Поддержка Tcl

Для поддержки автоматизации, исходники и пробники в системе поддерживают процедуры, описанные в этой главе, в виде команд Tcl. Пакет Tcl для исходников и пробников в системе содержится по умолчанию, когда вы запускаете **quartus_stp**.

Tcl интерфейс для исходников и пробников в системе – это мощная платформа помощи при отладки вашего проекта. Главным образом это помогает для отладки проектов, которым требуется переключение множества различных контрольных входов. Вы можете группировать различные команды, используя Tcl скрипт, чтобы получить свой собственный набор команд.

За дополнительной информацией о Tcl скриптах обратитесь к главе "Скриптирование Tcl" в томе 2 Настольной книги Quartus II. За дополнительной информацией о настройках и ограничениях в программе Quartus II, обратитесь к "Ссылке на руководство файлом настроек Quartus II". За дополнительной информацией о скриптах командной строки, обратитесь к главе "Скриптирование в командной строке" в томе 2 Настольной книги Quartus II.

В таблице 17-3 показаны Tcl команды, которые вы можете использовать совместно с Исходниками и пробниками в системе.

Таблица 17-3. Tcl команды для исходников и пробников в системе (часть 1 из 2)

Команда	**Аргумент**	**Описание**
start_insystem_source_probe	-device_name <имя чипа> -hardware_name <имя оборудования >	Открывает помеченный чип, используя определённое оборудование. Вызывайте эту команду перед началом любой транзакции.
get_insystem_source_ probe_instance_info	-device_name <имя чипа> -hardware_name <имя оборудования >	Возвращает список всех элементов *altsource_probe* в вашем проекте. Каждая запись возвращается в следующем формате: {<Индекс элемента>, <размер исходника>, <размер пробника>, <имя элемента>}
read_probe_data	-instance_index <индекс элемента> -value_in_hex (опционально)	Извлекает текущее значение пробника. Строка возвращает определённый статус на каждом пробнике, самый старший разряд – самый левый бит.
read_source_data	-instance_index <индекс элемента> -value_in_hex (опционально)	Извлекает текущее значение исходника. Строка возвращает определённый статус на каждом исходнике, MSB – самый левый бит.

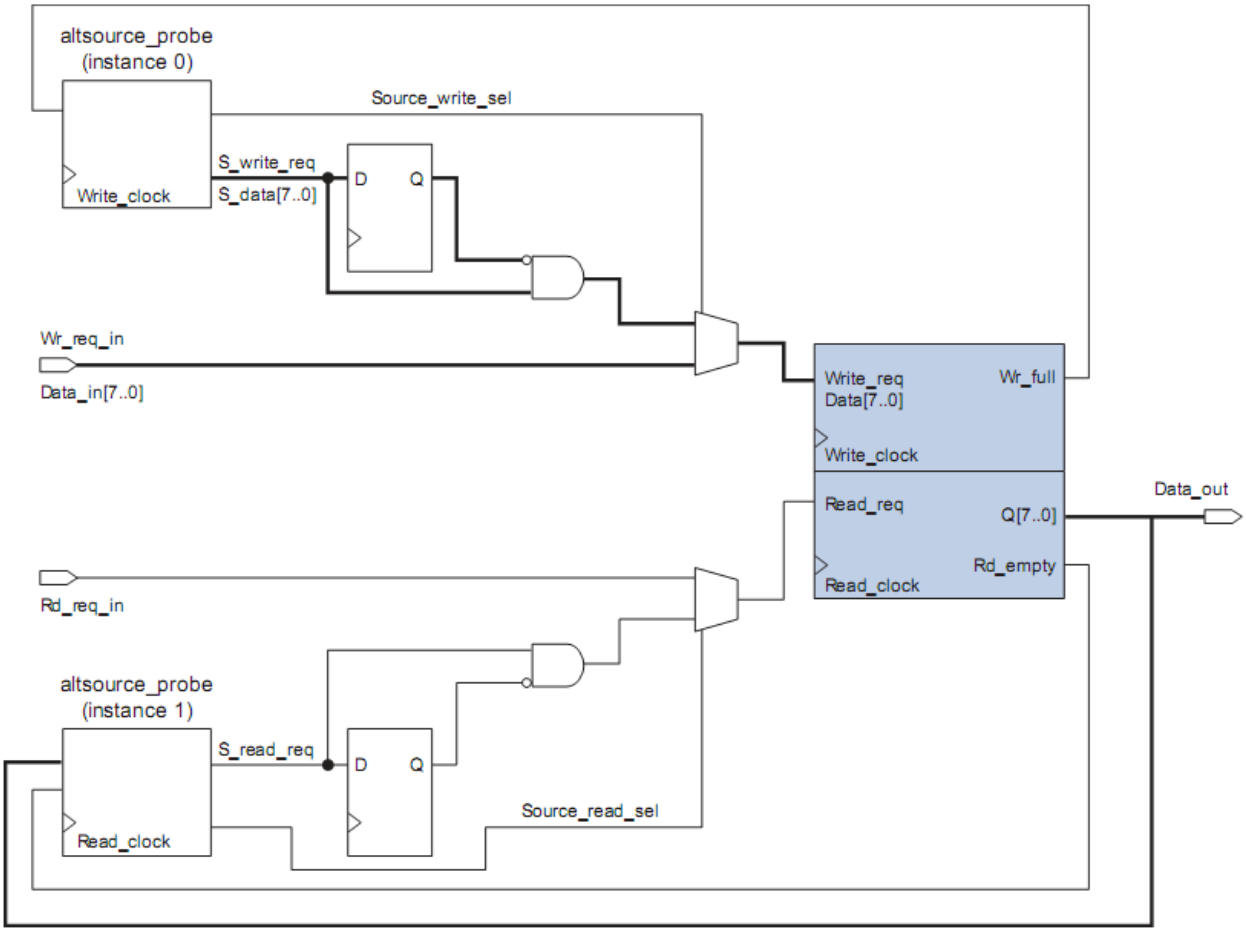
Таблица 17-3. Tcl команды для исходников и пробников в системе (часть 2 из 2)

Команда	**Аргумент**	**Описание**
write_source_data	-instance_index <индекс элемента> -value <значение> -value_in_hex (опционально)	Устанавливает значение для исходников. Бинарная строка посылается в порты исходников, самый старший разряд (MSB) – самый левый бит.
end_interactive_probe	нет	Отключает цепь JTAG. Используйте эту команду, когда завершены все транзакции.

В примере 17-1 показаны отрывки из Tcl скрипта процедуры, которая контролирует элементы *altsource_probe* проекта, показанного на рисунке 17-5. Пример проекта содержит DCFIFO с элементами *altsource_probe* для чтения из DCFIFO и записи в DCFIFO. Установка контрольных мультиплексоров добавляется к проекту для контроля за следованием данных в DCFIFO между входными выводами и элементами *altsource_probe*. Тактовый генератор добавляется для чтения запросов и записи запросов контрольных линий, чтобы гарантировать одиночную выборку чтения или записи. Элементы *altsource_probe*, которые используются со скриптом в примере 17-1, позволяют видеть содержимое FIFO, выполняя одиночную выборку записи и чтения операций, и отчитываются о состоянии статусных флагов переполнения или опустошения.

Скрипт Tcl прекрасно подходит для ситуации отладки, когда вы бы хотели опустошать или предварительно загружать FIFO в вашем проекте. Например, вы хотите использовать это средство для предварительной загрузки FIFO, чтобы создать состояние триггеров, которые вы установили в логическом анализаторе SignalTap II.

Figure 17–5. A DCFIFO Example Design Controlled by the Tcl Script in Example 17–1



Example 17–1. Tcl Script Procedures for Reading and Writing to the DCFIFO in Figure 17–5 (Part 1 of 2)

```
## Setup USB hardware  - assumes only USB Blaster is installed and
## an FPGA is the only device in the JTAG chain

set usb [lindex [get_hardware_names] 0]
set device_name [lindex [get_device_names -hardware_name $usb] 0]
## write procedure :  argument value is integer

proc write {value} {

    global device_name usb
    variable full

    start_insystem_source_probe -device_name $device_name -hardware_name $usb

    #read full flag
    set full [read_probe_data -instance_index 0]


    if {$full == 1} {end_insystem_source_probe
    return "Write Buffer Full"
    }
```

Example 17–1. Tcl Script Procedures for Reading and Writing to the DCFIFO in Figure 17–5 (Part 2 of 2)

```
    ##toggle select line, drive value onto port, toggle enable
    ##bits 7:0 of instance 0 is S_data[7:0]; bit 8 = S_write_req;
    ##bit 9 = Source_write_sel

    ##int2bits is custom procedure that returns a bitstring from an integer
    ## argument


    write_source_data -instance_index 0 -value /[int2bits [expr 0x200 | $value]]
    write_source_data -instance_index 0 -value [int2bits [expr 0x300 | $value]]

    ##clear transaction

    write_source_data -instance_index 0 -value 0

    end_insystem_source_probe
}

proc read {} {

    global device_name usb
    variable empty
    start_insystem_source_probe -device_name $device_name -hardware_name $usb

    ##read empty flag : probe port[7:0] reads FIFO output; bit 8 reads empty_flag

    set empty [read_probe_data -instance_index 1]


    if {[regexp {1........} $empty]} { end_insystem_source_probe
    return "FIFO empty" }

    ## toggle select line for read transaction
    ## Source_read_sel = bit 0; s_read_reg = bit 1

    ## pulse read enable on DC FIFO
    write_source_data -instance_index 1 -value 0x1 -value_in_hex
    write_source_data -instance_index 1 -value 0x3 -value_in_hex

    set x [read_probe_data -instance_index 1 ]

    end_insystem_source_probe

    return $x
}
```