EEC0055 - Digital Systems Design

2020/2021

Laboratory 2 - V1.0
Nov 17, 2020

This document presents the reference design to be used for the development of the final laboratory project and introduces the digital design development system for XILINX FPGAs (ISE 14.6). This software package includes a toolchain to perform the complete design flow, although the simulation processes will be done using QuestaSim.

The reference project implements a basic digital audio processing system, receiving a stereo analog signal from an audio source, converting it to the digital domain and sending it back to a digital to analog converter to output the analog audio signal. The whole project is described in Verilog HDL and your design task will be adding to this design a digital signal processing system for performing a basic set of operations, to be detailed later.

During the first laboratory class the students should install the design kit provided, understand the organization of the different folders and run the whole design flow according to the guide included in this document and, after concluding the design process, configure the FPGA board and play with the circuit. The user manual of the Atlys board can be found online in https://reference.digilentinc.com/atlys/atlys/refmanual.

Figure 1 presents a simplified block diagram illustrating the system implemented in the reference design. The audio CODEC, external to the FPGA, includes two ADCs and two DAC and a set of analog amplifiers, filters and multiplexers and performs the analog to digital and digital to analog conversion of a stereo audio signal with 18 bits/sample, 48 k samples/second. The datasheet of this integrated circuit is available online here. The CODEC control module (`LM4550_controler.v`) implements the interface with the audio codec and provides/receives to/from the signal processing modules the audio samples as 18 bit words, signed in two's complement. The signal processing block included in the reference design (`psdi_dsp.v`) performs a basic processing on the input samples and send them back to the audio CODEC. Open this module and identify the function implemented. Note that this module serves only to illustrate the mechanism to implement a basic function and the synchronization of the signal processing operations. The control of the audio CODEC and the configuration of the signal processing block from an external computer is implemented by an asynchronous serial interface and a bank of 32-bit input and output ports (modules `uart.v` and `ioports16V2018.v`). The 16 output ports and the 8 input ports can connect to the rest of the circuit for any purpose and they can be written/read using a small set of commands sent via a serial port and a Windows application is available for controlling the audio CODEC and writing/reading the in/out ports (see annex A).

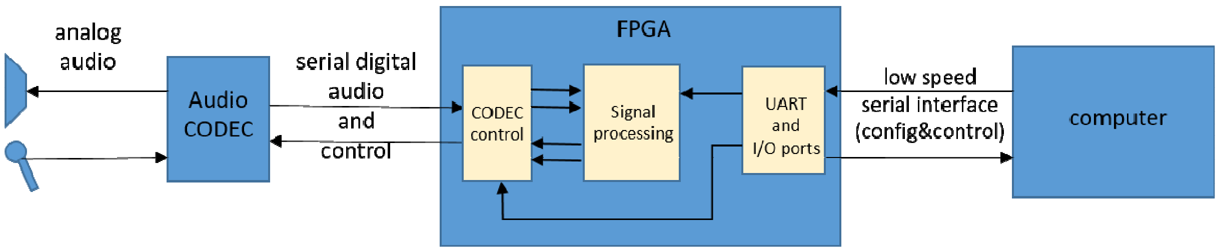


Figure 1 – Simplified block diagram of the reference design.

The whole design is synchronous with a single clock signal of frequency 12.288 MHz. In the design toplevel module, `s6base_top.v`, the global clock and reset signals are named `clock` and `reset`, respectively.

2 - Installation of the reference project

Download the archive `PSD-1920-LAB2.zip` and extract all files to a new working directory. Do not install the project into the desktop or any other directory containing spaces or other special characters (like accents) in the full pathname. The installed directory tree is:

Directory	**Contents**
`./matlab`	Matlab scripts (includes code for interfacing with the system and scripts for generating simulation data from audio .wav files)
`./doc`	Documentation (this guide, ther Audio CODEC datasheet and the manual of Atlys FPGA board)
`./impl`	ISE project directory (the project is already created)
`./sim`	Create here all the simulation projects for running QuestaSim outside of the ISE environment
`./simdata`	Data files used for simulation
`./src/data`	Other source files (user constraints file, RAM/ROM contents,…)
`./src/verilog-rtl`	RTL synthezisable Verilog modules
`./src/verilog-tb`	Simulation Verilog modules (testbench)
`./sw/bin`	The software application for accessing the FPGA system

As the simulator QuestaSim will now be launched from the XILINX ISE environment, the simulation process will run in the same directory as the implementation (`./impl`). When using QuestaSim to run simulations not launched from the ISE environment, create the simulation projects in the `./sim` directory.

The project provided contains all Verilog source files, either for synthesis and for simulation, Matlab scripts for creating simulation stimuli, and other files required for the implementation, all located in convenient directories referred above.

The main stages to perform the implementation of a digital system to the FPGA platform are summarized below. Note that in this project the first 5 steps have already been done. However, you should analyze the main Verilog modules to clearly understand their organization, particularly the toplevel module `./src/verilog-rtl/s6base_top.v` to identify its relationship to the block diagram shown in figure 1 and the testbench `./src/verilog-tb/s6base_tb.v` to understand the organization of the verification program implemented.

(1) **Conceive and design** the digital circuit at the RTL level: abstract block diagram, define the system hierarchy, set main design constraints and goals
(2) **Build the RTL digital models** of each sub-circuit (or *modules*, in Verilog HDL)
(3) **Verify each individual module** to check their functionality (functional simulation)
(4) **Assemble the whole circuit** by building the toplevel module
(5) Build a toplevel testbench
(6) **Perform a behavioral simulation** to verify the toplevel circuit
(7) **Synthesize the whole circuit**: set timing constraints, iterate with different synthesis options to meet the design goals; if needed redesign the RTL code.
(8) **Look at the synthesis warnings**: look for latch inference, unconnected outputs, unplanned logic trimming, pin-net width mismatch,…
(9) **Verify the circuit after synthesis** (post-translate simulation), look for possible errors due to wrong interconnections, bus width mismatch, …
(10) **Place and route the design** (physical synthesis): define the timing constraints, I/O placement and electrical constraints (already done in file ./src/data/s6base.ucf) and set the optimization goal and effort.

(11) **Check the static timing reports** to determine which parts of the design are not meeting the timing constraints. These reports are created by the Static Timing Analysis (STA) tool and calculates, among others, the propagation delays of all flop-to-flop paths that will determine the maximum clock frequency allowed by the circuit.

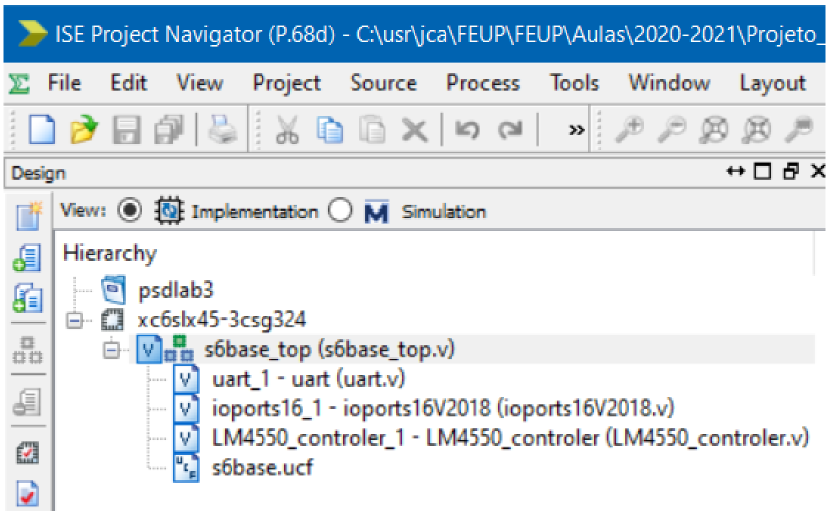
(12) **Verify the circuit after P&R** (post-route simulation). This includes the most accurate timing models of the logic blocks and interconnections.

(13) **“*Fabricate*” the circuit**: generate the bitstream file to configure the FPGA. Although there is no physical fabrication involved in this process, the term “*testing in silicon*” is currently used to refer to the physical implementation and experimentation in the real FPGA device.

3 - Implementatiom

Execute the application “ISE Design Suite 14.6” and close any project that may be open at startup (by default this tool will always open the last project). Open your project by selecting the file `./impl/psdlab3.xise`. The module `s6base_top` (file `./src/verilog-rtl/s6base_top.v`) is the top level Verilog module containing the whole system represented in figure 1. In the “Hierarchy” pane (top-left) you can browse through the project hierarchy and all the files associated to the design (figure 2).

In addition to the Verilog sources, the file `./src/data/s6base.ucf` (ucf = “*user constraints file*”) contains implementation constraints, as the assignment of the design inputs/outputs to the physical FPGA pins (constraints “`NET … LOC=`”) and a timing constraint specifying the minimum required clock period (“`TIMESPEC…`”). This file is essential for the implementation, otherwise the inputs and outputs will be connected to random FPGA pins with unpredictable results.


ISE Project Navigator (P.68d) - C:\usr\jca\FEUP\FEUP\Aulas\2020-2021\Projeto_
File
Edit
View
Project
Source
Process
Tools
Window
Layout
Design
View: Implementation Simulation

Hierarchy
- psdlab3
- xc6slx45-3csg324
  - s6base_top (s6base_top.v)
    - uart_1 - uart (uart.v)
    - ioports16_1 - ioports16V2018 (ioports16V2018.v)
    - LM4550_controler_1 - LM4550_controler (LM4550_controler.v)
    - s6base.ucf

Figure 2 - The project hierarchy as seen in ISE Project Navigator.

3.1 - Behavioural simulation

After building the RTL modules and verifying them individually (you can assume this has already been done in this project), next design stage is to verify the whole circuit Verilog model at the functional level. For this simulation you will use a simplified testbench (`./src/verilog-tb/s6base_tb.v`, already included in the project) that simulates the complete design described above. In this design the simulation will use only the primary inputs and primary outputs of the circuit, sending the operands and retrieving the results through the serial port. The simulation will be done using the QuestaSim simulator, but now the program will be launched from the ISE project navigator.

In the *Design* window select the view "*Simulation*" and choose "*Behavioural*" for the simulation process (figure 4). In the project hierarchy view, select the testbench module (`s6base_tb`) and execute the process "*Simulate Behavioural Model*". This will launch QuestaSim and run automatically the script `s6base_tb_fdo` that contains the commands necessary to compile the project and run the simulation. You can edit the file `s6base_tb.udo` (udo = "*user do file*") to customize the way the simulation is started. You may also customize the file `./impl/s6base_tb_wave.fdo` with commands to format the waveform window (a basic waveform configuration is available in the script `./impl/wave_functional_sim.do`).

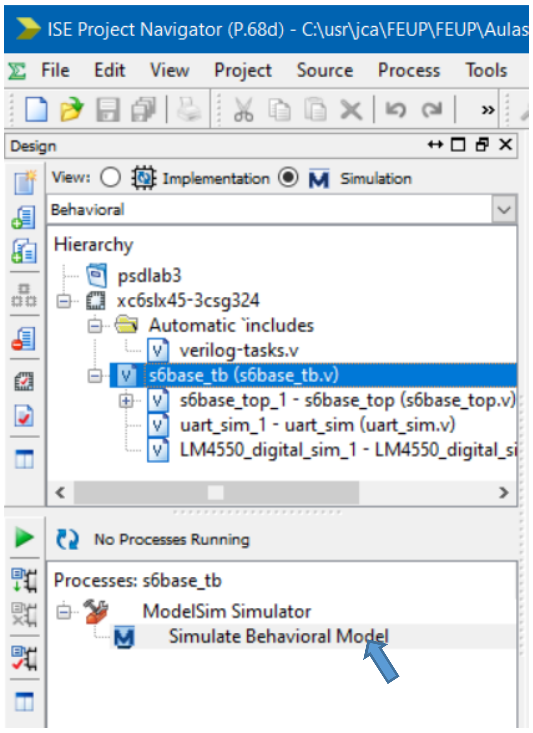


Figure 4 - Selection of the simulation process.

To rerun a simulation you don't need to close and re-invoke QuestaSim from the ISE Project Navigator, unless the hierarchy of the project has been modified (in that case ISE needs to regenerate a new `s6base_tb.fdo` file). If this is the case, just re-execute the command `do s6base_tb.fdo` in the QuestaSim command shell.

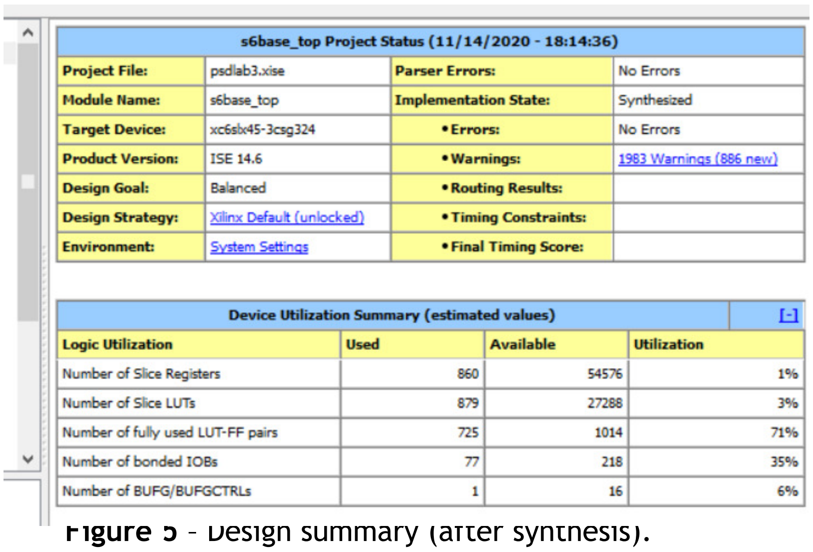
The testbench applies a series of audio samples generated by the script `./matlab/gensimdata.m` from an audio *.wav file. This script creates four files in directory `./simdata`: the audio samples to apply to the left and right channels of the CODEC simulation model (files `audioin_left.hex` and `audioin_right.hex`) and the expected output data from the circuit being designed (files `output_left_golden.hex` and `output_right_golden.hex`). The testbench also includes a self-checking process to compare the audio samples output by the circuit model and the expected data read from the golden files. The testbench also generates the text file `./simdata/audioout.dat` with the audio samples generated by the circuit model during the simulation. The script `./matlab/genoutwav.m` takes that file and generates an audio `*.wav`.

Analyse the files referred above to clearly understand the verification flow. erify the simulation results in the output text window and in the waveforms. Feel free to improve the testbench and write in <u>table 1 any relevant comments about this verification phase</u>.

3.2 - RTL synthesis

In the *design* window, choose the view "*Implementation*", select the module `s6base_top` and run the process "*Synthesize - XST*". Verify the warning messages generated during the synthesis. Note that most of the warnings ere related to unconnected outputs, the subsequent logic "trimming" of the parts of the circuit that produce those unused outputs and merging of flops with equivalent values

The RTL synthesis is driven by an extensive set of parameters that the designer can tune to meet the required design goals. These parameters can be edited by selecting "*Process Properties*" available for each implementation process (use the right-mouse button over "*Synthesize - XST*"). Verify the current configuration for the two first parameters (optimization goal and optimization effort) and analyse the synthesis results in the "Design summary" window. The most relevant results are the design area measured by the number of slice look-up tables (LUTs) and slice registers (flip-flops) and the estimated maximum clock frequency reported at the end of the synthesis report. Figure 5 shows the design summary window after concluding the synthesis process.



s6base_top Project Status (11/14/2020 - 18:14:36)			
Project File:	psdlab3.xise	**Parser Errors:**	No Errors
Module Name:	s6base_top	**Implementation State:**	Synthesized
Target Device:	xc6slx45-3csg324	• **Errors:**	No Errors
Product Version:	ISE 14.6	• **Warnings:**	1983 Warnings (886 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	860	54576	1%
Number of Slice LUTs	879	27288	3%
Number of fully used LUT-FF pairs	725	1014	71%
Number of bonded IOBs	77	218	35%
Number of BUFG/BUFGCTRLs	1	16	6%

Figure 5 – Design summary (after synthesis).

3.3 - Post-synthesis verification

Select the simulation process "*Post-Translate*" and perform a logic simulation. This simulation is very similar to the described in section 3.1 although now you will simulate a fully structural Verilog model generated by the synthesis process. Use a text editor (Notepad++) to see the contents of the Verilog file created by the synthesis `./impl/netgen/translate/s6base_top_translate.v`.

3.4 - Physical implementation (Map and Place&Route)

Execute now the process "*Implement design*". This will run the Translate, Map and Place&Route processes, creating the most accurate model of the circuit that represents the physical implementation on the FPGA. All the logic blocks and interconnections will be now annotated with propagation delays calculated from the models of the logic cells and the voltage and temperature (VT) operating conditions. If this succeeds, the summary window will now present the final utilization of the FPGA resources for the whole design.

Open the report "Post PAR static timing report" (window "Design summary", in the section "Detailed reports") and verify the maximum clock frequency estimated for this implementation. Browse through this report to identify the flop-to-flop patrhs with the longest propagation delays.

3.5 – Post-route verification

Run a "Post-route" simulation, using the same procedure described in section 3.4. Prior to the simulation, a Verilog netlist will be generated and annotated with the logic and net delays defined in a SDF file (*standard delay format*). Both files are created in the directory `./impl/netgen/par/`.

Modify the testbench to increase the clock frequency to 30% above the maximum frequency reported in the static timing report. Running a timing simulation under this condition will probably fail. Note that this simulation only will fail if the set of stimuli applied to the system under simulation activates one of the logic paths that will not support that clock frequency.

3.6 – FPGA configuration and experimentation in the FPGA board

Execute the process "Generate programming file" to generate the final *bitstream* file (`./impl/s6base_top.bit`). This file is the final product of the FPGA design and contains the programming data to configure the FPGA logic blocks and interconnections with your circuit.

Programming the FPGA in the Atlys board requires launching the application Adept provided by the board vendor Digilent. Run this application, choose the bit file and execute "Program".

To experiment your system connect an audio source to the 3.5 mm female jack line-in input (blue connector), connect a headphone set to the black jack (headphones out), run the control application `./sw/bin/lab3-2021.exe`, push the button "Setup codec" and you are ready to play with your circuit. If you are running this application for the first time in your PC, do "Scan serial ports" to detect the serial port where the FPGA board is connected and then "Save serial config".

Annex A

Functional description of module `ioports16V2018.v`

This circuit can be accessed from an external system (for example a PC) using a reduced set of commands that allow to write and read data to/from the 32-bit output and input ports (table A.1). A basic Windows application (`./sw/bin/lab3-2021.exe`, figure A.2) implements these commands. Note that this application displays the raw 32-bit unsigned integer values that have to be converted to the appropriate number format to compare the outputs to the expected results. A Matlab function is provided that already does those numeric conversions.

Table A.1 – Commands implemented by module `ioports16V2018.v`

Function	Command	Data (1)	Operation
RESET	8'b0001_xxxx	none	Set output ports to the reset state 'xxxx' field is not used (2)
WRITE	8'b0010_aaaa	4 bytes, write to output port	Write a 32-bit word to port address 'aaaa' (3)
READ	8'b0011_aaaa	4 bytes, read from input port	Read a 32-bit word from port 'aaaa' (4)

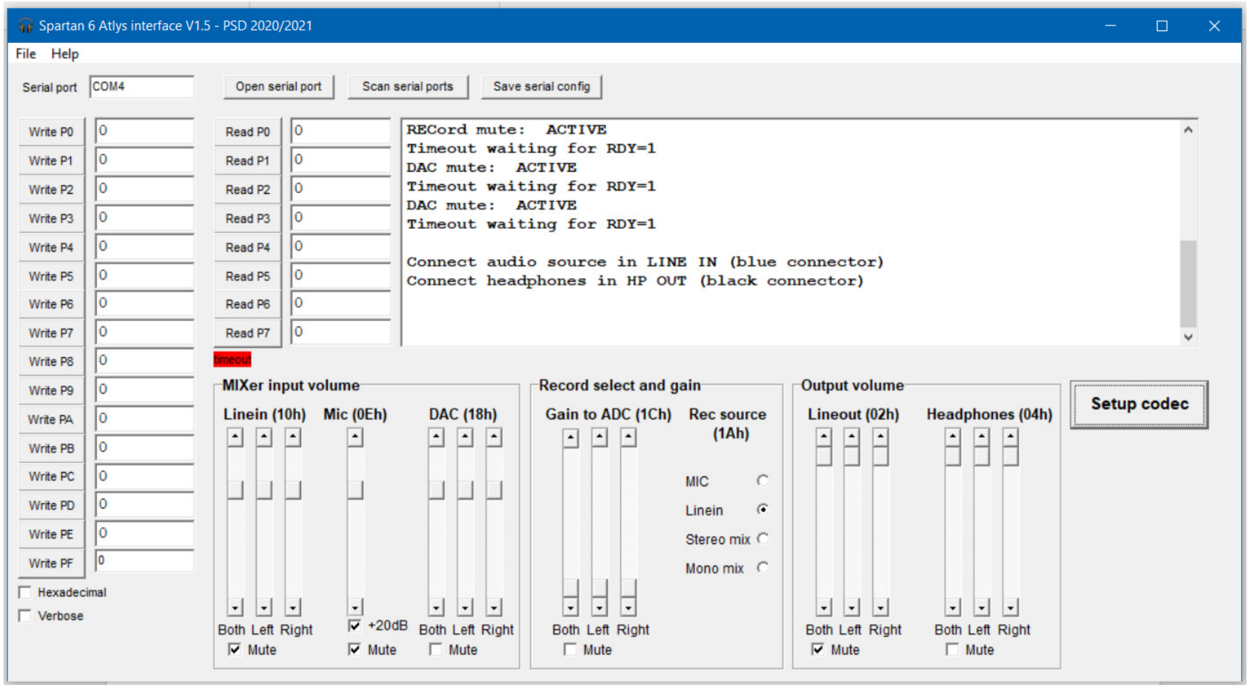
Notes:

(1) The 32-bit word is written/read in big endian format (MSByte first), following the 1-byte command.

(2) The reset state of each output port is 32'd0 by default. This can be configured at synthesis time by redefining the module parameters `INIT_Pxx`, where "xx" is the port address in decimal ("00" to "15").

(3) Output port 15 (4'b1111) only keeps the data written during a certain number of clock cycles (useful to generate N-clock cycle pulses). The number of clock cycles is set by parameter `OUT15_DELAY_OUT` and defaults to 1.

(4) Module `ioports16V2018` only implements 8 input ports (addresses 0 to 7). Reading from port 15 (4'b1111) returns a 32-bit identification number set as the parameter ATLYS_HWID (defaults to 32'h2020_2021)



Serial port: COM4

Write P0–PF: 0

Read P0–P7: 0

```
RECord mute:   ACTIVE
Timeout waiting for RDY=1
DAC mute:   ACTIVE
Timeout waiting for RDY=1
DAC mute:   ACTIVE
Timeout waiting for RDY=1

Connect audio source in LINE IN (blue connector)
Connect headphones in HP OUT (black connector)
```

Rec source: MIC [✓ Linein] Stereo mix Mono mix

Linein (10h): [✓ Mute]; Mic (0Eh): [✓ +20dB] [✓ Mute]; DAC (18h): Mute; Gain to ADC (1Ch): Mute; Lineout (02h): [✓ Mute]; Headphones (04h): Mute

Hexadecimal; Verbose

Figure A.2 – The software application for interfacing with the `ioports16V2018.v` module, including a set of controls to setup the audio CODEC.