Frequency Response and Its Effect on Rise-Time Accuracy*, Application Note 1420, 2002
- [2] Agilent, *Relating wideband DSO rise time to bandwidth: Lose the 0.35*, Agilent 55W-18024-2, 2009
- [3] J.P.Agrawal, O.Farook, C.R.Sekhar, *SIMULINK Laboratory Exercises in Communication Technology*, Proceedings of the 2005 American Society for Engineering Education, SUA, 2005.
- [4] F.Ahamed, A.Scorpino, *An educational digital communications project using FPGAs to implement a BPSK Detector*, IEEE Transactions on Education, Vol.48, No.1, 2005, pp.191-197.
- [5] M.F.Ain, M.S.Naghmash, Y.H.Chye, *Synthesis of HDL Code for FPGA Design Using System Generator*, European Journal of Scientific Research, Vol.45, No.1, 2010, pp.111-121.
- [6] J. Aisbett, *Automatic Modulation Recognition Using Time-Domain Parameters*, Journal Signal Processing, Vol.13, Issue 3, 1987, pp. 323-328.
- [7] I.Alecksi, Z.Hocenski, P.Horvat, *Acoustic Localization based on FPGA*, Proceedings of the 33<sup>rd</sup> International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 2010, pp.656-658.
- [8] Altera Corporation, *FPGA vs DSP Design Realibility and Maintenance*, USA, 2007.
- [9] Altera Corporation, *DSP Builder Handbook. Introduction to DSP Builder*, USA, 2011.
- [10] J. B. Anderson, *Digital Transmission Engineering*, IEEE Press - Wiley, Canada, 2005.
- [11] A.Awawadeh, S.S.U. Chander, A.Kichenaradjo, M.A. Soderstrand, *Hardware efficient BPSK and QPSK detector*, Proceedings of the 36<sup>th</sup> Asilomar Conference on Signals, Systems and Computers, USA, 2002, pp. 1060-1063.
- [12] O.Azarmanesh, S.Bilen, *Developing a rapid prototyping method using a Matlab/ Simulink/ FPGA development to enable importing legacy code*, in Proceedings of the SDR '08 Technical Conference and product Exposition, USA, 2008.
- [13] E.Azzouz, A.K.Nandi, *Automatic Modulation Recognition of Communication Signals*, Kluwer, Netherlands, 1996.
- [14] M.Barr, *A reconfigurable computing primer*, Multimedia System Design, Vol.9, 1998, pp. 44-47.
- [15] D.Benson, *The Design and Implementation of a GPS Receiver Channel*, DSP magazine, pp. 50-53.
- [16] D.Bernal, P.Closas, J.A.Fernandez-Rubio, *Digital I-Q Demodulation in Array Processing: Theory and Implementation*, Proceedings of the 16<sup>th</sup> European Signal Processing Conference, Switzerland, 2008.
- [17] S.U.Bhandari, S.Subbaraman, S.Pujari, *Digital Signal Modulator on FPGA using on the fly partial reconfiguration*, Proceedings on the International Conference on Advances in Computing, Control and

- Telecommunications, India, 2010, pp. 711-713.
- [18] T.Bhatt, D.McCain, *Matlab as a Development Enviroment for FPGA Design*, Proceedings of the 42<sup>nd</sup> Conference on Design Automation, 2005, pp. 607-610.
- [19] C.Bobda, *Introduction to Reconfigurable Computing. Architectures, Algorithms and Applications*, Springer Ed., Netherlands, 2007.
- [20] T.Brack, U.Wasenmuller, D.Schmidt, N.Wehn, *Design Space Exploration for Frequency Synchronization of BPSK/QPSK Bursts*, Advances in Radio Science, Vol.3, 2005, pp. 337-341.
- [21] S.C.Bu, G.I.Jee, H.S.Cho, S.H.Im, *A FPGA-based software GPS receiver design using Simulink*, Proceedings of the 2004 International Symposium on GNSS/GPS, Australia, 2004.
- [22] R.M.Buehrer, *Spread Spectrum Communications*, Summer, 2007.
- [23] S.Bushnaq, T.Nakura, M.Ikeda, K.Asada, *All digital wireless transceiver using modified BPSK and 2/3 sub-sampling technique*, Proceedings of the IEEE 8<sup>th</sup> International Conference on ASIC, China, 2009, pp. 469-472.
- [24] Y.T.Chan, L.G.Gadbois, *Identification of the Modulation Type of a Signal*, Journal Signal Processing, vol. 16, 1989, pp. 149-154.
- [25] H.S.Cho, S.H.Im, G.I.Jee, *A FPGA-based Software GPS Receiver Implementation Using Simulink and Xilinx System Generator*, Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation, USA, 2005, pp. 234-240.
- [26] P.P.Chu, M.Kifle, *A Reconfigurable Communications System for Small Spacecraft*, NASA Technical Report, TM-212534, 2004.
- [27] Y.H.Chye, M.F.Ain, N.M.Zawawi, *Design of BPSK Transmitter using FPGA with DAC*, Proceedings of the 2009 IEEE 9<sup>th</sup> Malaysia International Conference on Communications, Malaysia, 2009, pp. 451-456.
- [28] K.Compton, S.Hauck, *Reconfigurable computing: a survey of systems and software*. ACM Computing Surveys, 34(2), 2002, pp. 171-210.
- [29] G.Cuidong, Z.Qi, *Design and Simulation of a FPGA-Based Demodulator for Low-bit Remote Subcarrier Signal*, Proceedings of the International Conference on Wireless Networks and Information Systems, China, 2009, pp.85 - 88.
- [30] F.M.Demiri, U.Kafadar, S.Dikmese, H.Dincer, *FPGA Based Implementation of Communication Modulation*, Proceedings of the IEEE 15th Signal Processing and Communications Applications, Turkey, 2007, pp.1-4.
- [31] C.Dick, H.Pedersen, *Design and Implementation of High-Performance FPGA Signal Processing Datapaths for Software Defined Radio*, Proceedings of the 37<sup>th</sup> Conference on Signals Systems and Computers, Vol.1, 2003, pp.1-16.
- [32] C.Dick, *Rediscovering signal processing: a configurable logic based approach*, Proceedings of the 37<sup>th</sup> Conference on Signals Systems and Computers, Vol.2, 2003, pp. 1370-1374.
- [33] Digilent Inc., *Nexys2 board Reference Manual*, 2008.
- [34] T.W.Dittmer, *Digitally modulated RF systems: A primer on current techniques*, SMPTE Journal, Vol.108, Issue.9, Society of Motion Picture and Television Engineers, pp.1-7.
- [35] L.V. Dominguez, J.M.P. Borrallo, J.P. Garcia, B.R.Mezcua, *A General Approach to the Automatic Classification of Radiocommunication Signals*, Journal Signal Processing, vol.22, 1991, pp. 239-250.



- 
- [36] P.Dondon, J.M.Micouleau, P.Kadionik, *Improving Learning Efficiency for Digital Modulation Courses*, WSEAS Engineering Educational, Greece, Vol. 2, Issue 4, 2005.
- [37] P.Dondon, J.M.Micouleau, J.Legall, P.Kadionik, *Design of a Low Cost BPSK Modulator/ demodulator for a Practical Teaching of Digital Modulation Techniques*, Proceedings of the 4<sup>th</sup> International Conference on Engineering Educational, Greece, 2007, pp. 61-66.
- [38] R.Duren, J.Stevenson, M.Thompson, *A Comparison of FPGA and DSP Development Environments and Performance for Acoustic Array Processing*, Proceedings of the 50<sup>th</sup> Midwest Symposium on Circuits and Systems, Canada, 2007, pp.1177-1180.
- [39] G.Elamary, G.Chester, J.Neasham, *A simple digital VHDL QPSK modulator designed using CPLD/FPGAs for biomedical devices applications*, Proceedings of the World Congress on Engineering 2009 Vol I, London, UK, 2009, pp. 376-381.
- [40] G.Estrin, *Organization of Computer Systems – The fixed Plus variable Structure Computer*, in Proceedings of the Western Joint Computer Conference, USA, 1960, pp.33-40.
- [41] M.P. Fitz, *Fundamentals of Communication Systems*, McGraw-Hill, USA, 2007.
- [42] J. Frigo, T. Braun, J. Arrowood, and M. Gokhale, *Comparison of high-level FPGA design tools for a BPSK signal detection application*, Proceedings of the SDR '03 Technical Conference and product Exposition, USA, 2008.
- [43] P.Garcia, K.Compton, M.Schulte, E.Blem, W. Fu, *An Overview of Reconfigurable Hardware in Embedded Systems*, EURASIP Journal on Embedded Systems, Hindawi Publishing Corporation, Vol.2006, Article ID 56320, pp. 1-19.
- [44] I. Glover, P.Grant, *Digital Communications*, Prentice-Hall, England, 2000.
- [45] B.Godbole, D. Aldar, *Performance Improvement by Changing Modulation Methods for Software Defined Radios*, International Journal of Advanced Computer Science and Applications, Vol. 1, No. 6, 2010, pp.72-79.
- [46] M.Gokhale, P.S.Graham, *Reconfigurable Computing. Accelerating Computation with Field Programmable Gate Arrays*, Springer Ed., USA, 2005.
- [47] J.Gonçalves, J.Fernandes, M.Silva, *A Reconfigurable Quadrature Oscillator Based on a Direct Digital Synthesis System*, Proceedings of the XI Conference on Design of Circuits and Integrated Systems, Spain, 2006.
- [48] I.Grout, *Digital Systems Design with FPGAs and CPLDs*, Ed. Elsevier, USA, 2008.
- [49] A.Gürgör, F.Arikan, O.Arikan, *Simulation of a digital communication system*, in Proceedings of the 13th European Signal Processing Conference, Turkey, 2005.
- [50] M.A.Hadidi, R.S.Rzouq, J.S.Al-Azzih, *Design Flow for Data Transfer System Development by the Example of QPSK BaseBand System*, Proceedings on the 2<sup>nd</sup> International Conference on Education Technology and Computer, China, 2010, pp.V2-288 - V2-291.
- [51] D.Haessig, J.Hwang, S.Gallagher, M.Uhm, *Case- Study of a Xilinx System Generator Design Flow for Rapid Development of SDR Waveforms*, Proceedings of the SDR 05 Forum Technical Conference and Product Exposition, SUA, 2005
- [52] M.Haldar, A.Nayak, N.Shenoy, A.Choudhary, P.Banerjee, *FPGA Hardware*

- Synthesis from Matlab*, Proceedings of the 14<sup>th</sup> International Conference on VLSI Design, 2001, pp. 299-304.
- [53] M.Haldar, A.Mayak, A.Choudhary, P.Banerjee, *A system for Synthesizing Optimized FPGA Hardware from Matlab*, Proceedings of the 2001 International Conference on ComputerAided Design, 2001, pp. 314-319.
- [54] T.Hall, D.Anderson, *A Framework for Teaching Real-Time Digital Signal Processing With Field-Programmable Gate Arrays*, IEEE Transactions on Education, Vol. 48, No. 3, 2005, pp. 551-558.
- [55] T.Hall1, J.Hamblen, *Using FPGAs to Simulate and Implement Digital Design Systems in the Classroom*, Proceedings of the 2006 ASEE Southeastern Section Conference, USA, 2006.
- [56] P.H.Halpern, P.E.Mallory, *A Simple Method for Distinguishing Modulation Types*, IEEE Transition on ASSP, vol. 30, 1982, pp. 97-99.
- [57] K.N.Haq, A.Mansour, S.Nordholm, *Comparison of digital modulation classification based on statistical approach*, Proceedings of the 10<sup>th</sup> Postgraduate Electrical and Computer Symposium, Australia, 2009.
- [58] K.N.Haq, A.Mansour, S.Nordholm, *Recognition of digital modulated signals based on statistical parameters*, Proceedings of the 4<sup>th</sup> IEEE International Conference on Digital Ecosystems and Technologies, Dubai, 2010, pp. 565-570.
- [59] K.N.Haq, A.Mansour, S.Nordholm, *Classification of digital modulated signals based on time frequency representation*, Proceedings of the 4<sup>th</sup> International Conference on Signal Processing and Communication Systems, Australia, 2010, pp.1-5.
- [60] I.Hatai, I.Chakrabarti, *Multi-Standard Programmable Baseband Modulator for Next Generation Wireless Communication*, International Journal of Computer Networks & Communications, Vol.2, No.4, 2010, pp. 58-71.
- [61] S. Hauck, A. Agarwal, *Software Technologies for Reconfigurable Systems*, IEEE Transactions on VLSI Systems, 1996.
- [62] S.Hauck, T.W.Fry, M.M.Hosler, J.P.Kao, *The Chimaera Reconfigurable Functional Unit*, in Proceedings of the 5<sup>th</sup> Annual IEEE Symposium on FPGAs for Custom Computing Machines, USA, 1997, pp. 87-96.
- [63] S.Hauck, *The Future of Reconfigurable Systems*, Proceedings of the 5<sup>th</sup> Canadian Conference on Field Programmable Devices, Canada, 1998.
- [64] S. Hauck, *The Roles of FPGAs in Reprogrammable Systems*, Proceedings of the IEEE, Vol. 86, No. 4, 1998, pp. 615-638.
- [65] S.Hanck, A. DeHon, *Reconfigurable Computing. The theory and practice of FPGA-Based Computation*, Elsevier, USA, 2008.
- [66] J.E. Hipp, *Modulation Classification Based on Statistical Moments*, Milcom-86, vol. 2, 1986, pp. 20.2.1-6.
- [67] L.Honng, K.C.Ho, *Identification of digital modulation types using the wavelet transform*, Proceedings of the IEEE Military Communications Conference Proceedings, Vol.1, USA, 1999, pp. 427-431.
- [68] P.A.Hsiung, M.D.Santambrogio, C.H.Huang, *Reconfigurable System Design and Verification*, CRC Press, USA, 2009.
- [69] H.Hu, C.Yuan, *The design on FPGA-based Correlator in GPS Receiver using ISE*, International Journal of Intelligent Information Technology Application, Vol. 3, No.2, 2010, pp. 92-97.
- [70] B. Hunt, R. Lipsman, J. Rosenberg, K. Coombes, J. Osborn, G. Stuck , *A Guide to MATLAB for Beginners and Experienced Users*, Cambridge University Press, USA, 2001

- 
- [71] J.Hwang, B.Milne, N.Shirazi, J.Stroomer, *System Level Tools for DSP in FPGAs*, Proceedings of the 11<sup>th</sup> International Conference on Field Programmable Logic, Northern Ireland, U.K., 2001.
- [72] M.Islam, M.A.Hannan, S.A.Samad, A.Hussain, *Modulation Technique for Software Defined Radio Application*, Australian Journal of Basic and Applied Sciences, Vol.3, No.3, 2009, 1780-1785.
- [73] M.Islam, M.A.Hannan, S.A.Samad, A.Hussain, *Bit-Error-Rate (BER) for modulation technique using Software defined Radio*, Proceedings of the International Conference on Electrical Engineering and Informatics, Malaysia, 2009, pp.445-447.
- [74] Z.Jako, *Performance Improvement of Differential Chaos Shift Keying Modulation Scheme*, PhD Thesis, Hungary, 2003.
- [75] S.Jiang, J.Liang, Y.Liu, K.Yamazaki, M.Fujishima, *Modeling and Cosimulation of FPGA-Based SVPWM Control for PMSM*, Proceedings of the 31<sup>st</sup> Annual Conference of IEEE Industrial Electronics Society, USA, 2005, pp. 1538-1543.
- [76] S.H.Jin, J.U.Cho, D.R.Lee, J.H.Park, H.S.Kim, C.H.Lee, J.S.Choi, J.W.Jeon, *An FPGA-based Voice Signal Preprocessor for the Real-time Cross-correlation*, Proceedings of the International Conference on Control, Automation and Systems, Korea, 2007, pp.793-797.
- [77] F. Jondral, *Automatic Classification of High Frequency Signals*, Journal Signal Processing, vol. 9, 1985, pp. 177-190.
- [78] W.M.Jones, D.B.Larkins, *Integrating Digital Logic Design and Assembly Programming Using FPGAs in the Classroom*, Proceedings of the 49<sup>th</sup> Southeast Regional Conference, USA, 2011.
- [79] S.T.Karris, *Introduction to Simulink with Engineering Applications*, Orchard Publications, USA, 2006.
- [80] S.T.Karris, *Digital Circuit Analysis and Design with Simulink Modeling and Introduction to CPLDs and FPGAs*, Orchard Publications, USA, 2007.
- [81] K.Kim, A.Polydoros, *Digital modulation classification: the BPSK versus QPSK case*, Proceedings of the IEEE Military Communications Conference, Vol.2, SUA, 1988, pp.431-436.
- [82] N.F. Krasner, *Optimum Detection of Digitally Modulated Signals*, IEEE Transactions on Communications, vol. Com-30, 1982, pp. 885-895.
- [83] G.Kraus, *Introduction to Digital Signal Processing (DSP)*, VHF Communications, 2010, pp.23-37.
- [84] P.Krivić, G.Štimac, *FPGA Implementation of BPSK Modem for Telemetry Systems Operating in Noisy Environments*, Proceedings of the 33<sup>rd</sup> International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 2010, pp.1727-1731.
- [85] A.Krukowski, I.Kale, *Simulink/Matlab-to-VHDL Route for Full-Custom/ FPGA Rapid Prototyping of DSP Algorithms*, Proceedings of the Matlab DSP Conference, Finland, 1999.
- [86] T.Kuwahara, *FPGA-based Reconfigurable On-board Computing Systems for Space Applications*, PhD Thesis, Germany, 2010.
- [87] L.LanJun, Z.Xiaotong, W.Qin, L.Li, *An Underwater Acoustic Spread-Spectrum System Using RA Coding and Band-Pass Signal Acquisition*, Proceedings of the 5<sup>th</sup> International Conference on Wireless Communications, Networking and Mobile Computing, China, 2009, pp. 1-4.
- [88] LeCroy WaveSurfer, *Xs Series Oscilloscope Operator's Manual*, 2008

- [89] W.F.Lee, *VHDL, Coding and Logic Synthesis with Synopsys*, Academic Press, USA, 2000
- [90] T.Y.Lee, C.C.Hu, Y.L.Hsu, C.C.Tsai, R.S.Hsiao, *A Low-Cost GPS Satellite Signal Baseband System Using FPGA Prototyping*, Proceedings of International Computer Symposium, Taiwan, 2006, pp. 133-137.
- [91] F.F. Liedtke, *Computer Simulation of an Automatic Classification Procedure for Digitally Modulated Communication Signals with Unknown Parameters*, Journal Signal Processing, vol. 6, 1984, pp. 311-323.
- [92] Y.Liu, C.Tao, *Feedback Compensation Algorithm for BPSK/QPSK Carrier Synchronization*, Radio Engineering, Vol. 19, No. 1, 2010, pp.149-154.
- [93] J.Ma, *Signal and Image Processing Via Reconfigurable Computing*, Proceedings of the 1<sup>st</sup> Workshop on Information and Systems Technology, USA, 2003.
- [94] U. Madhow, *Fundamentals of Digital Communications*, Cambridge University Press, New York, 2008.
- [95] G.E.Martinez-Torres, J.M.Luna-Rivera, R.E.Balderas-Navarro, *FPGA-based Educational Platform for Wireless Transmission using System Generator*, Proceedings of the 2006 IEEE International Conference on Reconfigurable Computing and FPGAs, Mexico, 2006, pp. 1-9.
- [96] P.I.Martos, J.L.Bonadero, *FPGA-based digital demodulation*, Proceedings of the XII Reunion de Trabajo en Procesamiento de la Informacion y control, Argentina, 2007.
- [97] J.A.Maya, N.A.Casco, P.A.Roncagliolo, J.G.Garcia, *A high data rate BPSK receiver implementation in FPGA for high dynamics applications*, Proceedings of the 2011 VII Southern Conference on Programmable Logic, Spain, 2011, pp.233- 238.
- [98] R.Mehra, S.Devi, *Efficient Hardware Co-Simulation of Down Convertor for Wireless Communication Systems*, International journal of VLSI design & Communication Systems, Vol.1, No.2, 2010, pp.13-21.
- [99] U.Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Array*, Springer, Germany, 2001.
- [100] U.Meyer-Baese, A.Vera, A.Meyer-Baese, M.Pattichis, R.Perry, *Discrete wavelet transform FPGA design using MatLab/Simulink*, in Proceedings of SPIE The International Society For Optical Engineering, Vol. 6247, 2006, pp. 624-703.
- [101] D.Mihhailov, M.Kruus, A.Sudnitson, *FPGA Platform Based Digital Design Education*, Proceedings of the International Conference on Computer Systems and Technologies, Bulgaria, 2008, pp.IV.4-1 – IV.4-6.
- [102] L.Mingfu, L.Xunchao, Q.Guangdi, *Software Radio-Based Universal Digital Modulator Research*, Proceedings of the International Conference on Communication Software and Networks, China, 2009, pp. 821-824.
- [103] D.Mitic, A.Lubl, *Compare the results obtained by numerical computation and application of approximate formulas for the symmetric capacity of Rayleigh fading channel for BPSK and QPSK modulation*, Proceedings of the 33<sup>rd</sup> International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 2010, pp.553-558.
- [104] D.Mitic, A.Lubl, *Selection procedure for modulation using simulation in Matlab*, Proceedings of the 33<sup>rd</sup> International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 2010, pp.547-552.

- 
- [105] S.K.Mitra, *Digital Signal Processing – A Computer-Based Approach*, McGraw Hill, USA, 2001.
- [106] S.Mittal, S.Gupta, S.Dasgupta, *System Generator: The State-of-art FPGA Design tool for DSP Application*, Proceedings of the 3<sup>rd</sup> International Conference on Embedded Systems, Mobile Communication and Computing, India, 2008.
- [107] O.ANava, A.D.Perez, *Acceleration of Fractal Image Compression Using the Hardware-Software Co-design Methodology*, Proceedings of the International Conference on Reconfigurable Computing and FPGAs, Mexico, 2009, pp. 167-171.
- [108] E.Neborovski, V.Marinkovic, M.Katona, *Video quality assessment approach with field programmable gate arrays*, Proceedings of the 33<sup>rd</sup> International Convention on Information and Communication Technology, Electronics and Microelectronics, Croatia, 2010, pp.713-717.
- [109] D.Nguyen, P.Aarabi, A.Sheikholeslumi, *Real-Time Sound Localization using Field Programmable Gate Arrays*, Proceedings of the 2003 IEEE International Conference on Acoustics, Speech and Signal Processing, China, 2003, pp.573-576.
- [110] H.Nguyen, E.Shwedyk, *A first Course in Digital Communications*, Cambridge University Press, New York, 2009.
- [111] K.E.Nolan, L.Doyle, P.Mackenzie, D.O'Mahony, *Modulation scheme classification for 4G software radio wireless networks*, Proceedings of the International Conference on Signal Processing, Pattern Recognition and Applications, Greece, 2002, pp.25-31.
- [112] N.Nowsheen, C.Benson, M.Frater, *Design of a high frequency FPGA acoustic modem for underwater communication*, Proceedings of the IEEE OCEANS, Australia, 2010, pp.1-6.
- [113] K.Paulsson, M.Hübner, J.Becker, *Cost-and Power Optimized FPGA based System Integration: Methodologies and Integration of a Low-Power Capacity-based Measurement Application on Xilinx FPGAs*, Proceedings of the Design, Automation and Test in Europe, Germany, 2008, pp. 50-55.
- [114] R.Pavlik, *Binary PSK/CPFSK and MSK Bandpass Modulation Identifier based on the Complex Shannon Wavelet Transform*, Journal of Electrical Engineering, Vol.56, No. 3-4, 2005, 71-77.
- [115] V.A.Petroni, *Circuit Design with VHDL*, MIT Press, USA, 2004.
- [116] M.Pedzisz, A.Mansour, *Automatic modulation recognition of MPSK signals using constellation rotation and its 4<sup>th</sup> order cumulant*, Digital Signal Processing, vol.15(3), 2005, pp.295-304.
- [117] **S.O.Popescu**, A.S. Gontean, I. Lie, *Comparing FPGA Behavioral Simulation, Post – Route Simulation with real – life experiments*, Proceedings of the 15<sup>th</sup> International Symposium for Design and Technology of Electronics Packages, Hungary, 2009, pp. 75 – 80.
- [118] **S.O.Popescu**, A.S. Gontean, *Virtex – II Timing Simulation vs. Reality*, Telecommunications Forum Telfor, Serbia, 2009, pp. 755 – 758
- [119] **S.O.Popescu**, *Current Stage of Basic Modulation Implemented on the FPGA*, 1<sup>st</sup> Scientific Report, Romania, 2010.
- [120] **S.O.Popescu**, G. Budura, A.S. Gontean, *Review of PSK and QAM – Digital Modulation Techniques on FPGA*, Joint Conferences on Computational Cybernetics and Technical Informatics CONTI, Romania, 2010, pp. 327 -

- [121] **S.O.Popescu**, *Implementation of the BPSK modulation/ demodulation on FPGA*, 2<sup>nd</sup> Scientific Report, Romania, 2010.
- [122] **S.O.Popescu**, A.S.Gontean, F.Alexa, *Improved FPGA-based Detector*, Proceedings of the 6<sup>th</sup> IEEE International Symposium on Applied Computational Intelligence and Informatics, Romania, 2011, pp. 455-458.
- [123] **S.O.Popescu**, A.S.Gontean, G.Budura, *Simulation and Implementation of a BPSK modulator on FPGA*, Proceedings of the 6<sup>th</sup> IEEE International Symposium on Applied Computational Intelligence and Informatics, Romania, 2011, pp. 459-463.
- [124] **S.O.Popescu**, A.S.Gontean, D.Ianchis, *QPSK Modulator on FPGA*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp. 359-364.
- [125] **S.O.Popescu**, A.S.Gontean, D.Ianchis, *Implementation of a QPSK System on FPGA*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp. 365-370.
- [126] D. Ianchis, V. Tiponut, **S. Popescu**, Z. Haraszy, *Improved Collision Detection System Inspired from the Neural Network of the Locust*, Proceedings of the 9<sup>th</sup> IEEE International Symposium on Intelligence Systems and Informatics, Serbia, 2011, pp.211-215.
- [127] **S.O.Popescu**, A.S. Gontean, *Performance comparison of the BPSK and QPSK Modulation Techniques on FPGA*, Proceedings of the 17<sup>th</sup> International Symposium for Design and Technology of Electronics Packages, Romania, 2011, pp. 257 – 260.
- [128] **S.O.Popescu**, A.S. Gontean, G.Budura, *BPSK System on FPGA*, Proceedings of the 10<sup>th</sup> Jubilee International Symposium on Applied Machine Intelligence and Informatics, Slovakia, 2012, pp. 301-306.
- [129] **S.O.Popescu**, A.S. Gontean, G.Budura, *Hardware Co-Simulation of the BPSK and QPSK Systems on FPGA*, submitted at the 11<sup>st</sup> IFAC/IEEE International Conference on Programmable Devices and Embedded Systems, Czech Republic, *in press*.
- [130] B.E.Priyanto, C.L.Law, Y.L.Guan, *Design & implementation of all digital I-Q modulator and demodulator for high speed WLAN in FPGA*, Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and signal Processing, Vol.2, SUA, 2003, pp. 659-662.
- [131] J.G. Proakis, *Digital Communications, 4<sup>th</sup> edition*, McGraw Hill, New York, 2001.
- [132] S.M.Qasim, A.A.Telba, A.Y.AIMazroo, *FPGA Design and Implementation of Matrix Multiplier Architectures for Image and Signal Processing Applications*, International Journal of Computer Science and Network Security, Vol.10 No.2, 2010, pp. 168-176.
- [133] C.Quintans, M.D.Valdes, M.J.Moure, L.Fernandez-Ferreira, E.Mandado, *Digital Electronics Learning System Based on FPGA Applications*, Proceedings of the 35th ASEE/IEEE Frontiers in Education Conference, USA, 2005, pp. S2G 7-12.
- [134] S.Ramakrishnan, G.X.Gao, D.DeLorenzo, T.Walter, P.Enge, *Design and Analysis of Reconfigurable Embedded GNSS Receivers Using Model-Based Design Tools*, Proceedings of the ION GNSS 21<sup>st</sup> International Technical Meeting of the Satellite Division, USA, 2008, pp. 2293-2303.
- [135] S.Reddy, R.Reddy, *Design and FPGA Implementation of Channel Estimation*

- Method and Modulation Technique for MIMO System*, European Journal of Scientific Research, Vol.25 No.2, 2009, pp.257-265
- [136] A.S.Rodriguez, M.C. Mensinger, S.A.In, L.Yufeng, *Model-based software-defined radio(SDR) design using FPGA*, 2011 IEEE International Conference on Electro/Information Technology (EIT), USA, 2011, pp.1-6.
- [137] S.Sabat, K.Dapat, S.Udgata, *FPGA implementation of Entropy based spectrum sensing technique for Cognitive Radio*, International Journal of Recent Trends in Engineering, Vol 2, No. 5, November 2009, pp. 182-187.
- [138] M.Sadeghi, M.Gholami, *Precise time measuring using GPS Satellites "Simulation in Matlab"*, Proceedings of the 7<sup>th</sup> WSEAS International Conference on Telecommunications and Informatics, Turkey, 2008, pp.19-24.
- [139] T.Saidani, D.Dia, W.Elhamzi, M.Atri, R.Tourki, *Hardware Co-simulation For Video Processing Using Xilinx System Generator*, Proceedings of the World Congress on Engineering, Vol. I, 2009, U.K.
- [140] T.Sakla, D.Jain, S.Gautam, *Implementation of Digital QPSK Modulator by using VHDL/Matlab*, International Journal of Engineering and Technology, Vol. 2, Issue 9, 2010, pp. 4827-4831.
- [141] D.Santana-Gomez, I.A.Aguirre-Hernandez, Y.L.Rodriguez-Guarneros, *PSK Digital Modulation DSP Implementation Applied to Software Radio*, Proceedings of the 2006 Electronics, Robotics and Automotive Mechanics Conference, Mexico, 2006, pp. 106-109.
- [142] P.C.Sapiano, J.D.Martin, *Maximum likelihood PSK classifier*, Milcom, 1996, pp.1010-1014.
- [143] S.Sarma, V.K.Agrawal, K.Parameshwaran and S.Udupa, *Multiple Data Rate BPSK Modulation Baseband Processing Scheme for Satellite Data Telemetry*, Proceedings of the 2<sup>nd</sup> National Conference Mathematical Techniques: Emerging Paradigms for Electronics and IT Industries, India, 2008, pp. 136-149.
- [144] R. Schiphorst, F.W. Hoeksema, C. H. Slump, *A Real-Time GPP Software-Defined Radio Testbed for the Physical Layer of Wireless Standards*, Journal on Applied Signal Processing, Vol. 16, 2005, pp. 2664-2672
- [145] D.K.Sharma, A.Mishra, R.Saxena, *Analog&Digital modulation techniques: an overview*, International Journal of Computing Science and Communication Technologies, Vol. 3, Nn. 1, 2010, pp. 551-561.
- [146] C. Shi, J. Hwang, S.McMillan, A. Root, and V. Singh, *A system level resource estimation tool for FPGAs*, Proceedings of 14<sup>th</sup> International Conference on Field Programmable Logic and Application, Belgium, 2004, pp. 424-433.
- [147] J.A.Sills, *Maximum-likelihood modulation classification for PSK/QAM*, in Proceedings of the IEEE MILCOM, 1999, pp. 57-61.
- [148] E.M.Silva-Cruz, G.Jovanovic Dolecek, *Design and simulation of QPSK reconfigurable digital receiver*, Proceedings of the 53<sup>rd</sup> IEEE International Midwest Symposium on Circuits and Systems, USA, 2010, pp. 656 - 659.
- [149] M.Simon, *Bandwidth-Efficient Digital Modulation with Application to Deep-Space Communications*, Issued by the Deep-Space Communications and Navigation Systems Center of Excellence Jet Propulsion Laboratory California Institute of Technology, USA, 2001.

- [150] A.K.Singh, G.Singh, D.S Chauhan, *Software Defined Radio in Wireless Ad-hoc Network*, Proceedings of the SDR '08 Technical Conference and product Exposition, USA, 2008.
- [151] B.Sklar, *A Structured Overview of Digital Communications – a Tutorial Review – Part I*, IEEE Communications Magazine, 1983, pp.4-17.
- [152] D.J.Smith, *VHDL&Verilog Compared&Contrasted – Plus Modeled Example Written in VHDL, Verilog and C*, Proceedings of the 33<sup>rd</sup> Design Automation Conference, USA, 1996, pp. 771-776.
- [153] W.H.Mangione-Smith, B.L.Hutchings, *Configurable Computing: The Road Ahead*, Reconfigurable Architectures Workshop, Switzerland, 1997, pp. 81-96.
- [154] W.H.Mangione-Smith, B.Hutchings, D.Andrews, A.DeHon, C.Ebeling, R.Hartenstein, O.Mencer, J.Morris, K.Palem, V.K.Prasanna, H.A.E.Spaanenburg, *Seeking Solutions in Configurable Computing*, Computer, Vol. 30, No. 12, 1997, pp. 38-43.
- [155] W.Song, J.Zhang, Q.Yao, *Design and Implement of BPSK Modulator and Demodulator Based on Modern DSP Technology*, Proceedings of the 3<sup>rd</sup> IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, China, 2009, pp. 1135-1137.
- [156] W.Song, Q.Yao, *Design and Implement of QPSK Modem based on FPGA*, Proceedings of the 3<sup>rd</sup> IEEE International Conference on Computer Science and Information Technology, China, 2010, pp. 599-601.
- [157] R. Sorace, *Probability of error for BPSK modulation from electrostatic interference*, IEEE Transaction on Aerospace and Electronic Systems, Vol.35, No.4, 1999, pp. 1383–1387.
- [158] K.M.S.Soyjaudah, M.I.Jahmeerbacus, *A modular approach to the design and simulation of digital communication systems*, *International Journal of Electrical Engineering*, 2000, pp. 226-232.
- [159] J.Suris Pietri, *Rapid Radio: Analysis-Based Receiver Deployment*, PhD Thesis, USA, 2009.
- [160] Y.Tachwali, H.Refai, *Implementation of a BPSK Transceiver on Hybrid Software Defined Radio Platforms*, Proceedings of the 3<sup>rd</sup> International Conference on Information and Communication Technologies: From Theory to Applications, Syria, 2008, pp.1-5.
- [161] Tektronix, *Understanding Oscilloscope Bandwidth, Rise Time and Signal Fidelity*, Tektronix, 2002.
- [162] R. Tessier, W. Burlison, *Reconfigurable computing for digital signal processing: A survey*. *Jurnal of VLSI Signal Processing*, 28(1-2), 2001, pp. 7-27.
- [163] T.J.Todman, G.A.Constantinides, S.J.E.Wilton, O.Mencer, W.Luk, P.Y.K.Cheng, *Reconfigurable Computing: Architecture Design Methods and Applications*, IEEE Proceedings: Computers and Digital Techniques, Vol. 152, No.2, 2005, pp. 193-207.
- [164] N. Tredemick, *The case for reconfigurable computing*, *Microprocessor Report*, 10(10), 1996, pp. 25-27.
- [165] J.Villasenor, W.H.Mangione-Smith, *Configurable Computing*, *Scientific American*, 1997, pp. 66-71.
- [166] J.Vuillemin, P.Bertin, D.Roncin, M.Shand, H.Touati, P.Boucard, *Programmable Active Memories: Reconfigurable Systems Come of Age*, *IEEE Transactions on VLSI Systems*, Vol. 4, No. 1, 1996, pp. 56-69.



- 
- [167] R. Wain, I. Bush, M. Guest, M. Deegan, I. Kozin, C. Kitchen, *An overview of FPGAs and FPGA programming; Initial experiences at Daresbury*. Technical report, Computational Science and Engineering Department, CCLRC Caresbury Laboratory, 2006.
- [168] J. Wakerly – *Digital Design. Principles and Practices*, Third Edition, 2006.
- [169] C.S. Weaver, C.A. Cole, R.B. Krumland, M.L. Miller, *The Automatic Classification of Modulation Types by Pattern Recognition*, Stanford Electronics Laboratories, Technical Report No. 1829-2, 1969.
- [170] S. Wilson, *Digital Modulation and Coding*, Prentice Hall, USA, 1996.
- [171] R. Woods, J. McAllister, G. Lightbody, Y. Yi, *FPGA-based Implementation of Signal Processing Systems*, Wiley, U.K., 2008.
- [172] Xilinx Inc., *Efficient shift registers, LFSR counters and long-pseudo-random sequence generators*, Application Note, Xilinx, 1996.
- [173] Xilinx Inc., *PN generators using the SRL macro*, Application Note: Virtex Series, Virtex-II Series and Spartan-II Family, Xilinx, 2004.
- [174] Xilinx Inc., *Xilinx University Program Virtex-II Pro Development System. Hardware Reference Manual*, Xilinx, 2005.
- [175] Xilinx Inc., *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*, Xilinx, 2007.
- [176] Xilinx Inc., *Spartan-3E FPGA Starter Kit Board User Guide*, Xilinx, 2008.
- [177] Xilinx Inc., *System Generator for DSP. User Guide*, Xilinx, 2008.
- [178] Xilinx Inc., *Spartan-3E FPGA Family. Data Sheet*. Xilinx, 2009.
- [179] Xilinx Inc., *Xilinx System Generator v2.1 for Simulink. User Guide*, Xilinx, 2010.
- [180] F. Xiong, *Digital Modulation Techniques*, Artech House, Londra, 2000.
- [181] C. Xu, Y. Yuan, J. Wang, *A New Method for FPGA Development*, International Conference on Information Science and Technology, China, 2011, pp.314-317.
- [182] H. Yamada, Y. Osana, T. Ishimori, T. Ooya, M. Yoshimi, Y. Nishikawa, A. Funahashi, N. Hiroi, H. Amano, Y. Shibata, K. Oguri, *A Modular Approach to Heterogeneous Biochemical Model Simulation on an FPGA*, Proceedings of the International Conference on Reconfigurable Computing and FPGAs, Mexico, 2009, pp. 125-130.
- [183] T. Zahariadis, K. Pramataris, S. Voliotis, N. Zervos, *Digital Signal Processing on a Fully Programmable Platform*, Proceedings of the 4th EVRASIP Conference focused on Video/ Image Processing and Multimedia Communications, Croatia, 2003, pp. 257-263.
- [184] **S.O. Popescu**, A.S. Gontean, G. Budura, *Modern Implementation of a BPSK Modulator on FPGA*, submitted at the International Journal of Advanced Computer Science, unpublished.

## APPENDIX

The present work has attached a DVD where VHDL code and System Generator applications can be found.

The content of the DVD:

1. VHDL code of the BPSK Detector;
2. BPSK Modulator in Simulink and System Generator;
3. BPSK System in Simulink and System Generator;
4. VHDL code of the BPSK Modulator;
5. VHDL code of the BPSK Demodulator;
6. QPSK Modulator in Simulink and System Generator;
7. QPSK System in Simulink and System Generator;
8. VHDL code of the QPSK Modulator;
9. VHDL code of the QPSK Demodulator;
10. Hardware implementations of the BPSK modulator and system (Simulink/System Generator and VHDL code);
11. Hardware implementations of the QPSK modulator and system (Simulink/System Generator and VHDL code).





