KiT#49(5).qxd 5/28/2005 5:51 PM Page 98

Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

В представленных ранее публикациях цикла [1–5], посвященного рассмотрению встраиваемых восьмиразрядных микропроцессорных ядер семейства PicoBlaze™, реализуемых на основе ПЛИС фирмы Xilinx®, были приведены сведения о трех представителях этого семейства. После начала серийного выпуска кристаллов новой серии с архитектурой FPGA (Field Programmable Gate Array) Spartan™-3 [7] фирмой Xilinx была разработана соответствующая версия микропроцессорного ядра PicoBlaze, которая предназначена в первую очередь для применения в проектах, реализуемых на основе ПЛИС этого семейства. Кроме того новая версия ядра PicoBlaze может успешно использоваться в качестве основы для проектирования «систем-на-кристалле» (System-on-Chip), выполняемых на базе ПЛИС семейств Virtex™-II и Virtex-IIPRO™.

Валерий Зотов

walerry@km.ru

Микропроцессорное ядро PicoBlaze, предназначенное для применения в проектах, выполняемых на базе ПЛИС серий Spartan-3, Virtex-II и Virtex-IIPRO, отличается от других представителей этого семейства, рассмотренных ранее, расширенным спектром функциональных возможностей. Новая версия ядра представляет собой результат дальнейшего развития базового варианта микропроцессорного ядра PicoBlaze, реализуемого на основе кристаллов семейств Spartan™-II, Spartan-IIE, Virtex, Virtex-E. Максимальное использование архитектурных особенностей и ресурсов ПЛИС серий Spartan-3, Virtex-II и Virtex-IIPRO позволило обеспечить более высокие технические характеристики и поддержку дополнительных операций. Учитывая вышесказанное, основное внимание в статье уделяется отличиям характеристик, архитектуры и системы команд новой версии микропроцессорного ядра PicoBlaze от базового варианта, описание которого приведено в [1, 2].

Основные характеристики микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Большинство основных характеристик семейства встраиваемых микропроцессорных ядер PicoBlaze, представленных в [1], справедливы также и для версии, которая предназначена для реализации на базе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO. Поэтому далее приводятся только отличительные особенности рассматриваемой версии ядра PicoBlaze. В новой версии произошли следующие изменения (по сравнению с базовым вариантом микропроцессорного ядра [1]):

- увеличена разрядность шины адресов, которая составляет теперь 10 бит;
- разрядность шины команд расширена до 18 бит;
- в дополнение к блоку регистров общего назначения, включающему 16 восьмиразрядных регистров, предусмотрено наличие блока сверхоперативного запоминающего устройства емкостью 64 байт;
- увеличен объем встроенного ППЗУ микропрограмм, реализуемого на основе блочной памяти ПЛИС Block SelectRAM, который составляет 1024×18 разрядов;
- предусмотрена возможность формирования дополнительных признаков выполнения операции в арифметическо-логическом устройстве (АЛУ);
- модернизирована схема управления прерываниями;
- система команд расширена до 57 инструкций;
- глубина стека увеличена до 31 уровня;
- объем ресурсов кристалла, необходимых для реализации микропроцессорного ядра PicoBlaze в ПЛИС семейства Spartan-3, ограничивается 96 секциями (slices), что составляет 5% от полного объема логических ресурсов кристалла XC3S200 и менее 0,3% от логической емкости ПЛИС XC3S5000;
- возросла производительность; в зависимости от типа и класса быстродействия используемого кристалла, она достигает от 43 до 66 MIPS;
- разработан загрузчик программ *JTAG Program Loader*, позволяющий записывать новый программный код непосредственно в программную память ядра PicoBlaze через порт JTAG-интерфейса ПЛИС с помощью стандартного загрузочного кабеля;
- два варианта реализации компонентов ядра — в виде описаний на языках высокого уровня VHDL™ (VHSIC Hardware Description Language) и Verilog™.

Расширение функциональных возможностей новой версии микропроцессорного ядра привело к незначительному увеличению объема используемых ресурсов ПЛИС. По сравнению с базовым вариан-

KiT#49(5).qxd 5/28/2005 5:51 PM Page 99

том ядра PicoBlaze, объем требуемых ресурсов вырос на 26%, а по сравнению с версией, предназначенной для реализации в кристаллах семейств Virtex-II и Virtex-IIPRO [3], — на 14%.

Архитектура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Архитектура микропроцессорного ядра PicoBlaze, предназначенного для применения в кристаллах семейств Spartan-3, Virtex-II и Virtex-IIPRO, показана на рис. 1. В структурном отношении она отличается от архитектуры базовой версии, рассмотренной в [1], наличием дополнительного сверхоперативного запоминающего устройства (СОЗУ). Остальные различия проявляются на уровне отдельных структурных элементов. Эти отличия обусловлены расширением функциональных возможностей элементов в новой версии микропроцессорного ядра PicoBlaze за счет использования преимуществ архитектуры ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO.

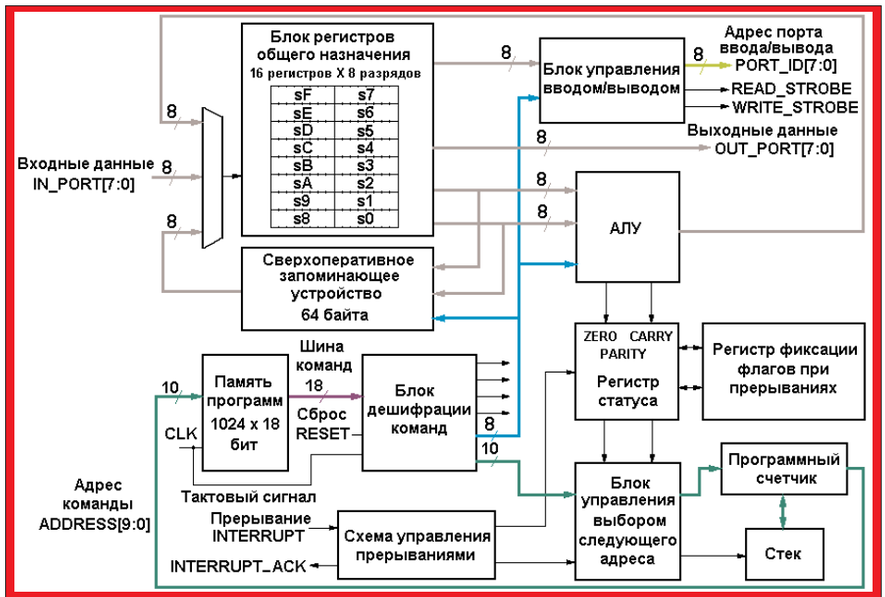

Рис. 1. Архитектура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Сверхоперативное запоминающее устройство в составе архитектуры создает предпосылки для существенного сокращения времени выполнения программы за счет исключения операций обращения к внешней (по отношению к микропроцессорному ядру) памяти. Содержимое каждого регистра общего назначения может быть передано в одну из ячеек СОЗУ и наоборот. Для обмена данными между блоком регистров общего назначения и СОЗУ предусмотрены специальные команды, время выполнения которых не превышает двух машинных циклов.

Восьмиразрядное арифметическо-логическое устройство (АЛУ) кроме логических функций, операций сложения, вычитания и сдвига, поддерживаемых базовой версией ядра, выполняет также операции поразрядного и арифметического сравнения двух операндов. В новой версии микропроцессорного ядра, кроме признаков нулевого результата ZERO Flag и переноса/заема CARRY Flag, АЛУ способно формировать признак четности/нечетности PARITY (при выполнении инструкции TEST), который отображается при помощи флага CARRY Flag.

Увеличение емкости программной памяти до 1024×18 разрядов повлекло за собой расширение шины адресов до десяти разрядов. Таким образом, верхняя граница адресного пространства памяти программ составляет 3FF. Изменение длины команд до 18 разрядов привело к соответствующему расширению шины команд.

Схема управления прерываниями, кроме внутренних управляющих сигналов, необходимых для выполнения процедуры обработки прерываний, формирует выходной сигнал interrupt_ack. Изменения в процедуре обработки прерываний коснулись также значения адреса, по которому указывается вектор процедуры обслуживания прерывания. В связи с расширением диапазона адресов программной памяти, в качестве адреса вектора прерывания, который в семействе микропроцессорных ядер PicoBlaze соответствует максимально допустимому значению адреса, используется значение 3FF.

В соответствии с увеличением ширины шины адресов, также изменена разрядность тех элементов архитектуры микропроцессорного ядра PicoBlaze, которые участвуют в процессе формирования и хранения адресов команд. Разрядность программного счетчика, регистров стека и блока управления выбором адреса следующей команды в новой версии составляет 10 бит.

Модернизация блока дешифрации команд, обусловленная необходимостью учета рассмотренных выше особенностей элементов архитектуры, привела, в частности, к изменению формата команд, поддерживаемых микропроцессорным ядром PicoBlaze, предназначенным для применения в кристаллах семейств Spartan-3, Virtex-II и Virtex-IIPRO.

Структура проекта микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Обобщенная структура варианта ядра PicoBlaze, предназначенного для применения в составе встраиваемых микропроцессорных систем, реализуемых на базе ПЛИС семейств Spartan-3,Virtex-II и Virtex-IIPRO, приведена на рис. 2. Она отличается от обобщенной структуры базовой версии ядра, рассмотренной в [1], только типом и параметрами модулей, которые входят в ее состав.

Исполнительный модуль — основа ядра — выполнен в форме компонента KCPSM3, представляющего собой макрос с относительным размещением. Описание этого макроса на языке VHDL содержится в файле kcpsm3.vhd, а на языке Verilog — в файле kcpsm3.v. В этой статье основное внимание уделяется VHDL-описанию ядра PicoBlaze и его компонентов. Интерфейс компонента KCPSM3 имеет несколько отличий по сравнению с соответствующим компонентом KCPSM микропроцессорного ядра PicoBlaze, реализуемого на базе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E [1]. Во-первых, изменилась размерность векторов, представляющих шины адресов и команд. Во-вторых, в описании интерфейса компонента KCPSM3 добавлен выходной порт interrupt_ack, который соответствует одноименному выходу сигнала подтверждения прерывания. Для декларации компонента KCPSM3 в составе VHDL-описания проектируемой системы применяются следующие выражения.

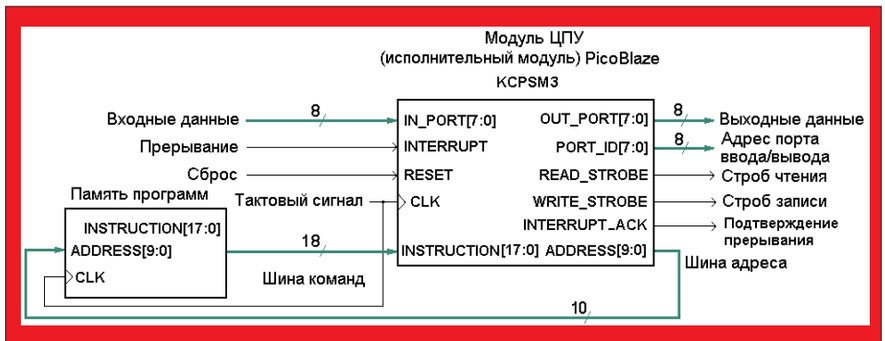

Рис. 2. Обобщенная структура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

KiT#49(5).qxd 5/28/2005 5:51 PM Page 100


```
component kcpsm3
    Port (
        address : out std_logic_vector(9 downto 0);
        instruction : in std_logic_vector(17 downto 0);
        port_id : out std_logic_vector(7 downto 0);
        write_strobe : out std_logic;
        out_port : out std_logic_vector(7 downto 0);
        read_strobe : out std_logic;
        in_port : in std_logic_vector(7 downto 0);
        interrupt : in std_logic;
        interrupt_ack : out std_logic;
        reset : in std_logic;
        clk : in std_logic
        );
    end component; (или точка?)
```

В приведенных здесь и далее выражениях декларации используется система обозначений интерфейсных цепей компонентов, подробно она описана в [1] для базового варианта микропроцессорного ядра PicoBlaze. Чтобы добавить в состав структурного описания архитектуры проектируемой системы экземпляр компонента KCPSM3, представляющего собой исполнительный модуль, необходимо воспользоваться следующей ниже конструкцией, указав вместо inst_name_processor идентификатор соответствующего экземпляра компонента.

```
inst_name_processor: kcpsm3
    port map(
        address => address_signal,
        instruction => instruction_signal,
        port_id => port_id_signal,
        write_strobe => write_strobe_signal,
        out_port => out_port_signal,
        read_strobe => read_strobe_signal,
        in_port => in_port_signal,
        interrupt => interrupt_signal,
        interrupt_ack => interrupt_ack_signal,
        reset => reset_signal,
        clk => clk_signal
    );
```

Для реализации программной памяти в новой версии микропроцессорного ядра PicoBlaze используется один модуль блочного ОЗУ Block SelectRAM, емкость которого в ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO составляет 18 кбит. Таким образом, максимальный объем загружаемой микропроцессорной программы вырос в четыре раза по сравнению с базовой версией ядра. Для описания модуля программной памяти в составе микропроцессорного ядра используется компонент с названием prog_rom, который представляет собой ППЗУ с информационной емкостью 1024×18 разрядов, реализуемое на основе однопортового блочного ОЗУ с аналогичной организацией. Выражения декларации этого компонента в составе VHDL-описания разрабатываемой системы выглядят следующим образом.

```
component prog_rom
    Port (
        address : in std_logic_vector(9 downto 0);
        instruction : out std_logic_vector(17 downto 0);
        clk : in std_logic
        );
end component;
```

Для создания экземпляра компонента, представляющего программную память, необходимо в состав VHDL-описания разрабатываемой системы включить оператор, который имеет следующий вид.

```
program: prog_rom
    port map(
        address => address_signal,
        instruction => instruction_signal,
        clk => clk_signal
    );
```

При практическом использовании выражений декларации и создания экземпляра компонента, соответствующего программной памяти, нужно вместо идентификатора prog_rom указать название компонента, совпадающее с идентификатором файла, в котором должен быть записан исходный текст загружаемой программы. Необходимость такой замены обусловлена особенностями функционирования ассемблера, который формирует описание содержимого ППЗУ программ на языках VHDL и Verilog в виде файлов с расширением vhd и v соответственно. Названия этих файлов описания содержимого программной памяти совпадают с идентификатором файла, содержащего исходный текст транслируемой программы.

Законченное описание встраиваемого микропроцессорного ядра PicoBlaze, состоящего из исполнительного модуля, сопряженного с программной памятью, выполнено в форме объекта EMBEDDED_KCPSM3. Ниже приведен полный текст описания объекта EMBEDDED_KCPSM3, который имеет ту же структуру, что и описание объекта EMBEDDED_KCPSM, представленное в [1].

```
--------------------------------------------------------------------------------
-- EMBEDDED_KCPSM3.VHD
--------------------------------------------------------------------------------
-- This file instantiates the KCPSM3 processor macro and connects the
-- program ROM.
--
-- NOTE: The name of the program ROM will probably need to be
changed to
-- reflect the name of the program (PSM) file applied to the assembler.
--------------------------------------------------------------------------------
-- Standard IEEE libraries
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--
--------------------------------------------------------------------------------
entity embedded_kcpsm3 is
    Port (      port_id : out std_logic_vector(7 downto 0);
           write_strobe : out std_logic;
          read_strobe : out std_logic;
             out_port : out std_logic_vector(7 downto 0);
              in_port : in std_logic_vector(7 downto 0);
            interrupt : in std_logic;
        interrupt_ack : out std_logic;
                reset : in std_logic;
                  clk : in std_logic);
end embedded_kcpsm3;
--------------------------------------------------------------------------------
-- Start of test achitecture
--
architecture connectivity of embedded_kcpsm3 is
--------------------------------------------------------------------------------
--
-- declaration of KCPSM3
--
  component kcpsm3
    Port (      address : out std_logic_vector(9 downto 0);
            instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
           write_strobe : out std_logic;
               out_port : out std_logic_vector(7 downto 0);
          read_strobe : out std_logic;
              in_port : in std_logic_vector(7 downto 0);
            interrupt : in std_logic;
        interrupt_ack : out std_logic;
                reset : in std_logic;
                  clk : in std_logic);
  end component;
--
-- declaration of program ROM
--
  component prog_rom
    Port (      address : in std_logic_vector(9 downto 0);
            instruction : out std_logic_vector(17 downto 0);
                  clk : in std_logic);
  end component;
--
--------------------------------------------------------------------------------
-- Signals used to connect KCPSM3 to program ROM
--
signal address : std_logic_vector(9 downto 0);
signal instruction : std_logic_vector(17 downto 0);
--
--------------------------------------------------------------------------------
-- Start of test circuit description
--
begin

  processor: kcpsm3
    port map(    address => address,
             instruction => instruction,
                 port_id => port_id,
            write_strobe => write_strobe,
                out_port => out_port,
             read_strobe => read_strobe,
                 in_port => in_port,
               interrupt => interrupt,
           interrupt_ack => interrupt_ack,
                   reset => reset,
                     clk => clk);

  program: prog_rom
    port map(    address => address,
             instruction => instruction,
                     clk => clk);

end connectivity;
--------------------------------------------------------------------------------
-- END OF FILE EMBEDDED_KCPSM3.VHD
--------------------------------------------------------------------------------
```

Объект EMBEDDED_KCPSM3 может применяться автономно или в качестве входящего в состав разрабатываемой системы элемента. В последнем случае этот объект представляется в форме одноименного компонента, декларация которого выполняется следующим образом.

```
component embedded_kcpsm3
    Port (
        port_id : out std_logic_vector(7 downto 0);
        write_strobe : out std_logic;
        read_strobe : out std_logic;
        out_port : out std_logic_vector(7 downto 0);
        in_port : in std_logic_vector(7 downto 0);
        interrupt : in std_logic;
        interrupt_ack : out std_logic;
        reset : in std_logic;
        clk : in std_logic);
    end component;
```

Создание экземпляра компонента EMBEDDED_KCPSM3 осуществляется с помощью оператора, который имеет следующий вид.

```
inst_name_embedded_processor: embedded_kcpsm3
    port map(
        port_id => port_id_signal,
        write_strobe => write_strobe_signal,
        read_strobe => read_strobe_signal,
        out_port => out_port_signal,
        in_port => in_port_signal,
        interrupt => interrupt_signal,
        interrupt_ack => interrupt_ack_signal,
        reset => reset_signal,
        clk => clk_signal
    );
```

При использовании данного оператора следует вместо метки inst_name_embedded_processor указать идентификатор соответствующего экземпляра компонента EMBEDDED_KCPSM3.

Все компоненты микропроцессорного ядра PicoBlaze, представленные выше, полностью совместимы с любой конфигурацией средств проектирования фирмы Xilinx серии ISE™ (Integrated Synthesis Environment), в том числе и со свободно распространяемой версией WebPACK™ ISE [6].

Общая характеристика системы команд микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

В отличие от других представителей семейства микропроцессорных ядер PicoBlaze, система команд новой версии ядра включает в себя 57 инструкций. При классификации

KiT#49(5).qxd 5/28/2005 5:51 PM Page 101

Таблица 1. Форматы команд переходов микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции								Поле адреса перехода										Мнемоника	Выполняемая операция
1	1	0	1	0	0	X	X	A	A	A	A	A	A	A	A	A	A	JUMP aaa	Безусловный переход
1	1	0	1	0	1	0	0	A	A	A	A	A	A	A	A	A	A	JUMP Z,aaa	Переход при условии, что флаг ZERO Flag находится в установленном состоянии
1	1	0	1	0	1	0	1	A	A	A	A	A	A	A	A	A	A	JUMP NZ,aaa	Переход при условии, что флаг ZERO Flag находится в сброшенном состоянии
1	1	0	1	0	1	1	0	A	A	A	A	A	A	A	A	A	A	JUMP C,aaa	Переход при условии, что флаг CARRY Flag находится в установленном состоянии
1	1	0	1	0	1	1	1	A	A	A	A	A	A	A	A	A	A	JUMP NC,aaa	Переход при условии, что флаг CARRY Flag находится в сброшенном состоянии
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица 2. Форматы команд вызова подпрограмм для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции								Поле адреса подпрограммы										Мнемоника	Выполняемая операция
1	1	0	0	0	0	X	X	A	A	A	A	A	A	A	A	A	A	CALL aaa	Безусловный вызов подпрограммы
1	1	0	0	0	1	0	0	A	A	A	A	A	A	A	A	A	A	CALL Z,aaa	Вызов подпрограммы при условии, что флаг ZERO Flag находится в установленном состоянии
1	1	0	0	0	1	0	1	A	A	A	A	A	A	A	A	A	A	CALL NZ,aaa	Вызов подпрограммы при условии, что флаг ZERO Flag находится в сброшенном состоянии
1	1	0	0	0	1	1	0	A	A	A	A	A	A	A	A	A	A	CALL C,aaa	Вызов подпрограммы при условии, что флаг CARRY Flag находится в установленном состоянии
1	1	0	0	0	1	1	1	A	A	A	A	A	A	A	A	A	A	CALL NC,aaa	Вызов подпрограммы при условии, что флаг CARRY Flag находится в сброшенном состоянии
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица 3. Форматы команд безусловного и условного возвратов из подпрограммы для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции								Нулевые разряды										Мнемоника	Выполняемая операция
1	0	1	0	1	0	X	X	0	0	0	0	0	0	0	0	0	0	RETURN	Безусловный возврат из подпрограммы
1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	RETURN Z	Возврат из подпрограммы при условии, что флаг ZERO Flag находится в установленном состоянии
1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	RETURN NZ	Возврат из подпрограммы при условии, что флаг ZERO Flag находится в сброшенном состоянии
1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	RETURN C	Возврат из подпрограммы при условии, что флаг CARRY Flag находится в установленном состоянии
1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	RETURN NC	Возврат из подпрограммы при условии, что флаг CARRY Flag находится в сброшенном состоянии
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

по функциональному признаку команды подразделяются на семь групп. К шести уже известным группам (они подробно рассмотрены в [2]) добавляется группа команд обмена данными между регистрами общего назначения и СОЗУ.

Кроме появления новых инструкций изменения произошли также в формате стандартных команд, поддерживаемых всеми версиями микропроцессорного ядра PicoBlaze, и их мнемонической записи. Преобразование формата инструкций в первую очередь связано с теми архитектурными особенностями, рассмотренными в предыдущих разделах. Для новой версии ядра длина всех команд в двоичном представлении составляет 18 разрядов.

В большинстве инструкций поменялась длина полей и, соответственно, коды выполняемых операций. В некоторых командах изменилось взаимное расположение полей. В последующих разделах для каждой функциональной группы команд будут представлены соответствующие форматы и мнемоника инструкций микропроцессорного ядра PicoBlaze, предназначенного для использования в составе проектов, реализуемых на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO. Процесс выполнения операций рассматривается подробно только для новых команд, которые не были представлены в [2].

Команды управления последовательностью выполнения операций в программе для ядра PicoBlaze, реализуемого на базе кристаллов семейств Spartan-3, Virtex-II и Virtex-IIPRO

В инструкциях безусловного и условных (ОДИН БЕЗУСЛОВНЫЙ И НЕСКОЛЬКО УСЛОВНЫХ?) переходов *JUMP* изменилась структура поля кода операции и длина поля адреса перехода. В соответствии с разрядностью шины адресов и размером адресного пространства программной памяти, длина поля адреса перехода увеличена до десяти двоичных разрядов. При мнемонической форме записи команд передачи управления в программе значение параметра, определяющего адрес перехода, указывается в виде последовательности, состоящей из трех шестнадцатеричных символов. Форматы команд безусловного и условных переходов *JUMP* для новой версии микропроцессорного ядра PicoBlaze представлены в таблице 1.

Модификация команд обращения к подпрограммам *CALL* также затронула структуру поля кода операции и длину поля адреса вызываемой процедуры. В новой редакции поле адреса вызываемой подпрограммы включает в себя десять двоичных разрядов. При мнемонической форме записи команд обращения к подпрограммам значение параметра, указывающее начальный адрес вызываемой процедуры, представляется в виде трехзначного шестнадцатеричного числа. Новая редакция форматов команд безусловного и условных вызовов подпрограмм *CALL* приведена в таблице 2.

Изменения в формате инструкций возврата из подпрограммы *RETURN* проявились в структуре тех же полей, что и в командах вызова подпрограмм. Для всех вариантов команд завершения выполняемой подпрограммы и передачи управления основной программе или подпрограмме, из которой производился вызов данной процедуры, изменены коды выполняемой операции. Преобразование форматов команд возврата из подпрограммы не отразилось на мнемонической форме записи команд *RETURN*, которая сохранилась без изменений. В таблице 3 представлены модифицированные форматы команд безусловного и условного возвратов из подпрограммы *RETURN*.

Группа логических команд микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II и Virtex-IIPRO

В группу логических команд кроме инструкций, представленных в [2], добавлена новая инструкция — *TEST*, которая будет подробно рассмотрена в конце этого раздела. В формате инструкций, относящихся к группе логических команд, произошли следующие изменения по сравнению с аналогичными инструкциями, поддерживаемыми базовой версией микропроцессорного ядра. Во-первых, длина поля кода операции увеличилась на два разряда и составляет 6 бит. Во-вторых, изменились коды логических операций.

В таблице 4 приведена новая редакция форматов команд поразрядных операций «Логическое И» (поразрядное умножение)(,?) *AND*, выполняемых над содержимым регистра общего назначения с номером *N* и константой

Таблица 4. Форматы команд поразрядных операций «Логическое И» микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	0	1	0	1	0	n	n	n	n	K	K	K	K	K	K	K	K	AND sN, kk	Поразрядное «Логическое И» содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	0	1	0	1	1	n	n	n	n	m	m	m	m	0	0	0	0	AND sN,sM	Поразрядное «Логическое И» содержимого регистров sN и sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

KiT#49(5).qxd 5/28/2005 5:51 PM Page 102

Таблица 5. Форматы команд поразрядных операций «Логическое ИЛИ» микропроцессорного ядра PicoBlaze, предназначенного для реализации на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	0	1	1	0	0	n	n	n	n	K	K	K	K	K	K	K	K	OR sN, kk	Поразрядное «Логическое ИЛИ» содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	0	1	1	0	1	n	n	n	n	m	m	m	m	0	0	0	0	OR sN,sM	Поразрядное «Логическое ИЛИ» содержимого регистров sN и sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица. 6. Форматы команд поразрядных операций «Исключающее ИЛИ» микропроцессорного ядра PicoBlaze, предназначенного для реализации на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	0	1	1	1	0	n	n	n	n	K	K	K	K	K	K	K	K	XOR sN, kk	Поразрядное «Исключающее ИЛИ» содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	0	1	1	1	1	n	n	n	n	m	m	m	m	0	0	0	0	XOR sN,sM	Поразрядное «Исключающее ИЛИ» содержимого регистров sN и sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица 7. Форматы инструкции загрузки данных в регистр общего назначения для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	0	0	0	0	0	n	n	n	n	K	K	K	K	K	K	K	K	LOAD sN, kk	Загрузка константы kk в регистр sN
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	0	0	0	0	1	n	n	n	n	m	m	m	m	0	0	0	0	LOAD sN,sM	Загрузка содержимого регистра sM в регистр sN
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица 8. Форматы команд поразрядного сравнения двух операндов для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	0	0	1	0	n	n	n	n	K	K	K	K	K	K	K	K	TEST sN, kk	Проверка на четность результата операции поразрядное «Логическое И» содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	0	0	1	1	n	n	n	n	m	m	m	m	0	0	0	0	TEST sN,sM	Проверка на четность результата операции поразрядное «Логическое И» содержимого регистров sN и sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

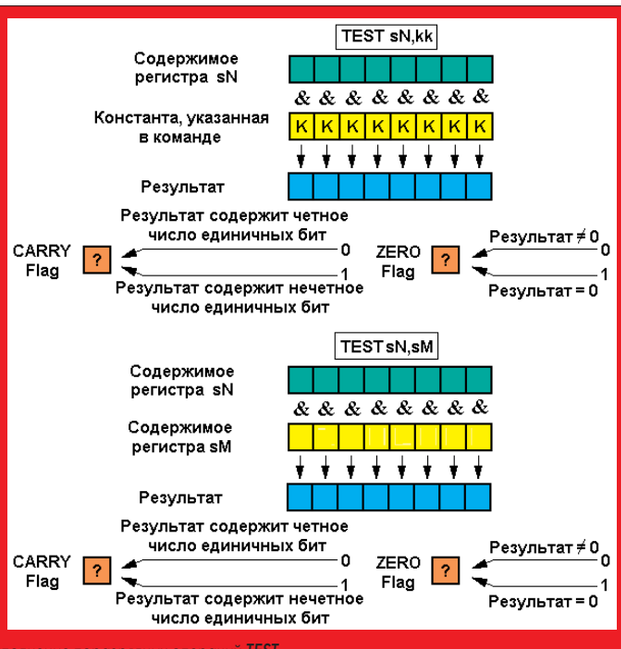

Рис. 3. Выполнение поразрядных операций *TEST*

kk, значение которой задается непосредственно в тексте инструкции, а также над содержимым двух регистров общего назначения с номерами *N* и *M*.

Форматы инструкций *OR*, предназначенных для выполнения операций поразрядного сложения двух операндов (поразрядное «Логическое ИЛИ») в рассматриваемой версии микропроцессорного ядра, определены в таблице 5 для двух вариантов. В первом случае операндами являются содержимое регистра общего назначения с номером *N* и константа *kk*, значение которой указывается в соответствующем поле команды, а во втором — содержимое двух регистров общего назначения с номерами *N* и *M*.

Новая редакция форматов команд *XOR*, используемых для выполнения поразрядной операции «Исключающее ИЛИ» с участием содержимого регистра общего назначения с номером *N* и константы *kk* или содержимого двух регистров с номерами *N* и *M*, представлена в таблице 6.

Таблица 7 представляет новую редакцию формата инструкций *LOAD*, предназначенных для загрузки константы или содержимого какого-либо регистра в выбранный регистр общего назначения.

Команда *TEST* предназначена для выполнения поразрядного сравнения двух операндов. Данная операция выполняется аналогично инструкции поразрядного умножения (поразрядное «Логическое И»). Различие заключается в том, что результат операции *TEST* не запоминается в регистре общего назначения, фиксируются только признаки (состояния флагов). Значение флагов ZERO Flag и CARRY Flag определяется полученным результатом. Если все разряды результирующего слова принимают значение логического нуля, то флаг нулевого результата ZERO Flag устанавливается в состояние, соответствующее логической единице. В противном случае (если хотя бы в одном разряде результата присутствует единичное значение) флаг нулевого результата ZERO Flag переключается в сброшенное состояние (логического нуля). Состояние флага CARRY Flag зависит от количества единичных разрядов в полученном результате. При нечетном количестве единичных бит в слове результата этот флаг устанавливается в состояние, соответствующее логической единице. В противном случае флаг CARRY Flag будет находиться в сброшенном состоянии.

Первым операндом в инструкциях поразрядного сравнения *TEST* всегда является содержимое регистра общего назначения, номер которого указан в виде значения первого параметра команды. В качестве второго операнда используется либо константа, значение которой указывается непосредственно в соответствующем поле команды, либо содержимое другого регистра общего назначения, номер которого задается в виде значения второго параметра инструкции. Команда *TEST* может использоваться для выделения требуемого разряда регистра и контроля его значения. В этом случае второй операнд исполняет роль маски. Форматы команд поразрядного сравнения двух операндов *TEST* приведены в таблице 8.

KiT#49(5).qxd 5/28/2005 5:51 PM Page 103

Команда *TEST sN,kk* выполняет операцию поразрядного сравнения содержимого регистра с номером *N* и константы *kk*. Для поразрядного сравнения содержимого двух регистров общего назначения с номерами *N* и *M* предназначена команда *TEST sN,sM*. Выполнение команд поразрядного сравнения иллюстрирует рис. 3.

Окончание в следующем номере.

KiT#50(6).qxd 8/22/2005 2:46 PM Page 122

Окончание, начало в № 5'2005.

Особенности микропроцессорного ядра PicoBlaze,

предназначенного для применения в проектах, реализуемых на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO

Валерий Зотов
walerry@km.ru

Группа арифметических команд микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Группа арифметических команд новой версии микропроцессорного ядра PicoBlaze пополнилась двумя вариантами инструкции *COMPARE*, которая будет представлена в заключительной части данного раздела. В структуре полей арифметических команд микропроцессорного ядра PicoBlaze, предназначенного для применения в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, произошли те же изменения (по сравнению с форматами аналогичных команд для базовой версии ядра [2]), что и в логических инструкциях, рассмотренных в предыдущем разделе.

В таблице 9 приведены модифицированные варианты форматов команд сложения *ADD* содержимого регистра с номером *N* и константы *kk*, а также содержимого двух регистров общего назначения с номерами *N* и *M* без учета переноса.

Таблица 10 представляет новую версию формата инструкций *ADDCY*, выполняющих операции суммирования двух операндов с учетом значения флага переноса, полученного при выполнении предыдущей команды.

Форматы инструкций *SUB*, предназначенных для выполнения операции вычитания содержимого регистра с номером *M* или константы *kk* из содержимого регистра с номером *N*, без учета заема, в новой редакции представлены в таблице 11.

Модифицированные варианты форматов команд *SUBCY*, используемых для вычисления разности двух операндов с учетом значения заема, образовавшегося при выполнении предыдущей операции, приведены в таблице 12.

Для осуществления арифметического (в отличие от поразрядного) сравнения значений двух восьмиразрядных операндов предназначена команда *COMPARE*. В качестве операндов при выполнении данной инструкции могут выступать содержимое регистра общего назначения и константа или содержимое двух регистров общего назначения. Первым операндом в инструкциях *COMPARE* всегда является содержимое регистра общего назначения, номер которого *N* указан в виде значе-

Таблица. 9. Форматы команд сложения двух операндов без учета переноса для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	1	0	0	0	n	n	n	n	K	K	K	K	K	K	K	K	ADD sN, kk	Сложение содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	1	0	0	1	n	n	n	n	m	m	m	m	0	0	0	0	ADD sN,sM	Сложение содержимого регистров sN и sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица. 10. Форматы команд сложения двух операндов с учетом переноса для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	1	0	1	0	n	n	n	n	K	K	K	K	K	K	K	K	ADDCY sN, kk	Сложение содержимого регистра sN и константы kk с учетом переноса
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	1	0	1	1	n	n	n	n	m	m	m	m	0	0	0	0	ADDCY sN,sM	Сложение содержимого регистров sN и sM с учетом переноса
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица. 11. Форматы команд вычитания без учета заема для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	1	1	0	0	n	n	n	n	K	K	K	K	K	K	K	K	SUB sN, kk	Вычитание из содержимого регистра sN константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	1	1	0	1	n	n	n	n	m	m	m	m	0	0	0	0	SUB sN,sM	Вычитание содержимого регистра sM из содержимого регистра sN
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица. 12. Форматы инструкций вычитания с учетом заема для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	1	1	1	0	n	n	n	n	K	K	K	K	K	K	K	K	SUBCY sN, kk	Вычитание из содержимого регистра sN константы kk с учетом заема
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	1	1	1	1	n	n	n	n	m	m	m	m	0	0	0	0	SUBCY sN,sM	Вычитание содержимого регистра sM из содержимого регистра sN с учетом заема
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

KiT#50(6).qxd 8/22/2005 2:46 PM Page 123

Таблица. 13. Форматы инструкций арифметического сравнения двух операндов для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле константы								Мнемоника	Выполняемая операция
0	1	0	1	0	0	n	n	n	n	K	K	K	K	K	K	K	K	COMPARE sN, kk	Сравнение содержимого регистра sN и константы kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	1	0	1	0	1	n	n	n	n	m	m	m	m	0	0	0	0	COMPARE sN,sM	Сравнение содержимого регистра sM и содержимого регистра sN
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

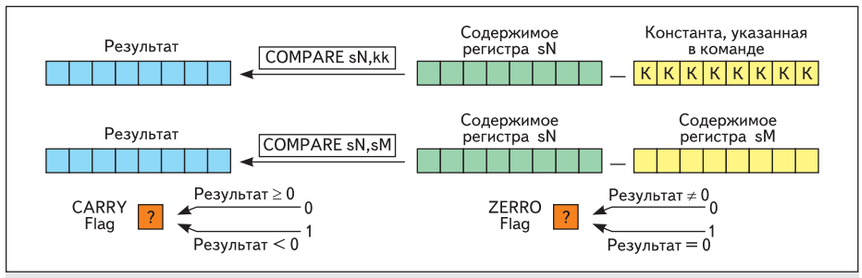

Рис. 4. Осуществление операций арифметического сравнения двух операндов

Таблица. 14. Форматы команд логического (арифметического) и циклического сдвига данных микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции					Поле номера регистра				Поле направления сдвига				Поле типа сдвига			Мнемоника	Выполняемая операция
0	0	0	0	0	n	n	n	n	0	0	0	1	1	1	0	SR0 sN	Логический сдвиг содержимого регистра sN вправо на один разряд с записью 0
0	0	0	0	0	n	n	n	n	0	0	0	1	1	1	1	SR1 sN	Логический сдвиг содержимого регистра sN вправо на один разряд с записью 1
0	0	0	0	0	n	n	n	n	0	0	0	1	0	1	0	SRX sN	Логический сдвиг содержимого регистра sN вправо с сохранением последнего разряда
0	0	0	0	0	n	n	n	n	0	0	0	1	0	0	0	SRA sN	Циклический сдвиг содержимого регистра sN вправо через разряд переноса/заема
0	0	0	0	0	n	n	n	n	0	0	0	1	1	0	0	RR sN	Циклический сдвиг содержимого регистра sN вправо без участия бита переноса
0	0	0	0	0	n	n	n	n	0	0	0	0	1	1	0	SL0 sN	Логический сдвиг содержимого регистра sN влево на один разряд с записью 0
0	0	0	0	0	n	n	n	n	0	0	0	0	1	1	1	SL1 sN	Логический сдвиг содержимого регистра sN влево на один разряд с записью 1
0	0	0	0	0	n	n	n	n	0	0	0	0	1	0	0	SLX sN	Логический сдвиг содержимого регистра sN влево с сохранением последнего разряда
0	0	0	0	0	n	n	n	n	0	0	0	0	0	0	0	SLA sN	Циклический сдвиг содержимого регистра sN влево через разряд переноса/заема
0	0	0	0	0	n	n	n	n	0	0	0	0	0	1	0	RL sN	Циклический сдвиг содержимого регистра sN влево без участия бита переноса
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

Таблица. 15. Форматы команд ввода-вывода микропроцессорного ядра PicoBlaze, предназначенного для реализации на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле адреса порта ввода/вывода								Мнемоника	Выполняемая операция
0	0	0	1	0	0	n	n	n	n	K	K	K	K	K	K	K	K	INPUT sN, kk	Чтение данных из порта ввода/вывода с адресом kk в регистр sN
1	0	1	1	0	0	n	n	n	n	K	K	K	K	K	K	K	K	OUTPUT sN, kk	Запись данных из регистра sN в порт ввода/вывода с адресом kk
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
0	0	0	1	0	1	n	n	n	n	m	m	m	m	0	0	0	0	INPUT sN,(sM)	Чтение данных из порта ввода/вывода с адресом, определяемым регистром sM, в регистр sN
1	0	1	1	0	1	n	n	n	n	m	m	m	m	0	0	0	0	OUTPUT sN,(sM)	Запись данных из регистра sN в порт с адресом, определяемым регистром sM
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

ния первого параметра команды. В качестве второго операнда выступает либо константа *kk*, значение которой указывается непосредственно в коде команды, либо содержимое другого регистра общего назначения с номером *M*, который задается в виде значения второго параметра команды. В таблице 13 показаны форматы двух вариантов команды сравнения *COMPARE*.

Исполнение команды *COMPARE* аналогично выполнению операции вычитания без учета заема, но полученный при этом результат не сохраняется. Информация о соотношении операндов отражается в состоянии флагов. Если значения операндов равны, то все разряды результирующего слова принимают значение логического нуля, и флаг нулевого результата ZERO Flag устанавливается в состояние, соответствующее логической единице. В остальных случаях флаг нулевого результата ZERO Flag находится в сброшенном состоянии. В том случае, если значение второго операнда больше значения первого операнда, то флаг CARRY Flag устанавливается в состояние, соответствующее логической единице. При другом соотношении операндов флаг CARRY Flag остается в сброшенном состоянии. Рис. 4 поясняет выполнение команд арифметического сравнения двух операндов.

Команды сдвига данных микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

В формате команд, предназначенных для выполнения операций сдвига данных, произошли те же изменения, что и в формате логических инструкций — длина поля кода операции увеличилась на два бита и составляет шесть двоичных разрядов, а также поменялось значение кода операции. Мнемоническая форма записи команд сдвига осталась прежней. Новые форматы инструкций логического (арифметического) и циклического сдвига данных, находящихся в регистре общего назначения с указанным номером, представлены в таблице 14.

Команды ввода-вывода микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Формат инструкций ввода-вывода, предназначенных для чтения данных из входного порта в заданный регистр общего назначения и передачи информации из указанного регистра в выходной порт, отличается от структуры аналогичных команд базового варианта ядра PicoBlaze длиной поля кода операции [2]. В командах ввода-вывода новой версии ядра длина поля кода операции составляет шесть двоичных разрядов. Кроме того, в инструкциях ввода-вывода поменялось значение кода операции. Мнемоническая форма записи этих инструкций сохранилась без изменений. Новые варианты форматов команд ввода-вывода с различными видами адресации приведены в таблице 15.

Команды обслуживания прерываний микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

В инструкциях, относящихся к группе команд обслуживания прерываний, изменилась длина поля кода операции и поля режима обработки прерываний. Длина каждого из этих полей команды в новой версии составляет девять разрядов. Кроме того, изменились зна-

KiT#50(6).qxd 8/22/2005 2:46 PM Page 124

Таблица. 16. Форматы команд обслуживания прерываний микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции									Поле режима обработки прерываний								Мнемоника	Выполняемая операция
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	RETURNI ENABLE	Возврат из процедуры обработки и установка режима запрета прерывания
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RETURNI DISABLE	Возврат из процедуры обработки и установка режима разрешения прерывания
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	ENABLE INTERRUPT	Установка режима разрешения прерывания
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	DISABLE INTERRUPT	Установка режима запрета прерывания
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды

Таблица 17. Форматы команд чтения и записи данных в сверхоперативное запоминающее устройство микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Поле кода операции						Поле номера регистра				Поле адреса ячейки								Мнемоника	Выполняемая операция
1	0	1	1	1	0	n	n	n	n	0	0	K	K	K	K	K	K	STORE sN, kk	Запись данных из регистра sN в ячейку СОЗУ с адресом kk
0	0	0	1	1	0	n	n	n	n	K	K	K	K	K	K	K	K	FETCH sN, kk	Чтение данных из ячейки СОЗУ с адресом kk в регистр sN
Поле кода операции						Поле номера первого регистра				Поле номера второго регистра				Нулевые разряды				Мнемоника	Выполняемая операция
1	0	1	1	1	1	n	n	n	n	m	m	m	m	0	0	0	0	STORE sN,(sM)	Запись данных из регистра sN в ячейку СОЗУ с адресом, определяемым регистром sM
0	0	0	1	1	1	n	n	n	n	m	m	m	m	0	0	0	0	FETCH sN,(sM)	Чтение данных из ячейки СОЗУ с адресом, определяемым регистром sM, в регистр sN
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Номер разряда микрокоманды	

чения кодов операций обработки прерываний. При этом мнемоническая форма записи инструкций, используемых для обработки прерываний, осталась без изменений.

Форматы команд возврата из процедуры обслуживания прерываний *RETURNI* и установки режима обслуживания прерываний в программе *ENABLE INTERRUPT* и *DISABLE INTERRUPT* для микропроцессорного ядра PicoBlaze, реализуемого на базе кристаллов семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, представлены в таблице 16.

Команды чтения и записи данных в сверхоперативное запоминающее устройство микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Команды чтения и записи данных в сверхоперативное запоминающее устройство предназначены для организации обмена данными между регистрами общего назначения и внутренней сверхоперативной памятью. В инструкциях, относящихся к рассматриваемой группе, используются два параметра. Значение первого параметра определяет номер регистра общего назначения, используемого в качестве приемника данных при выполнении операции чтения информации из СОЗУ или в качестве источника при осуществлении операции записи данных в СОЗУ. Значение второго параметра определяет адрес ячейки сверхоперативной памяти, к которой производится обращение. Адресация ячеек СОЗУ в инструкциях чтения и записи данных может осуществляться с помощью восьмиразрядной константы, значение которой задается непосредственно в команде, или содержимого регистра общего назначения с указанным номером. Форматы команд чтения и записи данных в сверхоперативное запоминающее устройство микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, приведены в таблице 17.

Команда *FETCH sN, kk* выполняет операцию чтения данных из ячейки СОЗУ с адресом kk в регистр общего назначения с номером sN. Чтение данных из ячейки СОЗУ, адрес которой указывает содержимое регистра sM, осуществляется с помощью инструкции *FETCH sN,(sM)*.

Передача содержимого регистра общего назначения с номером sN в ячейку СОЗУ с адресом kk осуществляется с помощью команды STORE sN, kk. Для записи данных из регистра sN в ячейку СОЗУ, адрес которой определяется содержимым регистра sM, предназначена команда STORE sN,(sM). Рис. 5 демонстрирует процесс выполнения операций чтения и записи данных в сверхоперативное запоминающее устройство при различных видах адресации.

Ассемблер микропроцессорного ядра PicoBlaze, предназначенного для использования в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Новый вариант микропроцессорного ядра PicoBlaze, предназначенный для применения в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, поддерживается соответствующей версией ассемблера, которая включена в состав архива файлов, предоставляемого пользователю. Новая версия ассемблера реализована в виде программы Kcpsm3.exe, которая формирует файлы описания содержимого программной памяти в различных форматах. Данная программа, в отличие от ассемблера Kcpsm2.exe [5], поддерживает все новые инструкции микропроцессорного ядра PicoBlaze, реализуемого на базе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, а также учитывает изменения в формате команд и параметры блочного ОЗУ используемых кристаллов.

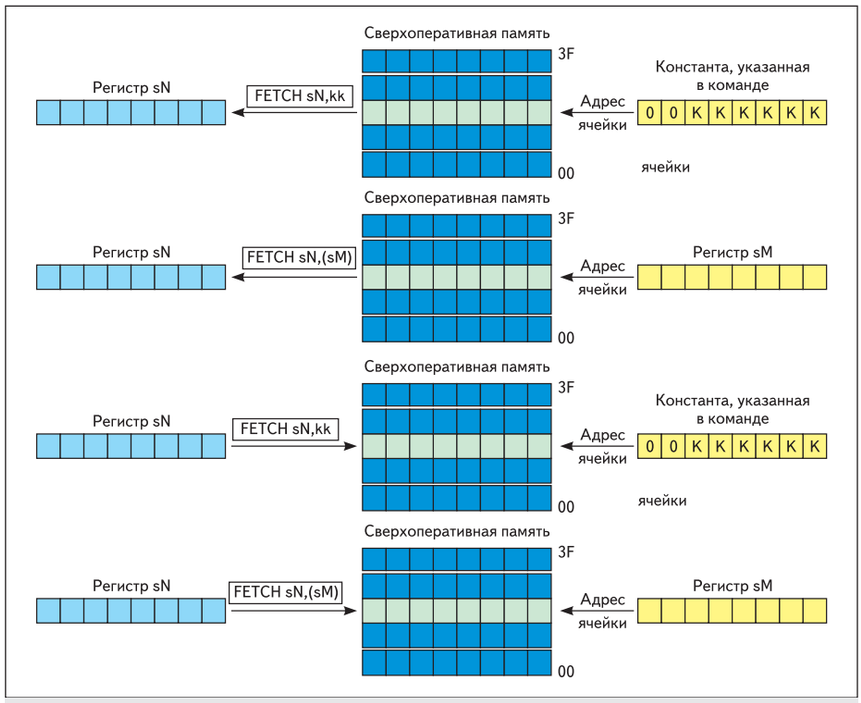

Рис. 5. Выполнение инструкций чтения и записи данных в сверхоперативное запоминающее устройство при различных видах адресации

KiT#50(6).qxd 8/22/2005 2:46 PM Page 125

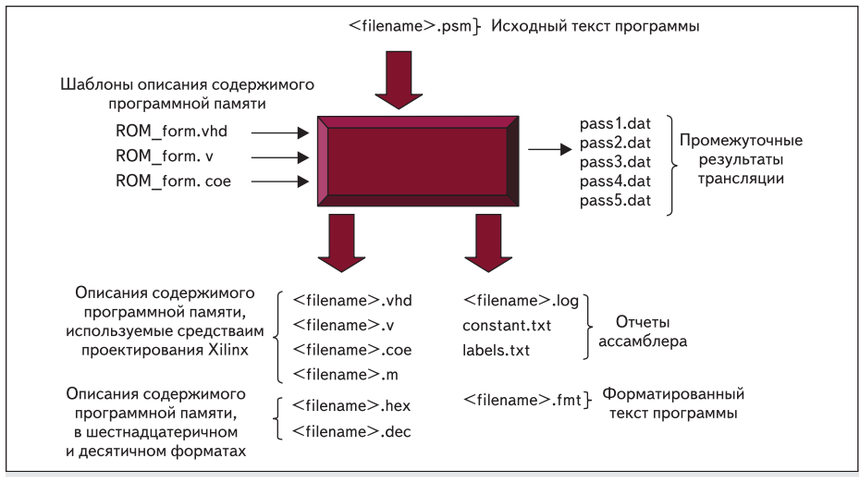

Рис. 6. Файлы, формируемые ассемблером микропроцессорного ядра PicoBlaze, предназначенного для реализации на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Трансляция исходного текста программ для микропроцессорного ядра PicoBlaze, встраиваемого в проекты, реализуемые на базе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, осуществляется с помощью командной строки, формат которой имеет следующий вид:

```
Kcpsm3.exe <идентификатор_файла_с_исходным_текстом_программы_на_языке_ассемблера>[.psm]
```

В приведенной выше командной строке квадратные скобки служат для обозначения необязательного параметра. Таким образом, используемое по умолчанию расширение файла, содержащего исходный текст программы, psm можно не указывать.

На рис. 6 в наглядном виде представлена информация об исходных файлах, используемых при трансляции программ микропроцессорного ядра PicoBlaze, предназначенного для реализации в кристаллах семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, и выходных файлах, формируемых ассемблером KCPSM3.

Для выполнения трансляции разработанной программы кроме основного файла, содержащего исходный текст этой программы на языке ассемблера, необходимы файлы шаблонов ROM_form.vhd, ROM_form.v и ROM_form.coe. Файлы ROM_form.vhd и ROM_form.v представляют собой шаблоны описания содержимого ППЗУ программ на языках высокого уровня VHDL и Verilog соответственно. В файле ROM_form.coe содержится шаблон описания содержимого ППЗУ микропрограмм в формате, воспринимаемом генератором ядер Xilinx CORE Generator.

При отсутствии ошибок в исходном тексте транслируемой программы ассемблером KCPSM3 формируется набор файлов, название которых за исключением нескольких отчетов совпадает с идентификатором исходного файла, а расширение соответствует типу содержащейся в них информации.

Все файлы, генерируемые новой версией ассемблера, можно условно разбить на пять групп. Первую группу составляют файлы, в которые записывается содержимое программной памяти в форматах, воспринимаемых средствами проектирования фирмы Xilinx серии ISE. В эту группу входят файлы описания содержимого программной памяти на языках VHDL и Verilog, а также в формате, воспринимаемом генератором ядер Xilinx CORE Generator, которые имеют расширение *vhd*, *v* и *coe* соответственно. VHDL- и Verilog-описания содержимого ППЗУ программ используются на этапах синтеза и моделирования проектируемой системы. Ко второй группе относятся файлы, описывающие содержимое ППЗУ микропрограмм в форматах, используемых утилитами, не входящими в состав САПР серии ISE. В эту группу входят файлы, содержащие результаты трансляции в виде кодов в шестнадцатеричном и десятичном представлении, которые имеют расширение *hex* и *dec* соответственно. Третью группу образуют файлы отчетов о ходе и результатах выполнения процесса трансляции. Файл с расширением *log* содержит детальную информацию о трансляции каждой строки ассемблерной программы. В файле *constant.txt* перечисляются все константы, используемые в программе, с указанием их значений. Файл *labels.txt* содержит информацию о метках, которые содержатся в программе, и адресах программной памяти, которые им соответствуют. В четвертую группу входит единственный файл, имеющий расширение *fmt*, в котором записан отформатированный вариант исходного текста программы на языке ассемблера. К пятой группе относятся файлы *pass1.dat–pass5.dat*, содержащие промежуточные результаты, полученные на различных фазах процесса трансляции. Информация, которая содержится в этих файлах, может использоваться в процессе отладки транслируемой программы.

При переходе от одной из прежних версий микропроцессорного ядра PicoBlaze к новому варианту, рассматриваемому в настоящей статье, следует обратить внимание на то, что разработанное ранее программное обеспечение не может использоваться непосредственно в новой версии ядра. Для переноса этих программ необходимо привести их исходные тексты в соответствие с форматами команд, поддерживаемыми ассемблером KCPSM3. Особое внимание нужно обратить на формат представления адресов переходов и подпрограмм, а также номеров регистров общего назначения. При этом нужно учитывать возможные различия в объеме программной памяти и блока регистров общего назначения в используемой ранее и новой версии микропроцессорного ядра PicoBlaze. Кроме того, различная глубина стека в этих версиях накладывает ограничения на количество вложенных вызовов подпрограмм. После корректировки исходных текстов программ следует выполнить их трансляцию с помощью новой версии ассемблера KCPSM3.

Загрузчик программ для микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

Заметным недостатком рассмотренных ранее микропроцессорных ядер семейства PicoBlaze [1–5] является необходимость повторного выполнения этапов синтеза и реализации аппаратной части разрабатываемой системы при внесении каких-либо изменений в программное обеспечение. На рис. 7 показан типовой маршрут проектирования встраиваемых микропроцессорных систем на основе восьмиразрядного ядра PicoBlaze. Он включает в себя следующие этапы [6]:

- формирование проекта и HDL-описания разрабатываемой микропроцессорной системы;
- подготовка исходного текста программы на языке ассемблера для проектируемой микропроцессорной системы;
- трансляция разработанной программы с помощью ассемблера;
- синтез разрабатываемой микропроцессорной системы;
- размещение и трассировка разрабатываемой микропроцессорной системы в кристалле;
- формирование конфигурационной последовательности проектируемой микропроцессорной системы;
- загрузка конфигурационной последовательности встраиваемой микропроцессорной системы в кристалл ПЛИС.

Для новой версии ядра, реализуемого на основе кристаллов семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, в архив файлов, пре-

KiT#50(6).qxd 8/22/2005 2:46 PM Page 126

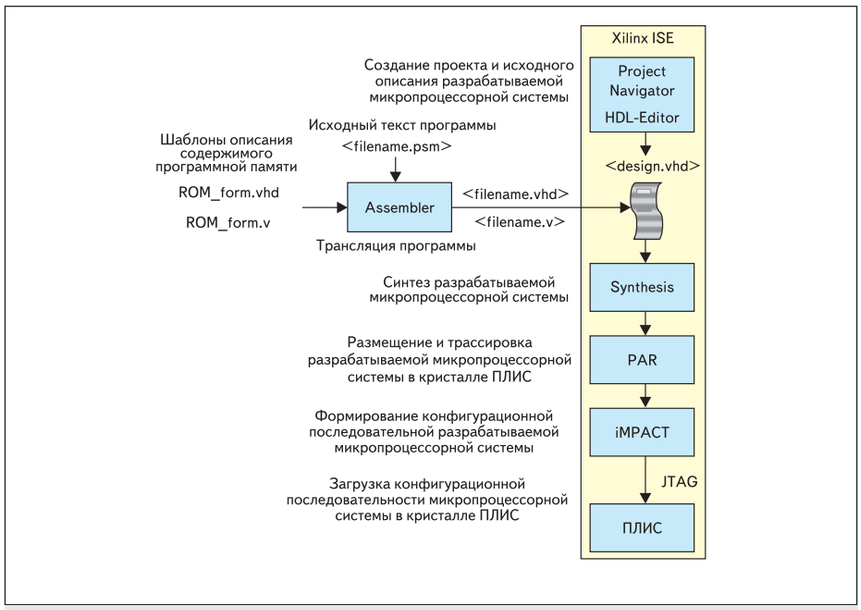

Рис. 7. Типовая последовательность этапов проектирования встраиваемых микропроцессорных систем на основе ядра PicoBlaze, реализуемых в ПЛИС FPGA семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4

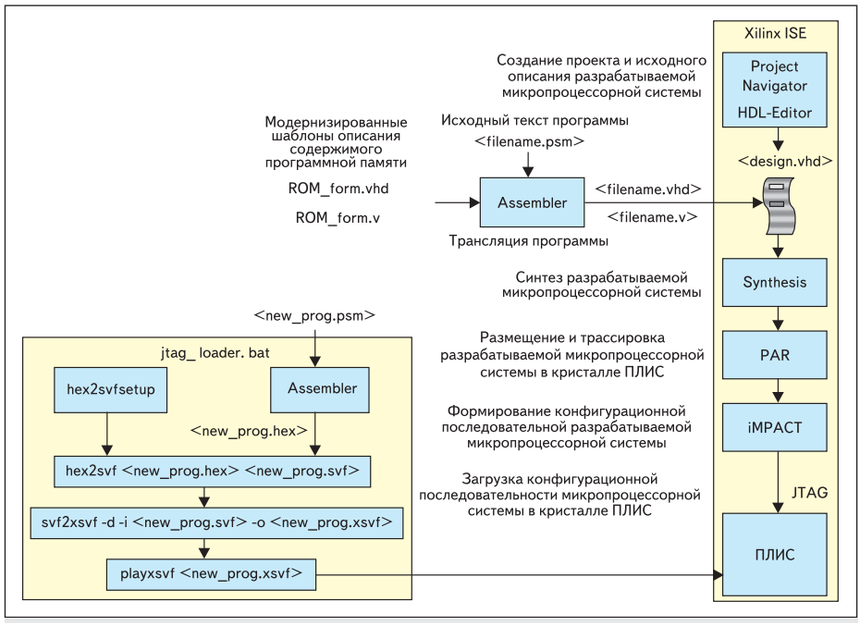

Рис. 8. Маршрут проектирования встраиваемых микропроцессорных систем на основе ядра PicoBlaze, реализуемых в ПЛИС FPGA семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, с использованием загрузчика программ JTAG Program Loader

доставляемых пользователю, включен загрузчик программного кода через порт JTAG-интерфейса применяемой ПЛИС. При использовании загрузчика *JTAG Program Loader* маршрут проектирования встраиваемых микропроцессорных систем на основе ядра PicoBlaze приобретает вид, показанный на рис. 8. Типовая ветвь этого маршрута, включающая этапы синтеза, реализации и формирования конфигурационной последовательности, выполняется только один раз. В случае модификации микропроцессорной программы достаточно выполнить трансляцию исходного текста этой программы и воспользоваться загрузчиком *JTAG Program Loader*. Запись кода новой микропроцессорной программы непосредственно в программную память ядра PicoBlaze осуществляется через порт JTAG-интерфейса с помощью стандартного загрузочного кабеля фирмы Xilinx, предназначенного для конфигурирования ПЛИС.

Загрузчик программ *JTAG Program Loader* выполнен в виде командного файла jtag_loader.bat и комплекта утилит преобразования формата и управления конфигурированием ПЛИС. Этот командный файл содержит набор директив, посредством которых осуществляется последовательный вызов утилит преобразования формата микропроцессорных программ и управления процессом их загрузки в ПЛИС. Исходным модулем для загрузчика является файл, содержащий результаты трансляции исходного текста программы в виде кодов в шестнадцатеричном представлении. Этот файл формируется ассемблером KCPSM3.

Для использования загрузчика программ *JTAG Program Loader* в структуру микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, были внесены следующие изменения. Программная память реализуется в виде двухпортового оперативного запоминающего устройства. Один из портов программной памяти сопряжен непосредственно с контроллером JTAG-интерфейса. Поэтому новая программа может быть записана в ППЗУ программ через порт JTAG-интерфейса ПЛИС. Кроме того, в блоке программной памяти дополнительно формируется сигнал сброса, который используется для инициализации модуля исполнительного устройства микропроцессорного ядра PicoBlaze. Такая модернизации модуля программной памяти обусловлена тем, что сразу же после загрузки новой программы исполнительное устройство должно быть переведено в исходное (начальное состояние). С учетом этих изменений необходимо в выражениях декларации и создания экземпляра компонента программной памяти добавить описание порта, соответствующего выходу сигнала сброса. При формировании структурного описания микропроцессорного ядра необходимо подключить выход сигнала сброса программной памяти *prog_rom* ко входу сброса исполнительного устройства *kcpsm3*.

Модернизированная структура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4 с поддержкой загрузки программного кода через порт JTAG-интерфейса, изображена на рис. 9.

В процессе трансляции программ, которые должны записываться в программную память микропроцессорного ядра PicoBlaze через порт JTAG-интерфейса ПЛИС с помощью загрузчика *JTAG Program Loader*, необходимо использовать модифицированные варианты шаблонов описания содержимого ППЗУ вместо файлов ROM_form.vhd и ROM_form.v. Эти варианты соответствуют ППЗУ программ, реализуемому в виде двухпортовой памяти. Модифицированные варианты шаблонов поставляются вместе с загрузчиком в форме файлов JTAG_Loader_ROM_form.vhd и JTAG_Loader_ROM_form.v соответственно.

KiT#50(6).qxd 8/22/2005 2:46 PM Page 127

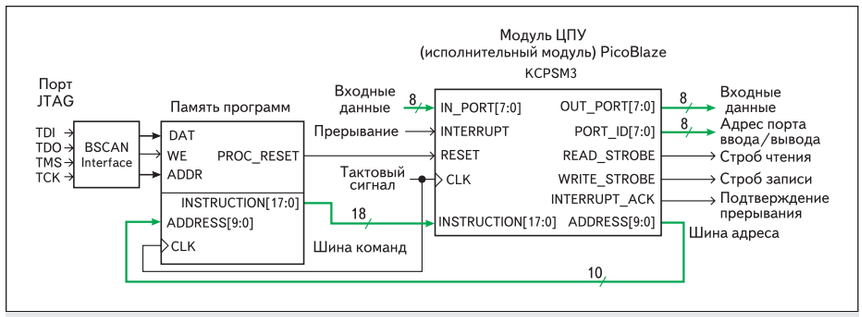

Рис. 9. Модернизированная структура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4 с поддержкой загрузки программ через порт JTAG-интерфейса

Заключение

Рассмотренная версия микропроцессорного ядра, предназначенная для реализации на основе ПЛИС семейств Spartan-3, Virtex-II, Virtex-IIPRO и Virtex-4, как и другие представители семейства PicoBlaze, является свободно распространяемой (бесплатной). Для получения архива файлов, включающего исходные описания компонентов ядра, соответствующую версию ассемблера и загрузчик программ *JTAG Program Loader*, следует обращаться в центр поддержки и продаж InlineGROUP (**www.plis.ru**) — официальному дистрибьютору фирмы Xilinx в России, Беларуси и Украине. Практическое освоение методов проектирования встраиваемых микропроцессорных систем, выполняемых на основе рассмотренного варианта ядра PicoBlaze, целесообразно осуществлять с помощью отладочной платы из инструментального комплекта Spartan-3 Starter Kit. Этот комплект отличается невысокой стоимостью и включает в себя помимо отладочной платы загрузочный кабель, который может применяться в дальнейшем для конфигурирования ПЛИС различных семейств, выпускаемых фирмой Xilinx, и средства проектирования WebPack ISE и Foundation ISE (ознакомительную версию).

Литература

1. Зотов В. PicoBlaze — семейство восьмиразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Компоненты и технологии, № 4, 2003.
2. Зотов В. Система команд микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E // Компоненты и технологии, № 5, 2003.
3. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства Virtex-II // Компоненты и технологии, № 6, 2003.
4. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II // Компоненты и технологии, № 7, 2003.
5. Зотов В. Разработка программ на языке ассемблера для семейства микропроцессорных ядер PicoBlaze // Компоненты и технологии, № 8, 2003.
6. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. М.: Горячая линия — Телеком, 2003.
7. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. М.: Горячая линия — Телеком, 2004.

www.finestreet.ru

Создание проекта встраиваемой системы на основе микропроцессорного ядра семейства PicoBlaze

Валерий ЗОТОВ
walerry@km.ru

В предыдущей части данного цикла было представлено семейство свободно распространяемых 8-разрядных ядер PicoBlaze [1–6]. После ознакомления с архитектурой, системой команд и ассемблером ядер этого семейства можно приступить к практическому изучению процесса разработки встраиваемых микропроцессорных систем на их основе.

Так как ПЛИС с архитектурой FPGA обладают более широкими функциональными возможностями по сравнению с кристаллами серий CPLD [8], для реализации встраиваемых микропроцессорных систем в большинстве случаев рекомендуется использовать микросхемы семейств FPGA. Поэтому в данной и последующих статьях основное внимание будет уделяться изучению процесса проектирования систем на основе ядер семейства PicoBlaze, реализуемых в ПЛИС серий FPGA. Настоящая статья знакомит со средствами и этапами проектирования встраиваемых микропроцессорных систем на основе ядер семейства PicoBlaze, а также с процессом создания нового проекта.

Средства проектирования встраиваемых микропроцессорных систем на основе 8-разрядных ядер семейства PicoBlaze

Для сквозного проектирования встраиваемых микропроцессорных систем на основе ядер семейства PicoBlaze необходимы следующие инструменты:

- пакет средств автоматизированного проектирования серии Xilinx ISE (Integrated Synthesis Environment);
- архив, содержащий модули исходных описаний и ассемблер для выбранной версии микропроцессорного ядра PicoBlaze;
- загрузочный кабель JTAG-интерфейса;
- аппаратный отладочный модуль.

Кроме того, для отладки разрабатываемых микропроцессорных кодов могут применяться специальные программные средства, выпускаемые «третьими» фирмами и производителями. Использование этих средств не является обязательным, но позволяет повысить эффективность и наглядность процесса отладки.

Средства проектирования серии Xilinx ISE выпускаются в трех конфигурациях: ISE Foundation, ISE BaseX и ISE WebPACK. Основное отличие между этими конфигурациями заключается в количестве поддерживаемых кристаллов и наборе дополнительных инструментов проектирования. К моменту подготовки материала данной статьи последней версией средств проектирования серии Xilinx ISE, доступной для разработчиков, являлась версия 7.1i. Ниже приводятся краткие рекомендации по выбору конфигурации САПР для разработки систем на основе ядер семейства PicoBlaze.

Пакет программных средств ISE Foundation представляет собой наиболее полную систему сквозного проектирования, которая поддерживает весь спектр ПЛИС фирмы Xilinx. Данную конфигурацию САПР следует использовать в том случае, когда кристалл, выбранный для реализации встраиваемой микропроцессорной системы, не поддерживается двумя другими пакетами. Для практического освоения методов проектирования встраиваемых микропроцессорных систем, реализуемых на основе ПЛИС фирмы Xilinx, можно воспользоваться бесплатной 60-дневной версией пакета САПР ISE Foundation.

Экономичная конфигурация средств проектирования ISE BaseX имеет более низкую стоимость по сравнению с ISE Foundation, но поддерживает не все типы ПЛИС. Пакет ISE BaseX позволяет выполнять проекты на основе всех кристаллов семейств CPLD и ПЛИС серий FPGA с логической емкостью не более 600 000 системных вентилей. Кроме того, модуль программирования iMPACT, входящий в состав пакета, может применяться для конфигурирования практически всех кристаллов, выпускаемых фирмой Xilinx. Для создания конфигурационной последовательности при этом используются другие средства проектирования, предоставляемые фирмой Xilinx. Пакет ISE BaseX рекомендуется применять в том случае, если в процессе разработки микропроцессорной системы предполагается использование генератора логических ядер CORE Generator и топологического редактора FPGA Editor (при выборе ПЛИС, логическая емкость которых не выходит за указанный выше предел).

Свободно распространяемая (бесплатная) модификация САПР ISE WebPACK поддерживает все кристаллы семейств CPLD и ПЛИС серий FPGA с логической емкостью не более 300 000 системных вентилей. Кроме того, одно из главных отличий пакета ISE WebPACK от конфигурации ISE BaseX заключается в отсутствии генератора логических ядер CORE Generator и топологического редактора FPGA Editor. Бесплатную конфигурацию САПР серии Xilinx ISE WebPACK целесообразно использовать в том случае, когда предполагается использовать следующие типы ПЛИС:

- кристаллы XCV50E — XCV300E семейства Virtex-E;
- кристаллы XC2V40 — XC2V250 семейства Virtex-II;
- кристалл XC2VP2 семейства Virtex-II Pro;
- кристаллы XC4VLX15, XC4VLX25 семейства Virtex-4;
- все кристаллы семейства Spartan-II;
- кристаллы XC2S50E — XC2S300E семейства Spartan IIE;
- кристаллы XC3S50 — XC3S1500 семейства Spartan-3;
- кристаллы XC3S100E — XC3S500E семейства Spartan-3E;
- кристаллы XC3S1000L, XC3S1500L семейства Spartan-3L;
- все кристаллы семейства CoolRunner-II.

Все конфигурации средств проектирования серии Xilinx ISE имеют одинаковую структуру и пользовательский интерфейс. Поэтому переход от одной конфигурации САПР к другой, например, от свободно рас-

пространяемой системы проектирования ISE WebPACK к полному пакету ISE Foundation, не требует дополнительных временных затрат для изучения нового пакета [9].

Возможные варианты получения архива, содержащего модули исходных описаний и ассемблер для выбранной версии микропроцессорного ядра PicoBlaze, подробно описаны в других статьях цикла [1, 3–5].

Кабель JTAG-интерфейса, необходимый для загрузки конфигурационной последовательности микропроцессорной системы в ПЛИС, можно изготовить самостоятельно [10]. Кроме того, фирмой Xilinx выпускается несколько вариантов универсального загрузочного JTAG-кабеля, предназначенных для сопряжения с различными внешними интерфейсами персонального компьютера.

Для аппаратной отладки разрабатываемой системы может использоваться самостоятельно изготовленное устройство или один из серийно выпускаемых универсальных инструментальных модулей, выполненных на основе ПЛИС фирмы Xilinx семейств CPLD. Преимуществами применения недорогих промышленно выпускаемых аппаратных модулей являются:

- сокращение общего времени процесса разработки микропроцессорной системы;
- исключение вероятных ошибок, вносимых при изготовлении печатной платы и монтаже компонентов;
- наличие в комплекте загрузочного JTAG-кабеля;
- комплектация всеми необходимыми средства проектирования.

Примером такого отладочного модуля является плата Xilinx Spartan-3 Starter Board, которая поставляется в составе инструментального комплекта Spartan-3 Starter Kit [7].

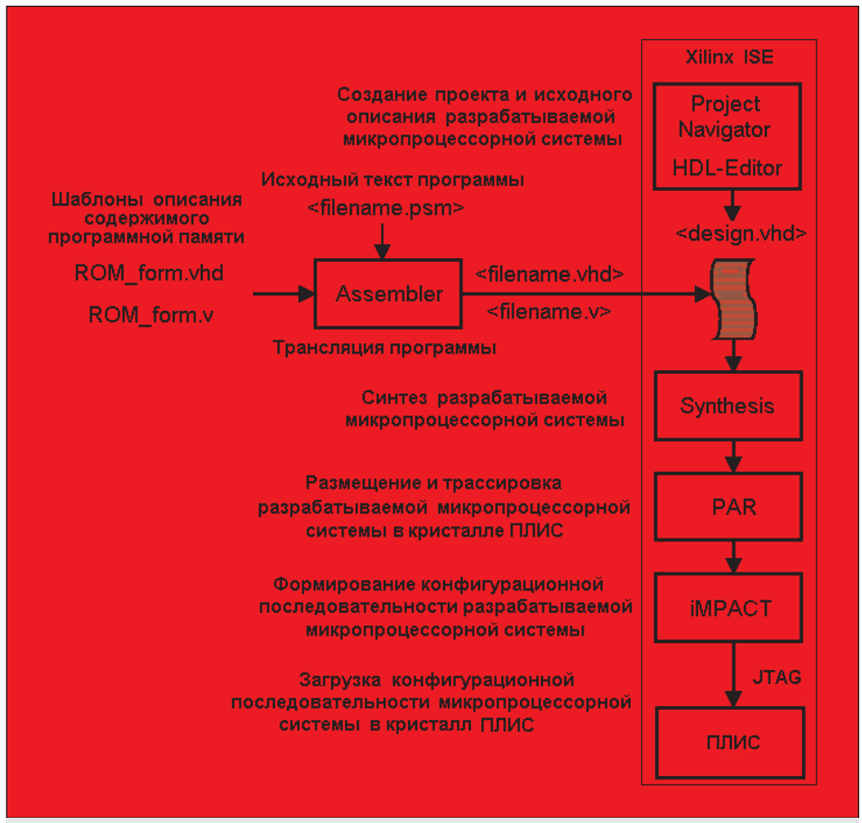

Рис. 1. Последовательность этапов проектирования встраиваемых систем на основе микропроцессорных ядер семейства PicoBlaze

Этапы проектирования встраиваемых систем на основе микропроцессорных ядер семейства PicoBlaze

Проектирование микропроцессорных систем с использованием 8-разрядных ядер семейства PicoBlaze включает в себя все стандартные этапы разработки цифровых устройств с аппаратной реализацией операций на базе ПЛИС фирмы Xilinx. Содержание и выполнение этих этапов в САПР серии Xilinx ISE подробно рассмотрено в другой работе автора [9], поэтому далее они освещаются кратко. Кроме того, маршрут проектирования микропроцессорных систем на основе ядер рассматриваемого семейства содержит этапы создания программного обеспечения для разрабатываемой системы.

В общем случае в процессе проектирования встраиваемых систем на основе микропроцессорных ядер семейства PicoBlaze можно выделить следующие этапы:

- выбор семейства, типа ПЛИС, соответствующей версии микропроцессорного ядра для реализации проектируемой системы и разработка ее архитектуры;
- создание исходного текста микропроцессорной программы на языке ассемблера;
- трансляция программы на языке ассемблера;
- создание нового проекта в САПР серии Xilinx ISE с указанием выбранного семейства и типа ПЛИС, а также средств синтеза и моделирования;
- подготовка описания проектируемой системы в схемотехнической, алгоритмической или текстовой форме;
- синтез проектируемой микропроцессорной системы;
- отладка программной и аппаратной части разрабатываемой системы методом функционального моделирования;
- размещение и трассировка проекта встраиваемой микропроцессорной системы в кристалле;
- отладка программной и аппаратной части разрабатываемой системы методом временного (полного) моделирования;
- формирование конфигурационной последовательности ПЛИС, соответствующей проекту разрабатываемой системы;
- программирование ПЛИС (загрузка проекта разработанной микропроцессорной системы в кристалл) или конфигурационного ПЗУ/ППЗУ.

Маршрут проектирования встраиваемых микропроцессорных систем в общем виде инвариантен по отношению к семействам ПЛИС, используемых для их реализации. При этом следует учитывать, что только содержание и выполнение начальных этапов разработки (создание нового проекта и подготовка исходных описаний программной и аппаратной части проектируемой системы) не зависит от выбранного типа ПЛИС. Содержание последующих этапов (синтеза, функционального и временного моделирования, размещения, трассировки и загрузки проекта в кристалл) различается для ПЛИС семейств CPLD и FPGA. С особенностями выполнения этих этапов при использовании микросхем серий CPLD можно ознакомиться в другой работе автора [9].

В наглядном виде типовой маршрут проектирования встраиваемых систем на основе ядер PicoBlaze показан на рис. 1. Рассматривая последовательность выполнения этапов, следует обратить внимание на то, что этапы разработки программной части проектируемой системы могут выполняться как до, так и после создания исходных описаний аппаратной части. Далее приводятся краткие пояснения по содержанию перечисленных выше этапов проектирования встраиваемых систем на основе микропроцессорных ядер семейства PicoBlaze.

www.finestreet.ru

Прежде чем приступить к созданию нового проекта, следует определиться с выбором метода описания микропроцессорной системы и, соответственно, средств синтеза. Исходная информация о проектируемой системе может быть представлена в виде принципиальных схем, описаний на языке HDL, диаграмм состояний, пакетов и библиотек пользователя. Учитывая, что все компоненты микропроцессорного ядра PicoBlaze выполнены на языках VHDL и Verilog, рекомендуется использовать их и для описания разрабатываемой системы. При этом целесообразно выбирать тот язык HDL, в терминах которого представлены описания компонентов микропроцессорного ядра. Так как наиболее распространенным языком описания компонентов микропроцессорных ядер семейства PicoBlaze является VHDL (все варианты ядер содержат соответствующие VHDL-описания), то далее рассматривается процесс разработки встраиваемых систем с использованием именно этого языка HDL в качестве основного.

При определении семейства и типа ПЛИС (и соответствующей версии ядра PicoBlaze) для реализации проекта микропроцессорной системы необходимо не только оценить ее сложность с учетом требований, предъявляемых к быстродействию, потребляемой мощности, условиям эксплуатации, но и учесть дополнительные факторы, такие как стоимость и возможность перепрограммирования в системе. Выбранное семейство или тип кристалла при необходимости достаточно легко можно изменить в процессе проектирования. При этом следует помнить, что в случае смены семейства ПЛИС (и соответственно версии ядра PicoBlaze) потребуется коррекция разработанной ранее программы. В процессе разработки архитектуры встраиваемой микропроцессорной системы, реализуемой на основе ядра PicoBlaze в ПЛИС фирмы Xilinx, определяется ее конфигурация и состав периферийных устройств.

Разработка программной части микропроцессорной системы включает в себя подготовку исходных текстов программ на языке ассемблера и их последующую трансляцию. Эти процедуры подробно рассмотрены в другой статье цикла [6].

На этапе подготовки описания проектируемой системы кроме формирования исходных текстов на языке HDL нужно также установить временные и топологические ограничения, которые должны учитываться при синтезе, размещении и трассировки проекта микропроцессорной системы в кристалле. В процессе синтеза на основании исходных модулей проекта формируется список соединений (netlist), содержащий набор примитивов или компонентов, который может быть реализован на основе ресурсов выбранного кристалла ПЛИС. Результаты синтеза используются далее в качестве исходных данных средствами размещения и трассировки. Отладка разрабатываемой системы, выполняемая методом функционального моделирования, производится без учета реальных значений задержек прохождения сигналов и позволяет проконтролировать соответствие выходных сигналов алгоритму работы микропроцессорной программы. На этапе размещения и трассировки проекта в кристалл производится распределение выполняемых функций в конфигурируемые логические блоки CLB (Configurable Logic Block) или макроячейки Macrocell (в зависимости от используемого семейства ПЛИС) и формирование необходимых связей в кристалле. В процессе выполнения этого этапа проектирования также определяются реальные значения задержек распространения сигналов, которые необходимы для полного временного моделирования системы. Основным результатом этапа размещения и трассировки является формирование файла, в котором содержится информация о конфигурации ПЛИС, реализующей проектируемую систему. Далее на основе этого файла создается конфигурационная битовая последовательность, предназначенная для программирования ПЛИС, или файл прошивки конфигурационного ПЗУ/ППЗУ. Завершением процесса разработки системы является загрузка конфигурационных данных в кристалл или программирование конфигурационного ПЗУ/ППЗУ с помощью соответствующих программ и загрузочного кабеля.

Следует обратить внимание на то, что этапы функционального и временного моделирования не являются обязательными. Тем не менее, использование высокоэффективных средств моделирования, включаемых в состав пакетов САПР фирмы Xilinx, в качестве отладочных инструментов позволяет обнаружить большинство возможных ошибок в аппаратной и программной части и, тем самым, значительно сократить общее время разработки системы [9]. При обнаружении ошибок на любом из этих этапов, например, логических ошибок на этапе функционального моделирования, или при получении неудовлетворительных результатов временного моделирования следует вернуться на стадию разработки исходных описаний проекта и микропроцессорной программы, внести необходимые изменения и повторить последующие этапы.

Перед тем как перейти к описанию выполнения всех перечисленных выше этапов процесса проектирования встраиваемых микропроцессорных систем на основе ядер семейства PicoBlaze в рамках САПР серии Xilinx ISE, следует определить понятие проекта.

Структура проекта встраиваемой микропроцессорной системы на основе ядра семейства PicoBlaze в САПР серии Xilinx ISE

Под проектом встраиваемой микропроцессорной системы в САПР серии Xilinx ISE понимается совокупность модулей (файлов), которые содержат информацию, необходимую для выполнения всех этапов процесса разработки данной системы на базе ПЛИС фирмы Xilinx. В отличие от проекта цифрового устройства с аппаратной реализацией операций [9], проект микропроцессорной системы включает в себя дополнительно файлы, используемые в процессе разработки программных средств. В структуре проекта в САПР серии Xilinx ISE ISE можно выделить следующие группы модулей:

- исходные описания проектируемого устройства в графической или текстовой форме;
- модули временных и топологических ограничений проекта;
- исходные тексты микропроцессорных программ на языке ассемблера;
- результаты трансляции микропроцессорных программ на языке ассемблера, представленные в различных форматах;
- документация, сопровождающая проект;
- промежуточные результаты, используемые в качестве исходных данных для последующих шагов проектирования;
- отчеты о выполнении основных этапов проектирования;
- функциональная и временная модели проектируемой системы;
- описания тестовых воздействий, необходимых для моделирования разрабатываемой системы, в текстовом и графическом формате;
- результаты функционального и временного моделирования проектируемой системы в графической и текстовой форме;
- отчеты, формируемые вспомогательными средствами пакета;
- окончательные результаты проектирования, используемые для конфигурирования ПЛИС.

Все модули проекта располагаются в одном каталоге (папке), название которого совпадает с названием проекта. Изначально проект представлен только заголовком и модулем, в котором указываются параметры проекта. Затем к проекту добавляются модули исходного описания аппаратной части проектируемой системы и исходные тексты микропроцессорных программ. Далее, после выполнения каждого этапа процесса разработки, в проект включаются результаты, полученные на этом этапе, и соответствующий отчет. Кроме того, разработчик может включить в проект необходимую текстовую документацию.

Создание нового проекта системы на основе ядра семейства PicoBlaze в Xilinx ISE

Прежде чем приступить к созданию нового проекта в САПР серии Xilinx ISE рекомендуется сформировать папку для хранения файлов этого проекта. Для этого можно воспользоваться средствами операционной системы Windows (например, Проводником) или какой-либо управляющей оболочки

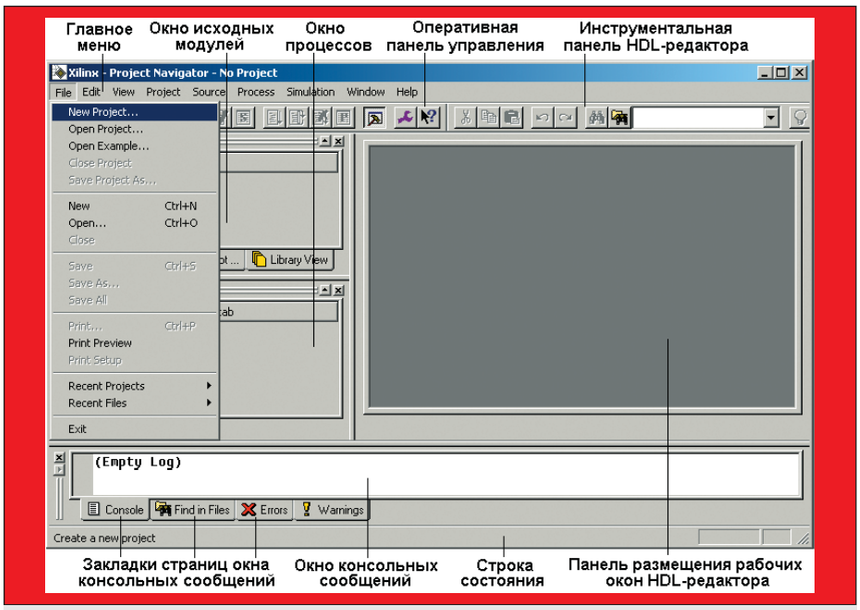

Рис. 2. Последовательность действий по созданию нового проекта в САПР серии Xilinx ISE

(файлового менеджера), например, Total Commander (Windows Commander). Имя создаваемой папки для нового проекта должно совпадать с его названием. Целесообразно хранить все проекты в специально созданном для этих целей каталоге, например, C:\Project. Каталог, в котором находятся все проекты пользователя, должен располагаться вне раздела, содержащего средства проектирования серии Xilinx ISE, чтобы при обновлении версии пакета САПР он не был удален.

В папку, предназначенную для хранения модулей нового проекта, нужно скопировать файлы исходных описаний используемой версии микропроцессорного ядра PicoBlaze, соответствующую версию ассемблера и шаблоны описания содержимого программной памяти. После этого следует в этой же папке сформировать файл, содержащий исходный текст микропроцессорной программы на языке ассемблера. Далее необходимо произвести трансляцию разработанной программы, при успешном завершении которой будут сгенерированы файлы описания содержимого программной памяти. Выполнение этих процессов подробно рассмотрено ранее [6].

Все последующие этапы разработки микропроцессорной системы выполняются в среде САПР серии Xilinx ISE. Поэтому далее необходимо открыть управляющую оболочку этого пакета *Навигатор проекта (Project Navigator)*, дважды щелкнув левой кнопкой мыши на пиктограмме  , расположенной на *Рабочем столе* ПК. При отсутствии данной пиктограммы можно воспользоваться кнопкой *Пуск (Start)*. После нажатия этой кнопки в открывшейся панели необходимо выбрать строку *Программы (Programs)*, а затем в предложенном списке выбрать группу программ Xilinx ISE 7.1i, в которой выделить строку *Project Navigator* и щелкнуть на ней левой кнопкой мыши. При успешном выполнении указанных операций на экране монитора отображается основное окно *Навигатора проекта*, вид которого представлен на рис. 2.

Для создания нового проекта в САПР серии Xilinx ISE следует выполнить команду *File* основного меню *Навигатора проекта*, а затем во всплывающем меню выбрать строку *New Project*, как показано на рис. 2.

В результате указанных действий запускается «мастер» формирования нового проекта New Project Wizard. Работа этого «мастера» начинается с вывода на экран диалоговой панели, вид которой показан на рис. 3.

В стартовой диалоговой панели «мастера» New Project Wizard должны быть определены следующие исходные данные, необходимые для создания нового проекта:

- название проекта;
- диск и каталог, в котором должен располагаться проект;
- тип (способ описания) модуля верхнего уровня иерархии проекта.

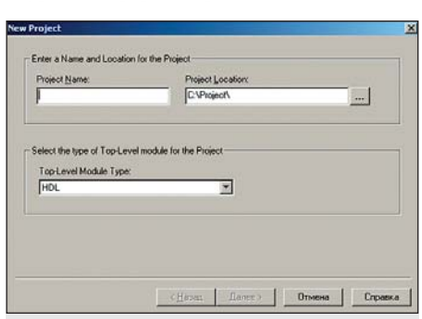

Рис. 3. Стартовая диалоговая панель «мастера» формирования нового проекта New Project Wizard в САПР серии Xilinx ISE

В первую очередь рекомендуется определить раздел (папку), в котором будет располагаться рабочий каталог проекта. Место расположения проекта на диске указывается в поле редактирования *Project Location* (рис. 3). По умолчанию в поле редактирования *Project Location* предлагаются диск и каталог, которые использовались в предыдущем проекте. Изменить расположение создаваемого проекта можно двумя способами: используя клавиатуру или стандартную панель навигации по дискам компьютера. При выборе первого метода следует поместить курсор на поле редактирования *Project Location* и щелкнуть левой кнопкой мыши, после чего ввести с клавиатуры имя диска и каталога. Если указываемый каталог отсутствует на диске, то он создается автоматически. Для выбора каталога, который уже существует, удобнее воспользоваться вторым способом, нажав кнопку с пиктограммой «...», расположенную справа от поля редактирования *Project Location*. В открывшейся диалоговой панели навигации следует с помощью мыши выбрать требуемый диск и каталог, а затем подтвердить сделанный выбор нажатием клавиши OK. После закрытия панели навигации выбранные параметры автоматически отображаются в поле редактирования *Project Location*.

Чтобы задать имя создаваемого проекта, необходимо поместить указатель мыши на поле редактирования *Project name* (рис. 3) и щелкнуть левой кнопкой мыши, после чего ввести с клавиатуры соответствующее название. Рекомендуется задавать мнемонические имена проектов, чтобы в последствии было удобнее ориентироваться при поиске требуемого проекта. Введенное название проекта автоматически добавляется в поле *Project Location*, определяя тем самым название рабочего каталога проекта.

Тип (способ описания) модуля верхнего уровня иерархии проекта определяется с помощью поля выбора *Top-Level Module Type*. Список, который открывается при нажатии кнопки, расположенной в правой части этого поля выбора, содержит четыре варианта: *HDL*, *Schematic*, *EDIF* и *NGC/NGO*. При использовании языков высокого уровня VHDL и Verilog для описания модуля верхнего уровня иерархии проекта в качестве значения параметра *Top-Level Module Type* устанавливается вариант *HDL*. Если исходный модуль верхнего уровня иерархии проекта будет представлять собой схему, выполненную в схемотехническом редакторе САПР серии Xilinx ISE, то следует выбрать вариант *Schematic*. В случаях, когда в качестве модуля верхнего уровня иерархии предполагается использовать описания в формате *EDIF* или *NGC/NGO*, должны быть указаны соответствующие варианты значения параметра *Top-Level Module Type*.

Установка значений всех необходимых параметров создаваемого проекта завершается

www.finestreet.ru

нажатием клавиши *Далее (Next)*, которая находится в нижней части диалоговой панели (рис. 3). Следует обратить внимание на то, что эта клавиша становится доступной только после определения значений всех перечисленных выше параметров. Если значение какого-либо параметра не задано, то данная клавиша остается в неактивном состоянии (ее название отображается серым цветом).

Окончание следует.

Литература

1. Зотов В. PicoBlaze — семейство восьмиразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Компоненты и технологии. 2003. № 4.
2. Зотов В. Система команд микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E // Компоненты и технологии. 2003. № 5.
3. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства Virtex-II // Компоненты и технологии. 2003. № 6.
4. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II // Компоненты и технологии. 2003. № 7.
5. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO // Компоненты и технологии. 2005. № 5–6.
6. Зотов В. Разработка программ на языке ассемблера для семейства микропроцессорных ядер PicoBlaze // Компоненты и технологии. 2003. № 8.
7. Зотов В. Инструментальный комплект Spartan-3 Starter Kit для практического освоения методов проектирования встраиваемых микропроцессорных систем на основе ПЛИС семейств FPGA фирмы Xilinx // Компоненты и технологии. 2005. № 7.
8. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. М.: Горячая линия-Телеком. 2004.
9. Зотов В. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. М.: Горячая линия — Телеком. 2003.
10. Зотов В. Средства программирования ПЛИС семейства CoolRunner-II фирмы Xilinx // Схемотехника. 2004. № 12.

KiT#53(9).qxd 11/14/2005 1:43 PM Page 6

Начало в № 7'2005

Создание проекта встраиваемой системы на основе микропроцессорного ядра семейства PicoBlaze

Валерий ЗОТОВ
walerry@km.ru

После нажатия клавиши *Далее (Next)* на экран выводится очередная диалоговая панель «мастера» New Project Wizard, с помощью которой определяются следующие параметры нового проекта:

- семейство ПЛИС, на базе которого разрабатывается микропроцессорная система;
- тип кристалла, выбираемого для реализации микропроцессорной системы;
- тип корпуса ПЛИС, выбранной для реализации проектируемой системы;
- категория быстродействия используемого кристалла;
- применяемые средства синтеза и моделирования проектируемой системы;
- язык HDL, используемый для формирования моделей разрабатываемой системы.

Вид этой диалоговой панели представлен на рис. 4.

Семейство ПЛИС, тип кристалла, корпуса, категория быстродействия, средства синтеза и моделирования, а также язык HDL, используемый для описания моделей, устанавливаются в форме таблицы параметров проекта (рис. 4). В первом столбце этой таблицы отображаются названия параметров (*Property Name*), а во втором — их значения (*Value*). Каждая ячейка столбца *Value* представляет собой поле выбора значения соответствующего параметра. Чтобы установить требуемое значение какого-либо параметра, следует активизировать соответствующее поле выбора, поместив на него курсор и щелкнув левой кнопкой мыши. После этого в правой части этого поля появляется кнопка управления выпадающим списком. При нажатии на нее отображается список возможных значений соответствующего параметра. Выбор требуемого значения осуществляется щелчком левой кнопки мыши на нужной строке. После выбора одной из строк выпадающего списка указанное в ней значение автоматически отображается в поле выбора.

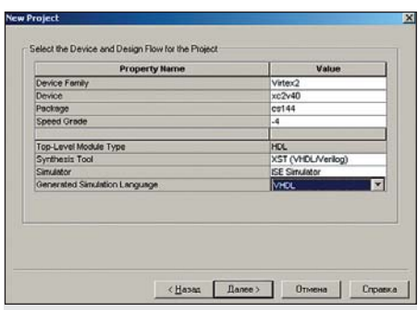

Рис. 4. Диалоговая панель выбора ПЛИС, средств синтеза и моделирования проектируемой микропроцессорной системы

Для определения семейства ПЛИС, на базе которого проектируется микропроцессорная система, следует воспользоваться полем выбора параметра *Device Family* (рис. 4). При нажатии на кнопку управления выпадающим списком этого поля на экране отображается список семейств ПЛИС, поддерживаемых используемой версией средств проектирования серии Xilinx ISE.

В поле выбора типа кристалла для реализации проектируемой микропроцессорной системы *Device* автоматически отображается тип ПЛИС, установленный по умолчанию для выбранного семейства. Для его изменения необходимо в выпадающем списке для параметра *Device* выделить строку, содержащую требуемый тип кристалла. Если разработчик затрудняется определить тип кристалла, необходимый для реализации проектируемой системы, то следует использовать автоматический выбор типа кристалла. Для этого в списке ПЛИС необходимо выбрать строку *Auto* для требуемого семейства ПЛИС. Программы трассировки определят кристалл с минимальным количеством ресурсов, необходимых для реализации разрабатываемой системы.

Тип корпуса кристалла, выбранного для реализации микропроцессорной системы, указывается в поле выбора *Package*. Если в качестве типа кристалла задано значение *Auto*, то в этом поле отображается символ «*», соответствующий режиму автоматического выбора типа корпуса. В этом случае также можно указать конкретный тип корпуса с неопределенным количеством выводов. Для этого следует выбрать значение *<тип корпуса>** из выпадающего списка, который появляется при щелчке левой кнопкой мыши в поле выбора *Package*. Если в поле *Device* указан определенный тип ПЛИС, то выпадающий список значений поля выбора *Package* содержит обозначения только тех типов корпусов, в которых выпускается данный кристалл.

Для определения категории быстродействия выбранного кристалла предназначено поле выбора параметра *Speed Grade*. Выпадающий список этого поля отображает градации быстродействия для выбранного типа ПЛИС. Если тип кристалла не конкретизирован (в поле *Device* указано значение *Auto*), то список содержит единственное значение «*», соответствующее режиму автоматического выбора категории быстродействия ПЛИС.

Набор поддерживаемых средств синтеза определяется выбранным семейством ПЛИС. Чтобы просмотреть этот набор и при необходимости изменить средства синтеза, предлагаемые по умолчанию, следует воспользоваться полем выбора *Synthesis Tool* (рис. 4). Состав поддерживаемых средств синтеза зависит также от используемой конфигурации средств проектирования серии Xilinx ISE. *Навигатор проекта* автоматически корректирует содержимое выпадающего списка инструментов синтеза в соответствии с конфигурацией САПР и выбранным семейством ПЛИС. По умолчанию предлагаются встроенные средства синтеза САПР серии Xilinx ISE — Xilinx Synthesis Technology (XST). Кроме того, разработчик может использовать следующие инструменты синтеза третьих фирм: системы Synplify или Synplify Pro, предлагаемые корпорацией Synplicity (**http://www.synplicity.com**), и систему LeonardoSpectrum фирмы Mentor Graphics (**http://www.mentor.com**). Соответствующие средства синтеза должны быть предварительно установлены на компьютере для работы под управлением САПР серии Xilinx ISE. В этом случае для указания требуемого инструмента синтеза следует выбрать строку с его названием в выпадающем списке параметра *Synthesis Tool*.

Для определения средств моделирования проектируемой системы нужно воспользоваться полем выбора параметра *Simulator*. Конфигурации системы проектирования ISE BaseX и ISE Foundation обладают собственными средствами моделирования ISE Simulator. Кроме того, все конфигурации средств проектирования поддерживают систему моделирования ModelSim. Для выбора одного из этих инструментов моделирования необходимо в выпадающем списке возможных значений параметра *Simulator* выделить стро-

KiT#53(9).qxd 11/14/2005 1:43 PM Page 7

ку с его названием. В случае использования других систем моделирования следует выбрать строку *Other*.

Параметр *Generated Simulation Language* позволяет определить язык HDL, используемый для автоматического формирования моделей проектируемой системы. В выпадающем списке значений этого параметра представлено два значения — *VHDL* и *Verilog*, которые соответствуют языкам высокого уровня, поддерживаемым средствами синтеза. В процессе генерации моделей разрабатываемой микропроцессорной системы целесообразно использовать тот же язык *HDL*, что был выбран для создания модулей исходного описания верхнего уровня иерархии и ее компонентов.

После определения значений всех параметров в диалоговой панели, показанной на рис. 4, следует нажать клавишу *Далее (Next)*, которая находится в нижней части этой панели. В результате на экране отображается следующая диалоговая панель «мастера» формирования нового проекта New Project Wizard, вид которой показан на рис. 5.

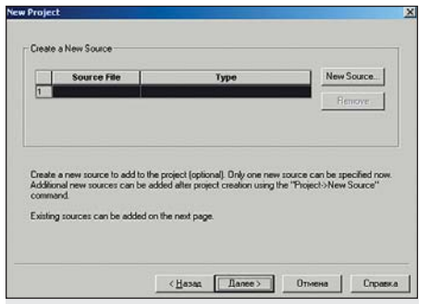

Рис. 5. Диалоговая панель создания нового исходного модуля проекта

Данная диалоговая панель предоставляет возможность создания нового модуля исходного описания микропроцессорной системы и включения его в состав проекта. Чтобы сформировать основу нового модуля исходного описания, следует нажать кнопку *New Source* (рис. 5). В результате выполненных действий открывается диалоговая панель, представленная на рис. 6, в которой необходимо выбрать тип нового модуля, задать его имя и указать место расположения создаваемого файла на диске. Новые модули исходного описания проектируемой микропроцессорной системы могут создаваться в любой последовательности. Целесообразно в первую очередь приступить к формированию модуля описания для верхнего уровня иерархии разрабатываемой системы.

В диалоговой панели, показанной на рис. 6, прежде всего, рекомендуется установить тип создаваемого исходного модуля, для чего в списке (рис. 6) следует выделить соответствующую строку, щелкнув на ней левой кнопкой мыши. Содержание списка возможных типов исходных модулей зависит от выбранного семейства ПЛИС и средств синтеза,

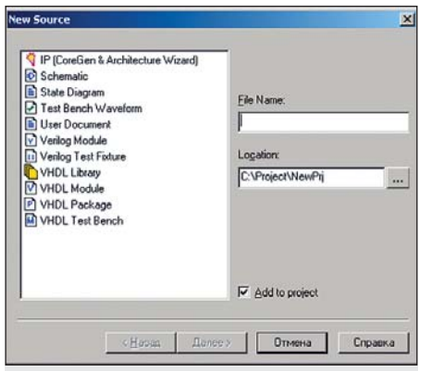

Рис. 6. Диалоговая панель установки параметров нового исходного модуля проекта

используемых в проекте. Учитывая, что все компоненты ядра выполнены в виде VHDL-описаний, следует выбрать в предложенном списке в качестве типа создаваемого модуля *VHDL Module*. Затем нужно активизировать поле редактирования названия модуля (файла) *File Name*, поместив на него курсор и щелкнув левой кнопкой мыши. Ввод имени файла осуществляется с помощью клавиатуры. Расширение имени файла устанавливается автоматически в соответствии с выбранным типом модуля. Место расположения создаваемого модуля на диске указывается в поле редактирования *Location* диалоговой панели (рис. 6). По умолчанию предлагается рабочий каталог формируемого проекта. Для создаваемых модулей исходного описания проекта рекомендуется использовать именно этот каталог. Особое внимание необходимо обратить на состояние индикатора автоматического включения модуля в состав проекта *Add to project*. Если флаг индикатора установлен (поле индикатора помечено маркером), то созданный модуль автоматически включается в состав формируемого проекта. По умолчанию флаг индикатора находится в установленном состоянии. Для модификации этого параметра достаточно щелкнуть левой кнопкой мыши, поместив курсор на поле индикатора. При этом состояние индикатора изменится на противоположное.

Установка значений всех необходимых параметров создаваемого модуля завершается нажатием клавиши *Далее (Next)*, которая находится в нижней части диалоговой панели (рис. 6). Если создается основа VHDL-модуля, то далее открывается диалоговая панель определения исходных данных, необходимых для автоматической генерации шаблона VHDL-описания. Вид этой диалоговой панели показан на рис. 7.

В поле редактирования *Entity Name* после его активизации необходимо указать имя описываемого объекта. По умолчанию предлагается идентификатор, совпадающий с названием создаваемого модуля. Имя архитектурного тела VHDL-описания указывается в поле редактирования *Architecture Name*. По умолчанию в качестве такого имени предлагается идентификатор *Behavioral*. Далее следует заполнить таблицу описания портов, которая содержит четыре столбца. Ячейки первого столбца представляют собой поле редактирования, в которое с помощью клавиатуры заносится идентификатор порта *Port Name*. Во второй колонке указывается тип порта *Direction*. Каждая ячейка этого столбца представляет собой поле выбора, выпадающий список которого содержит три значения, определяющие тип порта: *in* (входной), *out* (выходной) или *inout* (двунаправленный). Колонки *MSB* и *LSB* заполняются только для портов, представленных в виде шин и описываемых с помощью векторов. В столбце *MSB* указывается значения индекса соответствующего старшему разряду вектора, а в *LSB* — младшему. Если описание портов нового модуля не помещается в видимой части таблицы, следует воспользоваться элементами вертикальной прокрутки, расположенными вдоль правой границы таблицы. После внесения всех необходимых данных следует нажать кнопку *Далее (Next)* (рис. 7), в результате чего открывается панель с информацией, на основе которой выполняется формирование нового модуля VHDL-описания (рис. 8).

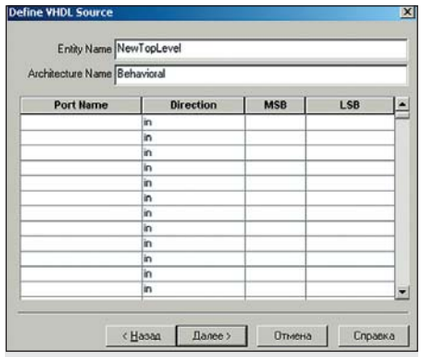

Рис. 7. Диалоговая панель определения исходных данных для формирования шаблона VHDL-описания

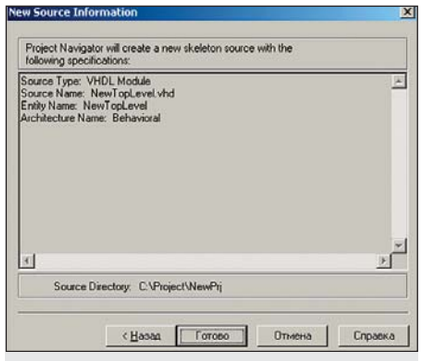

Рис. 8. Информационная панель «мастера» подготовки шаблона VHDL-описания, отражающая значения параметров создаваемого исходного модуля проекта

KiT#53(9).qxd 11/14/2005 1:43 PM Page 8

Если все данные, необходимые для создания основы нового VHDL-описания, указаны корректно, то далее нужно нажать кнопку *Готово (Finish)* в нижней части информационной панели (рис. 8), в результате чего открывается новое рабочее окно встроенного HDL-редактора *Навигатора проекта*, в котором отображается автоматически сформированный код. Этот код включает декларацию используемых библиотек и пакетов, интерфейса описываемого объекта entity и основу архитектурного тела VHDL-описания. Для получения законченного описания на языке VHDL необходимо основу архитектурного тела дополнить кодом, описывающим внутреннюю структуру или поведение объекта. Текст описания вводится с помощью клавиатуры или шаблонов встроенного HDL-редактора САПР серии Xilinx ISE. После завершения формирования модуля текстового описания следует сохранить его в виде файла на диске, используя команды Save или Save As из всплывающего меню File или кнопку, расположенную на оперативной панели управления *Навигатора проекта*.

Одновременно с открытием окна встроенного HDL-редактора на экран вновь выводится диалоговая панель создания нового исходного модуля, в которой отображается название этого модуля (рис. 5). В этой панели нужно нажать клавишу *Далее (Next)*, после чего на экране появляется следующая диалоговая панель «мастера» формирования нового проекта New Project Wizard, вид которой представлен на рис. 9.

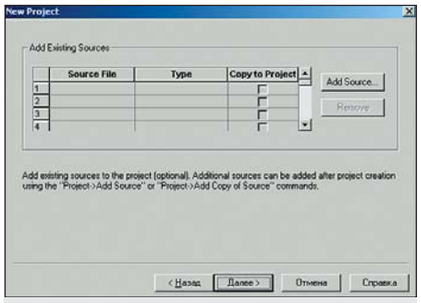

Рис. 9. Диалоговая панель включения существующих модулей исходного описания в состав создаваемого проекта

Эта диалоговая панель позволяет включить в состав формируемого проекта существующие модули исходного описания, в том числе и созданные ранее в рамках предыдущих проектов. Чтобы добавить существующий модуль в состав создаваемого проекта, следует воспользоваться кнопкой *Add Source* (рис. 9), после чего на экране отображается стандартная панель диалога открытия файла. В этой стандартной панели нужно выбрать соответствующий файл, подтвердив сделанный выбор нажатием клавиши *Открыть (Open)*, после чего название этого файла автоматически заносится в таблицу, расположенную в диалоговой панели. В первой колонке этой таблицы с названием Source File отображается название включаемого файла, во второй с названием *Type* — его тип. Третья колонка *Copy to project* содержит индикатор автоматического копирования модуля в состав проекта. Если флаг установлен (поле индикатора помечено маркером), то включаемый модуль автоматически копируется в рабочий каталог создаваемого проекта. Для изменения состояния индикатора на противоположное нужно поместить курсор на его поле и щелкнуть левой кнопкой мыши. Когда в состав проекта добавляется файл с расширением VHD, то на экран автоматически выводится диалоговая панель, позволяющая уточнить тип включаемого модуля (рис. 10).

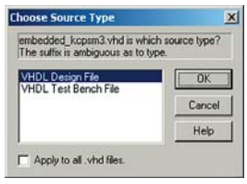

Рис. 10. Диалоговая панель выбора типа VHDL-модуля при включении его в состав создаваемого проекта

Если добавляемый файл с расширением VHD содержит исходное описание компонента проектируемой системы на языке VHDL, то в этой диалоговой панели нужно выбрать строку *VHDL Design File*. В том случае, если включаемый файл представляет собой описание тестового модуля, следует выбрать строку *VHDL Test Bench File*.

Рассмотренную процедуру нужно поочередно повторить для включения всех необходимых модулей в состав создаваемого проекта. Прежде всего, необходимо добавить в формируемый проект модули исходного описания микропроцессорного ядра PicoBlaze. Эти файлы были скопированы в рабочий каталог проекта сразу после его создания, но не были включены в проект. Поэтому для них индикатор автоматического копирования модуля в состав проекта, расположенный в колонке *Copy to project*, может находиться в выключенном состоянии. Кроме того, нужно не забыть добавить в состав проекта файл описания содержимого программной памяти, соответствующий разработанной микропроцессорной программе. Этот файл с именем *<название_микропроцессорной_программы>.vhd* должен быть автоматически сформирован ассемблером в процессе трансляции исходного текста программы. Когда все требуемые модули будут добавлены в состав формируемого проекта, следует нажать кнопку *Далее (Next)*, расположенную в нижней части диалоговой панели, представленной на рис. 9. В результате на экран выводится информационная панель, в которой отображаются установленные

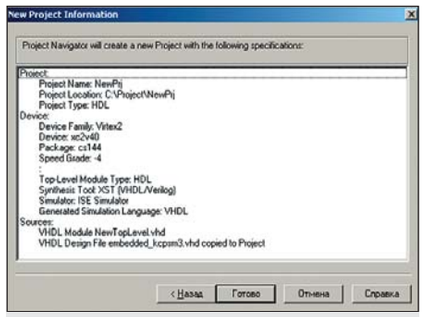

Рис. 11. Информационная панель «мастера» формирования нового проекта New Project Wizard

значения основных параметров создаваемого проекта. Вид этой панели показан на рис. 11.

Если необходимо изменить значение какого-либо параметра, то кнопка *Назад (Back)*, расположенная в нижней части информационной панели, позволяет вернуться к предыдущей диалоговой панели. Для завершения процесса формирования нового проекта следует нажать кнопку *Готово (Finish)* в нижней части информационной панели (рис. 11). После этого созданный проект автоматически открывается в рабочей области *Навигатора проекта*. При этом в области расположения рабочих окон отображается подробная информация о новом проекте (рис. 12).

На следующем шаге формирования проекта микропроцессорной системы, при необходимости, нужно сформировать недостающие модули исходного описания. (Например, HDL-описания компонентов проектируемой системы.) Для создания основы нового модуля исходного описания проекта следует нажать кнопку на оперативной панели, которая дублирует команду *New Source* из раздела *Project* основного меню *Навигатора проекта*. После этого на экран выводится диалоговая панель установки параметров нового исходного модуля проекта (рис. 6). Далее необходимо выполнить ту же последовательность действий, что и при создании модуля с помощью «мастера» формирования нового проекта New Project Wizard, которая была рассмотрена выше.

Кроме модулей исходного описания разрабатываемой микропроцессорной системы в состав проекта в большинстве случаев необходимо включить файл временных и топологических ограничений User Constraints File (UCF). Данный файл содержит дополнительную информацию для программ синтеза, размещения и трассировки, в частности информацию о привязке внешних цепей разрабатываемой микропроцессорной системы к выводам ПЛИС [9]. Поэтому остановимся на изучении только двух, наиболее часто используемых, типов ограничений.

Параметр LOC позволяет установить соответствие между внешними цепями проектируемой системы и номерами выводов ПЛИС, а также явно указать конфигурируемый ло-

KiT#53(9).qxd 11/14/2005 1:43 PM Page 9

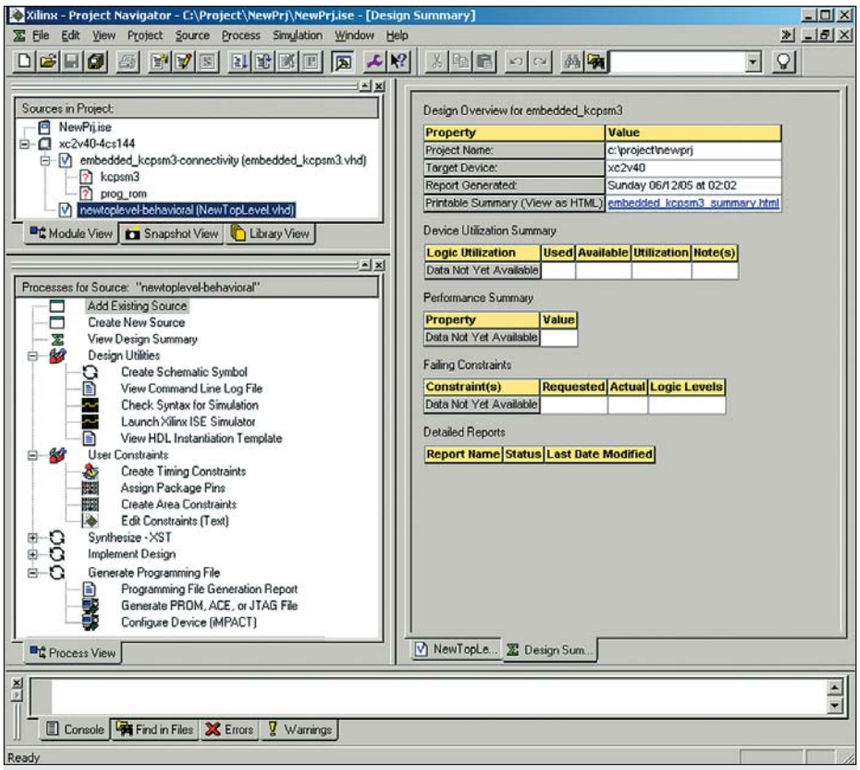

Рис. 12. Вид основного окна *Навигатора проекта* после создания нового проекта

гический блок (Configurable Logic Block, CLB) для реализации элементов проекта. Для привязки интерфейсных цепей проектируемой системы (подключаемых к контактам кристалла) к требуемым выводам ПЛИС используется следующий формат выражения ограничения:

```
NET <название_цепи> LOC=<номер_вывода_ПЛИС>;
```

Форма записи номера вывода ПЛИС в этом выражении зависит от типа используемого корпуса микросхемы. Для корпусов, относящихся к категории BGA (Ball Grid Array), номер вывода указывается в том же виде, в каком он представлен в документации. При использовании корпусов других типов номер вывода ПЛИС, указанный в документации, в выражении ограничения LOC сопровождается префиксом P. В качестве примеров топологических ограничений, используемых для указания соответствия внешних цепей разрабатываемой микропроцессорной системы и номеров выводов ПЛИС, приводятся следующие выражения

```
NET clk_s LOC=C5;
NET date1 LOC=P15;
```

Максимальное значение периода сигнала синхронизации для соответствующей цепи разрабатываемой микропроцессорной системы задается с помощью параметра *PERIOD*. Полный формат соответствующего выражения ограничения имеет вид:

```
NET <название_цепи_синхронизации> PERIOD=<длительность_периода> [<единицы_измерения>] [{HIGH | LOW} [<длительность_первой_фазы_периода> [<единицы_измерения>] ]],
```

Здесь значение HIGH или LOW указывает логический уровень сигнала в первой фазе периода, а элементы, указанные в квадратных скобках являются необязательными. По умолчанию установлены в качестве единиц измерения длительности наносекунды нс (ns) и одинаковая продолжительность состояний высокого и низкого уровня периода синхросигнала, в результате чего получается сокращенный формат записи:

```
NET <название_цепи_синхронизации> PERIOD =<длительность_периода> ;

Например, NET clock PERIOD=25ns;.
```

Таким образом, значение параметра *PERIOD* накладывает ограничение на время распространения сигналов по цепям и логике, подключенных между выходом одного и входом другого синхронного элемента (триггера, регистра или ОЗУ), которые тактируются одним и тем же сигналом синхронизации.

Для создания основы файла UCF следует воспользоваться кнопкой на оперативной панели или командой *New Source* из раздела *Project* основного меню *Навигатора проекта*.

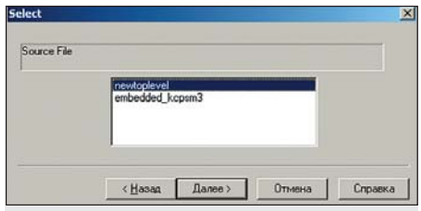

Рис. 13. Диалоговая панель выбора модуля исходного описания проекта, с которым ассоциируется создаваемый файл UCF

После этого в открывшейся диалоговой панели установки параметров нового модуля следует выделить строку *Implementation Constraints File* (поместив на нее курсор и щелкнув левой кнопкой мыши), которая в качестве типа нового модуля задает файл временных и топологических ограничений проекта (рис. 6). Затем, в этой же панели, нужно указать название нового файла UCF в поле редактирования *File Name* и нажать кнопку *Далее (Next)*. После этого на экран выводится следующая панель диалога, которая позволяет выбрать модуль исходного описания проекта, с которым ассоциируется создаваемый файл временных и топологических ограничений (рис. 13). Для этого следует расположить курсор мыши на требуемой строке списка, отображаемого в диалоговой панели, и щелкнуть левой кнопкой. Как правило, выбирается модуль описания верхнего уровня иерархии проекта.

Если все данные, необходимые для создания нового файла UCF, указаны корректно, то далее нужно нажать кнопку *Готово (Finish)* в нижней части информационной панели (рис. 13). Кнопка *Назад (Back)* позволяет при необходимости вернуться к предыдущим шагам его создания. При успешном завершении рассмотренного процесса в окне исходных модулей *Навигатора проекта* появляется строка с названием нового файла ограничений.

Для внесения информации в файл UCF можно использовать встроенный HDL-редактор или специальную программу Constraints Editor, которая на основании данных, указанных разработчиком в диалоговом режиме, автоматически формирует соответствующие выражения для описания ограничений проекта. Кроме того, в проектах, выполняемых на основе ПЛИС семейств FPGA, для этих целей может использоваться редактор назначения выводов кристалла и топологических ограничений PACE (Pinout and Area Constraints Editor). Все перечисленные программы входят в состав средств проектирования серии Xilinx ISE. Чтобы приступить к редактированию файла UCF, необходимо в окне исходных модулей *Навигатора проекта* щелчком левой кнопки мыши выделить строку с его названием, после чего в окне процессов развернуть строку «User Constraints» (рис. 12). Для изменения файла UCF во встроенном

KiT#53(9).qxd 11/14/2005 1:43 PM Page 10

текстовом редакторе следует дважды щелкнуть левой кнопкой мыши на строке *Edit Constraints (Text)*, в результате чего открывается новое окно редактирования. Ввод выражений временных и топологических ограничений осуществляется с помощью клавиатуры. При этом рекомендуется использовать шаблоны HDL-редактора, представленные в папке UCF. Чтобы выполненные изменения вступили в силу, необходимо сохранить файл временных и топологических ограничений на диске, воспользовавшись командой *Save* из всплывающего меню *File* или кнопкой  , расположенной на оперативной панели управления *Навигатора проекта*.

После того, как сформированы все необходимые исходные модули проекта, следует перейти к выполнению основных этапов разработки встраиваемой микропроцессорной системы. Выполнение этих в САПР серии Xilinx ISE будет рассмотрено в следующей публикации цикла.

Литература

1. Зотов В. PicoBlaze — семейство восьмиразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx. // Компоненты и технологии. 2003. № 4.
2. Зотов В. Система команд микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E // Компоненты и технологии. 2003. № 5.
3. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства Virtex-II // Компоненты и технологии. 2003. № 6.
4. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II // Компоненты и технологии. 2003. № 7.
5. Зотов В. Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейств Spartan-3, Virtex-II и Virtex-IIPRO // Компоненты и технологии. 2005. № 5-6.
6. Зотов В. Разработка программ на языке ассемблера для семейства микропроцессорных ядер PicoBlaze // Компоненты и технологии. 2003. № 8.
7. Зотов В. Инструментальный комплект Spartan-3 Starter Kit для практического освоения методов проектирования встраиваемых микропроцессорных систем на основе ПЛИС семейств FPGA фирмы Xilinx // Компоненты и технологии. 2005. № 7.
8. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. М.: Горячая линия-Телеком. 2004.
9. Зотов В. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. М.: Горячая линия — Телеком. 2003.
10. Зотов В. Средства программирования ПЛИС семейства CoolRunner-II фирмы Xilinx // Схемотехника. 2004. № 12.