Другие средства скрипта

Quartus II Tcl API содержит другие команды общего назначения и средства, описываемые в этой секции.

Обычное наименование шины

Программа Quartus II поддерживает обычное наименование шины. Обычное наименование шины необходимо для того, чтобы квадратные скобки, используемые для задания ширины шины в HDL коде, не вызывали интерпретации их в качестве команд Tcl. Например, один сигнал в шине с именем address идентифицирован как address[0] вместо address\[0\]. Обычное наименование шины удобно при создании назначений, как в примере 3-10.

Example 3–10. Natural Bus Naming

```
set_location_assignment -to address[10] Pin_M20
```

Программа Quartus II по умолчанию использует обычное наименование шины. Вы можете выключить обычное наименование шины с помощью команды **disable_natural_bus_naming**. За подробной информацией об обычном наименовании шины, введите следующее в командной строке Quartus II Tcl:

```
enable_natural_bus_naming -h
```

Опция сокращения имён

Вы можете использовать укороченные версии опций команд, пока они могут быть однозначно интерпретированы. Например, команда **project_open** имеет две опции: **-current_revision** и **-revision**. Вы сможете использовать следующие аббревиатуры опции **-revision**: **-r**, **-re**, **-rev**, **-revi**, **-revis** и **-revisio**. Вы можете использовать сокращение **-r**, потому что команда **project_open** не имеет более опций, начинающихся с буквы r. Однако команда **report_timing** содержит опции **-recovery** и **-removal**. Вы не можете использовать **-r** или **-re** для сокращения любой из этих опций, поскольку такая аббревиатура не будет уникальной для любой из опций.

Использование коллекции команд

Некоторые Quartus II Tcl функции возвращают очень большой объём данных, которые могут быть неэффективными в качестве списков Tcl. Эти структуры данных считаются коллекциями. Quartus II Tcl API использует коллекцию ID для доступа к коллекции. Две команды Quartus II Tcl работают с коллекциями: **foreach_in_collection** и **get_collection_size**. Используйте этот набор команд для доступа к переменным в ID коллекциях.

За дополнительной информацией о том, как Quartus II Tcl команды возвращают коллекцию ID, обратитесь к *foreach_in_collection* в разделе помощи Quartus II.

Команда foreach_in_collection

Команда **foreach_in_collection** похожа на Tcl команду **foreach**. Используйте её для итерации всех элементов коллекции. В примере 3-11 выводятся все элементы назначений в открытом проекте.

Example 3–11. Using Collection Commands

```
set all_instance_assignments [get_all_instance_assignments -name *]
foreach_in_collection asgn $all_instance_assignments {
    # Information about each assignment is
    # returned in a list. For information
    # about the list elements, refer to Help
    # for the get-all-instance-assignments command.
    set to [lindex $asgn 2]
    set name [lindex $asgn 3]
    set value [lindex $asgn 4]
    puts "Assignment to $to: $name = $value"
}
```

Команда get_collection_size

Используйте команду **get_collection_size** для получения количества элементов в коллекции. В примере 3-12 выводится количество глобальных назначений в открытом проекте.

Example 3–12. get_collection_size Command

```
set all_global_assignments [get_all_global_assignments -name *]
set num_global_assignments [get_collection_size $all_global_assignments]
puts "There are $num_global_assignments global assignments in your project"
```

Использование команды post_message

Для вывода сообщения в формате, похожем на сообщения программы Quartus II, используйте команду **post_message**. Сообщения, выводимые командной **post_message,** появляются во вкладке **System** окна Messages в графической оболочке Quartus II, все они записаны по стандарту, по которому запущен скрипт. Аргументы команды **post_message** содержат дополнительный тип сообщений и необходимую строку сообщения.

Тип сообщения может быть одним из следующих:

- info (по умолчанию) – информация
- extra_info – специальная информация
- warning – предупреждение
- critical_warning – критическое предупреждение
- error - ошибка

Если вы не задали тип, программа Quartus II использует по умолчанию info. Когда вы используете программу Quartus II в Windows, вы можете выделить цветом сообщения, отображаемые в системной командной строке с помощью команды **post_message**. Добавьте следующую строку в ваш **quartus2.ini** файл:

DISPLAY_COMMAND_LINE_MESSAGES_IN_COLOR = on

В примере 3-13 показано использование команды **post_message**.

Example 3–13. post_message command

```
post_message -type warning "Design has gated clocks"
```

Доступ к аргументам командной строки

Множество Tcl скриптов были разработаны для связи с аргументами командной строки, такими как имя проекта или версия. Глобальная переменная **quartus(args)** – это список аргументов, введённых в командной строке, следующих за именем Tcl скрипта. В пример 3-14 показан код, который выводит все аргументы в переменной **quartus(args)**.

Example 3–14. Simple Command-Line Argument Access

```
set i 0
foreach arg $quartus(args) {
    puts "The value at index $i is $arg"
    incr i
}
```

Если вы скопируете скрипт из предыдущего примера в файл с именем **print_args.tcl**, то он отобразит следующий выход, когда вы введёте в командной строке команду, показанную в примере 3-15.

Example 3–15. Passing Command-Line Arguments to Scripts

```
quartus_sh -t print_args.tcl my_project 100MHz ↵
The value at index 0 is my_project
The value at index 1 is 100MHz
```

Использование пакета cmdline

Вы можете использовать пакет **cmdline**, включенный в программу Quartus II, для более надёжного сохранения собственных аргументов командной строки. Пакет **cmdline** поддерживает аргументы командной строки с формами -<*option*> <*value*>.

В примере 3-16 используется пакет **cmdline**.

Example 3–16. cmdline Package

```
package require cmdline
variable ::argv0 $::quartus(args)
set options {
    { "project.arg" "" "Project name" }
    { "frequency.arg" "" "Frequency" }
}
set usage "You need to specify options and values"

array set optshash [::cmdline::getoptions ::argv $options $usage]
puts "The project name is $optshash(project)"
puts "The frequency is $optshash(frequency)"
```

Если вы сохранили эти команды в Tcl скрипте с именем **print_cmd_args.tcl,** вы сможете увидеть следующий результат, когда введёте команду, показанную в примере 3-23 в командной строке.

Example 3–17. Passing Command-Line Arguments for Scripts

```
quartus_sh -t print_cmd_args.tcl -project my_project -frequency 100MHz ↵
The project name is my_project
The frequency is 100MHz
```

Виртуально, все Quartus II Tcl скрипты должны открывать проект. В примере 3-18 открывается проект, а вы можете дополнительно задать имя версии. Пример проверяет наличие заданной версии проекта. Если она существует, пример открывает её текущую версию, или заданную вами версию.

Example 3–18. Full-Featured Method to Open Projects

```
package require cmdline
variable ::argv0 $::quartus(args)
set options { \
{ "project.arg" "" "Project Name" } \
{ "revision.arg" "" "Revision Name" } \
}
array set optshash [::cmdline::getoptions ::argv0 $options]

# Ensure the project exists before trying to open it
if {[project_exists $optshash(project)]} {

    if {[string equal "" $optshash(revision)]} {

        # There is no revision name specified, so default
        # to the current revision
        project_open $optshash(project) -current_revision
    } else {

        # There is a revision name specified, so open the
        # project with that revision
        project_open $optshash(project) -revision \
            $optshash(revision)
    }
} else {
    puts "Project $optshash(project) does not exist"
    exit 1
}
# The rest of your script goes here
```

Если вам не нужна подобная гибкость или проверка на ошибку, вы можете использовать только команду **project_open**, как показано в примере 3-19.

Example 3–19. Simple Method to Open Projects

```
set proj_name [lindex $argv 0]
project_open $proj_name
```

Использование оболочки Quartus II Tcl в интерактивном режиме

В этой секции описывается пример использования интерактивной оболочки **quartus_sh** для создания некоторый назначений проекта и компиляции учебного проекта фильтра с конечной импульсной характеристикой (КИХ-фильтра) (FIR). В этом примере подразумевается, что у вас есть файлы учебного проекта FIR filter в директории проекта.

Для начала введите следующее в системной командной строке для запуска интерактивной оболочки Tcl:

quartus_sh -s

Создайте новый проект - **fir_filter** - версии **filtref**, введя следующую команду в командной строке Tcl:

project_new -revision filtref fir_filter

Если файл проекта и имя проекта совпадают, программа Quartus II даёт имени версии то же самое имя, что и проекту.

Поскольку имя версии **filtref** создаётся для файла верхнего уровня, все файлы проекта автоматически получают его в иерархическом древе.

Далее, установите глобальное назначение для чипа с помощью следующей команды:

set_global_assignment -name family Cyclone

Чтобы узнать больше об именах назначений, которые вы можете использовать в опции -name, обратитесь к разделу помощи Quartus II.

Для параметров назначений, которые содержат пробелы, подобные параметры должны заключаться в кавычки.

Для быстрой компиляции проекта используйте пакет ::quartus::flow, который собственно экспортирует назначения нового проекта и компилирует проект, используя простую последовательность исполняемых компонентов командной строки. Сначала загрузите проект:

load_package flow

Возвращается следующее:

1.0

Для выполнения полной компиляции проекта FIR filter, используйте команду **execute_flow** с опцией -compile:

exectue_flow –compile

Эта команда компилирует учебный проект FIR filter, экспортирует назначения проекта и запускает quartus_map, quartus_fit, quartus_asm и quartus_sta. Эта последовательность соответствует выбору **Start Compilation** в меню Processing графической оболочки Quartus II.

Когда вы закончите проект, закройте его, используя команду **project_close**, как показано в примере 3-20.

Example 3–20.

```
project_close ↵
```

Для выхода из интерактивного режима Tcl, введите **exit** в командной строке.

Использование оболочки tclsh

В операционных системах UNIX и Linux, оболочка **tclsh**, включённая в программу Quartus II, инициализируется с минимальной переменной окружения PATH. В результате, системные команды могут быть не доступны в оболочке **tclsh**, поскольку определённые директории не находятся в переменной окружения PATH. Чтобы включить другие директории путь поиска оболочки **tclsh**, установите переменную окружения QUARTUS_INIT_PATH перед запуском оболочки **tclsh**. Директории в переменной окружения QUARTUS_INIT_PATH находятся оболочкой **tclsh** во время исполнения системной команды.