QSF Assignment Descriptions

Version 1.0

January 6, 2004

By

Altera Corporation

Table of Contents

1. Overview

The Altera® QSF (Quartus® Settings File) file is used to store key information about compiler settings, simulator settings and project settings for a Quartus® II project. This information includes any location assignments given to blocks used in the design. As a result, the QSF file represents a simple way for 3rd party developers to use their own placements for a design in Quartus II.

2. Executive Summary

This document describes the file format of a QSF file and, in particular, the format of location assignments. It has the following purposes:

- Describe the QSF file format for Quartus II 4.0
- Explain how users can use their own placement in a Quartus II flow via the QSF file
- Explain how to compare results of a 3rd party placement to Quartus II placement

This document makes the following assumptions:

- The reader understands how to produce a valid placement for an Altera device given an atom netlist
- The reader has read the WYSIWYG guide for relevant devices (Stratix™) and is familiar with the basic structure of atoms for relevant devices [1] [2] [3]

This file does not:

- Give detailed architecture information like block information and layout (please consult [4] for information on how to get detailed block information)
- Give detailed timing information (please consult [5] for information on how to get detailed timing information)

After reading this file and referenced material you should be able to do the following:

- Import placement results obtained with a 3rd party tool into the Quartus II flow
- Compare results of 3rd party placement tools with Quartus II placement

3. QSF File Format

The Quartus II QSF file is organized as a series of Tcl commands that assign each variable the appropriate value. The file is automatically generated with a set of valid default settings when a project is created in the Quartus II 4.0 design software. If any settings need to be changed they can either be changed in the QSF file, using Tcl, or through the GUI. Most of the settings are accessed via the Assignments menu. Detailed explanations of how to use the GUI to change settings can be found by using Quartus II design software on-line Help (Note that if a change is

made to a setting through the GUI, the QSF file is automatically modified to reflect this change, and vice-versa).

Please note that the QSF file replaces the CSF (Compiler Settings File), which stored location assignments in previous versions of the Quartus II design software. If you have a project from an older version of the Quartus II software and wish to import location assignments made in a CSF file, simply open the project with the new Quartus II 4.0 design file and a QSF file that preserves all CSF settings (including location assignments) will automatically be generated. Any further changes or additions should be made to the QSF file as described in this document because the CSF file is only used for creation of the QSF file.

For more details on the contents of the QSF file, please see the Quartus II design software on-line Help.

4. Assignments In the QSF File

Using the QSF file it is possible to use placement from a 3rd party tool in a Quartus II flow. Placement should produce valid assignments for netlist nodes to locations on the target chip (for assistance in specifying chip locations please see the *Altera XML Architecture Description File Detailed Design* document).

As mentioned previously, assignments can be made by either directly editing the QSF file or by using Tcl commands. Since the Tcl interface should not change from one distribution of Quartus II design software to the next, using it will help make your tools more forward compatible. For more information on Tcl scripting in the Quartus II flow, please see [6] and [7].

The Tcl commands necessary to make QSF assignments can be found in the Quartus II Project Tcl package. To load this package the following command must be executed:

```
package require ::quartus::project
```

After loading the package, assignments can be added to the QSF file through Tcl commands. To execute a series of Tcl instructions a script containing all commands should be created (remember to load the package in the script!) and run with the Quartus Tcl interpreter by executing the following command line directive:

```
quartus_sh –t <file_name>
```

Alternatively, to execute the script through the Quartus II design software GUI, one would open the view menu and choose *Utility Windows → Tcl Console* to bring up the console window. Next, in the console window enter the following command:

```
source <file_name>
```

For this command to work properly, your working directory must either contain the Tcl source file or you must specify the path as part of the file name (To check which directory you are currently in type “pwd” from the Tcl Console window).

The Tcl commands pertaining to location assignments that can be placed directly in the QSF file or in a separate script use the formats specified in the following subsections (for more information on the blocks described please see the WYSIWYG guide).

4.1 Logic Array Blocks (LABs)

To assign a node to a LAB, the following command can be used:

```
set_location_assignment LAB_X<x_loc>_Y<y_loc> -to <node_name>
```

The parameters for this statement are summarized in Table 1.

Table 1: LAB Assignment Parameters

Parameter	Description
<node_name>	String that represents the name of the node being placed. This should match the name of the node in the atom netlist (vqm file) exactly.
<x_loc>	Integer value of X co-ordinate of the destination LAB.
<y_loc>	Integer value of Y co-ordinate of the destination LAB.

e.g. The statement: `set_location_assignment LAB_X19_Y3 -to input_a` assigns the node 'input_a' to the LAB located at (19,3).

4.2 Logic Cells (LCs)

It is possible to place nodes in specific logic cells within a LAB. This type of placement is permissible but is recommended only if it is essential. Locking locations down to individual logic cells is very restrictive and may reduce the performance of the Quartus router. This routing will result in proper LC placement, and unless one can be confident that the specified LC placement is near optimal and totally valid (which is difficult to achieve), it is recommended that only LAB level assignments be used. However, if a logic cell assignment is necessary, it can be accomplished by adding a statement of the following form to the QSF file or a Tcl script:

```
set_location_assignment LC_X<x_loc>_Y<y_loc>_N<subloc> -to <node_name>
```

The parameters for this statement are summarized in Table 2.

Table 2: LC Assignment Parameters

Parameter	Description
<node_name>	See Table 1
<x_loc>	See Table 1
<y_loc>	See Table 1
<subloc>	Integer representing the sublocation of the destination LC in the chosen LAB. This value must be in a valid range, which is 0-9 inclusive for Stratix

	/ Stratix GX and Cyclone devices.

e.g. The statement: `set_location_assignment LC_X19_Y3_N2 -to input_a` assigns the node 'input_a' to the LC located at sub location two in the LAB at (19,3).

4.3 Pins

It is possible to assign inputs and outputs of a design to specific pins on the destination device. This can be accomplished by adding a statement of the following form to the QSF file or a Tcl script:

```
set_location_assignment Pin_<pin_name> -to <node_name>
```

The parameters for this statement are summarized in Table 3.

Table 3: LC Assignment Parameters

Parameter	Description
<io_name>	String that represents the name of the input or output node being placed. This should match the name in the atom netlist (vqm file) exactly
<pin_name>	String that represents the name of the pin to which the assignment is being made. The pin names can be found in [4] and are typically a letter followed by a number with no spaces in between (i.e. A1).

e.g. The statement: `set_location_assignment Pin_B3 -to input_a` assigns the node 'input_a' to the pin named B3 on the device.

Alternatively, you can specify which IO cell in the FPGA a given node should be implemented in, without specifying the pin name. This assignment has the form:

```
set_location_assignment IOC_X<x_loc>_Y<y_loc>_N<subloc> -to <node_name>
```

Table 4: IO Cell Assignment Parameters

Parameter	Description
<node_name>	See Table 1
<x_loc>	See Table 1
<y_loc>	See Table 1
<subloc>	Specifies which sub location (corresponds to a particular pin in the IO cell) to use.

e.g. The statement: `set_location_assignment IOC_X52_Y31_N5 -to input_a` assigns the node 'input_a' to sub location 5 of the IO cell located at (52,31).

4.4 MAC Blocks (DSP Blocks)

One of the more advanced features of Stratix and Stratix™ GX chips is the presence of multiply accumulate (MAC) blocks on the chip. Placement to a particular MAC (also referred to as DSP blocks, however the XML architecture descriptions uses the term MAC) is a feature that will be used in more sophisticated designs. Nodes can be assigned to specific MAC blocks with statements that follow the following format:

```
set_location_assignment DSP_X<x_loc>_Y<y_loc> -to <node_name>
```

The parameters for this statement are summarized in Table 5.

Table 5: MAC Block Parameters

Parameter	Description
<node_name>	String that represents the name of the node being placed. This should match the name of the node in the atom netlist (vqm file) exactly.
<x_loc>	Integer valued X position of the destination MAC's lower left corner.
<y_loc>	Integer valued Y position of the destination MAC's lower left corner.

e.g. The statement: `set_location_assignment DSP_X42_Y9 -to mult_a` assigns the node 'mult_a' to the MAC located at (42,9).

4.5 RAM Blocks

Another advanced feature of Stratix, Stratix GX and Cyclone chips is the presence of RAM blocks on the devices. These blocks come in three different sizes: M512 blocks that contain 512 bits of memory, M4K blocks that consist of 4096 bits of memory and M-RAM (mega RAM) blocks that consist of 512K bits of memory. All three types of RAM are present on Stratix and Stratix GX chips, while Cyclone devices have only the M4K blocks. Assignments to various blocks follow the format:

```
set_location_assignment <ram_type>_X<x_loc>_Y<y_loc> -to <node_name>
```

The parameters for this statement are summarized in Table 6.

Table 6: MAC Block Parameters

Parameter	Description
<node_name>	String that represents the name of the node being placed. This should match the name of the node in the atom netlist (vqm file) exactly.
<ram_type>	String that represents the type of RAM block to which the node is being

	assigned. This variable *must* be one of the following: M512, M4K or MRAM
<*x_loc*>	Integer valued X position of the destination RAM's lower left corner.
<*y_loc*>	Integer valued Y position of the destination RAM's lower left corner.

e.g. The statement: `set_location_assignment MRAM_X20_Y7 -to data_a` assigns the node 'data_a' to the M-RAM block located at (20,7) on the target device.

4.6 Entity Level Assignments

The above sections specify ways to assign nodes to specific locations, but it is also possible to specify the location of an entity without explicitly assigning each of its nodes. Entities can be assigned to blocks in the exact same way that nodes are assigned to blocks, except the node name is replaced by the name of the entity. The assignment forces all of the entity's nodes to be placed in the destination block (i.e. LAB, MAC or RAM).

In order for an entity level assignment to be valid, the destination block must contain all of the resources necessary to place the nodes of the entity. For instance, assigning an entity that contains pins to a LAB will result in an error, because the pins cannot be placed in the LAB.

4.7 Custom Regions

Entity level assignments to a block are restricted by the fact that the entity must fit completely within the specified block. Entities usually require several blocks, possibly of different types, and in such cases the assignments of section 5.6 are not sufficient. The concept of a Custom Region, however, allows entity assignments that span a set of blocks.

A Custom Region is a rectangular section of the chip and is specified by the bottom left and top right corners of the region. Both entities and nodes may be placed in the custom region, but for the assignment to be valid all necessary resources must exist in the region.

Custom Regions are not exclusive to entity assignments. Individual nodes may also be placed in a custom region. Assignments to custom regions follow the format below:

```
set_location_assignment CUSTOM_REGION_X<x1>_Y<y1>_X<x2>_Y<y2> –to <name>
```

The parameters for this statement are summarized in Table 7.

Table 7: Custom Region Parameters

Parameter	Description
<*name*>	String that represents the name of the node or entity being placed. This should match the name of the node or entity in the atom netlist (vqm file) exactly.
<*x1*>	Integer valued X position of the region's bottom left corner.
<*y1*>	Integer valued Y position of the region's bottom left corner.

<x2>	Integer valued X position of the region's top right corner.
<y2>	Integer valued Y position of the region's top right corner.

4.8 QSF File Example

The following is an example of a QSF file created for a simple design called "quip_des". The design consists of two inputs (in_1 and in_2) which are fed through an 'and' gate (and_gate) that drives the output (out_1). Several default options have been omitted from the example, so a typical QSF file will have more settings (note that lines beginning with a '#' are comments).

```
# Project-Wide Assignments
# ========================
set_global_assignment -name BDF_FILE quip_des.bdf

# Pin & Location Assignments
# ==========================
set_global_assignment -name RESERVE_PIN "AS INPUT TRI-STATED"
set_location_assignment LAB_X52_Y30 -to and_gate
set_location_assignment PIN_B3 -to in_1
set_location_assignment PIN_F4 -to in_2
set_location_assignment PIN_B2 -to out_1


# Timing Assignments
# ==================
set_global_assignment -name
INCLUDE_EXTERNAL_PIN_DELAYS_IN_FMAX_CALCULATIONS OFF

# Analysis & Synthesis Assignments
# ================================
set_global_assignment -name STRATIX_OPTIMIZATION_TECHNIQUE SPEED
set_global_assignment -name FAMILY Stratix
set_global_assignment -name TOP_LEVEL_ENTITY quip_des

# Fitter Assignments
# ==================
set_global_assignment -name SEED 11
set_global_assignment -name AUTO_RESTART_CONFIGURATION OFF
set_global_assignment -name DEVICE EP1S10F484C5

# Timing Analysis Assignments
# ===========================
set_global_assignment -name MAX_SCC_SIZE 50
set_global_assignment -name RUN_ALL_TIMING_ANALYSES OFF

# Simulator Assignments
# =====================
set_global_assignment -name GLITCH_INTERVAL 1NS

# Design Assistant Assignments
```

```
# ============================
set_global_assignment -name HCPY_ALOAD_SIGNALS OFF
set_global_assignment -name HCPY_VREF_PINS OFF
set_global_assignment -name HCPY_CAT OFF

# SignalTap II Assignments
# ========================
set_global_assignment -name HUB_INSTANCE_NAME SLD_HUB_INST
set_global_assignment -name HUB_ENTITY_NAME SLD_HUBCOMPILER_SETTINGS
```

Figure 1: QSF File Example

4.9 Tcl Example

To create a project with the same location assignments as the QSF file in Figure 1, one would create a file, call it “setup_project.tcl”, with the following commands:

```
package require ::quartus::project
# To always replace any existing project
project_new "quip_des" –overwrite

set_global_assignment -name DEVICE EP1S10F484C5
set_location_assignment LAB_X52_Y30 -to and_gate
set_location_assignment PIN_B3 -to in_1
set_location_assignment PIN_F4 -to in_2
set_location_assignment PIN_B2 -to out_1

project_close
```

Next, to compile this design, from the command line one could invoke the following commands:

```
quartus_sh -t setup_project.tcl
quartus_sh --flow compile quip_des
```

This will compile a project with the desired assignments, and makes no assumptions about the format of the QSF file.

5. Obtaining The Results

After the assignments are added to the QSF file, the remainder of the design compilation can be completed with the Quartus II design software. Note that not all nodes in the netlist must be placed. The Quartus tools will automatically place nodes that are not assigned to specific locations.

5.1 Suggested Flow

The suggested series of steps required to obtain results with a 3rd party placement is as follows:

1. Create a design using a hardware description language or schematics.
2. Synthesize the design in the Quartus II software or a 3rd party synthesis tool and produce an atom netlist.
3. Use the 3rd party placement tools in conjunction with Altera's XML architecture descriptions and Altera's XML point-to-point delay specifications to place the design.
4. Convert the placements to assignment statements in Tcl or by directly altering the project's QSF file.
5. Load the project in the Quartus II software and run the Fitter,Timing Analyzer and Assembler. This can be done from the command line by executing quartus_fit, quartus_tan and quartus_asm respectively. Alternatively, one could use the Quartus II GUI to run these flows.
6. The timing analysis and compilation report should contain all information necessary to provide useful feedback.

Following these steps will allow the user to benchmark their placement tools.

6. Comparing The Results

Often the goal of a design effort is to compare the results of a newly created placement tool to those of an existing module. As a result, it is tempting to simply run the entire Quartus flow on the design and compare the results to those obtained by following the method set out in section 6. However, doing so would not provide a true comparison of the tools. The Quartus placement algorithms make use of some highly sophisticated features that may be beyond the scope of a 3rd party developer. As a result, the Quartus flow has inherent advantages. If one wishes to perform a fair comparison, the following advanced options should be disabled.

6.1 Carry Chains

Carry chains are an advanced feature of the Quartus flow that offers many performance benefits. However, their implementation requires a level of complexity that extends well beyond the scope of many non-commercial placers. To disable carry chains, perform the following steps in the Quartus II GUI:

- Select Assignments Menu
- Select Settings
- Select Analysis & Synthesis Settings
- Click the "More Settings" button on the right hand portion of the window.
- Select Ignore CARRY Buffers from the list in the new window.

- Make the Setting equal to On using the drop down menu in the upper portion of the window

Alternatively, one can disable carry chains with the following Tcl command:

```
set_global_assignment -name "IGNORE_CARRY_BUFFERS" "On"
```

6.2 DSP Blocks

By default, the Quartus II design software will attempt to optimize placement by using DSP blocks whenever possible. If the placement being compared to the Quartus solution chooses to not use DSP blocks, disabling the use of DSP blocks will allow for a more accurate comparison. To limit the use of DSP blocks in Quartus, perform the following steps in the Quartus II GUI:

- Select Assignments Menu
- Select Settings
- Select Analysis & Synthesis Settings
- Select DSP Block Balancing (the top drop down menu on the right hand portion of the window)
- Set DSP Block Balancing to Logic Elements

Alternatively one can disable DSP blocks with the following Tcl command:

```
set_global_assignment -name "DSP_BLOCK_BALANCING" "Logic Elements"
```

7. Referenced Documents

1. "WYSIWYG Device Primitives User Guide For Stratix."
2. "Stratix RAM WYSIWYG User Guide."
3. "Stratix MAC WYSIWYG Description."
4. "Altera XML Architecture Description File Detailed Design"
5. "Altera XML Point To Point Delay File Detailed Design."
6. "Scripting with Tcl in the Quartus II Software.", Altera Application Note #312, available at: http://www.altera.com/literature/an/an312.pdf
7. "Command Line Scripting in the Quartus II Software.", Altera Application Note #309, available at: http://www.altera.com/literature/an/an309.pdf