Command Line Scripting 2

2014.06.30

QII52002 Subscribe Send Feedback

FPGA design software that easily integrates into your design flow saves time and improves productivity. The Altera® Quartus® II software provides you with a command-line executable for each step of the FPGA design flow to make the design process customizable and flexible.

The benefits provided by command-line executables include:

- Command-line control over each step of the design flow
- Easy integration with scripted design flows including makefiles
- Reduced memory requirements
- Improved performance

  The command-line executables are also completely interchangable with the Quartus II GUI, allowing you to use the exact combination of tools that you prefer.

Benefits of Command-Line Executables

The Quartus II command-line executables provide control over each step of the design flow. Each executable includes options to control commonly used software settings. Each executable also provides detailed, built-in help describing its function, available options, and settings.

Command-line executables allow for easy integration with scripted design flows. You can easily create scripts with a series of commands. These scripts can be batch-processed, allowing for integration with distributed computing in server farms. You can also integrate the Quartus II command-line executables in makefile-based design flows. These features enhance the ease of integration between the Quartus II software and other EDA synthesis, simulation, and verification software.

Command-line executables add flexibility without sacrificing the ease-of-use of the Quartus II GUI. You can use the Quartus II GUI and command-line executables at different stages in the design flow. For example, you might use the Quartus II GUI to edit the floorplan for the design, use the command-line executables to perform place-and-route, and return to the Quartus II GUI to perform debugging with the Chip Editor.

Command-line executables reduce the amount of memory required during each step in the design flow. Because each executable targets only one step in the design flow, the executables themselves are relatively compact, both in file size and the amount of memory used during processing. This memory usage reduction improves performance, and is particularly beneficial in design environments where heavy usage of computing resources results in reduced memory availability.

Related Information

Using the Quartus II Executables in Shell Scripts

Introductory Example

The following introduction to command-line executables demonstrates how to create a project, fit the design, and generate programming files.

The tutorial design included with the Quartus II software is used to demonstrate this functionality. If installed, the tutorial design is found in the <Quartus II directory>**/qdesigns/fir_filter** directory.

Before making changes, copy the tutorial directory and type the four commands shown in the introductory example below at a command prompt in the new project directory.

The <Quartus II directory>**/quartus/bin** directory must be in your `PATH` environment variable.

```
quartus_map filtref --source=filtref.bdf --family="Cyclone V"
quartus_fit filtref --part=EP3C10F256C8 --pack_register=minimize_area
quartus_asm filtref
quartus_sta filtref
```

The `quartus_map filtref --source=filtref.bdf --family="Cyclone V"` command creates a new Quartus II project called **filtref** with **filtref.bdf** as the top-level file. It targets the Cyclone® V device family and performs logic synthesis and technology mapping on the design files.

The `quartus_fit filtref --part=EP3C10F256C8 --pack_register=minimize_area` command performs fitting on the **filtref** project. This command specifies an EP3C10F256C8 device, and the `--pack_register=minimize_area` option causes the Fitter to pack sequential and combinational functions into single logic cells to reduce device resource usage.

The `quartus_asm filtref` command creates programming files for the **filtref** project.

The `quartus_sta filtref` command performs basic timing analysis on the **filtref** project using the Quartus II TimeQuest Timing Analyzer, reporting worst-case setup slack, worst-case hold slack, and other measurements.

You can put the four commands from the introductory example into a batch file or script file, and run them. For example, you can create a simple UNIX shell script called **compile.sh**, which includes the code shown in the UNIX shell script example below.

```
#!/bin/sh
PROJECT=filtref
TOP_LEVEL_FILE=filtref.bdf
FAMILY="Cyclone V"
PART=EP3C10F256C8
PACKING_OPTION=minimize_area
quartus_map $PROJECT --source=$TOP_LEVEL_FILE --family=$FAMILY
quartus_fit $PROJECT --part=$PART --pack_register=$PACKING_OPTION
quartus_asm $PROJECT
quartus_sta $PROJECT
```

Edit the script as necessary and compile your project.

Related Information

About Design Space Explorer

For more information about using all of the features of the quartus_sta executable. The TimeQuest Timing Analyzer employs Synopsys Design Constraints to fully analyze the timing of your design.

Command-Line Scripting Help

Help for command-line executables is available through different methods. You can access help built in to the executables with command-line options. You can use the Quartus II Command-Line and Tcl API Help browser for an easy graphical view of the help information.

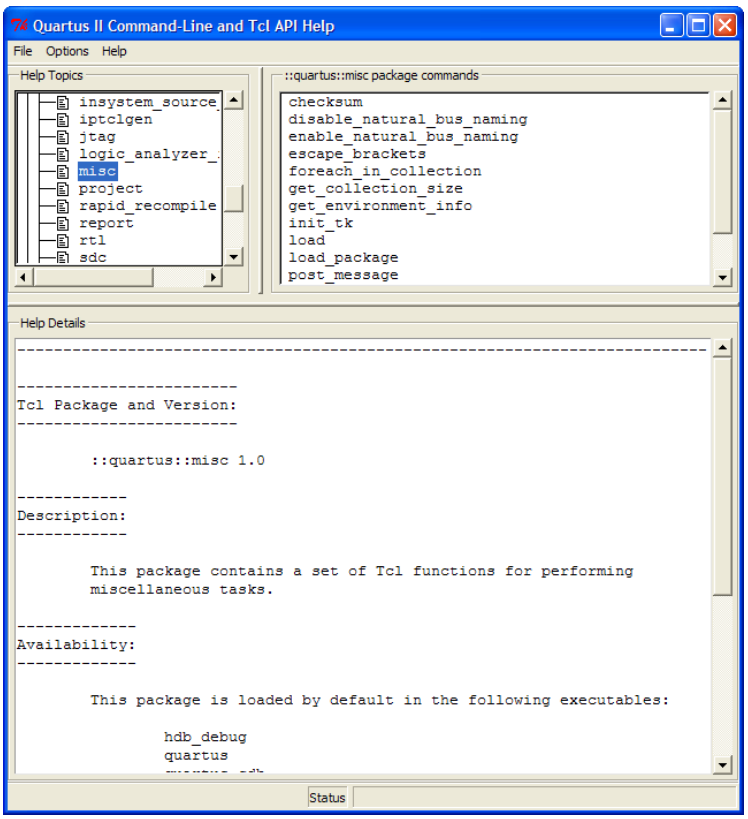
To use the Quartus II Command-Line and Tcl API Help browser, type the following command:

```
quartus_sh --qhelp
```

This command starts the Quartus II Command-Line and Tcl API Help browser, a viewer for information about the Quartus II Command-Line executables and Tcl API.

Use the `-h` option with any of the Quartus II Command-Line executables to get a description and list of supported options. Use the `--help=`<*option name*> option for detailed information about each option.

Figure 2-1: Quartus II Command-Line and Tcl API Help Browser



Project Settings with Command-Line Options

Command-line options are provided for many common global project settings and for performing common tasks.

You can use either of two methods to make assignments to an individual entity. If the project exists, open the project in the Quartus II GUI, change the assignment, and close the project. The changed assignment is updated in the **.qsf**. Any command-line executables that are run after this update use the updated assignment.

You can also make assignments using the Quartus II Tcl scripting API. If you want to completely script the creation of a Quartus II project, choose this method.

Related Information

- **Option Precedence** on page 2-4
- **Tcl Scripting**
- **QSF Reference Manual**

Option Precedence

If you use command-line executables, you must be aware of the precedence of various project assignments and how to control the precedence. Assignments for a particular project exist in the Quartus II Settings File (**.qsf**) for the project. Before the **.qsf** is updated after assignment changes, the updated assignments are reflected in compiler database files that hold intermediate compilation results.

All command-line options override any conflicting assignments found in the **.qsf** or the compiler database files. There are two command-line options to specify whether the **.qsf** or compiler database files take precedence for any assignments not specified as command-line options.

Any assignment not specified as a command-line option or found in the **.qsf** or compiler database file is set to its default value.

The file precedence command-line options are `--read_settings_files` and `--write_settings_files`.

By default, the `--read_settings_files` and `--write_settings_files` options are turned on. Turning on the `--read_settings_files` option causes a command-line executable to read assignments from the **.qsf** instead of from the compiler database files. Turning on the `--write_settings_files` option causes a command-line executable to update the **.qsf** to reflect any specified options, as happens when you close a project in the Quartus II GUI.

If you use command-line executables, be aware of the precedence of various project assignments and how to control the precedence. Assignments for a particular project can exist in three places:

- The **.qsf** for the project
- The result of the last compilation, in the **/db** directory, which reflects the assignments that existed when the project was compiled

- Command-line options

The precedence for reading assignments depends on the value of the `--read_settings_files` option.

Table 2-1: Precedence for Reading Assignments

Option Specified	Precedence for Reading Assignments
`--read_settings_files = on` (Default)	Command-line options The **.qsf** for the project Project database (db directory, if it exists) Quartus II software defaults
`--read_settings_files = off`	Command-line options Project database (db directory, if it exists) Quartus II software defaults

The table lists the locations to which assignments are written, depending on the value of the `--write_settings_files` command-line option.

Table 2-2: Location for Writing Assignments

Option Specified	Location for Writing Assignments
`--write_settings_files = on` (Default)	**.qsf** and compiler database
`--write_settings_files = off`	Compiler database

The example assumes that a project named **fir_filter** exists, and that the analysis and synthesis step has been performed (using the `quartus_map` executable).

```
quartus_fit fir_filter --pack_register=off
quartus_sta fir_filter
mv fir_filter_sta.rpt fir_filter_1_sta.rpt
quartus_fit fir_filter --pack_register=minimize_area     --write_settings_files=off

quartus_sta fir_filter
mv fir_filter_sta.rpt fir_filter_2_sta.rpt
```

The first command, `quartus_fit fir_filter` --pack_register=off, runs the `quartus_fit` executable with no aggressive attempts to reduce device resource usage.

The second command, `quartus_sta fir_filter`, performs basic timing analysis for the results of the previous fit.

The third command uses the UNIX `mv` command to copy the report file output from **quartus_sta** to a file with a new name, so that the results are not overwritten by subsequent timing analysis.

The fourth command runs **quartus_fit** a second time, and directs it to attempt to pack logic into registers to reduce device resource usage. With the `--write_settings_files=off` option, the command-line executable does not update the **.qsf** to reflect the changed register packing setting. Instead, only the

compiler database files reflect the changed setting. If the `--write_settings_files=off` option is not specified, the command-line executable updates the **.qsf** to reflect the register packing setting.

The fifth command reruns timing analysis, and the sixth command renames the report file, so that it is not overwritten by subsequent timing anlysis.

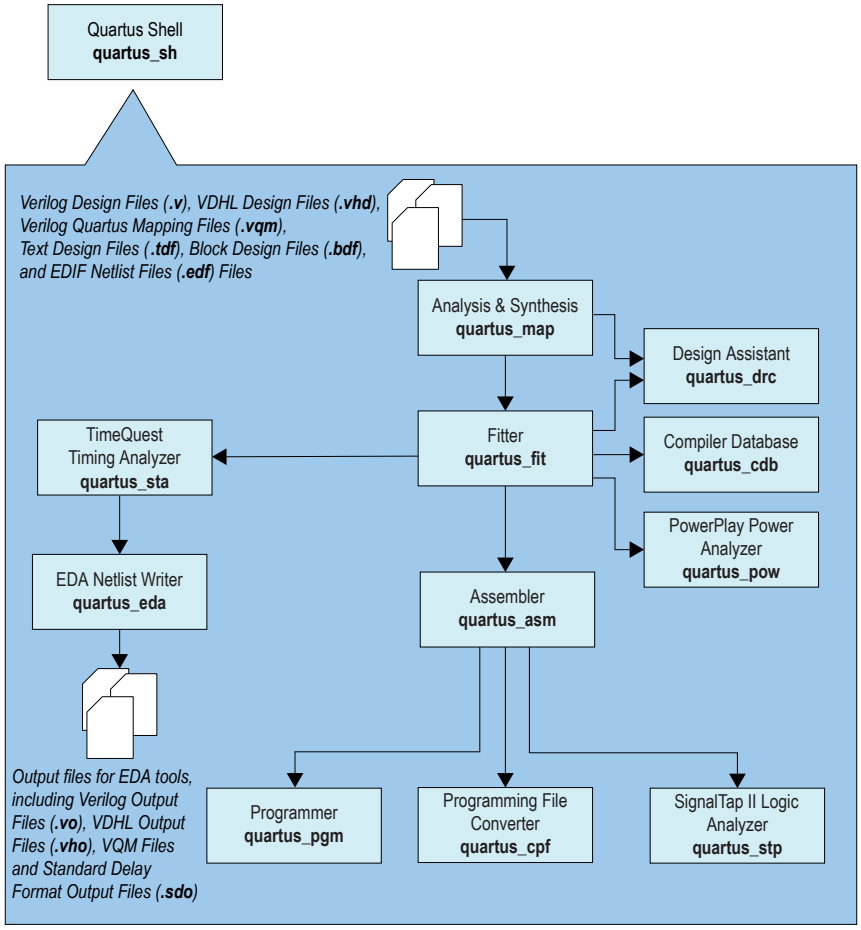
Use the options `--read_settings_files=off` and `--write_settings_files=off` (where appropriate) to optimize the way that the Quartus II software reads and updates settings files. In the following example, the `quartus_asm` executable does not read or write settings files because doing so would not change any settings for the project.

```
quartus_map filtref --source=filtref --part=EP3C10F256C8
quartus_fit filtref --pack_register=off --read_settings_files=off
quartus_asm filtref --read_settings_files=off --write_settings_files=off
```

Compilation with quartus_sh --flow

The figure shows a typical Quartus II FPGA design flow using command-line executables.

Figure 2-2: Typical Design Flow



Use the `quartus_sh` executable with the `--flow` option to perform a complete compilation flow with a single command. The `--flow` option supports the smart recompile feature and efficiently sets command-line arguments for each executable in the flow.

The following example runs compilation, timing analysis, and programming file generation with a single command:

```
quartus_sh --flow compile filtref
```

Tip: For information about specialized flows, type `quartus_sh --help=flow` at a command prompt.

Text-Based Report Files

Each command-line executable creates a text report file when it is run. These files report success or failure, and contain information about the processing performed by the executable.

Report file names contain the revision name and the short-form name of the executable that generated the report file, in the format <revision>.<executable>.**rpt**. For example, using the **quartus_fit** executable to place and route a project with the revision name **design_top** generates a report file named **design_top.fit.rpt**. Similarly, using the **quartus_sta** executable to perform timing analysis on a project with the revision name **fir_filter** generates a report file named **fir_filter.sta.rpt**.

As an alternative to parsing text-based report files, you can use the **::quartus::report** Tcl package.

Related Information

- **Tcl Scripting**
- **About Quartus II Tcl Scripting**

Using Command-Line Executables In Scripts

You can use command-line executables in scripts that control other software in addition to the Quartus II software. For example, if your design flow uses third-party synthesis or simulation software, and if you can run the other software at a command prompt, you can include those commands with Quartus II executables in a single script.

The Quartus II command-line executables include options for common global project settings and operations, but you must use a Tcl script or the Quartus II GUI to set up a new project and apply individual constraints, such as pin location assignments and timing requirements. Command-line executables are very useful for working with existing projects, for making common global settings, and for performing common operations. For more flexibility in a flow, use a Tcl script, which makes it easier to pass data between different stages of the design flow and have more control during the flow.

For example, a UNIX shell script could run other synthesis software, then place-and-route the design in the Quartus II software, then generate output netlists for other simulation software. This script shows a script that synthesizes a design with the Synopsys Synplify software, simulates the design using the Mentor Graphics ModelSim® software, and then compiles the design targeting a Cyclone V device.

```
#!/bin/sh
# Run synthesis first.
# This example assumes you use Synplify software
synplify -batch synthesize.tcl
# If your Quartus II project exists already, you can just
# recompile the design.
# You can also use the script described in a later example to
# create a new project from scratch
quartus_sh --flow compile myproject
# Use the quartus_sta executable to do fast and slow-model
# timing analysis
quartus_sta myproject --model=slow
quartus_sta myproject --model=fast
# Use the quartus_eda executable to write out a gate-level
# Verilog simulation netlist for ModelSim
quartus_eda my_project --simulation --tool=modelsim --format=verilog
# Perform the simulation with the ModelSim software
vlib cycloneV_ver
vlog -work cycloneV_ver /opt/quartusii/eda/sim_lib/cycloneV_atoms.v
vlib work
vlog -work work my_project.vo
vsim -L cycloneV_ver -t 1ps work.my_project
```

Related Information

- Tcl Scripting
- About Quartus II Tcl Scripting

Makefile Implementation

You can use the Quartus II command-line executables in conjunction with the `make` utility to automatically update files when other files they depend on change. The file dependencies and commands used to update files are specified in a text file called a makefile.

To facilitate easier development of efficient makefiles, the following "smart action" scripting command is provided with the Quartus II software:

```
quartus_sh --determine_smart_action
```

Because assignments for a Quartus II project are stored in the **.qsf**, including it in every rule results in unnecessary processing steps. For example, updating a setting related to programming file generation, which requires re-running only `quartus_asm`, modifies the **.qsf**, requiring a complete recompilation if the **.qsf** is included in every rule.

The smart action command determines the earliest command-line executable in the compilation flow that must be run based on the current **.qsf**, and generates a change file corresponding to that executable. For example, if `quartus_map` must be re-run, the smart action command creates or updates a file named **map.chg**. Thus, rather than including the **.qsf** in each makefile rule, include only the appropriate change file.

The sample makefile uses change files and the smart action command. You can copy and modify it for your own use. A copy of this example is included in the help for the makefile option, which is available by typing:

```
quartus_sh --help=makefiles
```

```
########################################################################
# Project Configuration:
#
# Specify the name of the design (project), the Quartus II Settings
# File (.qsf), and the list of source files used.
########################################################################
PROJECT = chiptrip
SOURCE_FILES = auto_max.v chiptrip.v speed_ch.v tick_cnt.v time_cnt.v
ASSIGNMENT_FILES = chiptrip.qpf chiptrip.qsf
########################################################################
# Main Targets
#
# all: build everything
# clean: remove output files and database
########################################################################
all: smart.log $(PROJECT).asm.rpt $(PROJECT).sta.rpt
clean:
 rm -rf *.rpt *.chg smart.log *.htm *.eqn *.pin *.sof *.pof db
map: smart.log $(PROJECT).map.rpt
fit: smart.log $(PROJECT).fit.rpt
asm: smart.log $(PROJECT).asm.rpt
sta: smart.log $(PROJECT).sta.rpt
smart: smart.log
########################################################################
# Executable Configuration
########################################################################
MAP_ARGS = --family=Stratix
```

```
FIT_ARGS = --part=EP1S20F484C6
ASM_ARGS =
STA_ARGS =
#####################################################################
# Target implementations
#####################################################################
STAMP = echo done >
$(PROJECT).map.rpt: map.chg $(SOURCE_FILES)
 quartus_map $(MAP_ARGS) $(PROJECT)
 $(STAMP) fit.chg
#####################################################################
# Project initialization
#####################################################################
$(ASSIGNMENT_FILES):
 quartus_sh --prepare $(PROJECT)
map.chg:
 $(STAMP) map.chg
fit.chg:
 $(STAMP) fit.chg
sta.chg:
 $(STAMP) sta.chg
asm.chg:
 $(STAMP) asm.chg
```

A Tcl script is provided with the Quartus II software to create or modify files that are specified as dependencies in the make rules, assisting you in makefile development. Complete information about this Tcl script and how to integrate it with makefiles is available by running the following command:

```
quartus_sh --help=determine_smart_action
```

Common Scripting Examples

You can create scripts including command line executable to control common Quartus II processes.

Create a Project and Apply Constraints

The command-line executables include options for common global project settings and commands. To apply constraints such as pin locations and timing assignments, run a Tcl script with the constraints in it. You can write a Tcl constraint file yourself, or generate one for an existing project.

From the Project menu, click **Generate Tcl File for Project**.

The example creates a project with a Tcl script and applies project constraints using the tutorial design files in the <*Quartus II installation directory*>**/qdesigns/fir_filter/** directory.

```
project_new filtref -overwrite
# Assign family, device, and top-level file
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C12F256C6
set_global_assignment -name BDF_FILE filtref.bdf
# Assign pins
set_location_assignment -to clk Pin_28
set_location_assignment -to clkx2 Pin_29
set_location_assignment -to d[0] Pin_139
set_location_assignment -to d[1] Pin_140
# Other assignments could follow
project_close
```

Save the script in a file called **setup_proj.tcl** and type the commands illustrated in the eample at a command prompt to create the design, apply constraints, compile the design, and perform fast-corner and slow-corner timing analysis. Timing analysis results are saved in two files, **filtref_sta_1.rpt** and **filtref_sta_2.rpt**.

```
quartus_sh -t setup_proj.tcl
quartus_map filtref
quartus_fit filtref
quartus_asm filtref
quartus_sta filtref --model=fast --export_settings=off
mv filtref_sta.rpt filtref_sta_1.rpt
quartus_sta filtref --export_settings=off
mv filtref_sta.rpt filtref_sta_2.rpt
```

Type the following commands to create the design, apply constraints, and compile the design, without performing timing analysis:

```
quartus_sh -t setup_proj.tcl
quartus_sh --flow compile filtref
```

The `quartus_sh --flow compile` command performs a full compilation, and is equivalent to clicking the **Start Compilation** button in the toolbar.

Check Design File Syntax

The UNIX shell script example shown in the example assumes that the Quartus II software **fir_filter** tutorial project exists in the current directory. You can find the **fir_filter** project in the <Quartus II directory>**/qdesigns/fir_filter** directory unless the Quartus II software tutorial files are not installed.

The `--analyze_file` option causes the **quartus_map** executable to perform a syntax check on each file. The script checks the exit code of the **quartus_map** executable to determine whether there is an error during the syntax check. Files with syntax errors are added to the `FILES_WITH_ERRORS` variable, and when all files are checked, the script prints a message indicating syntax errors.

When options are not specified, the executable uses the project database values. If not specified in the project database, the executable uses the Quartus II software default values. For example, the **fir_filter** project is set to target the Cyclone device family, so it is not necessary to specify the `--family` option.

```
#!/bin/sh
FILES_WITH_ERRORS=""
# Iterate over each file with a .bdf or .v extension
for filename in `ls *.bdf *.v`
do
# Perform a syntax check on the specified file
    quartus_map fir_filter --analyze_file=$filename
  # If the exit code is non-zero, the file has a syntax error
    if [ $? -ne 0 ]
    then
       FILES_WITH_ERRORS="$FILES_WITH_ERRORS $filename"
    fi
done
if [ -z "$FILES_WITH_ERRORS" ]
then
    echo "All files passed the syntax check"
    exit 0
else
    echo "There were syntax errors in the following file(s)"
    echo $FILES_WITH_ERRORS
    exit 1
fi
```

Create a Project and Synthesize a Netlist Using Netlist Optimizations

This example creates a new Quartus II project with a file **top.edf** as the top-level entity. The `--enable_register_retiming=on` and `--enable_wysiwyg_resynthesis=on` options cause **quartus_map** to optimize the design using gate-level register retiming and technology remapping.

The `--part` option causes **quartus_map** to target an EP3C10F256C8 device. To create the project and synthesize it using the netlist optimizations described above, type the command shown in this example at a command prompt.

```
quartus_map top --source=top.edf --enable_register_retiming=on
 --enable_wysiwyg_resynthesis=on --part=EP3C10F256C8
```

Archive and Restore Projects

You can archive or restore a Quartus II Archive File (**.qar**) with a single command. This makes it easy to take snapshots of projects when you use batch files or shell scripts for compilation and project management.

Use the `--archive` or `--restore` options for `quartus_sh` as appropriate. Type the command shown in the example at a command prompt to archive your project.

```
quartus_sh --archive <project name>
```

The archive file is automatically named <*project name*>**.qar.** If you want to use a different name, type the command with the `-output` option as shown in example the example.

```
quartus_sh --archive <project name> -output <filename>
```

To restore a project archive, type the command shown in the example at a command prompt.

```
quartus_sh --restore <archive name>
```

The command restores the project archive to the current directory and overwrites existing files.

Related Information

Managing Quartus II Projects

Perform I/O Assignment Analysis

You can perform I/O assignment analysis with a single command. I/O assignment analysis checks pin assignments to ensure they do not violate board layout guidelines. I/O assignment analysis does not require a complete place and route, so it can quickly verify that your pin assignments are correct.

```
quartus_fit --check_ios <project name> --rev=<revision name>
```

Update Memory Contents Without Recompiling

You can use two commands to update the contents of memory blocks in your design without recompiling. Use the `quartus_cdb` executable with the `--update_mif` option to update memory contents from **.mif** or **.hexout** files. Then, rerun the assembler with the `quartus_asm` executable to regenerate the **.sof**, **.pof**, and any other programming files.

```
quartus_cdb --update_mif <project name> [--rev=<revision name>]
quartus_asm <project name> [--rev=<revision name>]
```

The example shows the commands for a DOS batch file for this example. With a DOS batch file, you can specify the project name and the revision name once for both commands. To create the DOS batch file, paste the following lines into a file called **update_memory.bat.**

```
quartus_cdb --update_mif %1 --rev=%2
quartus_asm %1 --rev=%2
```

To run the batch file, type the following command at a command prompt:

```
update_memory.bat <project name> <revision name>
```

Create a Compressed Configuration File

You can create a compressed configuration file in two ways. The first way is to run `quartus_cpf` with an option file that turns on compression.

To create an option file that turns on compression, type the following command at a command prompt:

```
quartus_cpf -w <filename>.opt
```

This interactive command guides you through some questions, then creates an option file based on your answers. Use `--option` to cause **quartus_cpf** to use the option file. For example, the following command creates a compressed **.pof** that targets an EPCS64 device:

```
quartus_cpf --convert --option=<filename>.opt --device=EPCS64 <file>.sof <file>.pof
```

Alternatively, you can use the Convert Programming Files utility in the Quartus II software GUI to create a Conversion Setup File (**.cof**). Configure any options you want, including compression, then save the conversion setup. Use the following command to run the conversion setup you specified.

```
quartus_cpf --convert <file>.cof
```

Fit a Design as Quickly as Possible

This example assumes that a project called **top** exists in the current directory, and that the name of the top-level entity is **top**. The `--effort=fast` option causes the **quartus_fit** to use the fast fit algorithm to increase compilation speed, possibly at the expense of reduced f_{MAX} performance. The `--one_fit_attempt=on` option restricts the Fitter to only one fitting attempt for the design.

To attempt to fit the project called **top** as quickly as possible, type the command shown at a command prompt.

```
quartus_fit top --effort=fast --one_fit_attempt=on
```

Fit a Design Using Multiple Seeds

This shell script example assumes that the Quartus II software tutorial project called **fir_filter** exists in the current directory (defined in the file **fir_filter.qpf**). If the tutorial files are installed on your system, this project exists in the <Quartus II directory>/qdesigns<quartus_version_number> /**fir_filter** directory.

Because the top-level entity in the project does not have the same name as the project, you must specify the revision name for the top-level entity with the `--rev` option. The `--seed` option specifies the seeds to use for fitting.

A seed is a parameter that affects the random initial placement of the Quartus II Fitter. Varying the seed can result in better performance for some designs.

After each fitting attempt, the script creates new directories for the results of each fitting attempt and copies the complete project to the new directory so that the results are available for viewing and debugging after the script has completed.

```
#!/bin/sh
ERROR_SEEDS=""
quartus_map fir_filter --rev=filtref
# Iterate over a number of seeds
for seed in 1 2 3 4 5
do
echo "Starting fit with seed=$seed"
# Perform a fitting attempt with the specified seed
quartus_fit fir_filter --seed=$seed --rev=filtref
# If the exit-code is non-zero, the fitting attempt was
# successful, so copy the project to a new directory
if [ $? -eq 0 ]
then
     mkdir ../fir_filter-seed_$seed
     mkdir ../fir_filter-seed_$seed/db
     cp * ../fir_filter-seed_$seed
     cp db/* ../fir_filter-seed_$seed/db
else
     ERROR_SEEDS="$ERROR_SEEDS $seed"
fi
done
if [ -z "$ERROR_SEEDS" ]
then
echo "Seed sweeping was successful"
exit 0
else
echo "There were errors with the following seed(s)"
echo $ERROR_SEEDS
exit 1
fi
```

Tip: Use the Design Space Explorer (DSE) included with the Quartus II software script (by typing `quartus_sh --dse` at a command prompt) to improve design performance by performing automated seed sweeping.

Related Information

TimeQuest Timing Analyzer Quick Start Tutorial

For more information about the DSE, or type quartus_sh --help=dse at a command prompt.

The QFlow Script

A Tcl/Tk-based graphical interface called QFlow is included with the command-line executables. You can use the QFlow interface to open projects, launch some of the command-line executables, view report files, and make some global project assignments.

The QFlow interface can run the following command-line executables:

- `quartus_map` (Analysis and Synthesis)
- `quartus_fit` (Fitter)
- `quartus_sta` (TimeQuest timing analyzer)
- `quartus_asm` (Assembler)

- `quartus_eda` (EDA Netlist Writer)

  To view floorplans or perform other GUI-intensive tasks, launch the Quartus II software.

  Start QFlow by typing the following command at a command prompt:

  ```
  quartus_sh -g
  ```

  Tip: The QFlow script is located in the <Quartus II directory>**/common/tcl/apps/qflow/** directory.

Document Revision History

Table 2-3: Document Revision History

Date	Version	Changes
June 2014	14.0.0	Updated formatting.
November 2013	13.1.0	Removed information about -silnet `qmegawiz` command
June 2012	12.0.0	Removed survey link.
November 2011	11.0.1	Template update.
May 2011	11.0.0	Corrected quartus_qpf example usage. Updated examples.
December 2010	10.1.0	Template update. Added section on using a script to regenerate megafunction variations. Removed references to the Classic Timing Analyzer (quartus_tan). Removed Qflow illustration.
July 2010	10.0.0	Updated script examples to use quartus_sta instead of quartus_tan, and other minor updates throughout document.
November 2009	9.1.0	Updated Table 2–1 to add quartus_jli and quartus_jbcc executables and descriptions, and other minor updates throughout document.
March 2009	9.0.0	No change to content.

Date	Version	Changes
November 2008	8.1.0	Added the following sections: • "The MegaWizard Plug-In Manager" on page 2–11 "Command-Line Support" on page 2–12 "Module and Wizard Names" on page 2–13 "Ports and Parameters" on page 2–14 "Invalid Configurations" on page 2–15 "Strategies to Determine Port and Parameter Values" on page 2–15 "Optional Files" on page 2–15 "Parameter File" on page 2–16 "Working Directory" on page 2–17 "Variation File Name" on page 2–17 • "Create a Compressed Configuration File" on page 2–21 • Updated "Option Precedence" on page 2–5 to clarify how to control precedence • Corrected Example 2–5 on page 2–8 • Changed Example 2–1, Example 2–2, Example 2–4, and Example 2–7 to use the EP1C12F256C6 device • Minor editorial updates • Updated entire chapter using 8½" × 11" chapter template
May 2008	8.0.0	• Updated "Referenced Documents" on page 2–20. • Updated references in document.

Related Information

- **http://www.altera.com/literature/lit-qts_archive.jsp**