Продолжение. Начало в № 2`2010

Разработка VHDL-описаний цифровых устройств, проектируемых на основе ПЛИС фирмы Xilinx, с использованием шаблонов САПР ISE Design Suite

В третьей части статьи рассматриваются образцы применения библиотечных макросов, предназначенных для описания элементов проектируемых устройств, реализуемых на базе специализированных аппаратных блоков кристаллов программируемой логики с архитектурой FPGA [1], которые находятся в папке Device Macro Instantiation. Основное внимание уделяется шаблонам описания компонентов, предназначенных для реализации на базе ПЛИС семейств Spartan-6 LX и Spartan-6 LXT [12].

Валерий ЗОТОВ
walerry@km.ru

Шаблоны описаний экземпляров элементов, реализуемых на базе специализированных аппаратных блоков кристаллов программируемой логики с архитектурой FPGA

В составе современных кристаллов программируемой логики с архитектурой FPGA кроме стандартных логических ресурсов представлены различные специализированные аппаратные блоки [10–13], в том числе секции цифровой обработки сигналов. Средства разработки проектов и программирования ПЛИС Xilinx ISE Design Suite предоставляют несколько способов быстрой подготовки описаний элементов, реализуемых на базе таких блоков. Для этих целей могут эффективно применяться генератор параметризированных модулей CORE Generator [17] и «мастер» Architecture Wizard [18]. Кроме того, в САПР серии Xilinx ISE Design Suite имеется ряд шаблонов, которые содержат образцы использования библиотечных макросов, предназначенных для описания элементов проектируемых устройств, реализуемых на базе специализированных аппаратных блоков кристаллов программируемой логики с архитектурой FPGA. Эти шаблоны объединены в папку ***Device Macro Instantiation***, структура которой показана на рис. 12.

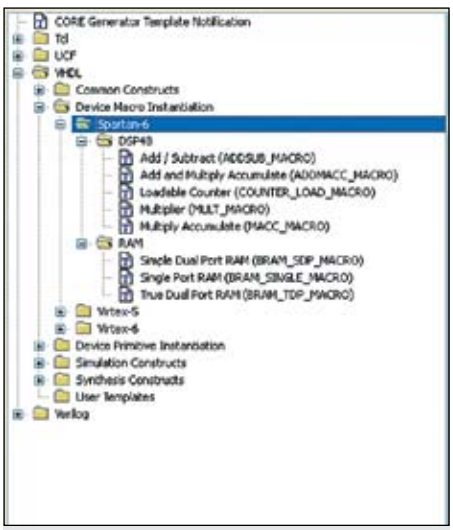

Рис. 12. Структура раздела Device Macro Instantiation шаблонов языка VHDL

Папка ***Device Macro Instantiation*** содержит несколько разделов, названия которых совпадают с обозначением серий ПЛИС с архитектурой FPGA, для которых предназначены шаблоны, находящиеся в этих разделах.

Шаблоны описаний экземпляров компонентов, выполненных в форме библиотечных макросов, предназначенных для реализации на базе специализированных аппаратных ресурсов ПЛИС серии Spartan-6

Раздел ***Spartan-6***, представленный в папке ***Device Macro Instantiation*** (рис. 12), включает в себя два подраздела — ***DSP48*** и ***RAM***. В подразделе ***DSP48*** находятся шаблоны описаний экземпляров элементов, которые предназначены для реализации на базе аппаратных секций цифровой обработки сигналов DSP48 A1 кристаллов программируемой логики семейств Spartan-6 LX и Spartan-6 LXT.

Add/Subtract (ADDSUB_MACRO) содержит конструкцию, предназначенную для описания экземпляра сумматора/вычитающего устройства, который реализуется на основе аппаратных секций цифровой обработки сигналов DSP48A1 в ПЛИС серии Spartan-6. Данный компонент выполнен в форме макроса ADDSUB_MACRO, который позволяет формировать описания сумматоров/вычитающих устройств с различной разрядностью:

```
-- ADDSUB_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (ADDSUB_MACRO_inst) and/or the port declarations
-- code: after the "=>" assignment maybe changed to properly
--       : reference and connect this function to the design.
--       : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--       : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- ADDSUB_MACRO: Variable width & latency - Adder / Subtrator implemented in a DSP48A1
   --          Spartan-6
   -- Xilinx HDL Language Template, version 11.4
--
   ADDSUB_MACRO_inst : ADDSUB_MACRO
   generic map (
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
            LATENCY => 2, -- Desired clock cycle latency, 0-2
            WIDTH => 48) -- Input / Output bus width, 1-48
   port map (
            CARRYOUT => CARRYOUT, -- 1-bit carry-out output signal
```

```
        RESULT => RESULT, -- Add/sub result output, width defined by WIDTH generic
        A => A, -- Input A bus, width defined by WIDTH generic
        ADD_SUB => ADD_SUB, -- 1-bit add/sub input, high selects add, low selects subtract
        B => B, -- Input B bus, width defined by WIDTH generic
        CARRYIN => CARRYIN, -- 1-bit carry-in input
        CE => CE, -- 1-bit clock enable input
        CLK =>CLK, -- 1-bit clock input
        RST => RST -- 1-bit active high synchronous reset
        );
  -- End of ADDSUB_MACRO_inst instantiation
```

Макрос ADDSUB_MACRO определен в составе пакета ***vcomponents*** библиотеки UNIMACRO. Поэтому для его использования необходимо включить в состав VHDL-описания разрабатываемого устройства ссылки на эту библиотеку и указанный пакет. Кроме того, нужно также добавить ссылки на библиотеку UNISIM и пакет ***vcomponents*** этой библиотеки, который содержит определения примитивов фирмы Xilinx, используемых в составе макросов. Выражения, предназначенные для включения указанных библиотек и их пакетов, представлены в начале шаблона ***Add/Subtract (ADDSUB_MACRO)***, после комментариев, которые содержат краткую информацию о рассматриваемом макросе. Эти выражения следует вставить в формируемое VHDL-описание перед декларацией объекта, представляющего разрабатываемое устройство.

В макросе ADDSUB_MACRO предусмотрено три настраиваемых параметра:

- DEVICE — определяет серию ПЛИС, на базе которой реализуется данный элемент (в рассматриваемом шаблоне по умолчанию используется значение “SPARTAN6”).
- LATENCY — указывает ожидаемое значение задержки, выраженное в количестве периодов тактового сигнала (по умолчанию установлено значение, равное 2).
- WIDTH — определяет разрядность входных и выходной шин данных сумматора/вычитающего устройства (значение, установленное по умолчанию, равно 48).

В описании интерфейса рассматриваемого макроса используется следующая система условных обозначений входных и выходных портов:

- CARRYOUT — выход сигнала переноса.
- RESULT [WIDTH -1:0] — выходная шина данных, представляющая значение результата вычислений суммы или разности.
- A [WIDTH -1:0] — входная шина данных, совокупность сигналов которой определяет значение первого операнда.
- ADD_SUB — вход сигнала управления выбором выполняемой операции (сложения или вычитания).
- B [WIDTH -1:0] — входная шина данных, совокупность сигналов которой определяет значение второго операнда.
- CARRYIN — вход сигнала переноса.
- CE — вход сигнала разрешения синхронизации.
- CLK — вход тактового сигнала.
- RST — вход сигнала синхронного сброса.

После включения текста шаблона ***Add/Subtract (ADDSUB_MACRO)*** в состав формируемого VHDL-описания, в первую очередь, нужно вместо ADDSUB_MACRO_inst указать идентификатор соответствующего экземпляра сумматора/вычитающего устройства. Затем необходимо задать требуемое значение разрядности для входных и выходной шин данных. Это значение может находиться в диапазоне от 1 до 48 двоичных разрядов, который обусловлен особенностями архитектуры аппаратных секций цифровой обработки сигналов DSP48 A1 в кристаллах программируемой логики серии Spartan-6 [12]. Далее следует указать идентификаторы сигналов, которые должны быть подключены к соответствующим интерфейсным портам макроса ADDSUB_MACRO.

Add and Multiple Accumulate (ADDMACC_MACRO) представляет собой шаблон описания экземпляра компонента, выполняющего операции умножения с накоплением, с использованием предварительного сумматора, представленного в составе архитектуры аппаратных секций цифровой обработки сигналов DSP48 A1 в ПЛИС семейств Spartan-6 LX и Spartan-6 LXT. Такие компоненты находят широкое применение в составе цифровых устройств различного назначения. Наиболее часто данные элементы востребованы в процессе проектирования устройств цифровой обработки сигналов и, прежде всего, при разработке многозвенных цифровых фильтров. Экземпляры указанных элементов могут создаваться на основе библиотечного макроса ADDMACC_MACRO, который предоставляет возможность формирования описаний умножителей-накопителей с функцией предварительного суммирования операндов с различной разрядностью:

```
-- ADDMACC_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (ADDMACC_MACRO_inst) and/or the port declarations
-- code: after the "=>" assignment maybe changed to properly
--       : reference and connect this function to the design.
--       : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--       : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- ADDMACC_MACRO: Add and Multiple Accumulate Function implemented in a DSP48A1
   --          Spartan-6
   -- Xilinx HDL Language Template, version 11.4
--
   ADDMACC_MACRO_inst : ADDMACC_MACRO
```

```
   generic map (
        DEVICE => "SPARTAN6", -- Target Device: "VIRTEX6", "SPARTAN6"
        LATENCY => 4, -- Desired clock cycle latency, 1-4
        WIDTH_PREADD => 18, -- Pre-Adder input bus width, 1-18
        WIDTH_MULTIPLIER => 18, -- Multiplier input bus width, 1-18
        WIDTH_PRODUCT => 48) -- MACC output width, 1-48
   port map (
        PRODUCT => PRODUCT, -- MACC result output, width defined by WIDTH_PRODUCT generic
        MULTIPLIER => MULTIPLIER, -- Multiplier data input, width determined by WIDTH_MULTIPLIER generic
        PREADDER1 => PREADDER1, -- Preadder data input, width determined by WIDTH_PREADDER generic
        PREADDER2 => PREADDER2, -- Preadder data input, width determined by WIDTH_PREADDER generic
        CARRYIN => CARRYIN, -- 1-bit carry-in input
        CE => CE, -- 1-bit input clock enable
        CLK => CLK, -- 1-bit clock input
        LOAD => LOAD, -- 1-bit accumulator load input
        LOAD_DATA => LOAD_DATA, -- Accumulator load data input, width defined by WIDTH_PRODUCT generic
        RST => RST -- 1-bit input active high synchronous reset
        );
   -- End of ADDMACC_MACRO_inst instantiation
```

В начале основного текста шаблона ***Add and Multiple Accumulate (ADDMACC_MACRO)*** представлены выражения, предназначенные для создания ссылок на библиотеки макросов и примитивов UNIMACRO и UNISIM соответственно, и их пакеты ***vcomponents***. Эти строки нужно вырезать и поместить в соответствующий раздел формируемого VHDL-описания разрабатываемого устройства.

Макрос ADDMACC_MACRO имеет пять настраиваемых параметров:

- WIDTH_PREADD — определяет количество двоичных разрядов входных шин данных предварительного сумматора.
- WIDTH_MULTIPLIER — устанавливает разрядность входных шин данных аппаратного умножителя.
- WIDTH_PRODUCT — определяет количество двоичных разрядов выходной шины данных, на которой отображается результат вычислений.
- DEVICE и LATENCY — имеют то же назначение, что и аналогичные параметры в предыдущем шаблоне.

Система условных обозначений интерфейсных портов макроса умножителя-накопителя с предварительным сумматором включает в себя следующие идентификаторы:

- PRODUCT [WIDTH_PRODUCT -1:0] — выходная шина данных, представляющая значение результата выполнения операций умножения с накоплением и предварительным суммированием входных данных;
- MULTIPLIER [WIDTH_MULTIPLIER -1:0] — входная шина данных, совокупность сигналов которой определяет значение одного из операндов умножителя;
- PREADDER1 [WIDTH_PREADDER -1:0] — входная шина данных, совокупность сигналов которой определяет значение первого слагаемого предварительного сумматора;
- PREADDER2 [WIDTH_PREADDER -1:0] — входная шина данных, совокупность сигналов которой определяет значение второго слагаемого предварительного сумматора;

- LOAD — вход управления загрузкой данных в аккумулятор;
- LOAD_DATA [WIDTH_PRODUCT -1:0] — входная шина данных, предназначенных для непосредственной загрузки в аккумулятор;
- CARRYIN — вход сигнала переноса;
- CE — вход сигнала разрешения синхронизации;
- CLK — вход тактового сигнала;
- RST — вход сигнала синхронного сброса.

Для практического использования рассматриваемого шаблона в составе формируемого VHDL-описания, прежде всего, следует заменить ADDMACC_MACRO_inst соответствующим идентификатором создаваемого экземпляра рассматриваемого компонента. Далее следует указать требуемые значения разрядности для входных и выходной шин данных. При этом нужно учитывать, что количество разрядов для входных шин данных предварительного сумматора и входной шины данных аппаратного умножителя можно выбрать в диапазоне от 1 до 18. Значение разрядности для выходной шины данных макроса ADDMACC_MACRO может находиться в диапазоне от 1 до 48 двоичных разрядов. После этого необходимо вставить в текст шаблона идентификаторы сигналов, которые должны быть подключены к соответствующим входным и выходным портам макроса умножителя-накопителя с функцией предварительного суммирования.

Loadable counter (COUNTER_LOAD_MACRO) включает в себя шаблон описания экземпляра программируемого реверсивного счетчика, реализуемого на базе ресурсов аппаратной секции цифровой обработки сигналов DSP48 A1 кристаллов программируемой логики серии Spartan-6:

```
--COUNTER_LOAD_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (COUNTER_LOAD_MACRO_inst) and/or the port declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--        : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- COUNTER_LOAD_MACRO: Loadable variable counter implemented in a DSP48A1
   --          Spartan-6
   -- Xilinx HDL Language Template, version 11.4
--
   COUNTER_LOAD_MACRO_inst : COUNTER_LOAD_MACRO
   generic map (
            COUNT_BY => X"000000000001", -- Count by value
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
            WIDTH_DATA => 48) -- Counter output bus width, 1-48
   port map (
            Q => Q, -- Counter ouput, width determined by WIDTH_DATA generic
            CLK => CLK, -- 1-bit clock input
            CE => CE, -- 1-bit clock enable input
            DIRECTION => DIRECTION, -- 1-bit up/down count direction input, high is count up
            LOAD => LOAD, -- 1-bit active high load input
            LOAD_DATA => LOAD_DATA, -- Counter load data, width determined by WIDTH_DATA generic
            RST => RST -- 1-bit active high synchronous reset
            );
   -- End of COUNTER_LOAD_MACRO_inst instantiation
```

В макросе COUNTER_LOAD_MACRO предусмотрено три настраиваемых параметра:

- COUNT_BY — определяет шаг счета (значение по умолчанию для этого параметра равно единице).
- DEVICE — указывает серию кристаллов программируемой логики, на базе которой должен быть реализован формируемый счетчик (по умолчанию используется значение "SPARTAN6").
- WIDTH_DATA — определяет разрядность формируемого счетчика (по умолчанию устанавливается максимально возможное количество двоичных разрядов, равное 48).

В описании интерфейса макроса COUNTER_LOAD_MACRO применяется следующая система условных обозначений входных и выходных портов:

- Q [WIDTH_DATA -1:0] — выходы соответствующих разрядов счетчика, объединенные в шину, количество разрядов в которой определяется значением параметра WIDTH_DATA.
- CLK — вход тактового сигнала (вход счета).
- CE — вход разрешения счета.
- DIRECTION — вход сигнала управления, значение которого определяет направление счета (возрастающее или убывающее).
- LOAD — вход разрешения параллельной загрузки данных в счетчик.
- LOAD_DATA [WIDTH_DATA -1:0] — параллельные входы данных, объединенные в шину, количество разрядов в которой определяется значением параметра WIDTH_DATA.
- RST — вход сигнала синхронного сброса.

При включении текста шаблона ***Loadable counter (COUNTER_LOAD_MACRO)*** в состав формируемого VHDL-описания нужно, прежде всего, поместить в начало создаваемого модуля ссылки на используемые библиотеки UNISIM, UNIMACRO и пакеты этих библиотек, которые расположены сразу после комментариев, содержащих краткую справочную информацию о макросе COUNTER_LOAD_MACRO. Затем следует заменить ***COUNTER_LOAD_MACRO_inst*** идентификатором соответствующего экземпляра счетчика. Далее необходимо указать требуемую разрядность счетчика, используя настраиваемый параметр WIDTH_DATA. Макрос COUNTER_LOAD_MACRO позволяет формировать описания счетчиков, разрядность которых может выбираться в диапазоне от 1 до 48 разрядов. Этот диапазон соответствует архитектурным особенностям секций цифровой обработки сигналов DSP48 A1, применяемых в составе кристаллов программируемой логики семейств Spartan-6 LX и Spartan-6 LXT. В завершение процесса подготовки описания конкретного экземпляра счетчика следует указать идентификаторы сигналов, подключаемых к соответствующим интерфейсным портам этого компонента.

Multiplier (MULT_MACRO) содержит образец использования библиотечного макроса MULT_MACRO, предназначенного для формирования описаний аппаратных умножителей, реализуемых на основе ресурсов секций цифровой обработки сигналов DSP48 A1 в ПЛИС серии Spartan-6:

```
-- MULT_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (MULT_MACRO_inst) and/or the port declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--        : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- MULT_MACRO: Multiply Function implemented in a DSP48A1
   --          Spartan-6
   -- Xilinx HDL Language Template, version 11.4
--
   MULT_MACRO_inst : MULT_MACRO
   generic map (
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
            LATENCY => 3, -- Desired clock cycle latency, 0-4
            WIDTH_A => 18, -- Multiplier A-input bus width, 1-18
            WIDTH_B => 18, -- Multiplier B-input bus width, 1-18
            WIDTH_ P => 36) -- Multiplier ouput bus width, 1-36
   port map (
            P => P, -- Multiplier ouput bus, width determined by WIDTH_P generic
            A => A, -- Multiplier input A bus, width determined by WIDTH_A generic
            B => B, -- Multiplier input B bus, width determined by WIDTH_B generic
            CE => CE, -- 1-bit active high input clock enable
            CLK => CLK, -- 1-bit positive edge clock input
            RST => RST -- 1-bit input active high reset
            );
   -- End of MULT_MACRO_inst instantiation
```

В макросе MULT_MACRO используются следующие настраиваемые параметры:

- DEVICE и LATENCY — имеют то же назначение, что и аналогичные параметры в шаблоне ***Add/Subtract (ADDSUB_MACRO)***.
- WIDTH_A — устанавливает значение разрядности входной шины данных *A*, совокупность сигналов которой определяет значение первого сомножителя.

- WIDTH_B — задает количество разрядов входной шины данных *B*, совокупность сигналов которой определяет значение второго сомножителя.
- WIDTH_P — устанавливает разрядность выходной шины данных умножителя, совокупность сигналов которой соответствует значению произведения.

В системе условных обозначений интерфейсных портов макроса умножителя представлены следующие идентификаторы входов и выходов:

- A [WIDTH_A -1:0] — входная шина данных умножителя, совокупность сигналов которой определяет значение первого сомножителя.
- B [WIDTH_B -1:0] — входная шина данных умножителя, совокупность сигналов которой определяет значение второго сомножителя.
- P [WIDTH_P -1:0] — выходная шина данных умножителя, совокупность значений сигналов которой образует результат вычисления произведения.
- CE — вход сигнала разрешения синхронизации.
- CLK — вход тактового сигнала.
- RST — вход сигнала сброса.

При формировании описания экземпляра аппаратного умножителя с помощью шаблона ***Multiplier (MULT_MACRO)*** нужно поместить совокупность ссылок на используемые библиотеки макросов и примитивов, приведенных в начале указанного шаблона, перед объявлением объекта описания. Далее следует указать идентификатор конкретного экземпляра умножителя вместо MULT_MACRO_inst. Затем необходимо задать требуемую разрядность входных шин умножителя, используя настраиваемые параметры WIDTH_A и WIDTH_B. При этом нужно учитывать, что шины *A* и *B* могут содержать от одного до восемнадцати двоичных разрядов. При необходимости можно изменить значение разрядности выходной шины умножителя, предлагаемое по умолчанию.

Multiply Accumulate (MACC_MACRO) предоставляет образец применения библиотечного макроса MACC_MACRO для подготовки описания экземпляра компонента, выполняющего операции умножения с накоплением, который предназначен для реализации на базе аппаратной секции цифровой обработки сигналов DSP48 A1 кристаллов программируемой логики семейств Spartan-6 LX и Spartan-6 LXT:

```
-- MACC_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (MACC_MACRO_inst) and/or the port declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--        : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- MACC_MACRO: Multiple Accumulate Function implemented in a DSP48A1
   --          Spartan-6
   -- Xilinx HDL Language Template, version 11.4
--
   MACC_MACRO_inst : MACC_MACRO
   generic map (
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
            LATENCY => 3, -- Desired clock cycle latency, 1-4
            WIDTH_A => 18, -- Multiplier A-input bus width, 1-18
            WIDTH_B => 18, -- Multiplier B-input bus width, 1-18
            WIDTH_P => 48) -- Accumulator output bus width, 1-48
   port map (
            P => P, -- MACC ouput bus, width determined by WIDTH_P generic
            A => A, -- MACC input A bus, width determined by WIDTH_A generic
            ADDSUB => ADDSUB, -- 1-bit add/sub input, high selects add, low selects subtract
            B => B, -- MACC input B bus, width determined by WIDTH_B generic
            CARRYIN => CARRYIN, -- 1-bit carry-in input to accumulator
            CE => CE, -- 1-bit active high input clock enable
            CLK => CLK, -- 1-bit positive edge clock input
            LOAD => LOAD, -- 1-bit active high input load accumulator enable
            LOAD_DATA => LOAD_DATA, -- Load accumulator input data,
              -- width determined by WIDTH_P generic
            RST => RST -- 1-bit input active high reset
            );
-- End of MACC_MACRO_inst instantiation
```

Макрос MACC_MACRO содержит пять настраиваемых параметров — DEVICE, LATENCY, WIDTH_A, WIDTH_B, WIDTH_P, которые имеют то же предназначение, что и в шаблоне описания экземпляра аппаратного умножителя, рассмотренном выше.

Система условных обозначений входных и выходных портов, используемая в описании интерфейса макроса умножителя-накопителя, включает в себя следующие идентификаторы:

- P [WIDTH_P -1:0] — выходная шина данных умножителя-накопителя, совокупность сигналов которой представляет собой значение результата вычислений.
- A [WIDTH_A -1:0] — входная шина данных умножителя, совокупность сигналов которой определяет значение первого сомножителя.
- ADDSUB — вход сигнала управления выбором операции (сложения или вычитания), выполняемой аккумулятором.
- B [WIDTH_B -1:0] — входная шина данных умножителя, совокупность сигналов которой определяет значение второго сомножителя.
- CARRYIN — вход сигнала переноса.
- CE — вход сигнала разрешения синхронизации.
- CLK — вход тактового сигнала.
- LOAD — вход управления загрузкой данных в аккумулятор.
- LOAD_DATA [WIDTH_P -1:0] — входная шина данных, предназначенных для непосредственной загрузки в аккумулятор.
- RST — вход сигнала синхронного сброса.

После включения текста шаблона ***Multiply Accumulate (MACC_MACRO)*** в состав формируемого описания необходимо переместить совокупность выражений, содержащих ссылки на используемые библиотеки и их пакеты, в начало создаваемого модуля. Затем следует заменить MACC_MACRO_inst идентификатором соответствующего экземпляра компонента, выполняющего операции умножения с накоплением. Выбор значений разрядности входных и выходной шин данных осуществляется таким же образом, как и при использовании шаблона описания аппаратного умножителя MULT_MACRO.

Подраздел *RAM* в разделе ***Spartan-6***, который представлен в папке ***Device Macro Instantiation*** (рис. 12), объединяет образцы применения библиотечных макросов для подготовки описаний элементов оперативных запоминающих устройств (ОЗУ) различного типа, реализуемых на основе блочной памяти Block RAM кристаллов программируемой логики одноименной серии. При использовании этих макросов нужно переместить выражения ссылок на используемые библиотеки и их пакеты в начало создаваемого модуля описания.

Simple Dual Port RAM (BRAM_SDP_MACRO) содержит конструкцию, предназначенную для описания экземпляра двухпортового ОЗУ с фиксированной функцией портов записи и чтения данных, реализуемого на основе блочной памяти Block RAM ПЛИС семейств Spartan-6 LX и Spartan-6 LXT. Для описания элементов оперативной памяти указанного типа используется библиотечный макрос BRAM_SDP_MACRO. В компонентах, формируемых с помощью этого макроса, один порт используется только для записи данных в ОЗУ, а второй — только для чтения информации из памяти:

```
--BRAM_SDP_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (BRAM_SDP_MACRO_inst) and/or the port declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
--for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--        : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- BRAM_SDP_MACRO: Simple Dual Port RAM
```

```
--          Spartan-6
-- Xilinx HDL Language Template, version 11.4
--
BRAM_SDP_MACRO_inst : BRAM_SDP_MACRO
generic map (
        BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or
"36Kb"
        DEVICE => "SPARTAN6" -- Target device: "VIRTEX5",
"VIRTEX6", "SPARTAN6"
        WRITE_WIDTH => 0, -- Valid values are 1-72 (37-72 only
valid when BRAM_SIZE="36Kb")
        READ_WIDTH => 0, -- Valid values are 1-72 (37-72 only
valid when BRAM_SIZE="36Kb")
        DO_REG => 0, -- Optional output register (0 or 1)
        INIT_FILE => "NONE",
        SIM_COLLISION_CHECK => "ALL", -- Collision check
enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or
"NONE"
        SIM_MODE => "SAFE", -- Simulation: "SAFE" vs
"FAST",
            -- see "Synthesis and Simulation Design Guide" for
details
        SRVAL => X"0000000000000000", -- Set/Reset value for
port output
        INIT => X"0000000000000000", -- Initial values on
output port
      -- The following INIT_xx declarations specify the initial
contents of the RAM
INIT_00 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_01 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_02 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_03 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_04 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_05 => X"0000000000000000000000000000000000000000000
0000000000000000000",
-- …
INIT_3E => X"0000000000000000000000000000000000000000000
0000000000000000000",
INIT_3F => X"0000000000000000000000000000000000000000000
0000000000000000000",
      -- The next set of INITP_xx are for the parity bits
INITP_00 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_01 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_02 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_03 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_04 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_05 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_06 => X"0000000000000000000000000000000000000000000
0000000000000000000",
INITP_07 => X"0000000000000000000000000000000000000000000
0000000000000000000")
port map (
        DO => DO, -- Output read data port
        DI => DI, -- Input write data port
        RDADDR => RDADDR, -- Input read address
        RDCLK => RDCLK, -- Input read clock
        RDEN => RDEN, -- Input read port enable
        REGCE => REGCE, -- Input read output register enable
        RST => RST, -- Input reset
        WE => WE, -- Input write enable
        WRADDR => WRADDR, -- Input write address
        WRCLK => WRCLK, -- Input write clock
        WREN => WREN -- Input write port enable
        );
-- End of BRAM_SDP_MACRO_inst instantiation
```

В макросе BRAM_SDP_MACRO предусмотрен ряд настраиваемых параметров, которые предназначены для определения конфигурации формируемого элемента двухпортовой оперативной памяти:

- BRAM_SIZE — задает информационную емкость создаваемого экземпляра ОЗУ (по умолчанию создается элемент оперативной памяти объемом 18 кбит).
- DEVICE — указывает серию ПЛИС, на базе которой реализуется формируемый компонент (по умолчанию для этого параметра установлено значение "SPARTAN6").
- WRITE_WIDTH — определяет разрядность входного порта, используемого для записи данных в ОЗУ.
- READ_WIDTH — устанавливает разрядность выходного порта, используемого для чтения данных из ОЗУ.
- DO_REG — позволяет задействовать выходной регистр в составе формируемого экземпляра двухпортового ОЗУ (по умолчанию предлагается нулевое значение, при котором выходной регистр не используется).
- INIT_FILE — определяет возможность применения файла инициализации содержимого создаваемого элемента оперативной памяти (по умолчанию установлено значение "NONE", при котором файл инициализации содержимого ОЗУ не используется).
- SIM_COLLISION_CHECK — задает режим обработки различных конфликтных ситуаций при осуществлении операций записи и чтения данных в процессе моделирования (по умолчанию предлагается значение "ALL").
- SIM_MODE — устанавливает режим моделирования формируемого экземпляра двухпортового ОЗУ (по умолчанию предлагается значение "SAFE").
- SRVAL — задает состояние выходного порта при активном уровне сигнала сброса/установки (по умолчанию устанавливается значение X"0000000000000000").
- INIT — определяет начальное состояние выходного порта данных (по умолчанию принимается значение X"0000000000000000").
- INIT_00 – INIT_3F — позволяют осуществить инициализацию содержимого соответствующих ячеек основной памяти (по умолчанию во все ячейки ОЗУ заносится нулевое значение).
- INITP_00 – INITP_07 — предоставляют возможность инициализации содержимого ячеек памяти, используемых для организации контроля четности.

Система условных обозначений, применяемых при описании интерфейса рассматриваемого макроса элемента двухпортовой оперативной памяти, включает в себя следующие идентификаторы входов и выходов:

- DO — информационные выходы порта чтения данных из ОЗУ;
- DI — информационные входы порта записи данных в ОЗУ;
- DADDR — адресные входы порта чтения данных из ОЗУ;
- RDCLK — вход сигнала синхронизации порта чтения данных;
- RDEN — вход сигнала разрешения порта чтения данных;
- REGCE — вход сигнала разрешения синхронизации для выходного регистра порта чтения данных;
- RST — вход сброса;
- WE — вход сигнала управления записью данных в ячейки элемента оперативной памяти;
- WRADDR — адресные входы порта записи данных в ОЗУ;
- WRCLK — вход сигнала синхронизации порта записи данных;
- WREN — вход сигнала разрешения порта записи данных.

Для формирования законченного описания элемента двухпортовой оперативной памяти на базе представленного шаблона необходимо указать идентификатор конкретного экземпляра ОЗУ вместо BRAM_SDP_MACRO_inst, его информационную емкость с помощью параметра BRAM_SIZE, разрядность портов чтения и записи данных, используя параметры READ_WIDTH и WRITE_WIDTH соответственно. Если в составе формируемого элемента двухпортовой оперативной памяти должен быть задействован выходной регистр, то следует присвоить единичное значение параметру DO_REG. При необходимости инициализации содержимого каких-либо ячеек ОЗУ нужно задать требуемые значения для соответствующих настраиваемых параметров INIT_00 – INIT_3F и INITP_00 – INITP_07.

Single Port RAM (BRAM_SINGLE_MACRO) включает в себя образец описания элемента однопортового ОЗУ, предназначенного для реализации на основе блочной памяти Block RAM в ПЛИС серии Spartan-6. Основой этого описания является библиотечный макрос BRAM_SINGLE_MACRO:

```
-BRAM_SINGLE_MACRO : In order to incorporate this function
into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (BRAM_SINGLE_MACRO_inst) and/or the port
declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs
to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
--        : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture
body---->
--
   -- BRAM_SINGLE_MACRO: Single Port RAM
   -- Spartan-6
   -- Xilinx HDL Language Template, version 11.4
   --
   BRAM_SINGLE_MACRO_inst : BRAM_SINGLE_MACRO
   generic map (
            BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or "36Kb"
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5",
"VIRTEX6", "SPARTAN6"
            DO_REG => 0, -- Optional output register (0 or 1)
            INIT_A => X"000000000", -- Initial values on output
port
            INIT_FILE => "NONE",
            WRITE_WIDTH => 0, -- Valid values are 1-72 (37-72 only
valid when BRAM_SIZE="36Kb")
            READ_WIDTH => 0, -- Valid values are 1-72 (37-72 only
valid when BRAM_SIZE="36Kb")
            SIM_MODE => "SAFE", -- Simulation: "SAFE" vs "FAST",
                -- see "Synthesis and Simulation Design Guide" for
details
            SRVAL => X"000000000", -- Set/Reset value for port
output
```

```
            WRITE_MODE => "WRITE_FIRST", -- "WRITE_FIRST",
"READ_FIRST" or "NO_CHANGE"
         -- The following INIT_xx declarations specify the initial
contents of the RAM
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
-- ...
   INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
         -- The next set of INITP_xx are for the parity bits
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000")
   port map (
            DO => DO, -- Output data
            ADDR => ADDR, -- Input address
            CLK => CLK, -- Input clock
            DI => DI, -- Input data port
            EN => EN, -- Input RAM enable
            REGCE => REGCE, -- Input output register enable
            RST => RST, -- Input reset
            WE => WE -- Input write enable
            );
   -- End of BRAM_SINGLE_MACRO_inst instantiationend
Behavioral;
```

Требуемый вариант конфигурации элемента однопортовой оперативной памяти, формируемого на основе макроса BRAM_SINGLE_MACRO, выбирается с помощью следующих настраиваемых параметров:

- BRAM_SIZE, DEVICE, DO_REG, INIT_FILE, WRITE_WIDTH, READ_WIDTH, SIM_MODE, SRVAL, INIT_00 – INIT_3F, INITP_00 – INITP_07 — имеют то же предназначение, что и в макросе двухпортовой памяти BRAM_SDP_MACRO.
- INIT_A — определяет начальное состояние выходного порта данных (по умолчанию принимается значение X"000000000").
- WRITE_MODE — устанавливает режим записи информации в ОЗУ (порядок выполнения операций записи и чтения данных при обращении к ячейкам памяти).

В системе условных обозначений интерфейсных портов макроса элемента однопортовой оперативной памяти BRAM_SINGLE_MACRO, кроме идентификаторов входов и выходов, представленных в шаблоне ***Simple Dual Port RAM (BRAM_SDP_MACRO)***, используются также следующие наименования:

- ADDR — адресные входы элемента оперативной памяти;
- EN — вход сигнала разрешения операций чтения и записи данных в ОЗУ.

При практическом использовании шаблона ***Single Port RAM (BRAM_SINGLE_MACRO)*** нужно заменить BRAM_SINGLE_MACRO_inst условным обозначением конкретного экземпляра однопортового ОЗУ, определить его объем, используя параметр настройки BRAM_SIZE, выбрать требуемые значения разрядности представления записываемых и считываемых данных с помощью параметров WRITE_WIDTH и READ_WIDTH соответственно. Кроме того, необходимо указать последовательность выполнения операций записи и чтения данных при обращении к ячейкам запоминающего устройства. Для этого нужно выбрать соответствующее значение параметра WRITE_MODE. Если для этого параметра используется значение "WRITE_FIRST", предлагаемое по умолчанию, то данные, поступающие во входной порт, записываются в соответствующую ячейку памяти (адрес которой задается комбинацией сигналов на адресных входах), после чего сразу передаются на выходы запоминающего устройства. На рис. 13 показаны временные диаграммы, которые поясняют работу элемента оперативной памяти, создаваемого на базе макроса BRAM_SINGLE_MACRO в режиме WRITE FIRST.

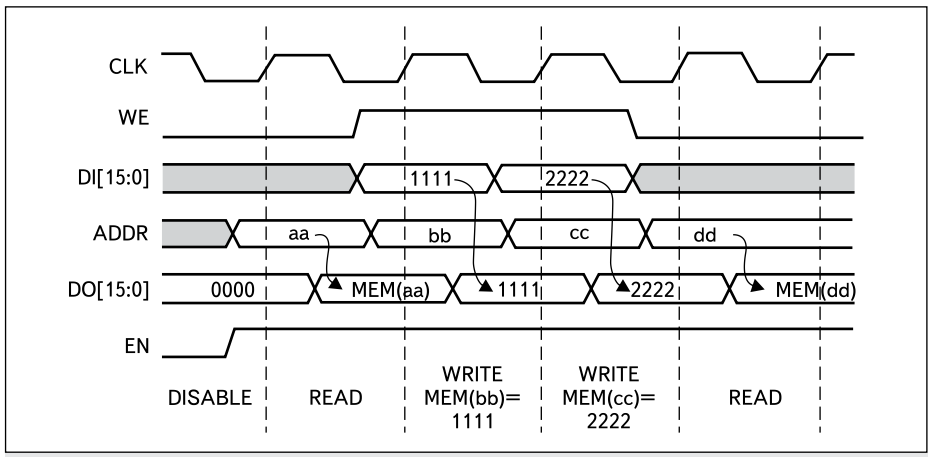

Рис. 13. Временные диаграммы, поясняющие функционирование элементов оперативных запоминающих устройств, реализуемых на основе блочной памяти ПЛИС, использующих режим записи WRITE FIRST

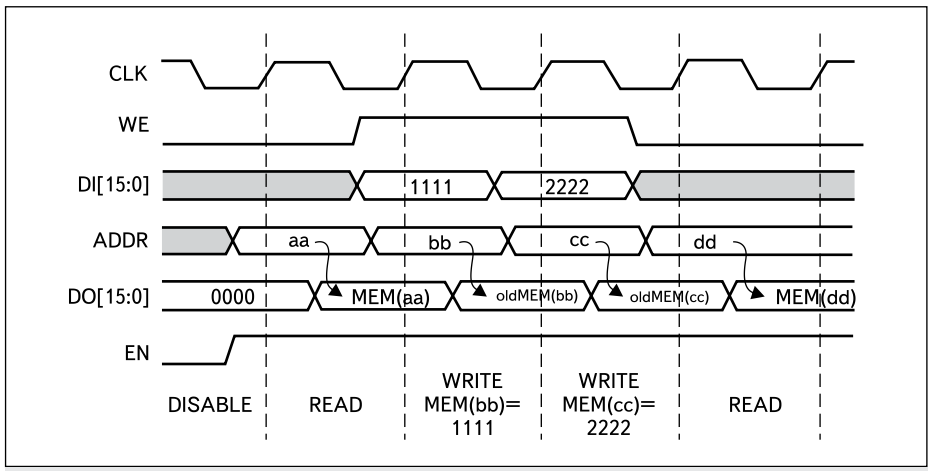

Рис. 14. Временные диаграммы, поясняющие функционирование элементов оперативных запоминающих устройств, реализуемых на основе блочной памяти ПЛИС, использующих режим READ FIRST

При выборе значения "READ_FIRST" в элементе оперативной памяти, формируемом с помощью шаблона ***Single Port RAM (BRAM_SINGLE_MACRO)***, будет установлен режим предварительного чтения данных из указанной ячейки перед записью новых данных в эту ячейку. Таким образом, при осуществлении операции записи данных, поступающих во входной порт создаваемого элемента ОЗУ, на его выходах будет отображаться информация, которая содержалась в соответствующей ячейке перед этим (на предыдущем такте). Временные диаграммы, демонстрирующие работу элемента однопортовой оперативной памяти, реализуемого на основе блочной памяти кристаллов программируемой логики с архитектурой FPGA, в режиме READ FIRST, приведены на рис. 14.

Когда для параметра WRITE_MODE указывается значение "NO_CHANGE", в создаваемом элементе ОЗУ будет установлен режим блокировки выходов при выполнении операции записи данных. В таком режиме на протяжении всего цикла записи информации в запоминающее устройство его выходы нахо-

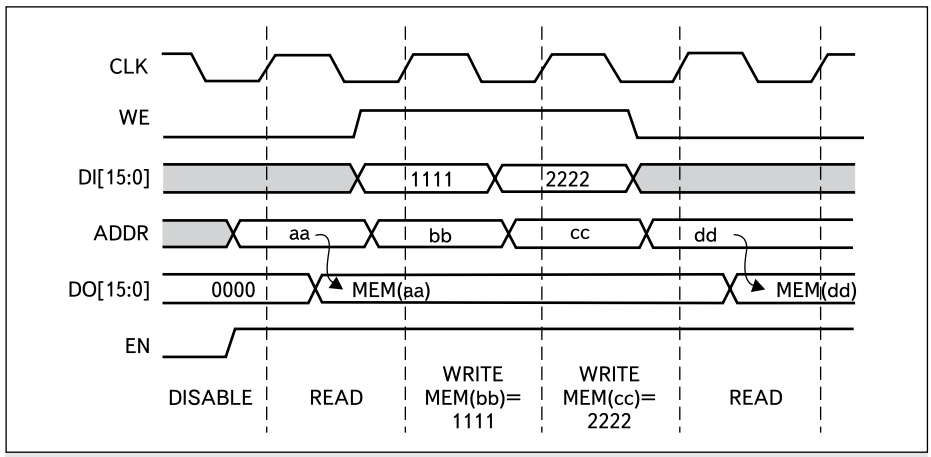

Рис. 15. Временные диаграммы, поясняющие работу элементов оперативных запоминающих устройств, реализуемых на основе блочной памяти ПЛИС, использующих режим NO CHANGE

дятся в зафиксированном состоянии, которое соответствует последним считанным данным, присутствовавшим в момент появления активного уровня сигнала на входе разрешения записи. На рис. 15 представлены временные диаграммы, которые иллюстрируют функционирование элемента оперативной памяти, создаваемого с помощью шаблона ***Single Port RAM (BRAM_SINGLE_MACRO)*** для последующей реализации на базе блочной памяти ПЛИС, в режиме NO CHANGE.

Для определения информации, которая автоматически должна записываться в ячейки формируемого элемента памяти после завершения процесса конфигурирования ПЛИС, следует воспользоваться соответствующими параметрами настройки макроса BRAM_SINGLE_MACRO INIT_00 – INIT_3F и INITP_00 – INITP_07.

True Dual Port RAM (BRAM_TDP_MACRO) представляет собой шаблон, предназначенный для формирования описания экземпляра полнофункционального двухпортового ОЗУ, реализуемого на основе блочной памяти Block RAM кристаллов программируемой логики семейств Spartan-6 LX и Spartan-6 LXT. В отличие от элементов оперативной памяти, создаваемых с помощью шаблона ***Simple Dual Port RAM (BRAM_SDP_MACRO)***, в полнофункциональных двухпортовых ОЗУ операции записи и чтения данных могут выполняться в любом сочетании одновременно по двум портам. Для описания экземпляра полнофункционального двухпортового запоминающего устройства используется библиотечный макрос ***BRAM_TDP_MACRO***:

```
--BRAM_TDP_MACRO : In order to incorporate this function into the design,
-- VHDL : the following instance declaration needs to be placed
-- instance : in the architecture body of the design code. The
-- declaration : (BRAM_TDP_MACRO_inst) and/or the port declarations
-- code : after the "=>" assignment maybe changed to properly
--        : reference and connect this function to the design.
--        : All inputs and outputs must be connected.
--
-- Library : In addition to adding the instance declaration, a use
-- declaration : statement for the UNISIM.vcomponents library needs to be
-- for : added before the entity declaration. This library
-- Xilinx : contains the component declarations for all Xilinx
-- primitives : primitives and points to the models that will be used
-- : for simulation.
--
-- Copy the following four statements and paste them before the
-- Entity declaration, unless they already exist.
--
Library UNISIM;
use UNISIM.vcomponents.all;
--
Library UNIMACRO;
use UNIMACRO.vcomponents.all;
-- <-----Cut code below this line and paste into the architecture body---->
--
   -- BRAM_TDP_MACRO: True Dual Port RAM
   --          Spartan-6
    -- Xilinx HDL Language Template, version 11.4
   --
   BRAM_TDP_MACRO_inst : BRAM_TDP_MACRO
   generic map (
            BRAM_SIZE => "18Kb", -- Target BRAM, "18Kb" or "36Kb"
            DEVICE => "SPARTAN6", -- Target Device: "VIRTEX5", "VIRTEX6", "SPARTAN6"
            DOA_REG => 0, -- Optional port A output register (0 or 1)
            DOB_REG => 0, -- Optional port B output register (0 or 1)
            INIT_A => X"000000000", -- Initial values on A output port
            INIT_B => X"000000000", -- Initial values on B output port
            INIT_FILE => "NONE",
            READ_WIDTH_A => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
            READ_WIDTH_B => 0, -- Valid values are 1-36 (19-36 only valid when BRAM_SIZE="36Kb")
            SIM_COLLISION_CHECK => "ALL", -- Collision check enable "ALL", "WARNING_ONLY", "GENERATE_X_ONLY" or "NONE"
            SIM_MODE => "SAFE", -- Simulation: "SAFE" vs "FAST",
               -- see "Synthesis and Simulation Design Guide" for details
            SRVAL_A => X"000000000", -- Set/Reset value for A port output
            SRVAL_B => X"000000000", -- Set/Reset value for B port output
            WRITE_MODE_A => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
            WRITE_MODE_B => "WRITE_FIRST", -- "WRITE_FIRST", "READ_FIRST" or "NO_CHANGE"
            WRITE_WIDTH_A => 0, -- Valid values are 1, 2, 4, 9, 18 or 36 (36 only valid when BRAM_SIZE="36Kb")
            WRITE_WIDTH_B => 0, -- Valid values are 1, 2, 4, 9, 18 or 36 (36 only valid when BRAM_SIZE="36Kb")
         -- The following INIT_xx declarations specify the initial contents of the RAM
   INIT_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
…
   INIT_3E => X"0000000000000000000000000000000000000000000000000000000000000000",
   INIT_3F => X"0000000000000000000000000000000000000000000000000000000000000000",
         -- The next set of INITP_xx are for the parity bits
   INITP_00 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_01 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_02 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_03 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_04 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_05 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_06 => X"0000000000000000000000000000000000000000000000000000000000000000",
   INITP_07 => X"0000000000000000000000000000000000000000000000000000000000000000")
         port map (
            DOA => DOA, -- Output port-A data
            DOB => DOB, -- Output port-B data
            ADDRA => ADDRA, -- Input port-A address
            ADDRB => ADDRB, -- Input port-B address
            CLKA => CLKA, -- Input port-A clock
            CLKB => CLKB, -- Input port-B clock
            DIA => DIA, -- Input port-A data
            DIB => DIB, -- Input port-B data
            ENA => ENA, -- Input port-A enable
            ENB => ENB, -- Input port-B enable
            REGCEA => REGCEA, -- Input port-A output register enable
            REGCEB => REGCEB, -- Input port-B output register enable
            RSTA => RSTA, -- Input port-A reset
            RSTB => RSTB, -- Input port-B reset
            WEA => WEA, -- Input port-A write enable
            WEB => WEB -- Input port-B write enable
            );
-- End of BRAM_TDP_MACRO_inst instantiation
```

Представленный макрос описания элемента полнофункционального двухпортового ОЗУ содержит совокупность настраиваемых параметров, которые позволяют выбрать требуемый вариант конфигурации запоминающего устройства:

- BRAM_SIZE, DEVICE, INIT_FILE, SIM_COLLISION_CHECK, SIM_MODE, INIT_00 – INIT_3F, INITP_00 – INITP_07 — выполняют те же функции, что и в макросе двухпортовой памяти BRAM_SDP_MACRO.
- DOA_REG — позволяет задействовать выходной регистр в составе порта чтения данных *A* формируемого экземпляра полнофункционального двухпортового ОЗУ (по умолчанию предлагается нулевое значение, при котором выходной регистр не используется).
- DOB_REG — предоставляет возможность использования выходного регистра в составе порта чтения данных *B* формируемого элемента полнофункциональной двухпортовой оперативной памяти (по умолчанию предлагается нулевое значение, при котором выходной регистр не используется).
- INIT_A и INIT_B — устанавливают начальное значение для выходных портов данных *A* и *B* соответственно (по умолчанию принимается значение X"000000000").
- READ_WIDTH_A и READ_WIDTH_B — определяют значения разрядности выходных портов *A* и *B* соответственно, используемых для чтения данных из ОЗУ.

- SRVAL_A и SRVAL_B — задают состояние выходных портов данных *A* и *B* соответственно, при активном уровне сигнала сброса/установки (по умолчанию устанавливается значение X“000000000”).
- WRITE_MODE_A и WRITE_MODE_B — устанавливают режим записи информации (порядок выполнения операций записи и чтения данных при обращении к ячейкам памяти) в полнофункциональном двухпортовом ОЗУ для первого и второго портов соответственно.
- WRITE_WIDTH_A и WRITE_WIDTH_B — определяют разрядность входных портов *A* и *B* соответственно, используемых для записи данных в элементе полнофункциональной двухпортовой оперативной памяти.

В описании интерфейса макроса *BRAM_TDP_MACRO* используются следующие идентификаторы входов и выходов:

- DOA и DOB — информационные выходы портов чтения данных *A* и *B* соответственно;
- ADDRA и ADDRB — адресные входы первого и второго порта соответственно;
- CLKA и CLKB — входы сигнала синхронизации для портов *A* и *B* соответственно;
- DIA и DIB — информационные входы первого и второго порта соответственно;
- ENA и ENB — входы сигнала разрешения для портов *A* и *B* соответственно;
- REGCEA и REGCEB — входы сигнала разрешения синхронизации для выходных регистров первого и второго порта соответственно;
- RSTA и RSTB — входы сброса для портов *A* и *B* соответственно;
- WEA и WEB — входы сигнала записи данных в элемент оперативной памяти для первого и второго порта соответственно.

Для подготовки законченного описания элемента полнофункциональной двухпортовой оперативной памяти с помощью шаблона ***True Dual Port RAM (BRAM_TDP_MACRO)*** нужно выполнить те же действия, что и при использовании библиотечных макросов *BRAM_SDP_MACRO* и *BRAM_SINGLE_MACRO*, которые были рассмотрены выше.

Продолжение следует

Литература

1. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx. Справочное пособие. М.: Горячая линия – Телеком, 2004.
2. Бибило П. Н. Основы языка VHDL. М.: Солон-Р, 2000.
3. Бибило П. Н. Синтез логических схем с использованием языка VHDL. М.: Солон-Р, 2002.
4. Уэйкерли Дж. Ф. Проектирование цифровых устройств. Том 1. М.: Постмаркет, 2002.
5. Поляков А. К. Языки VHDL и Verilog в проектировании цифровой аппаратуры. М.: Солон-Пресс, 2003.
6. 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.
7. Зотов В. Новый инструментальный комплект Spartan-3E Starter Kit для практического освоения методов проектирования встраиваемых микропроцессорных систем на основе ПЛИС семейств FPGA фирмы Xilinx // Компоненты и технологии. 2006. № 10.
8. Зотов В. Новый инструментальный комплект Spartan-3A Starter Kit для практического освоения методов проектирования и отладки цифровых устройств с аппаратной и программной реализацией операций, реализуемых на основе ПЛИС семейств FPGA фирмы Xilinx // Компоненты и технологии. 2007. № 9.
9. Зотов В. Новый инструментальный комплект от компании Avnet на основе ПЛИС FPGA семейства Spartan-3A фирмы Xilinx // Компоненты и технологии. 2008. № 8.
10. Зотов В. Инструментальный модуль компании Avnet для отладки проектов встраиваемых систем, разрабатываемых на базе нового семейства ПЛИС FPGA фирмы Xilinx Virtex-5 FXT // Компоненты и технологии. 2008. № 9.
11. Зотов В. Особенности архитектуры нового поколения высокопроизводительных ПЛИС FPGA фирмы Xilinx серии Virtex-6 // Компоненты и технологии. 2009. № 8.
12. Зотов В. Особенности архитектуры нового поколения ПЛИС FPGA фирмы Xilinx серии Spartan-6 // Компоненты и технологии. 2009. № 9.
13. Зотов В. Новое семейство высокопроизводительных ПЛИС с архитектурой FPGA фирмы Xilinx Virtex-6 HXT // Компоненты и технологии. 2010. № 1.
14. IEEE Standard VHDL Language Reference Manual — IEEE Std 1076–2002.
15. Суворова Е. А., Шейнин Ю. Е. Проектирование цифровых систем на VHDL. СПб.: БХВ-Петербург, 2003.
16. Сергиенко А. М. VHDL для проектирования вычислительных устройств. Киев: ЧП «Корнейчук», ООО «ТИД “ДС”», 2003.
17. Зотов В. Проектирование цифровых устройств, реализуемых на базе ПЛИС FPGA фирмы Xilinx, с использованием средств CORE Generator // Компоненты и технологии. 2006. № 12. 2007. № 1.
18. Зотов В. Разработка компонентов устройств цифровой обработки сигналов, реализуемых на базе аппаратных модулей DSP48E в ПЛИС FPGA серии Virtex-5, с помощью «мастера» Architecture Wizard САПР серии Xilinx ISE // Компоненты и технологии. 2008. № 12. 2009. № 1, 4–7.