Продолжение. Начало в № 9`2006

Микропроцессор своими руками-4.
Как отладить встроенный в FPGA микроконтроллер?

Иосиф КАРШЕНБОЙМ
losifk@narod.ru

В предыдущих частях статьи мы рассмотрели то, как можно отлаживать встроенные в FPGA микроконтроллеры, а также как работает порт JTAG. На сей раз нас интересует программно-аппаратная поддержка режима тестирования.

Технические подробности аппаратных устройств сопряжения

Аппаратные устройства сопряжения для разных микросхем изготавливаются как самими производителями микросхем, так и независимыми производителями отладочных средств. Но, к сожалению, в первом случае аппаратные устройства сопряжения довольно сильно отличаются друг от друга. Причем нет однообразия даже в названиях устройств — «Байт-Бластер», «Кабель № 3», «адаптер» и так далее. Традиционно такие устройства подключались на LPT-порт компьютера. На рис. 25–26 приведены схемы двух таких устройств. Первое предназначено для микросхем фирмы Xilinx, второе — для микросхем фирмы Altera [11]. Специально для данной статьи автором была сделана программа Jtag Tool, описание и способы работы с ней будут приведены далее. Программа Jtag Tool и ее исходные тексты могут быть свободно получены читателями с сайта автора. В таблицах 6–9 приведены характеристики аппаратных устройств сопряжения. Эти таблицы содержат названия переменных, которые используются в программе Jtag Tool. Так, например, сигналу DONE, изображенному на схеме, соответствует маска DONE_MASK, которая используется в программе Jtag Tool для работы с LPT-портом и указывает положение бита в байте, определяющего управление соответствующего сигнала.

Из множества проектов различных JTAG-адаптеров, которые имеются как на сайтах с открытыми проектами, так и на сайтах коммерческих фирм, хочется представить вниманию читателей следующий проект — Chameleon, с ним можно ознакомиться на сайте фирмы Amontec [12]. Внимания читателей заслуживает и акселератор для адаптера Chameleon [13]. Данное изделие близко к тому, что делала фирма Xilinx для своих адаптеров, работавших через LPT-порт.

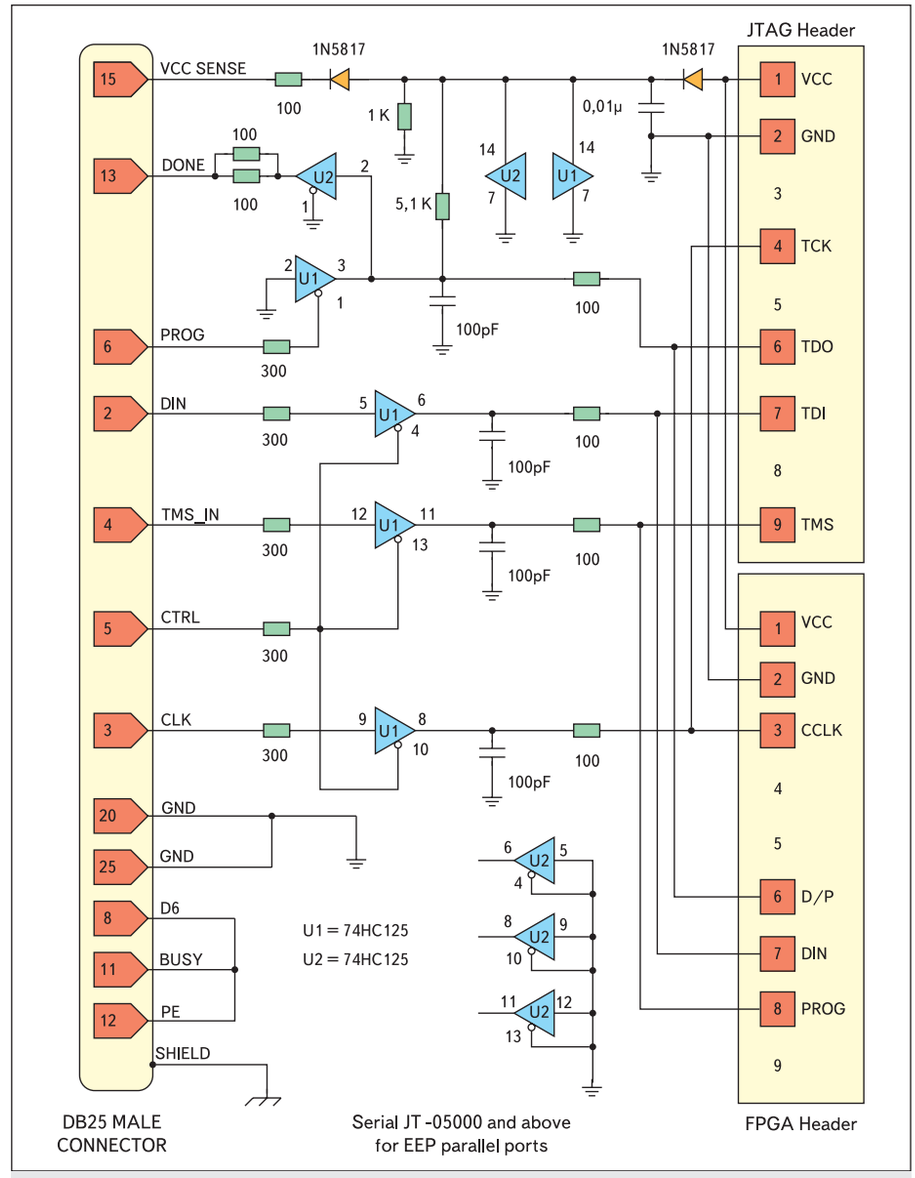

Рис. 25. Схема аппаратного адаптера, называемого по документации фирмы Xilinx «Кабель № 3»

Таблица 6. Сигналы и их описание для Cable3-Xilinx

Название сигнала	Конт. DB25	Направление, порт, бит, уровень	Инверс.	Маска	Jtag Header
VCC_SENSE	15	<= 15 –Error S3+	–	00001000	1
DONE_MASK, TDO_MASK	13	<= 13 +SelectIn S4+	–	00010000	6
PROG_MASK	6	=> 6 Data 4 D4	–	00010000	
D_IN, TDI_MASK	2	=> 2 Data 0 D0	–	00000001	7
TMS_MASK	4	=> 4 Data 2 D2	–	00000100	9
CTRL_MASK	5	=> 5 Data 3 D3	–	00001000	
TCK_MASK, CLK	3	=> 3 Data 1 D1	–	00000010	4
D6_MASK	8	=> 8 Data 6 D6	–	01000000	
BUSY_MASK	11	<= 11 +Busy S7–	+	10000000	
PE_MASK	12	<= 12 +PaperEnd S5+	–	00100000	
TRST_MASK	–			00000000	
SRST_MASK	–			00000000	
GND	20, 25				2

Примечание:
1. Знак «=>» указывает направление передачи сигнала от LPT-порта к порту JTAG.
2. Знак «<=» указывает направление передачи сигнала от порта JTAG к LPT-порту.
3. Обозначения «D0», «D1» и т. д. указывают на принадлежность сигнала к LPT-порту, регистру «Данные»; цифра указывает на разряд данного регистра.
4. Обозначения «S5», «S7» и т. д. указывают на принадлежность сигнала к LPT-порту, регистру «Статус»; цифра указывает на разряд данного регистра.
5. Выражение +Busy S7– означает, что сигнал Busy приходит в LPT-порт, в регистр «Статус», бит 7, причем активный низкий уровень приходящего сигнала читается как активный высокий уровень со стороны PC.

Таблица 8. Сигналы и их описание для ByteBlaster-Altera

Название сигнала	Конт. DB25	Направление, порт, бит, уровень	Инверс.	Маска	Jtag Header
VCC_SENSE	15	<= 15 –Error S3+	–	00001000	4
TDO_MASK	11	<= 11 +Busy S7–	–	10000000	3
TDI_MASK	8	=> 8 Data 6 D6	–	01000000	9
TMS_MASK	3	=> 3 Data 1 D1	–	00000010	5
CTRL_MASK	14	=> 14 –AutoFd C1–	–	00000010	
TCK_MASK	2	=> 2 Data 0 D0	–	00000001	1
D6_MASK	6	=> 6 Data 4 D4	–	00001000	
BUSY_MASK	10	<= 10 –Ack S6+	+	01000000	
PE_MASK	–		–	00000000	
TRST_MASK	–			00000000	
SRST_MASK	–			00000000	
GND	20, 25				2, 10

Примечание:
6. Знак «=>» указывает направление передачи сигнала от LPT-порта к порту JTAG.
7. Знак «<=» указывает направление передачи сигнала от порта JTAG к LPT-порту.
8. Обозначения «D0», «D1» и т. д. указывают на принадлежность сигнала к LPT-порту, регистру «Данные»; цифра указывает на разряд данного регистра.
9. Обозначения «S5», «S7» и т. д. указывают на принадлежность сигнала к LPT-порту, регистру «Статус»; цифра указывает на разряд данного регистра.
10. Выражение +Busy S7– означает, что сигнал Busy приходит в LPT-порт, в регистр «Статус», бит 7, причем активный низкий уровень приходящего сигнала читается как активный высокий уровень со стороны PC.

Таблица 7. Параметры инициализации порта для Cable3-Xilinx

Название параметра	Описание параметра	Маска
OUTPUT_INVERT	data port bits that should be inverted	00000000
INPUT_INVERT	status port that should be inverted	10000000
PORT_INIT	initialize data port with this value	00000000

Таблица 9. Параметры инициализации порта для Cable3-Xilinx

Название параметра	Описание параметра	Маска
OUTPUT_INVERT	data port bits that should be inverted	00000000
INPUT_INVERT	status port that should be inverted	10000000
PORT_INIT	Initialize data port with this value	00000000

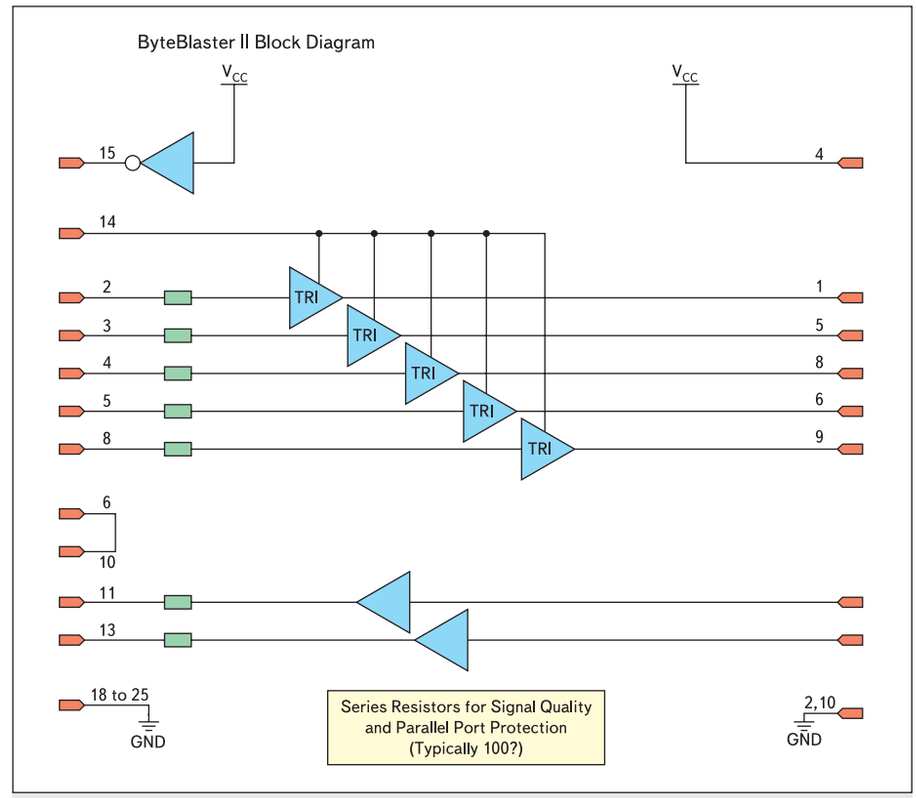

Рис. 26. Схема аппаратного адаптера, называемого по документации фирмы Altera ByteBlaster

Производительность работы аппаратных устройств сопряжения

С развитием интерфейса USB производители микросхем стали предлагать аппаратные устройства сопряжения, работающие по этому интерфейсу. Но, к сожалению, вся информация о драйверах этих устройств и о возможности обращаться к ним из «самодельной» программы полностью закрыта, либо предоставляется только на коммерческой основе. Независимые производители отладочных средств неоднократно пытались «выправить» эту ситуацию, но, к сожалению, универсальные аппаратные устройства сопряжения более дороги и не получили такого же широкого распространения, как «фирменные» средства сопряжения. В таблице 10 приведены некоторые характеристики, позволяющие оценить производительность и стоимость аппаратных адаптеров, которые выпускают различные компании (по данным, представленным фирмой MT-System).

Таблица 10. Сравнительные характеристики распространенных JTAG адаптеров

Адаптер	Подключение к ПК	Скорость загрузки, кбайт/с	Стоимость, $
Wiggler	LPT	16	20–25
Raven	LPT	64	250 (Chameleon)
ULINK	USB	28	200
J-Link	USB	120	250

Теперь мы переходим к рассмотрению компонентов в FPGA, при помощи которых возможно осуществлять отладку проектов пользователя. Начнем с микросхем фирмы Xilinx.

Компонент BSCAN_SPARTAN3 для микросхем фирмы Xilinx рассмотрим на примере компонента, применяемого для микросхем серии SPARTAN3. Он представлен как Primitive: Spartan-3 Boundary Scan Logic Control Circuit, и его описание можно найти в документе [14]. На рис. 27 этот компонент изображен в виде «черного ящика». Описание блока BSCAN_SPARTAN3 — на рис. 28–29. Данные по выводам блока BSCAN_SPARTAN3 приведены в таблице 11. Главное, что необхо-

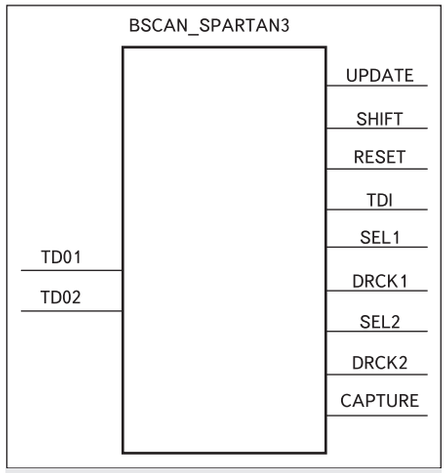

Рис. 27. Компонент Spartan-3 Boundary Scan Logic Control Circuit в виде «черного ящика»

Таблица 11. Краткое описание выводов блока BSCAN_SPARTAN3

Вывод блока	Описание сигнала
RESET	Выход TAP-контроллера, сигнал RESET
CAPTURE	Выход TAP-контроллера, сигнал CAPTURE
SHIFT	Выход TAP-контроллера, сигнал SHIFT
UPDATE	Выход TAP-контроллера, сигнал UPDATE
DRCK1	Data register output — USER1 functions — так в документации называет эти выводы производитель
DRCK2	Data register output — USER2 functions — так в документации называет эти выводы производитель
SEL1	Сигнал с дешифратора адреса — выполняются действия по адресу USER1
SEL2	Сигнал с дешифратора адреса — выполняются действия по адресу USER2
TDI	Выход TAP-контроллера, сигнал TDI, используется для записи данных в логику пользователя, которая находится по адресу USER1 и USER2
TDO1	Вход данных от логики пользователя, читаемой по адресу USER1
TDO2	Вход данных от логики пользователя, читаемой по адресу USER2

```
PORT
(
ir_out : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
tdo : IN STD_LOGIC ;
ir_in : OUT STD_LOGIC_VECTOR (1 DOWNTO 0);
                tck : OUT STD_LOGIC ;
                tdi : OUT STD_LOGIC ;
                tms : OUT STD_LOGIC ;
jtag_state_cdr : OUT STD_LOGIC ;

. . . здесь еще 14 выводов, соответствующих
состояниям TAP-контроллера устройства

                jtag_state_uir : OUT STD_LOGIC ;

                virtual_state_cdr : OUT STD_LOGIC ;
                virtual_state_cir : OUT STD_LOGIC ;
                virtual_state_e1dr : OUT STD_LOGIC ;
                virtual_state_e2dr : OUT STD_LOGIC ;
                virtual_state_pdr : OUT STD_LOGIC ;
                virtual_state_sdr : OUT STD_LOGIC ;
                virtual_state_udr : OUT STD_LOGIC ;
                virtual_state_uir : OUT STD_LOGIC);
```

Рис. 30. Часть описания блока sld_virtual_jtag на VHDL

```
VHDL Instantiation Template
-- Component Declaration for BSCAN_SPARTAN3 should be placed
-- after architecture statement but before begin keyword

component BSCAN_SPARTAN3
port (CAPTURE : out STD_ULOGIC;
DRCK1 : out STD_ULOGIC;
DRCK2 : out STD_ULOGIC;
RESET : out STD_ULOGIC;
SEL1 : out STD_ULOGIC;
SEL2 : out STD_ULOGIC;
SHIFT : out STD_ULOGIC;
TDI : out STD_ULOGIC;
UPDATE : out STD_ULOGIC;
TD01 : in STD_ULOGIC;
TD02 : in STD_ULOGIC);
end component;

-- Component Attribute specification for BSCAN_SPARTAN3
-- should be placed after architecture declaration but
-- before the begin keyword
-- Enter attributes here
-- Component Instantiation for BSCAN_SPARTAN3 should be
-- placed in architecture after the begin keyword

BSCAN_SPARTAN3_INSTANCE_NAME : BSCAN_SPARTAN3
port map (CAPTURE => user_CAPTURE,
DRCK1 => user_DRCK1,
DRCK2 => user_DRCK2,
RESET => user_RESET,
SEL1 => user_SEL1,
SEL2 => user_SEL2,
SHIFT => user_SHIFT,
TDI => user_TDI,
UPDATE => user_UPDATE,
TD01 => user_TD01,
TD02 => user_TD02);
```

Рис. 28. Описание блока BSCAN_SPARTAN3 на VHDL

```
Verilog Instantiation Template

// BSCAN_SPARTAN3:Boundary Scan primitve for connecting in-
ternal logic
// JTAG interface. Spartan-II
// Xilinx HDL Libraries Guide version 7.1i

BSCAN_SPARTAN3 BSCAN_SPARTAN3_inst (
.CAPTURE(CAPTURE), // CAPTURE output from TAP controller
.DRCK1(DRCK1), // Data register output — USER1 functions
.DRCK2(DRCK2), // Data register output — USER2 functions
.RESET(RESET), // Reset output from TAP controller
.SEL1(SEL1), // USER1 active output
.SEL2(SEL2), // USER2 active output
.SHIFT(SHIFT), // SHIFT output from TAP controller
.TDI(TDI), // TDI output from TAP controller
.UPDATE(UPDATE), // UPDATE output from TAP controller
.TDO1(TDO1), // Data input for USER1 function
.TDO2(TDO2) // Data input for USER2 function
);
```

Рис. 29. Описание блока BSCAN_SPARTAN3 на Verilog

димо отметить, так это то, что адреса команд, работающих с этими блоками, заданы жестко. Пользователь может использовать только две копии (instances) данного компонента в своем проекте. Но именно благодаря тому, что адреса фиксированы, блок содержит в своем составе дешифратор адресов, и пользователь имеет дело с двумя группами сигналов — одной для логики, работающей по адресу USER1, и другой — по адресу USER2.

Информации об этом компоненте крайне мало. Но поскольку он в библиотеке числится как Primitive, можно утверждать, что это аппаратный блок, а следовательно, его применение в проекте не должно сокращать ресурсы, предоставляемые пользователю.

Компонент sld_virtual_jtag для микросхем фирмы Altera

У Altera дела обстоят несколько иначе. Компонент для связи с проектом пользователя называется sld_virtual_jtag [15]. Основное его отличие от ранее рассмотренного в том, что sld_virtual_jtag может иметь в одном проекте до 128 копий (instances). Для конфигурации данной мегафункции может быть использован MegaWizard Plug-In Manager, поставляемый фирмой Altera. В инструкции Megafunction User Guide, глава 2 — Getting Started, на стр. 26–29 приведены примеры использования sld_virtual_jtag. Они выполнены на языках VHDL и Verilog. На рис. 30 — описание блока, получаемого при использовании в проекте копии функции sld_virtual_jtag. Данные по выводам блока sld_virtual_jtag приведены в таблице 12.

Основное отличие блока sld_virtual_jtag от предыдущего в том, что это не аппаратный примитив, а макрос, и, следовательно, при использовании такого блока в проекте для него необходимо выделить ресурсы кристалла.

Более подробное описание входов/выходов, параметров и конфигурирования мегафункции sld_virtual_jtag достойно отдельной статьи, поэтому здесь мы остановимся и перейдем к описанию программного инструмента, предназначенного для облегчения изучения работы с портом JTAG.

Таблица 12. Краткое описание выводов блока sld_virtual_jtag

Вывод блока	Описание сигнала
ir_in	Выходная шина для команд — STD_LOGIC_VECTOR (1 DOWNTO 0)
ir_out	Входная шина для команд — STD_LOGIC_VECTOR (1 DOWNTO 0)
tms	Выход сигнала управления
tdo	Вход данных
tck	Тактовая частота для JTAG-порта, выход
jtag_state_cdr, jtag_state_cir, jtag_state_e1dr, jtag_state_e1ir, jtag_state_e2dr, jtag_state_e2ir, jtag_state_pdr, jtag_state_pir, jtag_state_rti, jtag_state_sdr, jtag_state_sdrs, jtag_state_sir, jtag_state_sirs, jtag_state_tlr, jtag_state_udr, jtag_state_uir	Выходы TAP-контроллера, соответствующие состоянию автомата устройства
virtual_state_cdr, virtual_state_cir, virtual_state_e1dr, virtual_state_e2dr, virtual_state_pdr, virtual_state_sdr, virtual_state_udr, virtual_state_uir	Выходы TAP-контроллера, соответствующие состоянию автомата virtual_jtag-компонента

Программный инструмент

После того, как в нескольких предыдущих журналах были опубликованы статьи о различных программных инструментах, облегчающих разработку проектов, автор получил несколько писем читателей с просьбами выслать им свои программы. Поэтому можно смело сказать о том, что интерес к таким программам есть. Что касается данного случая с отладкой проектов по порту JTAG, то без программной поддержки представить себе отладку просто невозможно. Автором был произведен поиск для выяснения того, какими программами пользуются для аналогичных целей. Кто-то использует скрипты, кто-то программы, запускающиеся из командной строки, а кто-то — визуальные оболочки. Поскольку использование специализированных скриптов для многих читателей вызовет определенные трудности, то пришлось ос-

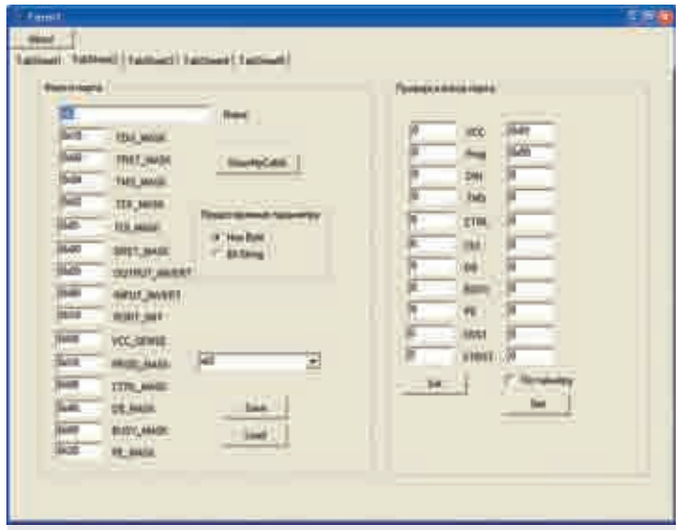

Рис. 31. Окно для ввода и индикации параметров аппаратного адаптера и для работы с LPT-портом

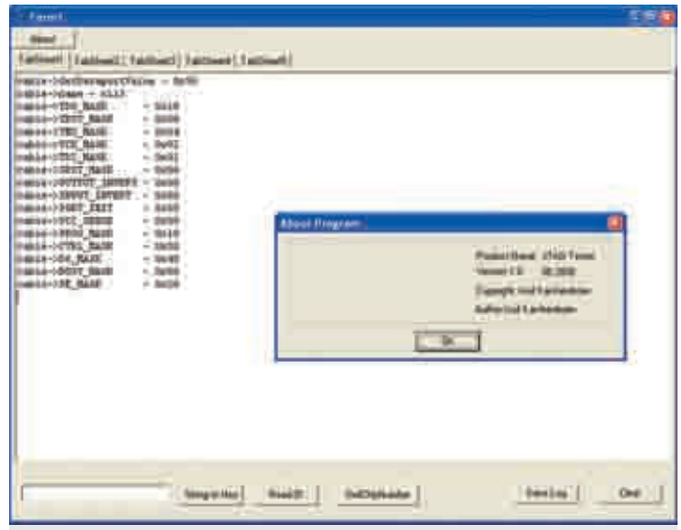

Рис. 32. Окно, на которое выдаются данные мониторинга и результаты работы теста

тановиться на обычных программах с визуальными оболочками, аналогичных тем, что автор разрабатывал в предыдущих проектах. А так как цель данной статьи — обучение технологии работы с портом JTAG, то основное внимание было уделено тому, чтобы предоставить пользователю как можно больше информации о том, что делает программа, какими данными хост обменивается с портом.

Программа должна быть легко перестраиваемой, так, чтобы пользователь смог работать с разными аппаратными адаптерами и различными типами микросхем. Поскольку у аппаратных адаптеров, подключаемых к порту USB, нет единого или хотя бы похожего протокола, то за основу был взят аппаратный адаптер, подключаемый к порту LPT. И последнее требование, которое автор хотел реализовать, — это то, чтобы программа не нуждалась в инсталляции и не была связана с реестром. Такое построение программы особенно удобно в том случае, если вы хотите быстро произвести какие-либо действия на «чужой» машине. Например, вы приходите в стендовую, где находится проверяемое изделие, берете Flash-диск, подключаете его и сразу производите тестирование изделия.

Пользовательский интерфейс и действия, выполняемые программным инструментом — Jtag Tool

Что позволяет этот инструмент?

Программа выполнена как визуальная оболочка, она представляет собой одну форму, на которой расположен блокнот с 5-ю страницами. Файл инициализации находится в той же рабочей директории, откуда запускается сама программа. Для работы с LPT-портом используется известный драйвер GiveIO. Драйвер устанавливается пользователем по прилагаемой к программе инструкции. Все операции с портом выделены в отдельный класс, так что, при желании, читатель сможет заменить нижний уровень программы и использовать любой другой драйвер. Точно так же все действия с аппаратным адаптером выделены в отдельный класс, поэтому можно легко заменить работу с LPT-портом на действия с USB-портом. В реальных условиях микросхемы обычно включены в JTAG-цепочку. И число таких микросхем может быть достаточно велико. Однако для целей обучения достаточно иметь возможность выполнять следующие задачи: работать только с одной микросхемой в цепочке, определять общее число микросхем в цепочке и читать их ID.

И еще одно требование к формату данных для тестов. Как показывает практика, наиболее удобно для таких программных инструментов иметь формат тестов, совместимый со стандартными программами, например с Excel. При использовании Excel-совместимых данных сразу появляется возможность делать сортировки, обработки данных, облегчается редактирование и т. д. В качестве формата был выбран текстовый, где в качестве разделителя используется знак «;».

Итак, во-первых, данная программа позволяет подключить к LPT-порту любой аппаратный адаптер и произвести настройку его параметров. На второй странице блокнота находится окно ввода и отображения параметров аппаратного адаптера и LPT-порта. Для того чтобы указать программе, какие биты порта используются для ввода и вывода данных, пользователь должен ввести параметры маски. Они могут быть введены и затем показаны как в шестнадцатиричном виде, так и в двоичном. Если в этом окне ввести новое название аппаратного адаптера и нажать кнопку Save, то оно добавится к списку адаптеров, находящемуся в выпадающем списке комбобокса, а также параметры этого адаптера сохранятся в файле инициализации данной программы. Теперь пользователь имеет возможность выбрать из списка требуемый ему адаптер и, нажав на кнопку Load, произвести загрузку в программу новых параметров аппаратного адаптера. Кнопка ShowMyCable позволяет вывести на окно просмотра текущие параметры аппаратного адаптера. Внешний вид этой страницы показан на рис. 31. Окно просмотра данных находится на первой странице закладки. На это окно будут выводиться все данные, полученные программой, а также данные, выводимые самой программой в процессе работы. Внешний вид этой страницы показан на рис. 32.

Во-вторых, третья страница блокнота (рис. 33) предназначена для обработки BSDL-файлов и хранения результатов этой обработки. Пользователь указывает программе BSDL файл той микросхемы, с которой он хочет работать. Программа выделяет из этого файла коды команд сканирования и их разрядность. Выделенные параметры пользователь имеет возможность сохранить, и далее, при запуске программы, все параметры будут подгружаться автоматически.

Кроме тех элементов управления и индикации, о которых было сказано выше, на данной странице присутствует еще один очень важный элемент управления — это комбобокс, который позволяет пользователю выбрать уровень мониторинга процесса тестирования.

Для того чтобы пользователь мог подробно изучить обмен данными с портом JTAG, предусмотрен мониторинг «на уровне битов порта» (рис. 34). Здесь приведена только небольшая часть трассировки команды Get ID.

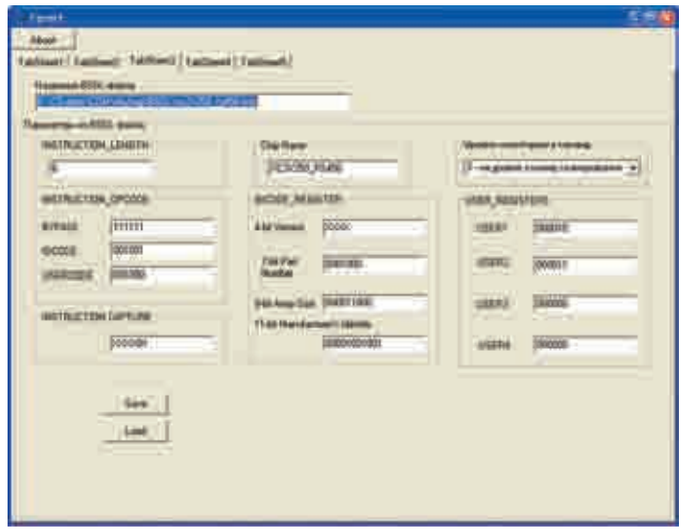

Рис 33. Окно для индикации параметров, выделенных из BSDL-файлов и комбобокса, задающего уровень мониторинга программы

```
тест — 1 Get ID
_WriteCmd — переход в TAP_TLR, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_TLR, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_TLR, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_TLR, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_TLR, TDI — 0, TMS — 1, DataBit — 54
переход в TAP_TLR, tms — 0x1F, tdi — 0x00, N — 0x05
_WriteCmd — переход в TAP_SI, TDI — 0, TMS — 0, DataBit — 50
_WriteCmd — переход в TAP_SI, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_SI, TDI — 0, TMS — 1, DataBit — 54
_WriteCmd — переход в TAP_SI, TDI — 0, TMS — 0, DataBit — 50
_WriteCmd — переход в TAP_SI, TDI — 0, TMS — 0, DataBit — 50
переход в TAP_SI, tms — 0x06, tdi — 0x00, N — 0x05
WriteCmd — запись команды в последнем байте, TDI — 1, TMS — 0, DataBit — 53, Read TDO — 0
WriteCmd — запись команды в последнем байте, TDI — 0, TMS — 0, DataBit — 52, Read TDO — 1
WriteCmd — запись команды в последнем байте, TDI — 0, TMS — 0, DataBit — 52, Read TDO — 1
WriteCmd — запись команды в последнем байте, TDI — 1, TMS — 0, DataBit — 53, Read TDO — 1
WriteCmd — запись команды в последнем байте, TDI — 0, TMS — 0, DataBit — 52, Read TDO — 1
запись команды в последнем байте, tms — 0x00, tdi — 0x09, N — 0x05, RdWriteTMS_SetTDI strScanIn — 11110
WriteCmd — запись последнего бита команды и переход в TapEndState, TDI — 0, TMS — 1, DataBit — 56, Read TDO — 1
WriteCmd — запись последнего бита команды и переход в TapEndState, TDI — 0, TMS — 1, DataBit — 56, Read TDO — 1
WriteCmd — запись последнего бита команды и переход в TapEndState, TDI — 0, TMS — 0, DataBit — 52, Read TDO — 1
WriteCmd — запись последнего бита команды и переход в TapEndState, TDI — 0, TMS — 1, DataBit — 56, Read TDO — 1
WriteCmd — запись последнего бита команды и переход в TapEndState, TDI — 0, TMS — 0, DataBit — 52, Read TDO — 1
запись последнего бита команды и переход в TapEndState, tms — 0x0B, tdi — 0x00, N — 0x05, RdWriteTMS_SetTDI strScanIn — 11111
cur_state — 0x03
```

Рис. 34. Пример работы в режиме «на уровне битов порта»

```
тест — 1 Get ID
переход в TAP_TLR, tms — 0x1F, tdi — 0x00, N — 0x05
переход в TAP_SI, tms — 0x06, tdi — 0x00, N — 0x05
запись команды в последнем байте, tms — 0x00, tdi — 0x09, N — 0x05, RdWriteTMS_SetTDI strScanIn — 10101
запись последнего бита команды и переход в TapEndState, tms — 0x0B, tdi — 0x00, N — 0x05, RdWriteTMS_SetTDI strScanIn — 11111
cur_state — 0x03
по переходам — 0x02
String IrScan — 110101
прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00100111
прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00000001
прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 10000010
прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00000011
прием последнего бита и переход в TapEndState, tms — 0x03, tdi — 0x00, N — 0x03, RdWriteTMS_SetTDI strScanIn — 110
cur_state — 0x08
по переходам — 0x01
String DrScan — 00000001110000010000000100100111
TapIrScan — 110101
TapDrScan — 00000001110000010000000010010011
```

Рис. 35. Пример работы в режиме «на уровне битов»

Для того чтобы перейти на мониторинг «ступенью выше», предусмотрен режим «на уровне битов» (рис. 35). Мониторинг также установлен «на уровне битов». На этом рисунке приводится трассировка команды Get ID.

Более высокий уровень — «на уровне команд сканирования» (рис. 36). Мониторинг проводится «на уровне команд сканирования». На этом рисунке приводится трассировка команды Get ID.

```
cur_state — 0x03
по переходам — 0x02
String IrScan — 110101
cur_state — 0x08
по переходам — 0x01
String DrScan — 00000001110000010000000100100111
TapIrScan — 110101
TapDrScan — 00000001110000010000000010010011
```

Рис. 36. Пример работы в режиме «на уровне команд сканирования»

Еще более высокая «ступень» — «на уровне макрокоманд». В этом режиме на мониторинг выведется только следующая строка «тест — 1 Get ID». В этой строке будет указан лишь текущий номер теста и выполненная команда. Таким образом, можно выбрать требуемый уровень детализации мониторинга, вплоть до полной отмены вывода информации на окно мониторинга, что необходимо при работе с логическими анализаторами или другими аппаратными средствами, критичными ко времени исполнения тестов.

В четвертом окне (рис. 37) пользователь формирует тесты, и затем они выполняются. Слева находится таблица, в которую помещаются команды и их параметры, а справа расположены органы управления, необходимые для работы с этим окном. В рамке «Команда–данные» расположены два комбобокса, при помощи верхнего можно выбрать тип команды, а при помощи нижнего — конкретное действие. Например, выбрав на верхнем комбобоксе тип команды — TapIrScan, в нижний загружается список из следующих действий: TapIrScan, Addr=USER1, Addr=USER2, Addr=Bypass, Addr=ID, Addr=USERCODE. Если выбрана радиокнопка Set, то по двойному клику по таблице данные из нижнего комбобокса будут записаны в таблицу. Если же выбрана радиокнопка Get, то данные из таблицы будут записаны в этот же комбобокс. Кнопка +5 Rows добавляет к таблице еще 5 строк.

Непосредственно под рамкой «Команда–данные» расположены две кнопки, при помощи которых пользователь может загрузить или сохранить набранный тест.

Теперь рассмотрим те элементы, при помощи которых пользователь управляет ходом выполнения теста. Для того чтобы указать программе, с какой строки необходимо выполнять тест, пользователь должен навести указатель мыши на требуемую строку таблицы и произвести один клик. При этом строка будет выделена синим цветом. Если теперь пользователь нажмет кнопку Step, то программа выполнит тот тест в таблице, который выделен синим цветом. После выполнения теста будет выделена синим цветом следующая строка в таблице. При повторном нажатии на кнопку Step будет выполнен следующий шаг. И если работа ведется в первых трех строках таблицы, то будет произведено выделение следующей строки, то есть, можно сказать, что синяя полоска сместится вниз на одну

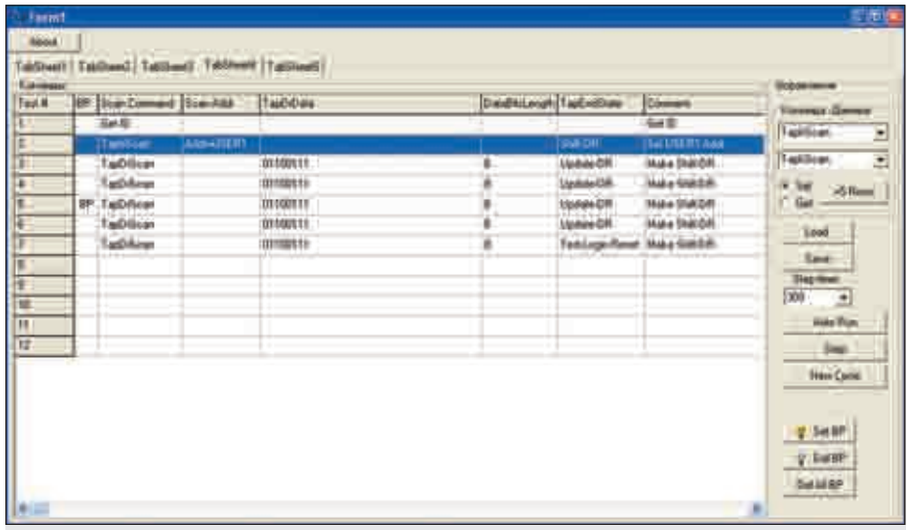

Рис. 37. Окно программы, в котором пользователь формирует тесты

позицию. А вот начиная с четвертой строки таблицы, кроме выделения следующей строки, одновременно вверх на одну строку сдвигается сама таблица. Это сделано для того, чтобы пользователь мог видеть, какие тесты уже были выполнены и какие сейчас предстоит выполнить. Если пользователь хочет изменить порядок выполнения тестов, например, вернуться на несколько тестов назад или пропустить несколько тестов, то ему для этого достаточно просто произвести выделение (см. выше) той строки, с которой он хочет продолжить работу.

Кнопка New Cycle переводит выделение на первую строку теста и обнуляет системные переменные так, чтобы пользователь мог начать новый цикл тестирования.

В центре рамки «Управление» расположены кнопки, позволяющие запустить выполнение тестов в автоматическом режиме. Для этого пользователь имеет возможность выбрать время для паузы при выполнении тестов. Далее, при нажатии на кнопку Auto Run программа запускает таймер, который имитирует нажатие кнопки Step. На кнопке Auto Run изменится надпись, появится Stop, и нажатие на эту кнопку приводит к остановке тестирования после выполнения текущей строки теста. Если же выполнение тестов завершается, то программа вновь меняет надпись на кнопке (Stop на Auto Run) и после этого выводит сообщение о том, что тестирование завершено на такой-то строке. Кроме этого, аналогичная надпись выводится в окне мониторинга.

Но для работы с большими по объему тестами этого может быть мало. Для удобства пользователя сделаны еще три кнопки. Set BP — установить точку останова, Del BP — убрать точку останова и Del All BP — убрать все точки останова. Для того чтобы установить точку останова, необходимо выделить нужную строку и нажать кнопку Set BP. Аналогичные действия надо выполнить, чтобы снять точку останова. На рис. 37 в строке № 5 показана установленная точка останова. Кнопка Del All BP позволяет быстро очистить весь тест от точек останова. Точки останова вызывают останов тестирования только при автоматическом тестировании. При этом выделится та строка, которая еще не выполнялась.

Для облегчения составления тестов были использованы некоторые идеи Доменика Раша (Dominic Rath) из его открытого проекта Open On-Chip Debugger (**w ww.openocd.berlios.d e/web**). Также учитывался опыт работы с микросхемами фирмы FTDI и их FTCJTAG DLL. Все это привело к тому, что программа выполнена так, что она формирует сигналы для управления TAP-контроллером автоматически. Она автоматически выполняет и все действия, необходимые для перевода TAP-контроллера из одного состояния в другое.

Формирование тестов упрощено до предела. Поэтому пользователю достаточно лишь указать то действие, которое он хочет выполнить, то есть прочитать ID микросхемы, выполнить команду сканирования регистра команд (IrScan) или команду сканирования регистра данных (DrScan). Для команды IrScan пользователь должен ввести адрес, по которому должно производиться сканирование. Этот адрес будет загружен в регистр адресов TAP-контроллера микросхемы, так что последующее обращение по команде DrScan будет работать с данными, находящимися по указанному адресу. Программа позволяет работать не с абсолютными адресами, а с символьными именами адресов регистров. Это сделано, чтобы упростить процедуру ввода адресов, а для этого необходимо в верхнем комбобоксе, который находится в рамке «Команда–данные», из выпадающего меню выбрать пункт TapIrScan. Тогда в нижний комбобокс загрузится список адресов TAP-контроллера, и именно тех адресов, которые были выделены из BSDL-файла, описывающего микросхему пользователя: Addr=USER1, Addr=USER2, Addr=Bypass, Addr=ID, Addr=USERCODE. В том случае, если в BSDL-файле не будет обнаружено адресов регистров под названиями USER1 и USER2, эти адреса пользователь может ввести вручную в окне номер 3 и произвести сохранение введенной информации.

Если для команды IrScan необходимо указать адрес, то, соответственно, для команды DrScan необходимо только указать данные, которые будут загружаться в микросхему. Также необходимо указать «длину» слова данных. Конечно, если записывать данные в виде строки битов — «01100111», то длина такой строки вычисляется достаточно просто. Но ведь для некоторых проектов длина регистра в проекте пользователя, находящемся в FPGA, может быть достаточно большой, поэтому, в таком случае, для получения более компактной формы записи будет удобней пользоваться шестнадцатиричной формой представления. А поскольку длина регистра пользователя может быть и не кратна 8, то тут и возникает потребность во введении параметра, указывающего на длину данных.

Еще один важный параметр, который необходимо ввести в таблицу, — это то состояние TAP-контроллера, которое он должен принять по окончании выполнения данного теста. Программа имеет внутреннюю переменную, хранящую номер текущего состояния TAP-контроллера. Программа проверяет, можно ли начать выполнять команду из текущего состояния. Если определено, что выполнение команды из текущего состояния невозможно, программа сначала выполняет переход в то состояние TAP-контроллера, которое будет являться исходным для выполнения данной команды. После выполнения команды программа аналогичным образом выполнит переход в то состояние TAP-контроллера, которое будет задано пользователем в таблице как конечное состояние для завершения теста. Таким образом, для большинства команд IrScan конечным состоянием автомата TAP-контроллера можно принять состояние Shift-DR. Для того чтобы упростить эту процедуру, необходимо в верхнем комбобоксе, который находится в рамке «Команда–данные», из выпадающего меню выбрать пункт TapEndState. Тогда в нижний комбобокс загрузится список состояний TAP-контроллера, которые предлагаются пользователю: Run-Test/Idle, Test-Logic-Reset, Shift-IR, Shift-DR, Update-DR. Что же касается команд DrScan, то для них можно выбрать конечным состоянием автомата TAP-контроллера состояние Update-DR — в том случае, если за этим тестом последует еще один тест с командой DrScan или при последующей остановке тестирования, может быть выбрано состояние Run-Test/Idle. Если же этот тест последний и автомат надо перевести в выключенное состояние, то необходимо выбрать состояние Test-Logic-Reset.

Для того чтобы еще более упростить формирование тестов, можно модернизировать

Рис. 38. Пример реализации фрагмента программы, показывающего соединение нескольких микросхем в цепочку

программу так, чтобы было можно устанавливать состояние TAP-контроллера, которое он должен принять по окончании выполнения данного теста, автоматически. Для этого всего лишь надо проанализировать таблицу и выяснить, какой будет следующий тест. Если текущий тест последний, то это значит, что его конечным состоянием должно быть Test-Logic-Reset. Если следующий тест — IrScan, то, естественно, надо делать переход в состояние Shift-IR. И, аналогично, если следующий тест — DrScan, то, точно так же, надо делать переход в состояние Shift-DR.

И последний, четвертый пункт. Для того чтобы пользователь смог развивать данную программу, в пятом окне выполнен фрагмент, предназначенный для визуального отображения характеристик нескольких микросхем, включенных в JTAG-цепочку (рис. 38). Нажав на кнопку «Создать», пользователь создает на форме набор кнопок, соответствующий числу, введенному пользователем. При прохождении указателя мыши по этим кнопкам программа выдает подсказку (хинт), текст которой соответствует надписи на кнопке.

Поскольку среди команд тестирования есть команда определения числа микросхем в цепочке, то пользователь сам сможет доработать программу, связав определение числа микросхем с последующим рисованием соответствующего числа кнопок.

И в заключение необходимо сказать еще несколько слов о том, что данная программа не представляет собой завершенный коммерческий продукт. Она предназначена только для того, чтобы пользователь смог сделать самые первые шаги. Поэтому были предприняты меры по упрощению составления тестов. Программа не была рассчитана на работу с граничным сканированием. Поэтому здесь реализован только режим IrScan, но нет команд Sample/Preload. Однако пользователь может сам доработать данное программное обеспечение до требуемых ему задач.

Но здесь необходимо обратить внимание читателя на следующий момент. Все фирмы, разработчики средств тестирования, всегда пишут о том, что JTAG-тестирование — это передовая технология, и тут все просто и нет нерешенных проблем. Да, для компаний, использующих данную технологию десятилетиями, возможно, это и так. Но для тех, кто только начинает работать с технологией использования порта JTAG, необходимы совершенно другие «правила игры».

Помните о том, что технология работы с портом JTAG — это не совсем безобидно и легко. Да, без сомнения, эта технология значительно облегчает жизнь пользователя. Но только при одном условии: порт JTAG и все, что с ним связано, — аппаратные адаптеры, кабели, питание и пр. — все это должно работать безотказно! Посмотрите в начало этой статьи. А именно туда, где было написано о последовательном продвижении тестов от центра к периферии. Всегда помните о том, что хотя ваше тестирующее оборудование исправно и проверено, но сама цепь JTAG, та цепь, по которой передаются тестовые сигналы, находится на проверяемой плате, и она, эта цепь, тоже может быть источником ошибок. Если цепочка бит, сдвигаемая в порт, может достигать «в длину» до десятков тысяч бит, то при сбое 1 бит на 1000 бит вы никогда не получите правильного результата! Мало того, в результате сдвига неверной последовательности вы можете случайным образом выдать воздействие на внутренние регистры микросхемы или на ее выводы. А это, в свою очередь, может привести к отказу самой микросхемы или той периферии, которая управляется данной микросхемой. Поэтому необходимо принимать определенные меры безопасности при работе с портом JTAG.

Как работает программа?

Блок-схема с названиями файлов, входящих в программу приведена на рис. 39.

В левой части рисунка показаны файлы, сгруппированные по выполняемым ими функциям. Это загрузка и сохранение данных инициализации и данных для таблиц, работа со строковыми функциями, выполнение функций, необходимых для инициализации работы всего приложения. С правой стороны показаны взаимосвязи между файлами, которые используются при работе с портом. В том случае, если возникнет необходимость подключить к программе другой аппаратный адаптер, например, работающий через USB, программа может быть легко доработана путем введения в нее еще одного класса, выполняющего аналогичные функции, но уже через USB-порт.

Что касается работы нижних уровней программы, работающих с портом, то здесь особо описывать нечего. В качестве драйвера порта был использован GiveIO. Управление битами порта ведется при помощи набора масок, которые вводятся пользователем. Этот набор масок ассоциируется с конкретным типом аппаратного адаптера. Таким образом, программа может быть легко настроена на различные реализации аппаратных адаптеров. На уровне класса MyTAP программа выполняет следующие функции:

```
String MyTAP::RdWriteTMS_SetTDI(char tms, char tdi, char N,
String strComment, int MonitorLevel).
```

Такая функция производит сдвиг до восьми бит данных в порт и читает, соответственно, до восьми бит данных из порта. Функции передается параметр, указывающий, сколько бит надо обработать, а также строка комментариев и параметр, определяющий уровень мониторинга. Уровни мониторинга и примеры работы программы с ними были приведены выше.

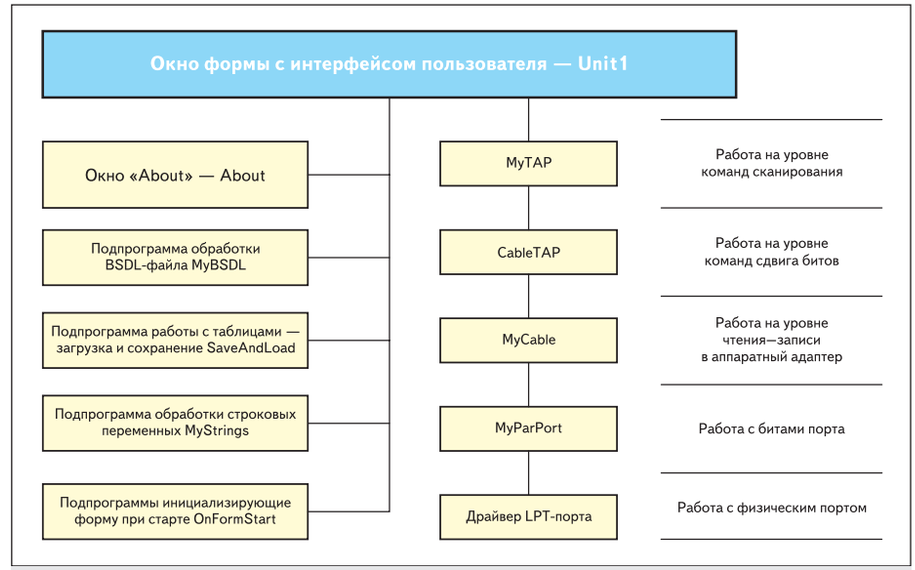

Рис. 39. Блок-схема программы Jtag Tool

Теперь давайте рассмотрим то, как программа выполняет управление TAP-контроллером в микросхеме пользователя. Для того чтобы выполнить переходы из одного состояния автомата в другое, в программе предусмотрен набор переменных и таблица, определяющая, какие и сколько бит должно быть выдано в порт. Всего TAP-контроллер имеет 16 состояний. Но переходы возможны только из определенных состояний и лишь в другие определенные диаграммой переходов состояния. Более того, для упрощения было принято, что в программе не будет использован ряд переходов и ряд состояний. Это позволяет существенно сократить таблицу переходов. Но, вместе с тем, это требует предварительной проверки. Пользователю разрешено выполнять переходы в следующие состояния: Test-Logic-Reset, Run-Test/Idle, Shift-DR, Pause-DR, SI — Shift-IR, PI — Pause-IR, Update-DR, Update-IR. После проверки того, что требуемое пользователю состояние разрешено, программа извлекает из таблицы число импульсов, которые необходимо выдать в порт, и данные, представляющие собой закодированную последовательность из нулей и единиц, которые будут выданы в порт JTAG.

Теперь вкратце опишем последовательность шагов при начале тестирования изделия. Будем считать, что сам компьютер, который выполняет тестирование, исправен. Далее переходим к проверке его LPT-порта. Можно напомнить и о том, что в LPT-порт можно записать данные и считать их. После правильного выполнения этих процедур мы точно будем знать, что порт у нас есть и он управляется компьютером. Теперь мы можем перенести «точку опоры» на один шаг ближе к проекту пользователя, а именно в сам LPT-порт. Теперь мы считаем, что порт точно исправен и, если не будут выполняться следующие тесты, то это значит, что не выполняет свои функции оборудование, находящееся за LPT-портом. Проверим наличие аппаратного адаптера. Для этого в аппаратном адаптере обычно используется перемычка, передающая сигнал с одного контакта порта на другой. Если этот тест проходит, то это значит, что аппаратный адаптер подключен к порту. В адаптерах фирмы Altera аналогичная перемычка была сделана не напрямую с контакта на контакт, а через микросхему, используемую для передачи сигналов порта. После выполнения тестирования по этому каналу передачи тестовых данных можно было сделать вывод о том, что микросхема исправна и на нее подано питающее напряжение. В аппаратном адаптере фирмы Xilinx для проверки питающего напряжения используется отдельный вход LPT-порта. Вся хитрость в проверке питающего напряжения заключается в том, что перед проверкой напряжения необходимо записать в этот бит LPT-порта 0. Если эти тесты выполняются, то теперь «точку опоры» можно перенести еще дальше — непосредственно на порт JTAG, находящийся на плате пользователя. Теперь мы должны определить то, насколько качественно выполнена сама JTAG-цепочка на плате. Для этого у программы в файле инициализации есть два параметра — максимальное число микросхем в цепочке и тестовая последовательность из нулей и единиц. По умолчанию или по сбросу все микросхемы должны находиться в режиме Bypass. Но для того чтобы надежно провести тестирование, выполним следующие действия. Подадим на вход TMS проверяемой микросхемы сигнал уровня 1 и выдадим 5 тактовых импульсов TCK. Это переведет TAP-контроллер в состояние Run-Test/Idle, что эквивалентно сбросу контроллера в исходное состояние. Затем отметим, что код команды Bypass состоит только из одних единиц. А это значит, что, выдавая последовательность единиц по команде IrScan, мы переведем все микросхемы в JTAG-цепочке в режим Bypass. Представим, что длина команды Bypass в битах будет не более 20 бит (реальное значение — от 4 до 11 бит). Максимальное число микросхем на плате пользователя мы получим из параметра MaxChipNumber. Для того чтобы гарантированно перевести все микросхемы в такое состояние, программа должна выдать по команде IrScan последовательность из 20* MaxChipNumber единиц. Теперь можно проверять число микросхем на плате и одновременно определять качество работы JTAG-цепочки. Сформируем такую последовательность данных — справа несколько нулей, потом добавляем к ним слева тестовую последовательность из нулей и единиц и затем, опять же слева, добавляем к полученной последовательности нули. Число этих нулей должно быть не менее, чем число микросхем в JTAG-цепочке. Полученная строка должна выглядеть примерно так: 0……000000**10010101101110**. Здесь для наглядности тестовая последовательность выделена жирным шрифтом. Теперь выполним команду DrScan и пропустим нашу тестовую последовательность через проверяемую плату. И сравним полученную строку с той строкой — тестовой последовательностью, которую мы сдвигали. Если вся JTAG-цепочка работает исправно, то в полученной строке будет содержаться фрагмент из нулей и единиц, соответствующий тестовой последовательности. Причем этот фрагмент будет смещен вправо (отставать) на число позиций, соответствующее числу микросхем в цепочке. Это происходит потому, что в режиме Bypass каждая микросхема, находящаяся в цепи JTAG, представляет собой только один триггер, и на нем информация, поступающая на вход TDI, задерживается только на один такт и передается далее на выход TDO. Вот таким способом можно программно определить число микросхем в JTAG-цепочке. Далее этот тест необходимо провести несколько раз и проверить результаты его работы. Если результаты совпадают, то это значит, что цепь TDI — TDO и TCK — работает исправно. Далее для того чтобы убедиться, что все работает исправно, можно провести чтение ID у микросхем. Если мы уже убедились, что данные ID читаются правильно, то можно утверждать, что чтение информации из микросхемы работает и на плате припаяна именно та микросхема, которую мы и ожидали увидеть.

Если же все тесты, описанные выше, проходят без ошибок, то можно считать, что порт JTAG работает исправно. Вот только теперь можно приступать к тестированию проекта пользователя. Итак, переносим «точку опоры» еще на один шаг ближе к проекту пользователя.

Проектирование аппаратной части

Что касается проекта в «железе», то здесь основное внимание будет уделено именно стыку проекта пользователя с портом JTAG. Что же касается узла, производящего отладку микроконтроллеров, то о нем речь пойдет позже. Итак, хост, JTAG-порт и связь с проектом пользователя.

Теперь рассмотрим рис. 40, где данные компоненты представлены в виде блок-схем. Начнем с самого понятного разработчику, а именно с той части, которая выделена голубым цветом и представляет собой логику отладочного блока пользователя. Эта логика находится в FPGA, там же, где и проект пользователя. И с этой логикой, на первый взгляд, все понятно. Ее можно «написать» самому, отсимулировать и, затем, скомпилировать. А вот JTAG-порт — это же аппаратная часть чипа, и детального описания ее работы нет. Есть только очень приблизительное описание работы порта. Ну а хост — это немного программ и аппаратный LPT или USB-порт. А ведь инженер-аппаратчик привык видеть все в виде осциллограмм или их симуляций. Вот поэтому автор предлагает воспользоваться еще одним небольшим проектом, сделанным им для данной статьи. Это симуляция описанных выше устройств, выполненная в среде ModelSim и написанная на языке Verilog. Основная цель данного проекта — помочь читателю увидеть последовательность смены состояний автомата TAP-контроллера в виде осциллограмм. Этот проект будет являться базой для «отладочного проекта».

Основой проекта является именно узел TAP-контроллера, который вместе с регистрами и другими дополнительными узлами образует модель JTAG-порта. Для того чтобы выдавать на JTAG-порт последовательность импульсов, в проекте используется блок tester. Этот узел выполнен как task. Для упрощения формирования последовательностей импульсов, выдаваемых узлом tester, воспользуемся результатами работы программного инструмента, описанного выше. Загрузим или сформируем тест чтения ID из микросхемы, установим мониторинг «на уровне битов» и запустим тест на выпол-

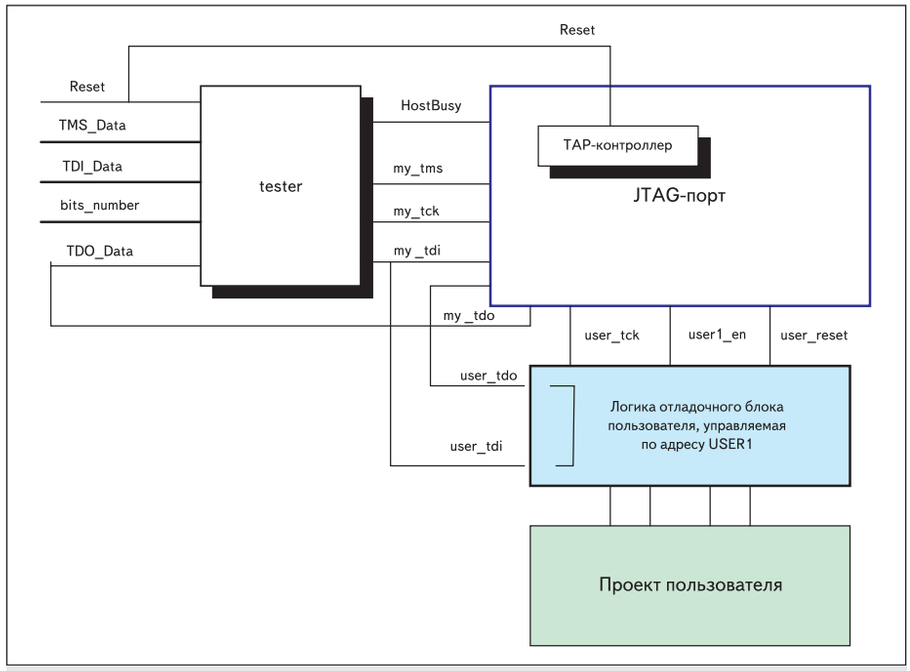

Рис. 40. Блок-схема проекта симуляции работы хоста, JTAG-порта и логики пользователя, выполненная в среде ModelSim и написанная на языке Verilog

нение. Даже при выключенном аппаратном адаптере мы получим в окне мониторинга трассировку теста. Ее можно использовать, чтобы задать параметры для task. На рис. 41 приведен фрагмент кода для симуляции режима чтения ID из микросхемы. Закомментированные строки, выделенные зеленым цветом, — это часть трассировки, взятая как результат работы программного инструмента. Строки черного цвета — то, что будет выполнять симулятор. Для упрощения чтение ID выполняется не за 32 такта, а только за 8.

Теперь более подробно можно рассмотреть структуру самого JTAG-порта. На рис. 42 приведена его блок-схема.

В состав порта входят TAP-контроллер, Регистр Инструкций, Декодер Адреса Инструкций, регистры Bypass, IDCODE, BoundaryScan и другие регистры. Кроме того, в состав порта входят два мультиплексора выходных данных и вспомогательный узел, формирующий сброс. На регистры и TAP-контроллер поступают тактовые импульсы TCK. Кроме того, на TAP-контроллер поступает сигнал TMS. Входные данные — TDI — поступают на регистры. Теперь давайте посмотрим на то, как это работает. (Для справки читатель может посмотреть раздел статьи «Переходы, которые необходимо выполнить при работе с командами», см. КиТ № 10`2006.) Если сигнал

```
// Get ID
//переход в TAP_TLR, tms — 0x1F, tdi — 0x00, N — 0x05
tester(32'h00FF, 32'h0000, 8, wTDO, RST); — это сделано, чтобы увидеть сброс «user_reset»

//переход в TAP_SI, tms — 0x06, tdi — 0x00, N — 0x05
tester(32'h0006, 32'h0000, 5, wTDO, RST);

//запись команды в последнем байте, tms — 0x00, tdi — 0x09, N — 0x05, RdWriteTMS_SetTDI strScanIn — 10101
tester(32'h0000, 32'h0009, 5, wTDO, RST);

//запись последнего бита команды и переход в TapEndState, tms — 0x0B, tdi — 0x00, N — 0x05, RdWriteTMS_SetTDI strScanIn — 11111
tester(32'h000B, 32'h0000, 5, wTDO, RST);

//cur_state — 0x03
//по переходам — 0x02
//String IrScan — 110101
//прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00100111
tester(32'h0000, 32'h0000, 8, wTDO, RST);
//прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00000001
//прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 10000010
//прием данных в байтах, tms — 0x00, tdi — 0x00, N — 0x08, RdWriteTMS_SetTDI strScanIn — 00000011
//прием последнего бита и переход в TapEndState, tms — 0x03, tdi — 0x00, N — 0x03, RdWriteTMS_SetTDI strScanIn — 110
tester(32'h0003, 32'h0000, 3, wTDO, RST);
//cur_state — 0x08
//по переходам — 0x01
```

Рис. 41. Фрагмент кода для симуляции режима чтения ID из микросхемы. Закомментированные строки — зеленого цвета. Это часть трассировки, взятая как результат работы программного инструмента. Строки черного цвета — то, что будет выполнять симулятор

TMS удерживать на входе более пяти тактов синхрочастоты, то TAP-контроллер выполнит сброс своей логики и будет находиться в состоянии Test-Logic-Reset. Далее, поступающие на TAP-контроллер импульсы TCK и сигнал TMS переведут его в другие состояния. При получении инструкции в состоянии Shift_IR, TAP-контроллер пройдет через состояние Update_IR. Во время сдвига инструкции из JTAG-порта, через его выход TDO, будут выдвигаться данные. При этом полученная инструкция будет сохранена и ее код поступит на дешифратор инструкций. Он выберет, какой из регистров будет скоммутирован на выход TDO. В этот же регистр будут грузиться данные при работе TAP-контроллера в состоянии Shift_DR. Результат работы симулятора при чтении ID представлен на рис. 43. Желтыми цифрами в верхней части рисунка обозначены периоды времени, соответствующие «task'ам», которые на рис. 41 показаны черным цветом. Желтым шрифтом на поле симуляции обозначены состояния TAP-контроллера. Последовательность переходов из одного состояния в другое показана желтыми стрелками.

Рассмотрим фазы сдвига инструкции, которые на рис. 43 показаны голубыми цифрами. Импульсы, отмеченные цифрами 1–5, соответствуют переходам в состояние Shift_IR. Само состояние Shift_IR обозначено цифрой 6. По окончании этого состояния автомат переходит через состояния 6, в состояния, обозначенные как 7–11, к состоянию Shift_DR. Состояние Shift_DR обозначено числом 12. По окончании работы с данными происходит переход автомата через состояние 13 в состояние Run_Test/Idle.

Голубым числом 14 показано формирование импульса «Сброс», который вырабатывает счетчик числа импульсов на входе TMS. При достижении заданного уровня, а именно пяти импульсов, счетчик останавливается и выдает на свой выход сигнал «Сброс». Этот сигнал используется логикой сопряжения JTAG-порта с логикой пользователя.

Голубым числом 15 показано формирование импульса разрешения выборки, котрый вырабатывается в соответствии с тем адресом, который был загружен в TAP-контроллер по команде IR-Scan. Часть из этих сигналов также используются логикой сопряжения JTAG-порта с логикой пользователя.

Для упрощения симуляции предлагаемый автором проект не содержит «логики пользователя». К ней автор относит те сдвиговые регистры и регистры хранения информации, которые необходимы пользователю для работы его отладочного узла. Кроме собственно узлов, преобразующих последовательный интерфейс в параллельный, пользователю еще понадобятся дополнительные узлы привязки асинхронного (с точки зрения системной синхрочастоты в проекте пользователя) проекта пользователя к своей, системной синхрочастоте. То есть к той синхрочастоте,

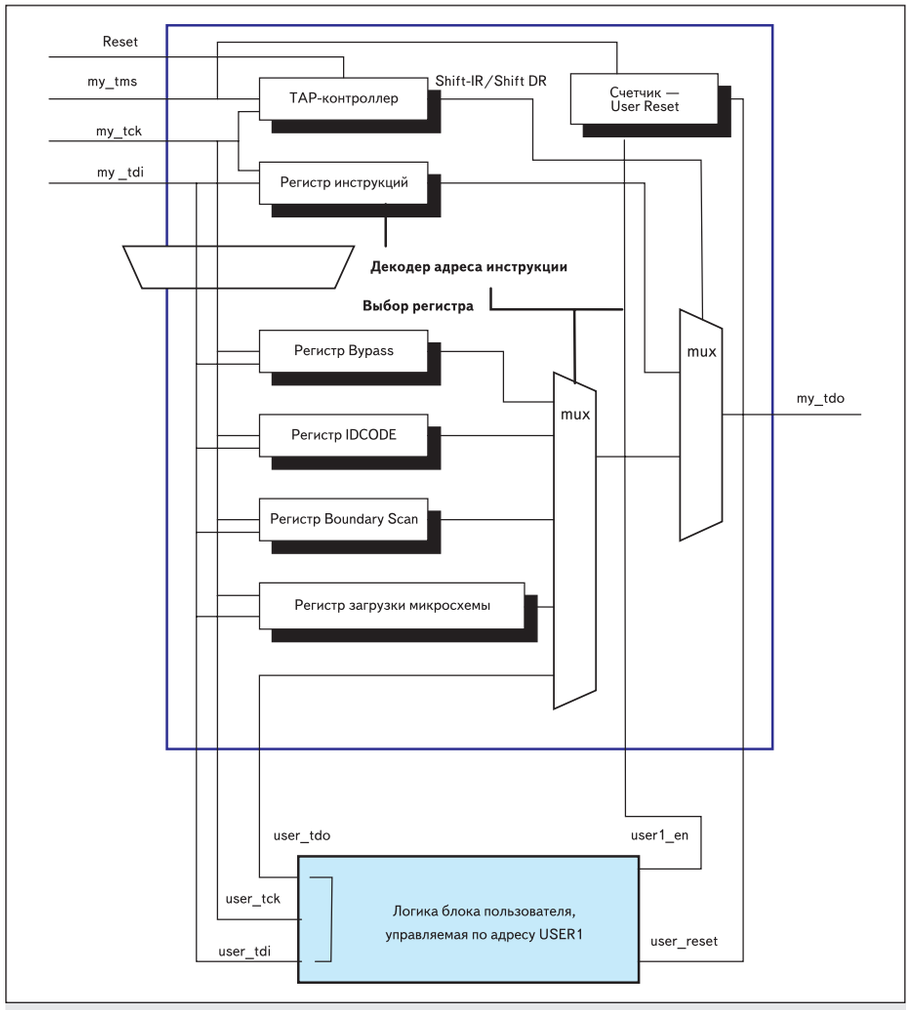

Рис. 42. Блок-схема JTAG-порта и логики пользователя

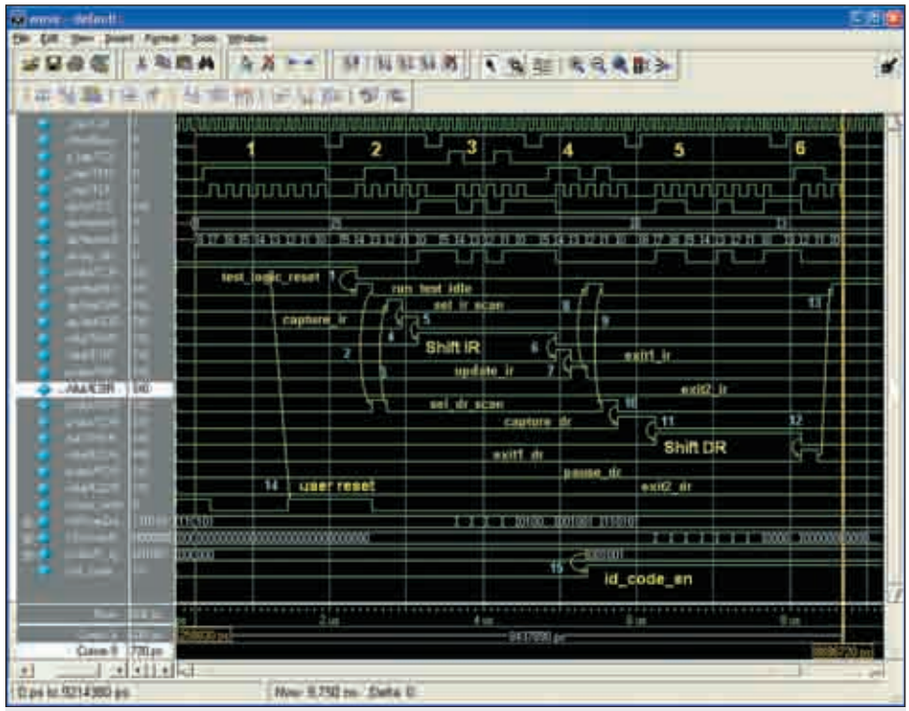

Рис. 43. Симуляция работы JTAG-порта при чтении ID из микросхемы

на которой работает весь остальной проект пользователя.

Проект в FPGA

Прежде чем предлагать что-то свое, необходимо дать читателям познакомиться с тем, что уже сделано и уже доступно для самостоятельного изучения.

Вот материал, представленный фирмой 3S. Он представляет интерес, в первую очередь, для «любителей» продукции фирмы Xilinx.

1. Using the JTAG Interface as a General-Purpose Communication Port. by Derek Wallace Silicon and Software Systems. **w ww.s3group.c om**. Xcell Online, Second Quarter 2005;
2. GNAT PROJECT. GNAT Example for Xilinx Spartan™-3 Starter Kit. Document Number: INT2004_57b_DD01. Silicon & Software Systems Ltd. **w ww.s3group.c om**;
3. General-purpose Native jtAg Tester (GNAT), **ww w.s3group.c om/design_expertise/fpga/**.

И, конечно, надо сослаться на материалы, представленные фирмой Altera.

Учебные проекты по Virtual JTAG находятся по адресу:

- **w ww.altera.c om/literature/ug/ug_virtual_jtag_design_example_1.zip**;
- **w ww.altera.c om/literature/ug/ug_virtual_jtag_design_example_2.zip**.

Продолжение следует

Фраза, конечно же, банальная, но все же. Хорошо было бы сначала сделать все проекты, написать все фрагменты статьи, отредактировать все картинки и только потом отдавать все в печать. Но жизнь показывает, что так работать невозможно — получается слишком большое отставание от жизни. Поэтому приходится делать все одновременно. И статью, и проекты к ней. Часть проектов уже выполнена — например, проект в Verilog'e и софтовый инструмент. Часть еще предстоит сделать. Поэтому, как и предупреждал автор в начале статьи, возможны некоторые задержки с выполнением остальных проектов. «Марафон» получился длительный, поэтому прошу вас, уважаемый читатель, отнестись к этому с пониманием. Надеюсь, что все задуманное удастся выполнить, а потому — продолжение следует…

Литература

9. **w ww.amontec.co m** JTAG Interface: Common Pinouts amt_ann003 (v1.1) Application Note.
10. EIA/JEP106, JEDEC Publication 106, Standard Manufacturer's Identification Code.
11. UG-BBII81204-1.1, **w ww.altera.c om**
12. Chameleon POD Block Diagram. **w ww.amontec.c om/chm_block_diagram.sh tml**
13. Акселератор для адаптера Chameleon. **w ww.amontec.c om/jtag_accelerator.sht ml**
14. Spartan-3E Libraries Guide for HDL Designs.
15. sld_virtual_jtag Megafunction. User Guide. Altera Corporation. June 2006.