Using ModelSim with Quartus II Block Design Files

A Block Design File to VHDL File Converter and ModelSim Starter

Jesse op den Brouw
The Hague University Of Applied Sciences
Department of Electrical Engeneering

J.E.J.opdenBrouw@hhs.nl

August 27, 2021

Abstract

Students at The Hague University Of Applied Sciences get their first glimpse at Digital Design education using only schematic entry using the Quartus II software environment. Schematic entry is done using a full screen WYSIWYG editor and generates Block Design Files (BDF files). This is a proprietary file type and is not supported outside the Quartus II environment.

Simulation is at this stage unknown to them. We try to hide as much as possible as not to distract their attention from the design process.

ModelSim is a full fledged VHDL and Verilog simulator and widely spread amongst digital system designers, but is unable to compile and simulate Quartus' BDF files.

Quartus provides an option to convert BDF files to VHDL files. Converting BDF files to VHDL files is a tedious and error prone operation and has to be done every time the design is updated.

This document describes a set of files as part of a *design flow* that deals with all of the problems mentioned above. The scripts run both on Windows and Linux operating systems. Both the Subscription Edition and the Web Edition are supported.

Contents

Listings

List of Figures

1 Introduction

Students of the faculty of Electrical Engineering at the The Hague University of Applied Sciences[1] [1] get acquainted with digital design in the first year of their study. The learning line consists of three courses.

In the first course they learn the basics of digital design like number systems, boolean algebra, logic gates, K-maps and some elementary knowledge of latches and flip-flops. At this stage, they do not use VHDL or any other HDL, and they do not know anything about simulation.

For practical work, the students use the Quartus II design software from Altera. As hardware platform, they use the DE0 Digital Systems Board supplied by Terasic [2]. It consists of a Cyclone III FPGA with about 15,000 cells, LEDs, switches, push buttons and seven segment displays.

All the practical work in the first course is done using schematic entry and using logic gates to complete the assignments. They do use hierarchies. The students use simulation but only to verify if their solution is correct; all the simulation scripts and testbenches have been prepared by faculty staff.

Schematic designs are saved in so-called Block Design Files. Block Design Files are proprietary files to Quartus. These files, recognizable by the extension `.bdf`, can be synthesized using the Quartus software.

Simulation is done with Modelsim. ModelSim is a well known and widespread VHDL and Verilog simulator, but is unable to compile and simulate BDF files.

Fortunately, Quartus has an option to convert BDF files to VHDL or Verilog files. This can be done by opening the appropriate BDF file and using the Create HDL File option. You can see a screenshot in Figure 1.

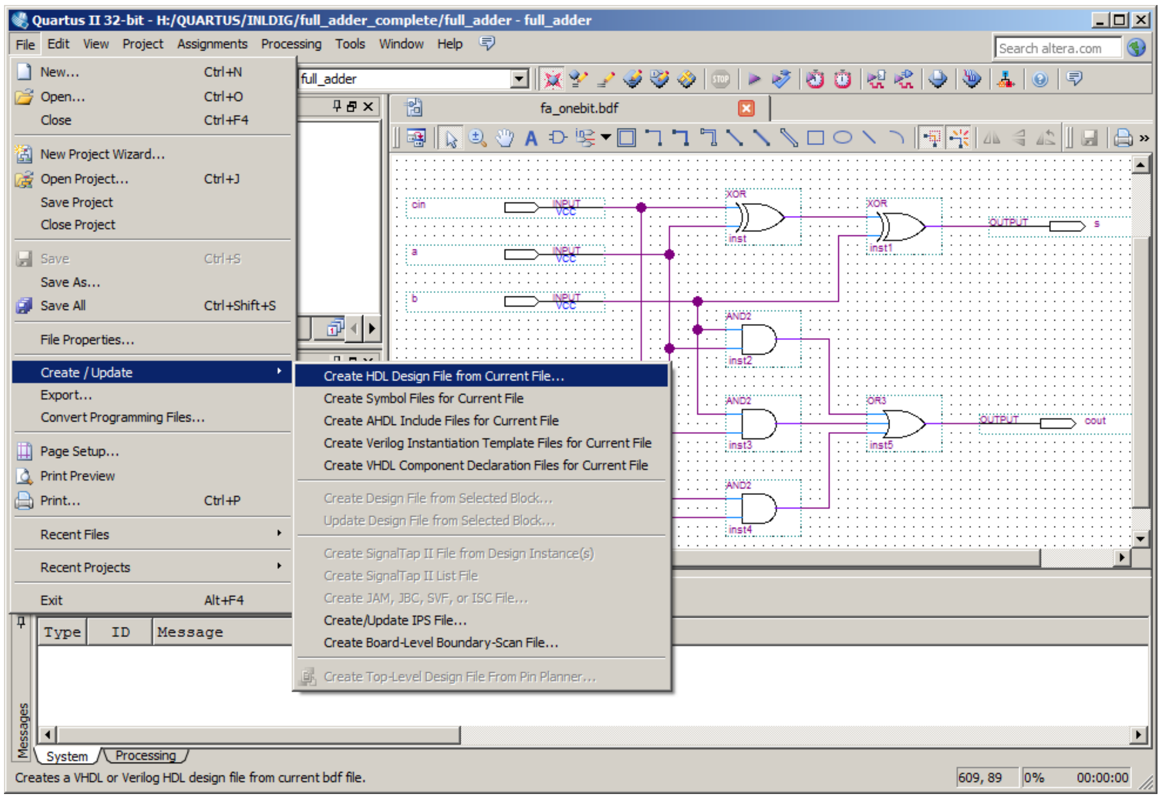
Of course, this has to be done for every file in the project and every time the files have changed. This is not only dull, but also error prone. You can easily forget to convert a changed file, render the design useless.

This document describes a set of files that deals with all of the problems mentioned above. There are two files: a script that handles the conversion of Block Design Files to VHDL files and starts ModelSim automatically, and a file that will set up the script as part of a so-called *design flow*. The files run both on Windows and Linux operating systems. Both the Subscription Edition and the Web Edition are supported.

This document consists of eight sections. Section 2 describes more about how a typical project is set up. Section 3 describes the environment the script runs in. Section 4 contains a first description on what the scripts actually does. Section 5 gives an in-depth explanation of the script's internal working. Section 6 gives some hints on how to set up a generic ModelSim command file. Section 7 describes the installation of the script and the design flow file. Section 8 gives some notes on things to avoid when creating schematics. At last, Section 9 deals about some known issues.

The intended audience are designers who make use of Block Design Files and want to use ModelSim as their favorite simulator, and faculty staff who want their students to use

[1] In Dutch: De Haagse Hogeschool


Quartus II 32-bit - H:/QUARTUS/INLDIG/full_adder_complete/full_adder - full_adder
File Edit View Project Assignments Processing Tools Window Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
Create SignalTap II File from Design Instance(s)
Create SignalTap II List File
Create JAM, JBC, SVF, or ISC File...
Create/Update IPS File...
Create Board-Level Boundary-Scan File...
Create Top-Level Design File From Pin Planner...
full_adder
fa_onebit.bdf
cin
a
b
XOR
inst
inst1
AND2
inst2
inst3
inst4
OR3
inst5
OUTPUT
s
cout
Type ID Message
System
Processing
609, 89
0%
00:00:00

Creates a VHDL or Verilog HDL design file from current bdf file.

Figure 1: *Converting a Block Design File to an HDL File.*

schematic entry of digital systems (and of course use ModelSim for simulation).

Please note that the script hasn't been tested with Mega Functions or MaxII functions, only with primitive functions such as AND, OR, and NOT.

There is a similar script for converting BDF files to Verilog files by Chris Zeh. See [3].

2 A Typical Quartus Project Setup

Before we tell more about the internals of the script it's best to give an overview of a typical Quartus project suited for this setup.

For the script to work, there are two true obligations: the name of the ModelSim command file must consist of the prefix `tb_` followed by the name of the top level design entity (which is not the design entity name of the testbench) and the extension `.do`, and the name of the top level design entity must be the same as the first part of the design filename. There are really no other obligations (even the top level filename can differ from the top level entity name, but it is included to force students to use some sensible filenames). Note that for BDF files, the name of the design entity is always the same as first part of the filename. Also note that this is not always true for HDL files. So if there's a top level design entity with the name `full_adder`, the corresponding filename must be `full_adder.bdf` (or `full_adder.vhd` as an example). It's good practice though to keep the filename of the testbench as close as possible to the design entity filename, so the name of the corresponding testbench filename should the same as the top level entity name with the prefix `tb_` and the extension `.vhd`. The typical project has the following files:

`full_adder.bdf` - the Block Design File with the top level design entity

`tb_full_adder.vhd` - the testbench file

`tb_full_adder.do` - the ModelSim command file

It's possible to use hierarchies, consisting of multiple BDF files. The script is able to process all BDF files in the current project. It's also possible to incorporate more that one ModelSim command file to simulate multiple (sub) designs. The only thing you have to do is to change the top level design entity name in the Quartus environment (and run Analysis and Synthesis, see Section 9).

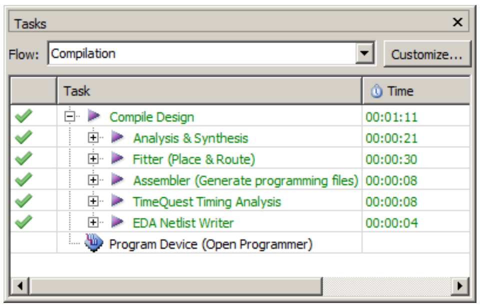
The project may contain other file types such as VHDL and Verilog files. The script will not touch these files as long as they are not generated from BDF files.

Only BDF files visible in the project environment (the files you see in the Files tab of the Project Navigator in the Quartus IDE) are processed, all other BDF files are not processed. Rationale for this is that you probably created the BDF file in the course of the project and forgot to delete the file when is wasn't needed anymore. Note however that all VHDL files associated with such BDF files are removed. This way, ModelSim will not compile them when using the code in Listing 13 in Section 6 (and it seems odd to have those VHDL files linger on).

It is possible to use a VHDL or Verilog file as the top level design entity. When instantiating designs from BDF files, you can just use the design entity names. Note that in the corresponding VHDL files, the architecture name is always `bdf_type`.

3 The Script's Environment

The script has to be installed as part of a so-called *design flow*. A design flow is a list of tasks that have to be done in order to fulfil the design's needs. Mostly, you have to synthesize the design, run the timing analyser and create a programming file for the device. Figure 2 gives an example of a completed flow.



Tasks

Flow: Compilation

	Task	Time
✓	Compile Design	00:01:11
✓	Analysis & Synthesis	00:00:21
✓	Fitter (Place & Route)	00:00:30
✓	Assembler (Generate programming files)	00:00:08
✓	TimeQuest Timing Analysis	00:00:08
✓	EDA Netlist Writer	00:00:04
	Program Device (Open Programmer)	

Figure 2: *An example of a completed design flow.*

The script is written in Tcl ("tickle"). Tcl is a *scripting language*, a language designed for automating tasks which could be done by hand by an human operator. The language provides a full set of flow control statements, functions and a lot of routines (in Tcl they all are called commands). For a introductory course on Tcl, see [4] and [5].

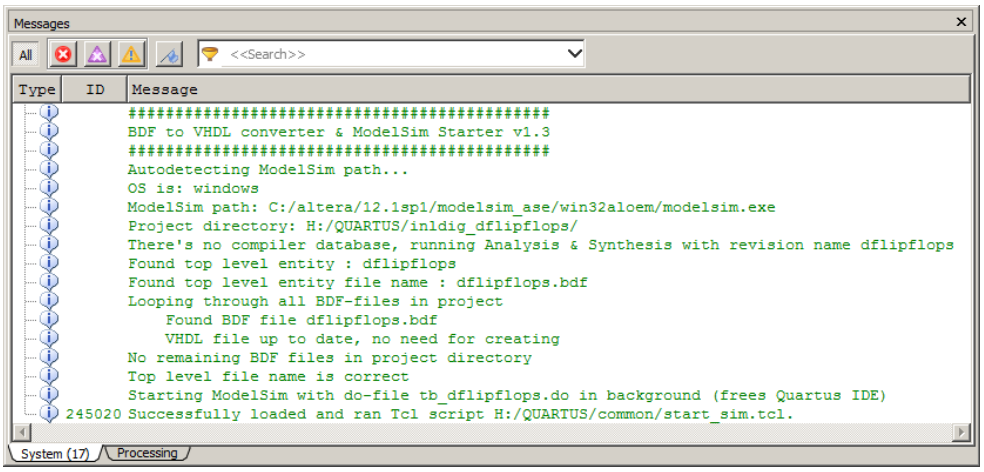
The Quartus environment heavily uses Tcl for scripting purposes and provides a set of packages. The packages provide an interface to Quartus' internal information. The script makes use of the `::quartus::project` Tcl package. More information can be found in [6].

Examples are: finding the top level design name, the project directory. For an extensive overview and examples, see [7] and [8].

The script makes use of the `quartus_map` command. This command is able to do a lot of things for you: create the design database, convert files, analysis and synthesis. See Quartus AN309[citation needed].

When the script runs, it prints information in the System tab of the Message window. This is done by the `post_message` command, optionally followed by a message type. An example of some output can be seen in Figure 3.

Note there's no way to pass arguments to the script, so you can't pass the name of the ModelSim command file. That's why the script always presumes a filename as described in Section 2.



```
        ############################################
        BDF to VHDL converter & ModelSim Starter v1.3
        ############################################
        Autodetecting ModelSim path...
        OS is: windows
        ModelSim path: C:/altera/12.1sp1/modelsim_ase/win32aloem/modelsim.exe
        Project directory: H:/QUARTUS/inldig_dflipflops/
        There's no compiler database, running Analysis & Synthesis with revision name dflipflops
        Found top level entity : dflipflops
        Found top level entity file name : dflipflops.bdf
        Looping through all BDF-files in project
            Found BDF file dflipflops.bdf
            VHDL file up to date, no need for creating
        No remaining BDF files in project directory
        Top level file name is correct
        Starting ModelSim with do-file tb_dflipflops.do in background (frees Quartus IDE)
 245020 Successfully loaded and ran Tcl script H:/QUARTUS/common/start_sim.tcl.
```

Figure 3: *An example output of the script.*

4 Quick overview of the script

This script does a number of things, but mainly it converts all BDF files in the current project environment into VHDL files and starts ModelSim with an associated command file. Of course there are a lot of build-in checks to determine if conversion and simulation is at all possible. A list of stages is given below:

1. Checks if the project is open.
2. Finds ModelSim execution path if none is provided. Linux and Windows supported.
3. Creates a project database if none is found.
4. Finds the top level entity name, checks if the top level entity name has an associated file, complains if none is found.
5. Loops through all BDF files found in the project environment and creates associated VHDL files if needed.
6. Removes all VHDL files from associated BDF files in the project directory but not in the project environment, but not VHDL files that do NOT have a associated BDF file[2]

[2]You probably have to read this sentence twice. See Section 5 for details.

7. Finds top level filename and creates DO filename.
8. Starts ModelSim with DO filename.

5 The Script's Internal Working

For the impatients: a complete printout of the script can be found in Appendix A.

The script starts with a lot of comment explaining the working of the script in shorthand. The script is unable to handle arguments due to the fact that it is called as part of a flow by the Quartus GUI. There is only one user option available as you can see in Listing 1. If you don't want the script to find the ModelSim install path, please fill in the user option.

```
# User input: set to the modelsim path. Keep empty for autodetect.
#set modelsim_exec_path "/opt/altera/12.1sp1/modelsim_ase/linuxaloem/vsim"
set modelsim_exec_path ""
```

Listing 1: *Set ModelSim path.*

Just as any script, it first prints a pretty banner. Currently, we have version 1.3 available. See Listing 2.

```
# Print a nice banner
post_message -type info "##############################################"
post_message -type info "BDF to VHDL converter & ModelSim Starter v1.3"
post_message -type info "##############################################"
```

Listing 2: *Print a nice banner.*

In the first stage, the script checks if there is there is an opened project. It is needed to continue. If there's no opened project, the script exits with a failure. This can be seen in Listing 3. (Here you can see an example of Quartus' API, `is_project_open` is a function of the project package.) For more information, see [6].

```
# Check for project opened.
if {![is_project_open]} {
    post_message -type error "There's no project open! Please open a project and
        rerun."
    return False
}
```

Listing 3: *Check for open project.*

Next, the script tries to find the install path of the ModelSim executable if the user left the install path option blank. First, the Quartus user environment is consulted and if this is not set, the script searches the Quartus installation directory for the ModelSim installation. This means that you have to install ModelSim somewere under the Altera root install path. Currently, the script can handle Windows and Linux systems. The script will use the ModelSim Starter Edition, even if it finds more than one ModelSim installation. The script bails out if it cannot find a ModelSim installation.

This part is presented in Listing 4. The found path is then normalized, which means that references such as `../` and `./` are removed. The path is displayed on screen.

```tcl
# Export any newly changed/added/removed assignments
export_assignments

# Autodetect ModelSim exec path if none is provided. First, the user
# preferences are consulted, then ModelSim is autodetected. For this to
# work, ModelSim must be installed within the Quartus environment.
if { [string length $modelsim_exec_path] == 0 } {
    post_message -type info "Autodetecting ModelSim path..."
    set opsys [string tolower [lindex $::tcl_platform(os) 0]]
    post_message -type info "OS is: $opsys"

    # Try to get ModelSim path from user preferences
    set modelsim_exec_path [get_user_option -name EDA_TOOL_PATH_MODELSIM_ALTERA]

    if { [string length $modelsim_exec_path] > 0} {
        # User has entered a path in EDA Tool Options...
        post_message -type info "Found user preference path: $modelsim_exec_path"
        set modelsim_exec_path [string map {"\\" "/"} $modelsim_exec_path]
        # Different OSes...
        switch $opsys {
            linux { append modelsim_exec_path "/vsim" }
            windows { append modelsim_exec_path "/modelsim.exe" }
            default { post_message -type error "Cannot continue: unknowm platform
                 is $opsys. Bailing out."
                return False }
        }
    } else {
        # Tries to find a ModelSim installation directory. Stops if found none is
            found.
        # Stops if more than one found AND one of them is NOT ModelSim ASE.
        # Continues if ModelSim ASE is found in multiple ModelSIm installations.
        set modelsim_exec_path $quartus(quartus_rootpath)
        append modelsim_exec_path "../"
        switch [llength [set modelsim_list [ glob -nocomplain -path
            $modelsim_exec_path modelsim* ]]] {
            0 { post_message -type error "ModelSim not installed in Quartus
                environment! Bailing out."
                return False }
            1 { set modelsim_exec_path [lindex $modelsim_list 0] }
            default { set modelsim_exec_path [lindex $modelsim_list [lsearch
                $modelsim_list *modelsim_ase]]
                if { [string length $modelsim_exec_path] > 0} {
                    post_message -type info "Found ModelSim path for Altera
                        Starter Edition (modelsim_ase)."
                } else {
                    post_message -type error "Multiple ModelSim installations
                        found! Bailing out."
                    return False
                }
            }
        }
        # Different OSes...
        switch $opsys {
            linux { append modelsim_exec_path "/linuxaloem/vsim" }
            windows { append modelsim_exec_path "/win32aloem/modelsim.exe" }
            default { post_message -type error "Cannot continue: unknowm platform
                 is $opsys. Bailing out."
                return False }
        }
    }
}

# Normalize path name (get rid of ../ and ./ etc)
set modelsim_exec_path [file normalize $modelsim_exec_path]
```

Listing 4: *Determine the install path of the ModelSim executable.*

Now, the script checks whether the ModelSim executable is executable. See Listing 5.

```
# Check if the ModelSim executable is executable...
if { [file executable $modelsim_exec_path] == 0 } {
    post_message -type error "ModelSim executable cannot be run by current user.
        Bailing out!" -submsgs {"You should check the path to the executable in
        this script or via menu" "Tools->Options->EDA Tool Options or your
        ModelSim installation is corrupt."}
    return False
}
```

Listing 5: *Check if ModelSim is executable.*

In the second stage, started in Listing 6, the project directory is set. Please read the comment provided with the code. (Normally, — the source is the documentation — should not hold true, but is this case it's pretty accurate.)

```
# Set the project directory. This is needed because if you have added a BDF file
# that is not in the project directory (e.g. ../<some_other_dir>/file.bdf),
# Quartus changes the current directory path ([pwd]). Yes, really, it does...
# Please note that BDF files outside the project directory are not supported.
# The problem is that Quartus creates a VHDL file in that same directory. This
# could overwrite an existing file. There's no option to provide an output
# directory.
set project_directory [get_project_directory]
post_message -type info "Project directory: $project_directory"
cd $project_directory
```

Listing 6: *Set the project directory.*

Next, the script tries to get the top level entity. Normally this is provided during set up of the Quartus project. If there's no top level entity found, the script will try to create one. It sets the current revision and starts the Quartus command `quartus_map` with a long list of options. Please note the `catch` statement along with the `exec` statement. When the command fails, the script issues an error message and exits. The `catch` command prevents that. Instead the script prints some information about the failure of the command. When this happens, there are five possibilities:

1. You have a project without a file containing the top level description. Probably a project without design files but with project files.
2. You have a top level design entity that cannot be synthesized, e.g. testbenches or high level descriptions used for simulation only.
3. You have an error in one of your design files.
4. If the current device is not supported in the current version of Quartus, the analysis will fail.
5. If you stripped the project of all non-essential files and at startup you change the device, and the previous device files are not installed, the analysis will fail.

In case of the second item, you can simply restart the script, because now there is a database and a top level design entity is available. In case of the fourth and fifth item, you handle as follows:

- Close Quartus.
- Open de directory of your Quartus project.
- Remove the file `defaults.qdf`.
- Remove de `db` and `incremental_db` directories.
- Restart Quartus and open your project.

The code is presented in Listing 7.

```
# Get current revision
# Check if there is a database. If not, create one.
set current_revision ""
if { [catch {get_top_level_entity}] } {
    set current_revision [get_current_revision]
    post_message -type info "There’s no compiler database, running Analysis &
        Synthesis with revision name $current_revision"
    # Running Analysis & Synthesis currenly crashes when there’s no file
    # containing the toplevel, that is, you have sole QPF and QSF files,
    # or there’s no way to systhesize the design (or any generic error for
    # that matter).
    set status [catch { exec quartus_map --read_settings_files=on
        --write_settings_files=off $current_revision -c $current_revision }
        result]
    if { $status != 0 } {
        post_message -type error "Creating database failed! There are five
            posibilities:"
        post_message -type error "1: you have a project without a file containing
             the top level description."
        post_message -type error "2: you have a design that cannot be
            synthesized."
        post_message -type error "3: you have an error in (one of) your design
            file(s)."
        post_message -type error "4: the current device is not supported in this
            version of Quartus."
        post_message -type error "5: you changed the target device and/or the
            device files of the previous/current device are not installed."
        post_message -type error "You can try rerunning the script. Bailing out."
        return False
    }
} else {
    set current_revision [get_current_revision]
}
# Echo the current revision
post_message -type info "Current revision: $current_revision"
```

Listing 7: *Check the current revision and database.*

At the fourth stage, the script tries to find the file containing the top level entity. It does this in three steps. It first finds the top level entity currently focused, then it finds the associated file containing top level entity and last it checks if the file really exists. When it fails, there is no file (you probably deleted it), but the project has a database. You have to enter a file. See Listing 8.

```
# Find top level entity currently !focused! See Quartus:
# Assignments->Settings->General->Top Level Entity
set top_level_entity [get_name_info -info entity_name [get_top_level_entity]]
post_message -type info "Found top level entity : $top_level_entity"

# Find the file containing top level entity
set top_level_entity_file_name [get_name_info -info file_location [
    get_top_level_entity]]
post_message -type info "Found top level entity file name :
    $top_level_entity_file_name"

# Check for empty top level filename. Does happen when in a completely stripped
# project the simulation is started (that is: there's no file containing the
# top level entity, but there is a project database).
if { [string compare $top_level_entity_file_name  ""] == 0} {
    post_message -type error "Top level filename is empty. Please enter a file,
        rerun Analysis & Synthesis and start again. Bailing out."
    return False;
}
```

Listing 8: *Find the file containing top level entity.*

In Listing 9, all BDF filenames in the project are saved (this will be later explained). Then we print a nice line about what we are going to do.

```
# Create a list of all BDF files in the project DIRECTORY. We need this list
# for later on. If a BDF file is in the project environment, we remove it from
# this list. At the end we have list of BDF files in the project directory but
# not in the project environment.
set all_bdf_design_file_names [glob -nocomplain -type f *.bdf]
```

Listing 9: *Create a list of all BDF files in the project directory.*

Now we come the the stage where the BDF files are converted to VHDL files (see Listing 10). This is a bit tricky to explain, but here is how it works. The project *directory* contains all files on disk, whereas the project *environment* contains only the files that were added to the Quartus project as seen in the Project Navigator window in the Quartus IDE.

The script loops through all the BDF files that were added to the project environment (these are the files that you want to be in the project). All these filenames are requested through the Quartus API. The filenames found are saved in a list for later use (we want them to be converted) but at the same time the filenames are removed from the list of all BDF files in the project directory. This way we have a list of all the BDF files we want to convert and we have list of BDF files that are in the project directory but not in the project environment (these files are somehow saved in the directory but are not used in the project).

```
# Find all BDF files in project. This excludes BDF files that are in the
# project directory but not in the project environment. For all BDF files in
# the project, create a VHDL file if needed. Currently works for current
# directory level.
post_message -type info "Looping through all BDF-files in project"

set all_bdf_design_file_names_in_project ""
foreach_in_collection asgn_id [get_all_assignments -type global -name BDF_FILE] {

    # Get next BDF file name
    set bdf_design_file_name  [get_assignment_info $asgn_id -value]
    # Add to list (for later use)
    lappend all_bdf_design_file_names_in_project $bdf_design_file_name
    # Remove the BDF file from the the list of all BDF files in the project
    # directory
    set all_bdf_design_file_names [lsearch -all -inline -not -exact
        $all_bdf_design_file_names $bdf_design_file_name]
    post_message -type info "    Found BDF file $bdf_design_file_name"

    # Test for design files outside of the current project directory and skip
    # them. The problem is that creating a VHDL file from such a BDF file
    # results in a VHDL file in the directory of the BDF file, not in the
    # project directory...
    if { [string compare [file tail $bdf_design_file_name] $bdf_design_file_name]
         != 0} {
        post_message -type critical_warning "Files outside the project directory
            are currently not supported! File skipped."
        continue
    }
    set vhdl_design_file_name [file tail [file rootname $bdf_design_file_name]]
    append vhdl_design_file_name ".vhd"

    set generate_vhdl_file 0
    if {![file exists $vhdl_design_file_name]} {
        # VHDL file does not exists and must be generated
        set generate_vhdl_file 1
        post_message -type info "    VHDL file does not exist, creating"
    } else {
        # VHDL file exists, check time stamp
        set vhdl_file_mtime [file mtime $vhdl_design_file_name]
        set bdf_file_mtime [file mtime $bdf_design_file_name]
        if {$vhdl_file_mtime < $bdf_file_mtime} {
            # VHDL file out of date
            set generate_vhdl_file 1
            post_message -type info "    VHDL file out of date, creating"
        }
    }

    if {$generate_vhdl_file == 1} {
        # Start the Quartus Mapper for generating VHDL description
        post_message -type info "exec quartus_map --read_settings_files=on
            --write_settings_files=off $current_revision -c $current_revision
            --convert_bdf_to_vhdl=$bdf_design_file_name"
        exec quartus_map --read_settings_files=on --write_settings_files=off
            $current_revision -c $current_revision
            --convert_bdf_to_vhdl=$bdf_design_file_name
    } else {
        post_message -type info "    VHDL file up to date, no need for creating"
    }
}
```

Listing 10: *Create VHDL files.*

Obviously, you don't want the last set of files to be converted (maybe you would, but then you have to add them to the project environment). Rationale for this is that we have students who create a lot of BDF files and then discard them from the project environment but forget them to delete. Note that this is done in just a few lines of code directly under the `foreach_in_collection` command.

Next, the script checks if the file is outside of the project directory. If the file is outside the project directory, a warning message is issued and the file is skipped for processing.

The last part of this stage converts the BDF files to VHDL files, but it only has to be done if the VHDL does not exists or if the VHDL file is outdated. This last one means that the BDF file was changed after the corresponding VHDL was created by the script. If a VHDL file should be created, the script starts a `quartus_map` command with the appropriate options.

At this point (this is the sixth stage), we have a list of all the BDF files that are not in the project environment (and hence they are not used by Quartus). All the corresponding VHDL files should be removed so that ModelSim will not accidentally use them during simulation. See Listing 11.

Obviously, this should be done by the users, but remind that this script was intentionally written to support students as an aid for learning digital design and not for learning how to use ModelSim.

By deleting all unwanted VHDL files we can set up a ModelSim command script containing a fine piece of code for including all VHDL files. See Listing 13 on page 14.

```
# All the BDF files in the project directory but NOT in the project environment
if { [llength $all_bdf_design_file_names] > 0} {
    post_message -type info "All remaining BDF files in project directory:
        $all_bdf_design_file_names"
} else {
    post_message -type info "No remaining BDF files in project directory"
}

# We remove all VHDL files for which a BDF file exists but not in the project
# environment. We do this so that ModelSim will not accidentally compile and
# load them.
foreach files $all_bdf_design_file_names {
    set vhdl_file_to_remove [file rootname $files]
    append vhdl_file_to_remove ".vhd"
    if {[file exists $vhdl_file_to_remove]} {
        post_message -type info "Removing VHDL file $vhdl_file_to_remove"
        file delete $vhdl_file_to_remove
    } else {
        post_message -type info "No VHDL file $vhdl_file_to_remove found."
    }
}
```

Listing 11: *Remove all VHDL files for which a BDF file exists but not in the project environment.*

The last two stages can best be seen in one view, as seen in Listing 12. For simulation to start, the filename where the top level entity resides should be the same as the name of the top level entity (For BDF files, the entity name is the same as the filename without the extension.) This is first checked, and the script issues an error if the check failed. By convention, and because the script can't handle command line arguments, the name of the command file that is passed as argument to ModelSim consists of the concatenation of the prefix `tb_` followed by the name of the top level entity name and the extension `.do` (this is the default extension for ModelSim command files). Then, if the command file exists, ModelSim is started. Please note the '&' at the end of the `exec` command. Doing this will prevent the execution of ModelSim to freeze the Quartus IDE. Of course, when the test fails, an error is issued.

```
# Check if the top level file name is what we expected.
if { [string compare [file rootname $top_level_entity_file_name]
    $top_level_entity] == 0 } {
```

```
    # Correct file name
    post_message -type info "Top level file name is correct."

    # Get user supplied ModelSim test bench file name, see if it is used.
    set modelsim_testbench_file_name [get_global_assignment -name
        EDA_SIMULATION_RUN_SCRIPT -section_id eda_simulation]
    set modelsim_testbench_enable_st [get_global_assignment -name
        EDA_TEST_BENCH_ENABLE_STATUS -section_id eda_simulation]
    if { ([string length $modelsim_testbench_file_name] > 0) && ([string equal
        $modelsim_testbench_enable_st "COMMAND_MACRO_MODE"] == 1) } {
        post_message -type info "Found user supplied command file."
    } else {
        # Check for ModelSim DO file name
        set modelsim_testbench_file_name "tb_${top_level_entity}.do"
    }
    if {[file exists $modelsim_testbench_file_name]} {
        # Found do file, start modelsim
        post_message -type info "Starting ModelSim with do-file
            $modelsim_testbench_file_name in background (frees Quartus IDE)"
        if { [catch { exec -ignorestderr ${modelsim_exec_path} -do
            $modelsim_testbench_file_name \& } result ] } {
            # Bummer, modelsim didn't start correctly...
            post_message -type error "ModelSim can't be started. Bailing out."
        }
    } else {
        # Bummer, do file not found or not by name convention
        post_message -type error "DO file not found or not by name convention (
            $modelsim_testbench_file_name). Bailing out."
    }
} else {
    # Incorrect file name
    post_message -type error "Top level file name is NOT correct (
        $top_level_entity_file_name,$top_level_entity). Bailing out."
}
```

Listing 12: *Check the top level and start ModelSim.*

Note that if the user provides a ModelSim command script via the Assignments menu, this script is used instead of the precomputed script.

6 Setting up a ModelSim command file

The ModelSim command file must contain all commands needed to fulfil the simulation. Normally, Quartus has the (annoying) habit of creating a few commands by itself and save them in a file with a strange name before it calls your command file. You can include compile command for every VHDL file the project contains, but as files are added or deleted, it is easy to include the code of Listing 13. This `foreach` loop compiles all VHDL files in the currect directory, which means it will also compile VHDL files that do not have an associated BDF file.

```
foreach vhd_file [ glob *.vhd ] {
    puts "Compiling: $vhd_file"
    vcom -2008 -work work $vhd_file
}
```

Listing 13: *Including all VHDL Files in a Modelsim Command File.*

When displaying signals, please note that the conversion create hierarchical names consisting of the prefix b2v_ and the *instantiation name*. Figure 1 shows some of the sub designs with instantiation names (actually, they are library elements).

So if you have a top level design entity called `full_adder` with a sub design entity called `xor` and the `xor` is instantiated with the name `inst` (the default, if you have more than one sub designs, they are called `inst1`, `inst2` etc.), using an internal signal `inva`, the signal in the xor design are labelled `full_adder/b2v_inst/inva` in ModelSim.

7 Installation and use of the script in a design flow

Installation of this script as a item in a so-called *design flow* is not complicated. You can create a flow by hand or use the example file in Listing 14. Creating a design flow is discussed in ()^{still not found}.

First, you should save the script in a directory. As you can see at the end of the Listing, we use `H:\QUARTUS\common` and name the script `start_sim.tcl` (the first version of the script just started the simulator as a test). The complete script can be found in Appendix A.

Then, simply create a file named `tmwc_BDF_Conversion_And_Simulation.tmf` (mind the extension) using a text editor and copy the the code from Listing 14 in Appendix A into the file. Then move the file into your Windows profile directory, which is usually something like `C:\users\<your_name>`. On Linux you have to place the file in a directory called `.quartus.altera` which is generated in your home directory when you start Quartus for the first time.

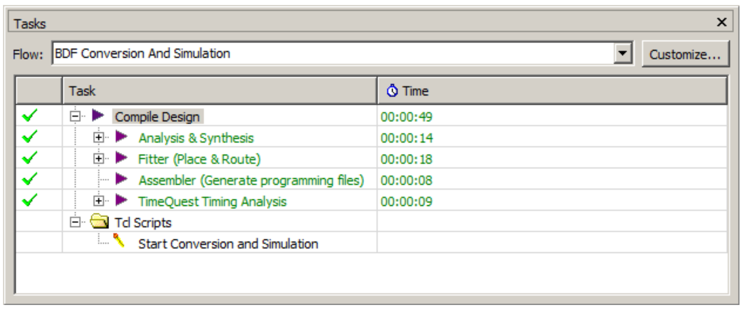
Note that the first part of code contains some predefined tasks such as synthesis, fitter and timing analyser. The second part of the code describes the whereabouts of the script. Please mind the code fragment where the path of the script is presented. It should match the path where you placed the script. Note: do *not* include whitespaces in the script's path!

Appendix C shows an example Windows Visual Basic script to automatically install the script and the design flow file in the user's profile directory.

```xml
<?xml version= "1.0"?>
<!--
* File name:    tmwc_BDF_Conversion_And_Simulation.tmf
* Date:         5 mar 2014
* Version:      1.1
* Author:       Jesse op den Brouw
*
* Description:  Flow File for Compiling A Quartus Project
                and Converting/Simulating BDF Files
*****************************************************************
-->
<tasks flowname = "BDF Conversion And Simulation" type = "user">
    <predefined_tasks>
        <id>fsm_viewer</id>
        <id>netlist_viewers</id>
        <id>open_chip_planner</id>
        <id>rtl_viewer</id>
        <id>start_analysis_elaboration</id>
        <id>start_analysis_synthesis</id>
        <id>start_assembler</id>
        <id>start_design_assistant_post_map</id>
        <id>start_fitter</id>
        <id>start_full_compilation</id>
        <id>start_partition_merge</id>
        <id>start_timing_analyzer</id>
        <id>tech_map_viewer_post_fit</id>
        <id>tech_map_viewer_post_map</id>
        <id>timequest_assignments</id>
    </predefined_tasks>
    <task>
        <id>Start Conversion and Simulation</id>
        <name>Start Conversion and Simulation</name>
        <item_bitmap>tcl_command</item_bitmap>
        <status_ok_if>project_is_open</status_ok_if>
        <action type = "tcl_command">H:/QUARTUS/common/start_sim.tcl</action>
    </task>
</tasks>
```

Listing 14: *The Design Flow File*

After you installed the appropriate files, you just simple start the Quartus software and select the BDF design flow from the Tasks pane. See Figure 4.



Tasks

Flow: BDF Conversion And Simulation

Customize...

	Task	Time
✓	Compile Design	00:00:49
✓	Analysis & Synthesis	00:00:14
✓	Fitter (Place & Route)	00:00:18
✓	Assembler (Generate programming files)	00:00:08
✓	TimeQuest Timing Analysis	00:00:09
	Tcl Scripts	
	Start Conversion and Simulation	

Figure 4: *The BDF Conversion And Simulation Design Flow.*

8 Things to Remember

Please note the following when creating your schematics:

- Do *not* use VHDL constructs and keywords anywhere in the schematics. This applies to anything that may render the created VHDL files containing unwanted and probably illegal VHDL constructs, such as keywords, entity names and expressions. Examples are: `is`, `in`, `out`, `of`, `signal`, `A<B`.
- Try to avoid uppercase characters, especially is names for input and output pins. VHDL is case insensitive (and so is ModelSim), but the pin planner isn't.

9 Known Issues

There are some known issues, which are all caused by the Quartus environment.

- If you change the top level design entry and run the script directly after that, the script will find the previous selected top level design entry as the current one. The workaround is simple: just rerun Analysis and Synthesis first and you'll be fine.

- Currently the script can't handle BDF files outside the project directory. The problem is that the Quartus command `quartus_map` has no option to specify the output directory. It just uses the directory of the input file. This can interfere with VHDL files in that directory.
- The script also can't handle library elements because of the aforementioned problem. (Note that the use of Mega Functions en MAXII functions is not tested.)
- Output from the script is first seen when ModelSim starts. Before that, all output is apparently internally buffered. This is annoying, but up till now there's no remedy for that.
- If a component in your design has an unused output (e.g., a carry out of the most significant bit of a full adder), Quartus will not generate an output signal in the VHDL file, hence you cannot use the signal in ModelSim.

A The Complete Tcl Script

The complete script can also be downloaded from https://github.com/jesseopdenbrouw/Convert-BDF-to-VHD

```
#
# TCL script for creating VHDL descriptions from a Block Design Files
# and starts the ModelsSim simulator with the top level design.
# Hierarchies are supported.
#
# Note: Your top level design entity must reside in a file with the name
#       <toplevel>.bdf or <toplevel>.vhd.
#       You must have a DO file with the name tb_<toplevel>.do
#       for simulation to work. Your DO file must contain all ModelSim
#       commands for the simulation to work (thus also all vcom and
#       vsim commands).
#
# Version 1.3beta11  -  2021/02/02
#
# (c)2021, Jesse op den Brouw, <j.e.j.opdenbrouw@hhs.nl>
# (c)2021, De Haagse Hogeschool [www.hhs.nl]
#
#
# This script works as follows:
#
#  1. It checks if the project is open.
#  2. Finds ModelSim execution path is none is provided. Linux and
#     Windows supported.
#  3. Creates a project database is none is found.
#  4. Finds the top level, checks if the top level entity has an
#     associated file, complains if none is found.
#  5. Loops through all BDF files found in the project environment
#     and creates accompanied VHDL files if needed.
#  6. Removes all VHDL files from accompanied BDF files IN the project
#     directory but NOT in the project environment, but not VHDL files
#     that do NOT have a accompanied BDF file.
#  7. Finds top level filename and creates DO filename.
#  8. Starts ModelSim with DO filename.
#
# Bugs: currently the script can't handle BDF-files outside of the
#       project directory.
#       currenty the script can't handle library elements
#
# Todo: hardening for use on Unices other than Linux
#       option to create Verilog files instead of VHDL files
#       testing on multiple revisions
#       handle files outside the project directory
#       handle library elements

# User input: set to the modelsim path. Keep empty for autodetect.
#set modelsim_exec_path "/opt/altera/12.1sp1/modelsim_ase/linuxaloem/vsim"
set modelsim_exec_path ""

# Print a nice banner
post_message -type info "############################################"
post_message -type info "BDF to VHDL converter & ModelSim Starter v1.3"
post_message -type info "############################################"

# Check for project opened.
if {![is_project_open]} {
    post_message -type error "There's no project open! Please open a project and
```

```
        rerun."
    return False
}

# Export any newly changed/added/removed assignments
export_assignments

# Autodetect ModelSim exec path if none is provided. First, the user
# preferences are consulted, then ModelSim is autodetected. For this to
# work, ModelSim must be installed within the Quartus environment.
if { [string length $modelsim_exec_path] == 0 } {
    post_message -type info "Autodetecting ModelSim path..."
    set opsys [string tolower [lindex $::tcl_platform(os) 0]]
    post_message -type info "OS is: $opsys"

    # Try to get ModelSim path from user preferences
    set modelsim_exec_path [get_user_option -name EDA_TOOL_PATH_MODELSIM_ALTERA]

    if { [string length $modelsim_exec_path] > 0} {
        # User has entered a path in EDA Tool Options...
        post_message -type info "Found user preference path: $modelsim_exec_path"
        set modelsim_exec_path [string map {"\\" "/"} $modelsim_exec_path]
        # Different OSes...
        switch $opsys {
            linux { append modelsim_exec_path "/vsim" }
            windows { append modelsim_exec_path "/modelsim.exe" }
            default { post_message -type error "Cannot continue: unknowm platform
                 is $opsys. Bailing out."
                return False }
        }
    } else {
        # Tries to find a ModelSim installation directory. Stops if found none is
             found.
        # Stops if more than one found AND one of them is NOT ModelSim ASE.
        # Continues if ModelSim ASE is found in multiple ModelSIm installations.
        set modelsim_exec_path $quartus(quartus_rootpath)
        append modelsim_exec_path "../"
        switch [llength [set modelsim_list [ glob -nocomplain -path
            $modelsim_exec_path modelsim* ]]] {
            0 { post_message -type error "ModelSim not installed in Quartus
                environment! Bailing out."
                return False }
            1 { set modelsim_exec_path [lindex $modelsim_list 0] }
            default { set modelsim_exec_path [lindex $modelsim_list [lsearch
                $modelsim_list *modelsim_ase]]
                if { [string length $modelsim_exec_path] > 0} {
                    post_message -type info "Found ModelSim path for Altera
                        Starter Edition (modelsim_ase)."
                } else {
                    post_message -type error "Multiple ModelSim installations
                        found! Bailing out."
                    return False
                }
            }
        }
        # Different OSes...
        switch $opsys {
            linux { append modelsim_exec_path "/linuxaloem/vsim" }
            windows { append modelsim_exec_path "/win32aloem/modelsim.exe" }
            default { post_message -type error "Cannot continue: unknowm platform
                 is $opsys. Bailing out."
```

```
                    return False }
        }
    }
}

# Normalize path name (get rid of ../ and ./ etc)
set modelsim_exec_path [file normalize $modelsim_exec_path]
post_message -type info "ModelSim path: $modelsim_exec_path"

# Check if the ModelSim executable is executable...
if { [file executable $modelsim_exec_path] == 0 } {
    post_message -type error "ModelSim executable cannot be run by current user.
        Bailing out!" -submsgs {"You should check the path to the executable in
        this script or via menu" "Tools->Options->EDA Tool Options or your
        ModelSim installation is corrupt."}
    return False
}

# Set the project directory. This is needed because if you have added a BDF file
# that is not in the project directory (e.g. ../<some_other_dir>/file.bdf),
# Quartus changes the current directory path ([pwd]). Yes, really, it does...
# Please note that BDF files outside the project directory are not supported.
# The problem is that Quartus creates a VHDL file in that same directory. This
# could overwrite an existing file. There’s no option to provide an output
# directory.
set project_directory [get_project_directory]
post_message -type info "Project directory: $project_directory"
cd $project_directory

# Get current revision
# Check if there is a database. If not, create one.
set current_revision ""
if { [catch {get_top_level_entity}] } {
    set current_revision [get_current_revision]
    post_message -type info "There’s no compiler database, running Analysis &
        Synthesis with revision name $current_revision"
    # Running Analysis & Synthesis currenly crashes when there’s no file
    # containing the toplevel, that is, you have sole QPF and QSF files,
    # or there’s no way to systhesize the design (or any generic error for
    # that matter).
    set status [catch { exec quartus_map --read_settings_files=on
        --write_settings_files=off $current_revision -c $current_revision }
        result]
    if { $status != 0 } {
        post_message -type error "Creating database failed! There are five
            posibilities:"
        post_message -type error "1: you have a project without a file containing
             the top level description."
        post_message -type error "2: you have a design that cannot be
            synthesized."
        post_message -type error "3: you have an error in (one of) your design
            file(s)."
        post_message -type error "4: the current device is not supported in this
            version of Quartus."
        post_message -type error "5: you changed the target device and/or the
            device files of the previous/current device are not installed."
        post_message -type error "You can try rerunning the script. Bailing out."
        return False
    }
} else {
    set current_revision [get_current_revision]
```

```
}
# Echo the current revision
post_message -type info "Current revision: $current_revision"

# Find top level entity currently !focused! See Quartus:
# Assignments->Settings->General->Top Level Entity
set top_level_entity [get_name_info -info entity_name [get_top_level_entity]]
post_message -type info "Found top level entity : $top_level_entity"

# Find the file containing top level entity
set top_level_entity_file_name [get_name_info -info file_location [
    get_top_level_entity]]
post_message -type info "Found top level entity file name :
    $top_level_entity_file_name"

# Check for empty top level filename. Does happen when in a completely stripped
# project the simulation is started (that is: there's no file containing the
# top level entity, but there is a project database).
if { [string compare $top_level_entity_file_name  ""] == 0} {
    post_message -type error "Top level filename is empty. Please enter a file,
        rerun Analysis & Synthesis and start again. Bailing out."
    return False;
}

# Create a list of all BDF files in the project DIRECTORY. We need this list
# for later on. If a BDF file is in the project environment, we remove it from
# this list. At the end we have list of BDF files in the project directory but
# not in the project environment.
set all_bdf_design_file_names [glob -nocomplain -type f *.bdf]

# Find all BDF files in project. This excludes BDF files that are in the
# project directory but not in the project environment. For all BDF files in
# the project, create a VHDL file if needed. Currently works for current
# directory level.
post_message -type info "Looping through all BDF-files in project"

set all_bdf_design_file_names_in_project ""
foreach_in_collection asgn_id [get_all_assignments -type global -name BDF_FILE] {

    # Get next BDF file name
    set bdf_design_file_name  [get_assignment_info $asgn_id -value]
    # Add to list (for later use)
    lappend all_bdf_design_file_names_in_project $bdf_design_file_name
    # Remove the BDF file from the the list of all BDF files in the project
    # directory
    set all_bdf_design_file_names [lsearch -all -inline -not -exact
        $all_bdf_design_file_names $bdf_design_file_name]
    post_message -type info "    Found BDF file $bdf_design_file_name"

    # Test for design files outside of the current project directory and skip
    # them. The problem is that creating a VHDL file from such a BDF file
    # results in a VHDL file in the directory of the BDF file, not in the
    # project directory...
    if { [string compare [file tail $bdf_design_file_name] $bdf_design_file_name]
         != 0} {
        post_message -type critical_warning "Files outside the project directory
            are currently not supported! File skipped."
        continue
    }
    set vhdl_design_file_name [file tail [file rootname $bdf_design_file_name]]
    append vhdl_design_file_name ".vhd"
    set generate_vhdl_file 0
    if {![file exists $vhdl_design_file_name]} {
        # VHDL file does not exists and must be generated
        set generate_vhdl_file 1
        post_message -type info "    VHDL file does not exist, creating"
    } else {
        # VHDL file exists, check time stamp
        set vhdl_file_mtime [file mtime $vhdl_design_file_name]
        set bdf_file_mtime [file mtime $bdf_design_file_name]
        if {$vhdl_file_mtime < $bdf_file_mtime} {
            # VHDL file out of date
            set generate_vhdl_file 1
            post_message -type info "    VHDL file out of date, creating"
        }
    }

    if {$generate_vhdl_file == 1} {
        # Start the Quartus Mapper for generating VHDL description
        post_message -type info "exec quartus_map --read_settings_files=on
            --write_settings_files=off $current_revision -c $current_revision
            --convert_bdf_to_vhdl=$bdf_design_file_name"
        exec quartus_map --read_settings_files=on --write_settings_files=off
            $current_revision -c $current_revision
            --convert_bdf_to_vhdl=$bdf_design_file_name
    } else {
        post_message -type info "    VHDL file up to date, no need for creating"
    }
}

# All the BDF files in the project directory but NOT in the project environment
if { [llength $all_bdf_design_file_names] > 0} {
    post_message -type info "All remaining BDF files in project directory:
        $all_bdf_design_file_names"
} else {
    post_message -type info "No remaining BDF files in project directory"
}

# We remove all VHDL files for which a BDF file exists but not in the project
# environment. We do this so that ModelSim will not accidentally compile and
# load them.
foreach files $all_bdf_design_file_names {
    set vhdl_file_to_remove [file rootname $files]
    append vhdl_file_to_remove ".vhd"
    if {[file exists $vhdl_file_to_remove]} {
        post_message -type info "Removing VHDL file $vhdl_file_to_remove"
        file delete $vhdl_file_to_remove
    } else {
        post_message -type info "No VHDL file $vhdl_file_to_remove found."
    }
}

# Check if the top level file name is what we expected.
if { [string compare [file rootname $top_level_entity_file_name]
    $top_level_entity] == 0 } {
    # Correct file name
    post_message -type info "Top level file name is correct."

    # Get user supplied ModelSim test bench file name, see if it is used.
    set modelsim_testbench_file_name [get_global_assignment -name
        EDA_SIMULATION_RUN_SCRIPT -section_id eda_simulation]
    set modelsim_testbench_enable_st [get_global_assignment -name
        EDA_TEST_BENCH_ENABLE_STATUS -section_id eda_simulation]
    if { ([string length $modelsim_testbench_file_name] > 0) && ([string equal
        $modelsim_testbench_enable_st "COMMAND_MACRO_MODE"] == 1) } {
        post_message -type info "Found user supplied command file."
    } else {
        # Check for ModelSim DO file name
        set modelsim_testbench_file_name "tb_${top_level_entity}.do"
    }
    if {[file exists $modelsim_testbench_file_name]} {
        # Found do file, start modelsim
        post_message -type info "Starting ModelSim with do-file
            $modelsim_testbench_file_name in background (frees Quartus IDE)"
        if { [catch { exec -ignorestderr ${modelsim_exec_path} -do
            $modelsim_testbench_file_name \& } result ] } {
            # Bummer, modelsim didn't start correctly...
            post_message -type error "ModelSim can't be started. Bailing out."
        }
    } else {
        # Bummer, do file not found or not by name convention
        post_message -type error "DO file not found or not by name convention (
            $modelsim_testbench_file_name). Bailing out."
    }
} else {
    # Incorrect file name
    post_message -type error "Top level file name is NOT correct (
        $top_level_entity_file_name,$top_level_entity). Bailing out."
}
```

Listing 15: *The Complete Script.*

B The Design Flow File

The complete design flow file can also be downloaded from https://github.com/jesseopdenbrouw/Convert-BDF-to-VHDL.

```
<?xml version= "1.0"?>
<!--
* File name:     tmwc_BDF_Conversion_And_Simulation.tmf
* Date:          5 mar 2014
* Version:       1.1
* Author:        Jesse op den Brouw
*
* Description:   Flow File for Compiling A Quartus Project
                 and Converting/Simulating BDF Files
****************************************************************
-->
<tasks flowname = "BDF Conversion And Simulation" type = "user">
    <predefined_tasks>
        <id>fsm_viewer</id>
        <id>netlist_viewers</id>
        <id>open_chip_planner</id>
        <id>rtl_viewer</id>
        <id>start_analysis_elaboration</id>
        <id>start_analysis_synthesis</id>
        <id>start_assembler</id>
        <id>start_design_assistant_post_map</id>
        <id>start_fitter</id>
        <id>start_full_compilation</id>
        <id>start_partition_merge</id>
        <id>start_timing_analyzer</id>
        <id>tech_map_viewer_post_fit</id>
        <id>tech_map_viewer_post_map</id>
        <id>timequest_assignments</id>
    </predefined_tasks>
    <task>
        <id>Start Conversion and Simulation</id>
        <name>Start Conversion and Simulation</name>
        <item_bitmap>tcl_command</item_bitmap>
        <status_ok_if>project_is_open</status_ok_if>
        <action type = "tcl_command">H:/QUARTUS/common/start_sim.tcl</action>
    </task>
</tasks>
```

Listing 16: *The Design Flow File*

C An Example of a Install Script on Windows

```
'
'    install_flow.vbs
'
' (c)2018, J. op den Brouw <J.E.J.opdenBrouw@hhs.nl>
'
' This program installs the INLDIG flow file into the user's profile folder
'

' Warn if undefined variables
Option Explicit

' Constants for file manipulation
Const fsoForReading = 1
Const fsoForWriting = 2
Const fsoForAppending = 8

' Declare variables
Dim startsimfilename
Dim objWSHshell
Dim userProfileFolder
Dim fso
Dim currentFolderName
Dim flowFileName
Dim objFile
Dim oldContent
Dim newContent
Dim result

' The script to start from the flow
startsimfilename = "start_sim.tcl"
' The flow file
flowFileName = "tmwc_BDF_Conversion_And_Simulation.tmf"

' Find user profile folder
Set objWSHshell = WScript.CreateObject( "WScript.Shell" )
userProfileFolder = objWSHshell.ExpandEnvironmentStrings( "%USERPROFILE%" )

'Wscript.Echo "The user profile folder is " & userProfileFolder & vbCrLf & "Press
     OK to continue"

' Find current folder
Set fso = CreateObject("Scripting.FileSystemObject")
currentFolderName = fso.GetParentFolderName(WScript.ScriptFullName)

' Check if script file exists, bail out if not
If not fso.FileExists(fso.BuildPath(currentFolderName, startsimfilename)) Then
  MsgBox "Missing " & startsimfilename & "! Cannot continue!" & vbCrLf & "Press
      OK to stop execution", vbCritical, "ERROR!"
  Wscript.Quit
End If

' Check if flow file exists, bail out if not
If not fso.FileExists(fso.BuildPath(currentFolderName, flowFileName)) Then
  MsgBox "Missing " & flowFileName & "! Cannot continue!" & vbCrLf & "Press OK to
       stop execution", vbCritical, "ERROR!"
  Wscript.Quit
End If
```

```
'MsgBox "File " & startsimfilename & " exists in current folder." & vbCrLf & "
    Press OK to continue", vbOKOnly, "Copying scripts for INLDIG"

'Read in flow file
set objFile=fso.OpenTextFile(fso.BuildPath(currentFolderName, flowFileName),
    fsoForReading)
oldContent=objFile.ReadAll
objFile.Close
'MsgBox oldContent

' Now change all \ for / ...
currentFolderName = Replace(currentFolderName, "\", "/")
'Wscript.Echo "The current folder is " & currentFolderName & vbCrLf & "Press OK
    to continue"

' Replace default path name with new path name
newContent=replace(oldContent,"H:/QUARTUS/INLDIG/common/" & startsimfilename, chr
    (34) & currentFolderName & "/" & startsimfilename & chr(34) ,1,-1,0)
'MsgBox newContent

' Give the user the option to continue the installation or to abort
result = MsgBox ("Ready to install the flow file in the user's profile folder." &
     vbCrLf & "Press OK to continue, Cancel to cancel.", vbOKCancel, "Install
    scripts for Quartus INLDIG flow")
If result = vbCancel Then
  MsgBox "Cancelling installation", vbOKOnly, "Cancel pressed"
  Wscript.Quit
End If

' Trap any errors
Err.Clear
On Error Resume Next

'Write flow file to user profile folder
set objFile=fso.OpenTextFile(fso.BuildPath(userProfileFolder, flowFileName),
    fsoForWriting, true)
If Err.Number <> 0 Then
  MsgBox "Could not open flow file for writing!" & vbCrLf & "Press OK to stop
      execution", vbCritical, "ERROR!"
  Wscript.Quit
End If
Err.Clear
objFile.Write newContent
If Err.Number <> 0 Then
  MsgBox "Could not write the content to the flow file!" & vbCrLf & "Press OK to
      stop execution", vbCritical, "ERROR!"
  Wscript.Quit
End If
objFile.Close
Err.Clear

' Give the user visual feedback
MsgBox "Installed the flow file in the user's profile folder." & vbCrLf & "Press
    OK to end the installation.", vbOKOnly, "Install scripts for Quartus INLDIG
    flow"
```

Listing 17: *An Example of a Install Script on Windows*

D Changelog & To Do

Changelog to the documentation:

22-01-2014: Added failure feature 4 and 5 if analysis fails.

05-03-2014: Changed TMF file name to reflect the fact that BDF files are converted.

08-03-2014: Minor spelling corrections.

09-03-2014: New screenshot BDF Conversion And Simulation

19-09-2018: Updated the script, changed command batch file for VBscript, typos.

27-08-2021: Update the script and documentation.

References

[1] De Haagse Hogeschool. *The Hague University of Applied Sciences, English Version*. 2018. URL: http://www.thehagueuniversity.com/ (visited on 09/19/2018) (cit. on p. 3).

[2] Terasic. *DE0 Digital Systems Board*. URL: http://www.terasic.com.tw/cgi-bin/page/archive.pl?No=364 (cit. on p. 3).

[3] Chris Zeh. *TCL Macro for Top-Level Schematic to Verilog Conversion (For ModelSim Simulation)*. URL: http://idle-logic.com/2012/06/09/tcl-macro-for-top-level-schematic-to-verilog-conversion-for-modelsim-simulation/ (cit. on p. 4).

[4] –. *Introductory Course on Tcl*. 2013. URL: http://www.tcl.tk/man/tcl8.5/tutorial/Tcl0.html (visited on 08/08/2013) (cit. on p. 5).

[5] Brent B. Welch, Ken Jones, and Jeffrey Hibbs. *Practical Programming in Tcl and Tk*. 4th. Upper Saddle River, NJ: Pearson Education Inc., 2003 (cit. on p. 5).

[6] Altera. *Quartus Project Tcl Package*. 2012. URL: https://www.intel.com/content/www/us/en/programmable/quartushelp/13.1/mergedProjects/tafs/tafs/tcl_pkg_project_ver_6.0.htm (visited on 09/19/2018) (cit. on pp. 5, 7).

[7] Altera. *Tcl Scripting*. 2013. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/tclscriptrefmnl.pdf (visited on 09/19/2018) (cit. on p. 6).

[8] Altera. *Quartus II Tcl Examples*. 2012. URL: https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/tcl.html (visited on 09/19/2018) (cit. on p. 6).