Создание проектов и назначений

Преимуществом Tcl скриптирования API – это возможность запросто создать скрипт, который сделает все назначения для имеющегося проекта. Вы можете использовать скрипт в любое время для восстановления настроек вашего проекта в известное состояние. В меню **Project** кликните **Generate Tcl File for Project** для автоматического генерирования **.tcl** файла со всеми вашими назначениями. Вы можете использовать этот файл для воссоздания проекта. Этот файл является хорошей отправной точкой изучения команд управления проектом и команд назначений.

Обратитесь к секции "Режим интерактивной оболочки" на стр. 3-6 за информацией об использовании скрипта. Информация о скриптах для всех настроек и назначений проекта Quartus II находится в *Справочном руководстве QSF*.

В примере 3-2 показано, как создавать проект, делать назначения и компилировать проект. В нём используются файлы учебного проекта **fir_filter**, расположенный в директории установки **qdesigns**. Запустите этот скрипт в директории **fir_filter** исполняемым компонентом **quartus_sh**.

Example 3–2. Create and Compile a Project

```
load_package flow

# Create the project and overwrite any settings
# files that exist
project_new fir_filter -revision filtref -overwrite
# Set the device, the name of the top-level BDF,
# and the name of the top level entity
set_global_assignment -name FAMILY Cyclone
set_global_assignment -name DEVICE EP1C6F256C6
set_global_assignment -name BDF_FILE filtref.bdf
set_global_assignment -name TOP_LEVEL_ENTITY filtref
# Add other pin assignments here
set_location_assignment -to clk Pin_G1
# compile the project
execute_flow -compile
project_close
```

Назначения, созданные или изменённые во время открытия проекта, не фиксируются в файле настроек Quartus II (**.qsf**) пока вы явно не вызовете export_assignments или project_close (без -dont_export_assignments). В этих случаях при запуске execute_flow, программа Quartus II автоматически фиксирует изменения.

Проекты в чипах HardCopy

За информацией об использовании скриптированного процесса проектирования для проектов HardCopy II, обратитесь к главе "Скиптированное проектирование для чипов HardCopy II" в настольной книге HardCopy.

Отдельная глава в настольной книге HardCopy называется "Временные ограничения для чипов HardCopy II" также содержит информация о скриптированном проектировании для чипов HardCopy II с акцентом на временные ограничения.

Компиляция проектов

Вы можете запустить исполняемые компоненты Tcl скриптов в командной строке Quartus II. Используйте включённый пакет **flow** для запуска различных процессов компиляции Quartus II, или запускайте каждый исполняемый компонент отдельно.

Пакет flow

Пакет **flow** содержит две команды для запуска исполняемых компонентов в командной строке Quartus II, каждую в отдельности или вместе в стандартной последовательности компиляции. Команда **execute_module** позволяет вам запустить отдельный исполняемый компонент в командной строке Quartus II. Команда **execute_flow** позволяет вам запустить некоторые или все модули в широко применяемых комбинациях. Используйте пакет **flow** вместо использования вызова системы для запуска исполняемых компонентов скриптов Quartus II.

Другой способ – это запускать исполняемые компоненты Quartus II в среде Tcl с помощью Tcl команды **qexec**, для Quartus II версии – это команда Tcl **exec**. Например, для запуска Анализа и Синтеза Quartus II для имеющегося проекта, используйте:

qexec "quartus_map <*project_name*>"

Когда вы используете команду **qexec** для компиляции проекта, назначения, сделанные в Tcl скрипте (или в оболочке Tcl) не экспортируются в базу данных проекта и в файл настроек перед началом компиляции. Используйте команду **export_assignments** или компилируйте проект с командами из пакета **flow**, чтобы добиться экспорта назначений в базу данных проекта и файл настроек.

Вы можете запускать исполняемые компоненты прямо из оболочки Tcl, используя тот же синтаксис, что и в системной командной строке. Например, для запуска Quartus II technology mapper для имеющегося проекта, используйте:

quartus_map <*project_name*>

Компиляция всех версий

Вы можете использовать простой Tcl скрипт для компиляции всех версий вашего проекта. Сохраните скрипт, показанный в примере 3-3, в файл с именем **compile_revisions.tcl** и введите следующее для его запуска:

quartus_sh -t compile_revisions.tcl <*project name*>

Example 3–3. Compile All Revisions

```
load_package flow
project_open [lindex $quartus(args) 0]
set original_revision [get_current_revision]
foreach revision [get_project_revisions] {
    set_current_revision $revision
    execute flow -compile
}
set_current_revision $original_revision
project_close
```

Отчёты

Иногда необходимо изъять информацию из отчёта о компиляции для анализа результатов. Quartus II Tcl API предлагает простой способ доступа к данным отчёта такой, что вам не понадобится писать скрипты для анализа текстовых файлов отчётов.

Если вы знаете, к какой ячейке или ячейкам вы хотите получить доступ, то используйте команду **get_report_panel_data** и задайте имена столбцов и строк (или координаты x и y), а также соответствующий заголовок отчёта. Вы можете многократно использовать поиск данных по панели отчёта. Чтобы сделать это, используйте цикл чтения одной строки отчёта за один раз с помощью команды **get_report_panel_row**.

Заголовок столбца в панели отчётов находится в нулевой строке. Если вы используете цикл чтения одной строки отчёта за один раз, вы можете начинать с первой строки, чтобы пропустить строку-заголовок. Команда **get_number_of_rows** возвращает количество строк в панели отчётов, включая строку заголовка столбцов. Поскольку количество строк включает в себя строку заголовка столбцов, ваш цикл может продолжаться до индекса, который меньше чем количество строк, как показано в примере 3-5.

Панели отчётов имеют иерархическую структуру, и каждый уровень иерархии выделяется строкой "||" в названии панели. Например, имя панели отчёта настроек компоновщика - **Fitter||Fitter Settings** – поскольку он находится в папке Fitter. Панели с наивысшим уровнем иерархии не используют строку "||". Например, панель отчёта настроек процесса называется **Flow Settings**.

В коде примера 3-4 выводится список всех имён панелей отчётов вашего проекта. Вы можете запускать этот код с любым исполняемым компонентом, поддерживающим этот пакет отчётов.

Example 3–4. Print All Report Panel Names

```
load_package report
project_open myproject
load_report
set panel_names [get_report_panel_names]
foreach panel_name $panel_names {
post_message "$panel_name"
}
```

В примере 3-5 выводится количество неудачных путей в каждом тактовом домене вашего проекта. В нём используется цикл доступа к каждой строке в панели отчёта итога работы временного анализатора. Тактовые домены перечислены в столбце с именем **Type** в формате Clock Setup:'<*clock name*>'. Прочая общая информация представлена в столбце также представлена в столбце **Type**. Если в столбце **Type** встречается последовательность “Clock Setup*”, скрипт выводит количество неудачных путей, перечисленных в столбце с именем **Failed Paths**. Вы можете запускать этот код с любым исполняемым компонентом, поддерживающим этот пакет отчётов.

Example 3–5. Print Number of Failing Paths per Clock

```
load_package report
project_open my-project
load_report
set report_panel_name "Timing Analyzer||Timing Analyzer Summary"
set num_rows [get_number_of_rows -name $report_panel_name]

# Get the column indices for the Type and Failed Paths columns
set type_column [get_report_panel_column_index -name \
   $report_panel_name "Type"]
set failed_paths_column [get_report_panel_column_index -name \
   $report_panel_name "Failed Paths"]

# Go through each line in the report panel
for {set i 1} {$i < $num_rows} {incr i} {

   # Get the row of data, then the type of summary
   # information in the row, and the number of failed paths
   set report_row [get_report_panel_row -name \
   $report_panel_name -row $i]
   set row_type [lindex $report_row $type_column]
   set failed_paths [lindex $report_row $failed_paths_column]
   if { [string match "Clock Setup*" $row_type] } {
      puts "$row_type has $failed_paths failing paths"
   }
}
unload_report
```

Создание .csv файлов для Excel

Программа Microsoft Excel иногда используется для манипулирования результатами временного анализа. Вы можете создать **.csv** файл для импорта в Excel данных из любого отчёта Quartus II. В примере 3-6 показан простой способ создания **.csv** файла с данными из панели отчёта временного анализатора. Вы можете изменить скрипт для использования аргументов командной строки для пропуска имени проекта, панели отчёта и выходного файла. Вы можете запускать этот код с любым исполняемым компонентом, поддерживающим этот пакет отчётов.

Example 3–6. Create .csv Files from Reports

```
load_package report
project_open my-project

load_report

# This is the name of the report panel to save as a CSV file
set panel_name "Timing Analyzer||Clock Setup: 'clk'"
set csv_file "output.csv"

set fh [open $csv_file w]
set num_rows [get_number_of_rows -name $panel_name]

# Go through all the rows in the report file, including the
# row with headings, and write out the comma-separated data
for { set i 0 } { $i < $num_rows } { incr i } {
    set row_data [get_report_panel_row -name $panel_name \
        -row $i]
    puts $fh [join $row_data ","]
}

close $fh
unload_report
```

Временной анализ

Программа Quartus II содержит полные Tcl APIs для временного анализатора TimeQuest в пакетах **sta**, **sdc** и **sdc_ext**. Эта секция содержит только вводную информацию о TimeQuest Tcl API.

Временной анализатор Quartus II TimeQuest поддерживает SDC команды из пакета **sdc**.

Обратитесь к главе "Временной анализатор Quartus II TimeQuest" в томе 3 настольной книги Quartus II за подробной информацией о том, как выполнять временной анализ во временном анализаторе Quartus II TimeQuest.

Автоматическое исполнение скрипта

Вы можете сконфигурировать скрипты на автоматический запуск в любой точке во время компиляции. Воспользуйтесь этим средством для автоматического запуска скриптов, выполняющих собственные варианты отчётов, создающих специфические назначения или выполняющих прочие задачи.

Три глобальных назначения контролируют, когда автоматически запускать скрипт:

- PRE_FLOW_SCRIPT_FILE – перед началом процесса
- POST_MODULE_SCRIPT_FILE – после завершения модуля
- POST_FLOW_SCRIPT_FILE – после завершения процесса

В исполняемом компоненте Quartus II модуль выполняет один пункт в процессе. Например, два модуля: Анализ и Синтез (**quartus_map**) и Временной Анализ (**quartus_sta**).

Процесс – это последовательность модулей, которые запускает программа Quartus II с предопределёнными опциями. Например, компиляция проекта – это процесс, который обычно состоит из следующих пунктов (выполняемых выделенным модулем):

1. Анализ и Синтез (**quartus_map**)
2. Компоновщик (**quartus_fit**)
3. Ассемблер (**quartus_asm**)
4. Временной анализатор (**quartus_sta**)

Прочие процессы описаны в разделе помощи Tcl команды **execute_flow**. К тому же, многие команды в меню **Processing** графической оболочки Quartus II связаны с этим процессом разработки.

Чтобы сделать автоматические назначения, запустите скрипт и добавьте назначения с помощью следующей формы в ваш **.qsf** файл проекта:

set_global_assignment -name <*assignment name*> <*executable*>:<*script name*>

Программа Quartus II запустит скрипт, как показано в примере 3-7.

Example 3–7.

```
<executable> -t <script name> <flow or module name> <project name> <revision name>
```

Первый аргумент, сохраняемый в переменной **argv** (или в quartus(args)), это имя процесса или модуля, запущенного в исполнение, и зависимый от используемых вами назначений. Второй аргумент – это имя проекта, а третий аргумент – это имя версии.

Когда вы используете назначение POST_MODULE_SCRIPT_FILE, заданный скрипт автоматически запускается после каждого исполняемого компонента в процессе. Вы можете использовать строку сравнения с именем модуля (первый аргумент пропускается скриптом), чтобы выделить процессу скрипта определённый модуль.

Примеры исполняемых компонентов

В примере 3-8 показана работа автоматического исполнения скрипта в полном процессе, если ваш проект называется **top** с текущей версией **rev_1**, и у вас есть следующие назначения в **.qsf** файле вашего проекта.

Example 3–8.

```
set_global_assignment -name PRE_FLOW_SCRIPT_FILE quartus_sh:first.tcl
set_global_assignment -name POST_MODULE_SCRIPT_FILE quartus_sh:next.tcl
set_global_assignment -name POST_FLOW_SCRIPT_FILE quartus_sh:last.tcl
```

Когда вы компилируете ваш проект, назначение PRE_FLOW_SCRIPT_FILE вызывает следующую команду, запускаемую перед началом компиляции:

quartus_sh -t first.tcl compile top rev_1

Затем программа Quartus II запускает компиляцию с анализом и синтезом, выполняемую исполняемым компонентом **quartus_map**. После завершения анализа и синтеза, назначение POST_MODULE_SCRIPT_FILE вызывает запуск следующей команды:

quartus_sh -t next.tcl quartus_map top rev_1

Затем программа Quartus II продолжает компиляцию с компоновщиком, выполняемую исполняемым компонентом **quartus_fit**. После завершения компоновщика, назначение POST_MODULE_SCRIPT_FILE вызывает запуск следующей команды:

```
quartus_sh -t next.tcl quartus_fit top rev_1
```

Соответствующая команда запускается после каждой следующей стадии компиляции. Когда компиляция завершена, назначение POST_FLOW_SCRIPT_FILE вызывает запуск следующей команды:

```
quartus_sh -t last.tcl compile top rev_1
```

Контролирование процессов

Назначение POST_MODULE_SCRIPT_FILE вызывает запуск скрипта после каждого модуля. Поскольку один и тот же скрипт запускается послед каждого модуля, вы должны включить в него некоторые условные выражения, которые ограничивают использование вашего скрипта в определённых модулях.

Например, если вы хотите, чтобы скрипт запускался только после временного анализатора, вы должны включить в него тест, схожий с представленным в примере 3-9. Этот тест проверяет процесс или имя модуля в качестве первого аргумента скрипта и исполняет код, когда модуль называется **quartus_sta**.

Example 3–9. Restrict Processing to a Single Module

```
set module [lindex $quartus(args) 0]

if [string match "quartus_sta" $module] {

    # Include commands here that are run
    # after timing analysis
    # Use the post-message command to display
    # messages
    post_message "Running after timing analysis"
}
```

Отображение сообщений

Поскольку способом работы программы Quartus II является автоматический запуск скриптов, вы должны использовать команду **post_message** для отображения сообщений, вместо команды **puts**. Такое требование применяется только для скриптов, запускаемых после трёх назначений, перечисленных в секции "Автоматическое исполнение скрипта" на стр. 3-12.

Обратитесь к секции "Использование команды post_message" на стр. 3-16 за подробной информацией об этой команде.