

# DESIGN AND IMPLEMENTATION OF FPGA BASED SOFTWARE DEFINED RADIO USING SIMULINK HDL CODER

**Dr. Hikmat N. Abdullah**  
[hikmat\\_04@yahoo.com](mailto:hikmat_04@yahoo.com)

**B.Sc. Hussein A. Hadi**  
[Hussein.ali\\_2007@yahoo.com](mailto:Hussein.ali_2007@yahoo.com)

**University of Al-Mustansiryah, College of Engineering, Electrical  
Engineering Department, Baghdad-Iraq.**

## 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 has 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.

## 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 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 solution of above big problems can be solved by using the software defined radio (SDR), which comprised of both software and hardware, it use a reprogrammable ability of field programmable gate array (FPGA) or digital signal processor (DSP) to built an open architecture with software implementation of radio frequencies 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 generic platform that switches its functionalities by using software control. In this work, an efficient short cycle design flow has been proposed. With this design flow, the designer could implement his design models originally written as Matlab codes or simulink blocks using FPGA board without the need to learn VHDL or even other FPGA design entries. As well as, this approach reduces the time required to complete the hardware implementation. It will give the beginner designer, for instance the student, a better and easy understanding of how different design parts behave using his/her written Matlab codes/simulink blocks. However, the automatic translation of Matlab code/simulink blocks to VHDL one requires extra requirements. The written Matlab code/simulink blocks should take into ahead what is so called fixed point arithmetic notations (Embedded Matlab™ [4]).

## **2. Generation of VHDL Codes 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. But these designs could not be suited to real hardware. Simulink HDL coder [5] is a new tool, which comes with 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 architects and designers can spend more time on fine-tuning algorithms and models through rapid prototyping and experimentation and less time on HDL coding. Simulink HDL coder compatibility checker utility can be run to examine MATLAB-Simulink model semantics and blocks for HDL code generation compatibility, then by invoking the coder, 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 your 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 are used but in another easily way which can be followed by invoke the ModelSim manually. The test bench feature increases confidence in the correctness of the generated code and saves time spent 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, easily the generated HDL code can be exported 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 could be downloaded to

FPGA boards could be summarized in the flow chart shown in Fig.1.

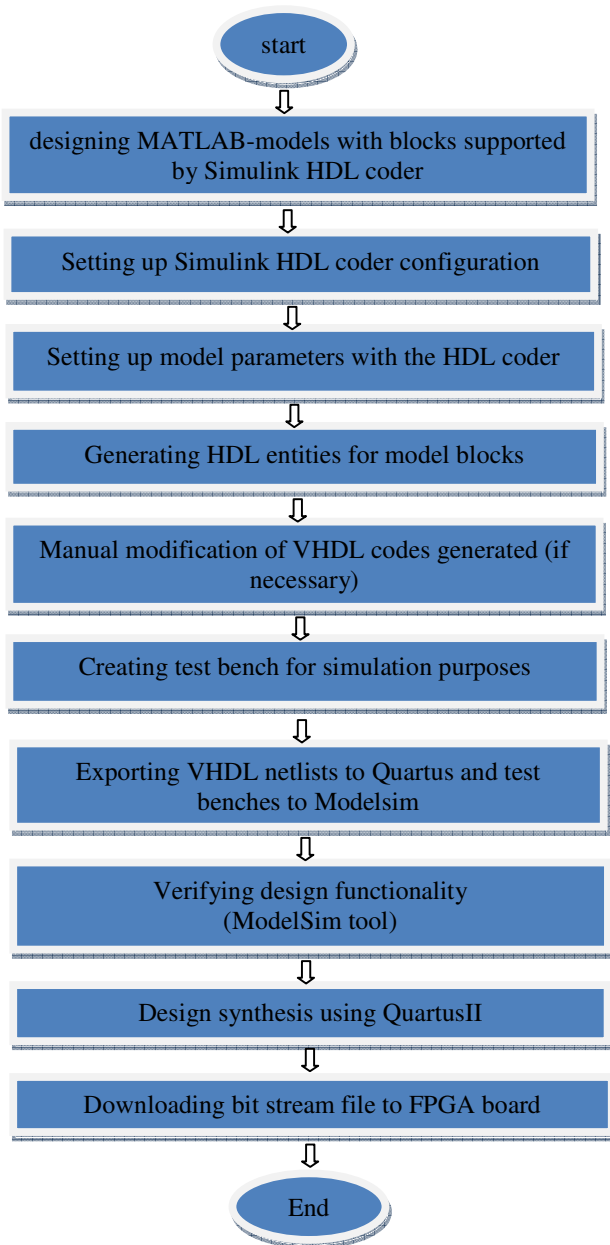


Fig.1 design flow for realizing MATLAB-Simulink models using FPGA boards.

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

Some MATLAB-Simulink blocks, especially those contain complex functions like encoders/decoders, modulators/demodulators, ..etc. could not be converted to VHDL codes. To solve this problem, these blocks are redesigned using their basic components such that it could be converted to VHDL codes.

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.5 shows that the transceiver part in Fig.3 which consists of two branches, each branch support 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 the other is off: when the input1 of the control circuit is 0, the lower branch will turns on while the upper branch will turns off. The opposite thing happend when input1 is decided as logic one.

The modulators/demodulators in Fig.4 have been designed using embedded MATLAB functions (m-files) while other blocks designed by MATLAB-Simulink blocks supported by Simulink HDL coder. For example, Fig.5 shows the implementation of convolutional encoder. In Fig.5, a convolutional encoder of rate 1/2 with constraint length 7 and code array 171 and 133 is used [6]. The constraint length denotes the number of shift registers over, 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. 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

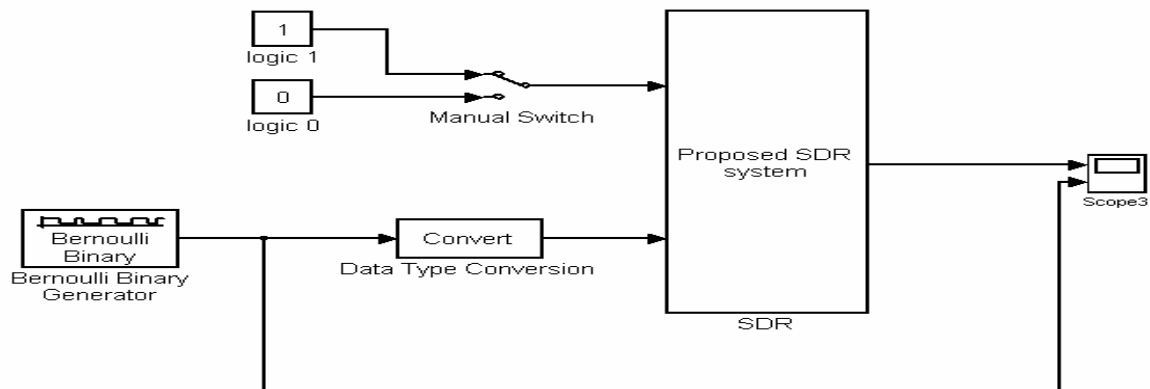


Fig.2 The designed SDR receiver using MATLAB-Simulink blocks supported by Simulink HDL coder.

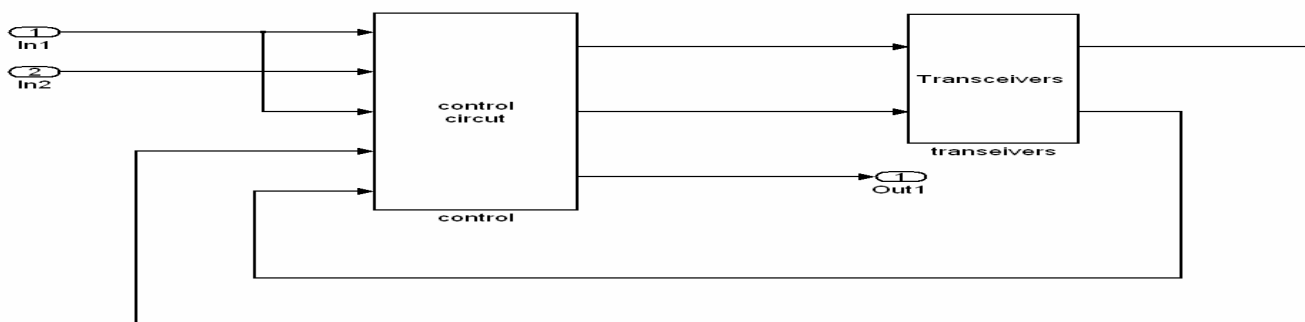


Fig.3 The details of SDR block in Fig.2

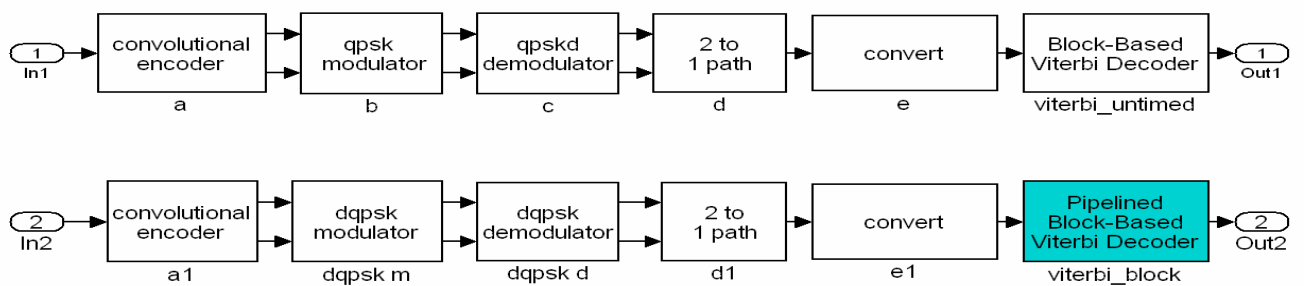


Fig.4 further detail of the transceiver construction

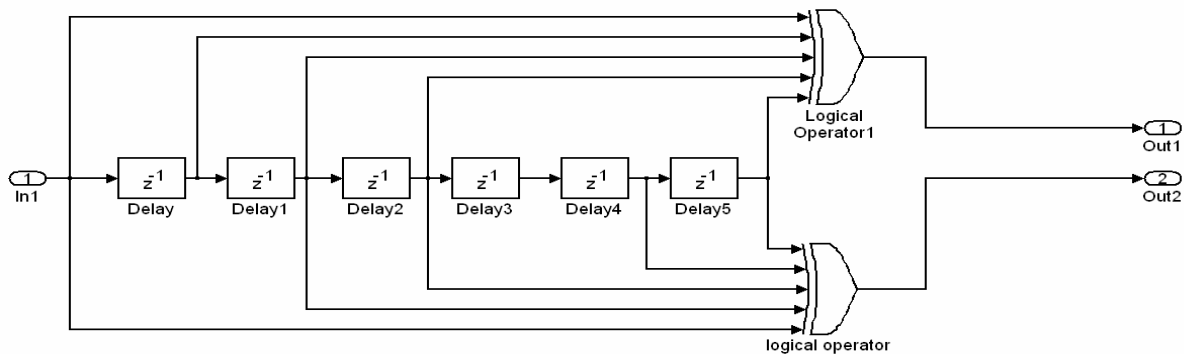


Fig.5 Implementation of the convolutional encoder using MATLAB-Simulink blocks supported by HDL coder.

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 to 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<sup>®</sup> HDL Coder<sup>™</sup> generates script files for use with HDL simulation and synthesis tools. Script generation is executed automatically when code generation is initiated. By default, Simulink HDL Coder generates script files that are compatible

with the Mentor Graphics<sup>®</sup> ModelSim<sup>®</sup> HDL simulator and with Synplicity<sup>®</sup> Synplify<sup>®</sup> 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 *makehdlb* 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<sup>®</sup> Synplify<sup>®</sup> synthesis software as mentioned above), therefore a control file (MATLAB file) is used to change the coder settings to be compatible 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 generation quickly and consistently. *hdlsetup* command

```

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']);

```

Fig.6 Example MATLAB control file used to change HDL coder configuration

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* can specify 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. It is possible to be changed according to what the designer need. 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:

- **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 SDR become ready, they exported to Mentor Graphics ModelSim compilation script for compilation purposes and to QuartusII synthesis script for synthesis purposes.

## 2.8 Verifying Design Functionality using ModelSim tool:

The correct functionality of SDR has been verified using Altera/Mentor Graphics ModelSim 6.4a simulation tool. For this purpose, the test bench codes are compiled and simulated using the generated compilation and simulation scripts by the HDL coder. The simulation script displays all inputs and outputs in the model (including the reference signals *sdr\_tb/out1\_ref*) in the Mentor Graphics as waveforms. The simulation results using ModelSim tool would be presented and discussed in this section. Figures 7 through 13 show the waveforms at many 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\_ref* which is a reference signal and can be used for comparison with output data. In figure 7 some spikes can be seen. These spikes have been removed by using delay units at some positions in the viterbi decoder. Figure 8 shows the waveforms of 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

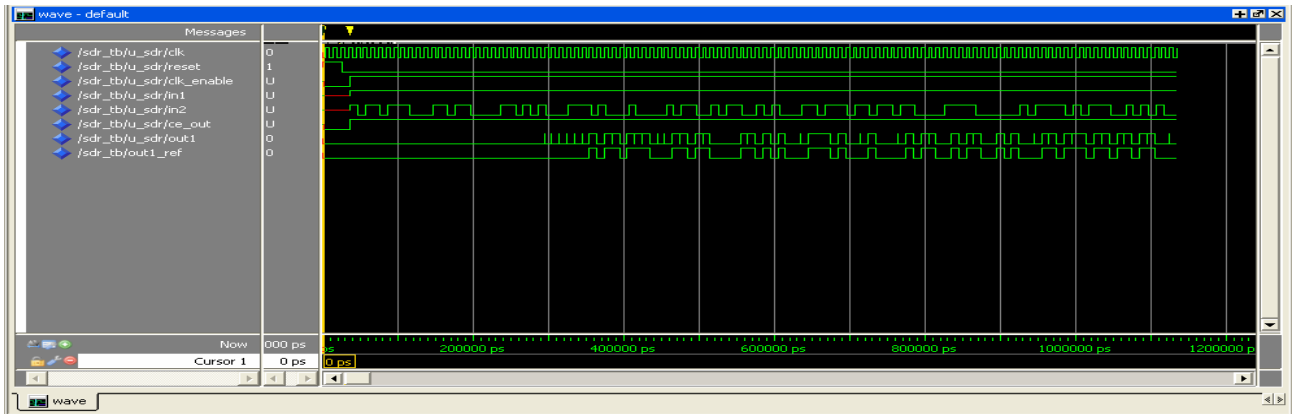


Fig 7 the input and output waveforms when the input to control switch is logic 1.

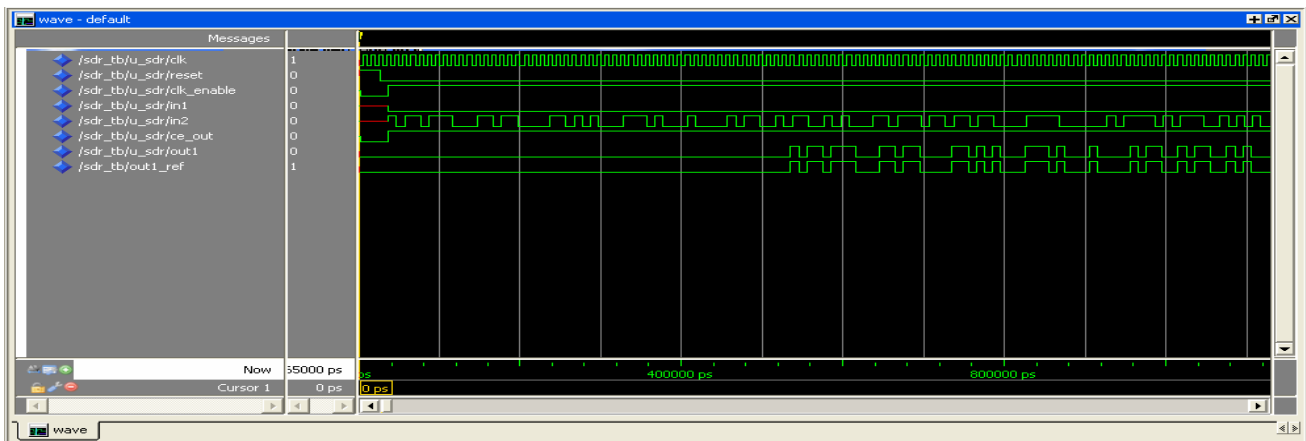


Fig 8 The input and output waveforms of the improved SDR system when the input to the control switch is logic 0.

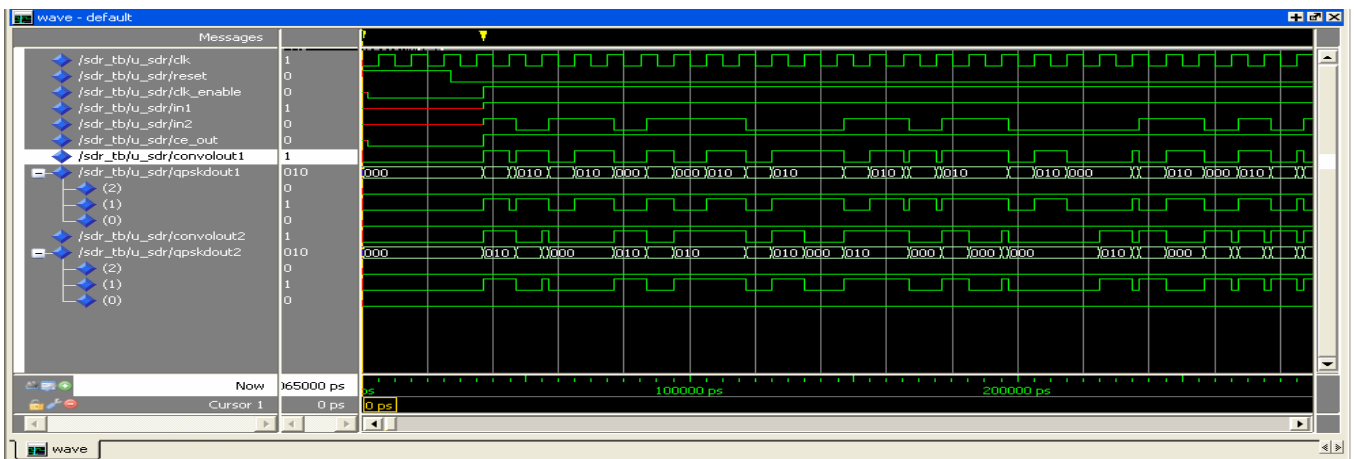


Fig.9 the convolution encoder and QPSK demodulator outputs.



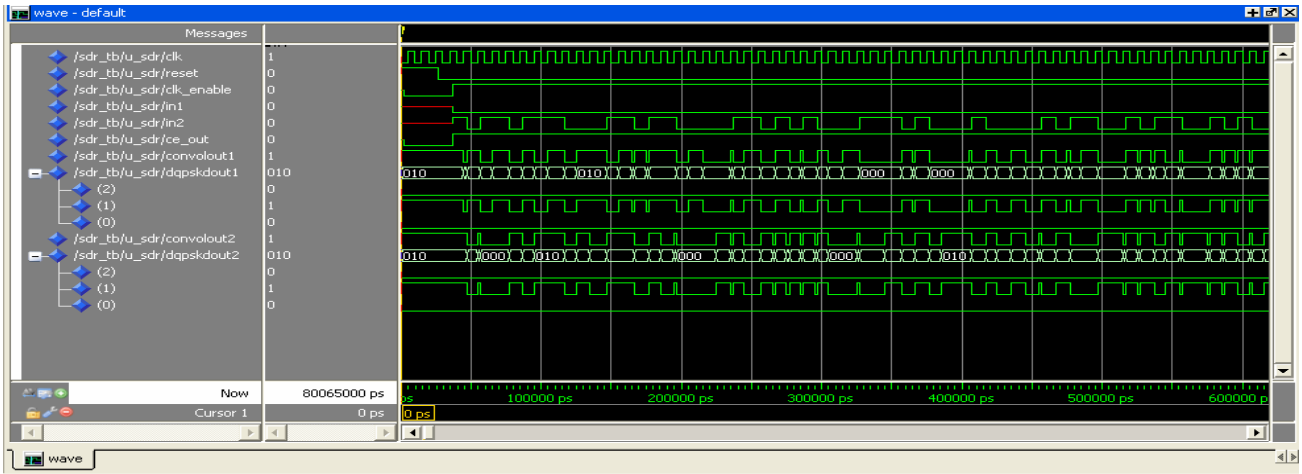


Fig.10 the convolutional encoder and DQPSK demodulator outputs.

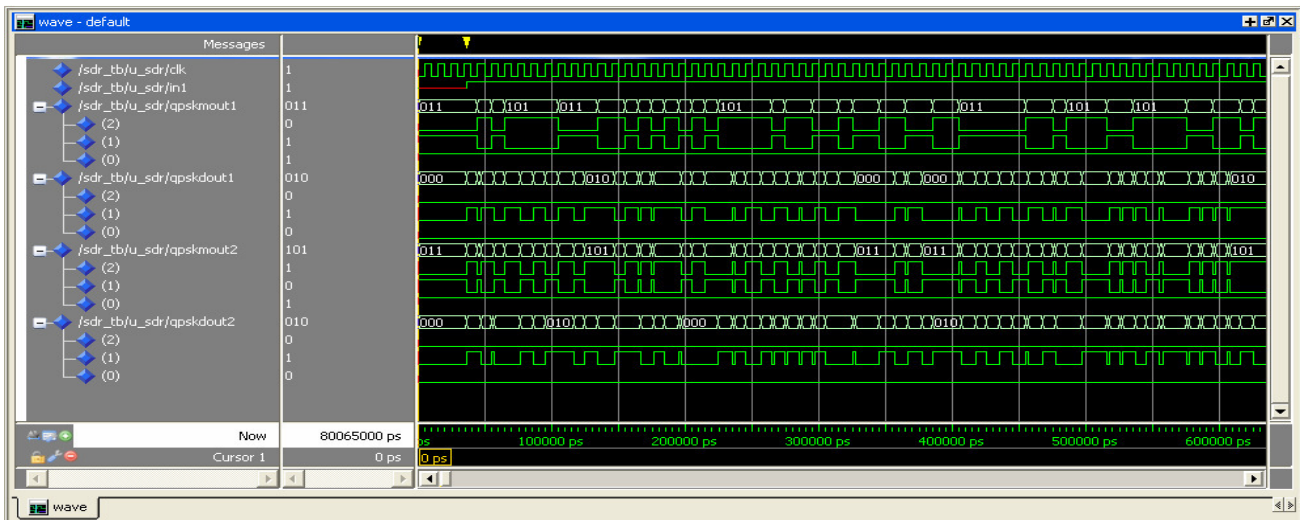


Fig.11 the QPSK modulator and demodulator outputs.

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, which is performed on the gate-level netlist for speed or for area. 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 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 has been 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.

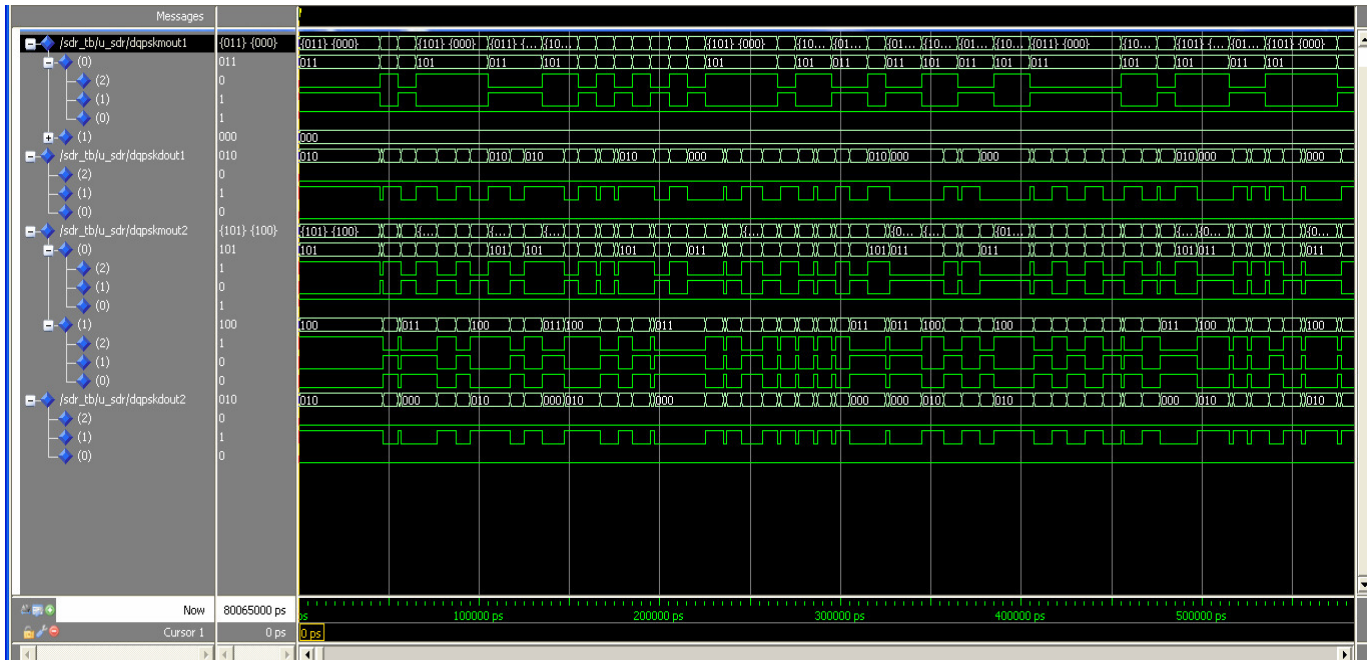


Fig.12 the DQPSK modulator and demodulator outputs.

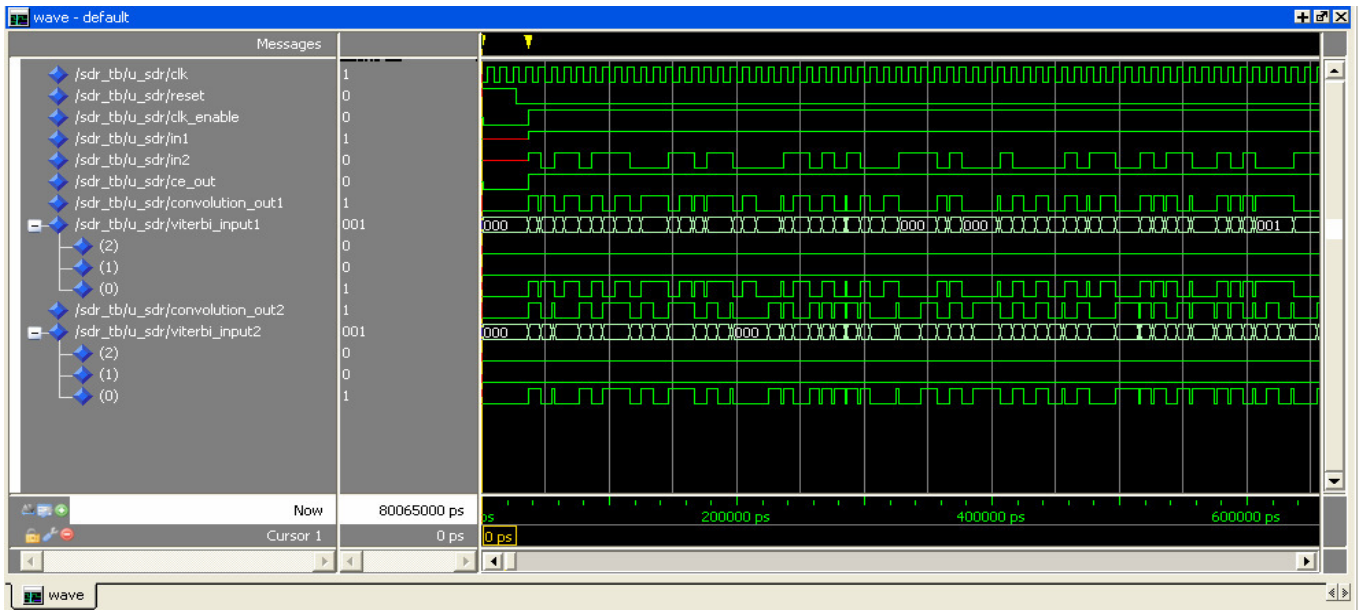


Fig.13 the convolution encoder outputs and viterbi decoder inputs.

### 2.10 Downloading Bit Stream File to FPGA Board:

The synthesis process would also produce a bit stream file that can be downloaded in the FPGA board. The bit stream file of the SDR has been successfully downloaded to Altera-Cyclone II FPGA family of EP2C35F672C6 board after installing necessary drivers on PC. The test operation the physical functionality of the SDR has been done by simply interfacing a function generator to apply input data and

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 QuartusII package.

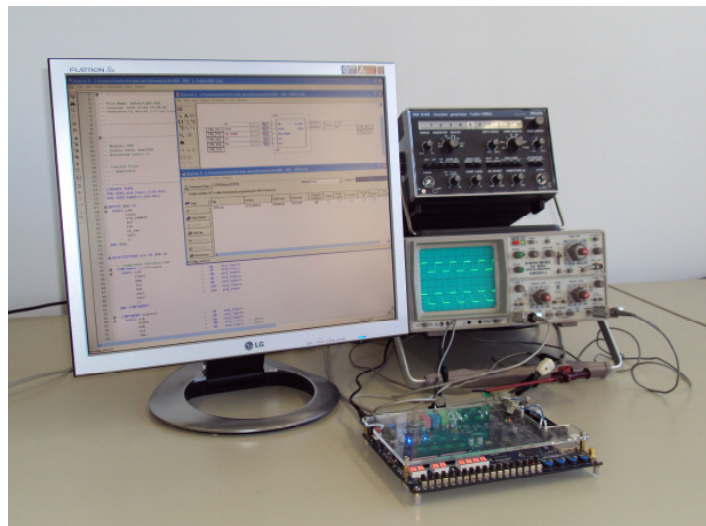


Fig.14 the implementation of proposed SDR system using Cyclone II DE2 kit

Table.1 Summary of Synthesis Reports

<i>Maximum period 22.326 ns</i>			
<i>Maximum frequency: 44.79 MHz</i>			
<i>Maximum path delay from the any node: 22.326 ns</i>			
<i>Device utilization for EP2C35F672C6</i>			
<b><u>Resource</u></b>	<b><u>Used</u></b>	<b><u>Available</u></b>	<b><u>Utilization</u></b>
<i>IOs</i>	<i>7</i>	<i>475</i>	<i>1%</i>
<i>Total Logic Elements</i>	<i>9,654</i>	<i>33,216</i>	<i>29%</i>
<i>Total Memory bits</i>	<i>1,664</i>	<i>483,840</i>	<i>&lt;1 %</i>
<i>Total PLLs</i>	<i>0</i>	<i>4</i>	<i>0 %</i>

#### 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 function generator of rate 500 KHz has been applied to the kit while the output has been measured by 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.

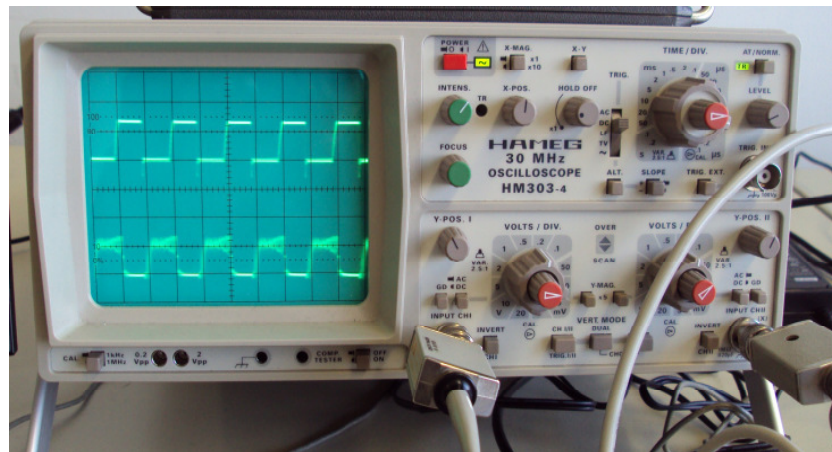


Fig. 15 the output signal of DE2 development kit.

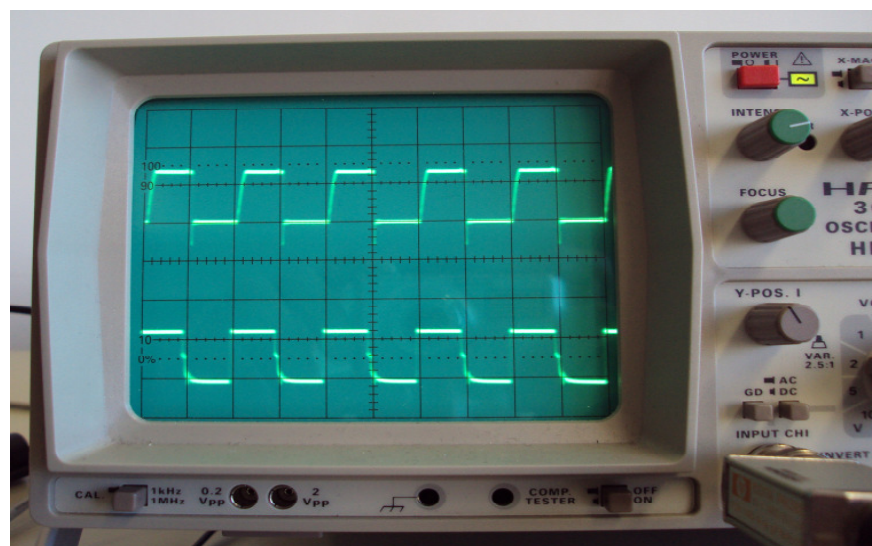


Fig 16 the output from the DE2 development kit



## CONCLUSIONS:

A Baseband SDR system was successfully developed using Altera CycloneII 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. In order that the design be compiled and synthesized successfully using Quartus II, the real values data should be changed first to fixed point data, for example 0.707 real value was handled as 707. The Simulink HDL coder does not generate HDL code for all MATLAB-Simulink blocks, so some blocks in the design should be redesigned using the basic operation principle with elements supported by Simulink HDL coder.

## ACKNOWLEDGEMENTS:

This work has been done at digital techniques laboratory in the faculty of information, media and Electrical Engineering at Cologne University of Applied Sciences (CUAS)-Germany, during a short term fellowship financed by Ministry of Higher Education and Scientific Research (MOHESR) in Iraq. So, the authors would like to thank all the academic and technical staff in the laboratory especially Prof. Dr. Ing. Georg Hartung and Dip. Ing. Peter Polig. for their support and kind hosting. Also they would like to thank to Mr. Dawood Sulaiman from MOHESR and Mr. Ahmed Salih from Al-Mustansiryah

University for their help and encouragement before and during the fellowship.

## REFERENCES:

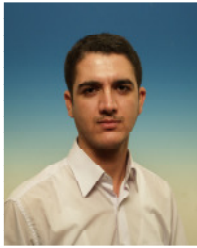
- [1] P. G. Burns, "Software Defined Radio for 3 G", Artech House – England, 2003.
- [2] "Software-Defined Radio", Wipro Technologies, 2002.
- [3] Grégory, E.N, M.S and François. V, "Transaction Level Modeling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM" [Design, Automation & Test in Europe Conference & Exhibition](#), 16-20 April 2007
- [4] Hikmat N. Abdullah 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](http://www.xilinx.com/products/software/sysgen/app_docs/user_guide_Chapter_10_Section_3_Subsection_61.htm).
- [7] B. Sklar, *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](http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm).
- [9] V. A. Pedroni, "Circuit Design with VHDL", Massachusetts Institute of Technology (MIT) Press, 2004.



Hikmat N. Abdullah was born in Baghdad, Iraq in 1974. He obtained his B.Sc. in Electrical Engineering in 1995, M.Sc. in Communication Engineering in 1998 at University of Al-Mustansiryah, Iraq and Ph. D. in Communication Engineering in 2004 at University of Technology, Iraq. He is interested in subjects of synchronization of direct sequence spread spectrum systems, channel coding and application of FPGA technology in communication systems. Since 1998 he works as a lecturer in the Electrical Engineering Department, college of Engineering, in Al-Mustansiryah University, Iraq.



Safa'a A. Ali was born in Baghdad, Iraq in 1977. He obtained his B.Sc. in Mathematical sciences in 1999 at University of Baghdad, Iraq. His B.Sc. in Electrical Engineering in 2006, M.Sc. in Communication Engineering in 2009 at University of Al- Mustansiryah, Iraq. He is interested in subjects of mathematical engineering analysis, multicarrier systems, and application of FPGA technology in communication systems.



Hussein A. Hadi was born in Babylon, Iraq in 1985. He obtained his B.Sc. in Electrical Engineering in 2007 at University of Al-Mustansiryah, Iraq. He is interested in subjects of software defined radio and application of FPGA technology in communication systems.