Организация памяти микропроцессорного ядра MicroBlaze

В предшествующих публикациях цикла были представлены основные характеристики, особенности архитектуры [1], система команд [2] семейства 32-разрядных микропроцессорных ядер MicroBlaze, предназначенных для реализации встраиваемых систем на основе ПЛИС серий FPGA фирмы Xilinx, и комплекс средств автоматизированного проектирования Xilinx Embedded Development Kit (EDK) [3]. Прежде чем приступить непосредственно к рассмотрению процесса проектирования микропроцессорных систем на основе ядра MicroBlaze в рамках пакета Xilinx EDK, необходимо остановиться на вопросах, связанных с распределением адресного пространства памяти и организацией хранения данных.

Валерий Зотов

walerry@euro.ru

Ограничения адресного пространства, производительности и объема памяти для ядра MicroBlaze

В архитектуре микропроцессорного ядра MicroBlaze используется 32-разрядная шина адреса. Таким образом, адресное пространство памяти ограничивается диапазоном от 0x00000000 до 0xFFFFFFFF. Доступ к памяти может осуществляться через шину Local Memory Bus (LMB) или через шину On-chip Peripheral Bus (OPB) [1]. Локальная шина LMB предназначена, в первую очередь, для обеспечения высокоскоростного доступа к памяти, расположенной непосредственно на кристалле. Эта память реализуется на основе элементов блочного ОЗУ Block RAM (BRAM) ПЛИС семейств FPGA фирмы Xilinx. Глобальная шина OPB может использоваться для сопряжения ядра MicroBlaze и памяти, расположенной как внутри, так и вне кристалла. Кроме того, эта шина предназначена для организации взаимодействия микропроцессорного ядра и периферийных устройств. Следует обратить внимание на то, что подключение памяти, реализуемой на основе внутренних ресурсов ПЛИС, через шину OPB приводит к снижению производительности системы по сравнению с вариантом, когда интерфейс между микропроцессорным ядром и внутренней памятью строится на основе локальной шины LMB. Выполнение операций записи и чтения данных из блочной памяти ПЛИС при использовании интерфейса шины LMB занимает два машинных цикла. Если сопряжение микропроцессорного ядра с памятью, расположенной непосредственно на кристалле, осуществляется по шине OPB, то для выполнения операции записи данных в память требуется три машинных цикла, а для операции чтения — четыре. Время доступа к внешней памяти (по отношению к ПЛИС) во многом зависит от параметров микросхем, применяемых для ее реализации. В настоящее время в большинстве случаев время доступа к памяти, расположенной вне кристалла, составляет от пяти до семи машинных циклов.

Суммарный объем памяти, реализуемой непосредственно на кристалле, ограничивается соответствующими физическими ресурсами используемой ПЛИС. Максимальное значение этого параметра зависит от выбранного семейства и типа кристалла FPGA [4–7].

Распределение адресного пространства памяти ядра MicroBlaze

Типовая структура распределения адресного пространства памяти MicroBlaze показана на рис. 1. Начальной (стартовой) позицией адресного пространства памяти является нулевое значение (0x00000000). Конечная граница адресного пространства зависит от конфигурации проектируемой системы, но не может превышать максимального значения (0xFFFFFFFF).

В начальной секции адресного пространства располагается диапазон адресов, зарезервированный для памяти, которая расположена непосредственно на кристалле и сопряжена с микропроцессорным ядром по локальной шине LMB. Следующий диапазон адресов предназначен для внутренней памяти ПЛИС, доступ к которой осуществляется по шине OPB. Далее, в направлении увеличения значения адреса, следует диапазон адресов внешней памяти, подключенной к шине OPB. В конечной секции адресного пространства памяти обычно находится диапазон адресов периферийных модулей проектируемой микропроцессорной системы. Между перечисленными диапазонами адресов в общем случае могут присутствовать неиспользуемые интервалы адресного пространства.

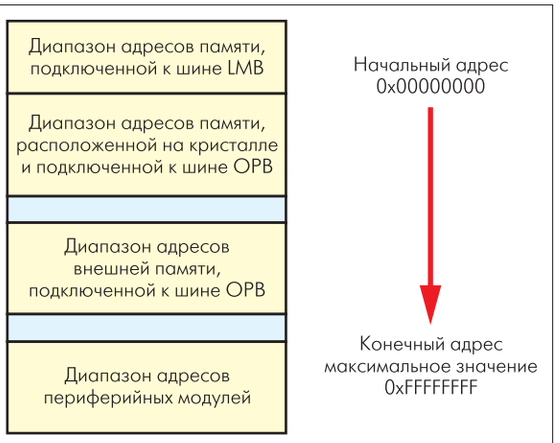

Рис. 1. Карта типового распределения адресного пространства памяти

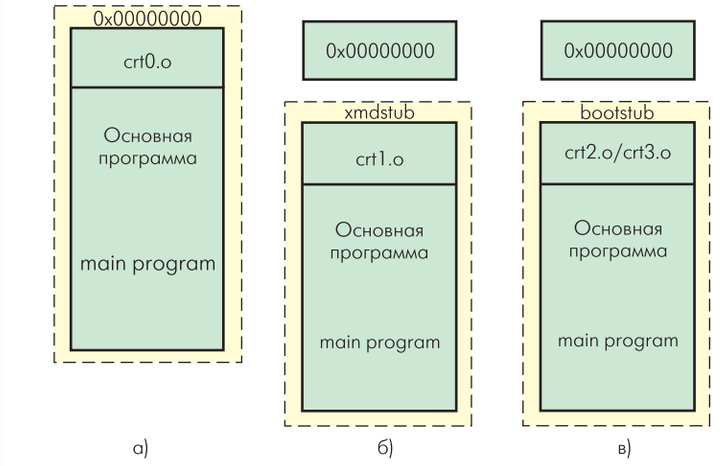

Рис. 2. Варианты размещения исполняемого кода программ в адресном пространстве памяти

Карта фактического распределения адресного пространства памяти определяется в спецификации аппаратной части проектируемой микропроцессорной системы MHS (Microprocessor Hardware Specification). В файле MHS указывается диапазон адресов для каждого используемого компонента разрабатываемой системы, в том числе для блочной памяти ПЛИС BRAM, подключаемой к шинам LMB и OPB, внешней памяти и периферийных модулей. Информация о необходимом диапазоне адресов для IP-ядер, входящих в комплект пакета Xilinx EDK, содержится в соответствующей документации. В спецификации аппаратной части разрабатываемой микропроцессорной системы MHS задается базовое и максимальное значение адреса для каждого применяемого IP-компонента.

Часть адресного пространства памяти микропроцессорного ядра MicroBlaze в диапазоне от 0x00000000 по 0x00000018 зарезервирована для выполнения специальных функций. В частности, в ячейках памяти этого диапазона содержится информация об адресах переходов, выполняемых при сбросе микропроцессорного ядра, обработке прерываний и исключений. При включении питания или поступлении сигнала сброса в ядре MicroBlaze управление автоматически передается по адресу 0x00000000. При появлении запроса прерывания осуществляется переход по адресу 0x00000010. В случае возникновения исключений автоматически производится передача управления по адресу 0x00000008.

Размещение исполняемого кода программ в памяти ядра MicroBlaze

Адресное пространство памяти можно условно разбить на две части: системную область и область, используемую прикладными программами. Под системную область, как правило, отводится диапазон адресного пространства, начинающийся с нулевого значения адреса (0x00000000). Использование этой части памяти определяется выбранным режимом исполнения прикладной программы. Микропроцессорное ядро MicroBlaze поддерживает несколько режимов (сценариев) выполнения прикладных программ. Каждому из этих сценариев соответствует определенная схема размещения программ в адресном пространстве памяти.

Первый вариант сценария используется для выполнения прикладных программ, написанных на языках высокого уровня C/C++, которые не требуют поддержки автозагрузки и отладки. В этом случае в процессе генерации исполняемого кода компоновщик присоединяет к объектному файлу основной программы системный файл crt0.o. Исполняемый код, соответствующий данному системному файлу, загружается в системную область памяти, начиная с нулевого адреса. Исполняемый код прикладной программы располагается непосредственно вслед за кодом файла crt0.o. В более наглядной форме данный способ размещения прикладной программы в адресном пространстве памяти микропроцессорного ядра MicroBlaze поясняет рис. 2*а*.

Второй вариант сценария выполнения прикладных программ применяется в случае, когда необходимо обеспечить поддержку отладочных средств Xilinx Microprocessor Debugger (XMD). Для этого в системную область памяти микропроцессорного ядра MicroBlaze должен быть загружен соответствующий модуль *xmdstub*. Данный модуль располагается в системной области памяти, начиная с нулевого адреса (0x00000000). При этом компоновщик в процессе формирования исполняемого кода присоединяет к объектному файлу основной программы системный файл crt1.o. Сгенерированный исполняемый код прикладной программы, включающий в себя файл crt1.o, размещается в области памяти, которая начинается с фиксированного значения адреса. Для рассматриваемого варианта сценария в качестве значения этого адреса, установленного по умолчанию, используется 0x00000400. Рис. 2*б* иллюстрирует размещение исполняемого кода программных модулей в памяти, соответствующее второму варианту сценария.

Третий вариант сценария предназначен для реализации режима автозагрузки прикладной программы. Для осуществления этого сценария в системную область памяти ядра MicroBlaze необходимо загрузить соответствующий специальный модуль *bootstub*. Данный модуль размещается в системной области памяти с нулевого адреса (0x00000000). Кроме того, в процессе компоновки исполняемого кода к объектному файлу основной программы должен быть присоединен системный файл crt2.o или crt3.o. Какой из этих системных файлов будет скомпонован с основной программой, зависит от выбранной опции компилятора. Сформированный исполняемый код прикладной программы, включающий в себя файл crt2.o или crt3.o, располагается в области памяти, начинающейся с фиксированного адреса. По умолчанию для данного варианта сценария установлено значение этого адреса 0x00000100. Схема размещения исполняемого кода программных модулей в памяти микропроцессорного ядра MicroBlaze, соответствующая третьему варианту сценария, представлена на рис. 2*в*.

Для того чтобы сформировать объектный код разрабатываемой прикладной программы, соответствующий одному из рассмотренных выше сценариев выполнения, следует воспользоваться специальными параметрами компилятора. Параметр -xl-mode-executable используется для компиляции прикладных программ, не требующих при выполнении поддержки средств отладки и автозагрузки. Параметр -xl-mode-xmdstub применяется в процессе компиляции программ, которые должны выполняться в режиме отладки. Параметры компилятора -xl-mode-bootstrap -xl-mode-bootstrap-reset предназначены для генерации объектного кода прикладных программ, которые должны поддерживать режим автозагрузки.

Организация хранения данных в памяти и регистрах общего назначения ядра MicroBlaze

Как уже упоминалось при рассмотрении команд передачи данных, микропроцессорное ядро MicroBlaze предусматривает три формата записи и извлечения данных из памяти: в виде 32-разрядных (полных) слов, 16-разрядных слов (полуслов) и 8-разрядных слов (побайтно). В процессе разработки программ на языке ассемблера или языках высо-

Таблица 1. Организация 32-разрядных слов данных в адресном пространстве памяти и регистрах ядра MicroBlaze

Адрес байта	n	n+1	n+2	n+3
Номер байта в слове	0	1	2	3
Порядок следования байт в слове	Старший байт most significant byte (MSByte)			Младший байт least significant byte (LSByte)
Номер бита в слове	0 … 31			
Порядок следования бит в слове	старший бит … младший бит			
	most significant bit (MSB) … least significant bit (LSB)			

Таблица 2. Организация 16-разрядных слов данных в адресном пространстве памяти и регистрах ядра MicroBlaze

Адрес байта	n	n+1
Номер байта в слове	0	1
Порядок следования байт в слове	Старший байт most significant byte (MSByte)	Младший байт least significant byte (LSByte)
Номер бита в слове	0 … 15	
Порядок следования бит в слове	старший бит … младший бит	
	most significant bit (MSB) … least significant bit (LSB)	

Таблица 3. Организация 8-разрядных слов данных в адресном пространстве памяти и регистрах ядра MicroBlaze

Адрес байта	n
Номер байта в слове	0
Номер бита в слове	0 … 7
Порядок следования бит в слове	старший бит … младший бит
	most significant bit (MSB) … least significant bit (LSB)

кого уровня C/C++, а также при создании описаний модулей микропроцессорной системы с использованием языков высокого уровня VHDL и Verilog необходимо располагать информацией о способе нумерации битов данных, передаваемых по шинам и записываемых в регистры общего назначения, и порядке размещения байтов, составляющих слово, в адресном пространстве памяти.

В архитектуре MicroBlaze используется следующее соглашение о нумерации разрядов шин и регистров. Старший разряд (старший значащий бит) MSB (most significant bit) имеет нулевой порядковый номер. Соответственно младший разряд (младший значащий бит) LSB (least significant bit) при 32-разрядной архитектуре шин и регистров обозначается порядковым номером 31. При этом следует учитывать, что ряд интерфейсных ядер, предлагаемых фирмой Xilinx, использует противоположный способ нумерации битов данных, при котором нулевой порядковый номер соответствует младшему разряду. В качестве примера ядер, использующих альтернативный способ нумерации битов данных, можно привести модуль интерфейса шины PCI — *PCI Core*. Данный модуль может применяться в виде компонента, который включается в состав встраиваемых микропроцессорных систем, проектируемых на основе ядра MicroBlaze.

В микропроцессорном ядре MicroBlaze байты, составляющие слово данных, располагаются в адресном пространстве памяти таким образом, при котором старшему значащему байту слова соответствует меньшее (начальное) значение адреса, а младшему байту — большее. При этом нумерация байт в слове начинается со старшего значащего байта. Иными словами, байты слова данных размещаются в адресном пространстве памяти в порядке убывания значимости. Данный способ организации хранения данных в памяти и регистрах в англоязычной литературе обозначается термином Big-endian.

Таблицы 1–3 в наглядной форме поясняют порядок нумерации битов и байтов в структуре слов данных различной длины и способ их организации в адресном пространстве памяти ядра MicroBlaze.

Некоторые компоненты встраиваемых микропроцессорных систем могут использовать обратный порядок размещения байтов слова данных в адресном пространстве памяти, при котором младший (начальный) адрес соответствует младшему значащему байту. Такой способ организации данных в адресном пространстве памяти обозначается термином Little-endian. Информация о типе организации данных для применяемого компонента содержится в документации, описывающей этот компонент. Для того чтобы применять компоненты, использующие организацию данных Little-endian, в системах, основанных на микропроцессорном ядре MicroBlaze, в состав VHDL- или Verilog-описаний необходимо включить секцию кода, в которой выполняется преобразование структуры слов данных.

В качестве примера ниже приводится VHDL-описание модуля блочной памяти BRAM, доступ к которой осуществляется через шину OPB. Модуль блочной памяти строится на основе базовых элементов двухпортового ОЗУ, информационная емкость которого составляет 16384 бит (16 кбит). Каждый порт блочной памяти имеет организацию 2048 слов × 8 разрядов с контролем по четности. Таким образом, для создания 32-разрядного модуля блочной памяти необходимо число базовых элементов, кратное 4. Базовый элемент блочного ОЗУ представлен в виде компонента RAMB16_S9_S9.

Шина OPB предполагает использование организации данных Big-endian, а в компонентах базовых элементов блочной памяти применяется Little-endian. Декларация объекта OPB_BRAM, который описывает модуль блочной памяти, подключенной к шине OPB, выглядит следующим образом:

```
--Interface Between BRAM and MicroBlaze

library IEEE;
use IEEE.std_logic_1164.all;

entity OPB_BRAM is
generic (
        C_BASEADDR : std_logic_vector(0 to 31) := X»B000_0000»;
        C_NO_BRAMS : natural := 4; -- Can be 4,8,16,32 only
        C_VIRTEXII : boolean := true
        );
port (
-- Global signals
OPB_Clk : in std_logic;
OPB_Rst : in std_logic;

        -- OPB signals
OPB_ABus : in std_logic_vector(0 to 31);
OPB_BE : in std_logic_vector(0 to 3);
OPB_RNW : in std_logic;
OPB_select : in std_logic;
OPB_seqAddr : in std_logic;
OPB_DBus : in std_logic_vector(0 to 31);
OPB_BRAM_DBus : out std_logic_vector(0 to 31);
OPB_BRAM_errAck : out std_logic;
OPB_BRAM_retry : out std_logic;
OPB_BRAM_toutSup : out std_logic;
OPB_BRAM_xferAck : out std_logic;

        -- OPB_BRAM signals (other port)
BRAM_Clk : in std_logic;
BRAM_Addr : in std_logic_vector(0 to 31);
BRAM_WE : in std_logic_vector(0 to 3);
BRAM_Write_Data : in std_logic_vector(0 to 31);
BRAM_Read_Data : out std_logic_vector(0 to 31)
);
end entity OPB_BRAM;
```

Кроме собственно выражений декларации, описывающих интерфейс и параметры объекта OPB_BRAM, в приведенном фрагменте представлены также ссылки на используемую библиотеку и пакет. Объем создаваемого модуля блочной памяти (в виде числа базовых элементов) определяется с помощью параметра C_NO_BRAMS. Параметр C_BASEADDR предназначен для установки значения базового адреса.

Архитектурное тело объекта OPB_BRAM, представляющего модуль блочной памяти, который предназначен для сопряжения с микропроцессорным ядром через шину OPB, содержит структурное описание этого объекта на языке VHDL. В состав архитектурного тела описания объекта OPB_BRAM входят четыре раздела. В первом разделе приводится декларация компонента RAMB16_S9_S9, представляющего 2-портовое ОЗУ, используемое в качестве базового элемента для построения модуля блочной памяти. Этот раздел содержит следующий VHDL-код:

```
architecture IMP of OPB_BRAM is

-- BRAM Component Declaration (little-endian)

component RAMB16_S9_S9
port (
        DIA : in std_logic_vector (7 downto 0);
        DIB : in std_logic_vector (7 downto 0);
        DIPA : in std_logic_vector (0 downto 0);
        DIPB : in std_logic_vector (0 downto 0);
        ENA : in std_ulogic;
        ENB : in std_ulogic;
        WEA : in std_ulogic;
        WEB : in std_ulogic;
        SSRA : in std_ulogic;
        SSRB : in std_ulogic;
```

```
        CLKA : in std_ulogic;
        CLKB : in std_ulogic;
        ADDRA : in std_logic_vector (10 downto 0);
        ADDRB : in std_logic_vector (10 downto 0);
        DOA : out std_logic_vector (7 downto 0);
        DOB : out std_logic_vector (7 downto 0);
        DOPA : out std_logic_vector (0 downto 0);
        DOPB : out std_logic_vector (0 downto 0)
);
end component;
```

Второй раздел включает в себя стандартные конструкции, используемые для декларации внутренних сигналов и переменных. Текст данного раздела не представлен в настоящей статье. Вся необходимая информация о типе используемых внутренних сигналов и переменных приводится далее в виде комментариев.

Третий раздел архитектурного тела описания объекта OPB_BRAM содержит последовательность выражений, выполняющих преобразование структуры слов данных из формата Little-endian в формат Big-endian. Содержимое этого раздела выглядит следующим образом:

```
--Swap BRAM Little-endian Data to Big-endian

BE_to_LE : for I in 0 to 31 generate
opb_dbus_le(I ) <= OPB_DBus(31-I);
bram_write_data_le(I) <= BRAM_Write_Data(31-I);
BRAM_Read_Data(I) <= bram_Read_Data_LE(31-I);
opb_ABus_LE(I) <= OPB_ABus(31-I);
bram_Addr_LE(I) <= BRAM_Addr(31-I);
end generate BE_to_LE;
```

В четвертом разделе архитектурного тела описания объекта OPB_BRAM представлены выражения, предназначенные для создания экземпляров компонентов базовых элементов ОЗУ, образующих модуль блочной памяти. Этот раздел содержит следующий VHDL-код:

```
--BRAM Instantiation

All_Brams : for I in 0 to C_NO_BRAMS-1 generate
By_8 : if (C_NO_BRAMS = 4) generate
RAMB16_S9_S9_I : RAMB16_S9_S9
port map (
DIA => opb_DBUS_LE(((I+1)*8-1) downto I*8), --[in std_logic_vector(7 downto 0)]
DIB =>bram_Write_Data_LE(((I+1)*8)-1 downto I*8), --[in std_logic_vector (7 downto 0)]
DIPA => null_1, -- [in std_logic_vector (7 downto 0)]
DIPB => null_1, -- [in std_logic_vector (7 downto 0)]
ENA => '1', -- [in std_ulogic]
ENB => '1', -- [in std_ulogic]
WEA => opb_WE(I), -- [in std_ulogic]
WEB => BRAM_WE(I), -- [in std_ulogic]
SSRA => '0', -- [in std_ulogic]
SSRB => '0', -- [in std_ulogic]
CLKA => OPB_Clk, -- [in std_ulogic]
CLKB => BRAM_Clk, -- [in std_ulogic]
ADDRA => opb_ABus_LE(12 downto 2), -- [in std_logic_vector (10 downto 0)]
ADDRB => bram_Addr_LE(12 downto 2), -- [in std_logic_vector (10 downto 0)]
DOA=>opb_BRAM_DBus_LE_I(((I+1)*8-1)downto I*8),--[out std_logic_vector(7 downto 0)]
DOB =>bram_Read_Data_LE(((I+1)*8-1) downto I*8),--[out std_logic_vector(7 downto 0)]
DOPA => open, -- [out std_logic_vector (0 downto 0)]
DOPB => open -- [out std_logic_vector (0 downto 0)]
);
end generate By_8;
```

В представленном разделе все выражения, в которых описывается соответствие портов компонентов и используемых внутренних сигналов и переменных, сопровождаются комментариями, в которых указывается тип этих сигналов и переменных.

Литература

1. Зотов В. MicroBlaze — семейство тридцатидвухразрядных микропроцессорных ядер, реализуемых на основе ПЛИС фирмы Xilinx // Компоненты и технологии. 2003. № 9.
2. Зотов В. Система команд микропроцессорного ядра MicroBlaze // Компоненты и технологии. 2004. № 1–3.
3. Зотов В. Embedded Development Kit — система проектирования встраиваемых микропроцессорных систем на основе ПЛИС серий FPGA фирмы Xilinx. 2004. № 4.
4. Кнышев Д. А., Кузелин М. О. ПЛИС фирмы Xilinx: описание структуры основных семейств. М.: Издательский дом «Додэка-XXI». 2001.
5. Кузелин М. ПЛИС фирмы Xilinx: семейство Spartan™-II // Компоненты и технологии. 2001. № 3.
6. Кузелин М. ПЛИС фирмы Xilinx: семейство Virtex™-II // Chip News / Инженерная микроэлектроника. 2002. № 2.
7. Spartan-3 FPGA Handbook. Xilinx Inc. 2003.