PicoBlaze — семейство восьмиразрядных микропроцессорных ядер,

реализуемых на основе ПЛИС фирмы Xilinx

Этой публикацией мы открываем цикл статей, в которых рассматриваются встраиваемые микропроцессорные модули (ядра), предназначенные для применения в составе проектов, реализуемых на основе ПЛИС фирмы Xilinx.

Валерий Зотов

walerry@euro.ru

Современные семейства программируемых логических интегральных схем (ПЛИС) предоставляют широкие возможности для реализации проектируемого устройства на базе одного кристалла. Это обусловлено, в первую очередь, внедрением новых технологий производства, позволяющих значительно увеличить объем логических и трассировочных ресурсов кристаллов. При выполнении разработки «системы на кристалле» (System-on-Chip), реализующей в одном корпусе ПЛИС функции процессора и периферийных устройств, целесообразно использовать готовые микропроцессорные ядра. Их применение позволяет ощутимо сократить длительность цикла проектирования разрабатываемой системы.

Микропроцессорные ядра, предоставляемые фирмой Xilinx

Фирма Xilinx наряду с выпуском новых семейств ПЛИС, отличающихся высокими техническими характеристиками, предоставляет разработчикам готовые отлаженные модули микропроцессорных ядер с различной архитектурой. В рамках программы AllianceCORE пользователям доступны ядра с архитектурой широко применяемых микропроцессоров различных производителей, таких, как Z80 фирмы Zilog, PIC семейств 125x, 1655x, 165x фирмы Microchip, 8051 и др. Кроме того, фирма Xilinx предлагает семейства ядер с оригинальной архитектурой, оптимизированной для реализации на основе ПЛИС различных серий (Soft Processor). К числу таких ядер относятся семейства PicoBlaze и MicroBlaze. Общие характеристики микропроцессорных ядер этих семейств представлены в таблице.

Таблица. Общие характеристики микропроцессорных ядер семейств PicoBlaze и MicroBlaze

Тип микропроцессорного ядра	Архитектура	Разрядность шин	Производительность	Объем требуемых ресурсов	Поддерживаемые семейства ПЛИС	Средства разработки
PicoBlaze	RISC 8 разрядов	8 разрядов данных и адресов	40 MIPS 116 МГц	35 CLBs	Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, CoolRunner-II	Ассемблер
MicroBlaze	RISC 32 разряда	32 разряда данных и команд	100 D-MIPS 150 МГц	225 CLBs	Spartan-II, Spartan-IIE, Virtex, Virtex-E, Virtex-II, Virtex-II Pro	Development Kits (компилятор, ассемблер, отладчик)

Семейство микропроцессорных ядер PicoBlaze является свободно распространяемым (бесплатным). Для их получения следует обратиться к Web-странице ht tp://w ww. xilinx.com/ipcenter/processor_central/picoblaze/index.htm. Чтобы скачать все необходимые материалы, относящиеся к конкретному ядру этого семейства, необходимо выполнить процедуру бесплатной регистрации.

Изучение элементов семейства PicoBlaze начнем с ядра, предназначенного для применения в системах, выполняемых на основе ПЛИС серий Spartan-II, Spartan-IIE, Virtex, Virtex-E.

Основные характеристики микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E

Отличительными особенностями микропроцессорного ядра PicoBlaze, предназначенного для применения в ПЛИС семейства Spartan-II, Spartan-IIE, Virtex, Virtex-E являются:

- гибкая архитектура с раздельными шинами данных и команд;
- разрядность шины данных — 8 бит;
- разрядность шины адресов — 8 бит;
- разрядность шины команд — 16 бит;
- восьмиразрядное арифметическо-логическое устройство (АЛУ), реализующее логические функции, операции сложения, вычитания и сдвига;
- поддержка 49 команд;
- постоянное время выполнения всех команд — два машинных цикла;
- шестнадцать регистров общего назначения;
- пятнадцатиуровневый стек;

- возможность поддержки до 256 входных и выходных портов;
- встроенное ППЗУ микропрограмм, выполненное на основе блочной памяти ПЛИС Block SelectRAM, объем которого составляет 256×16 разрядов;
- поддержка прямого, косвенного и непосредственного режимов адресации;
- реализация в виде модулей исходного описания на языке VHDL с учетом оптимального размещения и трассировки в кристалле соответствующего семейства;
- минимальный объем ресурсов кристалла, используемый для реализации микропроцессорного ядра, позволяет без труда разместить в кристалле другие функциональные модули проектируемой системы, включая интерфейсы ввода-вывода (в ПЛИС семейства Spartan-IIE ядро PicoBlaze занимает всего лишь 76 секций (slices), что составляет 9% от полного объема логических ресурсов кристалла XC2S50E и 2,5% от логической емкости ПЛИС XC2S300E);
- интерфейс микропроцессорного ядра обеспечивает оптимальное его сопряжение с периферийными модулями, реализуемыми на основе свободных логических ресурсов кристалла;
- достаточно высокая производительность, достигающая 40 MIPS (в зависимости от типа используемого кристалла);
- наличие ассемблера, формирующего необходимые файлы различного формата, обеспечивает высокую скорость и наглядность процесса разработки программ;
- наличие входа сброса (инициализации), позволяющего перевести микропроцессор в начальное состояние;
- полная совместимость компонентов ядра со всей линией средств разработки проектов и программирования ПЛИС ISE (Integrated Synthesis Environment) фирмы Xilinx (WebPACK ISE [1–6, 10], Base ISE, Foundation ISE и Alliance ISE);
- возможность моделирования ядра в составе разрабатываемых проектов с помощью системы ModelSim XE [7–9], которая включена в состав свободно распространяемого пакета САПР WebPACK ISE.

Структура проекта микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E

Структура микропроцессорного ядра PicoBlaze, предназначенного для применения в составе проектов, выполняемых на основе ПЛИС семейства Spartan-II, Spartan-IIE, Virtex, Virtex-E, показана на рис. 1. Она включает в себя программную память и модуль центрального процессорного устройства (ЦПУ) (исполнительного устройства) PicoBlaze. В качестве ППЗУ микропрограмм используется один модуль блочной памяти Block SelectRAM кристаллов указанных семейств, объем которого составляет 4 кбит. Этот модуль блочной памяти конфигурируется как однопортовое ОЗУ с организацией 256×16 разрядов.

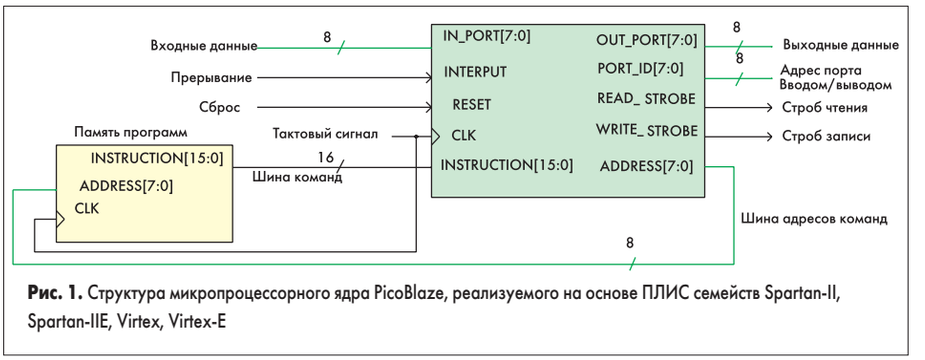

Рис. 1. Структура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E

Микропроцессорное ядро PicoBlaze предоставляется пользователям в виде архива, в котором содержится комплект файлов, включающий в себя необходимые модули VHDL-описаний, ассемблер, тестовые примеры, иллюстрирующие использование компонентов ядра. Кроме того, в состав комплекта входят дополнительные файлы, которые могут быть использованы для включения в состав проекта универсального асинхронного приемопередатчика UART (Universal Asynchronous Receiver-Transmitter).

Каждый элемент структуры микропроцессорного ядра PicoBlaze (рис. 1) представлен в форме соответствующего компонента, выполненного в виде макроса с относительным размещением. Функции исполнительного модуля реализует компонент KCPSM (Constant (k) Coded Programmable State Machine). Описание этого модуля на языке VHDL содержится в файле kcpsm.vhd. Для применения компонента KCPSM в качестве элемента проектируемой системы необходимо, прежде всего, включить в состав ее описания выражения декларации, которые выглядят следующим образом.

```
component kcpsm
    Port (
        address : out std_logic_vector(7 downto 0);
        instruction : in std_logic_vector(15 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;
        reset : in std_logic;
        clk : in std_logic
        );
end component;
```

В приведенных выражениях декларации используется следующая система обозначений интерфейсных цепей компонента. Идентификаторы clk, reset и interrupt описывают соответственно входы тактового сигнала, сброса и прерывания, а read_strobe и write_strobe соответствуют выходам сигналов, сопровождающих операции чтения и записи данных в порты ввода-вывода. Векторы in_port и out_port представляют соответственно входную и выходную шины данных. Вектор address соответствует выходной шине адресов команд, instruction — входной шине команд, port_id — выходной шине адресов портов ввода-вывода.

Для создания одного экземпляра компонента kcpsm, представляющего модуль ЦПУ, необходимо в состав структурного описания архитектуры проектируемой системы включить следующий оператор:

```
inst_name_processor: kcpsm
    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,
        reset => reset_signal,
        clk => clk_signal
        );
```

Вместо идентификатора *inst_name_processor* следует задать метку, определяющую позиционное обозначение экземпляра компонента. Кроме того, названия сигналов, указанные в операторе, должны соответствовать именам сигналов, которые используются в описании проектируемого устройства.

Таким же способом, в виде компонента с названием prog_rom, описывается модуль программной памяти в составе разрабатываемой системы. Вначале выполняется декларация этого компонента, которая выглядит следующим образом.

```
component prog_rom
    Port (
        address : in std_logic_vector(7 downto 0);
        instruction : out std_logic_vector(15 downto 0);
        clk : in std_logic
        );
end component;
```

В описании интерфейса компонента prog_rom идентификатор clk соответствует входу тактового сигнала, address — входной шине адресов команд, instruction — выходной шине команд. Создание экземпляра компонента, представляющего программную память, выполняется следующим оператором.

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

При практическом использовании этого компонента следует учитывать, что его название должно совпадать с идентификатором файла, имеющего расширение PSM, в котором записан исходный текст программы.

Помимо отдельных компонентов микропроцессорного ядра PicoBlaze пользователю также доступно VHDL-описание объекта EMBEDDED_KCPSM, который представляет собой подсистему, состоящую из модуля ЦПУ и подключенной к нему программной памяти. Описание этого объекта является наглядной иллюстрацией практического использования

компонентов микропроцессорного ядра, и поэтому далее полностью приводится его текст.

```
-- EMBEDDED_KCPSM.VHD
--
----------------------------------------------------------------------
--
-- 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_kcpsm 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;
                reset : in std_logic;
                  clk : in std_logic);
end embedded_kcpsm;
--
----------------------------------------------------------------------
--
-- Start of test achitecture
--
architecture connectivity of embedded_kcpsm is
--
----------------------------------------------------------------------
--
-- declaration of KCPSM
--
 component kcpsm
   Port (     address : out std_logic_vector(7 downto 0);
          instruction : in std_logic_vector(15 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;
                reset : in std_logic;
                  clk : in std_logic);
 end component;
--
-- declaration of program ROM
--
 component prog_rom
   Port (    address : in std_logic_vector(7 downto 0);
          instruction : out std_logic_vector(15 downto 0);
                  clk : in std_logic);
 end component;
--
----------------------------------------------------------------------
--
-- Signals used to connect KCPSM to program ROM
--
signal      address : std_logic_vector(7 downto 0);
signal instruction : std_logic_vector(15 downto 0);
--
----------------------------------------------------------------------
--
-- Start of test circuit description
--
begin

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

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

end connectivity;

----------------------------------------------------------------------
--
-- END OF FILE EMBEDDED_KCPSM.VHD
```

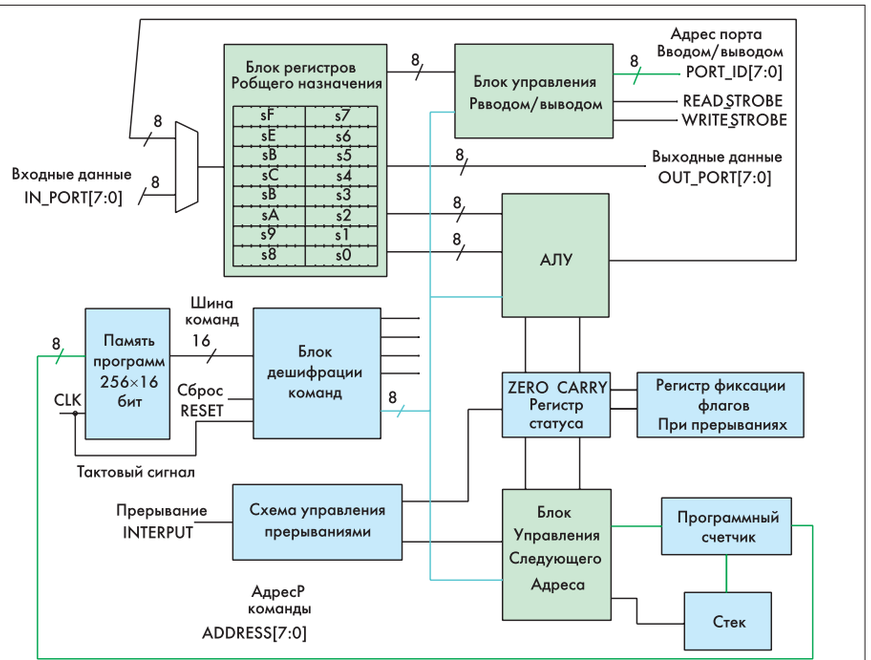

Рис. 2. Архитектура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E

Приведенное описание объекта EMBEDDED_KCPSM имеет стандартную структуру исходного модуля, выполненного с использованием языка VHDL. В начале описания указаны ссылки на используемые стандартные библиотеки и пакеты. Затем следуют операторы, описывающие интерфейс объекта. Далее приводится структурное описание архитектуры объекта.

Объект EMBEDDED_KCPSM может применяться как автономно, так и в виде модуля, входящего в состав разрабатываемой системы. В последнем случае модуль EMBEDDED_KCPSM используется в форме одноименного компонента. Для декларации компонента *embedded_kcpsm* в составе VHDL-представления проектируемой системы используется следующая конструкция.

```
component embedded_kcpsm
      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;
               reset : in std_logic;
               clk : in std_logic
               );
end embedded_kcpsm;
```

В описании интерфейса компонента *embedded_kcpsm* используется та же система обозначений, что и для компонента *kcpsm*. Конкретный экземпляр рассматриваемого компонента в структурном описании проектируемой системы создается с помощью оператора, текст которого представлен ниже. При практическом использовании этого оператора следует заменить метку *inst_name_embedded_processor* идентификатором, который, как правило, соответствует позиционному обозначению экземпляра компонента в составе описания проекта. При описании внешних связей компонента embedded_kcpsm должны быть указаны названия соответствующих сигналов, используемых в проекте.

```
inst_name_embedded_processor: embedded_kcpsm
      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,
               reset => reset_signal,
               clk => clk_signal
               );
```

Архитектура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E

Архитектура микропроцессорного ядра PicoBlaze основана на концепции раздельных шин данных и команд (гарвардская архитектура). Такая организация магистралей процессора позволяет добиться высокой скорости выполнения операций. Структурное представление архитектуры микропроцессорного ядра PicoBlaze, предназначенного для применения в кристаллах семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E, показано на рис. 2.

Основными элементами архитектуры микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E, являются:

- восьмиразрядное АЛУ;
- блок регистров общего назначения;
- регистр статуса (флагов);
- регистр фиксации флагов при выполнении прерываний;
- программный счетчик (счетчик команд);
- блок управления вводом-выводом;
- стек;
- схема управления прерываниями;
- блок управления выбором адреса следующей команды;

- блок дешифрации команд;
- память программ.

Блок АЛУ выполняет логические и арифметические операции над восьмиразрядными операндами. В качестве операндов может использоваться содержимое любого из шестнадцати регистров общего назначения, а также константы, указанные непосредственно в тексте команды. Результат выполнения операции заносится в тот же регистр, который являлся источником первого операнда.

Регистр статуса содержит значения флагов (признаков), формируемых блоком АЛУ при выполнении арифметических и логических операций. Этот регистр содержит два разряда. В первый разряд записывается состояние флага нулевого результата ZERO Flag, а во второй — значение флага переноса (заема) CARRY Flag. Флаг нулевого результата ZERO Flag переключается в установленное состояние (состояние высокого логического уровня) в случае, если результатом арифметической или логической операции является нуль. При получении других результатов этот флаг сбрасывается в состояние низкого логического уровня. Флаг переноса CARRY Flag устанавливается в том случае, если в результате операции сложения из самого старшего разряда происходит перенос или заем при выполнении вычитания. Анализ состояния флагов, записанных в соответствующие разряды регистра статуса, производится при выполнении условных команд. Кроме того, регистр статуса принимает участие в процессе выполнения операций сдвига.

Регистр фиксации флагов предназначен для сохранения текущих значений признаков результата операции АЛУ, записанных в регистр статуса, перед выполнением процедуры обслуживания прерывания.

Блок регистров общего назначения содержит шестнадцать восьмиразрядных регистров, обозначаемых по порядку s0... sF. Эти регистры предназначены для хранения данных, поступающих из входных портов ввода-вывода, операндов и результатов выполнения операций. Все регистры имеют одинаковый статус (полностью равноправны). Любой из них может использоваться в качестве аккумулятора.

Схема управления прерываниями формирует комбинацию управляющих сигналов, необходимых для выполнения процедуры обработки прерываний. Микропроцессорное ядро PicoBlaze изначально содержит единственную цепь сигнала прерывания. Для создания комплексной системы прерываний, поддерживающей комбинацию нескольких линий сигналов, следует использовать дополнительную логику, которая реализуется на основе свободных ресурсов кристалла.

Блок управления вводом-выводом предназначен для формирования адреса входного или выходного порта PORT_ID[7:0], к которому производится обращение, а также сигналов WRITE_STROBE и READ_STROBE, указывающих тип выполняемой операции (записи или чтения в указанный порт). Адрес порта ввода-вывода может задаваться в программе в виде абсолютного значения или в форме ссылки на один из регистров общего назначения, содержимое которого определяет номер порта. Во время выполнения операции ввода INPUT данные из входного порта, номер которого определяет комбинация значений сигналов в шине адреса порта PORT_ID[7:0], могут быть загружены в любой из шестнадцати регистров общего назначения. Эта процедура сопровождается формированием импульсного сигнала READ_STROBE. При осуществлении операции вывода OUTPUT информация из любого регистра общего назначения может быть передана в выходной порт, номер которого указывает комбинация значений сигналов в шине адреса порта PORT_ID[7:0]. Выполнение операции вывода сопровождается стробом записи WRITE_STROBE.

Память программ представляет собой запоминающее устройство, реализуемое в виде однопортового ОЗУ на основе блочной памяти кристалла Block SelectRAM, в котором хранится последовательность выполняемых команд.

Блок дешифрации команд на основании данных, поступающих с выходов программной памяти, формирует совокупность управляющих сигналов, необходимых для выполнения соответствующей операции, которые подаются на все блоки микропроцессорного ядра.

Управление последовательностью выполнения команд в составе программы осуществляется с помощью программного счетчика. Сигналы на его выходах образуют адрес выборки следующей команды. Эти сигналы по шине адресов команд передаются на адресные входы программной памяти. Режим работы программного счетчика (счет или загрузка) определяется состоянием сигналов, формируемых в блоке управления выбором следующего адреса команды. В основном режиме при отсутствии прерываний, команд переходов, вызовов и возврата из подпрограмм содержимое программного счетчика автоматически увеличивается на единицу при исполнении текущей операции. Таким образом, реализуется последовательная выборка и выполнение команд программы.

В процессе выполнения команд перехода в программный счетчик производится загрузка значения, соответствующего адресу, по которому осуществляется передача управления в программе. После исполнения команды перехода программный счетчик продолжает работу в инкрементном режиме, но, уже начиная с нового значения, которое было записано при ее выполнении.

При вызове подпрограмм также осуществляется принудительное изменение содержимого программного счетчика. В него загружается значение, которое соответствует начальному адресу выполняемой подпрограммы. Перед этим прежнее содержимое программного счетчика заносится в стек (в регистр, адрес которого определяется текущим значением указателя стека). В процессе выполнения команд подпрограммы производится последовательное инкрементирование нового содержимого программного счетчика. После завершения подпрограммы, при выполнении команды возврата в основную программу, из стека извлекается последнее записанное значение адреса, которое увеличивается на единицу и загружается в программный счетчик. Тем самым осуществляется переход к выполнению очередной команды, следующей после вызова подпрограммы. Так как глубина стека составляет пятнадцать уровней, то он может хранить одновременно до пятнадцати адресов возврата. Поэтому в процессе исполнения подпрограмм допускаются вложенные вызовы других подпрограмм. При очередном вложенном вызове подпрограммы содержимое программного счетчика заносится в следующий регистр стека. Процесс выполнения каждой вложенной подпрограммы завершается «выталкиванием» из стека последнего записанного значения, которое, увеличиваясь на единицу, заносится в программный счетчик.

Процедура обработки прерывания выполняется подобно вызову подпрограммы, но имеются несколько отличий. Одно из них заключается в том, что при наличии активного уровня сигнала на входе прерывания в программный счетчик загружается адрес вектора соответствующей процедуры обработки. При этом кроме записи в стек адреса текущей исполняемой команды производится сохранение состояния признаков результата операции АЛУ (содержимого регистра статуса) на момент прерывания в регистре фиксации флагов. После завершения процесса обработки прерывания происходит автоматическое восстановление значений флагов в регистре статуса.

В начальный момент времени функционирования микропроцессорного ядра, а также при подаче внешнего сигнала сброса, выходы программного счетчика устанавливаются в нулевое состояние (низкого логического уровня). Таким образом управление передается первой команде программы.

Команды, поддерживаемые микропроцессорным ядром PicoBlaze

Совокупность команд, поддерживаемых микропроцессорным ядром PicoBlaze, можно разбить по функциональному признаку на шесть групп. К первой группе относятся команды, управляющие последовательностью выполнения операций в программе, и команды обработки подпрограмм. В эту группу входят команды переходов JUMP, вызова подпрограмм CALL и возврата из подпрограмм RETURN. Каждая из перечисленных команд представлена как в безусловной форме, так и в условном формате (то есть может исполняться только при выполнении определенного условия).

Вторую группу образуют логические команды. Эти команды выполняют поразрядные операции «Логическое И» (поразрядное умножение) AND, «Логическое ИЛИ» (поразрядное сложение) OR, «Исключающее ИЛИ» XOR. В качестве операндов могут выступать содержимое любого регистра общего назначения и константа, указанная в тексте команды, а также содержимое двух регистров блока РОН. К этой же группе относится команда загрузки значения в регистр общего назначения

LOAD. С помощью этой команды в выбранный регистр может записываться константа, указанная в тексте команды, или содержимое другого регистра блока РОН.

В третью группу входят арифметические команды. Команды ADD и ADDCY предназначены для получения суммы двух операндов без учета и с учетом входного переноса соответственно. Операция вычитания также может выполняться без учета и с учетом заема с помощью команд SUB и SUBCY соответственно. Значение одного из операндов, участвующих в арифметических операциях, содержится в регистре общего назначения, номер которого указан в тексте команды. В качестве второго операнда может использоваться значение, заданное непосредственно в коде команды, или содержимое другого регистра блока РОН.

Четвертую группу составляют команды сдвига. Они позволяют реализовать операции логического (арифметического) или циклического сдвига данных, записанных в регистре общего назначения, номер которого указан в тексте команды. В процессе выполнения одной команды сдвиг производится на один разряд. Сдвиг может осуществляться как влево (в сторону младшего разряда), так и вправо (в сторону старшего разряда). Операции циклического сдвига могут выполняться с участием разряда переноса регистра статуса.

Пятая группа включает в себя команды ввода-вывода. Эти команды предназначены для организации обмена данными между регистрами, входящими в блок РОН, и портами ввода-вывода. Выполнение операций чтения данных из входного порта в регистр общего назначения (INPUT) и записи информации из регистра в выходной порт (OUTPUT) рассмотрено выше, в разделе описания архитектуры микропроцессорного ядра PicoBlaze.

Последняя группа объединяет команды, используемые для обслуживания прерываний. В нее входят команды разрешения ENABLE INTERRUPT и запрета прерываний DISABLE INTERRUPT.

Более подробно синтаксис и выполнение команд микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E, будут рассмотрены в следующей публикации цикла.

Программирование микропроцессорного ядра PicoBlaze

Исходный текст программы в кодах ассемблера записывается в файл с расширением psm, название которого должно содержать не более восьми символов. Из этого файла ассемблер формирует VHDL-описание начального содержимого блочного ОЗУ, исполняющего роль программной памяти. Результирующий файл с расширением vhd включается в состав разрабатываемого проекта и используется далее в процессе синтеза, моделирования, размещения и трассировки в кристалле.

Ассемблер выполнен в виде DOS-приложения с названием KCPSM.EXE. Этот файл должен располагаться в рабочем каталоге проекта вместе с шаблонами ROM_form.vhd и ROM_form.coe. Для активизации ассемблера используется командная строка, которая имеет формат kcpsm *<название_файла_программы>[.psm]*

Инициализация программной памяти микропроцессорного ядра с помощью набора данных, которые соответствуют машинным кодам разработанной программы, выполняется автоматически в процессе загрузки конфигурационной последовательности проекта в кристалл.

Для практического изучения функционирования и применения рассматриваемого микропроцессорного ядра PicoBlaze можно воспользоваться универсальным инструментальным модулем *SET-StarterKit*, подробное описание которого публиковалось ранее в нашем журнале [11].

Литература

1. Зотов В. WebPACK ISE — свободно распространяемый пакет проектирования цифровых устройств на базе ПЛИС фирмы Xilinx // Компоненты и технологии. 2001. № 6.
2. Зотов В. WebPACK ISE: Интегрированная среда разработки конфигурации и программирования ПЛИС фирмы Xilinx. Создание нового проекта // Компоненты и технологии. 2001. № 7.
3. Зотов В. Схемотехнический редактор пакета WebPACK ISE. Создание принципиальных схем и символов // Компоненты и технологии. 2001. № 8.
4. Зотов В. Синтез проектов, реализуемых на базе ПЛИС FPGA фирмы Xilinx, в САПР WebPACK ISE // Компоненты и технологии. 2002. № 3.
5. Зотов В. Реализация проектов на базе ПЛИС семейств FPGA фирмы Xilinx в САПР WebPACK ISE // Компоненты и технологии. 2002. № 4.
6. Зотов В. Конфигурирование ПЛИС семейств FPGA фирмы Xilinx в САПР WebPACK ISE // Компоненты и технологии. 2002. № 5.
7. Зотов В. ModelSim — система HDL-моделирования цифровых устройств // Компоненты и технологии. 2002. № 6.
8. Зотов В. Функциональное моделирование цифровых устройств, проектируемых на базе ПЛИС фирмы Xilinx в среде САПР WebPACK ISE // Компоненты и технологии. 2002. № 7.
9. Зотов В. Временное моделирование цифровых устройств, проектируемых на базе ПЛИС фирмы Xilinx в среде САПР WebPACK ISE // Компоненты и технологии. 2002. № 8.
10. Зотов В. Новая версия свободно распространяемого пакета проектирования WebPACK ISE фирмы Xilinx // Компоненты и технологии. 2003. № 1.
11. Зотов В. Инструментальный комплект *SET-StarterKit* для практического освоения методов проектирования цифровых устройств на основе ПЛИС семейств FPGA фирмы Xilinx // Компоненты и технологии. 2002. № 9.