Journal of Mobile Communication 4 (1): 9-16, 2010
ISSN: 1990-794X

Design and Synthesis of HDL Code Based on High Level FPGA Design Flow for SDR Model

Mohd F. Ain, Majid S. Naghmash and Y.H. Chye
School of Electrical and Electronic Engineering, University Sains Malaysia,
Engineering Campus, 14300 Nibong Tebal, Penang, Malaysia

Abstract: There has been considerable recent interest in defining a Hardware Abstraction Layer (HAL) to facilitate code reuse in the signal processing subsystems of software-defined radios. HDL for FPGA-based signal processing is a significant aspect of such HAL efforts. In this study, we show how a platform-based approach to FPGA design that provides a high level of design abstraction can also provide the ability to target multiple FPGA families from a single source model. The approach combines direct mapping of a Simulink model with code generation of register-transfer level HDL. By exploiting retiming and other optimizations available through logic synthesis, it is possible to obtain very efficient realizations of signal processing functions. This research complements HAL recommendations by focusing on mechanisms, guidelines and methodologies for constructing signal processing functions in FPGAs. It helps to address requirements for executable specifications as well as providing source that can be compiled though automatic code generation. At last, this new design technique would help in designing and realizing SDR-3G wireless communication system and accelerate the transition to 4G wireless communication system.

Key words: FPGA, software defined radio, digital signal processing, modeling, HDL, simulation, synthesis, implementation verification

INTRODUCTION

Field-Programmable Gate Arrays (FPGAs) are widely used to implement physical layer signal processing functions for Software-Defined Radios SDRs (Dick and Hwang, 2004). FPGAs provide very high performance custom hardware solutions and can be reconfigured in system and when bringing up a new waveform in the modem.

Despite their reprogrammability, they have historically been considered part of the hardware within a modem rather than part of the software. Consequently, the SDR software control layer or Software Communications Architecture (SCA) has largely ignored issues related to the specification, configuration, signal transport or inter-component interfaces that are important to the platform provider of an SDR that exploits FPGAs. In this study, we describe a platform-based approach for obtaining portable FPGA source code, whilst simultaneously providing executable specifications, test harnesses and golden test vectors (i.e., providing accurate input/output relations for establishing conformance to specification through simulation). The approach treats a high-level system model specified in Simulink as the source code for an FPGA implementation. A block in the model may map onto a set of intellectual property blocks provided by the vendor that exploit vendor-specific device resources to implement the block's function efficiently in a number of FPGA families. Alternatively, a block may map onto a behavioral description in a hardware description language that is inherently portable. It is on the latter case that we focus in this study. The approach extends widely used FPGA design techniques using industry standard design tools. Although, described in terms of proprietary (though commercially available) tools for Xilinx FPGAs, out approach is equally applicable to other devices.

MATERIALS AND METHODS

Register transfer level HDL: Major FPGA vendors have multiple device product lines, each of which may be further divided into different families, each of which is further divided into different part types that differ in available resources, speed grade and package. For example, Xilinx has two primary FPGA product lines: Virtex, which targets highest performance and gate density and whose most recent families include Virtex-4

Corresponding Author: Mohd F. Ain, School of Electrical and Electronic Engineering, University Sains Malaysia, Engineering Campus, 14300 Nibong Tebal, Penang, Malaysia

and Virtex-II Pro (Xilinx, 2003a, b) and the Spartan family, which targets high volume and lower cost applications. Spartan-3 is the most recently introduced family member (Xilinx, 2006). Because a new FPGA family is introduced roughly every 12-18 months and the design cycle for a major SDR design can be a significant fraction of this period, the implications of code portability (or more accurately, non-portability) are clear. Often a system must be built to target a family in advance of broadly available silicon.

The prevailing abstraction in hardware description languages for FPGA design is Register Transfer Level (RTL), which can be synthesized into device-specific logic resources (De Micheli, 1994). At this level of abstraction, a design is a network of combinational circuits separated by registers. Registers and other circuit elements are represented behaviorally through idioms inferable by commercial synthesis tools. This style of coding allows the user to specify for example an addition operation with the operator '+', with the synthesis tool mapping this appropriately to device specific architecture primitives. Considerable progress has been made over recent years in commercial synthesis tools to efficiently target FPGAs. In addition to technology mapping, synthesis tools also apply optimization algorithms to a circuit that preserve behavior, while improving the circuit quality under well-defined criteria (typically logic area or performance). Of particular interest is retiming, which is the reallocation of unit delays (e.g., registers) throughout a circuit in order to reduce the number of combinational logic levels (De Micheli, 1994). There is a close correlation between the largest number of logic levels and the frequency with which bounding registers can be clocked without setup or hold time violations, so retiming is a particularly effective synthesis optimization.

SDR design using xilinx system generator DSP design tools: This study focuses on the system level design using DSP Design Tools. Based the available resources from (Proakis and Salehi, 2008; Haessig *et al*., 2005), a simple 16-QAM (Quadrature Amplitude Modulation) SDR (Software Defined Radio) transmitter and receiver model is designed for Virtex-4 FPGA architecture in Xilinx System Generator and MATLAB/Simulink environment. The GUI (Graphical User Interface) of the design is model-based structure using Xilinx specific blockset as shown in Fig. 1.

Xilinx FPGA design flow and software tools: The basic flow for designing with most every FPGA vendor is shown in Fig. 2. This flow also illustrates the software tools used specifically for the Xilinx FPGA design. Xilinx owns and maintains a complete tool set for the entire FPGA design flow, some of which is in collaboration with individual companies. Essentially, all of its tools are integrated under one umbrella called the Integrated Software Environment (ISE) package. Simulation and testing of the SDR transmitter and receiver design were done using *System Generator*, a system level modeling tool from Xilinx (2003a, b). This tool can be used for designing and testing DSP systems for FPGAs in a visual data flow environments such as MATLAB *Simulink*. This diagram shows that we can use Xilinx System Generator's blocks in the design and generate a synthesizable design which can be implemented using Xilinx ISE's Project Navigator (Xilinx, 2007a, b). It also uses ModelSim block which is a helper block to invoke ModelSim simulator and actually simulate the design. The simulator's output is fed back to Simulink for verification and the results can be displayed using Simulink's sinks. The techniques have been incorporated in the HDL Simulation and ModelSim behavioural synthesis tool that reads in high-level descriptions of DSP applications written in MATLAB and automatically generates synthesizable RTL models in VHDL or Verilog. Experimental results are reported and mapped onto the Xilinx Virtex-4 FPGAs. Once the overall design and budget has been defined from the system level, the FPGA design flow can begin. Today, most designs begin with a Hardware Description Language (HDL), either Verilog or VHDL, which can be entered

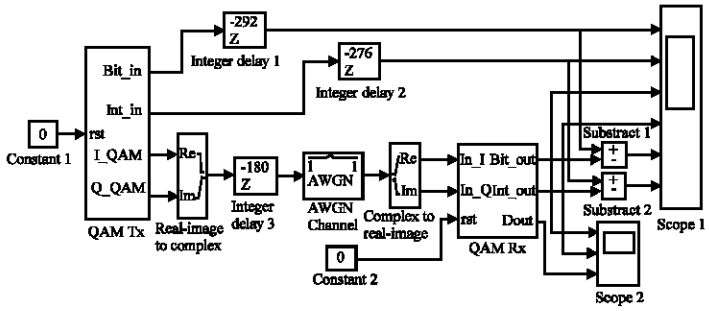

Fig. 1: SDR design using system generator

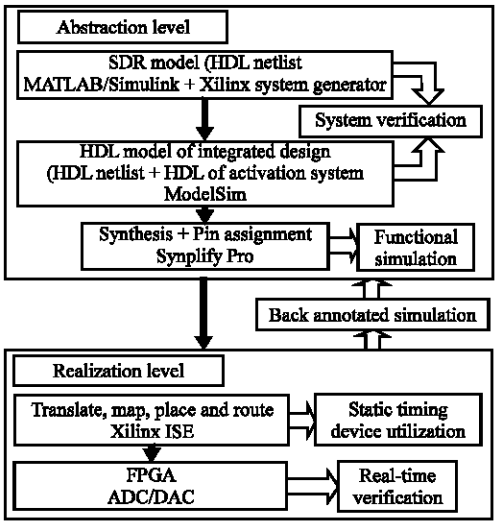

Fig. 2: System generator based FPGA design flow

using any basic text editor. Sometimes Verilog and VHDL are referred to as RTL or register transfer logic. Multiple levels of abstraction can be built upon each other to create hierarchical designs. These abstraction levels allow for incremental and iterative refinement of gate-level implementation of the system design.

Included as part of ISE software suite is a tool called Core Generator (Avnet, 2006). This is a GUI based tool that offers a designer parameterized logic Intellectual Property (IP) cores that have been optimized (for area and speed) to be implemented on Xilinx FPGAs. These ready-made cores offers functions that range from simple counters, comparators, adders, multipliers, to full system-level building blocks such as memories, FIFOs, filters and transforms. The structural abstraction level of HDL allows logic designers to implement these cores right into the source code as a netlist. Once the netlist has been complete, it needs to be functionally verified. Functional verification tests the design to determine if it is working as intended. First a test-bench (high level HDL stimulus file) is created that wraps around a design to stimulate the design inputs (Xilinx, 2007a, b).

Depending on the type of design, sometimes the test-benches are more advanced and monitor the design outputs by using additional logic to check the results and thereby creating automated self checking tests. In either case, it is necessary to visually verify and debug the signals at all levels of hierarchy in the design on wave simulator. A company called Mentor Graphics produces an HDL simulation and debug environment called ModelSim. If an HDL design is purely behavioural, the simulator will most likely be able to properly simulate the design. However, since the HDL source code may contain either Xilinx IP cores or lower level macros specific to the FPGA architecture, a set of Xilinx libraries that indicate how each of the higher level blocks should behave must be made available to the simulator. Xilinx solves this issue by working with its own version of ModelSim, which includes all necessary Xilinx libraries to functionally verify any abstraction level for any Xilinx FPGA. Model Sim can be run as a stand-alone program or it can be executed from within ISE. During synthesis, the HDL source code including any Xilinx specific cores are translated into a gate-level structural netlist. This netlist is then optimized for implementation on Xilinx device.

There are two synthesis tools used in the Xilinx FPGA design flow (Xilinx, 2007a, b). As part of the ISE suite, Xilinx offers its own synthesis tool called Xilinx Synthesis Tool or XST. The ISE tools also contain synthesis tool called Synplify Pro produced by Synplicity (2007). This synthesis tool is an industry standard tool with design libraries available to support nearly every major FPGA platform. Although, both tools essentially yield the same final result, Synplify Pro is generally the synthesis tool of choice, especially if the FPGA design is a test platform for an ASIC implementation which Synplicity also supports. The input file types to a synthesizer are either V (Verilog) or VHD (VHDL) with the output file type of Synplify being an EDIF (Electronic Data Interchange Format) file and the output file of XST being an NGD (Native Generic Database) file. Since the netlist has not been mapped into Xilinx specific building blocks at this stage, synthesis tools cannot give accurate timing results in its timing and area log files only estimation. The output of the Synthesis tool is then fed into the next stage of the design flow, which is called Implementation in the Xilinx flow and is the core utility of the ISE software suite. Before this step is executed, the User Constraints File (UCF) is typically filled out. The most critical information in the constraints file is the pin locations for each I/O specified in the HDL design file and the timing information such as the system clock frequency (Avnet, 2006). The constraints file also allows a designer to specify specific mapping of gates in the netlist to specific Xilinx blocks as well as the placement of these blocks. Further, it allows specific timing constraints on a per I/O basis for any critical timing paths. ISE contains a built in GUI called PACE (Pin And Constraints Editor) for the purpose of entering all the constraints. The Implementation step of Fig. 2, reads in the constraints file and consists of three major steps: translate, map and place and route. The Translate step essentially flattens the output of the synthesis tool into a large single netlist. A netlist in general is a big list of gates (typically NAND/NOR) and is compressed at this stage to remove any

hierarchy. The Map step groups the logical symbols in the flattened netlist into physical components specific to the target device. The Place and Route step then places each of these physical components onto the FPGA chip and connects them through the switch matrix and dedicated routing lines. Then, timing information is generated in log files that indicate both the propagation delay through each building block in the architecture as well as the actual routing delay of the wires connecting the building blocks together.

The ISE Implementation stage outputs a NGD (Native Generic Database) file (Avnet, 2006). Just as the synthesis tools output an HDL simulation netlist, so do the ISE Implementation tools. However, this time these simulation files contain all of the timing information that was generated in the Map and Place and Route stage. These files can be used for 2 purposes. First they can be read back into the Model Sim simulator just as before. This is called back annotated timing simulation.

This type of simulation is much more time consuming and difficult, since all of the propagation and wiring delays are evident on each signal. Second, they can be used for static timing i.e., timing analysis that does not depend on stimulus to the design circuit. This step is critical in ASIC design flows and is also available in the FPGA design flow through ISE tool. Notice once more the iterative verification process that leads back through HDL simulator.

Once the design had been fully verified with the correct timing then final configuration bit file that will eventually be downloaded into the FPGA can be generated by the main ISE tool. Once the bit file has been created, another tool in the ISE suite called IMPACT is used to program either the FPGA directly or through JTAG interface (Avnet Memec, 2005), i.e., standard cable connected to computer through parallel port. For direct programming, the driver of the target FPGA must be activated and the bit file is downloaded into the FPGA via IMPACT. Afterwards, real-time verification for the implemented FPGA design (before ADC and after DAC) will be executed.

RESULTS AND DISCUSSION

HDL design of activation system: The ModelSim software is used to design Verilog HDL module of activation of plug-in Avnet Electronic Marketing P240 Analog Module (Xilinx, 2007a, b) and Virtex-4 FPGA on-board ICS8442 Programmable LVDS (Low Voltage Differential Signaling) Clock Synthesizer (Avnet Memec, 2005) with specific configurations. The HDL module of activation system should include all the specifications, characteristics and features described above so that ADC (Texas Instrument, 2007) and DAC (Texas Instrument, 2005) in P240 can function properly. The simulation result of HDL Module of Activation System for both the transmitter and receiver (which use the same Virtex-4 FPGA model) is shown in Fig. 3.

HDL module of integrated design: The HDL module of activation system and HDL netlist of SDR model are verified firstly before being combined to become HDL module of integrated design. The simulation results for the HDL netlist of SDR transmitter and receiver are shown in Fig. 4 and 5, respectively. Notice these simulation results are similar to the System Generator design. Considering the real-time implementation of integrated design using FPGA and P240 Analog Module, the ADC and DAC in P240 should be activated and configured first prior to the running of FPGA design in order to avoid instability of ADC and DAC that can produce undesired outputs to or from FPGA during process of configuring ADC and DAC. Thus, the main clock enable (ce) of FPGA design is disabled during the transfer of Serial Programming Interface (SPI) codes to ADC and DAC in P240 (Texas Instrument, 2007).

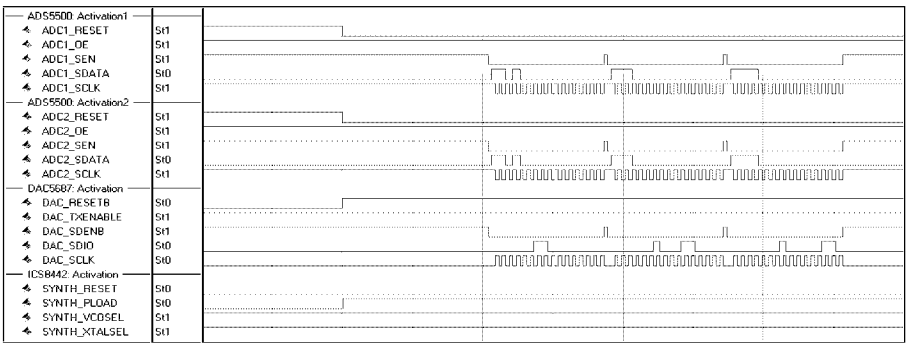

Fig. 3: Simulation result of HDL module of activation system

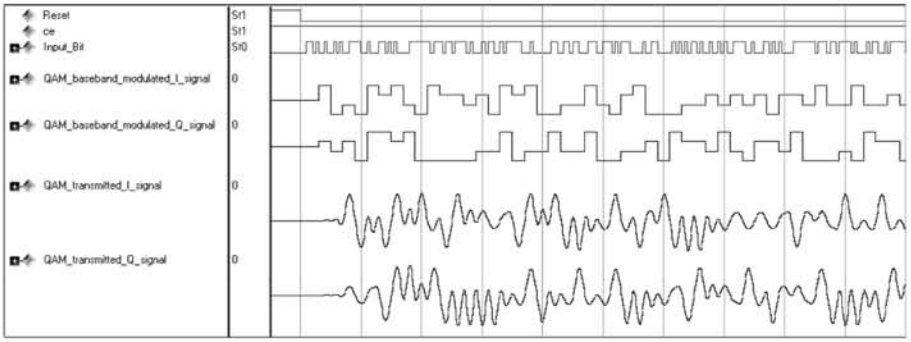

Fig. 4: Simulation result of HDL netlist of 16-QAM transmitter

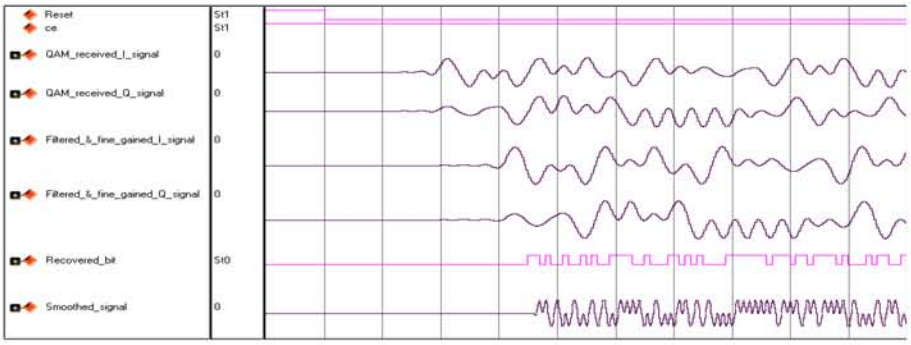

Fig. 5: Simulation result of HDL netlist of 16-QAM receiver

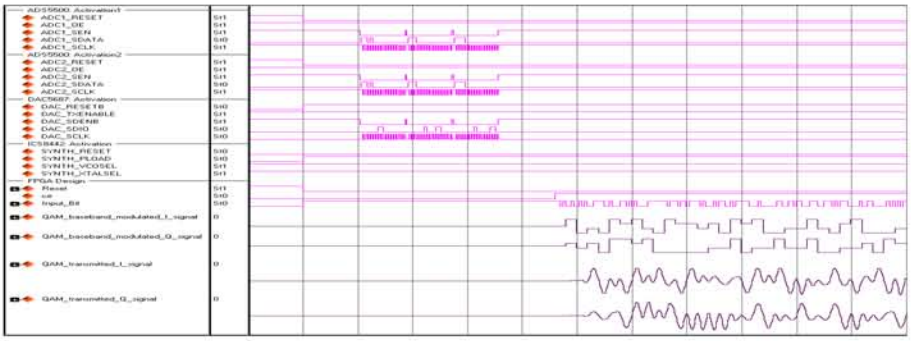

Fig. 6: Simulation result of HDL module of integrated design (16-QAM transmitter)

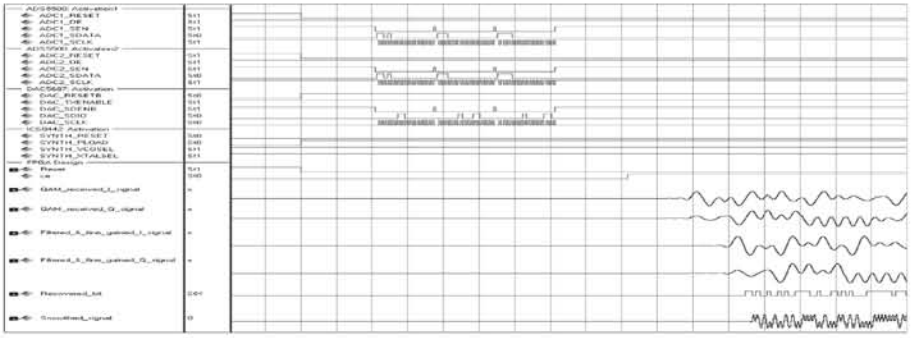

Fig. 7: Simulation result of HDL module of integrated design (16-QAM receiver)

Controlling the main 'ce' would be easier rather than clock enable clear (ce_clr), which requires additional logics to adjust sampling phase of all the multi-sample data when de-asserted (Xilinx, 2007a, b). The simulation results of HDL module of integrated design for the transmitter and receiver are shown in Fig. 4-7, respectively.

Synthesis of integrated design: Though Xilinx ISE has its own synthesis tool called XST (Xilinx Synthesis Technology) but it can only synthesize HDL netlist generated from System Generator. Therefore, Synplify Pro software is used to perform logic synthesis for the HDL module of integrated design in 2 stages (Synplicity, 2007).

Logic compilation and optimization: Compile the HDL module of integrated design to Xilinx FPGA structural elements and then optimize the integrated design making it as small as possible to improve circuit performance.

Technology mapping: Map the optimized integrated design to Xilinx FPGA logic components using architectural specific techniques.

Timing characteristics is another important issue that will affect the performance of FPGA implementation. Thus, the estimated period (required path delay) for the FPGA element must not exceed the requested clock period. So, timing slack (requested period-estimated period) should be positive value otherwise the integrated design has to be reworked. The clock frequencies are set to 100 MHz for CLK_100 (ADC/DAC SPI process) and 80 MHz for LIO_CLKIN_1 (16-QAM SDR transmitter and receiver) (Avnet Memec, 2005). The estimated timing report for the synthesized design meets the timing constraints as shown in Table 1 and 2. The Starting Clock of system relates to Xilinx IP core used in the integrated design. Once the synthesis of the integrated design is error free in terms of functionality and timing, FPGA pins (pad locations) are assigned accordingly referring to user guide of (Avnet Memec, 2005).

FPGA implementation: The synthesis output files that are required in Xilinx ISE software are in EDIF (Electronic Design Interface File) and UCF (User Constraints File) formats, which represent optimized netlist of integrated design and timing constraints and FPGA pin assignment, respectively.

To implement the synthesized design into Virtex-4 FPGA development board, Xilinx ISE performs steps as described in Fig. 2, i.e., Translate, Map, Place and Route (PAR), Bit Generation and Program Download to FPGA for the SDR model. Make sure no errors for each step described earlier. The timing requirement is satisfied as shown in the post-PAR (final) static timing report in Table 3 and 4.

Due to no error and warning for DRC (Design Rules Checker) and bit-stream generation process, the configuration bit-stream file is downloaded to Virtex-4 FPGA board. The input and output waveforms are tested in physical assembly of the SDR transmitter and receiver as shown in Fig. 8.

The ADC input and DAC output signals in P240 Analog Module for the SDR transmitter and receiver are connected to oscilloscope in order to display real-time result as shown in Fig. 9 and 10. Both the transmitted and received signal should be similar,

Table 1: Estimated timing report for 16-QAM transmitter

Starting clock	Requested frequency (MHz)	Estimated frequency (MHz)	Requested period	Estimared period	Slack
CLK_100	100.0	189.4	10.000	5.280	4.720
LIO_CLKIN_1	80.0	218.7	12.500	4.573	7.927
System	100.0	511.8	10.000	1.954	8.046

Table 2: Estimated timing report for 16-QAM receiver

Starting clock	Requested frequency (MHz)	Estimated frequency (MHz)	Requested period	Estimared period	Slack
CLK_100	100.0	189.4	10.000	5.280	4.720
LIO_CLKIN_1	80.0	134.1	12.500	7.456	5.044
System	100.0	216.6	10.000	4.617	5.383

Table 3: Post-PAR static timing report for 16-QAM transmitter

Constraint	Check	Worst case slack	Best case achievable	Timing error
TS_CLK_100 = PERIOD TIMEGRP "CLK_100"	SETUP	4.412 ns	5.588 ns	0
10 ns HIGH 50%	HOLD	0.542 ns		0
TS_LIO_CLKIN_1 = PERIOD TIMEGRP "LIO_CLKI	SETUP	5.607 ns	6.8393 ns	0
N_1" 12.5 ns HIGH 50%	HOLD	0.257 ns		0

Table 4: Post-PAR static timing report for 16-QAM receiver

Constraint	Check	Worst case slack	Best case achievable	Timing error
TS_LIO_CLKIN_1 = PERIOD TIMEGRP "LIO_CLKI	SETUP	1.465 ns	11.035 ns	0
N_1" 12.5 ns HIGH 50%	HOLD	0.358 ns		0
TS_CLK_100 = PERIOD TIMEGRP "CLK_100"	SETUP	5.021 ns	4.979 ns	0
10 ns HIGH 50%	HOLD	0.512 ns		0

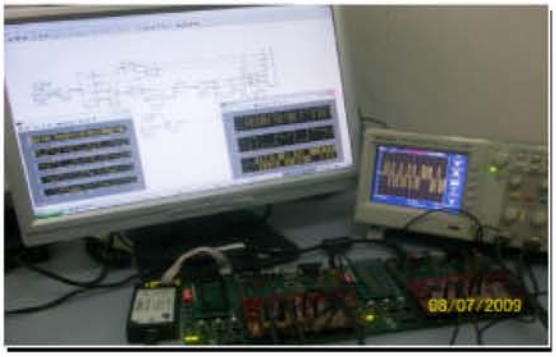

Fig. 8: Physical assembly of SDR transmitter and receiver

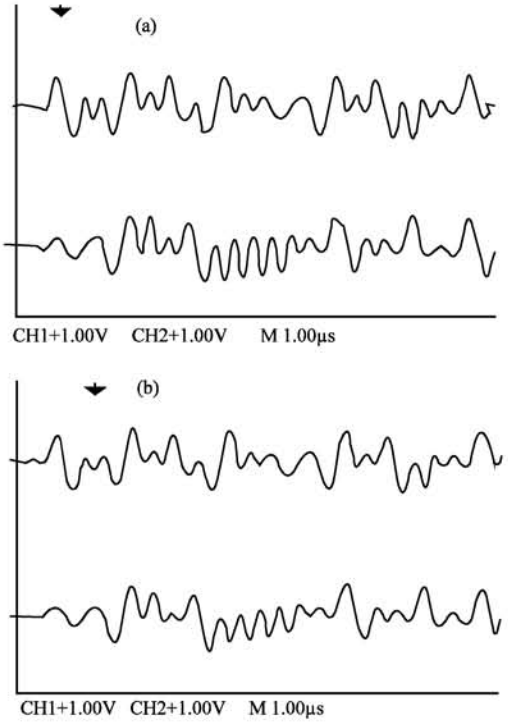

Fig. 9: Real-time (a) transmitted (IQ) (b) received (IQ) 16-QAM signals

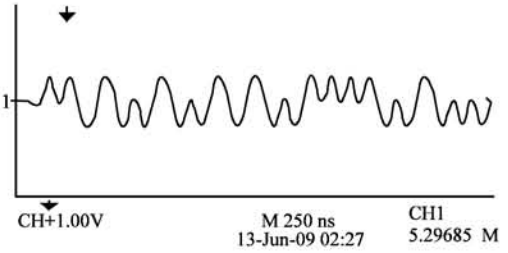

Fig. 10: Real-time pulse-shaping signal for demodulated bit

although noise effect and distortion may occur (real-world application issue). Notice that the empirical (real-time) result is similar to the simulated result.

CONCLUSION

VHDL behavioral modeling is useful in digital systems design because the designer can model the circuit in a program that simulates the circuit operation rather than spend time on complex finite state machines or truth tables. This greatly facilities and reduces the design time for a large digital system. The simulation waveforms presented in this study have proven the reliability of the VHDL implementation to describe the characteristics and the architecture of the digital design. The simulated waveforms also have shown the observer how long the test result can be achieved by using test-bench file. Software defined radio will most likely be the radio of the future.

The architecture has been developed for the two systems and the method of designing the base band of the SDR transceiver for a multi standard protocol has been devised. This topic is promised topic for the software radio systems and all communication.

With the increased adoption of FPGAs as signal processors comes an increased expectation for design flows and methodologies that support similar programming models to general purpose and DSP processors. Although, FPGA source code is not as widely portable as code for general purpose microprocessors, we have demonstrated how System Generator and similar design tools provide considerable progress towards this end. We have shown how a single System Generator model can be used to specify both behavior and implementation producing a generic RTL implementation suitable for an FPGA. The design exploits retiming and logic synthesis optimizations in order to achieve high performance.

REFERENCES

Avnet Memec, Inc., 2005. Virtex-4 MB development board user's guide. Ver. 3.0, Phoenix, Arizona, USA., pp: 1-42. http://www.files.em.avnet.com/files/177/v4mb_user_guide_3_0.pdf.

Avnet, Inc., 2006. P240 analog module user guide. Rev. 1.0, Phoenix, Arizona, USA., pp: 1-25. http://www.files.em.avnet.com/files/177/p240_analog-ug.pdf.

De Micheli, G., 1994. Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York.

Dick, C. and J. Hwang, 2004. FPGAs: A Platform-Based Approach to Software Radios. In: Software Defined Radio: Baseband Technologies for 3G Handsets and Basestations, Tuttlebee, W.H.W. (Ed.). John Wiley and Sons Ltd., England, pp: 235-272.

Haessig, D., J. Hwang, S. Gallagher and M. Uhm, 2005. Case-study of a xilinx system generator design flow for rapid development of SDR waveforms. Proceedings of the SDR 05 Technical Conference and Product Exposition, Nov. 14-18, California, USA., pp: 1-6.

Proakis, J.G. and M. Salehi, 2008. Digital Communications. 5th Edn., McGraw-Hill, New York, pp: 95-148.

Synplicity, Inc., 2007. Synplicity FPGA Synthesis Reference Manual. Synplicity, Inc., Sunnyvale, California, USA., pp: 1-20.

Texas Instrument, Inc., 2005. DAC5687: 16-bit, 500 Msps, 2x-8x interpolation dual-channel Digital-to-Analog Converter (DAC). http://focus.ti.com/lit/ds/symlink/dac5687.pdf.

Texas Instrument, Inc., 2007. ADS5500: 14-bit, 125 Msps, analog-to-digital converter data sheet. Rev. G, Dallas, Texas, USA., pp: 1-29. http://focus.ti.com/lit/ds/symlink/ads5500.pdf.

Xilinx, Inc., 2003a. DSP Design Flows in FPGA Tutorial Slides. Xilinx Press, San Jose, California, USA., pp: 1-82.

Xilinx, Inc., 2003b. Virtex-II Pro and Virtex-II Pro X FPGA user guide. Ver. 4.2, San Jose, California, USA., pp: 1-559. http://www.xilinx.com/support/ documentation/user_guides/ug012.pdf.

Xilinx, Inc., 2006. Spartan-3 FPGA family data sheet. Ver. 2.5, San Jose, California, USA., pp: 1-217. http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.

Xilinx, Inc., 2007a. System generator for DSP user guide. Release 9.2.01, San Jose, California, USA., pp: 1-346.

Xilinx, Inc., 2007b. Xilinx ISE 9.2i software manuals: Constraints guide and development system reference guide. San Jose, California, USA., pp: 1-844.