# Особенности микропроцессорного ядра PicoBlaze, предназначенного для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II

**В предыдущих публикациях данного цикла [1–3] были рассмотрены элементы семейства 8-разрядных микропроцессорных ядер PicoBlaze, предназначенные для использования в проектах, которые выполняются на базе ПЛИС серий FPGA фирмы Xilinx. Несмотря на то, что кристаллы серий CPLD обладают значительно меньшими функциональными возможностями по сравнению с представителями семейств FPGA, некоторые из них могут применяться для реализации встраиваемых микропроцессорных систем. Для этих целей фирма Xilinx предлагает наиболее компактную версию микропроцессорного ядра PicoBlaze, которая предназначена для применения в проектах, реализуемых на основе ПЛИС семейства CoolRunner-II.**

**Валерий Зотов**

walerry@euro.ru

Встраиваемый микропроцессорный модуль для ПЛИС семейства CoolRunner-II разработан на основе микропроцессорного ядра PicoBlaze, предназначенного для реализации на базе кристаллов семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E. Поэтому в настоящей статье при рассмотрении характеристик, архитектуры и системы команд микропроцессорного ядра PicoBlaze для ПЛИС семейства CoolRunner-II представлены только его отличительные особенности по сравнению с базовым ядром, описание которого было опубликовано ранее [1–2].

Данное семейство кристаллов отличается от других серий ПЛИС CPLD, выпускаемых фирмой Xilinx, высоким быстродействием, низкой потребляемой мощностью и наличием микросхем с достаточно большим объемом логических ресурсов. Более подробно характеристики кристаллов семейства CoolRunner-II представлены в отдельной статье [4].

## Основные характеристики ядра PicoBlaze, реализуемого на основе ПЛИС семейства CoolRunner-II

Микропроцессорное ядро PicoBlaze, предназначенное для разработки систем на основе кристаллов семейства CoolRunner-II, обладает основными техническими характеристиками, схожими с характеристиками базового модуля, представленными в первой публикации цикла [1]. Наиболее существенные различия проявляются в вопросе применения встроенного ППЗУ микропрограмм, объеме блока регистров общего назначения и производительности. Рассматриваемый представитель семейства PicoBlaze отличается от микропроцессорного ядра для ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E следующими особенностями:

- возможность применения в проектах, выполняемых на основе кристаллов CoolRunner-II с числом макроячеек 256 и более (XC2C256, XC2C384, XC2C512);
- объем блока регистров общего назначения — восемь регистров по восемь разрядов;
- отсутствие встроенного ППЗУ микропрограмм;
- четырехуровневый стек;
- объем ресурсов кристалла, необходимых для реализации микропроцессорного ядра PicoBlaze в ПЛИС CoolRunner-II, составляет 212 макроячеек, что соответствует 83% от полного объема логических ресурсов кристалла XC2C256 (при этом используется 155 регистров и 53 пользовательских вывода ПЛИС, что составляет 61 и 45% от полного объема этих ресурсов).

## Структура проекта ядра PicoBlaze, реализуемого на основе CoolRunner-II

В кристаллах семейства CoolRunner-II, как и в ПЛИС CPLD других серий, отсутствуют ресурсы выделенной блочной памяти. Использование же основных логических ресурсов ПЛИС (триггеров, входящих в состав макроячеек) для формирования программной памяти встраиваемого процессорного модуля крайне неэффективно. Поэтому в версии микропроцессорного ядра PicoBlaze, предназначенной для реализации в кристаллах семейства CoolRunner-II, исключен модуль встроенного ППЗУ микропрограмм. Таким образом, структура рассматриваемого варианта встраиваемого микропроцессорного ядра PicoBlaze, в отличие от рассмотренных ранее представителей этого семейства, содержит только исполнительный модуль.

Комплект микропроцессорного ядра PicoBlaze, предназначенного для семейства CoolRunner-II, содержит три группы файлов, упакованных в архив. Каждая из этих групп расположена в отдельном каталоге (разделе) архива. Первая группа содержит файлы всех необходимых модулей описаний на языке VHDL. Файлы этой группы расположены в разделе VHDL. Вторая группа включает в себя исполняемый программный модуль ассемблера asm.exe и файл его исходного описания на языке C.

В эту же группу включен файл, содержащий тестовую программу на языке ассемблера, и файлы, полученные в результате обработки этой программы ассемблером. Данная группа файлов находится в разделе C. Третья группа файлов, расположенная в разделе DEMO_TEST, представляет собой тестовый проект, иллюстрирующий использование компонентов ядра. Демонстрационный проект и все компоненты ядра полностью совместимы с любой из конфигураций средств проектирования фирмы Xilinx серии ISE (Integrated Synthesis Environment), включая свободно распространяемую — WebPACK ISE [3].

Исполнительный модуль выполнен в форме компонента *picoblaze*, представляющего собой макрос с относительным размещением, описание которого на языке VHDL содержится в файле *picoblaze.vhd*. В этом описании используется ряд компонентов следующего (более низкого) уровня иерархии, которые в большинстве своем соответствуют элементам архитектуры микропроцессорного ядра PicoBlaze. Описания этих компонентов находятся в соответствующих файлах, которые расположены в том же разделе, что и файл описания компонента верхнего уровня иерархии *picoblaze*. Названия файлов, содержащих описания компонентов нижнего уровня иерархии, как правило, совпадают с именами этих компонентов. Выражения декларации компонента *picoblaze* в составе VHDL-описания проектируемой системы выглядят следующим образом.

```
component picoblaze
   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 та же система обозначений интерфейсных цепей компонентов микропроцессорного ядра PicoBlaze, что и для компонентов базового варианта, подробно описанная ранее [1]. Включение экземпляра компонента, представляющего исполнительный модуль *picoblaze*, в состав структурного описания архитектуры проектируемой системы осуществляется с помощью следующей конструкции.

```
inst_name_processor: picoblaze
   port map(
                address => address_name,
                instruction => instruction_name,
                port_id => port_id_name,
                write_strobe => write_strobe_name,
                out_port => out_port_name,
                read_strobe => read_strobe_name,
                in_port => in_port_name,
                interrupt => interrupt_event,
                reset => reset_name,
                clk => clk_name
              );
```

В приведенном шаблоне следует вместо идентификатора *inst_name_processor* задать метку, которая, как правило, соответствует позиционному обозначению создаваемого экземпляра компонента. Кроме того, в операторе port map нужно указать названия сигналов, которые используются в описании проектируемого устройства.

Для отладки разрабатываемой программы методом моделирования ядра в составе создаваемого проекта с помощью системы ModelSim XE [5–8] можно использовать компонент виртуального ПЗУ микропрограмм. Этот компонент представляет ПЗУ информационной емкостью 4 кбит с организацией 256×16 разрядов. В качестве шаблона для декларации и создания экземпляра компонента виртуального ПЗУ микропрограмм можно использовать соответствующие выражения, приведенные для модуля программной памяти ядра PicoBlaze, реализуемого на базе ПЛИС семейств Spartan-II, Spartan-IIE, Virtex, Virtex-E [1]. Название компонента виртуального ПЗУ микропрограмм должно совпадать с именем файла, в котором содержится текст отлаживаемой программы на языке ассемблера.

Тестовый проект, включенный в состав комплекта микропроцессорного ядра PicoBlaze, можно использовать в качестве образца VHDL-описания отладочной системы. Основу этой системы образует VHDL-описание объекта DEMO, в структуру которого входит исполнительный модуль и подключенный к нему модуль виртуальной программной памяти. Ниже приведен полный текст описания тестовой системы, который также демонстрирует методику применения компонента *picoblaze* в составе проектируемого устройства.

```
----------------------------------------------------------------------
-- Standard IEEE libraries
--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------
entity demo is
   Port (output : out std_logic_vector(7 downto 0);
         reset : in std_logic;
         clk : in std_logic);
   end demo;
----------------------------------------------------------------------
-- Start of test achitecture
--
architecture Behavioral of demo is
----------------------------------------------------------------------
 component picoblaze
   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 demo_test
   Port (
         address : in std_logic_vector(7 downto 0);
         dout : out std_logic_vector(15 downto 0);
         clk : in std_logic
         );
   end component;
----------------------------------------------------------------------
-- Signals used to connect picoblaze to program ROM and I/O logic
--
signal address : std_logic_vector(7 downto 0);
signal instruction : std_logic_vector(15 downto 0);
signal port_id : std_logic_vector(7 downto 0);
signal out_port : std_logic_vector(7 downto 0);
signal in_port : std_logic_vector(7 downto 0);
signal write_strobe : std_logic;
signal read_strobe : std_logic;
signal interrupt_event : std_logic;
--signal reset : std_logic;
--
----------------------------------------------------------------------
-- Start of circuit description
--
begin
-- Inserting picoblaze and the program memory
--
   processor: picoblaze
         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_event,
                  reset => reset,
                  clk => clk
                  );

   program: demo_test
         port map(
                  address => address,
                  dout => instruction,
                  clk => clk);
   --
   -- Unused inputs on processor
   --
   in_port <= «00000000»;
   interrupt_event <= '0';
   -- reset <= '0';
   --
   -- adding the output registers to the processor

   IO_registers: process(clk)
   begin

   -- waveform register at address 01

   if clk'event and clk='1' then
       if port_id(0)='1' and write_strobe='1' then
           output <= out_port;
       end if;
   end if;
 end process IO_registers;
end Behavioral;
----------------------------------------------------------------------
-- END OF FILE demo.VHD
--
```

В структуре представленного описания можно выделить три раздела. В первом разделе описания указаны ссылки на используемые стандартные библиотеки и пакеты. Следующий раздел содержит операторы, описывающие интерфейс объекта DEMO. В третьем разделе приводится структурное описание архитектуры этого объекта. Этот раздел состоит из пяти секций. Первая секция содержит выражения декларации используемых компонентов. Во второй секции представлены выражения декларации внутренних сигналов устройства. В третьей секции описывается собственно структура объекта DEMO. Четвертая секция содержит выражения, которые определяют значения сигналов на неиспользуемых входах микропроцессорного ядра. Пятая секция описывает процесс формирования выходных сигналов объекта DEMO.

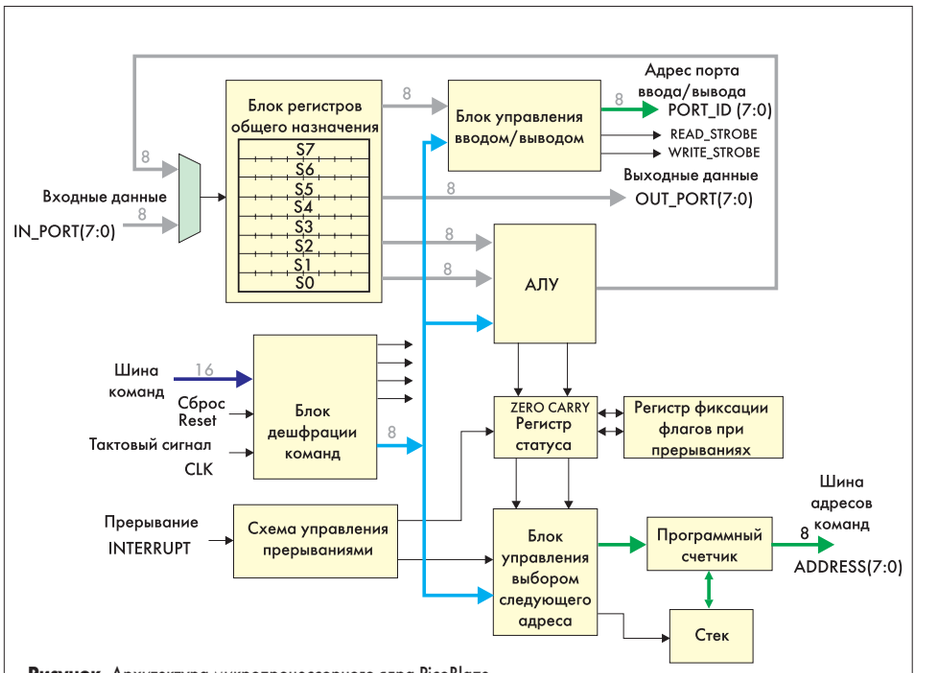


**Рисунок.** Архитектура микропроцессорного ядра PicoBlaze, реализуемого на основе ПЛИС семейства CoolRunner-II

## Архитектура ядра PicoBlaze, реализуемого на основе CoolRunner-II

Архитектура микропроцессорного ядра PicoBlaze, предназначенного для применения в кристаллах семейства CoolRunner-II, изображена на рисунке. В структурном отношении она отличается от архитектуры базового микропроцессорного модуля [1] только отсутствием встроенного блока программной памяти. Кроме того, имеются различия, которые проявляются уже на уровне отдельных структурных элементов. Эти отличия обусловлены, прежде всего, ограниченным объемом ресурсов ПЛИС семейства CoolRunner-II по сравнению с кристаллами серий Spartan-II, Spartan-IIE, Virtex, Virtex-E. Поэтому одной из основных задач при разработке рассматриваемого варианта микропроцессорного ядра PicoBlaze являлась минимизация ресурсов кристалла, необходимых для его реализации. Решение этой задачи достигнуто за счет сокращения функциональных возможностей отдельных структурных элементов.

Вдвое уменьшен объем блока регистров общего назначения, который в новой версии содержит восемь восьмиразрядных регистров, обозначаемых соответствующими порядковыми номерами s0–s7.

Почти в четыре раза (с пятнадцати уровней до четырех) сокращена глубина стека программного счетчика. Тем самым накладывается более жесткое ограничение на количество вложенных вызовов процедур в разрабатываемой программе.

Модернизация блока дешифрации команд, обусловленная необходимостью снижения объема используемых ресурсов ПЛИС, привела, в частности, к изменению формата команд, поддерживаемых микропроцессорным ядром PicoBlaze, которое предназначено для применения в кристаллах семейства CoolRunner-II.

## Система команд ядра PicoBlaze, реализуемого на основе CoolRunner-II

Базовая система команд микропроцессорного ядра PicoBlaze, встраиваемого в проекты, выполняемые на основе ПЛИС семейства CoolRunner-II, включает в себя 49 инструкций [2]. При классификации команд по функциональному признаку они подразделяются на шесть уже известных групп. Изменения произошли только в формате команд. В ряде инструкций поменялась длина полей и коды выполняемых операций. При этом полная длина команд не изменилась и по-прежнему составляет шестнадцать двоичных разрядов. В некоторых командах изменилось взаимное расположение полей. Мнемоническая форма записи инструкций сохранилась без изменений.

При необходимости разработчик может расширить существующую систему команд, дополнив ее собственными инструкциями. Для этого нужно внести соответствующие изменения в файлы исходного описания микропроцессорного ядра *picoblaze.vhd* и ассемблера *asm.cpp*.

В последующих разделах будут представлены соответствующие форматы инструкций для каждой функциональной группы, которые входят в базовую систему команд рассматриваемого представителя семейства микропроцессорных ядер PicoBlaze.

**Таблица 1.** Форматы команд переходов ядра PicoBlaze, реализуемого на основе CoolRunner-II

| 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 | JUMP aa | Безусловный переход |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | A | A | A | A | A | A | A | A | JUMP Z,aa | 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 | JUMP NZ,aa | 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 | JUMP C,aa | 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 | JUMP NC,aa | Переход при условии, что флаг CARRY Flag находится в сброшенном состоянии |
| 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 ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

## Команды управления последовательностью выполнения операций в программе для ядра PicoBlaze, реализуемого на базе CoolRunner-II

В командах безусловного и условных переходов *JUMP* изменились значения и структура поля кода операции. Форматы команд безусловного и условных переходов *JUMP* в новой редакции представлены в таблице 1.

Модификация команд обращения к подпрограммам *CALL* также затронула структуру поля кода операции. Поле адреса вызываемой подпрограммы осталось без изменений. Новые версии форматов команд безусловного и условных вызовов подпрограмм *CALL* приведены в таблице 2.

Преобразование инструкций возврата из подпрограммы *RETURN* проявилось в изменении структуры полей и значений кода выполняемой операции. В таблице 3 представлены модифицированные форматы команд безусловного и условного возврата из подпрограммы *RETURN*.

## Группа логических команд ядра PicoBlaze, предназначенного для кристаллов CoolRunner-II

В формате инструкций, относящихся к группе логических команд, произошли следующие изменения по сравнению с аналогичными инструкциями, представленными ранее [2]. Во-первых, все команды этой группы имеют одинаковую длину поля кода операции. В новой редакции это поле включает в себя пять двоичных разрядов. Во-вторых, длина полей команд, в которых указываются номера регистров, используемых при выполнении операции, уменьшилась на один бит и составляет три двоичных разряда. Изменение длины полей, предназначенных для определения номеров регистров, обусловлено двукратным сокращением объема блока регистров общего назначения. В качестве номеров регистров *N* и *M*, которые указываются при мнемонической форме записи инструкций, могут использоваться любые числа в диапазоне от 0 до 7.

Новая редакция форматов команд поразрядных операций «Логическое И» (поразрядное умножение) AND, выполняемых над содержимым одного из регистров общего назначения и константой *kk*, значение которой задается непосредственно в тексте инструкции, а также над содержимым двух регистров общего назначения, приведена в таблице 4.

Модифицированные форматы инструкций OR, предназначенных для выполнения операций поразрядного сложения двух операндов (поразрядное «Логическое ИЛИ»), определены в таблице 5 для двух вариантов. В первом случае операндами является содержимое любого из восьми регистров общего назначения и константа *kk*, значение которой указывается в соответствующем поле команды, а во втором — содержимое двух регистров 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.

Форматы инструкций *LOAD*, предназначенных для загрузки константы или содержимого какого-либо регистра в выбранный регистр общего назначения, в новой редакции приведены в таблице 7.

## Группа арифметических команд ядра PicoBlaze, предназначенного для CoolRunner-II

В структуре полей арифметических команд ядра PicoBlaze, предназначенного для применения в кристаллах CoolRunner-II, произошли те же изменения (по сравнению с форматами аналогичных команд, приведенными ранее [2]), что и в логических инструкциях, рассмотренных в предыдущем разделе.

Модифицированные варианты форматов команд сложения *ADD* содержимого регистра с номером *N* и константы *kk* или содержимого двух регистров общего назначения с номерами *N* и *M* без учета переноса представлены в таблице 8.

Новая версия форматов инструкций *ADDCY*, предназначенных для вычисления суммы двух операндов с учетом значения флага переноса, полученного при выполнении предыдущей операции, приведена в таблице 9.

**Таблица 3.** Форматы команд безусловного и условного возврата из подпрограммы для ядра PicoBlaze, реализуемого на основе CoolRunner-II

| Поле кода операции | | | | | | | | Поле адреса | | | | | | | | Мнемоника | Выполняемая операция |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | X | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RETURN | Безусловный возврат из подпрограммы |
| 1 | 0 | 0 | 1 | 0 | 1 | 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 | 0 | 1 | 0 | 1 | 0 | 1 | 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 | 0 | 1 | 0 | 1 | 1 | 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 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RETURN NC | Возврат из подпрограммы при условии, что флаг CARRY Flag находится в сброшенном состоянии |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Номер разряда микрокоманды | |

**Таблица 4.** Форматы команд поразрядных операций «Логическое И» ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

**Таблица 5.** Форматы команд поразрядных операций «Логическое ИЛИ» ядра PicoBlaze, реализуемого на основе CoolRunner-II

| Поле кода операции | | | | | Поле номера регистра | | | Поле константы | | | | | | | | Мнемоника | Выполняемая операция |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 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 | 1 | 0 | 1 | 0 | n | n | n | m | m | m | 0 | 0 | 0 | 0 | 0 | OR sN,sM | Поразрядное «Логическое ИЛИ» содержимого регистров sN и sM |
| 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» ядра PicoBlaze, реализуемого на основе CoolRunner-II

| Поле кода операции | | | | | Поле номера регистра | | | Поле константы | | | | | | | | Мнемоника | Выполняемая операция |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 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 | 1 | 0 | 1 | 1 | n | n | n | m | m | m | 0 | 0 | 0 | 0 | 0 | XOR sN,sM | Поразрядное «Исключающее ИЛИ» содержимого регистров sN и sM |
| 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 ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

**Таблица 8.** Форматы команд сложения двух операндов без учета переноса для ядра PicoBlaze, реализуемого на основе CoolRunner-II

| Поле кода операции | | | | | Поле номера регистра | | | Поле константы | | | | | | | | Мнемоника | Выполняемая операция |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 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 | n | n | n | m | m | m | 0 | 0 | 0 | 0 | 0 | ADD sN,sM | Сложение содержимого регистров sN и sM |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Номер разряда микрокоманды | |

**Таблица 9.** Форматы команд сложения двух операндов с учетом переноса для ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

**Таблица 10.** Форматы команд вычитания без учета заема для ядра PicoBlaze, реализуемого на основе CoolRunner-II

| Поле кода операции | | | | | Поле номера регистра | | | Поле константы | | | | | | | | Мнемоника | Выполняемая операция |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 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 | n | n | n | m | m | m | 0 | 0 | 0 | 0 | 0 | SUB sN,sM | Вычитание содержимого регистра sM из содержимого регистра sN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Номер разряда микрокоманды | |

**Таблица 11.** Форматы инструкций вычитания с учетом заема для ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

**Таблица 12.** Форматы команд логического или циклического сдвига данных ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

Форматы инструкций *SUB*, используемых для выполнения операции вычитания с участием тех же операндов, что и в командах сложения, без учета заема, в новой редакции представлены в таблице 10.

Модифицированные варианты форматов команд *SUBCY*, предназначенных для вычисления разности двух операндов с учетом значения заема, образовавшегося при выполнении предыдущей операции, приведены в таблице 11.

## Команды сдвига данных для ядра PicoBlaze, предназначенного для кристаллов CoolRunner-II

В формате команд, предназначенных для выполнения операций сдвига данных, изменилась длина полей кода операции и номера регистра. Преобразованные форматы инструкций логического (арифметического) и циклического сдвига данных, находящихся в регистре общего назначения с указанным номером, представлены в таблице 12.

## Команды ввода-вывода ядра PicoBlaze для CoolRunner-II

Структура инструкций ввода-вывода, используемых для организации чтения данных из входного порта в заданный регистр общего назначения и передачи информации из указанного регистра в выходной порт, отличается от структуры аналогичных команд, приведенной ранее [2], длиной полей кода операции и номеров регистров. В новой редакции команд ввода-вывода указанные поля содержат соответственно пять и три двоичных разряда.

Новые варианты форматов команд ввода-вывода с различными видами адресации входных и выходных портов приведены в таблице 13.

## Команды обслуживания прерываний ядра PicoBlaze для CoolRunner-II

В формате инструкций обслуживания прерываний изменилось взаимное расположение и длина поля кода операции и поля режима обработки прерываний.

Новая редакция форматов команд возврата из процедуры обслуживания прерываний *RETURNI* и установки режима обработки прерываний в программе *ENABLE INTERRUPT* и *DISABLE INTERRUPT* представлена в таблице 14.

Для практического освоения рассмотренного варианта микропроцессорного ядра PicoBlaze и для аппаратной отладки проектов, включающих это ядро, можно воспользоваться инструментальным комплектом *CoolRunner-II Design Kit*, возможности которого были рассмотрены на страницах «КиТ» [9].

На этом завершается рассмотрение характеристик, архитектуры и системы команд встраиваемых восьмиразрядных микропроцессорных модулей семейства PicoBlaze. В следующей публикации цикла будут обсуждаться вопросы использования ассемблера для данного семейства микропроцессорных ядер.

## 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. CoolRunner-II — новое поколение высокопроизводительных ПЛИС CPLD фирмы Xilinx с микромощным потреблением // Схемотехника. 2003. № 5–10.
5. Зотов В. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. М.: Горячая линия — Телеком. 2003.
6. Зотов В. ModelSim — система HDL-моделирования цифровых устройств // Компоненты и технологии. 2002. № 6.
7. Зотов В. Функциональное моделирование цифровых устройств, проектируемых на базе ПЛИС фирмы Xilinx в среде САПР WebPACK ISE // Компоненты и технологии. 2002. № 7.
8. Зотов В. Временное моделирование цифровых устройств, проектируемых на базе ПЛИС фирмы Xilinx в среде САПР WebPACK ISE // Компоненты и технологии. 2002. № 8.
9. Зотов В. Инструментальный комплект *CoolRunner-II Design Kit* для практического освоения методов программирования ПЛИС семейств CPLD фирмы Xilinx // Компоненты и технологии. 2003. № 2.

**Таблица 13.** Форматы команд ввода-вывода ядра PicoBlaze, реализуемого на основе CoolRunner-II

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

**Таблица 14.** Форматы команд обслуживания прерываний ядра PicoBlaze, реализуемого на основе CoolRunner-II

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