*Окончание. Начало в № 12 `2008*

# Разработка компонентов устройств ЦОС, реализуемых на базе аппаратных модулей DSP48E в ПЛИС FPGA серии Virtex-5, с помощью «мастера» Architecture Wizard САПР серии Xilinx ISE

**В шестой части статьи завершается изучение процесса разработки компонентов высокоскоростных устройств ЦОС, предназначенных для реализации на базе аппаратных секций DSP48E в ПЛИС с архитектурой FPGA [1] семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT, осуществляемого с использованием «мастера» Architecture Wizard САПР серии Xilinx ISE [3, 4]. В этой части рассматривается процедура формирования описаний элементов, выполняющих операции умножения с накоплением (вычитанием) с расширенным набором функциональных возможностей. Для каждого варианта структуры указанных элементов, предоставляющего соответствующие дополнительные возможности, приведены примеры сгенерированных описаний, предназначенных для реализации на основе кристаллов серии Virtex-5 [2, 5–7]. Здесь же приведена краткая информация о разработке высокопроизводительных комплексных умножителей-накопителей на основе компонентов, создаваемых с помощью «мастера» Architecture Wizard.**

**Валерий ЗОТОВ**
walerry@km.ru

## Подготовка описаний элементов, осуществляющих операции умножения с накоплением (вычитанием) с расширенным набором функциональных возможностей, реализуемых на базе аппаратных секций DSP48E в ПЛИС серии Virtex-5, с помощью «мастера» Architecture Wizard САПР серии Xilinx ISE

Процесс формирования описаний элементов, выполняющих операции умножения с накоплением (вычитанием) с расширенным набором функциональных возможностей, для последующей реализации на основе аппаратных секций DSP48E в ПЛИС семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT начинается с той же последовательности действий, что и при подготовке описаний аналогичных компонентов с традиционной структурой, содержащей умножитель и обычный аккумулятор. Данная последовательность действий, включающая в себя процедуры запуска «мастера» Architecture Wizard, выбора типа генерируемого компонента, определения комбинации входных и выходных информационных шин данных и их разрядности, была подробно рассмотрена в предыдущей части статьи. После выполнения указанных действий следует перейти к диалоговой панели «мастера» настройки параметров с заголовком *Operation Mode Setup – MACC*, которая предназначена для выбора типа выполняемой операции и структуры создаваемого элемента. Первоначальный вид этой диалоговой панели показан на рис. 30 (см. КиТ № 6 `2009, стр. 51).

Тип операции, выполняемой аккумулятором генерируемого элемента, указывается с помощью группы кнопок с зависимой фиксацией *Add/Subtract control*. В аккумуляторах элементов, выполняющих умножение с накоплением (вычитанием), обладающих набором дополнительных функциональных возможностей, могут быть реализованы те же варианты операций, что и в аналогичных компонентах с традиционной структурой. Если указанные операции в генерируемом элементе должны выполняться с учетом значения внешнего сигнала входного переноса, то нужно установить в состояние «Включено» индикатор *Use a CARRYIN*, находящийся во встроенной панели *Carry option*.

Далее необходимо выбрать один из трех предлагаемых вариантов структуры, предоставляющих возможность реализации дополнительных операций в создаваемом элементе, воспользовавшись для этого группой кнопок с зависимой фиксацией, которые расположены во встроенной панели *OPMODE Control* (рис. 30). Для генерации описания элемента, осуществляющего операции умножения с накоплением (вычитанием) с учетом значений данных, поступающих на входную шину PCIN, предназначенную для каскадного соединения модулей ЦОС, следует нажать кнопку *Use a dedicated cascade result from previous DSP slice in chain (PCIN pin)*. Компоненты с такой конфигурацией часто применяются при проектировании высокоскоростных устройств цифровой обработки сигналов, включающих

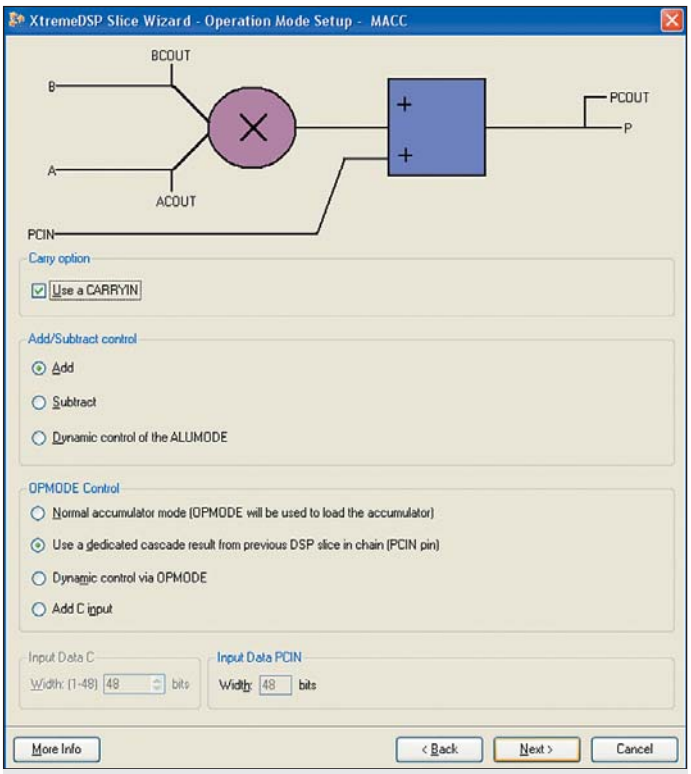


**Рис. 34.** Вид диалоговой панели Operation Mode Setup – MACC «мастера» настройки параметров элементов с использованием результатов вычислений, осуществляемых предшествующим аппаратным модулем DSP48E ПЛИС серии Virtex-5

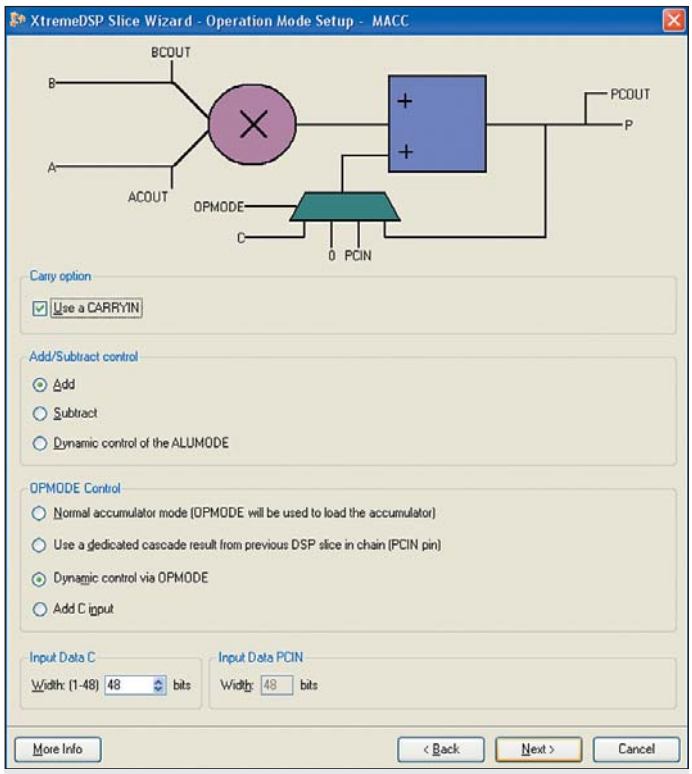


**Рис. 35.** Вид диалоговой панели Operation Mode Setup – MACC «мастера» настройки параметров элементов при выборе варианта с возможностью динамического изменения конфигурации структуры и источников входных данных

в себя несколько звеньев, каждое из которых реализуется на базе отдельной аппаратной секции DSP48E в ПЛИС серии Virtex-5. В таких устройствах операции умножения с накоплением, выполняемые одним звеном, должны осуществляться с использованием результата вычислений, формируемого аппаратной секцией DSP48E, на базе которой реализовано предыдущее звено. В частности, такие компоненты используются в составе устройств, выполняющих операции умножения с накоплением над комплексными значениями.

При установке кнопки *Use a dedicated cascade result from previous DSP slice in chain (PCIN pin)* в нажатое положение диалоговая панель «мастера» настройки параметров элементов, выполняющих операции умножения с накоплением, с заголовком *Operation Mode Setup – MACC* автоматически преобразуется к виду, представленному на рис. 34.

В верхней части этой диалоговой панели отображается выбранный вариант структуры элементов, выполняющих операции умножения с накоплением с использованием результатов вычислений, осуществляемых предшествующей секцией DSP48E ПЛИС серии Virtex-5 в каскадной цепочке аппаратных модулей ЦОС. Кроме того, в нижней части становится доступной встроенная панель с заголовком *Input Data PCIN*, в которой отображается значение параметра *Width*, информирующего о разрядности дополнительной входной шины данных PCIN. Как и в других элементах, генерируемых с помощью «мастера» *Architecture Wizard* для последующей реализации на основе аппаратных секций DSP48E, эта шина имеет фиксированное значение разрядности, которое составляет 48 бит.

Чтобы сформировать описание компонента, осуществляющего операции умножения с накоплением (вычитанием), который обладает возможностью динамического изменения конфигурации структуры в процессе его функционирования, необходимо нажать кнопку *Dynamic control via OPMODE*. В этом случае в состав интерфейса генерируемого элемента автоматически добавляется входная 7-разрядная шина OPMODE, которая предназначена для управления выбором источников входных данных и выполняемой функции арифметическо-логического блока аппаратной секции DSP48E, используемой для реализации создаваемого компонента. Сформированный элемент с динамически изменяемой конфигурацией структуры позволяет выполнять различные функции, поддерживаемые архитектурой аппаратных секций DSP48E ПЛИС серии Virtex-5 [5–7].

После переключения в нажатое положение кнопки *Dynamic control via OPMODE* диалоговая панель с заголовком *Operation Mode Setup – MACC* приобретает вид, изображенный на рис. 35.

В нижней части представленной диалоговой панели в доступное состояние автоматически переключается не только встроенная панель с заголовком *Input Data PCIN*, но и встроенная панель *Input Data C*. Последняя встроенная панель содержит поле редактирования *Width*, которое предназначено для определения значения разрядности дополнительной входной шины данных C. В отличие от шины PCIN разрядность шины данных C может выбираться в пределах, определяемых архитектурой аппаратных секций DSP48E ПЛИС серии Virtex-5. Требуемое значение разрядности может быть указано с помощью клавиатуры непосредственно в поле редактирования *Width* или установлено с помощью двух кнопок, расположенных в правой части этого поля. Данные кнопки позволяют в пошаговом режиме уменьшить или увеличить значение, представленное в поле редактирования, до требуемого количества разрядов. Допустимый диапазон изменения параметра *Width* для дополнительной входной шины данных C составляет от одного до 48 двоичных разрядов.

Если необходимо подготовить описание элемента, выполняющего операции умножения с накоплением с использованием значений, представленных на дополнительной входной шине данных C арифметическо-логического блока аппаратной секции DSP48E, то во встроенной панели *OPMODE Control* нужно нажать кнопку *Add C input*. При этом диалоговая панель с заголовком *Operation Mode Setup – MACC* принимает вид, показанный на рис. 36. В этой диалоговой панели следует определить разрядность дополнительной входной шины данных C таким же образом, как и в предыдущем варианте.

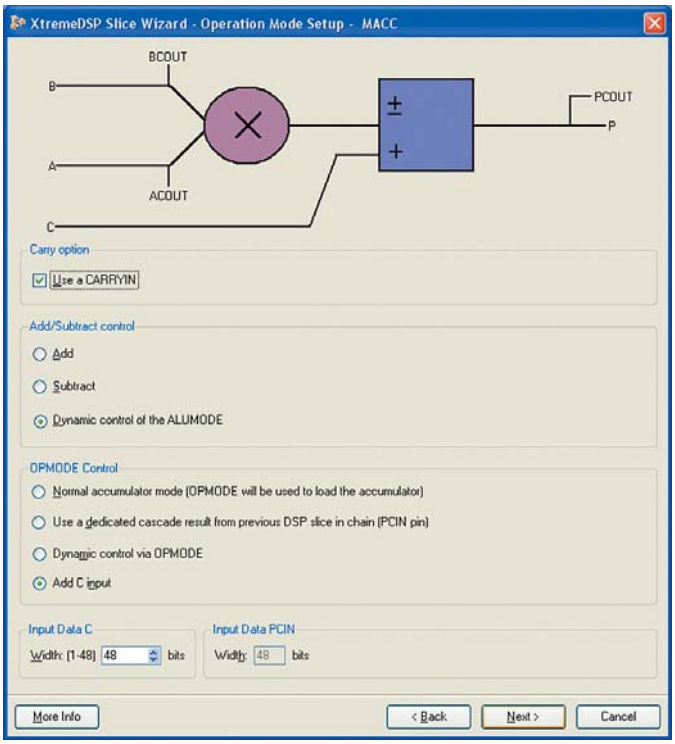


**Рис. 36.** Вид диалоговой панели Operation Mode Setup — MACC «мастера» настройки параметров элементов при выборе варианта конфигурации с использованием дополнительной входной шины данных C

После выбора требуемого варианта конфигурации элемента, предназначенного для выполнения операций умножения с накоплением, с расширенным набором функциональных возможностей необходимо перейти к третьей диалоговой панели «мастера» настройки, которая имеет заголовок *Pipelining and CE/RST Setup – MACC*. Вид этой диалоговой панели для рассматриваемого типа формируемых компонентов при различных вариантах организации конвейерной обработки входных и выходных данных приведен на рис. 31–33 (см. КиТ № 6 `2009, стр. 52, 53). В рассматриваемой диалоговой панели нужно указать количество конвейерных регистров, устанавливаемых на входных информационных шинах данных, определить необходимость использования буферных регистров на шинах управления и в цепи внешнего сигнала входного переноса, а также выбрать входы управления для каждого из этих регистров. Эта процедура выполняется так же, как и при формировании описаний элементов, осуществляющих операции умножения с накоплением, с традиционной структурой. Следует обратить внимание на то, что буферные регистры на входах шины OPMODE, используемой для управления выбором источников входных данных и выполняемой функции арифметическо-логического блока аппаратной секции DSP48E, могут включаться в состав структуры генерируемого компонента только при выборе варианта с динамически изменяемой конфигурацией (при нажатой кнопке *Dynamic control via OPMODE*).

Для проверки установленных параметров конфигурации формируемого элемента, выполняющего операции умножения с накоплением (вычитанием) с поддержкой расширенного набора функциональных возможностей, и запуска процесса автоматической генерации его описания следует открыть заключительную информационную панель «мастера» настройки параметров с заголовком *Summary – MACC*. Процесс автоматического формирования файлов описания компонента с выбранным вариантом конфигурации активизируется так же, как и при создании элементов, выполняющих операции умножения с накоплением, с традиционной структурой.

В последующих разделах приводятся примеры описаний элементов, осуществляющих операции умножения с накоплением (вычитанием), для каждого варианта конфигурации, рассмотренного выше.

## Пример описания элемента, выполняющего операции умножения с накоплением и вычитанием с учетом результатов вычислений предшествующего модуля ЦОС, сформированного с помощью «мастера» Architecture Wizard для реализации на базе аппаратных модулей DSP48E в ПЛИС серии Virtex-5

В качестве примера описания компонента, выполняющего операции умножения с накоплением и вычитанием, который предназначен для использования в составе многозвенных устройств ЦОС, реализуемых на основе аппаратных секций DSP48E в ПЛИС семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT, далее приводится VHDL-описание элемента *macc_add_sub_pc_14_12_32*, сгенерированное с помощью «мастера» Architecture Wizard. Этот элемент осуществляет операции перемножения значений, представленных в виде 14- и 12-разрядного двоичного кода на соответствующих информационных входных шинах данных, с последующим вычислением суммы или разности произведения и содержимого аккумулятора с учетом значений результатов выполнения операций в предшествующем модуле ЦОС и внешнего сигнала входного переноса. Итоговый результат вычислений, выполняемых элементом *macc_add_sub_pc_14_12_32*, поступает на выходную шину данных в форме 32-разрядного двоичного кода. В качестве входных портов, определяющих значения сомножителей, в этом элементе используются шины данных A и B аппаратной секции DSP48E. Для выбора типа операции, осуществляемой аккумулятором рассматриваемого элемента, в составе его интерфейса предусмотрена 4-разрядная шина управления ALUMODE_IN.

Сформированный текст описания элемента *macc_add_sub_pc_14_12_32* на языке VHDL имеет следующий вид:

```
--Command: xaw2vhdl-st D:\PRJ\macc_add_sub_pc_14_12_32.xaw
D:\PRJ\macc_add_sub_pc_14_12_32
--Design Name: macc_add_sub_pc_14_12_32
--Device: xc5vsx50t-ff1136-3
--
-- Module macc_add_sub_pc_14_12_32
-- Generated by Xilinx Architecture Wizard
-- Written for synthesis tool: XST
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
--
entity macc_add_sub_pc_14_12_32 is
  port (
      ALUMODE_IN          : in std_logic_vector (3 downto 0);
      A_IN                : in std_logic_vector (11 downto 0);
      B_IN                : in std_logic_vector (13 downto 0);
      CARRYIN_IN          : in std_logic;
      CEALUMODE_IN        : in std_logic;
      CEA1_IN             : in std_logic;
      CEA2_IN             : in std_logic;
      CEB1_IN             : in std_logic;
      CEB2_IN             : in std_logic;
      CECARRYIN_IN        : in std_logic;
      CEMULTCARRYIN_IN    : in std_logic;
      CEM_IN              : in std_logic;
      CEP_IN              : in std_logic;
      CLK_IN              : in std_logic;
      PCIN_IN             : in std_logic_vector (47 downto 0);
      RSTALLCARRYIN_IN    : in std_logic;
      RSTALUMODE_IN       : in std_logic;
      RSTA_IN             : in std_logic;
      RSTB_IN             : in std_logic;
      RSTCTRL_IN          : in std_logic;
      RSTM_IN             : in std_logic;
      RSTP_IN             : in std_logic;
      ACOUT_OUT           : out std_logic_vector (29 downto 0);
      BCOUT_OUT           : out std_logic_vector (17 downto 0);
      PCOUT_OUT           : out std_logic_vector (47 downto 0);
      P_OUT               : out std_logic_vector (31 downto 0)
      );
end macc_add_sub_pc_14_12_32;
--
architecture BEHAVIORAL of macc_add_sub_pc_14_12_32 is
  signal GND_BUS_3        : std_logic_vector (2 downto 0);
  signal GND_BUS_18       : std_logic_vector (17 downto 0);
```

```
    signal GND_BUS_30           : std_logic_vector (29 downto 0);
    signal GND_BUS_48           : std_logic_vector (47 downto 0);
    signal GND_OPMODE           : std_logic;
    signal P_float              : std_logic_vector (15 downto 0);
    signal VCC_OPMODE           : std_logic;
begin
    GND_BUS_3(2 downto 0) <= «000»;
    GND_BUS_18(17 downto 0) <= «000000000000000000»;
    GND_BUS_30(29 downto 0) <= «000000000000000000000000000000»;
    GND_BUS_48(47 downto 0) <=
          «000000000000000000000000000000000000000000000000»;
    GND_OPMODE <= '0';
    VCC_OPMODE <= '1';
    DSP48E_INST : DSP48E
    generic map(
            ACASCREG => 1,
            ALUMODEREG => 1,
            AREG => 2,
            AUTORESET_PATTERN_DETECT => FALSE,
            AUTORESET_PATTERN_DETECT_OPTINV => «MATCH»,
            A_INPUT => «DIRECT»,
            BCASCREG => 1,
            BREG => 2,
            B_INPUT => «DIRECT»,
            CARRYINREG => 1,
            CARRYINSELREG => 0,
            CREG => 0,
            MASK => x«3FFFFFFFFFFF»,
            MREG => 1,
            MULTCARRYINREG => 1,
            OPMODEREG => 0,
            PATTERN => x«000000000000»,
            PREG => 1,
            SEL_MASK => «MASK»,
            SEL_PATTERN => «PATTERN»,
            SEL_ROUNDING_MASK => «SEL_MASK»,
            USE_MULT => «MULT_S»,
            USE_PATTERN_DETECT => «NO_PATDET»,
            USE_SIMD => «ONE48»
            )
        port map (
            A(29)=>A_IN(11),
            A(28)=>A_IN(11),
            A(27)=>A_IN(11),
            A(26)=>A_IN(11),
            A(25)=>A_IN(11),
            A(24)=>A_IN(11),
            A(23)=>A_IN(11),
            A(22)=>A_IN(11),
            A(21)=>A_IN(11),
            A(20)=>A_IN(11),
            A(19)=>A_IN(11),
            A(18)=>A_IN(11),
            A(17)=>A_IN(11),
            A(16)=>A_IN(11),
            A(15)=>A_IN(11),
            A(14)=>A_IN(11),
            A(13)=>A_IN(11),
            A(12)=>A_IN(11),
            A(11 downto 0)=>A_IN(11 downto 0),
            ACIN(29 downto 0)=>GND_BUS_30(29 downto 0),
            ALUMODE(3 downto 0)=>ALUMODE_IN(3 downto 0),
            B(17)=>B_IN(13),
            B(16)=>B_IN(13),
            B(15)=>B_IN(13),
            B(14)=>B_IN(13),
            B(13 downto 0)=>B_IN(13 downto 0),
            BCIN(17 downto 0)=>GND_BUS_18(17 downto 0),
            C(47 downto 0)=>GND_BUS_48(47 downto 0),
            CARRYCASCIN=>GND_OPMODE,
            CARRYIN=>CARRYIN_IN,
            CARRYINSEL(2 downto 0)=>GND_BUS_3(2 downto 0),
            CEALUMODE=>CEALUMODE_IN,
            CEA1=>CEA1_IN,
            CEA2=>CEA2_IN,
            CEB1=>CEB1_IN,
            CEB2=>CEB2_IN,
            CEC=>VCC_OPMODE,
            CECARRYIN=>CECARRYIN_IN,
            CECTRL=>VCC_OPMODE,
            CEM=>CEM_IN,
            CEMULTCARRYIN=>CEMULTCARRYIN_IN,
            CEP=>CEP_IN,
            CLK=>CLK_IN,
            MULTSIGNIN=>GND_OPMODE,
            OPMODE(6)=>GND_OPMODE,
            OPMODE(5)=>GND_OPMODE,
            OPMODE(4)=>VCC_OPMODE,
            OPMODE(3)=>GND_OPMODE,
            OPMODE(2)=>VCC_OPMODE,
            OPMODE(1)=>GND_OPMODE,
            OPMODE(0)=>VCC_OPMODE,
            PCIN(47 downto 0)=>PCIN_IN(47 downto 0),
            RSTA=>RSTA_IN,
            RSTALLCARRYIN=>RSTALLCARRYIN_IN,
            RSTALUMODE=>RSTALUMODE_IN,
            RSTB=>RSTB_IN,
            RSTC=>GND_OPMODE,
            RSTCTRL=>RSTCTRL_IN,
            RSTM=>RSTM_IN,
            RSTP=>RSTP_IN,
            ACOUT(29 downto 0)=>ACOUT_OUT(29 downto 0),
            BCOUT(17 downto 0)=>BCOUT_OUT(17 downto 0),
            CARRYCASCOUT=>open,
            CARRYOUT=>open,
            MULTSIGNOUT=>open,
            OVERFLOW=>open,
            P(47 downto 32)=>P_float(15 downto 0),
            P(31 downto 0)=>P_OUT(31 downto 0),
            PATTERNBDETECT=>open,
            PATTERNDETECT=>open,
            PCOUT(47 downto 0)=>PCOUT_OUT(47 downto 0),
            UNDERFLOW=>open
            );
end BEHAVIORAL;
```

В структуре элемента *macc_add_sub_pc_14_12_32* применяются все входные, выходные и буферные регистры, представленные в составе архитектуры аппаратных секций DSP48E ПЛИС семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT. Для каждого из этих регистров предусмотрены индивидуальные входы сигналов сброса и разрешения синхронизации, которые включены в состав интерфейса сгенерированного компонента.

В приведенном VHDL-описании элемента *macc_add_sub_pc_14_12_32* используется следующая система условных обозначений интерфейсных портов:

- A_IN [11:0] — 12-разрядная входная шина данных, совокупность сигналов которой определяет значение первого сомножителя;
- B_IN[13:0] — 14-разрядная входная шина данных, совокупность сигналов которой определяет значение второго сомножителя;
- ALUMODE_IN[3:0] — 4-разрядная входная шина управления выбором типа операции, осуществляемой в аккумуляторе сформированного элемента;
- CARRYIN_IN — вход внешнего сигнала переноса;
- CEALUMODE_IN — вход сигнала разрешения синхронизации для буферного регистра, установленного на шине управления ALUMODE_IN;
- CEA1_IN, CEA2_IN, CEB1_IN и CEB2_IN — входы сигналов разрешения синхронизации для буферных регистров, задействованных на входных информационных шинах данных A_IN и B_IN соответственно;
- CECARRYIN_IN — вход сигнала разрешения синхронизации для буферного регистра, установленного в цепи внешнего сигнала входного переноса;
- CEMULTCARRYIN_IN — вход сигнала разрешения синхронизации для буферного регистра, задействованного в цепи внутреннего сигнала входного переноса;
- CEM_IN — вход сигнала разрешения синхронизации для буферного регистра, установленного на выходе умножителя (на входе арифметическо-логического блока используемой аппаратной секции DSP48E) сгенерированного элемента;
- CEP_IN — вход сигнала разрешения синхронизации для выходного регистра сформированного элемента;
- CLK_IN — вход сигнала синхронизации;
- PCIN_IN[47:0] — 48-разрядная входная шина данных, на которую поступает значение результата вычислений, осуществляемых предшествующим модулем ЦОС;
- RSTALLCARRYIN_IN — вход сигнала сброса буферных регистров, задействованных в цепях входного переноса;
- RSTALUMODE_IN — вход сигнала сброса для буферного регистра, установленного на шине управления ALUMODE_IN;
- RSTA_IN и RSTB_IN — входы сигналов сброса буферных регистров, установленных на входных информационных шинах данных A_IN и B_IN соответственно;
- RSTCTRL_IN — вход сигнала сброса буферного регистра, установленного на шине управления выбором источников входных данных для арифметическо-логического блока используемой аппаратной секции DSP48E;
- RSTM_IN — вход сигнала сброса буферного регистра, задействованного на выходе умножителя (на входе арифметическо-логического блока) сгенерированного элемента;
- RSTP_IN — вход сигнала сброса для выходного регистра сформированного элемента;
- ACOUT_OUT[29:0] — 30-разрядная выходная шина данных, используемая для подключения ко входной шине ACIN_IN следующего модуля ЦОС при каскадном соединении;
- BCOUT_OUT[17:0] — 18-разрядная выходная шина данных, предназначенная для подключения ко входной шине данных BCIN_IN следующего модуля ЦОС при каскадном соединении;
- PCOUT_OUT[47:0] — 48-разрядная выходная шина данных, применяемая при каскадном соединении модулей ЦОС;
- P_OUT[31:0] — 32-разрядная выходная шина данных, на которой отображается значение итогового результата выполняемых операций.

Для определения требуемых значений основных атрибутов библиотечного примитива DSP48E, образующего основу сформированного VHDL-описания элемента *macc_add_sub_pc_14_12_32*, нужно включить в состав файла временных и топологических ограничений проекта разрабатываемого устройства ЦОС следующую совокупность выражений, которые содержатся в дополнительном файле с расширением ucf, сгенерированном «мастером» Architecture Wizard:

```
# Generated by Xilinx Architecture Wizard
# --- UCF Template Only ---
# Cut and paste these attributes into the project's UCF file, if desired
INST DSP48E_INST ACASCREG = 1;
INST DSP48E_INST ALUMODEREG = 1;
INST DSP48E_INST AREG = 2;
INST DSP48E_INST AUTORESET_PATTERN_DETECT = FALSE;
INST DSP48E_INST AUTORESET_PATTERN_DETECT_OPTINV =
MATCH;
INST DSP48E_INST A_INPUT = DIRECT;
INST DSP48E_INST BCASCREG = 1;
```

```
INST DSP48E_INST BREG = 2;
INST DSP48E_INST B_INPUT = DIRECT;
INST DSP48E_INST CARRYINREG = 1;
INST DSP48E_INST CARRYINSELREG = 0;
INST DSP48E_INST CREG = 0;
INST DSP48E_INST MASK = 3FFFFFFFFFFF;
INST DSP48E_INST MREG = 1;
INST DSP48E_INST MULTCARRYINREG = 1;
INST DSP48E_INST OPMODEREG = 0;
INST DSP48E_INST PATTERN = 000000000000;
INST DSP48E_INST PREG = 1;
INST DSP48E_INST SEL_MASK = MASK;
INST DSP48E_INST SEL_PATTERN = PATTERN;
INST DSP48E_INST SEL_ROUNDING_MASK = SEL_MASK;
INST DSP48E_INST USE_MULT = MULT_S;
INST DSP48E_INST USE_PATTERN_DETECT = NO_PATDET;
INST DSP48E_INST USE_SIMD = ONE48;
```

## Пример описания многофункционального элемента, осуществляющего операции умножения с накоплением и вычитанием с дополнительными возможностями, сформированного с помощью «мастера» Architecture Wizard для реализации на базе аппаратных модулей DSP48E в ПЛИС серии Virtex-5

Примером многофункционального компонента, включающего в себя аппаратный умножитель и аккумулятор с возможностью динамического выбора типа выполняемой операции и источников входных данных, предназначенного для реализации на базе аппаратных модулей DSP48E кристаллов семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT, является описание элемента *macc_add_sub_dc_18_20_35*. Этот элемент выполняет функции вычисления произведения 18- и 20-разрядных значений входных данных с последующим вычитанием или суммированием с содержимым аккумулятора, в зависимости от кода выбора типа операции, представленного на соответствующей шине управления. Основными входными информационными портами в элементе *macc_add_sub_dc_18_20_35* являются порты ACIN_IN и BCIN_IN, сопряженные с шинами данных ACIN и BCIN аппаратной секции DSP48E, которая используется для реализации сформированного компонента. Совокупности логических уровней сигналов этих шин определяют значения сомножителей в рассматриваемом элементе.

Операции суммирования и вычитания в сгенерированном элементе *macc_add_sub_dc_18_20_35* могут осуществляться с учетом значений внешнего сигнала входного переноса и данных, поступающих из дополнительных входных информационных портов. В качестве таких портов в рассматриваемом элементе используются дополнительная 22-разрядная входная шина данных C_IN, подключенная к соответствующим младшим разрядам шины C арифметическо-логического блока используемой аппаратной секции DSP48E, и 48-разрядная шина PCIN_IN, предназначенная для подключения выходной шины PCOUT предыдущей секции при каскадном наращивании аппаратных модулей ЦОС. Выбор источников дополнительных данных (соответствующих входных шин) производится подачей соответствующего двоичного кода на 7-разрядную шину управления OPMODE_IN.

Управление типом операции, выполняемой в аккумуляторе рассматриваемого элемента, осуществляется с помощью 4-разрядной шины ALUMODE_IN. Значение результата вычислений, производимых элементом *macc_add_sub_dc_18_20_35*, отображается на выходной шине данных в виде 35-разрядного двоичного кода.

Текст VHDL-описания элемента *macc_add_sub_dc_18_20_35*, сгенерированный с помощью «мастера» Architecture Wizard, выглядит следующим образом:

```
--Command: xaw2vhdl-st D:\PRJ\macc_add_sub_dc_18_20_35.xaw
D:\PRJ\macc_add_sub_dc_18_20_35
--Design Name: macc_add_sub_dc_18_20_35
--Device: xc5vfx70t-ff1136-3
--
-- Module macc_add_sub_dc_18_20_35
-- Generated by Xilinx Architecture Wizard
-- Written for synthesis tool: XST
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
--
entity macc_add_sub_dc_18_20_35 is
   port (
      ACIN_IN              : in std_logic_vector (19 downto 0);
      ALUMODE_IN           : in std_logic_vector (3 downto 0);
      BCIN_IN              : in std_logic_vector (17 downto 0);
      CARRYINSEL_IN        : in std_logic_vector (2 downto 0);
      CARRYIN_IN           : in std_logic;
      CEALUMODE_IN         : in std_logic;
      CEA1_IN              : in std_logic;
      CEA2_IN              : in std_logic;
      CEB1_IN              : in std_logic;
      CEB2_IN              : in std_logic;
      CECARRYIN_IN         : in std_logic;
      CECTRL_IN            : in std_logic;
      CEC_IN               : in std_logic;
      CEMULTCARRYIN_IN     : in std_logic;
      CEM_IN               : in std_logic;
      CEP_IN               : in std_logic;
      CLK_IN               : in std_logic;
      C_IN                 : in std_logic_vector (21 downto 0);
      OPMODE_IN            : in std_logic_vector (6 downto 0);
      PCIN_IN              : in std_logic_vector (47 downto 0);
      RSTALLCARRYIN_IN     : in std_logic;
      RSTALUMODE_IN        : in std_logic;
      RSTA_IN              : in std_logic;
      RSTB_IN              : in std_logic;
      RSTCTRL_IN           : in std_logic;
      RSTC_IN              : in std_logic;
      RSTM_IN              : in std_logic;
      RSTP_IN              : in std_logic;
      ACOUT_OUT            : out std_logic_vector (29 downto 0);
      BCOUT_OUT            : out std_logic_vector (17 downto 0);
      PCOUT_OUT            : out std_logic_vector (47 downto 0);
      P_OUT                : out std_logic_vector (34 downto 0)
      );
end macc_add_sub_dc_18_20_35;
--
architecture BEHAVIORAL of macc_add_sub_dc_18_20_35 is
   signal GND_BIT          : std_logic;
   signal GND_BUS_18       : std_logic_vector (17 downto 0);
   signal GND_BUS_30       : std_logic_vector (29 downto 0);
   signal P_float          : std_logic_vector (12 downto 0);
begin
   GND_BIT <= '0';
   GND_BUS_18(17 downto 0) <= «000000000000000000»;
   GND_BUS_30(29 downto 0) <= «000000000000000000000000000000»;
   DSP48E_INST : DSP48E
   generic map(
            ACASCREG => 1,
            ALUMODEREG => 1,
            AREG => 2,
            AUTORESET_PATTERN_DETECT => FALSE,
            AUTORESET_PATTERN_DETECT_OPTINV => «MATCH»,
            A_INPUT => «CASCADE»,
            BCASCREG => 1,
            BREG => 2,
            B_INPUT => «CASCADE»,
            CARRYINREG => 1,
            CARRYINSELREG => 0,
            CREG => 1,
            MASK => x«3FFFFFFFFFFF»,
            MREG => 1,
            MULTCARRYINREG => 1,
            OPMODEREG => 1,
            PATTERN => x«000000000000»,
            PREG => 1,
            SEL_MASK => «MASK»,
            SEL_PATTERN => «PATTERN»,
            SEL_ROUNDING_MASK => «SEL_MASK»,
            USE_MULT => «MULT_S»,
            USE_PATTERN_DETECT => «NO_PATDET»,
            USE_SIMD => «ONE48»
            )
      port map (
            A(29 downto 0)=>GND_BUS_30(29 downto 0),
            ACIN(29)=>ACIN_IN(19),
            ACIN(28)=>ACIN_IN(19),
            ACIN(27)=>ACIN_IN(19),
            ACIN(26)=>ACIN_IN(19),
            ACIN(25)=>ACIN_IN(19),
            ACIN(24)=>ACIN_IN(19),
            ACIN(23)=>ACIN_IN(19),
            ACIN(22)=>ACIN_IN(19),
            ACIN(21)=>ACIN_IN(19),
            ACIN(20)=>ACIN_IN(19),
            ACIN(19 downto 0)=>ACIN_IN(19 downto 0),
            ALUMODE(3 downto 0)=>ALUMODE_IN(3 downto 0),
            B(17 downto 0)=>GND_BUS_18(17 downto 0),
            BCIN(17 downto 0)=>BCIN_IN(17 downto 0),
            C(47)=>C_IN(21),
            C(46)=>C_IN(21),
            C(45)=>C_IN(21),
            C(44)=>C_IN(21),
            C(43)=>C_IN(21),
            C(42)=>C_IN(21),
            C(41)=>C_IN(21),
            C(40)=>C_IN(21),
            C(39)=>C_IN(21),
            C(38)=>C_IN(21),
            C(37)=>C_IN(21),
            C(36)=>C_IN(21),
            C(35)=>C_IN(21),
            C(34)=>C_IN(21),
            C(33)=>C_IN(21),
            C(32)=>C_IN(21),
            C(31)=>C_IN(21),
            C(30)=>C_IN(21),
            C(29)=>C_IN(21),
            C(28)=>C_IN(21),
            C(27)=>C_IN(21),
            C(26)=>C_IN(21),
            C(25)=>C_IN(21),
            C(24)=>C_IN(21),
            C(23)=>C_IN(21),
            C(22)=>C_IN(21),
            C(21 downto 0)=>C_IN(21 downto 0),
            CARRYCASCIN=>GND_BIT,
            CARRYIN=>CARRYIN_IN,
            CARRYINSEL(2 downto 0)=>CARRYINSEL_IN(2 downto 0),
            CEALUMODE=>CEALUMODE_IN,
            CEA1=>CEA1_IN,
            CEA2=>CEA2_IN,
            CEB1=>CEB1_IN,
            CEB2=>CEB2_IN,
            CEC=>CEC_IN,
            CECARRYIN=>CECARRYIN_IN,
            CECTRL=>CECTRL_IN,
            CEM=>CEM_IN,
            CEMULTCARRYIN=>CEMULTCARRYIN_IN,
            CEP=>CEP_IN,
            CLK=>CLK_IN,
            MULTSIGNIN=>GND_BIT,
            OPMODE(6 downto 0)=>OPMODE_IN(6 downto 0),
            PCIN(47 downto 0)=>PCIN_IN(47 downto 0),
            RSTA=>RSTA_IN,
            RSTALLCARRYIN=>RSTALLCARRYIN_IN,
            RSTALUMODE=>RSTALUMODE_IN,
            RSTB=>RSTB_IN,
            RSTC=>RSTC_IN,
            RSTCTRL=>RSTCTRL_IN,
            RSTM=>RSTM_IN,
            RSTP=>RSTP_IN,
            ACOUT(29 downto 0)=>ACOUT_OUT(29 downto 0),
            BCOUT(17 downto 0)=>BCOUT_OUT(17 downto 0),
            CARRYCASCOUT=>open,
            CARRYOUT=>open,
            MULTSIGNOUT=>open,
            OVERFLOW=>open,
            P(47 downto 35)=>P_float(12 downto 0),
            P(34 downto 0)=>P_OUT(34 downto 0),
            PATTERNBDETECT=>open,
            PATTERNDETECT=>open,
            PCOUT(47 downto 0)=>PCOUT_OUT(47 downto 0),
            UNDERFLOW=>open
            );
end BEHAVIORAL;
```

Архитектура элемента *macc_add_sub_dc_18_20_35* выполнена на основе структуры с двухступенчатой организацией конвейерной обработки входных и выходных данных. При этом все задействованные входные, выходные и буферные регистры имеют раздельные входы сигналов разрешения синхронизации и сброса, которые включены в состав интерфейса сгенерированного элемента.

Основу системы условных обозначений интерфейсных портов, используемых в VHDL-описании элемента *macc_add_sub_dc_18_20_35*, образует совокупность идентификаторов, применяемых в описании интерфейса компонента *macc_add_sub_pc_14_12_32*, которая была приведена в предыдущем разделе. Кроме того, в составе интерфейса элемента *macc_add_sub_dc_18_20_35* представлены следующие входные порты:

- BCIN_IN[17:0] — 18-разрядная входная шина данных, совокупность сигналов которой определяет значение первого сомножителя.
- ACIN_IN [19:0] — 20-разрядная входная шина данных, совокупность сигналов которой определяет значение второго сомножителя;
- CARRYINSEL_IN[2:0] — входная 3-разрядная шина управления выбором источника сигнала переноса;
- CECTRL_IN — вход сигнала разрешения синхронизации буферного регистра, установленного на шине управления выбором источников входных данных для арифметическо-логического блока аппаратной секции DSP48E, используемой для реализации сформированного элемента;
- CEC_IN — вход сигнала разрешения синхронизации для буферного регистра, подключаемого к дополнительной информационной шине данных C_IN;
- C_IN[21:0] — 22-разрядная входная шина данных, совокупность сигналов которой определяет значение дополнительного операнда;
- RSTC_IN — вход сигнала сброса буферного регистра, подключаемого к дополнительной шине данных C_IN;
- OPMODE_IN[6:0] — 7-разрядная шина управления выбором источников входных данных и выполняемой функции арифметическо-логического блока аппаратной секции DSP48E, применяемой для реализации сгенерированного элемента.

При использовании сформированного VHDL-кода элемента *macc_add_sub_dc_18_20_35* в составе описания разрабатываемого устройства ЦОС нужно добавить в файл временных и топологических ограничений проекта САПР серии Xilinx ISE приведенную далее совокупность выражений:

```
# Generated by Xilinx Architecture Wizard
# --- UCF Template Only ---
# Cut and paste these attributes into the project's UCF file, if desired
INST DSP48E_INST ACASCREG = 1;
INST DSP48E_INST ALUMODEREG = 1;
INST DSP48E_INST AREG = 2;
INST DSP48E_INST AUTORESET_PATTERN_DETECT = FALSE;
INST DSP48E_INST AUTORESET_PATTERN_DETECT_OPTINV = MATCH;
INST DSP48E_INST A_INPUT = CASCADE;
INST DSP48E_INST BCASCREG = 1;
INST DSP48E_INST BREG = 2;
INST DSP48E_INST B_INPUT = CASCADE;
INST DSP48E_INST CARRYINREG = 1;
INST DSP48E_INST CARRYINSELREG = 0;
INST DSP48E_INST CREG = 1;
INST DSP48E_INST MASK = 3FFFFFFFFFFF;
INST DSP48E_INST MREG = 1;
INST DSP48E_INST MULTCARRYINREG = 1;
INST DSP48E_INST OPMODEREG = 1;
INST DSP48E_INST PATTERN = 000000000000;
INST DSP48E_INST PREG = 1;
INST DSP48E_INST SEL_MASK = MASK;
INST DSP48E_INST SEL_PATTERN = PATTERN;
INST DSP48E_INST SEL_ROUNDING_MASK = SEL_MASK;
INST DSP48E_INST USE_MULT = MULT_S;
INST DSP48E_INST USE_PATTERN_DETECT = NO_PATDET;
INST DSP48E_INST USE_SIMD = ONE48;
```

## Пример описания элемента, предназначенного для выполнения операций умножения с накоплением с участием дополнительных операндов, подготовленного с помощью «мастера» Architecture Wizard для реализации на базе аппаратных модулей DSP48E в ПЛИС серии Virtex-5

Формирование описаний высокоскоростных элементов, осуществляющих операции умножения с накоплением с участием дополнительных операндов, с помощью «мастера» Architecture Wizard для последующей реализации на основе аппаратных секций DSP48E в ПЛИС семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT показано в настоящем разделе на примере элемента *macc_add_const_c_18_14_38*. Сгенерированный элемент выполняет умножение входных отсчетов на постоянный коэффициент и последующее накопление произведений с суммированием значений, поступающих на дополнительную входную шину данных. Вычисление итогового результата в элементе *macc_add_const_c_18_14_38* производится с учетом значения внешнего сигнала входного переноса. Значения перемножаемых отсчетов поступают в виде 18-разрядного двоичного кода на входную шину BCIN аппаратной секции DSP48E, используемой для реализации рассматриваемого элемента. Постоянный коэффициент в сформированном элементе представлен в 14-разрядном двоичном формате. В качестве источника дополнительного операнда в компоненте *macc_add_const_c_18_14_38* используется входная 24-разрядная шина данных C_IN, подключенная непосредственно к соответствующим младшим разрядам шины C арифметическо-логического блока применяемого аппаратного модуля DSP48E. Итоговый результат выполняемых операций отображается на выходной шине данных сгенерированного элемента в форме 38-разрядного двоичного значения.

Сформированный VHDL-код описания элемента *macc_add_const_c_18_14_38* имеет следующий вид:

```
--Command: xaw2vhdl-st D:\PRJ\macc_add_const_c_18_14_38.xaw
D:\PRJ\macc_add_const_c_18_14_38
--Design Name: macc_add_const_c_18_14_38
--Device: xc5vsx35t-ff665-3
--
-- Module macc_add_const_c_18_14_38
-- Generated by Xilinx Architecture Wizard
-- Written for synthesis tool: XST
--
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
library UNISIM;
use UNISIM.Vcomponents.ALL;
--
entity macc_add_const_c_18_14_38 is
    port (
        BCIN_IN                 : in std_logic_vector (17 downto 0);
        CARRYIN_IN              : in std_logic;
        CEA1_IN                 : in std_logic;
        CEA2_IN                 : in std_logic;
        CEB1_IN                 : in std_logic;
        CEB2_IN                 : in std_logic;
        CECARRYIN_IN            : in std_logic;
        CEC_IN                  : in std_logic;
        CEMULTCARRYIN_IN        : in std_logic;
        CEM_IN                  : in std_logic;
        CEP_IN                  : in std_logic;
        CLK_IN                  : in std_logic;
        C_IN                    : in std_logic_vector (23 downto 0);
        RSTALLCARRYIN_IN        : in std_logic;
        RSTA_IN                 : in std_logic;
        RSTB_IN                 : in std_logic;
        RSTC_IN                 : in std_logic;
        RSTM_IN                 : in std_logic;
        RSTP_IN                 : in std_logic;
        ACOUT_OUT               : out std_logic_vector (29 downto 0);
        BCOUT_OUT               : out std_logic_vector (17 downto 0);
        PCOUT_OUT               : out std_logic_vector (47 downto 0);
        P_OUT                   : out std_logic_vector (37 downto 0)
        );
end macc_add_const_c_18_14_38;
--
architecture BEHAVIORAL of macc_add_const_c_18_14_38 is
    signal GND_A                : std_logic;
    signal GND_ALUMODE          : std_logic;
    signal GND_BUS_3            : std_logic_vector (2 downto 0);
    signal GND_BUS_18           : std_logic_vector (17 downto 0);
    signal GND_BUS_30           : std_logic_vector (29 downto 0);
    signal GND_BUS_48           : std_logic_vector (47 downto 0);
    signal GND_OPMODE           : std_logic;
    signal P_float              : std_logic_vector (9 downto 0);
    signal VCC_A                : std_logic;
    signal VCC_OPMODE           : std_logic;
begin
    GND_A <= '0';
    GND_ALUMODE <= '0';
    GND_BUS_3(2 downto 0) <= «000»;
    GND_BUS_18(17 downto 0) <= «000000000000000000»;
    GND_BUS_30(29 downto 0) <= «000000000000000000000000000000»;
    GND_BUS_48(47 downto 0) <=
      «000000000000000000000000000000000000000000000000»;
    GND_OPMODE <= '0';
    VCC_A <= '1';
    VCC_OPMODE <= '1';
    DSP48E_INST : DSP48E
    generic map(
            ACASCREG => 1,
            ALUMODEREG => 0,
            AREG => 1,
            AUTORESET_PATTERN_DETECT => FALSE,
            AUTORESET_PATTERN_DETECT_OPTINV => «MATCH»,
            A_INPUT => «DIRECT»,
            BCASCREG => 1,
            BREG => 1,
            B_INPUT => «CASCADE»,
            CARRYINREG => 1,
            CARRYINSELREG => 0,
            CREG => 1,
            MASK => x«3FFFFFFFFFFF»,
            MREG => 1,
            MULTCARRYINREG => 1,
            OPMODEREG => 0,
            PATTERN => x«000000000000»,
            PREG => 1,
            SEL_MASK => «MASK»,
            SEL_PATTERN => «PATTERN»,
            SEL_ROUNDING_MASK => «SEL_MASK»,
            USE_MULT => «MULT_S»,
            USE_PATTERN_DETECT => «NO_PATDET»,
            USE_SIMD => «ONE48»
            )
    port map (
            A(29)=>GND_A,
            A(28)=>GND_A,
            A(27)=>GND_A,
            A(26)=>GND_A,
            A(25)=>GND_A,
            A(24)=>GND_A,
            A(23)=>GND_A,
```

```
        A(22)=>GND_A,
        A(21)=>GND_A,
        A(20)=>GND_A,
        A(19)=>GND_A,
        A(18)=>GND_A,
        A(17)=>GND_A,
        A(16)=>GND_A,
        A(15)=>GND_A,
        A(14)=>GND_A,
        A(13)=>GND_A,
        A(12)=>GND_A,
        A(11)=>GND_A,
        A(10)=>GND_A,
        A(9)=>GND_A,
        A(8)=>GND_A,
        A(7)=>GND_A,
        A(6)=>GND_A,
        A(5)=>GND_A,
        A(4)=>VCC_A,
        A(3)=>VCC_A,
        A(2)=>VCC_A,
        A(1)=>GND_A,
        A(0)=>GND_A,
        ACIN(29 downto 0)=>GND_BUS_30(29 downto 0),
        ALUMODE(3)=>GND_ALUMODE,
        ALUMODE(2)=>GND_ALUMODE,
        ALUMODE(1)=>GND_ALUMODE,
        ALUMODE(0)=>GND_ALUMODE,
        B(17 downto 0)=>GND_BUS_18(17 downto 0),
        BCIN(17 downto 0)=>BCIN_IN(17 downto 0),
        C(47)=>C_IN(23),
        C(46)=>C_IN(23),
        C(45)=>C_IN(23),
        C(44)=>C_IN(23),
        C(43)=>C_IN(23),
        C(42)=>C_IN(23),
        C(41)=>C_IN(23),
        C(40)=>C_IN(23),
        C(39)=>C_IN(23),
        C(38)=>C_IN(23),
        C(37)=>C_IN(23),
        C(36)=>C_IN(23),
        C(35)=>C_IN(23),
        C(34)=>C_IN(23),
        C(33)=>C_IN(23),
        C(32)=>C_IN(23),
        C(31)=>C_IN(23),
        C(30)=>C_IN(23),
        C(29)=>C_IN(23),
        C(28)=>C_IN(23),
        C(27)=>C_IN(23),
        C(26)=>C_IN(23),
        C(25)=>C_IN(23),
        C(24)=>C_IN(23),
        C(23 downto 0)=>C_IN(23 downto 0),
        CARRYCASCIN=>GND_A,
        CARRYIN=>CARRYIN_IN,
        CARRYINSEL(2 downto 0)=>GND_BUS_3(2 downto 0),
        CEALUMODE=>VCC_OPMODE,
        CEA1=>CEA1_IN,
        CEA2=>CEA2_IN,
        CEB1=>CEB1_IN,
        CEB2=>CEB2_IN,
        CEC=>CEC_IN,
        CECARRYIN=>CECARRYIN_IN,
        CECTRL=>VCC_OPMODE,
        CEM=>CEM_IN,
        CEMULTCARRYIN=>CEMULTCARRYIN_IN,
        CEP=>CEP_IN,
        CLK=>CLK_IN,
        MULTSIGNIN=>GND_A,
        OPMODE(6)=>GND_OPMODE,
        OPMODE(5)=>VCC_OPMODE,
        OPMODE(4)=>VCC_OPMODE,
        OPMODE(3)=>GND_OPMODE,
        OPMODE(2)=>VCC_OPMODE,
        OPMODE(1)=>GND_OPMODE,
        OPMODE(0)=>VCC_OPMODE,
        PCIN(47 downto 0)=>GND_BUS_48(47 downto 0),
        RSTA=>RSTA_IN,
        RSTALLCARRYIN=>RSTALLCARRYIN_IN,
        RSTALUMODE=>GND_A,
        RSTB=>RSTB_IN,
        RSTC=>RSTC_IN,
        RSTCTRL=>GND_A,
        RSTM=>RSTM_IN,
        RSTP=>RSTP_IN,
        ACOUT(29 downto 0)=>ACOUT_OUT(29 downto 0),
        BCOUT(17 downto 0)=>BCOUT_OUT(17 downto 0),
        CARRYCASCOUT=>open,
        CARRYOUT=>open,
        MULTSIGNOUT=>open,
        OVERFLOW=>open,
        P(47 downto 38)=>P_float(9 downto 0),
        P(37 downto 0)=>P_OUT(37 downto 0),
        PATTERNBDETECT=>open,
        PATTERNDETECT=>open,
        PCOUT(47 downto 0)=>PCOUT_OUT(47 downto 0),
        UNDERFLOW=>open
        );
end BEHAVIORAL;
```

В представленном VHDL-описании элемента *macc_add_const_c_18_14_38* используются те же условные обозначения входных и выходных портов, что и в описании интерфейса компонентов *macc_add_sub_pc_14_12_32* и *macc_add_sub_dc_18_20_35*, которые были рассмотрены в предыдущих разделах. Структура элемента *macc_add_const_c_18_14_38* построена с применением одноступенчатой организации конвейерной обработки данных. Все входные, выходные и буферные регистры, используемые в составе этой структуры, имеют индивидуальные входы сигналов сброса и разрешения синхронизации, которые представлены в виде соответствующих интерфейсных портов в описании сгенерированного элемента. Кроме того, в состав интерфейса элемента *macc_add_const_c_18_14_38* включены дополнительные выходные шины ACOUT, BCOUT и PCOUT, предоставляющие возможность использования этого компонента в составе каскадного соединения аппаратных модулей ЦОС.

Для установки значений всех необходимых атрибутов библиотечного примитива DSP48E, используемого в качестве основного компонента VHDL-описания элемента *macc_add_const_c_18_14_38*, следует поместить в файл временных и топологических ограничений проекта разрабатываемого устройства ЦОС в САПР серии Xilinx ISE следующую совокупность выражений:

```
# Generated by Xilinx Architecture Wizard
# --- UCF Template Only ---
# Cut and paste these attributes into the project's UCF file, if desired
INST DSP48E_INST ACASCREG = 1;
INST DSP48E_INST ALUMODEREG = 0;
INST DSP48E_INST AREG = 1;
INST DSP48E_INST AUTORESET_PATTERN_DETECT = FALSE;
INST DSP48E_INST AUTORESET_PATTERN_DETECT_OPTINV =
MATCH;
INST DSP48E_INST A_INPUT = DIRECT;
INST DSP48E_INST BCASCREG = 1;
INST DSP48E_INST BREG = 1;
INST DSP48E_INST B_INPUT = CASCADE;
INST DSP48E_INST CARRYINREG = 1;
INST DSP48E_INST CARRYINSELREG = 0;
INST DSP48E_INST CREG = 1;
INST DSP48E_INST MASK = 3FFFFFFFFFFF;
INST DSP48E_INST MREG = 1;
INST DSP48E_INST MULTCARRYINREG = 1;
INST DSP48E_INST OPMODEREG = 0;
INST DSP48E_INST PATTERN = 000000000000;
INST DSP48E_INST PREG = 1;
INST DSP48E_INST SEL_MASK = MASK;
INST DSP48E_INST SEL_PATTERN = PATTERN;
INST DSP48E_INST SEL_ROUNDING_MASK = SEL_MASK;
INST DSP48E_INST USE_MULT = MULT_S;
INST DSP48E_INST USE_PATTERN_DETECT = NO_PATDET;
INST DSP48E_INST USE_SIMD = ONE48;
```

## Разработка компонентов, выполняющих операции умножения с накоплением над комплексными значениями и реализуемых на базе аппаратных секций DSP48E в ПЛИС FPGA серии Virtex-5, с использованием элементов, формируемых с помощью «мастера» Architecture Wizard

Кроме компонентов, выполняющих операции умножения с накоплением вещественных значений, в процессе проектирования устройств цифровой обработки сигналов могут использоваться элементы, осуществляющие аналогичные операции над комплексными значениями. Для реализации таких элементов, предназначенных для применения в составе высокоскоростных устройств ЦОС, выполняемых на базе ПЛИС семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT, необходимо использовать аппаратные секции DSP48E.

При разработке умножителей-накопителей комплексных значений в качестве компонентов можно применять описания элементов, сформированных с помощью «мастера» Architecture Wizard САПР серии Xilinx ISE. В общем случае алгоритм функционирования элементов, выполняющих операции умножения с накоплением над комплексными значениями, поясняет выражение:

$$Q = \sum_{n=1}^{N} (A_n + ja_n) \times (B_n + jb_n), \qquad (2)$$

где $A_n$ и $a_n$ — значения вещественной и мнимой части первого сомножителя на $n$-й итерации цикла накопления; $B_n$ и $b_n$ — значения вещественной и мнимой части второго сомножителя на $n$-й итерации соответственно.

Путем несложных преобразований выражение (2) можно привести к следующему виду:

$$Q = \sum_{n=1}^{N} ((A_n B_n - a_n b_n) + j(A_n b_n + B_n a_n)). \qquad (3)$$

Таким образом, выражения для вычисления вещественной (*ReQ*) и мнимой части (*ImQ*) результата операций умножения с накоплением комплексных значений можно представить в более наглядном виде, отражающем структуру элемента, реализующего эти операции:

$$ReQ = \sum_{n=1}^{N} A_n B_n - \sum_{n=1}^{N} a_n b_n, \qquad (4)$$

$$ImQ = \sum_{n=1}^{N} A_n b_n + \sum_{n=1}^{N} B_n a_n. \qquad (5)$$

Как следует из выражений (4) и (5), для реализации элемента, осуществляющего операции накопления произведений двух комплексных значений, достаточно четырех обычных (вещественных) умножителей-накопителей с традиционной структурой, процесс формирования которых был представлен в предыдущей части статьи, а также сумматор и вычитающее устройство. Если в структуре такого элемента использовать вместо умножителей-накопителей с традиционной архитектурой компоненты с возможностью динамического изменения конфигурации источников входных данных и выполняемой операции, рассмотренные в предыдущих разделах, то дополнительный сумматор и вычитающее устройство не потребуются. Выполняемые ими функции могут быть реализованы на базе тех же аппаратных секций DSP48E.

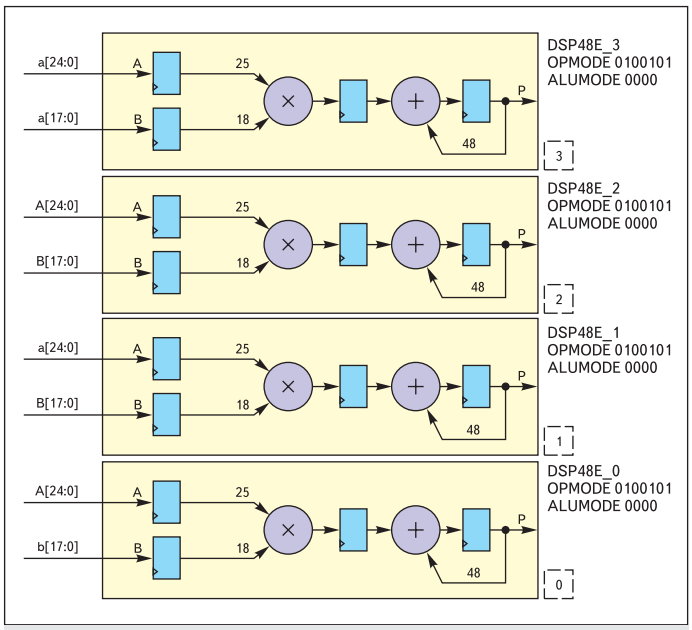


**Рис. 37.** Структура элемента, выполненная с использованием элементов, формируемых с помощью «мастера» Architecture Wizard для реализации на базе аппаратных секций DSP48E

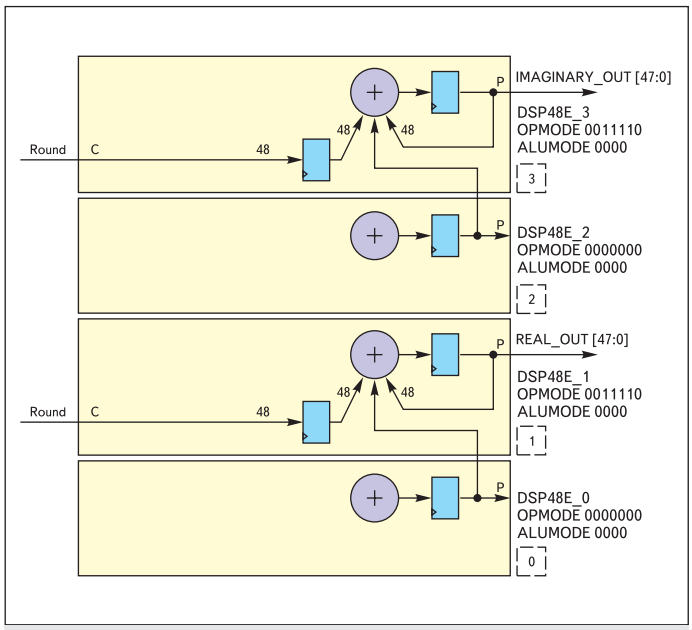


**Рис. 38.** Вид структуры элемента, выполненной с использованием элементов, формируемых с помощью «мастера» Architecture Wizard для реализации на базе аппаратных секций DSP48E, при вычислении итоговых результатов

На рис. 37 изображена структура элемента, осуществляющего операции умножения с накоплением над комплексными значениями, которая построена на основе компонентов, создаваемых с помощью «мастера» Architecture Wizard для последующей реализации на базе аппаратных секций DSP48E. Данный элемент предназначен для вычисления результата накопления произведений двух комплексных значений, представленных в 25- и 18-разрядном двоичном коде.

При выполнении заключительной итерации цикла накопления произведений на шины управления выбором источников входных данных и выполняемой функции арифметическо-логического блока каждой секции DSP48E, используемой в составе представленной структуры, подаются новые коды, соответствующие требуемым операциям. В результате этого структура элемента, изображенная на рис. 37, преобразуется к виду, показанному на рис. 38.

После вычисления итоговых результатов в соответствии с выражениями (4) и (5) производится восстановление первоначальных кодов на шинах управления выбором источников входных данных и выполняемой функции в используемых секциях DSP48E. При этом структура комплексного умножителя-накопителя вновь приобретает вид, приведенный на рис. 37.

Основным преимуществом представленной структуры элементов, осуществляющих операции умножения с накоплением над комплексными значениями, является минимальное количество аппаратных секций DSP48E, требуемых для ее реализации. Недостаток этой структуры проявляется в необходимости приостановки операций обработки входных данных для вычисления итоговых значений. Поэтому для применения в устройствах ЦОС с непрерывным процессом обработки входных данных вместо структуры, изображенной на рис. 37–38, нужно использовать структуру, представленную на рис. 39.

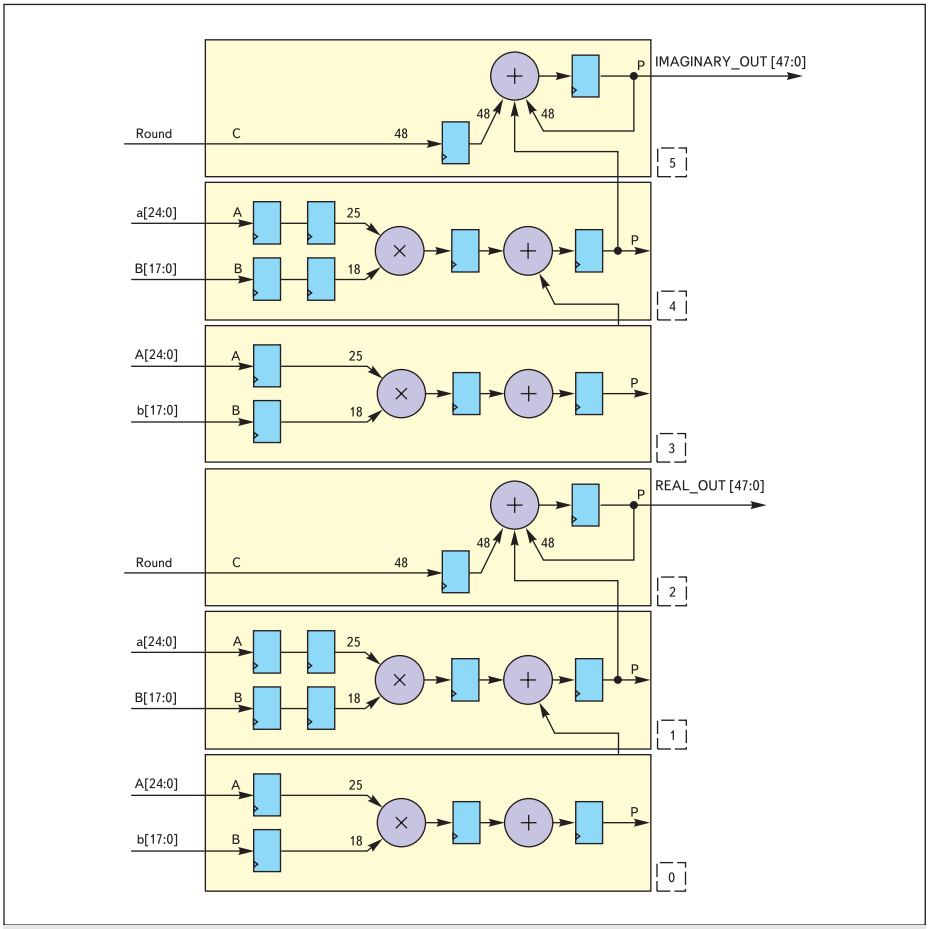


**Рис. 39.** Структура элемента для реализации на базе аппаратных секций DSP48E в устройствах ЦОС с непрерывным процессом обработки входных данных

Для реализации элементов с этой структурой необходимо шесть секций DSP48E, но при этом достигается непрерывность процесса обработки поступающих данных.

## Заключительные замечания

На этом завершается изучение процесса формирования наиболее распространенных компонентов высокопроизводительных устройств ЦОС, реализуемых на основе аппаратных секций DSP48E в кристаллах семейств Virtex-5 LX, Virtex-5 LXT, Virtex-5 SXT, Virtex-5 FXT и Virtex-5 TXT. Представленные примеры VHDL-описаний элементов, подготовленных с помощью «мастера» Architecture Wizard, наглядно демонстрируют возможность оперативного изменения значений параметров этих компонентов в среде встроенного HDL-редактора САПР серии Xilinx ISE, не прибегая к повторной процедуре их генерации. Особенности разработки более сложных узлов и функциональных блоков устройств цифровой обработки сигналов, выполняемых на базе аппаратных модулей DSP48E в ПЛИС серии Virtex-5, будут рассмотрены в одной из последующих публикаций.

## Литература


1. Кузелин М. О., Кнышев Д. А., Зотов В. Ю. Современные семейства ПЛИС фирмы Xilinx / Справочное пособие. М.: Горячая линия – Телеком, 2004.
2. 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.
3. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPack ISE. М.: Горячая линия – Телеком, 2003.
4. Зотов В. Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы Xilinx. М.: Горячая линия – Телеком, 2006.
5. Virtex-5 FPGA XtremeDSP Design Considerations. User Guide. Xilinx, 2009.
6. Virtex-5 Family Overview. Xilinx, 2009.
7. Virtex-5 FPGA User Guide. Xilinx, 2009.
8. Зотов В. Проектирование цифровых устройств, реализуемых на базе ПЛИС FPGA фирмы Xilinx, с использованием средств CORE Generator // Компоненты и технологии. 2006. № 12. 2007. № 1.
9. Зотов В. Разработка базовых компонентов цифровых устройств, реализуемых на базе ПЛИС FPGA фирмы Xilinx, с помощью генератора параметризированных модулей CORE Generator // Компоненты и технологии. 2007. № 2–12. 2008. № 1–8.