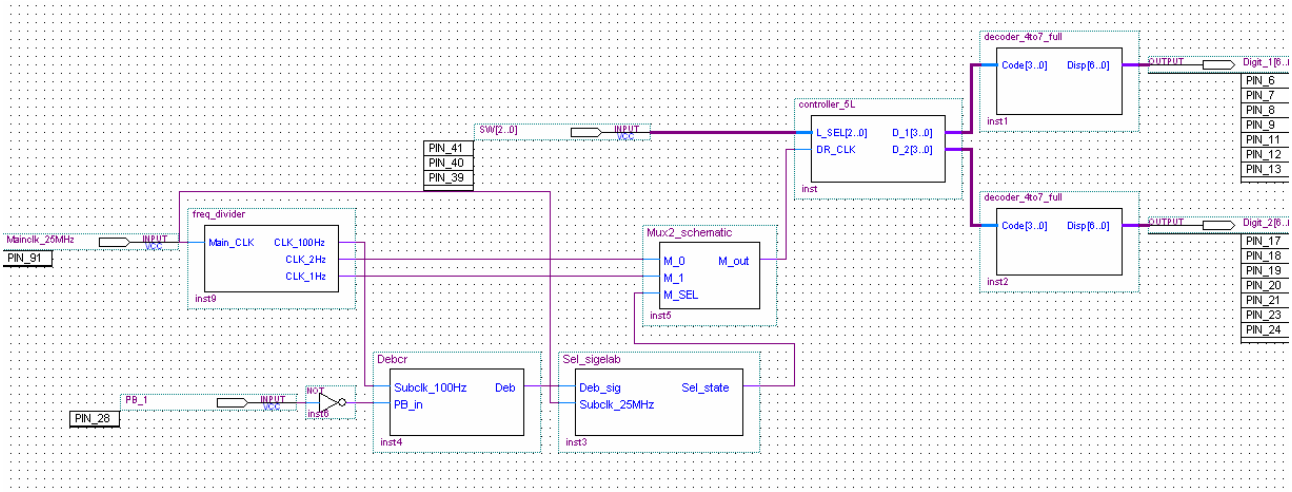
TECHNICAL UNIVERSITY OF KOSICE

Faculty of Electrical Engineering and Informatics
Department of Electronics and Multimedia Communications

ELECTRONIC SYSTEMS WITH FPGA CIRCUITS

DISPLAY ANIMATION

PERFORMING FUNCTIONAL AND TIMING SIMULATION WITH MODELSIM-ALTERA SOFTWARE



Professors:

Milos Drutarovsky
Pavol Galayda
Pavol Pavelka

Student:

Attilio Piacente

INDEX

1. INTRODUCTION

The purpose of this work is to introduce the simulation software "ModelSim-Altera Web Edition 6.1g" distributed by the Mentor Graphics Corporation and to use it to perform functional and timing simulation of my previous work.
Before reporting the results obtained by using this software for both functional and timing simulation a brief information about ModelSim-Altera software will be given in the first three chapters.
The ModelSim-Altera software is a special software, developed by the Mentor Graphics Company, to use for designs made with QuartusII software.
The main differences between ModelSim-Altera and ModelSim PE or ModelSim SE are:

- ModelSim-Altera Edition software is licensed as a single language, either VHDL or Verilog HDL for each active subscription and supports only Altera gate-level libraries. On the other hand ModelSim PE and ModelSim SE software are dual-language simulators, in fact it is possible to simulate designs containing either Verilog HDL, VHDL, or both;
- The ModelSim-Altera software includes all ModelSim PE features including behavioural simulation, HDL testbenches, and Tool Command Language (TCL) scripting;
- A drawback of the ModelSim-Altera software is that it is slower than the ModelSim SE software.

Two editions of this software are available, on the web, one is the "ModelSim-Altera Web Edition" software and the other is the "ModelSim-Altera Edition" software. These two editions are the same except for some differences:

- ModelSim-Altera Web Edition's performance is one half that of ModelSim-Altera Edition;
- ModelSim-Altera Web Edition has a line limit of 10,000 lines compared to the unlimited number of lines allowed in the ModelSim-Altera Edition.

Despite the limitations of the "ModelSim-Altera Web Edition" software, in respect of the "ModelSim-Altera Edition", for this work the performances of the former one are enough to perform functional and timing simulation. More detailed information are available in [11].

2. DOWNLOAD, INSTALLATION AND REGISTRATION OF SOFTWARE

A brief information, for the Windows XP platform, about the download, the installation and the registration of the "ModelSim-Altera Web Edition 6.1g" software is given in this section.
Since the "Altera QuartusII 6.1 Web Edition" version has been used for the design of my project it is possible to see, from the Altera's web site (*https://www.altera.com/support/software/download/eda_software/modelsim/msm-index.jsp*), that the compatible version of ModelSim-Altera is the "ModelSim-Altera Web Edition 6.1g".
Before installing this software it is necessary that "Altera QuartusII 6.1 Web Edition" is already installed.
The web page from which it is possible to download the "ModelSim-Altera Web Edition 6.1g" software is "*https://www.altera.com/support/software/download/eda_software/modelsim/dnl-msim-61g.jsp*".
Once downloaded the installation is completely guided by the installation software.

At this point it is necessary to request, in a completely free manner, the "QuartusII Web Edition and ModelSim-Altera Web Edition Licenses", this could be done by accessing the following web site "*https://www.altera.com/support/licensing/free_software/lic-q2web.jsp*" and complying the module. Once finished this procedure, an e-mail will be send to the specified e-mail address. A file with "*.dat" extension is attached to this e-mail. To obtain a successful registration the following points have to be performed:

- The "*.dat" file has to be put inside the QuartusII program folder;
- A new system variable has to be set in the environment variable dialog box of the operating system. The name of the system variable is "LM_LICENSE_FILE" and the path, that reaches the "*.dat" file, to specify is, for example, "*C:\<altera_installation_folder_path>\000E350D07BD__0_58686351430922.dat*" as shown in figure 1:

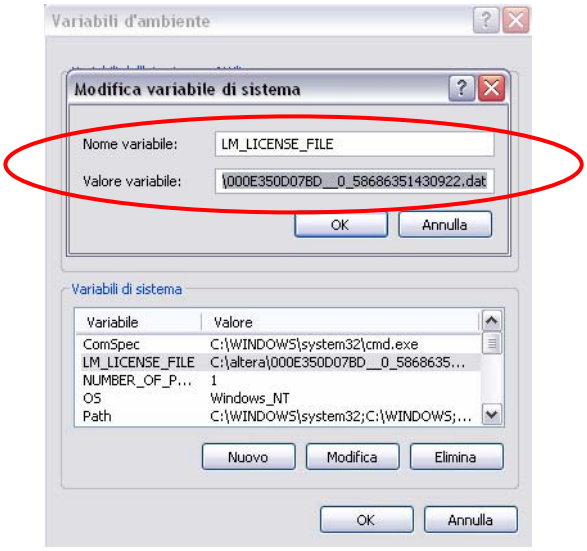


Figure 1: System variable settings

Once completed all these procedures it is possible to start ModelSim-Altera in an independent manner or it could be started directly from QuartusII by using the NativeLink feature. The latter procedure will be explained later.

3. ALTERA DESIGN FLOW WITH MODELSIM-ALTERA AND QUARTUSII SOFTWARE

3.1. Introduction

A brief information about the Altera design flow with ModelSim-Altera and QuartusII software is given in this section.
There are two ways to use the ModelSim-Altera software:

- Using the NativeLink feature in the QuartusII software that facilitates the transfer of information between the QuartusII software and EDA tools, allowing to run ModelSim within the QuartusII software.
- Using ModelSim-Altera in an independent way, using the Graphical User Interface (GUI) or the Tool Command Language (TCL) to setup and execute simulations.

In this sections both methods will be explained for both functional and timing simulations.
The design flow is reported in figure 2. As it is possible to see usually a post synthesis simulation is performed that verifies the functionality of a design after synthesis has been performed, this type of simulation is not discussed in this work.

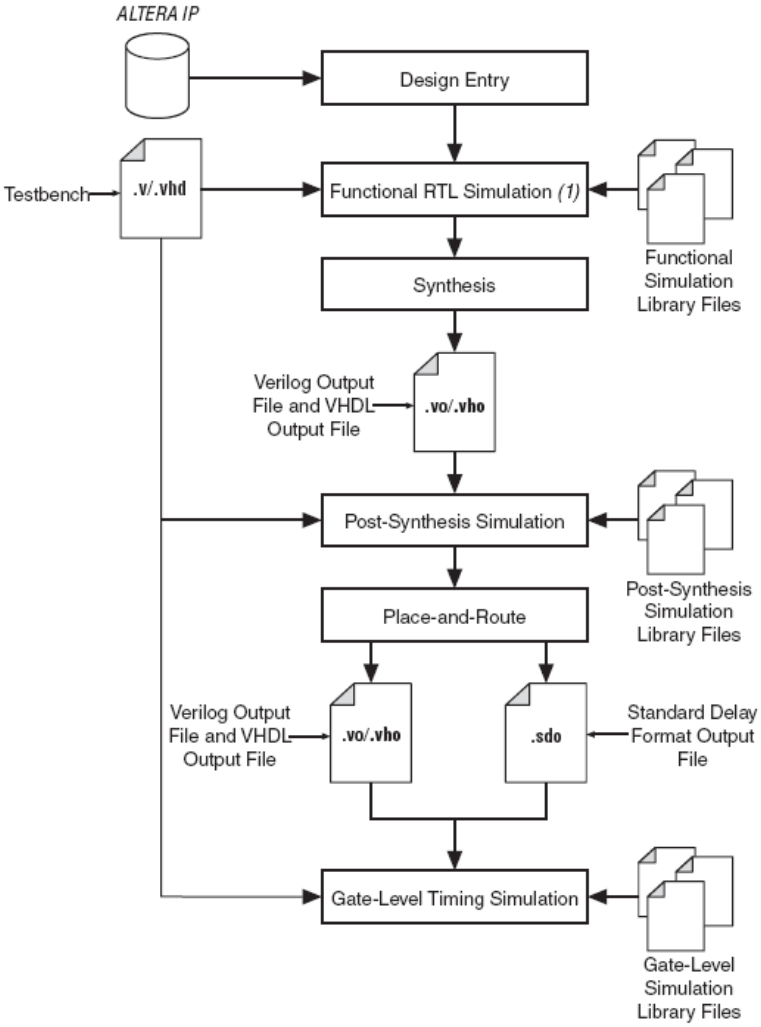


Figure 2: Altera Design Flow

A functional Register Transfer Level (RTL) simulation is performed before a Gate Level simulation or Post Synthesis simulation. The main task of the functional RTL simulation is to verify the functionality of the design before synthesis and place-and-route.
The Gate Level timing simulation is a post place-and-route simulation. Its task is to verify the operation of the design after the worst-case timing delays have been calculated.
A very important feature of ModelSim-Altera software is the very robust debug environment that allows the user to discover the problem in case of inexact results.

3.2. Using ModelSim-Altera with NativeLink

The NativeLink feature allows to set up the QuartusII software to automatically launch third-party EDA software, so that ModelSim-Altera software and QuartusII software can interact each other through an EDA interface. In particular Electronic Design Automation (EDA) is the category of tools for designing and producing electronic systems ranging from printed circuit boards (PCBs) to integrated circuits.

Some functionalities, such as functional simulation and timing simulation can be directed to run automatically as part of a compilation from within the QuartusII software.
So finally using the NativeLink feature of QuartusII it is possible to execute simulations in ModelSim-Altera software directly from the QuartusII software.
Once created a project within QuartusII, to be able to perform functional and timing simulation the following steps have to be accomplished:

- Setup the NativeLink;
- Setup and Perform functional simulation;
- Setup and Perform timing simulation;
- Create and Setup a testbench;

3.2.1. Setup the NativeLink

With this procedure the user could specify which EDA tool the QuartusII software has to use to perform a simulation. To make able QuartusII to use the ModelSim-Altera software it is necessary to specify the path to the simulation tool. This is done by following these steps from within the QuartusII software:

- Open the **Option** dialog box from the **Tool** menu;
- From the **Category** list select **General** and the **EDA tool options** (highlighted in blue in figure 3);
- Specify for the desired **EDA Tool**, **ModelSim-Altera**; the path to reach the executable files of the tool, for ModelSim-Altera are stored in win32aloem (highlighted in red in figure 3);
- Click **ok**.

Figure 3 reports the Options dialog box:

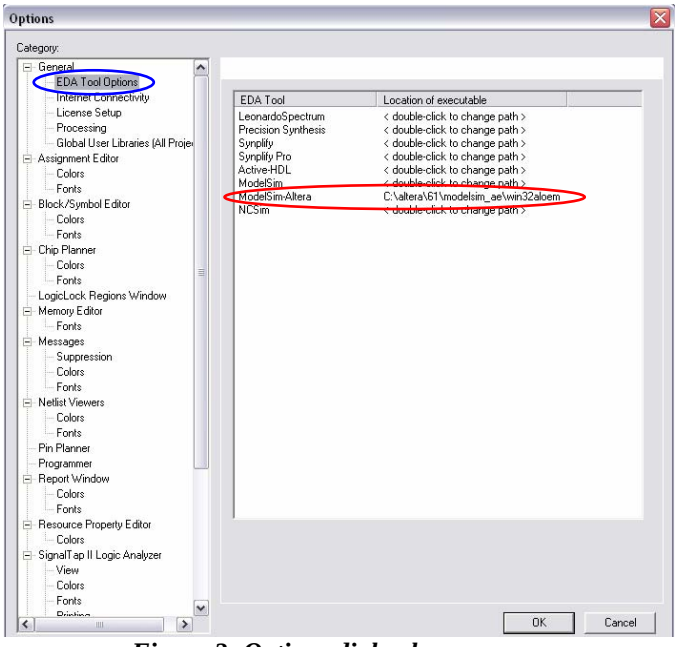


Figure 3: Options dialog box

This setting has to be performed only the first time after the installation of the QuartusII software.

3.2.2. Setup and Perform functional simulation

To perform a functional RTL simulation with the ModelSim-Altera software from the QuartusII software, the following steps has to be accomplished:

- Open the **EDA Tool Settings** dialog box from the **Assignment** menu;
- Select **Simulation** from the **EDA Tool Settings** list (highlighted in blue in figure 4);
- Select **ModelSim-Altera** from the **Tool name** box (highlighted in green in figure 4);
- In the **EDA Netlist writer** options select **VHDL** as **Format for output netlist** (highlighted in red in figure 4);
- In the **NativeLink settings** box select the appropriate testbench file; creation an setup of a testbench file is explained in 3.2.4 (highlighted in red dashed line in figure 4);
- Click **ok**;
- From the **Processing** menu point on **Start** and click on **Start Analysis and Elaboration**; This command collects all file name information and builds a design hierarchy in preparation for simulation;
- To run the simulation from the **Tool** menu point on **EDA Simulation Tool** and click on **Run EDA RTL Simulation**; this command runs automatically ModelSim-Altera software, compiles all design files and performs the simulation.

Figure 4 reports the Settings dialog box:

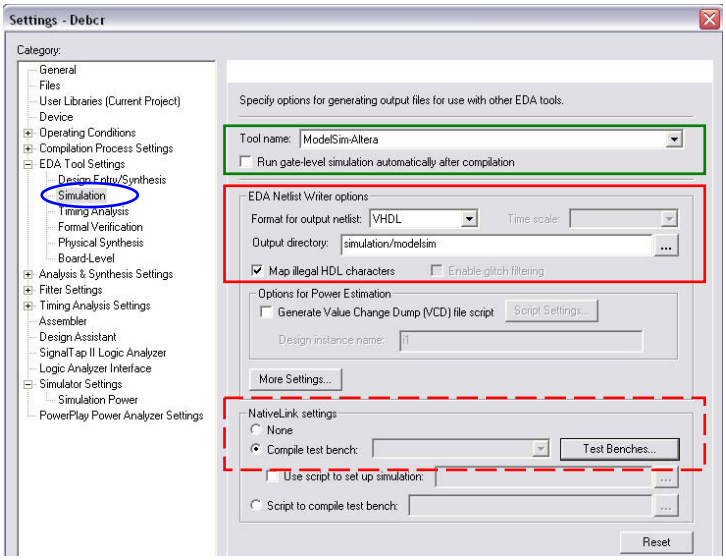


Figure 4: Settings dialog box for functional simulation

3.2.3. Setup and Perform timing simulation

To perform a gate-level timing simulation with the ModelSim-Altera software from the QuartusII software, the following steps has to be accomplished:

- Open the **EDA Tool Settings** dialog box from the **Assignment** menu;
- Select **Simulation** from the **EDA Tool Settings** list (highlighted in blue in figure 5);

- Select **ModelSim-Altera** from the Tool name box (highlighted in green in figure 5);
- Turn on **Run Gate Level Simulation** automatically after compilation (highlighted in green in figure 5);
- In the **NativeLink settings** box select the appropriate testbench file, creation and setup of a testbench file is explained in 3.2.4 (highlighted in red in figure 5);
- Click ok;
- From the **Processing** menu, point on **Start** and click on **Start Compilation**; during this compilation a simulation netlist for the design is also created;
- After the compilation the ModelSim-Altera software, runs automatically the simulation.

Figure 5 reports the Settings dialog box:

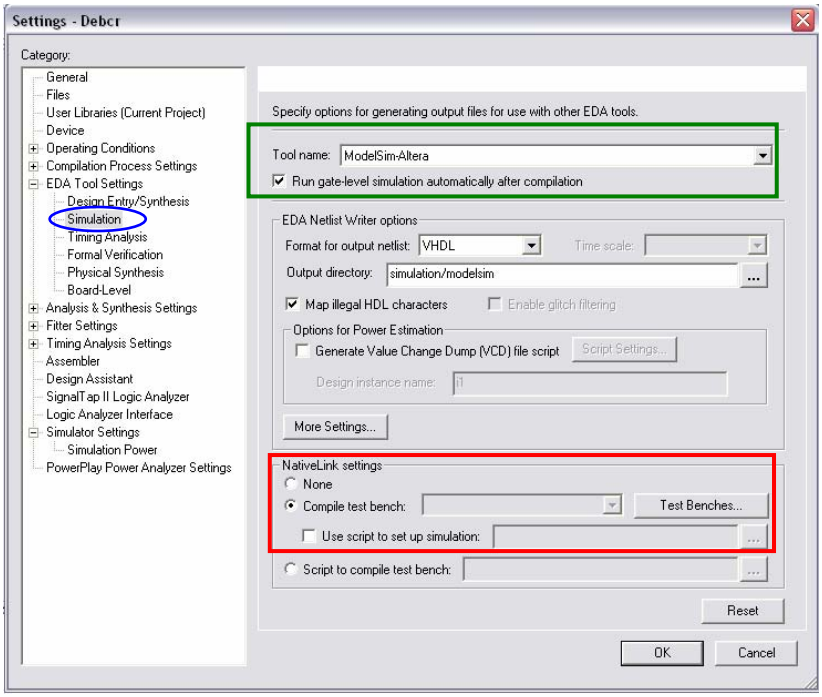


Figure 5: Settings dialog box for timing simulation

3.2.4. Create and Setup a testbench

To perform either a functional or a timing simulation a testbench file is necessary. If a vector waveform file (*.vwf) has been created using the QuartusII software it is possible to create, from this file, a Verilog HDL or VHDL testbench (.vht) file.
This generated testbench includes the behaviour of the input stimulus and applies it to the instantiated top-level FPGA design. To create this file it is sufficient to open the vector waveform file and to export it using the **Export** tool available on the **File** menu.
One problem encountered during the simulation with this method is that the QuartusII software doesn't translate the vector waveform file in an appropriate manner in a VHDL testbench (.vht) file. This problem occurs when the simulation time exceeds the 2^{31}-1 picoseconds. In fact once started the simulation in the ModelSim-Altera software the reported error is like :

** Error: <file_containing_error_location> <error_location> Integer value exceeds INTEGER'high.

To overcome this problem the only solution is that, once created the VHDL testbench (.vht) file, to edit manually the file by changing all time values from picoseconds to a higher scale value like microseconds or milliseconds.
At this point it is necessary to setup the testbench, this is done by following these steps:

- Open the **EDA Tool Settings** dialog box from the **Assignment** menu;
- Select **Simulation** from the **EDA Tool Settings** list;
- Under the **NativeLink settings** select **Compile testbench** and click on **Test Benches** (highlighted in red in figure 5);
- Using this dialog box create a **New** test bench (highlighted in red dashed line in figure 6);
- In the **Test bench name** field write the test bench setup name that identifies the different testbench setups using this format *<Test_Bench_Name.vht>* (highlighted in green in figure 6);
- In the **Test bench entity** box, type in the top-level testbench entity name in the following form: *<Vector_Waveform_File_name*_vhd_vec_tst> (highlighted in blue in figure 6);
- In the **Instance box** type "*i1*" (highlighted in red in figure 6);
- Write the simulation time in the **Run** box (highlighted in black in figure 6);
- Add the previously exported VHDL testbench file (.vht) to the testbench files section using the **Browse** button (encircled in green in figure 6).

Now the test bench files are set and ready to use for functional and timing simulation. In figure 6 the Test Benches and the New Test Bench Settings dialog boxes are reported:

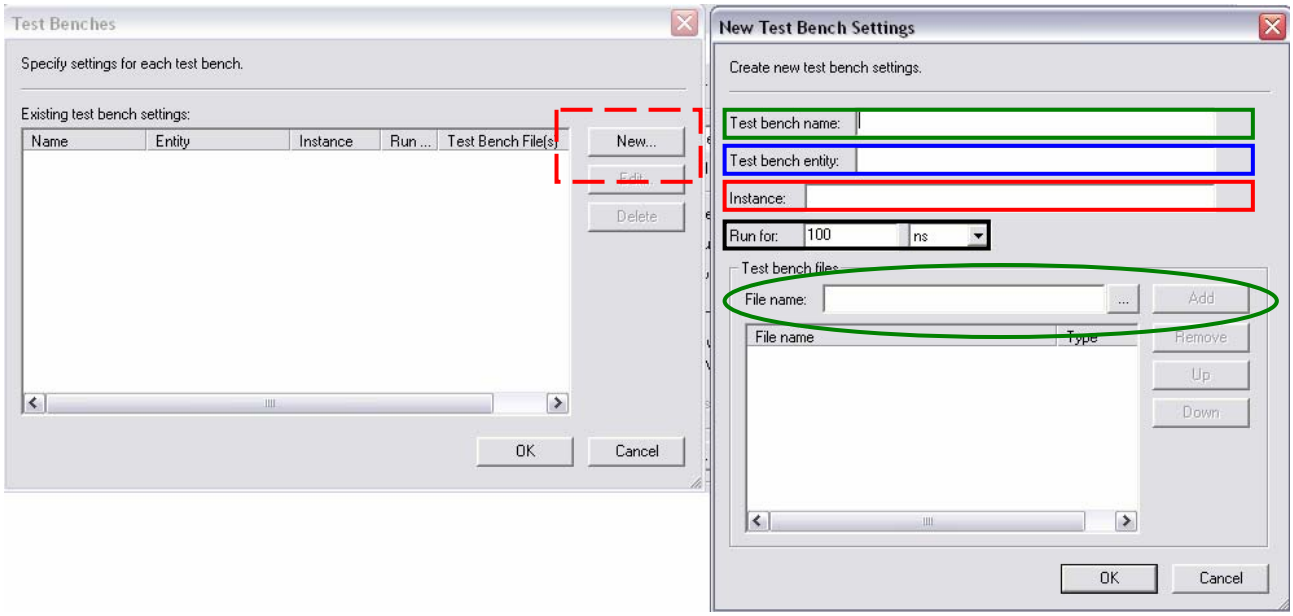


Figure 6: Settings dialog box for timing simulation

As it is possible to see, using the NativeLink feature, all files, that are needed for a simulation in the ModelSim-Altera software, are created and passed to the software directly by the QuartusII software. This way of working is very comfortable and it reduces the design time, because the designer is not obligated to write additional files if he wants to use a different software from that available in QuartusII software for simulations.

3.3. ModelSim-Altera

There are two ways to perform functional and timing simulations within ModelSim software, one is by using the ModelSim GUI and the other is to use TCL scripts. In this section both ways are explained together so that it's possible to see which TCL command corresponds to a GUI command. More detailed information about the TCL will be given in the next chapter.
With ModelSim it is possible to perform functional and timing simulation either for Verilog HDL or VHDL designs. In this section only the procedure to simulate VHDL designs is explained.
The simulation flow that is followed in both methods consist of the following steps:

- Create New Project and add Design files;
- Create Libraries;
- Map to Libraries;
- Compile source code and testbenches;
- Load the Design;
- Add Design stimulus;
- Run the simulation.

By using the ModelSim-Altera software the second and the third steps are already done, so they will not be explained.
Before creating a new project within ModelSim-Altera it is necessary to create a project folder. It is very important to take in account that it is necessary to create this folder in a path in which no spaces are present, otherwise the ModelSim-Altera is not able to perform its tasks.
Once opened the ModelSim-Altera software it's necessary to set the new working directory:

- Using the GUI:
  - Open the **Change Directory** form the **File** menu (highlighted in red in figure 7);
  - Select the exact path to reach the previously created folder (highlighted in blue in figure 7);
  - Click **OK**.

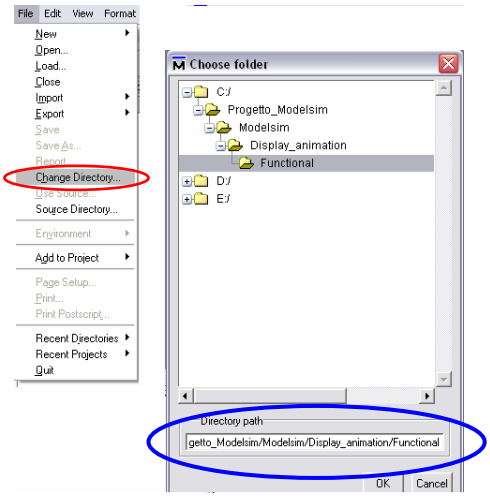


Figure 7: GUI-Changing directory

- Using TCL:
  - Type the following command in the command prompt: "*cd <path_of_the_project_directory>*".

Once set the working directory it's possible to perform either a functional or a timing simulation.

3.3.1. Functional Simulation

The previously mentioned steps in 3.3, for a functional simulation, will be explained in this section. Before starting the explanation of the presented steps it is necessary to have, for the functional simulation all VHDL files of a generic project, implemented with the Quartus II software. For example if a project implemented in QuartusII contains various subunits constructed by using VHDL code, Megafunctions or schematics, each subunit has to be translated in VHDL. This could be done directly from QuartusII software by pointing on **Create/Update** in the **File** menu and clicking on **Create HDL design file for current file**, for each design unit.
Once created these files it is necessary to put them in the working folder of the ModelSim-Altera software previously created.

- **Create New Project and add Design files;**
  - Using the GUI:
    - Point on **New** in the **File** menu and click on **Project**;
    - Compile the **Create Project** dialog box;
    - Click **OK**;
    - Point on **Add to Project** in the **File** menu and click on **Existing Files**;
    - Using the **browse** button search the VHDL files to add to the project.
  - Using TCL:
    - To create a new project write:
      "project new <path_of_the_project_directory> <name_of_the_project>";
    - To add new files write:
      "project addfile <file_name.vhd>".
- **Compile source code and testbenches:**
  - Using the GUI:
    - Click on **Compile All** in the **Compile** menu.
  - Using TCL:
    - To compile every VHDL file in the work folder write for everyone:
      "vcom -2002 -explicit -novitalcheck -no1164 -novital <file_name.vhd>".
- **Load the Design:**
  - Using the GUI:
    - Click on **Start Simulation** on the **Simulation** menu;

- In the **Start Simulation dialog box** click on **work** (highlighted in red in figure 8) and select the top level entity (highlighted in blue in figure 8) and click ok.

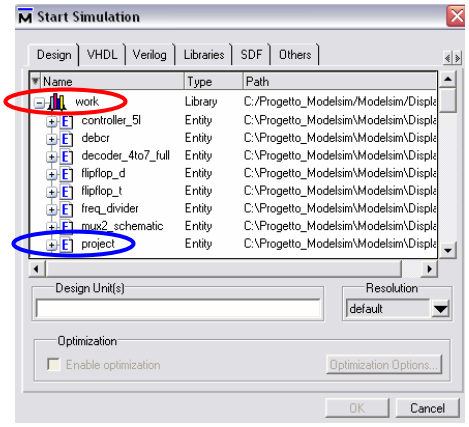


Figure 8: GUI-Starting simulation

- Using TCL:
  - To load the design write:
    "*vsim work.<top_level_entity_name>*".

➢ **Add Design stimulus:**

- Using the GUI:
  - Point on **Debug Windows** in the **View** menu and click on **Wave**;
  - Drag signals to monitor and input signals from the **Object Window** and drop them in the **Wave window**;
  - Create stimulus by using the appropriate command (highlighted in blue in figure 9) in the sliding menu that compares by the right clicking on the desired signal.

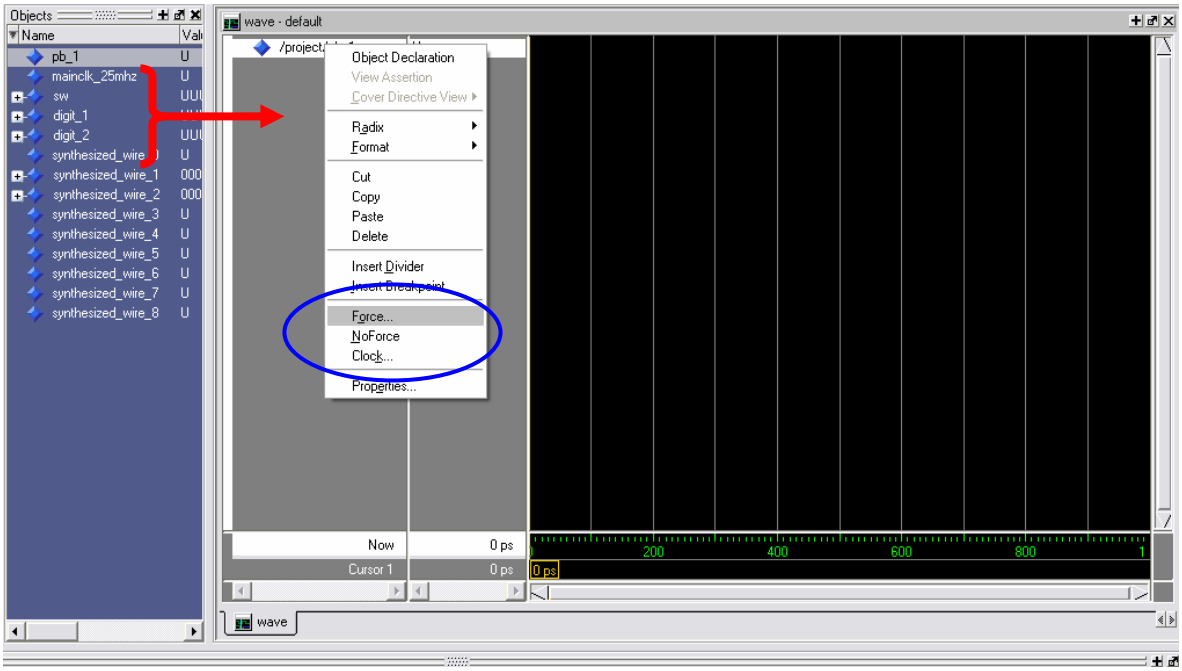


Figure 9: GUI-Adding design stimulus

- Using TCL:
  - To add signal to the Debug Window write:
    "add wave -noupdate -format Logic </project_file_name/signal_name>";
  - To create stimulus write:
    "force -freeze sim: /<name_of_the _project_file >/ <signa_namel> <start_value> <delay>, <final_value> {duration}, -r <period>".

- **Run the simulation:**

  - Using the GUI or TCL:
    - Write in the command Prompt window:
      " run <desired_simulation_time_period>".

3.3.2. Timing Simulation

The previously mentioned steps in 3.3, for a functional simulation, will be explained in this section. To perform Gate Level timing simulation, the ModelSim-Altera software requires information about how the design was placed into device specific architectural blocks. To transfer this information to the ModelSim-Altera software the QuartusII software is able to create a type of file in the form of VHDL Output File (.vho) for VHDL designs. Beside this information other data regarding timing information is needed and it is stored in the Standard Delay Format Output File (.sdo), which annotates the delays for the elements found in the VHDL Output File.
To create these two files, from the QuartusII software, once the project is compiled, it is sufficient to follow these steps:

- Click on **EDA Tool Settings** on the **Assignments** menu;
- From the **Category list** select **EDA Tool Settings** and click on **Simulation**;
- In the **Tool name list** select **ModelSim-Altera**;
- In the **EDA Netlist Writer option** box select **VHDL** as **Format for output netlist**;
- Click **OK**;
- point on **Start** in the **Processing** menu and click on **EDA Netlist Writer**.

Once created these files, that are available in the folder "simulation/modelsim" of the working directory, it is necessary to put them in the working folder of the ModelSim-Altera software previously created.

- **Create New Project and add Design files;**

  - Using the GUI:
    - Point on **New** in the **File** menu and click on **Project**;
    - Compile the **Create Project** dialog box;
    - Click **OK**;
    - Point on **Add to Project** in the **File** menu and click on **Existing Files**;
    - Using the **browse** button search the ".vho" file to add to the project.

- Using TCL:
  - To create a new project write:
    "project new <path_of_the_project_directory> <name_of_the_project>";
  - To add new files write:
    "project addfile <file_name.vho>".

➢ **Compile source code and testbenches:**

- Using the GUI:
  - Click on **Compile All** in the **Compile** menu.

- Using TCL:
  - To compile every VHDL file in the work folder write for everyone:
    "vcom -2002 -explicit -novitalcheck -no1164 -novital <file_name.vho>".

➢ **Load the Design:**

- Using the GUI:
  - Click on **Start Simulation** on the **Simulation** menu;
  - In the **Start Simulation dialog box** click on **SDF** tab and click on **Add** (highlighted in red in figure 10);
  - Search the ".sdo" file by using the **browse** option;
  - Click **OK**;
  - Return on the **Design** tab and click on the top level entity (highlighted in blue in figure 8) in the **work** library (highlighted in red in figure 8);
  - Click **OK**.

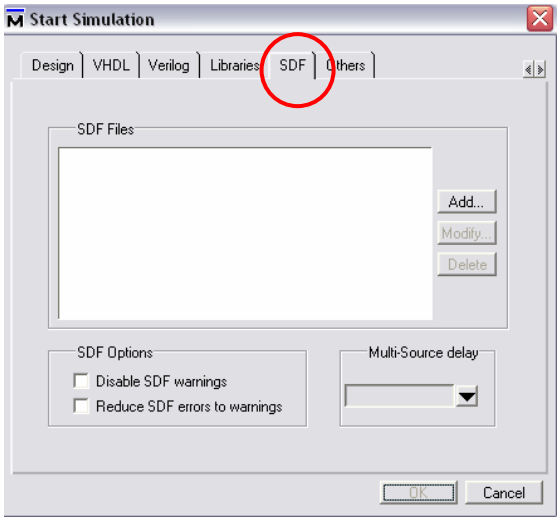


Figure 10: GUI-Starting timing simulation

- Using TCL:
  - To load the design write:
    "vsim -sdftyp <file_name_vhd.sdo> work.<file_name>";

- **Add Design stimulus:**

  - Using the GUI:
    - Point on **Debug Windows** in the **View** menu and click on **Wave**;
    - Drag signals to monitor and input signals from the **Object Window** and drop them in the **Wave window**;
    - Create stimulus by using the appropriate command in the sliding menu that compares by the right clicking on the desired signal;

  - Using TCL:
    - To add signal to the Debug Window write:
      "*add wave -noupdate -format Logic </project_file_name/signal_name>*";
    - To create stimulus write:
      "*force -freeze sim: /<name_of_the _project_file >/ <signa_namel> <start_value> <delay>, <final_value> {duration}, -r <period>*".

- **Run the simulation:**

  - Using the GUI or TCL:
    - Write in the command Prompt window:
      "*run <desired_simulation_time_period>*".

For more detailed information about GUI and TCL see respectively [5] and [8].

4. SIMULATING THE PROJECT:"DISPLAY ANIMATION"

4.1. Introduction

The procedure of performing functional and timing simulation of the project "Display Animation" created with QuartusII software is presented in this chapter.
The TCL scripts will be adopted to perform these simulations, without using neither the NativeLink feature neither the ModelSim-Altera GUI commands.
The functional simulation will be reported in paragraph 4.2. This simulation was performed on the entire design.
The timing simulation will be reported in paragraph 4.3, this simulation was performed on a simplified design, in fact because of the great amount of time needed by the simulator the frequency divider block was not included in the test design.
Before explaining the two simulation procedures it's useful to describe in a briefly manner the design block diagram that is reported in figure 11:

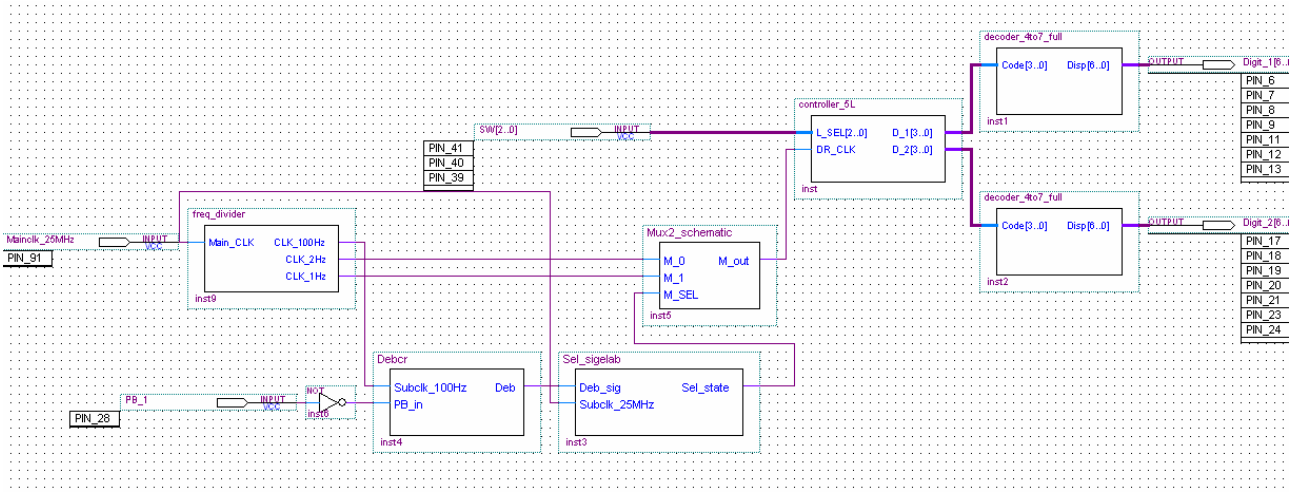


Figure 11: Design block diagram

The input ports of the design are:

- Mainclk_25MHz;
- PB_1;
- SW[2..0].

The output ports are:

- Digit_1[6..0];
- Digit_2[6..0].

The building blocks of the design are:

- Freq_divider, which was written in VHDL code;
- Debcr, which was created by using a schematic file in which a Megafunction is used (SHR_RR);
- Sel_sigelab, which was created by using a schematic file in which two different Megafunctions were used (FlipFlop_D and flipflop_T);
- Mux2_schematic, which was implemented by using a schematic;
- Controller_5L, which was created by using VHDL code;
- Decoder_4to7_full, which was created by using VHDL code.

In both simulation procedure a ".do" file (named "Display_animation.do" for the functional simulation and "Display_animation_sp.do" for the timing simulation) will be written to permit to execute the simulation, in the ModelSim-Altera software, in a completely automatic manner after setting the working directory. This file refers to other two files named "SigDec.do" and "Signals.do" that contain respectively, add signal and setting statements for the wave window, and the stimulus signals for the simulations. The content of these files is explained in the next sections.

4.2. Functional simulation

To perform this type of simulation it is necessary to create a VHDL file of the top level entity of the project and the VHDL files of every subunit used in the project, that were listed in 4.1.
This task could be accomplished directly from the QuartusII software in the following manner:

- For the top level entity and the subunits created with schematic it is sufficient to point on **Create/Update** in the **File** menu and click on **Create HDL design File for current File**;
- For the subunits created with the **MegaWizard Plug-In Manager** it is sufficient to turn on, in the **Summary** tab of the dialog box, the **VHDL component declaration file**;

Once these files are all available they have to be copied in a folder that will be used as working directory for the ModelSim-Altera software.
At this point, once started the ModelSim-Altera software the working directory has to be set by writing on the command prompt window the following code:

"cd C:/Progetto_Modelsim/Modelsim/Display_animation/Functional"

Where the working directory in which all VHDL files were copied is "Functional".
Now it is possible to create a new project in this software the code used to this purpose is :

"project new C:/Progetto_Modelsim/Modelsim/Display_animation/Functional Project"

Where "Project" is the project name.
To add all the VHDL design files previously created to the project the following command lines were used:

"project addfile SHR_RR.vhd
project addfile flipflop_T.vhd
project addfile FlipFLop_D.vhd
project addfile Sel_sigelab.vhd
project addfile freq_divider.vhd
project addfile Mux2_schematic.vhd
project addfile decoder_4to7_full.vhd
project addfile Debcr.vhd
project addfile controller_5L.vhd
project addfile Project.vhd"

The following code is used to compile all these added files in the working directory:

"vcom -2002 -explicit -novitalcheck -novital SHR_RR.vhd
vcom -2002 -explicit -novitalcheck -novital flipflop_T.vhd
vcom -2002 -explicit -novitalcheck -novital FlipFLop_D.vhd
vcom -2002 -explicit -novitalcheck -novital Sel_sigelab.vhd
vcom -2002 -explicit -novitalcheck -novital freq_divider.vhd
vcom -2002 -explicit -novitalcheck -novital Mux2_schematic.vhd
vcom -2002 -explicit -novitalcheck -novital decoder_4to7_full.vhd
vcom -2002 -explicit -novitalcheck -novital Debcr.vhd
vcom -2002 -explicit -novitalcheck -novital controller_5L.vhd
vcom -2002 -explicit -novitalcheck -novital Project.vhd"

Where the ***"-2002"*** argument states that the compiler is to support VHDL-2002; ***"-explicit"*** directs the compiler to resolve ambiguous function overloading by favouring the explicit function definition over the implicit function definition; ***"-novitalcheck"*** Disables Vital level 1 checks and

also Vital level 0 checks; ***"-novital"*** causes vcom to use VHDL code for VITAL procedures rather than the accelerated and optimized timing and primitive packages built into the simulator kernel.
The following code is used to start simulation:

"vsim work.Project"

Where the command vsim invokes the VSIM simulator.
To open the Wave window, in which the stimulus signals and the output signals will be displayed, the following code is used:

"view signals wave"

At this point it is useful to use a separate file where the add signal statements and the setting statements for the wave window are specified. This file is named "SigDec.do" and contains the following code :

```
"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /project/pb_1
add wave -noupdate -format Logic /project/mainclk_25mhz
add wave -noupdate -format Logic /project/sw
add wave -noupdate -format Logic /project/digit_1
add wave -noupdate -format Logic /project/digit_2
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {6300ms}"
```

The code written in blue is used to handle the messaging on the command prompt window; the code written in green is used to drag signals from the objects window and drop them in the wave window; the code written in red is used to setup the wave window and the cursors.
The "SigDec.do" file is launched from the command prompt window by writing this code:

" do SigDec.do"

It is more comfortable to create the stimulus signals in a separate file that is named "Signals.do" and contains the following code:

"restart -f
force -freeze sim:/project/pb_1 1 0, 1 {6300 ms}
force -freeze sim:/project/mainclk_25mhz 1 0, 0 {19850 ps} -r 39700ps
force -freeze sim:/project/sw(2) 0 0, 1 {1500 ms}
force -freeze sim:/project/sw(2) 0 3800ms, 0 {6300 ms}
force -freeze sim:/project/sw(1) 0 0, 1 {3800 ms}
force -freeze sim:/project/sw(0) 0 0, 1 {3800 ms}
run 6300 ms"

The code written in blue reloads the design elements and sets the simulation time to zero; the code written in green specifies the stimulus signals.
For example a clock signal could be created by using the following code:

"*force -freeze sim:/<name_of_the_project_file>/<signal_name> <start_value> <delay>, <final_value> {duty_cycle}, -r <signal_period>"*

where "*freeze"* is used to freeze the signal at the specific value "<start_value>" after a specified delay time "*<delay>*" until it is forced again or until it is unforced with a noforce command; "*<final_value>*" is the value at which the signal has to be forced after a specified time "*{duty_cycle}*", "*-r"* repeats the force command, "*<signal_period>*" is the time at which to start the repeating cycle;
The code written in red starts the simulation for the specified time duration, if no time unit is specified the default time unit is picoseconds.
The "Signals.do" file is launched from the command prompt window by writing this code:

" do Signals.do"

The simulation results for the stimulus signals specified in the "Signals.do" file is reported in the figure 12.

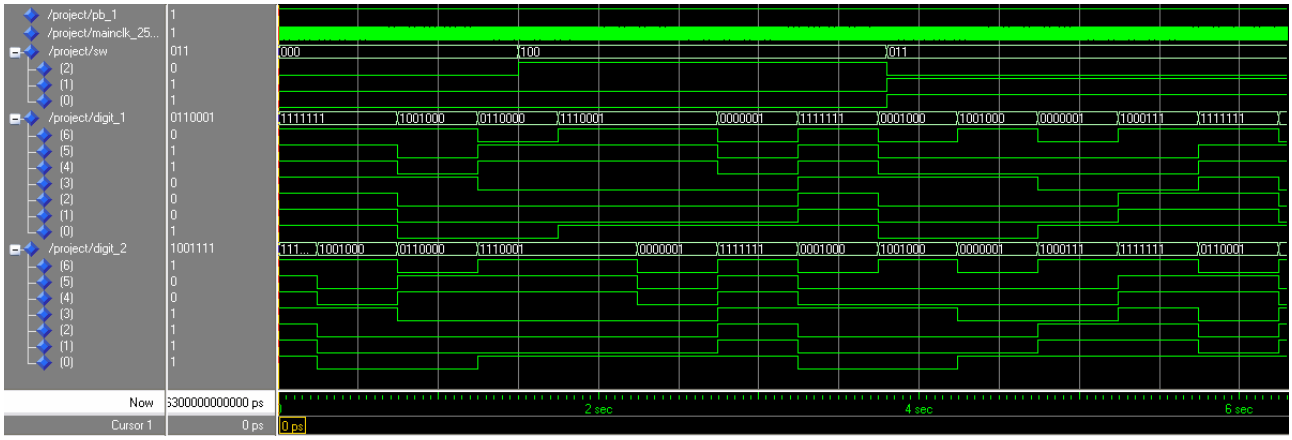


Figure 12: Functional Simulation of the entire design

As expected the simulation results confirm the right working way of the total design. In the first 3.5 seconds of the simulation the output is animating the "***HELLO***" word that corresponds to the switch configuration "***sw='000'***" and after it till 6.5 seconds the "***AHOJ***" word is animating that corresponds to the switch configuration "***sw='100'***". In the final part of the simulation it's possible to see that the "***CIAO***" word is animating that corresponds to the switch configuration "***sw='011'***".
To check if the design is effectively working on a FPGA device a timing simulation is required. This task is discussed in the next section.

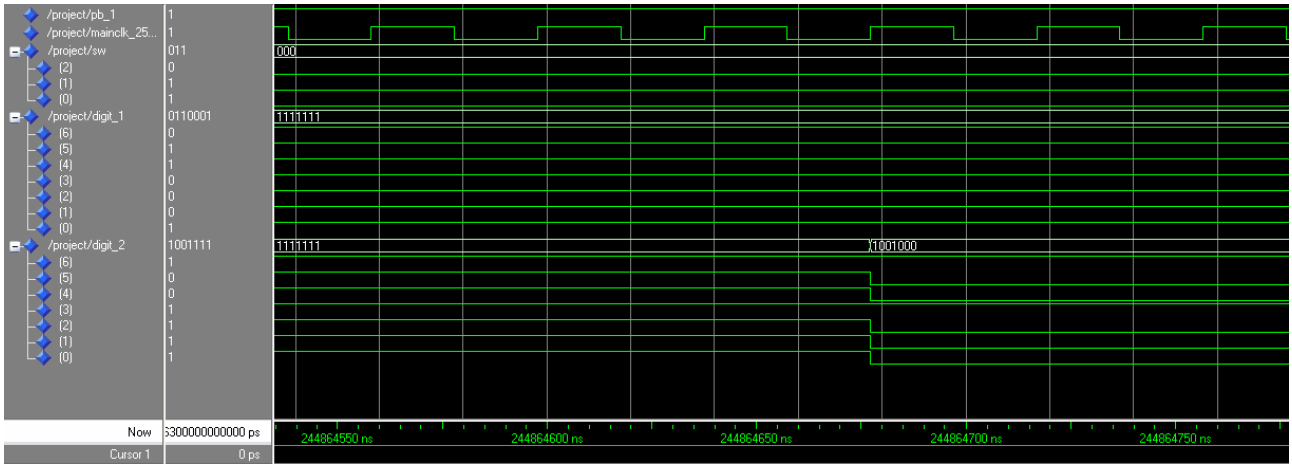


Figure 13: Zoom on first frame displaying

The zoom is performed on the time interval that shows the entrance of the first frame of the "***HELLO***" word animation. From this zoom it's possible to see that the change in the output is performed at the rising edge of the clock signal.
To automate this procedure it is possible to write all these TCL scripts in a file "Display_animation.do" :

"***quit -sim***
project new C:/Progetto_Modelsim/Modelsim/Display_animation/Functional Project
project addfile SHR_RR.vhd
project addfile flipflop_T.vhd
project addfile FlipFLop_D.vhd
project addfile Sel_sigelab.vhd
project addfile freq_divider.vhd
project addfile Mux2_schematic.vhd
project addfile decoder_4to7_full.vhd
project addfile controller_5L.vhd
project addfile Debcr.vhd
project addfile Project.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital SHR_RR.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital flipflop_T.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital FlipFLop_D.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital Sel_sigelab.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital freq_divider.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital Mux2_schematic.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital decoder_4to7_full.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital Debcr.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital controller_5L.vhd
vcom -2002 -explicit -novitalcheck -no1164 -novital Project.vhd
vsim work.Project
view signals wave
do SigDec.do
do Signals.do"

And invoke it from the command prompt window once set the working directory, with the following command:

"***do Display_animation.do***"

4.3. Timing Simulation

Before starting to explain the timing simulation procedure some encountered problems have to be explained:

- First of all, the ModelSim-Altera software doesn't have already the needed precompiled device library. In fact the project implemented in QuartusII software, mentioned in the last report, was implemented on a device of the FLEX10K family. The library of this device family is neither precompiled in the ModelSim-Altera software, neither available on the web and neither available in the QuartusII software directory "***<QuartusII installation directory>\eda\sim_lib***" (as specified in [3]).
  To overcome to this problem a new device of another family has been used. The used device is the "EPF10K200SRC240-3" of the "FLEX10KE" family. The library of this device family is already precompiled in the ModelSim-Altera software.
- To overcome to problems due to the high amount of time needed by the timing simulation of the entire design, I have decided to perform the timing simulation on a simplified version of the design. This simplified version doesn't include the frequency divider block. In the following figure the block diagram of the simplified design is reported:

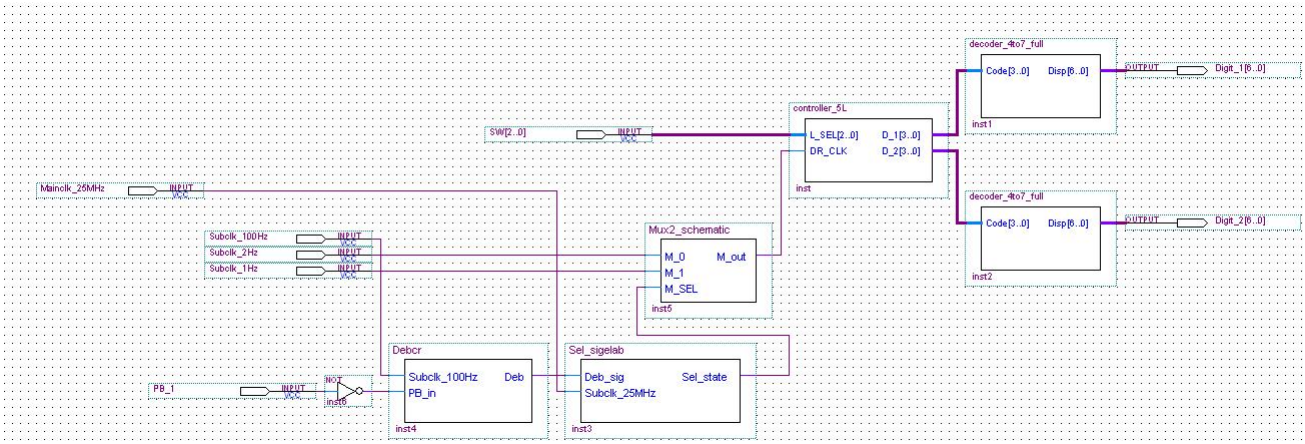


Figure 14: Simplified design block diagram

By removing the "Freq_divider" block three new inputs have to be specified:

- Subclk_100Hz;
- Subclk_2Hz;
- Subclk_1Hz.

To perform the timing the ModelSim-Altera software requires information about how the design was placed into device-specific architectural blocks and timing information which annotates the delay for the elements used in the design. The QuartusII software provides these information in the form of a Verilog Output File (.vo) for Verilog HDL designs or a VHDL Output File (.vho) for VHDL designs and in the Standard Delay Format Output File (.sdo).
The VHDL Output File (.vho) and the Standard Delay Format Output File (.sdo) could be created directly from the QuartusII software in the following manner:

- Click **Start compilation** on the **Processing** menu;
- Click **EDA Tool Settings** in the **Assignment** menu;
- Expand **EDA Tool Settings** in the **Category** list and click on **Simulation**;
- In the **Tool name** list select **ModelSim-Altera**;

- Under **EDA Netlist Writer options**, select **VHDL** as **Format for the output netlist**;
- Click **OK**;
- Point to **Start** on the **Processing** menu and click on **Start EDA Netlist writer**.

Once these files are created they are available in the following directory "*<QuartusII_project_directory>\simulation\modelsim>*". These files have to be copied in a folder that will be used as working directory for the ModelSim-Altera software, this is "*C:/Progetto_Modelsim/Modelsim/Display_animation_sp/Timing*".
At this point, once started the ModelSim-Altera software the working directory has to be set by writing on the command prompt window the following code:

"cd C:/Progetto_Modelsim/Modelsim/Display_animation_sp/Timing"

Where the working directory in which the .vho and .sdo files were copied is "Timing".
Now it is possible to create a new project in this software the code used to this purpose is :

"project new C:/Progetto_Modelsim/Modelsim/Display_animation_sp/Timing Project"

Where "Project" is the project name.
To add VHDL Output File (.vho) previously created to the project the following command line was used:

"project addfile Project.vho"

The following code is used to compile the added file in the working directory:

"vcom -2002 -explicit -novitalcheck -novital Project.vho"

The following code is used to start simulation:

"vsim -sdftyp Project_vhd.sdo work.Project"

Where the command vsim invokes the VSIM simulator.
To open the Wave window, in which the stimulus signals and the output signals will be displayed, the following code is used:

"view signals wave"

At this point it is useful to use a separate file where the add signal statements and the setting statements for the wave window are specified. This file is named "SigDec.do" and contains the following code :

```
"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /project/pb_1
add wave -noupdate -format Logic /project/mainclk_25mhz
add wave -noupdate -format Logic /project/subclk_100hz
add wave -noupdate -format Logic /project/subclk_2hz
add wave -noupdate -format Logic /project/subclk_1hz
add wave -noupdate -format Logic /project/sw
```

```
add wave -noupdate -format Logic /project/digit_1
add wave -noupdate -format Logic /project/digit_2
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {6300ms}"
```

The code written in blue is used to handle the messaging on the command prompt window; the code written in green is used to drag signals from the objects window and drop them in the wave window; the code written in red is used to setup the wave window and the cursors.
The "SigDec.do" file is launched from the command prompt window by writing this code:

" do SigDec.do"

It is more comfortable to create the stimulus signals in a separate file that is named "Signals.do" and contains the following code:

```
"restart -f
force -freeze sim:/project/pb_1 1 0, 1 {6300 ms}
force -freeze sim:/project/mainclk_25mhz 0 0, 1 {40us} -r 80us
force -freeze sim:/project/subclk_100hz 0 0, 1 {5ms} -r 10ms
force -freeze sim:/project/subclk_2hz 0 0, 1 {250 ms} -r 500ms
force -freeze sim:/project/subclk_1hz 0 0, 1 {500 ms} -r 1000ms
force -freeze sim:/project/sw(2) 0 0, 1 {1500 ms}
force -freeze sim:/project/sw(2) 0 3800ms, 0 {6300 ms}
force -freeze sim:/project/sw(1) 0 0, 1 {3800 ms}
force -freeze sim:/project/sw(0) 0 0, 1 {3800 ms}
run 6300 ms"
```

The code written in blue reloads the design elements and sets the simulation time to zero; the code written in green specifies the stimulus signals; the code written in red starts the simulation for the specified time duration.
The "Signals.do" file is launched from the command prompt window by writing this code:

" do Signals.do"

The simulation results for the stimulus signals specified in the "Signals.do" file is reported in the figure 15:

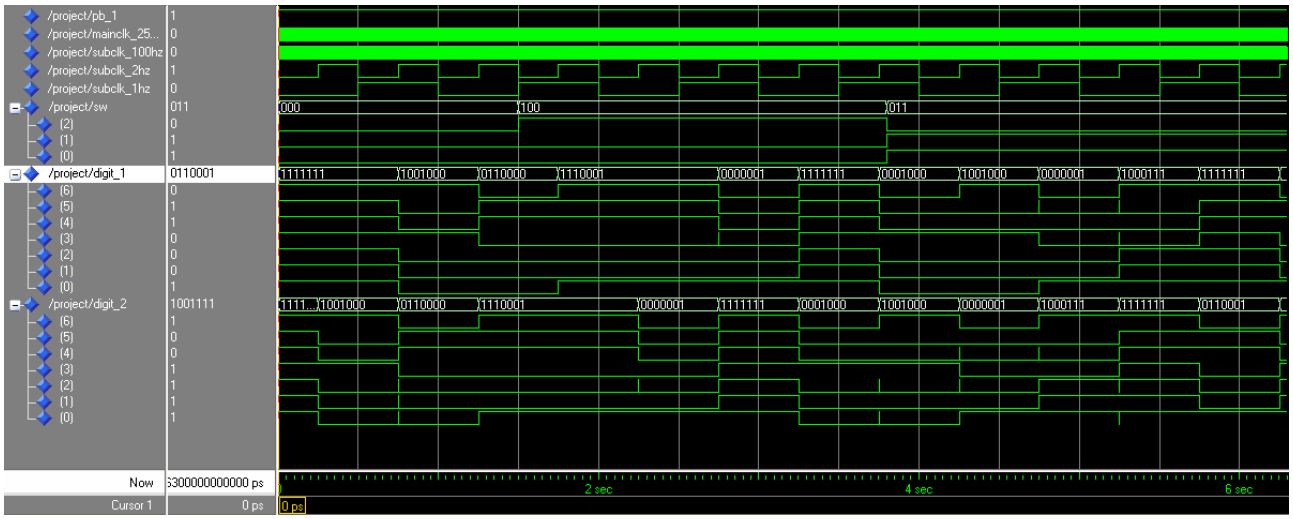


Figure 15: Timing Simulation of the simplified design

Also from the timing simulation results that the circuit is working in a correct way. As it's possible to see from the zoom shown in figure 16, that is a zoom on the transition from the first animation frame of the "***HELLO***" word animation and the second frame, the time interval required from the rising edge of the selected clock signal, that is subclk_2hz, to the valid output generation is of 27.418ns. This delay is mainly due to the time required by the "***controller_5L***" block to generate the outputs that drive the two decoders and to the time required by the decoders to generate the appropriate outputs that drive the two displays. In fact from figure 16 it's possible to see that these decoder's outputs reach their correct values in different time instants.
However the total delay of 27.418ns is acceptable because it is smaller than the time required by a led of a seven segment display to be turned on. This means that the correct working of the circuit and the visualization of the animation is not compromised.

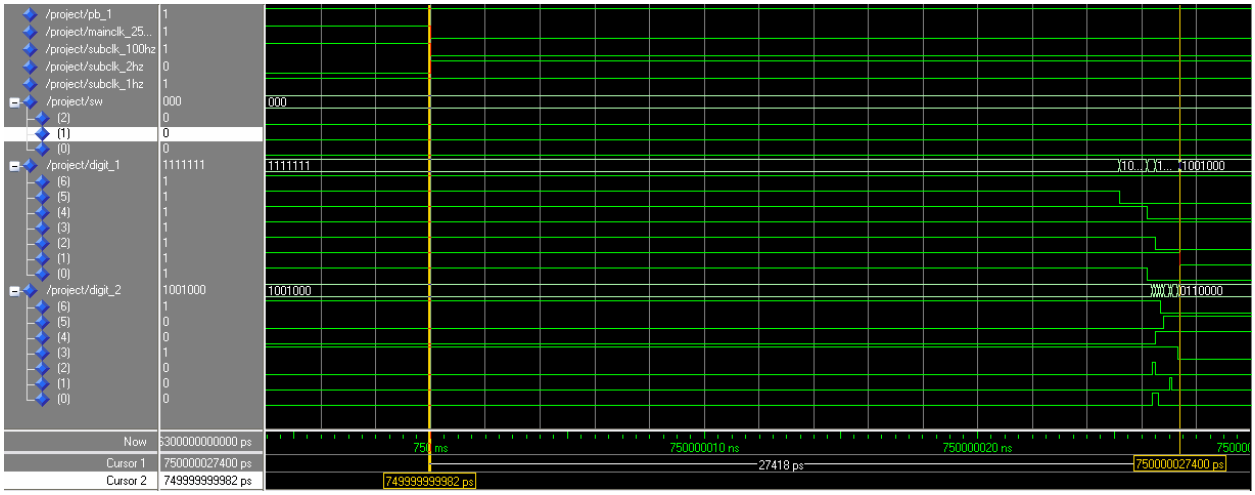


Figure 16: Zoom on the Timing Simulation of the simplified design

In figure 16 it is possible to note also the presence of some glitches on the outputs digit_1 and digit_2. These glitches however don't compromise neither the working of the circuit neither disturb the visualization of the animation on the displays, because the duration of the longest glitch is shorter than the time required by a led of the seven segment display to be turned on. In fact for example the duration of the longest glitch in figure 17, that is on the line digit_2(0), is of 200ps.

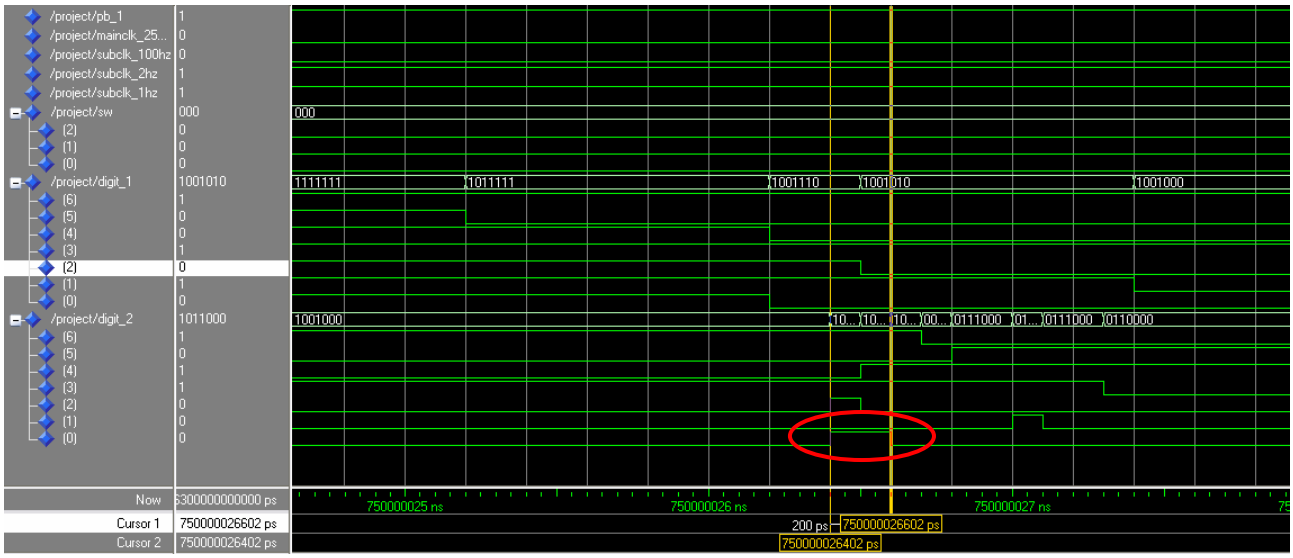


Figure 17: Further zoom on a glitch in the Timing Simulation of the simplified design

To automate this procedure it is possible to write all these TCL scripts in a single file "Display_animation_sp.do" :

"***quit -sim***
project new C:/Progetto_Modelsim/Modelsim/Display_animation_sp/Timing Project
project addfile Project.vho
vcom -2002 -explicit -novitalcheck -novital Project.vho
vsim -sdftyp Project_vhd.sdo work.Project
view signals wave
do SigDec.do
do Signals.do"

And invoke it from the command prompt window once set the working directory.

5. SIMULATING THE SUBUNITS OF THE PROJECT:"DISPLAY ANIMATION"

5.1. Introduction

To make more complete this report, the timing simulations of each subunit of the entire design will be reported in a briefly manner in this chapter. For each simulated subunit the used code will also be shown.
As mentioned in section 4.3, also for this simulations the FLEX10KE family devices will be used, to be able to perform the timing simulation within ModelSim-Altera software.
To simplify the simulation procedure all the necessary code is written in a ".do" file that permits to perform the simulation in an automatic way once the file has been invoked from the command prompt window of the ModelSim-Altera software, after setting the working directory.
Before starting Modelsim-Altera it is necessary to create the VHDL Output File (.vho) and the Standard Delay Format Output File (.sdo) of each subunit and put them in the appropriate ModelSim-Altera working directory as explained in section 4.3.

5.2. Freq-divider

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/freq_divider/Timing"***

To run automatically all the simulation process using the freq_divider.do file the following command has to be written in the command prompt window:

"***do freq_divider.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/freq_divider/Timing freq_divider
project addfile freq_divider.vho
vcom -2002 -explicit -novitalcheck -novital freq_divider.vho
vsim -sdftyp freq_divider_vhd.sdo work.freq_divider
view signals wave
do SigDec.do
do Signals.do"

Where the SigDec.do file content is:

"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /freq_divider/main_clk
add wave -noupdate -format Logic /freq_divider/clk_100hz
add wave -noupdate -format Logic /freq_divider/clk_1hz
add wave -noupdate -format Logic /freq_divider/clk_2hz
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {550ms}"

And the Signals.do file content:

"restart -f
force -freeze sim:/freq_divider/main_clk 0 0, 1 {9500 ps} -r 19 ns
run 550 ms"

As it's possible to see from the Signals.do file the input frequency is of 52MHz and not of 25.175MHz as the clock signal frequency used in the real design. A higher frequency value is used to reduce the simulation time. However it's possible to note that the introduced delays of this component are not so high to compromise the correct working of the circuit.
The simulation result is reported in figure 18.

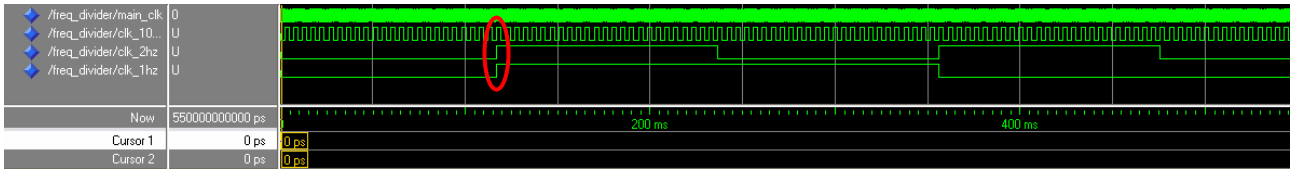


Figure 18: Timing Simulation of the frequency divider

From the figure 19, that is a zoom on the highlighted zone of figure 18, results that the delay between the rising edge of the clock signal main_clk and the output clock signal clk_1hz is of about 12.277ns.

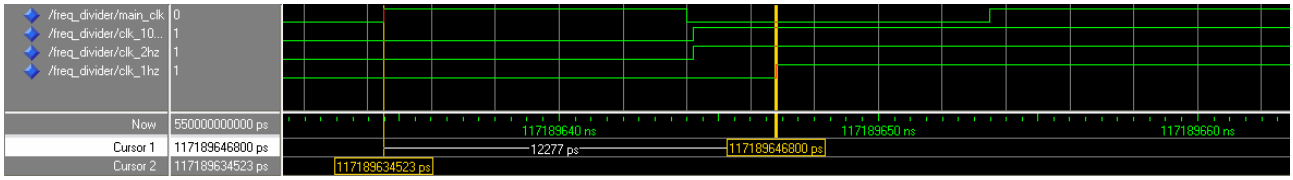


Figure 19: Further zoom on the Timing Simulation of the frequency divider

5.3. Debcr

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/debouncing/Timing"***

To run automatically all the simulation process using the debcr.do file the following command has to be written in the command prompt window:

"***do debcr.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/debouncing/Timing Debcr
project addfile Debcr.vho
vcom -2002 -explicit -novitalcheck -novital Debcr.vho
vsim -sdftyp Debcr_vhd.sdo work.Debcr
view signals wave
do SigDec.do
do Signals.do"

Where the content of the SigDec.do file is:

"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /debcr/subclk_100hz
add wave -noupdate -format Logic /debcr/pb_in
add wave -noupdate -format Logic /debcr/deb
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {150 ms}"

And the content of the Signal.do file is:

"restart -f
force -freeze sim:/debcr/subclk_100hz 0 0, 1 {5 ms} -r 10 ms
force -freeze sim:/debcr/pb_in 0 0, 1 {2 ms}
force -freeze sim:/debcr/pb_in 0 3ms, 1 {4 ms}
force -freeze sim:/debcr/pb_in 0 5ms, 1 {6 ms}
force -freeze sim:/debcr/pb_in 0 7ms, 1 {8 ms}
force -freeze sim:/debcr/pb_in 0 9ms, 1 {10 ms}
force -freeze sim:/debcr/pb_in 1 11ms, 0 {113 ms}
force -freeze sim:/debcr/pb_in 1 114 ms, 0 {115 ms}
force -freeze sim:/debcr/pb_in 1 116ms, 0 {117 ms}
force -freeze sim:/debcr/pb_in 1 118ms, 0 {119 ms}
run 150 ms"

The simulation result is reported in figure 20:

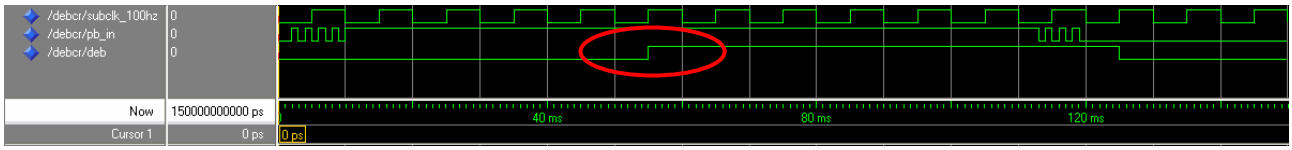


Figure 20: Timing Simulation of the debouncing circuit

As it is possible to see this subunit works properly. To be able to evaluate the time delay introduced by this circuit a zoom on the red circled zone is performed in the following figure:

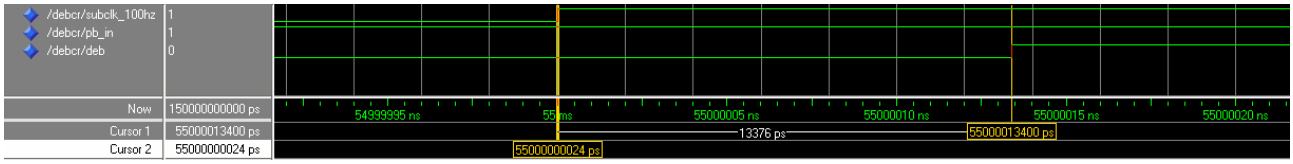


Figure 21:Zoom on the timing simulation of the debouncing circuit

The time delay between the rising edge of the clock signal and the output of the circuit is of 13.376ns, that doesn't compromise the correct working of the circuit.

5.4. Sel_sigelab

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/Sigelab/Timing"***

To run automatically all the simulation process using the Sel_sigelab.do file the following command has to be written in the command prompt window:

"***do Sel_sigelab.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/Sigelab/Timing Sel_sigelab
project addfile Sel_sigelab.vho
vcom -2002 -explicit -novitalcheck -novital Sel_sigelab.vho
vsim -sdftyp Sel_sigelab_vhd.sdo work.Sel_sigelab
view signals wave
do SigDec.do
do Signals.do"

Where the content of the SigDec.do file is:

"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /sel_sigelab/subclk_25mhz
add wave -noupdate -format Logic /sel_sigelab/deb_sig
add wave -noupdate -format Logic /sel_sigelab/sel_state
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0

configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {30ms}"

And the content of the Signals.do file is:

"restart -f
force -freeze sim:/sel_sigelab/deb_sig 0 0, 1 {5 ms} -r 10ms
force -freeze sim:/sel_sigelab/subclk_25mhz 0 0, 1 {19 ns} -r 38ns
run 30 ms"

The simulation result is reported in figure 22:

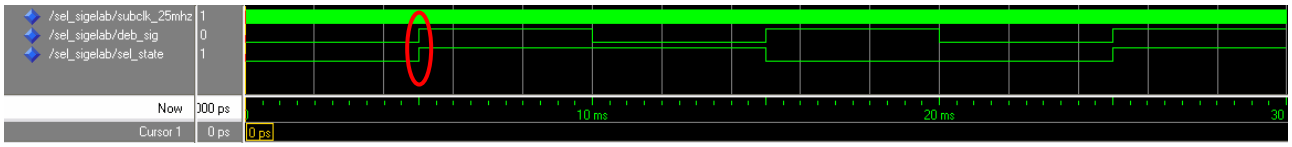


Figure 22: Timing Simulation of the Selection signal elaboration circuit

As it is possible to see this subunit works properly. By performing a zoom on the red circled zone that is shown in the following figure:

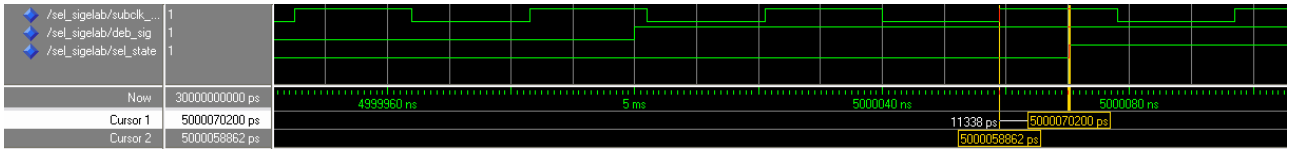


Figure 23:Zoom on the timing simulation of the Selection signal elaboration circuit

It is possible to see that the elaborated signal is available on the output 11.338ns after the rising edge of the clock signal. However this delay doesn't affect the correct working of the circuit.

5.5. Mux2_schematic

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/Mux2/Timing"***

To run automatically all the simulation process using the Mux_2.do file the following command has to be written in the command prompt window:

"***do Mux_2.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/Mux2/Timing Mux_2
project addfile Mux_2.vho
vcom -2002 -explicit -novitalcheck -novital Mux_2.vho

vsim -sdftyp Mux_2_vhd.sdo work.Mux_2
view signals wave
do SigDec.do
do Signals.do”

Where the content of SigDec.do is:

“onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /mux_2/m_sel
add wave -noupdate -format Logic /mux_2/m_0
add wave -noupdate -format Logic /mux_2/m_1
add wave -noupdate -format Logic /mux_2/m_out
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {500ms}”

And the content of Signals.do is:

“restart -f
force -freeze sim:/mux_2/m_0 1 0, 1 {500 ms}
force -freeze sim:/mux_2/m_1 0 0, 0 {500 ms}
force -freeze sim:/mux_2/m_sel 0 0, 0 {115 ms}
force -freeze sim:/mux_2/m_sel 1 116ms, 0 {260 ms}
force -freeze sim:/mux_2/m_sel 0 261ms, 1 {408 ms}
force -freeze sim:/mux_2/m_sel 1 409ms, 1 {500 ms}
run 500 ms”

The simulation result is reported in figure 24:

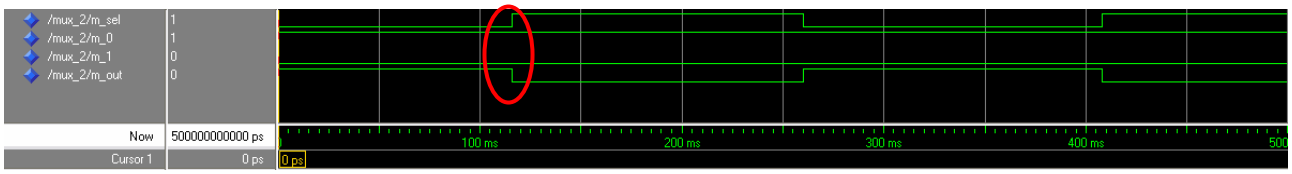


Figure 24: Timing Simulation of the Multiplexer

As it is possible to see this subunit works properly. By performing a zoom on the red circled zone that is shown in the following figure:

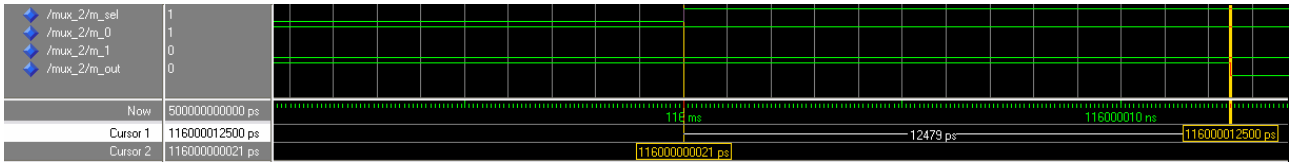


Figure 25:Zoom on the timing simulation of the Multiplexer

It is possible to see that the input signal m_1 is available on the output m_out after 12.479ns after the value change of the m_sel signal. However this delay doesn't affect the correct working of the circuit.

5.6. Controller_5L

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/Controller_5L/Timing"***

To run automatically all the simulation process using the controller_5l.do file the following command has to be written in the command prompt window:

"***do controller_5l.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/Controller_5L/Timing controller_5L
project addfile controller_5L.vho
vcom -2002 -explicit -novitalcheck -novital controller_5L.vho
vsim -sdftyp controller_5L_vhd.sdo work.controller_5L
view signals wave
do SigDec.do
do Signals.do"

Where the content of the SigDec.do fileis:

"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /controller_5l/l_sel
add wave -noupdate -format Logic /controller_5l/dr_clk
add wave -noupdate -format Logic /controller_5l/d_1
add wave -noupdate -format Logic /controller_5l/d_2
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4

configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {8000 ms}"

And the content of the Signals.do file is:

"restart -f
force -freeze sim:/controller_5l/dr_clk 0 0, 1 {250 ms} -r 500 ms
force -freeze sim:/controller_5l/l_sel(2) 0 0, 1 {1174 ms}
force -freeze sim:/controller_5l/l_sel(2) 0 3858 ms, 0 {8000 ms}
force -freeze sim:/controller_5l/l_sel(1) 0 0, 1 {4697 ms}
force -freeze sim:/controller_5l/l_sel(0) 0 0, 1 {4697 ms}
run 8000 ms"

The simulation result is reported in figure 26:

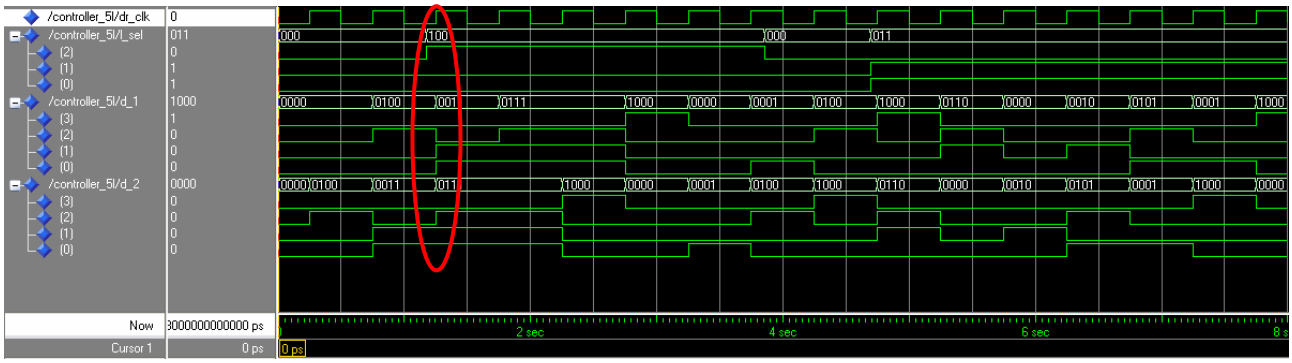


Figure 26: Timing Simulation of the Controller

As it is possible to see this subunit works properly. By performing a zoom on the red circled zone that is shown in the following figure:

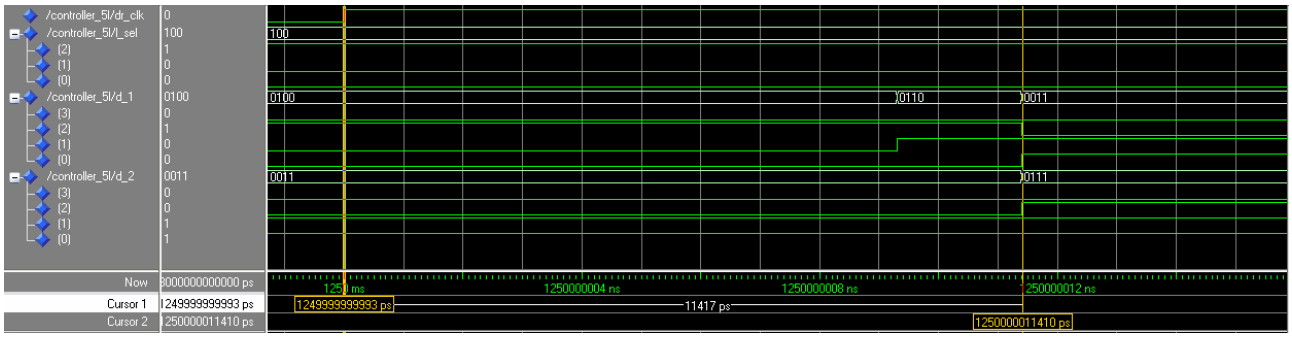


Figure 27:Zoom on the timing simulation of the Controller

It is possible to see that the time needed by the controller unit to generate a valid output is about 11.417ns. This time delay however doesn't affect the correct working of the entire circuit in which this controller is placed because it works at lower frequencies.

5.7. Decoder_4to7_full

Once opened the ModelSim-Altera software the working directory has to be set with the following code:

"***cd C:/Progetto_Modelsim/Modelsim/Decoder/Timing"***

To run automatically all the simulation process using the decoder_4to7_full.do file the following command has to be written in the command prompt window:

"***do decoder_4to7_full.do***"

The content of this file is:

"quit -sim
project new C:/Progetto_Modelsim/Modelsim/Decoder/Timing decoder_4to7_full
project addfile decoder_4to7_full.vho
vcom -2002 -explicit -novitalcheck -novital decoder_4to7_full.vho
vsim -sdftyp decoder_4to7_full_vhd.sdo work.decoder_4to7_full
view signals wave
do SigDec.do
do Signals.do"

Where the content of SigDec.do is:

"onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate -format Logic /decoder_4to7_full/code
add wave -noupdate -format Logic /decoder_4to7_full/disp
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ns} 0}
configure wave -namecolwidth 150
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
update
WaveRestoreZoom {0 ns} {16000 ms}"

And the content of Signals.do is:

"restart -f
force -freeze sim:/decoder_4to7_full/code(3) 0 0, 1 {4000 ms} -r 8000 ms
force -freeze sim:/decoder_4to7_full/code(2) 0 0, 1 {2000 ms} -r 4000 ms

force -freeze sim:/decoder_4to7_full/code(1) 0 0, 1 {1000 ms} -r 2000 ms
force -freeze sim:/decoder_4to7_full/code(0) 0 0, 1 {500 ms} -r 1000 ms
run 16000 ms"

The simulation result is reported in figure 28:

Figure 28: Timing Simulation of the Decoder

As it is possible to see this subunit works properly. By performing a zoom on the red circled zone that is shown in the following figure:

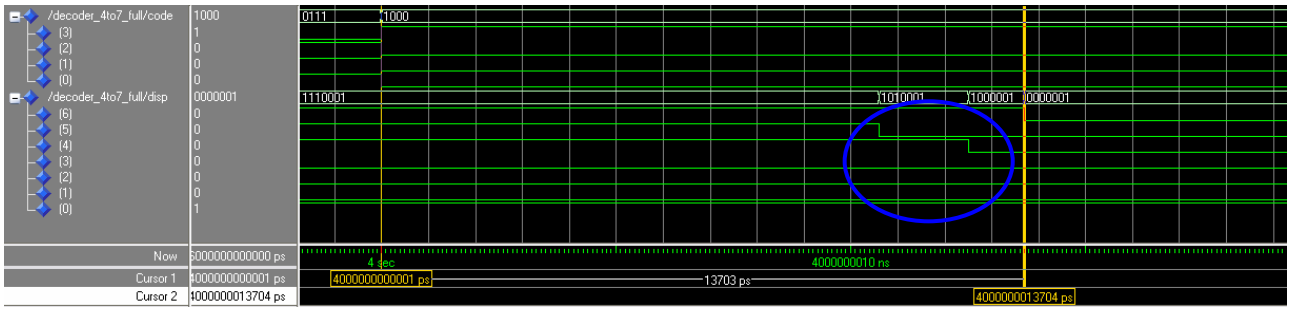


Figure 29:Zoom on the timing simulation of the Decoder

The decoder has to drive a seven segment display. As it is possible to see the time interval needed by the circuit to process an input code is about 13.703ns. However the changing in the outputs, highlighted in the blue circle doesn't have effects on the display because the time needed by a led to be turned on is much more longer than the time interval needed by the decoder to set its outputs in a properly manner.

6. CONCLUSIONS

In this work the ModelSim-Altera software has been introduced and used to perform either functional either timing simulations on the project reported in the previous work created with QuartusII software.
By using the ModelSim-Altera software various problems has been encountered. For example, as explained in section 3.2.4, some files like the VHDL testbench (.vht) files, created by the QuartusII software are not created in an appropriate manner for the ModelSim-Altera software when the simulation time exceeds a certain limit and so the user has to edit these files before performing simulations.
Moreover other incompatibilities regard the VHDL code exported from QuartusII software, in fact it is possible that certain created VHDL files contain some code lines that could not be interpreted in a correct way by the ModelSim-Altera software. These problems could be solved by using the debug tool of the ModelSim-Altera software.

Another encountered problem is that not all libraries for the Altera devices are available for the ModelSim-Altera software as explained in section 4.3.
Moreover it has been noted that the required time amount to perform timing simulation on the entire project is very high this is due mainly because the utilized version of the ModelSim-Altera software was the Web edition that has lesser performance than the full edition as explained in section 1.
Beside these problems, that were solved in an appropriate manner, however it was possible to perform the functional and the timing simulations of the entire project and on the various subunits constituting the project.
As it's possible to see in section 4 and section 5 all elements of the circuit and the entire circuit work as expected in a right way.

REFERENCES

[1] Help On-line of Altera QuartusII
[2] Help On-line of ModelSim-Altera
[3] QuartusII Handbook volume 3, Mentor Graphics ModelSim Support, Altera QII53001-7.1.0, May 2007
[4] ModelSim-Altera, User's manual, Version 6.1g, Mentor Graphics Corporation, August 2006
[5] ModelSim-Altera, GUI Reference, Version 6.1g, Mentor Graphics Corporation, August 2006
[6] Using ModelSim-Altera in a QuartusII Design Flow, Altera Application Note 204, December 2002, ver. 1.2
[7] ModelSim-Altera, Tutorial, Version 6.1g, Mentor Graphics Corporation, August 2006
[8] ModelSim-Altera, Command Reference, Version 6.1g, Mentor Graphics Corporation, August 2006
[9] ***http://en.wikipedia.org/wiki/Electronic_design_automation***, May 2007
[10] ***http://www.altera.com/support/software/nativelink/simulation/modelsim/eda_pro_msim_timing_sim.html***, May 2007
[11] ***http://www.altera.com/products/software/products/model/eda-ms.html***, May 2007
[12] Standard VITAL ASIC Modeling specifications, Draft IEEE 1076.4/D1, April 2000