Design and Implementation of a FPGA Based Software Defined Radio Using Simulink HDL Coder

Dr. Hikmat N. Abdullah & Hussein A. Hadi*

Received on: 2/12/2009
Accepted on: 30/6/2010

Abstract

This paper presents the design procedure and implementation results of a proposed software defined radio (SDR) using Altera Cyclone II family board. The implementation uses Matlab/Simulink™, Embedded Matlab™ blocks, and Cyclone II development and educational board. The design is first implemented in Matlab/Simulink™ environment. It is then converted to VHDL level using Simulink HDL coder. The design is synthesized and fitted with Quartus II 9.0 Web Edition® software, and downloaded to Altera Cyclone II board. The results show that it is easy to develop and understand the implementation of SDR using programmable logic tools. The paper also presents an efficient design flow of the procedure followed to obtain VHDL netlists that can be downloaded to FPGA boards.

Keywords: FPGA, Embedded Matlab, software defined radio.

تصميم و بناء راديو معرف برمجيا بمصفوفة البوابات المبرمجة الواسعة بأستخدام مشفر لغة الكيان المادي الكتلي

الخلاصة

هذا البحث يعرض خطوات التصميم ونتائج البناء لمنظومة راديو معــرف برمجــا (SDR) باستخدام لوح مصفوفة البوابات المبرمجة الواسعة (FPGA) من عائلة Altera CycloneII. وقــد اســتخدم البنــاء الادوات Matlab/SimulinkTM و Embedded MatlabTM ولــوح CycloneII التعليمي. في البدء تم بناء التصميم باستخدام محيط Matlab/simulinkTM ثم بعد ذلك تم تحويلــه الى شفرة VHDL باستخدام برنامج التحويل Simulink HDL Coder ومــن ثــم تــم تركيــب وملائمة التصميم بما يناسب لوح Altera Cyclone II باستخدام البرنامج Quartus II 9.0 Web Edition® ومن ثم حقنه في اللوح المذكور. النتائج التي تم الحصول عليها بينت بانه مــن الســهل بناء وتطوير منظومة SDR باستخدام الادوات المنطقية المبرمجة. كما قد عرض البحــث مخطــط انسيابي لطريقة تصميم كفوءة بالامكان استخدامها للحصول على شفرات VHDL بالامكان حقنهــا في الواح ال FPGA لتصاميم تم انشاؤها باستخدام Matlab/SimulinkTM.

1. Introduction

The twentieth century saw the explosion of hardware defined radio (HDR) as a means of communicating all forms of audible, visual and machine-generated information over vast distances. Most radios are hardware defined with little or no software control; they are fixed in function for mostly consumer items for

* Engineering College, University of Al- Mustansiryah /Baghdad

https://doi.org/10.30684/etj.28.23.12
2412-0758/University of Technology-Iraq, Baghdad, Iraq
This is an open access article under the CC BY 4.0 license http://creativecommons.org/licenses/by/4.0

broadcast reception. They have a short life and are designed to be discarded and replaced [1]. Over the last few years, analog radio systems are being replaced by digital radio systems for various radio applications in military, civilian and commercial spaces. In addition to this, programmable hardware modules are increasingly being used in digital radio systems at different functional levels.

Commercial wireless communication industry is currently facing problems due to constant evolution of link-layer protocol standards (2.5G, 3G, and 4G), existence of incompatible wireless network technologies in different countries inhibiting deployment of global roaming facilities and problems in rolling-out new services/features due to wide-spread presence of legacy subscriber handsets [2]. The mentioned problems can be solved by using a software defined radio (SDR), which comprises both software and hardware. It uses the reprogrammable ability of field programmable gate arrays (FPGA) or digital signal possessors (DSP) to built an open architecture with software implementation of radio frequency functions such as Modulation/demodulation, coding/ decoding etc. [3].

SDR in a few words is a radio that promises to solve the gap between link-layer protocol standards and provide a quick solution of global roaming problems by building a generic platform that switches its functionalities by software control. In this work, an efficient short cycle design flow is being proposed. With this design flow, the designer is able to implement his design models, which have been originally written in Matlab or simulink, using a FPGA board without the need to learn VHDL or even other FPGA design entries.

Another advantage is that this approach reduces the time required to complete the hardware implementation. It gives the beginner designer, for instance the student, a better and more easy understanding of how different design parts behave using her/his written Matlab code/simulink blocks. However, the automatic translation of Matlab code/simulink blocks to VHDL requires some extra requirements. The written Matlab code/simulink blocks should be implemented the so called fixed point arithmetic notation (Embedded Matlab^{TM} [4]).

2. Generation of VHDL Code for MATLAB-Simulink Models

The algorithms and designs used to define systems are normally modeled using high level software languages like MATLAB, MATLAB-Simulink or C. Their disadvantage is that they normally cannot be used to synthesize real hardware. The Simulink HDL coder [5] is a new tool, which comes with the MATLAB-Simulink software package and can be used to generate hardware description language (HDL) code based on Simulink® models and Stateflow® finite-state machines. The coder brings the Model-Based Design approach into the domain of application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) development. Using the coder, system

PDF created with pdfFactory Pro trial version www.pdffactory.com

architects and designers can spend more time on fine-tuning algorithms and models through rapid prototyping and experimentation and less time on HDL coding. The Simulink HDL coder compatibility checker utility can be run to examine MATLAB-Simulink model semantics and blocks for HDL code generation compatibility. Then the coder can be invoked, using either the command line or the graphical user interface.

The coder generates VHDL or Verilog code that implements the design embodied in the model. Usually, a corresponding test bench also can be generated. The test bench with HDL simulation tools can be used to drive the generated HDL code and evaluate its behavior. The coder generates scripts that automate the process of compiling and simulating the code in these tools. EDA Simulator Link™ MQ, EDA Simulator Link IN or EDA Simulator Link DS software can be used from the MathWorks™ to cosimulate generated HDL entities within a Simulink model.

In this work, the EDA Simulator Link™ MQ is used. Another easy possibility whould be to invoke the ModelSim manually. The test bench feature increases confidence in the correctness of the generated code and saves time on test bench implementation. The design and test process is fully iterative. At any point, the designer can return to the original model, make modifications, and regenerate code. When the design and test phases of the project have been completed successfully, the generated HDL code can be exported easily to synthesis and layout tools for hardware realization. The coder generates synthesis scripts for the Synplify® family of synthesis tools.

The procedure followed to obtain VHDL netlists that can be downloaded to the FPGA board is summarized in the flow chart shown in Fig.1.

2.1 Design of Simulink Models with Blocks Supported by Simulink HDL Coder

Some MATLAB-Simulink blocks, especially those which contain complex functions like encoders/decoders, modulators/ demodulators etc. cannot be converted to VHDL code. To solve this problem, these blocks are redesigned using basic components such that they could be converted to VHDL code. Fig. 2 shows the designed SDR system using MATLAB-Simulink blocks supported by the HDL coder while Figs. 3 and 4 show further details of the blocks in Fig. 2. Fig. 4 shows the transceiver part in Fig. 3 which consists of two branches. Each branch supports a different type of modulation scheme, while the coding scheme used is convolutional code. The control circuit of Fig. 3 can be used to decide which transceiver is on and which is off: when the input 1 of the control circuit is 0, the lower branch will turn on, while the upper branch will turn off. The opposite happens when input 1 is decided as logic one.

The modulators/demodulators in Fig. 4 have been designed using embedded MATLAB functions (m-files) while other blocks are designed by MATLAB-Simulink blocks supported by the Simulink HDL coder. For example, Fig. 5 shows the

PDF created with pdfFactory Pro trial version www.pdffactory.com

implementation of convolutional encoder. It has a rate of 1/2 with constraint length 7 and a code array of 171 and 133 is used [6]. The constraint length denotes the number of shift registers, which the modulo-2 sum of the input data is performed. The rate 1/2 signifies that for every 1 bit input, the encoder will output 2 encoded bits. A Viterbi decoder is used to decode the convolutionally encoded signal by finding an optimal path through all the possible states of the encoder [7]. There are two steps in the decoding process. The first step is to weigh the cost of incoming data against all possible data input combinations. Either a Hamming or Euclidean metric may be used to determine the cost [8]. The second step is to traceback through the trellis and determine the optimal path. The length of the trace through the trellis can be controlled by the traceback length parameter [8].

The constraint length of 7 and the code array 171 and 133 used for decoding are the same as in convolutional encoder. The traceback length parameter, that is, the number of trellis states processed before the decoder makes a decision on a bit, is set to 34. The decoder outputs the data bits which are later grouped accordingly. The following steps have been followed in order to ensure that the redesigned model is suited for HDL code generation.

a- A library of all blocks that are currently supported for HDL code generation is created by constructing models for the blocks in this library.
b- The HDL compatibility of the designed model is checked by generating an HDL code generation check report.

2.2 Set up HDL Coder Configuration

Simulink® HDL Coder™ generates script files for use with HDL simulation and synthesis tools. Script generation is executed automatically when code generation is initiated. By default the Simulink HDL Coder generates script files that are compatible with the Mentor Graphics® ModelSim® HDL simulator and with Synplicity® Synplify® synthesis software. By overriding script generation defaults, Simulink HDL Coder can be programmed to generate scripts for most EDA tools. EDA script generation can be customized via the Simulink HDL Coder GUI, or by setting *makehdl* or *makehdltb* properties at the command line, or in a control file.

In this work, the ModelSim-Altera 6.4a Starter Edition package as simulator and Quartus II 9.0 Web Edition as synthesis software have been used. The default settings of Simulink HDL coder are not compatible with Quartus II (compatible with Synplicity® Synplify® synthesis software as mentioned above), therefore a control file (MATLAB file) is used to change the coder settings to be combatable with Quartus II 9.0 synthesis software. Fig. 6 shows an example control file.

2.3 Setup Model Parameters with HDL Coder

Before generating a VHDL code, some parameters of the model must be set. Rather than doing this manually, the *hdlsetup* command with *set_param* function is used to obtain HDL code

PDF created with pdfFactory Pro trial version www.pdffactory.com

generation quickly and consistently. *hdlsetup* command configures the **Solver** options that are recommended or required by the coder. The *hdlsetup* also configures the model start and stop times (for the generated test benches) and fixed-step size.

The model start and stop times determine the total simulation time. This in turn determines the size of data arrays that are generated to provide stimulus and output data for generated test benches. For the designed model, a computation of 10 seconds of test data does not take a significant amount of time. Computation of sample values for more complex models can be time consuming. In such cases, the total simulation time can be decreased.

2.4 Generating HDL Entities for the Designed System

In this step, the *makehdl* function is used to generate HDL code for each subsystem of the designed system as an independent entity. *makehdl* also generates script files for third-party HDL simulation and synthesis tools. *makehdl* specifies numerous properties that control various features of the generated code. In this work, the defaults for all *makehdl* properties are used. As a result to using *makehdl* command, the following files would be generated.

- **SDR.vhd:** VHDL code. This file contains an entity definition and RTL architecture implementing the SDR.
- **SDR_quartus.tcl:** Quartus synthesis script.
- **SDR_compile.do:** Mentor Graphics ModelSim compilation script (vcom command) to compile the generated VHDL code.
- **SDR_map.txt:** Mapping file. This report file maps generated entities (or modules) to the subsystems that generated them.

2.5 Manual Modification of the Generated VHDL Codes

The generated codes should be studied carefully as it is possible for the designer to change and optimize it according to his needs. However, this step can be passed by designing an efficient MATLAB-Simulink model. In this work, few codes only have been slightly modified.

2.6 Generation of Test Benches for Simulation Purposes

The test bench generation function, *makehdltb*, has been used to generate VHDL test benches. The test bench is designed to drive and verify the operation of system entity that was generated by HDL coder. The generated test bench includes:

a- Stimulus data generated by signal sources connected to the entity under test.

b- Output data generated by the entity under test. During a test bench run, this data is compared to the outputs of the VHDL model, for verification purposes.

c- Clock, reset and clock enable inputs to drive the entity under test.

d- A component instantiation of the entity under test.

e- Code to drive the entity under test and compare its outputs to the expected data.

The test bench and script files generated by *makehdltb* are:

PDF created with pdfFactory Pro trial version www.pdffactory.com

- **SDR_tb.vhd:** VHDL test bench code and generated test and output data.
- **SDR_tb_compile.do:** Mentor Graphics ModelSim compilation script (*vcom* commands). This script compiles and loads both the entity to be tested (SDR.vhd) and the test bench code (SDR_tb.vhd).
- **SDR_tb_sim.do:** Mentor Graphics ModelSim script to initialize the simulator, set up wave window signal displays and run a simulation.

2.7 Exporting VHDL Netlists and Test Benches

After the VHDL netlists and test benches of the SDR are finished, they are exported to Mentor Graphics ModelSim using a compilation script for compilation purposes and to a QuartusII synthesis script for synthesis purposes.

2.8 Verifying Design Functionality using ModelSim tool

The correct functionality of SDR was verified using the Altera/Mentor Graphics ModelSim 6.4a simulation tool. For this purpose, the test bench code was compiled and simulated using the generated compilation and simulation scripts by the HDL coder. The simulation script displays all inputs and outputs of the model (including the reference signals sdr_tb/out1_ref) in Mentor Graphics as waveforms. The simulation results using ModelSim are presented and discussed in this section. Figures 7 through 13 show the waveforms at different positions in the system.

In Fig. 7, the ce_out (testing signal) is high when clk_enable is high and it is low when clk_enable is low, while out1_refwhich is a reference signal that can be used for comparison with the output data. Here one can see some spikes. These spikes have been removed by using delay units at some positions in the viterbi decoder. Figure 8 shows the waveforms of the improved system after adding the delay units mentioned above.

When comparing the output signals of figures 7 and 8, the improvement can be easily recognized. Figures 9 through 13 show the detailed waveforms of the system, i.e. signals at intermediate points. These waveforms show the influence of signal through the system and could aid to verify the right operation of the proposed system.

2.9 Design Synthesis using Quartus II

Design Synthesis is a process that starts from a high level of logic abstraction (typically Verilog or VHDL) and automatically creates a lower level of logic abstraction using a library of primitives. The first step in the synthesis process is compilation. Compilation is the conversion of the high-level VHDL language, which describes the circuit at the Register Transfer Level (RTL), into a netlist at the gate level. The second step is optimization of speed and/or area. This is performed on the gate-level netlist. At this stage, the design can be simulated. Finally, place-and-route (fitter) software will generate the physical layout for a PLD/FPGA chip or will generate the masks for an ASIC [9].

In this work, Quartus II 9.0 software has been used, providing a complete design environment for

PDF created with pdfFactory Pro trial version www.pdffactory.com

system on a programmable chip (SOPC) design, which ensures easy design entry, fast processing, and straightforward device programming. Altera-Cyclone II FPGA family with EP2C35F672C6 board is used as target device for implementation purpose. The **SDR_quartus.tcl** generated by HDL coder was imported to Quartus II design project.

Pin assignments have been added then to the design project for the purpose of applying physical inputs and measuring the physical outputs. Then the project has been compiled and synthesized successfully.

2.10 Downloading the Bit Stream File to the FPGA Board

The synthesis process produces a bit stream file that can be downloaded in the FPGA board. The bit stream file of the SDR was successfully downloaded to the Altera-Cyclone II FPGA family of the EP2C35F672C6 board after installing the necessary drivers on the PC. The test of the physical functionality of the SDR was done by simply interfacing a function generator in order to apply the input data and an oscilloscope to monitor the recovered data. Fig. 14 shows a photo of the implemented system using Cyclone II DE2 kit.

3. Summary of Synthesis Reports

Table 1 shows the summary of Synthesis reports obtained from the Qaurtus II package.

4. Implementation results

After compiling the VHDL code by using Quartus II and downloading the bit streams successfully to Cyclone II DE2 EP2C35F672C6 kit, TTL data from the function generator with a rate of 500 KHz has been applied to the kit (Note, this rate can be expand up to 44.79 MHz) while measuring the output with an oscilloscope. Fig. 15 shows these output data when the input to the control circuit is logic 1.

The distortion in the output waveform of Fig. 15 is referred, as a practical effect, to the spikes shown in the simulation waveforms of Fig. 7 which are discussed in section 2.8. Fig. 16 shows an improved version of the waveforms of Fig. 15 after adding some delay units to the original design.

Conclusions

A baseband SDR system has been successfully developed using the Altera Cyclone II EP2C35F672C6 FPGA development and educational kit. During the implementation stage, the operation of SDR was tested using Altera/Mentor Graphics ModelSim 6.4a.

The hardware implementation results show that SDR module is working as correctly as obtained using both Modelsim and MATLAB-Simulink simulations. For a successful compilation and synthesization using Quartus II, the real values data has to be first changed to fixed point data (e.g. 0.707 to 707).

The Simulink HDL coder does not generate HDL code for all MATLAB-Simulink blocks, so some blocks in the design had to be redesigned using basic operations and elements supported by Simulink HDL coder.

PDF created with pdfFactory Pro trial version www.pdffactory.com

References

[1]. V, “Transaction Level M[1] Burns, P. G., “Software Defined Radio
For 3 G”, Artech House – England, 2003.

[2] “Software-Defined Radio”, Wipro Technologies, 2002.

[3] Grégory, E.N, M.S and François odeling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM” Design, Automation & Test in Europe Conference & Exhibition, 16-20 April 2007

[4] Abdullah, Hikmat N. and Alejandro Valenzuela “A Joint Matlab/FPGA Design of AM Receiver for Teaching Purposes”. EMNT2008 conference, Munich University of Technology, Germany, 2008.

[5] “Simulink® HDL Coder™ 1.5 User’s guide ", the MathWorks, Inc. 2009.

[6] Blockset reference, Xilinx user guide. Retrieved on Sep 18, 2005 http://www.xilinx.com/products /software/sysgen/app_docs/user _guide_Chapter_10 _Section_3_Subsection_61.htm.

[7] Sklar, B. Digital communications: Fundamentals and Applications, 2nd edition. Prentice- Hall, Upper Saddle River, NJ, 2001.

[8] Xilinx User Guide. Retrieved on July 23 http://www.xilinx.com/products/ software/sysgen/app_docs/user_guide.htm..

[9] Pedroni, V. A., “Circuit Design with VHDL”, Massachusetts Institute of Technology (MIT) Press, 2004.

PDF created with pdfFactory Pro trial version www.pdffactory.com

Table (1) Summary of Synthesis Reports

Maximum period 22.326 ns
Maximum frequency: 44.79 MHz
Maximum path delay from the any node: 22.326 ns

Device utilization for EP2C35F672C6

Resource	***Used***	***Available***	***Utilization***
IOs	*7*	*475*	*1%*
Total Logic Elements	*9,654*	*33,216*	*29%*
Total Memory bits	*1,664*	*483,840*	*<1 %*
Total PLLs	*0*	*4*	*0 %*

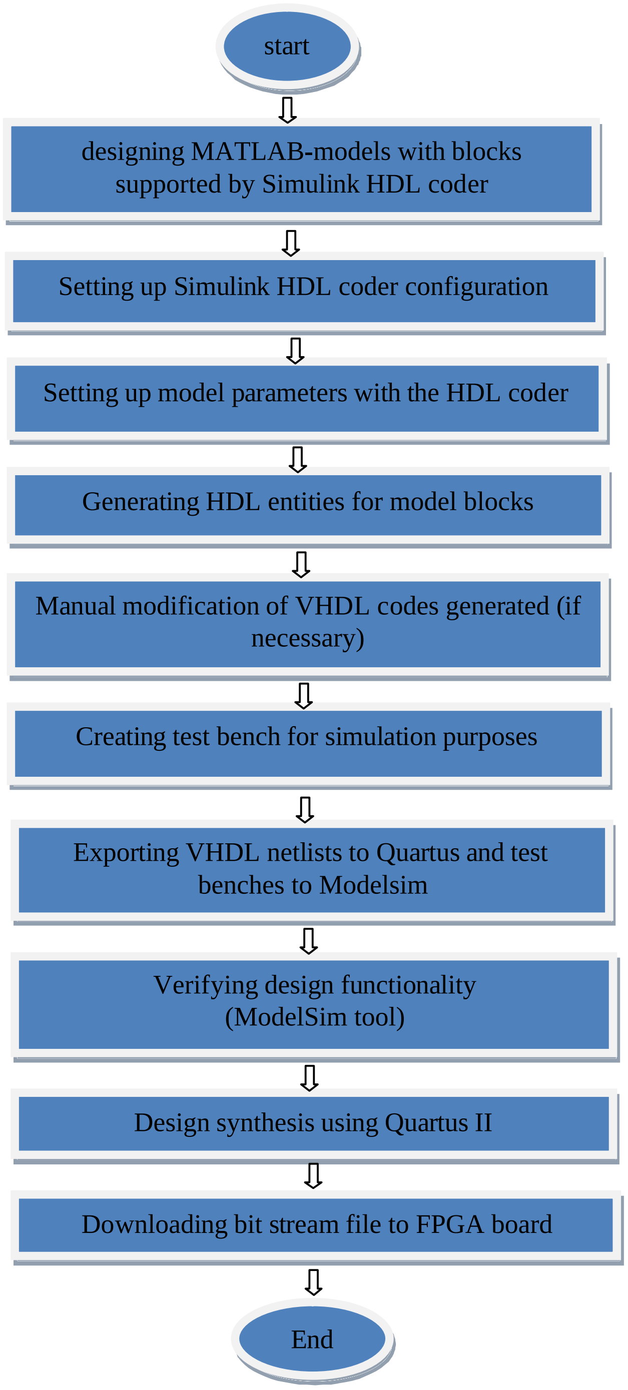

start
designing MATLAB-models with blocks supported by Simulink HDL coder
Setting up Simulink HDL coder configuration
Setting up model parameters with the HDL coder
Generating HDL entities for model blocks
Manual modification of VHDL codes generated (if necessary)
Creating test bench for simulation purposes
Exporting VHDL netlists to Quartus and test benches to Modelsim
Verifying design functionality (ModelSim tool)
Design synthesis using Quartus II
Downloading bit stream file to FPGA board
End

Figure (1) Design flow for realizing MATLAB-Simulink models using FPG A Boards

PDF created with pdfFactory Pro trial version www.pdffactory.com

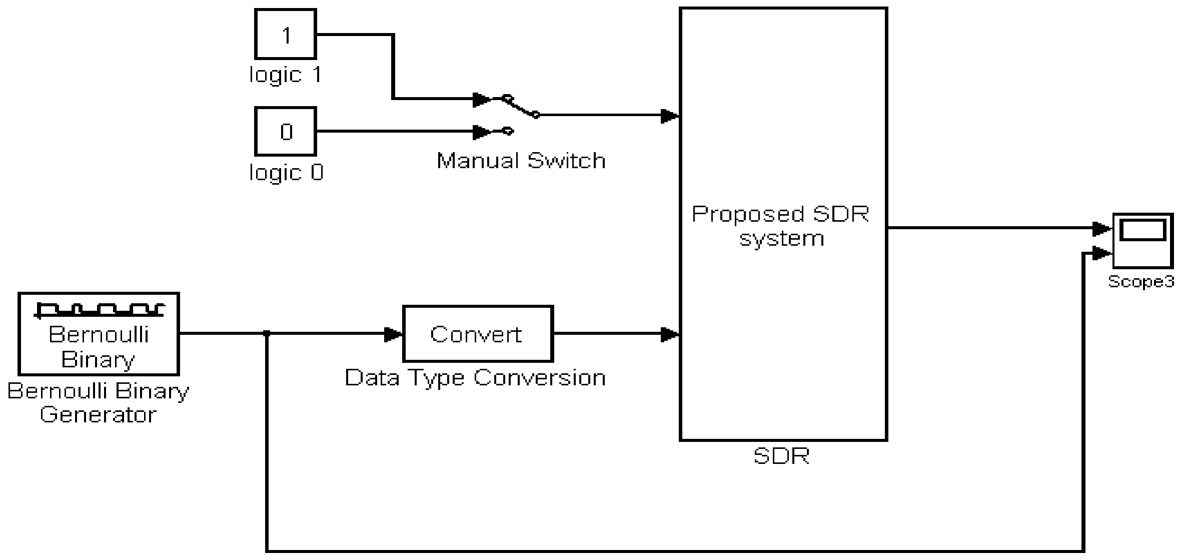


Figure (2) the designed SDR receiver using MATLAB-Simulink blocks supported By Simulink HDL coder

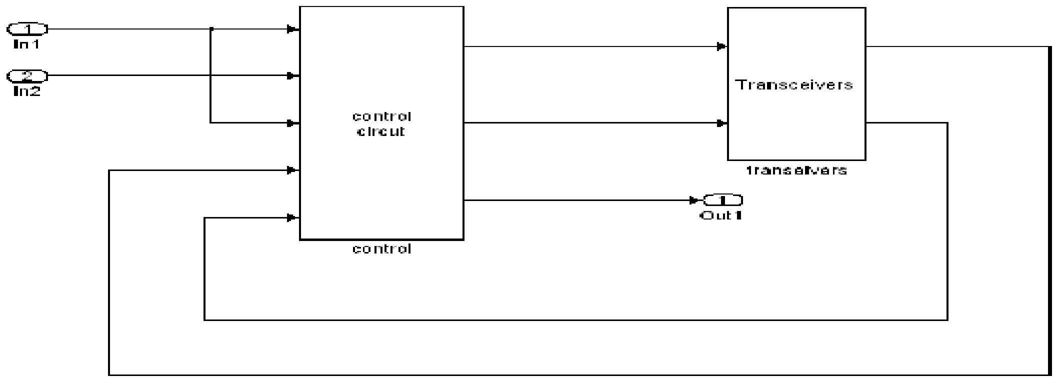


Figure (3) the details of SDR block in Figure (2)

PDF created with pdfFactory Pro trial version www.pdffactory.com

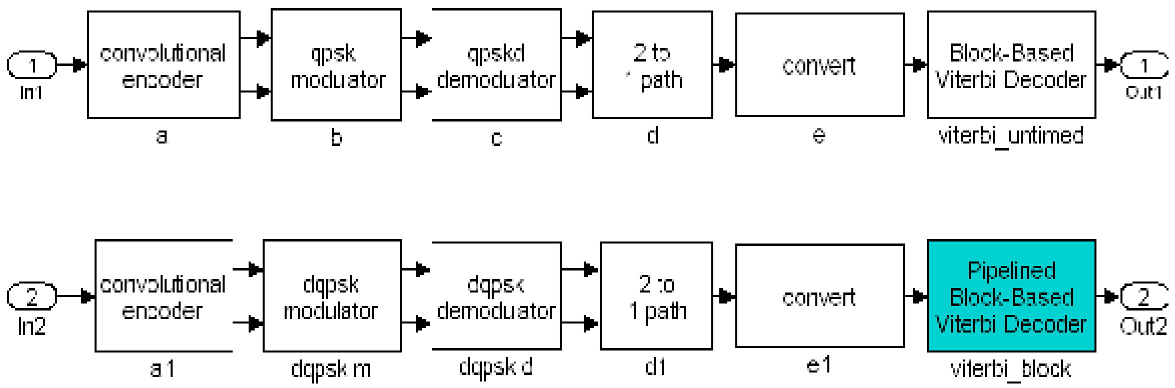


Figure (4) further detail of the transceiver construction

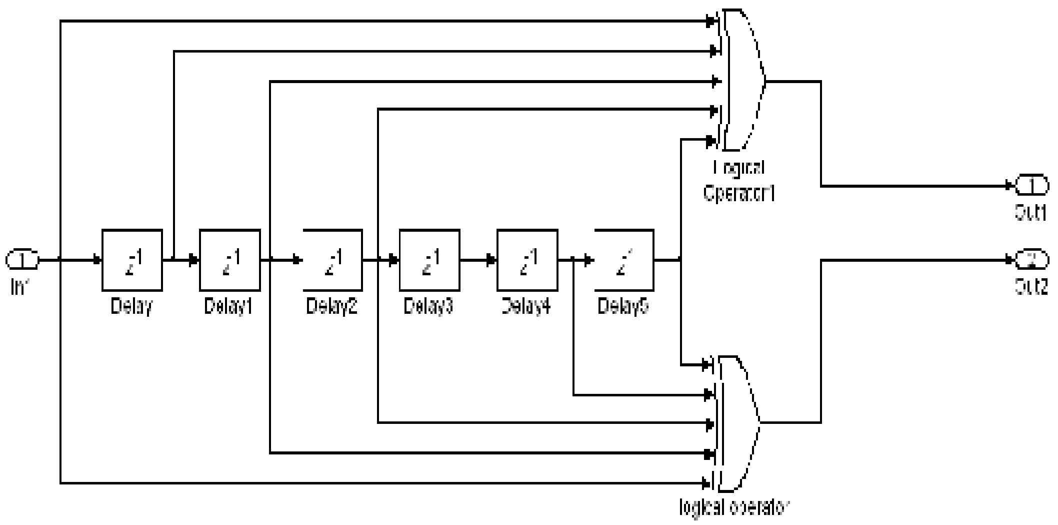
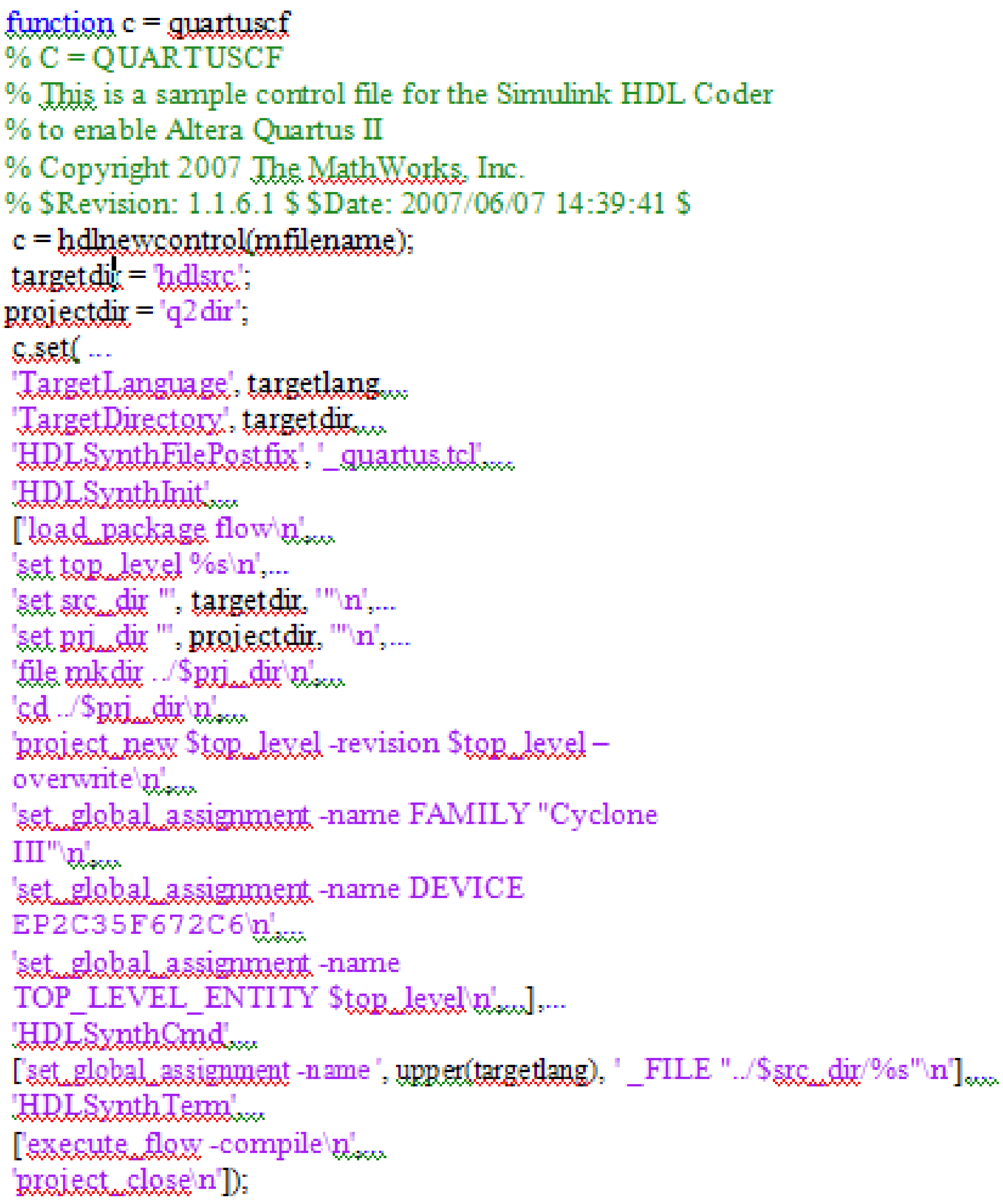


Figure (5) Implementation of the convolutional encoder using MATLAB-Simulink Blocks supported by HDL coder

PDF created with pdfFactory Pro trial version www.pdffactory.com



```
function c = quartuscf
% C = QUARTUSCF
% This is a sample control file for the Simulink HDL Coder
% to enable Altera Quartus II
% Copyright 2007 The MathWorks, Inc.
% $Revision: 1.1.6.1 $ $Date: 2007/06/07 14:39:41 $
 c = hdlnewcontrol(mfilename);
 targetdir = 'hdlsrc';
projectdir = 'q2dir';
 c.set( ...
 'TargetLanguage', targetlang,...
 'TargetDirectory', targetdir,...
 'HDLSynthFilePostfix', '_quartus.tcl',...
 'HDLSynthInit',...
 ['load_package flow\n',...
 'set top_level %s\n',...
 'set src_dir "', targetdir, '"\n',...
 'set prj_dir "', projectdir, '"\n',...
 'file mkdir ../$prj_dir\n',...
 'cd ../$prj_dir\n',...
 'project_new $top_level -revision $top_level –
 overwrite\n',...
 'set_global_assignment -name FAMILY "Cyclone
 III"\n',...
 'set_global_assignment -name DEVICE
 EP2C35F672C6\n',...
 'set_global_assignment -name
 TOP_LEVEL_ENTITY $top_level\n',...],...
 'HDLSynthCmd',...
 ['set_global_assignment -name ', upper(targetlang), ' _FILE "../$src_dir/%s"\n'],...
 'HDLSynthTerm',...
 ['execute_flow -compile\n',...
 'project_close\n']);
```

Figure (6) Example MATLAB control file used to change HDL coder configuration

PDF created with pdfFactory Pro trial version www.pdffactory.com

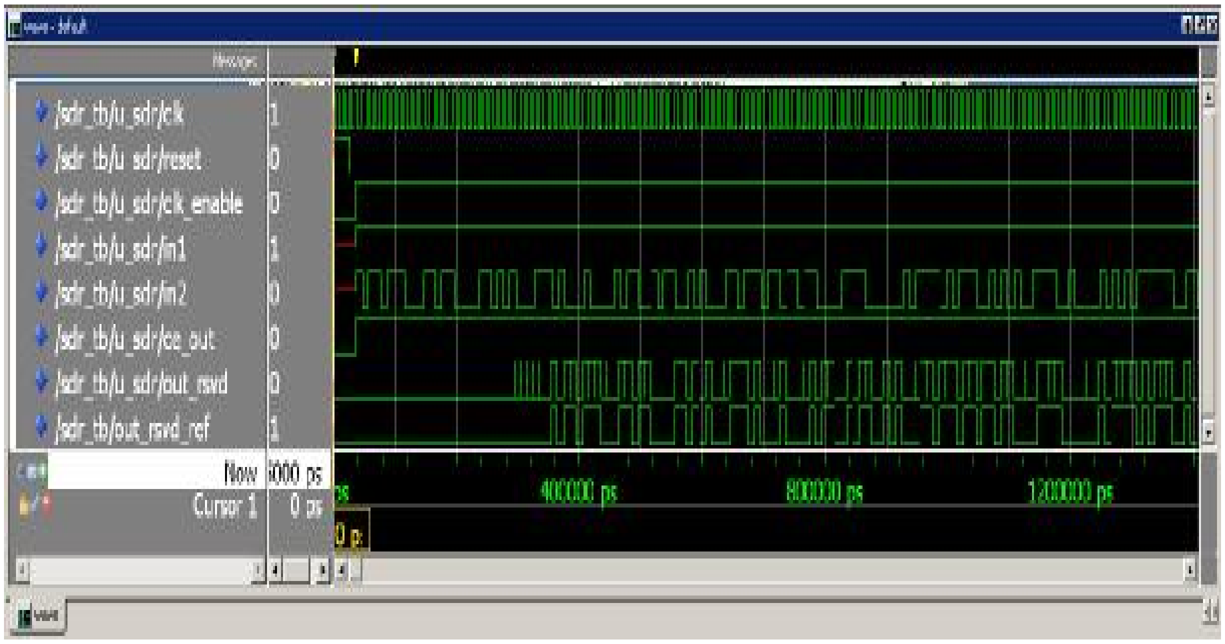


Figure (7) the input and output waveforms when the input to control switch is logic 1

PDF created with pdfFactory Pro trial version www.pdffactory.com

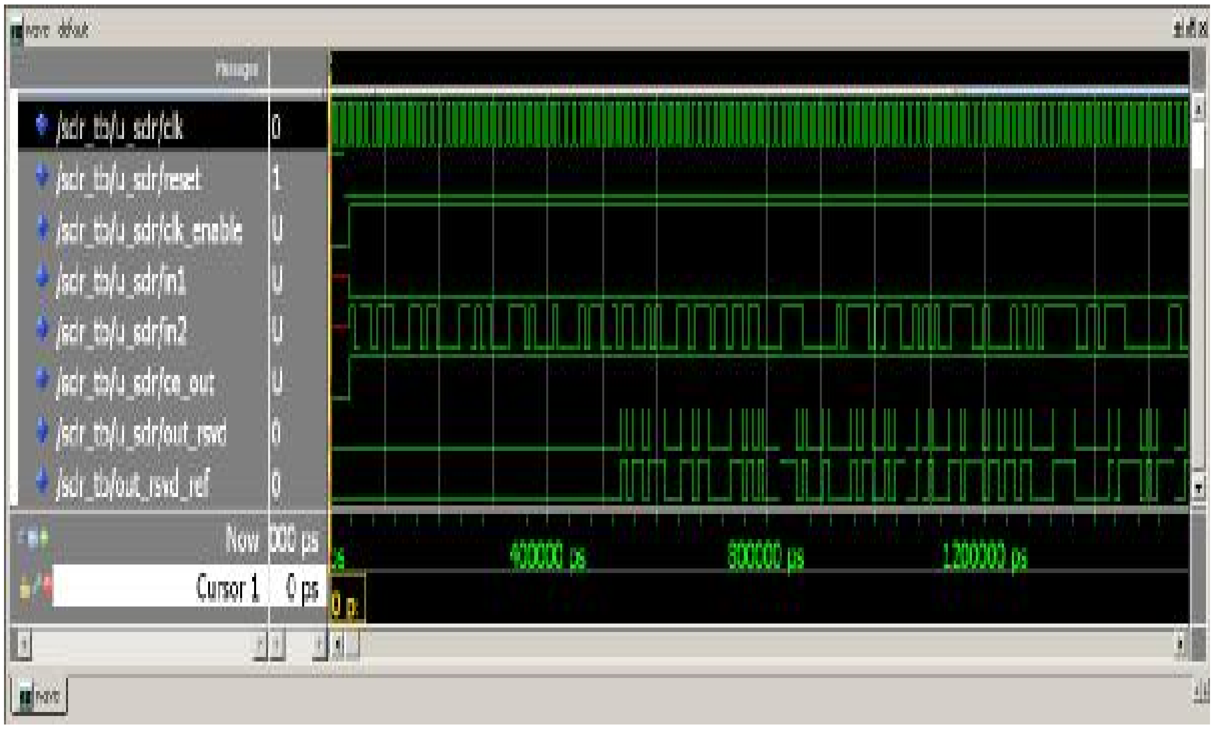


Figure (8) the input and output waveforms of the improved SDR system when the input to the control switch is logic 0

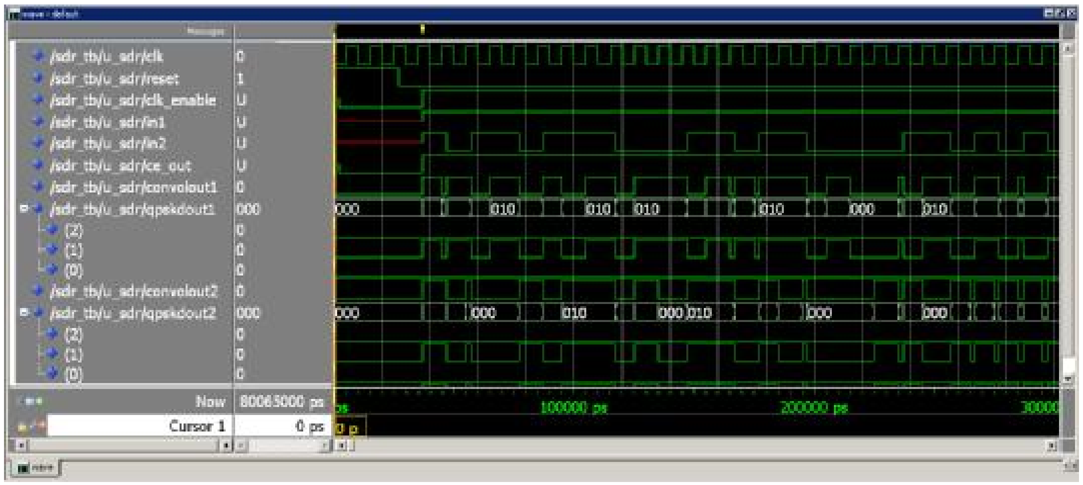


Figure (9) the convolution encoder and QPSK demodulator outputs

PDF created with pdfFactory Pro trial version www.pdffactory.com

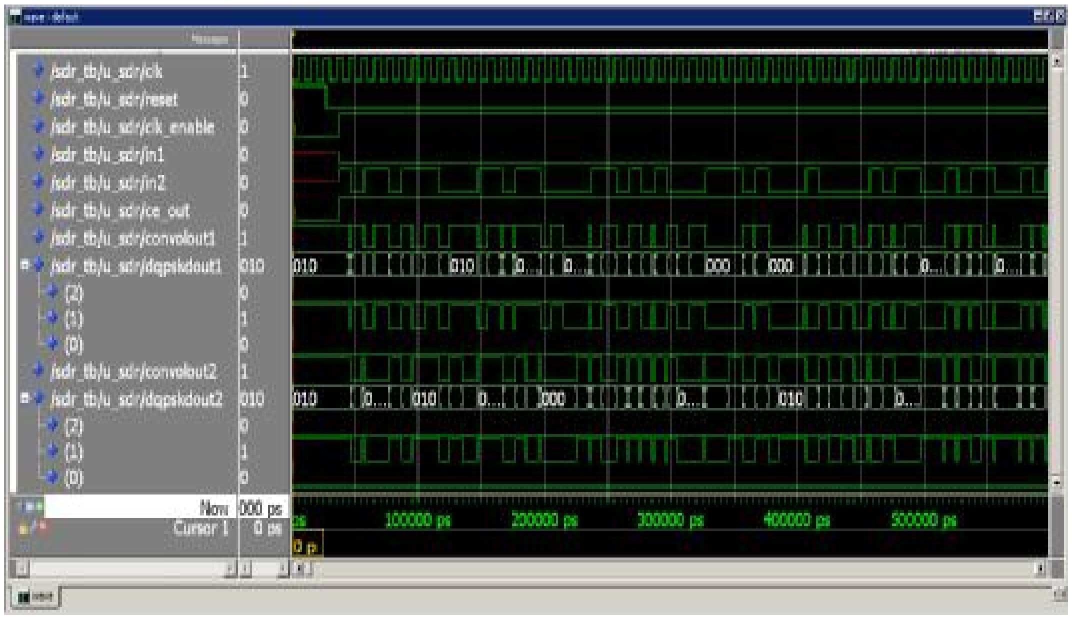


Figure (10) the convolutional encoder and DQPSK demodulator outputs

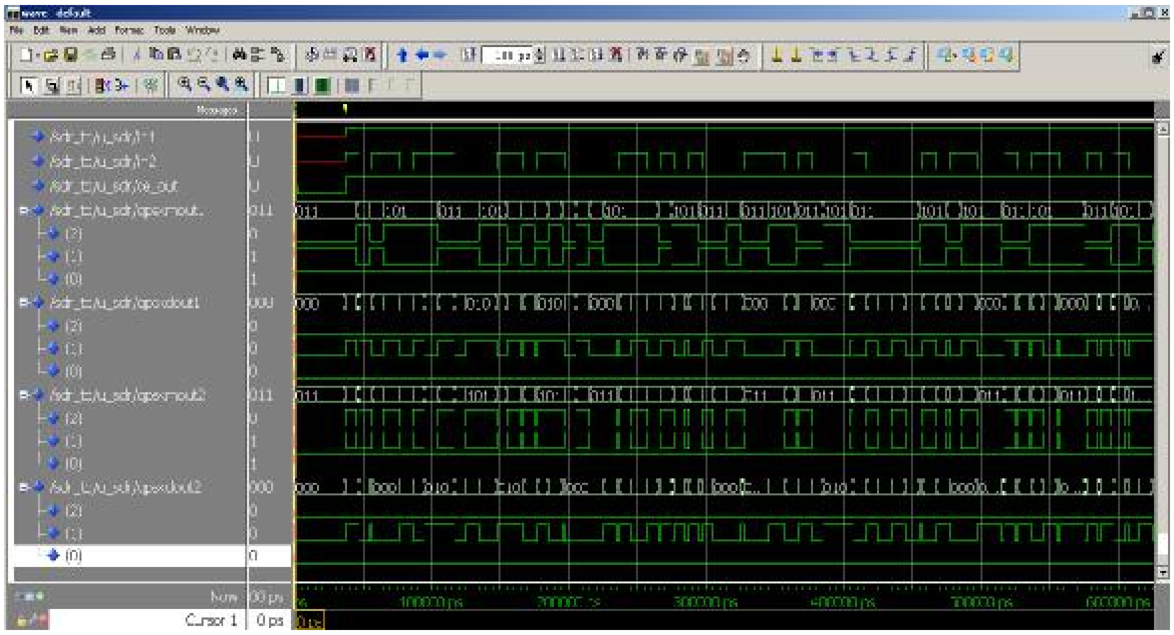


Figure (11) The QPSK modulator and demodulator outputs

PDF created with pdfFactory Pro trial version www.pdffactory.com

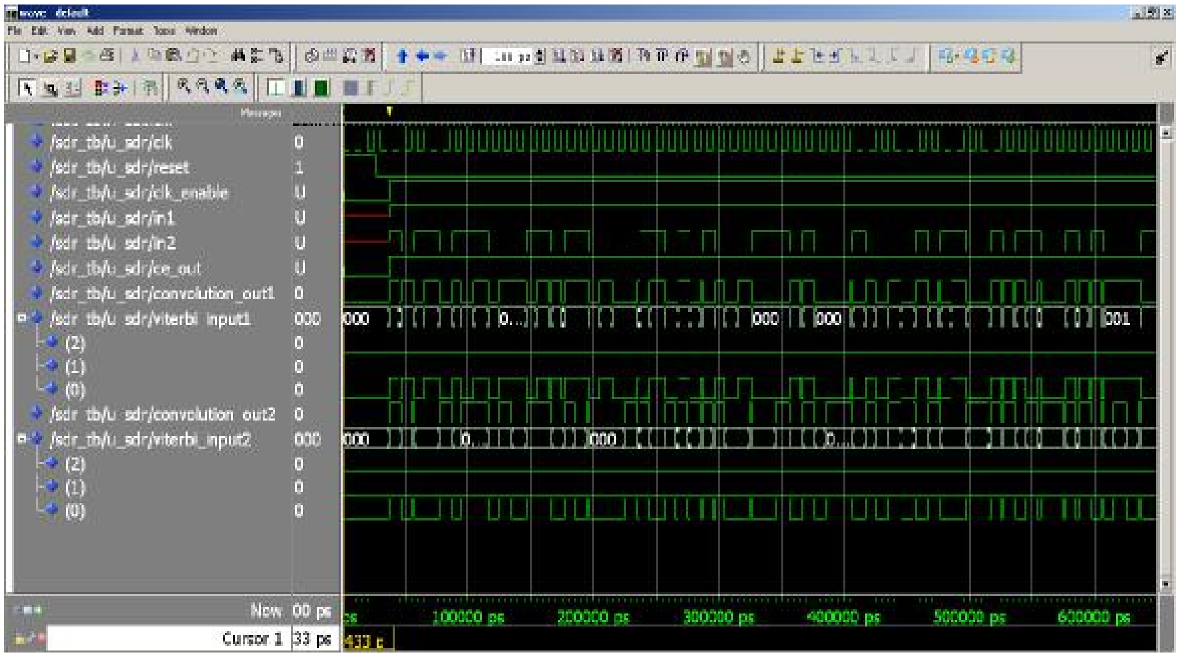


Figure (12) The DQPSK modulator and demodulator outputs

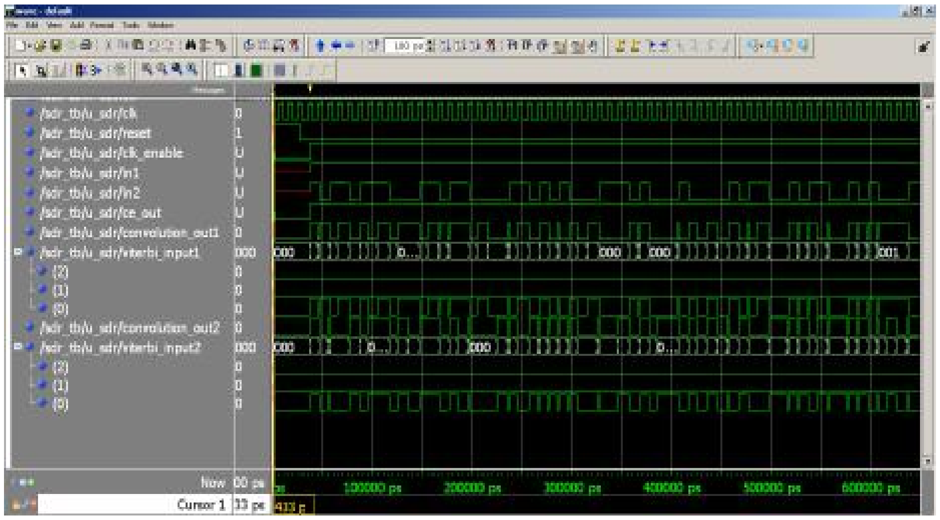

/sdr_tb/u_sdr/clk 0
/sdr_tb/u_sdr/reset 1
/sdr_tb/u_sdr/clk_enable U
/sdr_tb/u_sdr/in1 U
/sdr_tb/u_sdr/in2 U
/sdr_tb/u_sdr/ce_out U
/sdr_tb/u_sdr/convolution_out1 0
/sdr_tb/u_sdr/viterbi_input1 000
/sdr_tb/u_sdr/convolution_out2 0
/sdr_tb/u_sdr/viterbi_input2 000
Now 00 ps
Cursor 1 33 ps

Figure (13) the convolution encoder outputs and viterbi decoder inputs

PDF created with pdfFactory Pro trial version www.pdffactory.com

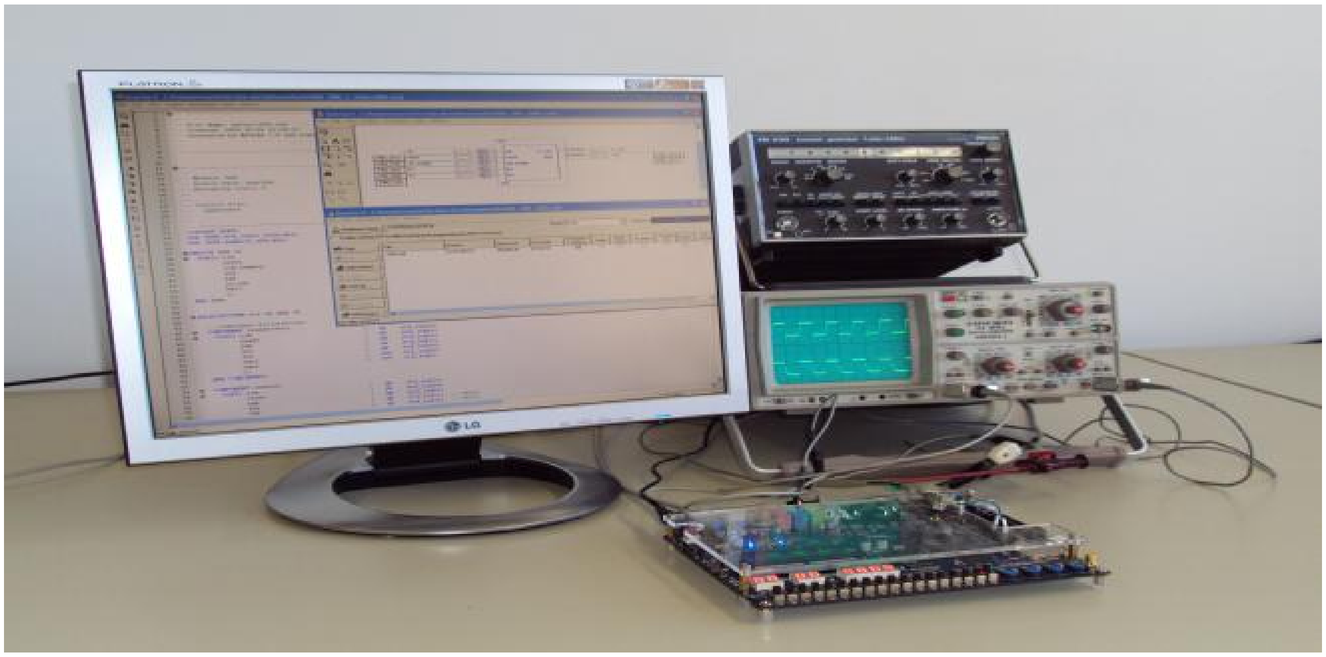


Figure (14) the implementation of proposed SDR system using Cyclone II DE2 kit

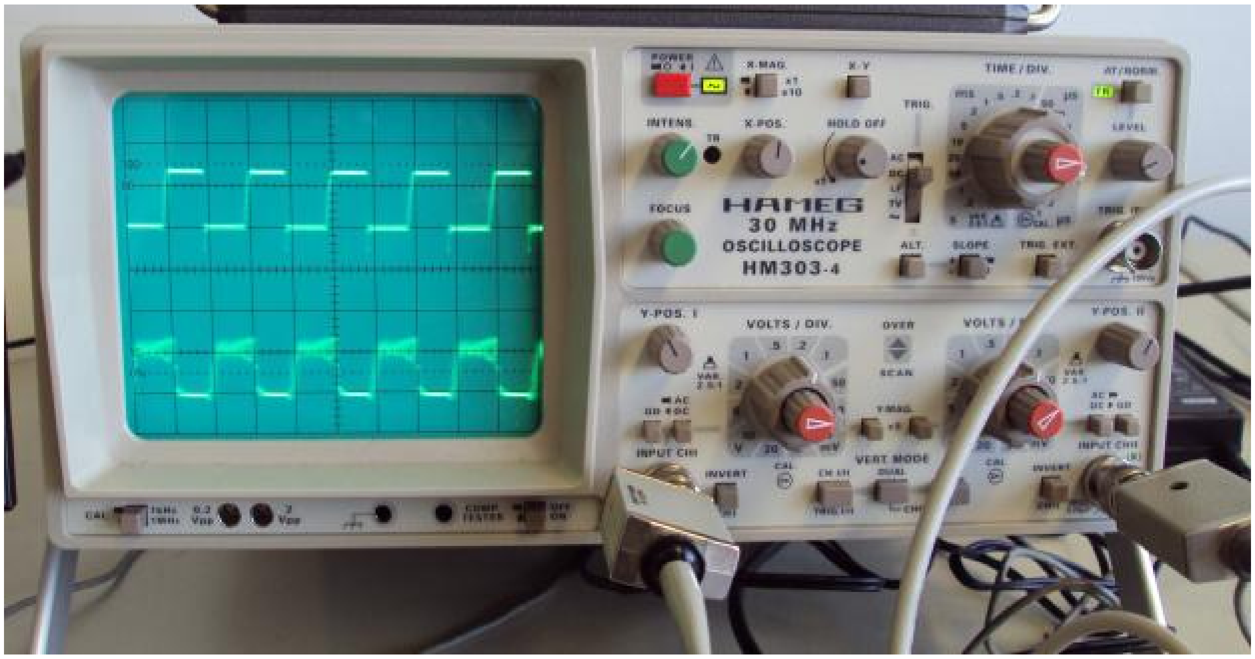


Figure (15) the output signal of DE2 development kit

PDF created with pdfFactory Pro trial version www.pdffactory.com

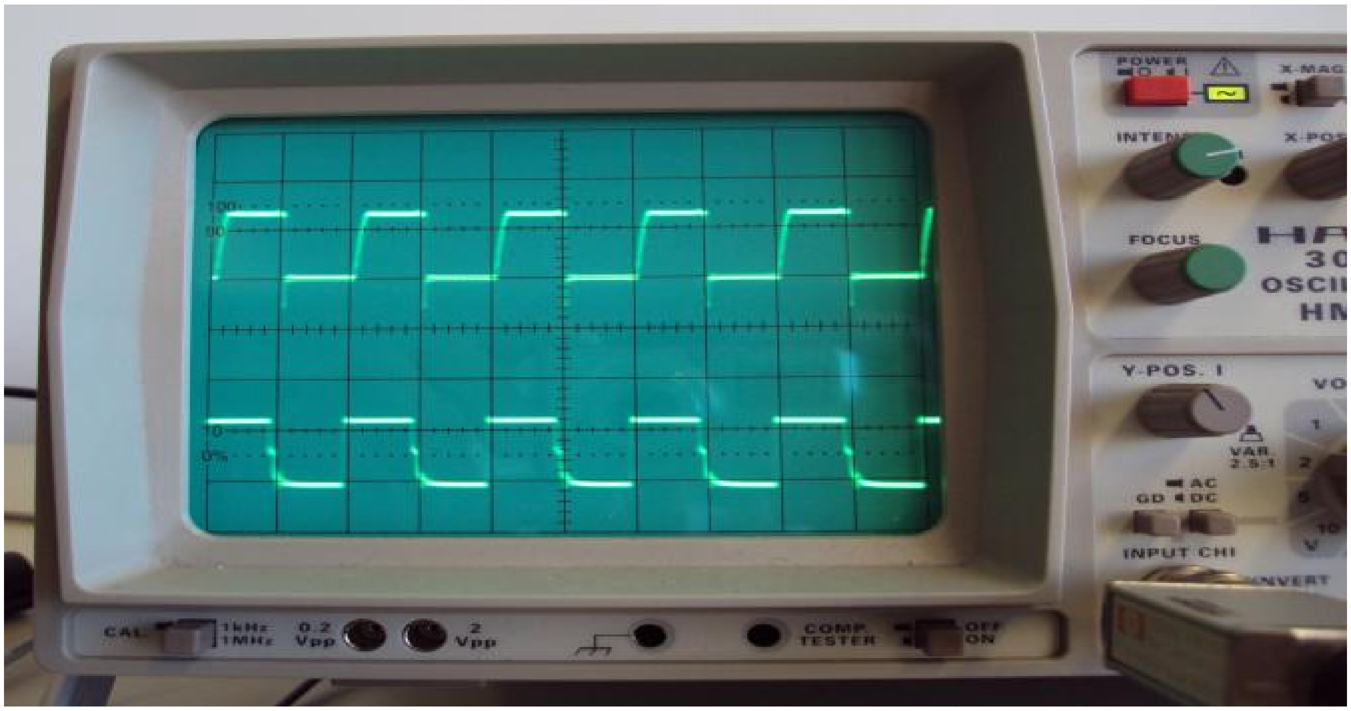


Figure (16) the output from the DE2 development kit

PDF created with pdfFactory Pro trial version www.pdffactory.com