ETIN01 IC-project & Verification, digital

OFDM Symbol Detection
Verification Report

Ma Ling (770821-8867)
Peyman Pouyan (831230-6296)

Nov 2009

Abstract

This report describes the FPGA verification procedure of the OFDM Symbol Detection design.

After completing the project and having the correct result in the simulation, it is the time to verify our results in the real Hardware. In order to fulfill this task, there are two ways. One way is to fabricate the ASIC Hardware which is so costly and time consuming, and another way is to use a FPGA. Using FPGA is a good way to make prototypes and verify the designs fastly and less costly.

Before verification, the binary file is prepared in ISE and downloaded in FPGA. Then the test vector is imported into the pattern generator. The output from FPGA board can be observed in waveform window. By comparing the output in Matlab, ISE behavioral simulation and Logical Analyzer, we verify our design works well.

1. Introduction

OFDM is a popular technique nowadays. The symbol detection is a key issue of this technique. There are some algorithms developed to detect the OFDM symbol and frequency offset. The CP-based synchronization algorithm is the major one. But the traditional algorithm has a high computation complexity. We employ a low complexity algorithm and the traditional one together to detect the OFDM symbol start and carrier frequency offset. The details about RTL development are written in the OFDM Symbol Detection Report [1].

To verify our design in FPGA, a binary file is created and downloaded into FPGA board firstly. Then we use logic analyzer, we connect it to the board, import the test data and observe the output data. If the output data is the same as the output in the MATLAB and ISE behavioral simulation, we can make a conclusion that our design works well.

The tools used in verification are listed in the following sections.

1.1 Verification software tools

Matlab R2009a

Xilinx ISE 9.1

1.2 Verification hardware devices

Aigilent Logical analyzer 16822A

Digilent Spartan-3 XC3S200-FT256 Board

2. Preparation for verification

Before verifying our design through logic analyzer, we should create a binary file for the design in ISE.

In order to do this, we create a new project and add all our vhdl source files into this project in ISE. The source files are listed in Appendix III. In addition, we need to recreate our memories with the Core generator of the ISE.

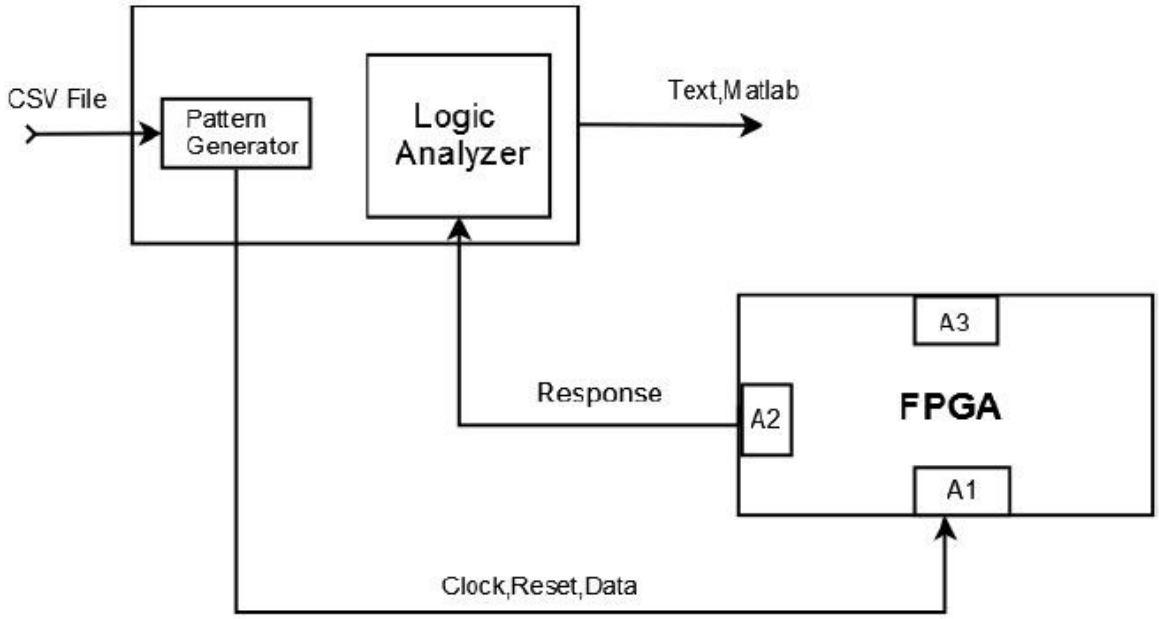
After having the correct output in the behavioral simulation, the FPGA pins are allocated to our input and output signals. This is done in I/O Pin Planning part of ISE software and we get a suitable UCF file for our design. The UCF file is attached in Appendix I.

Next Process is to synthesize the design to have the RTL gate logic. In this process, we need to make sure there are no Latches in our design. After the synthesis is done without any problem, we do a post-route simulation in the ISE to be sure of not having any major glitches that may cause the design doesn't run properly in the FPGA. A maximum clock frequency is gotten in the Place & Route to be 26MHZ which will be used in the Logic Analyzer settings later. The table in Appendix II lists our design summary after the P&R process in the Xilinx ISE.

The last step is to program the FPGA with ISE Impact and create the binary file for the verification.

3. FPGA Verification

For the verification by logic analyzer, firstly, we should make a test vector .csv file in matlab as input data. Then we import the .csv file into the Pattern Generator which is connected to FPGA extension port A1. In addition, we should capture the output from FPGA extension port A2 which is connected to Logic Analyzer. Finally, we can observe the output in the Waveform window. The overall frame is drawn as below. The details about the pod connection are listed in Appendix I.



3.1 Creating Test Vector

Based on the input data for RTL model, the test vector is constructed after converting the input data from decimal to hex. To make sure there is enough time to process the symbol data, some redundant test vectors are inserted. There are 135 sets of redundant data between the first and second symbol, 73 168 sets between the second and third symbol, and 272 sets between the rest symbols. Corresponding to these redundant test data, the Valid_in signal should be disabled.

To test our design, we create 250 symbols totally. But there are only less than 50 symbols imported due to the logic analyzer memory limitation.

3.2 Verification Procedure in Logic Analyzer

In the Logic Analyzer, we firstly set the Buses/Signals for the Pattern Generator and connect the device's output slot cable to the extension port, and then we set the Logic Analyzer's Buses/Signals. The clock frequency in pattern generator should be set to a value less than or equal to value of the maximum clock frequency gotten in ISE. Finally we set the trigger sequence for logic analyzer. The trigger sequence means the time when the output waveform starts to be generated. For our design, the Valid_Out signal works as the trigger.

3.3 Results

Figure 1, 2 and 3 show the output in Matlab, ISE behavioral simulation and Agilent Logic Analyzer.

The Expected result should be"CFO=0.20" and "Symbol Start=7".

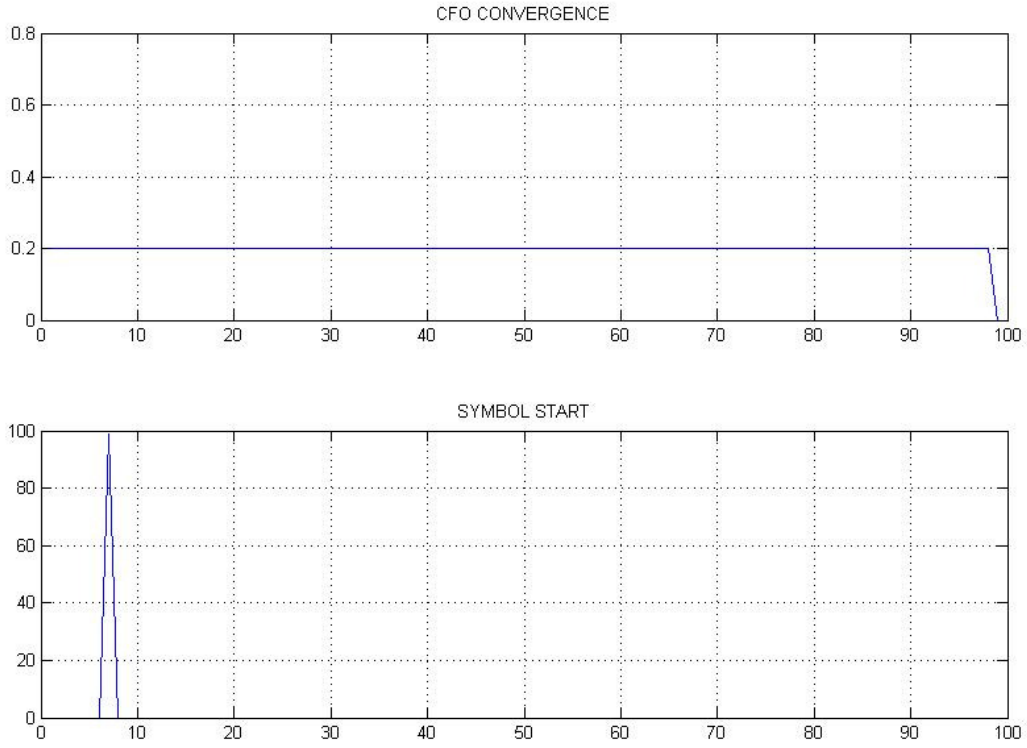


Figure 1:CFO and Symbol Start in Matlab

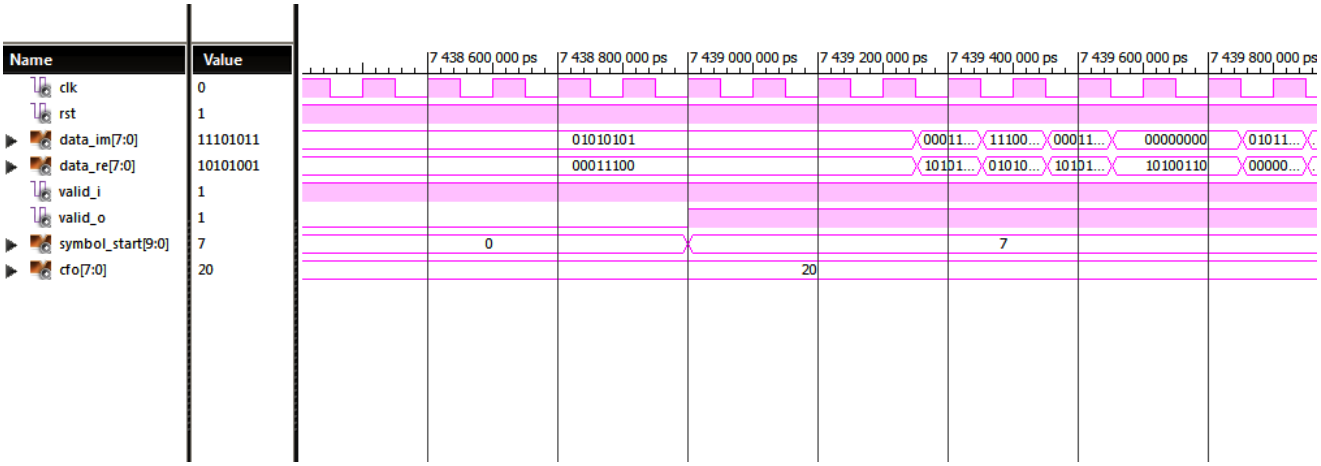


Figure 2:CFO and Symbol Start in ISE behavioral simulation

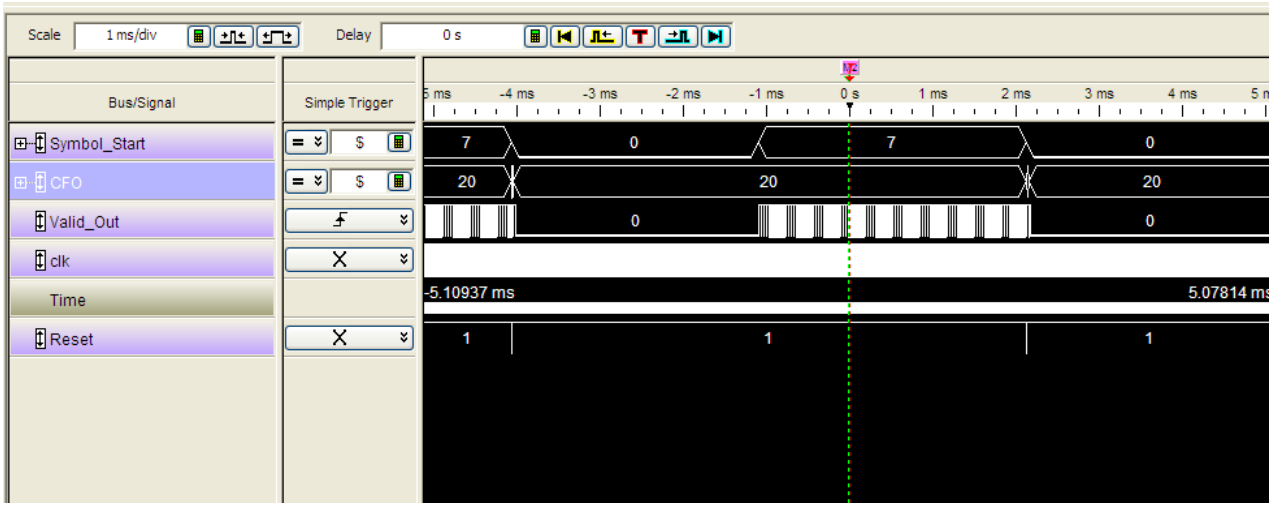


Figure 3:CFO and Symbol Start in Logic Analyzer

For the processing time, how much time do we need to have the correct CFO and Symbol start?

The following picture shows the correct Symbol Start and CFO comes after 3.71ms in the Logic Analyzer.

371985	3.719850 ms	0	20	0	0	1
371986	3.719860 ms	0	20	0	0	1
371987	3.719870 ms	0	20	0	0	1
371988	3.719880 ms	0	20	0	1	1
371989	3.719890 ms	7	20	0	1	1
371990	3.719900 ms	7	20	0	0	1
371991	3.719910 ms	7	20	0	0	1
371992	3.719920 ms	7	20	0	0	1
371993	3.719930 ms	7	20	0	1	1

The results in Matlab, ISE behavioral simulation and Agilent Logic Analyzer are all the same which means that the verification has been done correctly.

3.4 Problem

At the first verification, the Symbol Start from Logic Analyzer was correct but we had a 0.1-0.5 % deviation in CFO. Then we go back to the laboratory and do all the steps again to find out the reason.

After investigating all possible reasons for that, we checked the post route simulation once again with more attention and we found that our output from the look up table is not stable.

To fix the problem we used a flip-flop after the output of look up table. Consequently, the deviation was gone and we got a correct response for CFO too.

4. Conclusion

The new algorithm of detecting OFDM symbol start and frequency offset is successfully implemented in FPGA. The output from Matlab, ISE behavioral simulation and Logic Analyzer are identical. The correct symbol start and frequency offset can be gotten in logic analyzer. These all verify that our design works properly.

Appendix I

The .ucf file shows how the input and output signal in the design are mapped to the FPGA pins. This relation together with Agilent pod connection is listed in the tables below.

INPUT DATA	**FPGA PIN**	**Slot Pod**	**Extension Port**
Real0	R5	B1[0]	A1-13
Real1	L4	B1[1]	A1-14
Real2	C2	B1[2]	A1-15
Real3	G3	B1[3]	A1-16
Real4	C1	B1[4]	A1-17
Real5	K4	B1[5]	A1-18
Real6	B1	B1[6]	A1-19
Real7	M10	B1[7]	A1-22
Imag0	N7	B2[0]	A1-5
Imag1	L5	B2[1]	A1-6
Imag2	T8	B2[2]	A1-7
Imag3	N3	B2[3]	A1-8
Imag4	R6	B2[4]	A1-9
Imag5	M4	B2[5]	A1-10
Imag6	T5	B2[6]	A1-11
Imag7	M3	B2[7]	A1-12
Valid_in	F3	B3[0]	A1-23
Reset	G4	B3[1]	A1-24
Clock	N8		A1-4

OUTPUT DATA	**FPGA PIN**	**Slot Pod**	**Extension Port**
Symbol Start0	D5	A1[0]	A2-5
Symbol Start1	C5	A1[1]	A2-6
Symbol Start2	D6	A1[2]	A2-7
Symbol Start3	C6	A1[3]	A2-8
Symbol Start4	E7	A1[4]	A2-9
Symbol Start5	C7	A1[6]	A2-10
Symbol Start6	D7	A1[7]	A2-11
Symbol Start7	C8	A1[8]	A2-12
Symbol Start8	D8	A1[9]	A2-13
Symbol Start9	D10	A1[10]	A2-15
CFO0	A3	A2[0]	A2-16
CFO1	B4	A2[1]	A2-17
CFO2	A4	A2[2]	A2-18
CFO3	B5	A2[3]	A2-19
CFO4	A5	A2[4]	A2-20
CFO5	B6	A2[5]	A2-21
CFO6	B7	A2[6]	A2-22
CFO7	A7	A2[7]	A2-23
Valid_Out	A9	A3[0]	A2-26
Clock		A3[1]	A1-4

Appendix II

The design summary report is listed below.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	399	3,840	10%
Number of 4 input LUTs	1,697	3,840	44%
Number of occupied Slices	1,053	1,920	54%
Number of Slices containing only related logic	1,053	1,053	100%
Number of Slices containing unrelated logic	0	1,053	0%
Total Number of 4 input LUTs	1,868	3,840	48%
Number used as logic	1,697		
Number used as a route-thru	171		
Number of bonded IOBs	38	173	21%
Number of RAMB16s	3	12	25%
Number of MULT18X18s	12	12	100%
Number of BUFGMUXs	1	8	12%
Average Fanout of Non-Clock Nets	3.04		

Appendix III

All source files are listed below:

ofdm_types.vhd

ofdmsyn.vhd

AlgorithmFSM.vhd

CorrelatorFSM.vhd

lfsr.vhd

LUTRom.vhd

tb_ofdm.vhd

Reference

[1] Ma Ling, Peyman Pouyan. ETIN01 ICproject & Verification, digital, OFDM Symbol Detection Report.

[2] Spartan-3 Starter Kit Board User Guide, Xilinx.

Acknowledgments

Thanks to Joachim Rodrigues, Isael Diaz, Deepak Dasalukunte, Chenxin Zhang and Johan Löfgren for helping us to complete our whole project. It's our pleasure to study with your group.