Продолжение. Начало в № 2 ’2007

Разработка базовых компонентов цифровых устройств, реализуемых на базе ПЛИС FPGA фирмы Xilinx® с помощью генератора параметризированных модулей CORE Generator

Валерий ЗОТОВ
walerry@km.ru

Система условных обозначений интерфейсных портов в описаниях умножителей, создаваемых на основе параметризированного модуля Multiplier Generator с помощью средств CORE Generator

Система условных обозначений входных и выходных портов в описаниях умножителей, генерируемых с помощью различных версий параметризированного модуля *Multiplier Generator*, выглядит следующим образом:

- clk — вход тактового сигнала выходного регистра;
- a[M:0] — входная шина данных A с разрядностью M+1;
- b[M:0] — входная шина данных B с разрядностью M+1;
- o[N:0] — комбинационные выходы генерируемого элемента (асинхронная выходная шина данных с разрядностью N+1);
- q[N:0] — регистровые выходы формируемого элемента (синхронная выходная шина данных с разрядностью N+1);
- p[N:0] — выходная шина данных умножителей, построенных на основе версии v9.0 рассматриваемого ядра;
- a_signed — вход выбора формы представления данных (со знаком или без знака) на входной шине A;
- loadb — вход сигнала управления загрузкой нового значения коэффициента с входной шины B;
- swapb — вход сигнала SWAPB, осуществляющего переключение банков внутренней памяти, используемой для хранения значений коэффициентов;
- load_done — выход сигнала LOAD_DONE, информирующего о состоянии процесса загрузки нового значения коэффициента;
- ce — вход сигнала разрешения синхронизации выходного регистра;
- aclr — вход сигнала асинхронного сброса выходного регистра;
- sclr — вход сигнала синхронного сброса выходного регистра;
- nd — вход сигнала New Data (ND), сообщающего о присутствии новых входных данных;
- rfd — выход сигнала Ready for Data (RFD), информирующего о готовности приема новых входных данных;
- rdy — выход сигнала Ready, подтверждающего готовность (достоверность) выходных данных.

Из представленной совокупности идентификаторов в описании интерфейса сгенерированного умножителя будут присутствовать условные обозначения только тех входных и выходных портов, которые поддерживаются соответствующей версией параметризированного модуля *Multiplier Generator* и были указаны разработчиком при определении его параметров. Основное отличие в системе условных обозначений интерфейсных портов элементов, формируемых с помощью разных версий данного ядра, проявляется в названиях выходных портов. Это отличие обусловлено тем, что в умножителях, выполняемых на основе версий v7.0 и v8.0 параметризированного модуля *Multiplier Generator*, могут одновременно присутствовать два выходных порта различного типа: регистровый и комбинационный. В элементах, создаваемых с помощью версии v9.0 рассматриваемого ядра, используется только один выходной порт, тип которого выбирается пользователем в процессе работы соответствующего «мастера» настройки параметров.

Пример описания умножителя, сформированного на основе параметризированного модуля Multiplier Generator версии v7.0

В качестве примера описания умножителя, сгенерированного с помощью параметризированного модуля *Multiplier Generator* версии v7.0, далее приведен текст VHDL-описания элемента multiplier_v7_16. Данный элемент предназначен для выполнения операции умножения 16-разрядных значений входных данных на коэффициенты, загружаемые в процессе функционирования разрабатываемого устройства. Формат представления данных, поступающих на входную шину A, (со знаком или без знака) выбирается с помощью соответствующего уровня сигнала на входе a_signed:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY multiplier_v7_16 IS
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(15 downto 0);
      b: IN std_logic_VECTOR(15 downto 0);
      o: OUT std_logic_VECTOR(17 downto 0);
      q: OUT std_logic_VECTOR(17 downto 0);
      a_signed: IN std_logic;
      loadb: IN std_logic;
      load_done: OUT std_logic;
      swapb: IN std_logic;
      ce: IN std_logic;
      aclr: IN std_logic;
      sclr: IN std_logic;
      rfd: OUT std_logic;
      nd: IN std_logic;
      rdy: OUT std_logic
      );
END multiplier_v7_16;
--
ARCHITECTURE multiplier_v7_16_a OF multiplier_v7_16 IS
-- synopsys translate_off
component wrapped_multiplier_v7_16
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(15 downto 0);
      b: IN std_logic_VECTOR(15 downto 0);
      o: OUT std_logic_VECTOR(17 downto 0);
      q: OUT std_logic_VECTOR(17 downto 0);
      a_signed: IN std_logic;
      loadb: IN std_logic;
      load_done: OUT std_logic;
      swapb: IN std_logic;
      ce: IN std_logic;
      aclr: IN std_logic;
      sclr: IN std_logic;
      rfd: OUT std_logic;
      nd: IN std_logic;
      rdy: OUT std_logic
      );
end component;
--
-- Configuration specification
   for all : wrapped_multiplier_v7_16 use entity
XilinxCoreLib.mult_gen_v7_0(behavioral)
```

```
        generic map(
                    c_a_type => 2,
                    c_mem_type => 2,
                    c_has_sclr => 1,
                    c_reg_a_b_inputs => 1,
                    c_has_q => 1,
                    c_has_o => 1,
                    bram_addr_width => 9,
                    c_v2_speed => 0,
                    c_baat => 16,
                    c_output_hold => 0,
                    c_b_constant => 0,
                    c_has_loadb => 1,
                    c_has_b => 1,
                    c_use_luts => 1,
                    c_has_rdy => 1,
                    c_has_nd => 1,
                    c_pipeline => 0,
                    c_has_a_signed => 1,
                    c_b_type => 0,
                    c_sqm_type => 0,
                    c_standalone => 1,
                    c_enable_rlocs => 1,
                    c_b_value => «01»,
                    c_mult_type => 4,
                    c_has_aclr => 1,
                    c_has_swapb => 1,
                    c_has_load_done => 1,
                    c_out_width => 18,
                    c_mem_init_prefix => «mem»,
                    c_b_width => 16,
                    c_a_width => 16,
                    c_has_rfd => 1,
                    c_sync_enable => 1,
                    c_has_ce => 1,
                    c_stack_adders => 1
                    );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_multiplier_v7_16
        port map (
                    clk => clk,
                    a => a,
                    b => b,
                    o => o,
                    q => q,
                    a_signed => a_signed,
                    loadb => loadb,
                    load_done => load_done,
                    swapb => swapb,
                    ce => ce,
                    aclr => aclr,
                    sclr => sclr,
                    rfd => rfd,
                    nd => nd,
                    rdy => rdy
                    );
-- synopsys translate_on
--
END multiplier_v7_16_a;
```

Для декларации сформированного компонента умножителя multiplier_v7_16 в состав соответствующего раздела VHDL-описания проектируемого устройства необходимо добавить следующую последовательность выражений:

```
component multiplier_v7_16
    port (
        clk: IN std_logic;
        a: IN std_logic_VECTOR(15 downto 0);
        b: IN std_logic_VECTOR(15 downto 0);
        o: OUT std_logic_VECTOR(17 downto 0);
        q: OUT std_logic_VECTOR(17 downto 0);
        a_signed: IN std_logic;
        loadb: IN std_logic;
        load_done: OUT std_logic;
        swapb: IN std_logic;
        ce: IN std_logic;
        aclr: IN std_logic;
        sclr: IN std_logic;
        rfd: OUT std_logic;
        nd: IN std_logic;
        rdy: OUT std_logic
        );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of multiplier_v7_16: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of multiplier_v7_16: component is true;
```

Для описания конкретного экземпляра компонента умножителя вида multiplier_v7_16 нужно воспользоваться оператором, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_умножителя_multiplier_v7_16>:
multiplier_v7_16
        port map (
                    clk => clk,
                    a => a,
                    b => b,
                    o => o,
                    q => q,
                    a_signed => a_signed,
                    loadb => loadb,
                    load_done => load_done,
                    swapb => swapb,
                    ce => ce,
                    aclr => aclr,
                    sclr => sclr,
                    rfd => rfd,
                    nd => nd,
                    rdy => rdy
                    );
```

Формирование описаний умножителей на основе параметризированного модуля Multiplier Generator версии v8.0 с помощью средств CORE Generator

В параметризированном модуле *Multiplier Generator* версии v8.0, в отличие от рассмотренной модификации v7.0 данного ядра, отсутствуют некоторые, редко используемые, функциональные возможности. В частности, ядро *Multiplier Generator* версии v8.0 не поддерживает возможность формирования умножителей последовательного типа. Кроме того, верхняя граница поддерживаемого диапазона разрядности входных данных в формируемых элементах, выполняющих операцию умножения на постоянный коэффициент, снижена до 32 разрядов. Сокращение функциональных возможностей в рассматриваемой версии параметризированного модуля позволило добиться повышения уровня оптимизации и производительности формируемых умножителей.

Заметные изменения произошли в конфигурации диалоговых панелей «мастера» настройки параметров ядра *Multiplier Generator*. Далее, в настоящем разделе, рассматривается процесс разработки умножителей с помощью параметризированного модуля *Multiplier Generator* версии v8.0.

«Мастер» настройки параметров данной версии ядра включает три или четыре диалоговые панели, в зависимости от типа создаваемого умножителя. В стартовой диалоговой панели данного «мастера», вид которой представлен на рис. 60, выбирается тип формируемого умножителя, разрядность входных шин данных, а также тип аппаратных ресурсов, используемых для его реализации.

Тип формируемого умножителя указывается с помощью индикатора состояния *Constant-Coefficient Multiplier*, который расположен во встроенной панели *Multiplier Type* стартовой диалоговой панели. Если данный индикатор находится в состоянии «Выключено», то формируемый элемент будет осуществлять операцию перемножения пары значений сигналов, представленных одновременно на двух входных шинах данных. Для генерации описаний элементов, выполняющих умножение входных данных, поступающих на шину A, на заданный коэффициент, нужно установить данный индикатор в состояние «Включено». При этом следует определить способ задания значения коэффициента в формируемом умножителе. Для этого необходимо воспользоваться индикатором состояния *Reloadable Constant*, который также представлен во встроенной

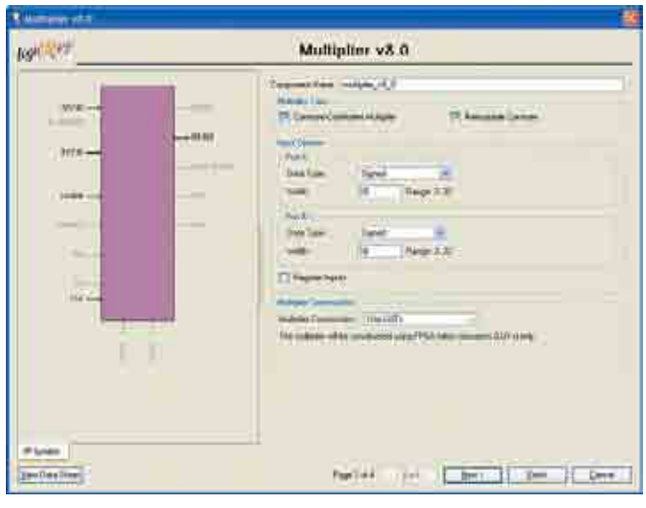

Рис. 60. Вид стартовой диалоговой панели «мастера» настройки параметров ядра умножителя Multiplier Generator версии v8.0

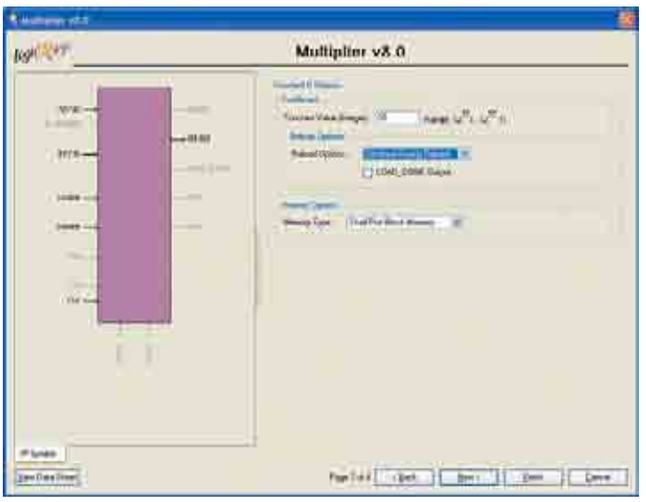

Рис. 61. Вид диалоговой панели Constant B Options «мастера» настройки параметров ядра умножителя Multiplier Generator версии v8.0, предназначенной для определения параметров задаваемого коэффициента

панели *Multiplier Type* (рис. 60). Когда данный индикатор находится в состоянии «Выключено», в создаваемом умножителе используется фиксированное значение коэффициента, определяемое в соответствующей диалоговой панели «мастера» настройки. Чтобы сформировать описание умножителя, в котором новое значение коэффициента может загружаться в процессе работы проектируемого устройства, следует установить индикатор *Reloadable Constant* в состояние «Включено».

Выбор формы представления значений для каждой входной шины данных осуществляется с помощью полей выбора *Data Type*, которые расположены во встроенных панелях *Port A* и *Port B* соответственно (рис. 60). Если в данном поле выбора указан вариант *Signed*, то входные данные на соответствующей шине воспринимаются как значения со знаком. При выборе варианта *Unsigned* входные данные интерпретируются как операнды без знака.

Разрядность входных шин данных A и B указывается в полях редактирования *Width*, которые также находятся во встроенных панелях *Port A* и *Port B*. При этом справа от каждого поля редактирования приводится информация о допустимом диапазоне значений данного параметра, который соответствует выбранному типу умножителя.

Чтобы включить в состав описания формируемого умножителя входные регистры, необходимо установить в состояние «Включено» индикатор *Register Inputs*. Входные регистры рекомендуется использовать при создании умножителей, в которых скорость выполнения операций является критичной.

Определение вида ресурсов ПЛИС, используемых для реализации формируемого умножителя, осуществляется с помощью поля выбора *Multiplier Construction*, которое представлено в одноименной встроенной панели (рис. 60). Содержание выпадающего списка данного поля выбора зависит от архитектурных особенностей семейства ПЛИС, предназначенного для реализации разрабатываемого умножителя. Если в кристалле, выбранном для реализации создаваемого умножителя, отсутствуют специализированные аппаратные ресурсы (18-разрядные блоки умножения Multiplier Blocks и блоки XtremeDSP), то этот список содержит единственную строку *Use LUTs*. Для семейств ПЛИС, в состав архитектуры которых входят аппаратные 18-разрядные блоки умножения Multiplier Blocks, в выпадающем списке поля выбора *Multiplier Construction* кроме указанной строки присутствуют еще два варианта: *Use 18×18 Multiplier Blocks* и *UseHybrid*. При использовании кристаллов семейства Virtex-4 для реализации создаваемого умножителя этот выпадающий список включает в себя следующие варианты: *Use LUTs*, *Use XtremeDSP Slice* и *UseHybrid*.

Чтобы сгенерировать описание умножителя, реализуемого на основе таблиц преобразования LUT, нужно в выпадающем списке поля выбора *Multiplier Construction* указать вариант *Use LUTs*. Для реализации формируемого умножителя на базе аппаратных 18-разрядных блоков умножения Multiplier Blocks, необходимо в этом списке выбрать строку *Use 18×18 Multiplier Blocks*. Чтобы сформировать описание умножителя, выполняемого на основе ресурсов блоков XtremeDSP, следует в предлагаемом списке выбрать вариант *Use XtremeDSP Slice*. При выборе варианта *UseHybrid* в составе разрабатываемого умножителя будут использованы аппаратные 18-разрядные блоки умножения Multiplier Blocks или блоки XtremeDSP (в зависимости от используемого семейства ПЛИС) в сочетании с ресурсами стандартной программируемой логики.

При генерации описания элемента, выполняющего операцию умножения входных данных на заданные коэффициенты, в качестве второй диалоговой панели «мастера» настройки параметров ядра умножителя *Multiplier Generator* версии v8.0 открывается панель *Constant B Options*. Данная диалоговая панель предназначена для определения параметров используемых коэффициентов. Вид этой диалоговой панели показан на рис. 61.

Для определения начального значения коэффициента, используемого в умножителе, нужно воспользоваться полем редактирования *Constant Value (Integer)*, которое расположено во встроенной панели *Coefficient* (рис. 61). Если в стартовой диалоговой панели был выбран вариант умножителя с возможностью загрузки нового значения коэффициента в процессе работы, то становятся доступными поле выбора *Reload Options* и индикатор состояния *LOAD_DONE Output*, находящиеся во встроенной панели *Reload Options*. Поле выбора *Reload Options* позволяет определить режим работы формируемого умножителя при загрузке нового значения коэффициента. Если в данном поле выбора указан вариант *Stop During Reload*, то будет сгенерировано описание умножителя с одним банком внутренней памяти, в котором при осуществлении загрузки нового значения коэффициента выполнение текущей операции умножения прерывается. Использование этого варианта может привести к появлению недостоверных данных, соответствующих результатам выполнения текущей операции умножения, на выходной шине умножителя. При выборе варианта *Continue During Reload* будет сформировано описание умножителя с двумя банками внутренней памяти, применение которых обеспечивает возможность загрузки нового значения коэффициента без прекращения выполнения текущей операции умножения. Данный вариант обеспечивает достоверность выходных данных умножителя при загрузке новых значений коэффициентов. Чтобы задействовать в создаваемом умножителе выход сигнала LOAD_DONE, сообщающего о завершении процесса загрузки нового значения коэффициента, нужно перевести в состояние «Включено» индикатор *LOAD_DONE Output*.

Определение типа памяти, используемой в создаваемом умножителе, выполняется с помощью поля выбора *Memory Type*, которое расположено во встроенной панели *Memory Options* (рис. 61). При выборе варианта *Distributed Memory* в составе формируемого умножителя применяется распределенная память ПЛИС, реализуемая на базе таблиц преобразования LUT. Если кристалл, предназначенный для реализации создаваемого элемента, обладает ресурсами блочной памяти, то для их использования нужно в поле выбора *Memory Type* указать вариант *Dual Port Block Memory*.

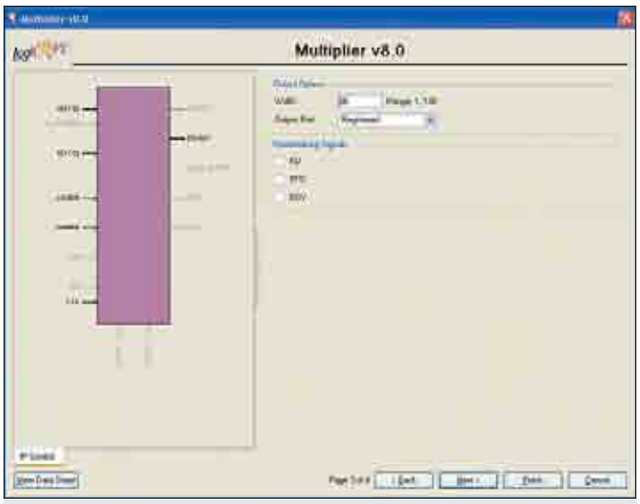

Рис. 62. Вид диалоговой панели «мастера» настройки ядра Multiplier Generator версии v8.0, предназначенной для определения параметров выходных шин формируемого умножителя и выбора сигналов подтверждения

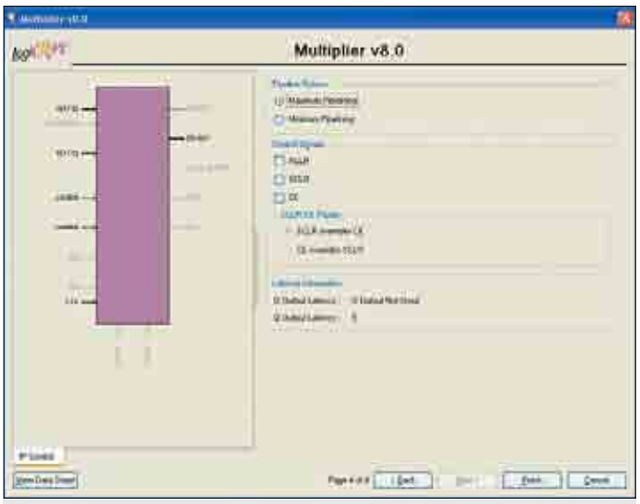

Рис. 63. Заключительная диалоговая панель «мастера» настройки параметров ядра Multiplier Generator версии v8.0, предназначенная для выбора варианта конвейерной обработки данных и сигналов управления выходного регистра в формируемом умножителе

Следующая (предпоследняя) диалоговая панель «мастера» настройки ядра *Multiplier Generator* версии v8.0 используется для указания параметров выходных шин создаваемого умножителя и выбора сигналов подтверждения. Вид данной диалоговой панели приведен на рис. 62.

Для определения разрядности выходных шин данных формируемого умножителя предназначено поле редактирования *Width*, которое расположено во встроенной панели *Output Options*. Тип выходных портов, включаемых в состав создаваемого умножителя, указывается с помощью поля выбора *Output Port*, находящегося в этой же встроенной панели (рис. 62). Если в выпадающем списке данного поля выбрана строка *Non-Registered Output*, то в формируемом умножителе будут задействованы только комбинационные (асинхронные) выходы. Для создания умножителя с регистровыми (синхронными) выходами нужно выбрать в представленном списке вариант *Registered Output*. Чтобы включить в состав генерируемого умножителя и комбинационные, и регистровые выходы, следует в поле выбора *Output Port* указать вариант *Both*.

Индикаторы состояния, представленные во встроенной панели *Handshaking Signals* (рис. 62), предназначены для выбора требуемых входов и выходов сигналов подтверждения. Чтобы задействовать в разрабатываемом умножителе вход сигнала New Data, нужно установить в состояние «Включено» индикатор *ND*. Для использования в формируемом умножителе выходов сигналов Ready for Data и/или Ready, следует переключить в активное состояние индикатор *RFD* и/или *RDY* соответственно.

Заключительная диалоговая панель «мастера» настройки параметров ядра *Multiplier Generator* версии v8.0 предоставляет пользователю возможность выбора одного из вариантов конвейерной обработки данных и сигналов управления выходного регистра в формируемом умножителе. Вид этой диалоговой панели показан на рис. 63.

Параметры конвейерной обработки данных в разрабатываемом умножителе выбираются с помощью двух кнопок с зависимой фиксацией, которые представлены во встроенной панели *Pipeline Options*. Если в нажатом состоянии зафиксирована кнопка *Maximum Pipelining*, то в структуре умножителя используется максимальное число дополнительных регистров, необходимых для организации конвейерной обработки данных. Для выбора структуры умножителя с минимальным количеством ступеней конвейерной обработки данных нужно переключить в нажатое положение кнопку *Minimum Pipelining*.

При включении в состав создаваемого умножителя выходного регистра становятся доступными индикаторы состояния и кнопки, расположенные во встроенной панели *Control Signals* (рис. 63). Индикаторы состояния позволяют выбрать управляющие входы выходного регистра, а кнопки — определить соотношение приоритетов сигналов на этих входах. Чтобы задействовать в выходном регистре входы асинхронного и/или синхронного сброса, нужно установить в состояние «Включено» индикаторы *ACLR* и/или *SCLR* соответственно. Для применения в составе умножителя выходного регистра с входом разрешения сигнала синхронизации, следует переключить в активное состояние индикатор *CE*. Если в выходном регистре предполагается совместное использование входов синхронного сброса и разрешения синхронизации, то необходимо указать приоритеты соответствующих сигналов с помощью группы кнопок с зависимой фиксацией *SCLR/CE Priority*. При установке в нажатое состояние кнопки *CE overrides SCLR* сигнал на входе разрешения синхронизации будет иметь более высокий приоритет, чем сигнал на входе синхронного сброса. Для назначения более высокого приоритета сигналу синхронного сброса по сравнению с сигналом разрешения синхронизации следует переключить в нажатое состояние кнопку *SCLR overrides CE*.

Встроенная информационная панель *Latency Information*, расположенная в нижней части заключительной диалоговой панели «мастера» настройки параметров ядра *Multiplier Generator* версии v8.0, содержит сведения о задержках выходных сигналов и возможные рекомендации по увеличению производительности формируемого умножителя.

Примером результата использования параметризированного модуля *Multiplier Generator* версии v8.0 является приведенное далее описание умножителя multiplier_v8_0_36. Этот элемент выполняет операцию умножения над двумя 36-разрядными значениями со знаком, представленными одновременно на входных шинах данных A и B. Результат выполнения операции отображается на регистровых (синхронных) и комбинационных (асинхронных) выходах. Сформированный умножитель реализуется на основе 18-разрядных аппаратных блоков Multiplier Blocks:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY multiplier_v8_0_36 IS
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(35 downto 0);
      b: IN std_logic_VECTOR(35 downto 0);
      o: OUT std_logic_VECTOR(71 downto 0);
      q: OUT std_logic_VECTOR(71 downto 0);
      ce: IN std_logic;
      sclr: IN std_logic;
      rfd: OUT std_logic;
      nd: IN std_logic;
```

```
        rdy: OUT std_logic
        );
END multiplier_v8_0_36;
--
ARCHITECTURE multiplier_v8_0_36_a OF multiplier_v8_0_36 IS
-- synopsys translate_off
component wrapped_multiplier_v8_0_36
   port (
        clk: IN std_logic;
        a: IN std_logic_VECTOR(35 downto 0);
        b: IN std_logic_VECTOR(35 downto 0);
        o: OUT std_logic_VECTOR(71 downto 0);
        q: OUT std_logic_VECTOR(71 downto 0);
        ce: IN std_logic;
        sclr: IN std_logic;
        rfd: OUT std_logic;
        nd: IN std_logic;
        rdy: OUT std_logic
        );
end component;
--
-- Configuration specification
   for all : wrapped_multiplier_v8_0_36 use entity
XilinxCoreLib.mult_gen_v8_0(behavioral)
        generic map(
                        c_a_type => 0,
                        c_mem_type => 0,
                        c_has_sclr => 1,
                        c_has_q => 1,
                        c_reg_a_b_inputs => 1,
                        c_has_o => 1,
                        c_family => «spartan3»,
                        bram_addr_width => 0,
                        c_v2_speed => 0,
                        c_baat => 0,
                        c_output_hold => 0,
                        c_b_constant => 0,
                        c_has_loadb => 0,
                        c_has_b => 0,
                        c_use_luts => 0,
                        c_has_rdy => 1,
                        c_has_nd => 1,
                        c_pipeline => 1,
                        c_has_a_signed => 0,
                        c_b_type => 0,
                        c_standalone => 0,
                        c_sqm_type => 0,
                        c_b_value => «1010»,
                        c_enable_rlocs => 0,
                        c_mult_type => 1,
                        c_has_aclr => 0,
                        c_mem_init_prefix => «mgv8»,
                        c_has_load_done => 0,
                        c_has_swapb => 0,
                        c_out_width => 72,
                        c_b_width => 36,
                        c_a_width => 36,
                        c_has_rfd => 1,
                        c_sync_enable => 1,
                        c_has_ce => 1,
                        c_stack_adders => 0
                        );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_multiplier_v8_0_36
        port map (
                        clk => clk,
                        a => a,
                        b => b,
                        o => o,
                        q => q,
                        ce => ce,
                        sclr => sclr,
                        rfd => rfd,
                        nd => nd,
                        rdy => rdy
                        );
-- synopsys translate_on
--
END multiplier_v8_0_36_a;
```

Блок декларации представленного умножителя multiplier_v8_0_36 при использовании его в качестве компонента разрабатываемого устройства имеет следующий вид:

```
component multiplier_v8_0_36
   port (
        clk: IN std_logic;
        a: IN std_logic_VECTOR(35 downto 0);
        b: IN std_logic_VECTOR(35 downto 0);
        o: OUT std_logic_VECTOR(71 downto 0);
        q: OUT std_logic_VECTOR(71 downto 0);
        ce: IN std_logic;
        sclr: IN std_logic;
        rfd: OUT std_logic;
        nd: IN std_logic;
        rdy: OUT std_logic
        );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of multiplier_v8_0_36: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of multiplier_v8_0_36: component is true;
```

Для создания экземпляра умножителя multiplier_v8_0_36 в описании разрабатываемого устройства нужно воспользоваться оператором, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_умножителя_multiplier_v8_0_36>:
multiplier_v8_0_36
        port map (
                        clk => clk,
                        a => a,
                        b => b,
                        o => o,
                        q => q,
                        ce => ce,
                        sclr => sclr,
                        rfd => rfd,
                        nd => nd,
                        rdy => rdy
                        );
```

Генерация описаний умножителей на основе параметризированного модуля Multiplier Generator версии v9.0 с помощью средств CORE Generator

Версия v9.0 параметризированного модуля умножителя *Multiplier Generator* представляет собой результат дальнейшего развития модификации v8.0 данного ядра, рассмотренной в предыдущем разделе. В новой версии ядра исключен ряд редко востребованных функциональных возможностей и добавлена поддержка архитектурных особенностей ПЛИС новых семейств. Наиболее существенными отличиями параметризированного модуля *Multiplier Generator* версии v9.0 от предыдущей модификации этого ядра являются:

- отсутствие поддержки сигналов подтверждения в создаваемых элементах;
- исключение возможности использования режима (и соответствующего входа) асинхронного сброса в выходных регистрах формируемых умножителей;
- расширение поддерживаемого диапазона разрядности входной шины данных генерируемых элементов, предназначенных для выполнения операции умножения входных значений на заданный коэффициент, за счет подъема верхней границы до 64 разрядов;
- наличие возможности генерации описаний умножителей с функцией округления результата выполнения операции (только при использовании кристаллов семейств Virtex-4 и Virtex-5);
- исключение поддержки динамически изменяемого формата представления данных (со знаком или без знака), поступающих на входную шину A;
- возможность создания умножителей только с одним выходным портом, тип которого выбирается разработчиком с помощью «мастера» настройки параметров ядра;
- отсутствие поддержки создания умножителей с возможностью загрузки новых значений коэффициентов в процессе работы проектируемого устройства;
- возможность точного определения количества ступеней (дополнительных регистров) при организации конвейерной обработки данных в формируемых умножителях;
- изменение структуры диалоговых панелей «мастера» настройки параметров ядра.

При обновлении разработанных ранее проектов следует обратить особое внимание на отсутствие обратной совместимости версии v9.0 параметризированного модуля *Multiplier Generator* с предыдущими модификациями этого ядра.

В составе «мастера» настройки параметров новой версии ядра используются три диалоговые панели, в каждой из которых присутствует дополнительная встроенная панель *Resource Estimates*, предназначенная для отображения оценочной информации о различных типах ресурсов ПЛИС, используемых для реализации формируемого умножителя. Эта встроенная панель располагается в той же области, что и изображение условного графического образа генерируемого умножителя (рис. 64). Для отображения встроенной панели *Resource Estimates* достаточно поместить курсор на закладку с ее названием и щелкнуть левой кнопкой мыши. Вид диалоговой панели «мастера» настройки с открытой встроенной информационной панелью изображен на рис. 65.

Стартовая диалоговая панель «мастера» настройки параметров ядра *Multiplier Generator* версии v9.0 предназначена для выбора типа создаваемого умножителя и определения разрядности входных шин данных (рис. 64). Тип генерируемого умножителя выбирается с помощью двух кнопок с зависимой фиксацией, которые расположены во встроенной панели *Multiplier Type*. Для формирования описания параллельного умножителя, осуществляющего перемножение значений данных, представленных на входных шинах, нужно зафиксировать в нажатом состоянии кнопку *Parallel Multiplier*. При нажатой кнопке *Constant Coefficient Multiplier* генерируется описание элемента, выполняющего операцию умножения входных данных на заданный коэффициент. Разрядность входных шин данных и формат представления значений (со знаком или без знака) в создаваемом умножителе определяются так же, как и в версии ядра *Multiplier Generator* v8.0, рассмотренной в предыдущем разделе.

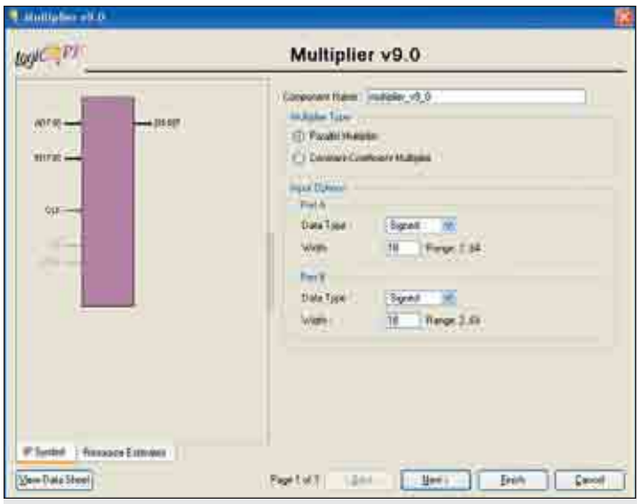

Рис. 64. Вид стартовой диалоговой панели «мастера» настройки параметров ядра умножителя Multiplier Generator версии v9.0

Рис. 65. Вид второй диалоговой панели «мастера» настройки параметров ядра Multiplier Generator версии v9.0 при создании параллельных умножителей

Вид второй диалоговой панели «мастера» настройки параметров ядра *Multiplier Generator* версии v9.0 зависит от типа умножителя, выбранного в стартовой диалоговой панели. При формировании параллельных умножителей вторая диалоговая панель «мастера» имеет вид, показанный на рис. 65.

Данная панель позволяет указать вид ресурсов ПЛИС, используемых для реализации создаваемого умножителя, а также критерий возможной оптимизации. Способ реализации формируемого умножителя определяется с помощью поля выбора *Multiplier Construction*. В отличие от «мастера» настройки предыдущих версий параметризированного модуля *Multiplier Generator* выпадающий список данного поля выбора содержит всего два варианта: *Use LUTs* и *UseMults*. При выборе первого варианта создаваемый умножитель будет реализован на базе таблиц преобразования LUT. Чтобы использовать для построения генерируемого элемента ресурсы аппаратных блоков умножения Multiplier Blocks или блоков XtremeDSP (в зависимости от выбранного семейства ПЛИС), следует в поле выбора *Multiplier Construction* указать вариант *UseMults*.

Если выбран второй вариант реализации создаваемого умножителя, то пользователю предоставляется возможность определения одного из двух критериев дополнительной оптимизации. Выбор требуемого критерия осуществляется с помощью двух кнопок с зависимой фиксацией, которые расположены во встроенной панели *Optimization Options* (рис. 65). Чтобы использовать в качестве критерия оптимизации формируемого умножителя достижение максимальной производительности, нужно зафиксировать в нажатом состоянии кнопку *Speed Optimized*. Для минимизации площади кристалла, используемой для реализации генерируемого умножителя, следует переключить в нажатое положение кнопку *Area Optimized*.

При генерации описания элемента, выполняющего операцию умножения значений входных данных на постоянный коэффициент, вторая диалоговая панель «мастера» приобретает вид, представленный на рис. 66.

С помощью данной панели определяется значение постоянного коэффициента и выбирается тип ресурсов ПЛИС, используемых для реализации формируемого умножителя. Следует обратить внимание на то, что для коэффициента, указываемого в поле редактирования *Constant Value (Integer)* (рис. 66), существенно расширен диапазон допустимых значений по сравнению с предыдущей версией параметризированного модуля *Multiplier Generator*. Увеличение абсолютных значений границ этого диапазона связано, прежде всего, с расширением поддерживаемого диапазона разрядности входной шины данных генерируемых умножителей на постоянный коэффициент — до 64 разрядов.

Для определения типа ресурсов кристалла, на основе которых будет построен создаваемый умножитель, нужно воспользоваться полем выбора *Memory Type*, которое расположено во встроенной панели *Memory Options* (рис. 66). Содержимое выпадающего списка данного поля выбора автоматически корректируется в соответствии с архитектурными особенностями семейства ПЛИС, используемого для реализации формируемого умножителя. Для всех семейств кристаллов, поддер-

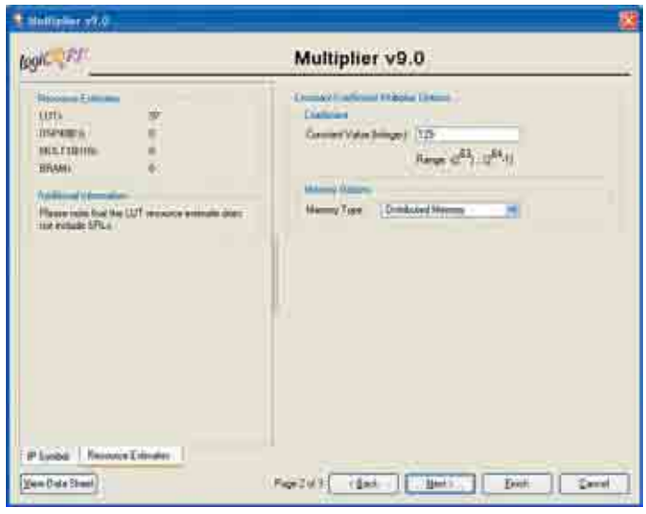

Рис. 66. Вид второй диалоговой панели «мастера» настройки параметров ядра Multiplier Generator версии v9.0 при создании умножителей значений входных данных на постоянный коэффициент

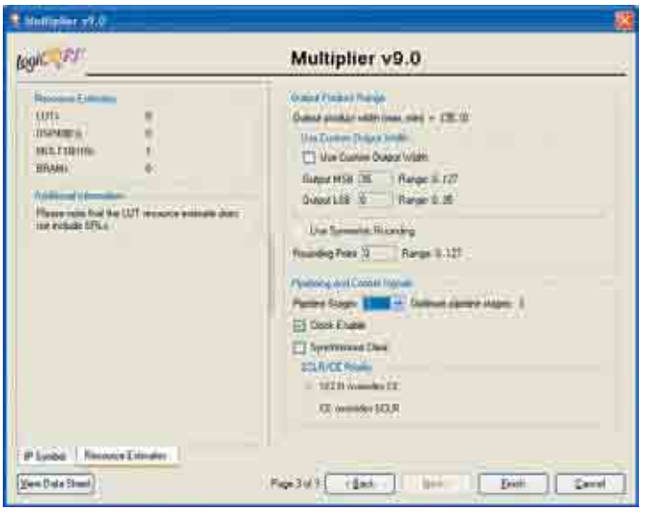

Рис. 67. Вид заключительной диалоговой панели «мастера» настройки параметров ядра умножителя Multiplier Generator версии v9.0

живаемых данной версией параметризированного модуля *Multiplier Generator*, в этом списке представлен вариант *Distributed Memory*, при выборе которого генерируется описание умножителя, реализуемого на базе таблиц преобразования LUT. Чтобы сформировать описание умножителя, выполняемого на основе ресурсов блочной памяти ПЛИС, следует выбрать вариант *Block Memory*. Если для реализации создаваемого умножителя был выбран кристалл, в составе архитектуры которого представлены аппаратные блоки умножения Multiplier Blocks или блоки XtremeDSP, то выпадающий список поля выбора *Memory Type* будет содержать дополнительную строку *Dedicated Multiplier*. При выборе последнего варианта создается описание умножителя, выполняемого на базе соответствующего аппаратного блока ПЛИС с возможным привлечением ресурсов стандартной программируемой логики кристалла.

Третья, заключительная, диалоговая панель «мастера» настройки ядра *Multiplier Generator* версии v9.0 используется для определения параметров выходного порта и количества ступеней при организации конвейерной обработки данных в создаваемом умножителе. Вид этой диалоговой панели приведен на рис. 67.

Разрядность выходного порта формируемого умножителя, необходимая для представления получаемого результата без округления, вычисляется автоматически на основании значений разрядности входных шин данных, указанных в стартовой панели «мастера» настройки (рис. 64). Информация о разрядности выходного порта отображается в строке *Output product width (max, min)*, которая расположена во встроенной панели *Output Product Range* (рис. 67). Эта информация приводится в виде совокупности пары значений: номера старшего значащего разряда (most-significant bit, MSB) и номера младшего значащего разряда (least-significant bit, LSB). При этом пользователю предоставляется возможность изменения отображаемых значений. Для изменения разрядности выходного порта и номеров старшего и младшего разрядов нужно установить в состояние «Включено» индикатор *Use Custom Output Width*, который находится во встроенной панели с тем же названием. После этого становятся доступными поля редактирования *Output MSB* и *Output LSB*, расположенные в этой же встроенной панели. Первоначально в этих полях отображаются автоматически вычисленные значения номеров старшего (*Output MSB*) и младшего (*Output LSB*) разрядов получаемого результата умножения. При необходимости можно отредактировать представленные значения в пределах допустимых диапазонов, которые указаны справа от полей редактирования.

Если разрабатываемое устройство предназначено для реализации в кристаллах семейств Virtex-4 или Virtex-5 и критерием оптимизации является достижение максимального быстродействия, то, при необходимости, пользователь может сформировать умножитель, выполняющий вычисления произведения с округлением результата. Для этого, прежде всего, нужно активизировать режим редактирования разрядности выходного порта, установив во включенное состояние индикатор *Use Custom Output Width*. В результате этих действий становится доступным индикатор состояния *Use Symmetric Rounding*. Чтобы сгенерировать описание умножителя, функционирующего в режиме вычислений произведения с округлением результата, следует перевести этот индикатор в состояние «Включено». При этом в поле *Point Rounding* автоматически отображается номер двоичного разряда, до которого осуществляется округление. Фактически в этом поле отражается значение номера младшего значащего разряда выходного порта, указанное в поле редактирования *Output LSB*.

Для определения количества ступеней (дополнительных регистров) при организации конвейерной обработки данных и выбора входов управления выходного регистра (при его использовании) в формируемом умножителе предназначены поле выбора *Pipeline Stages* и индикаторы состояния *Clock Enable* и *Synchronous Clear*, расположенные во встроенной панели *Pipelining and Control Signals* заключительной диалоговой панели «мастера» настройки (рис. 67). Количество регистров, необходимых для организации конвейерной обработки данных, указывается в поле выбора *Pipeline Stages*. Если в выпадающем списке возможных вариантов этого поля выбрано нулевое значение, то будет сформировано описание умножителя с комбинационными выходами. При выборе единичного значения в этом списке выполняется генерация описания умножителя, в состав которого включен выходной регистр. В том случае, когда в поле выбора *Pipeline Stages* указывается значение больше единицы, в структуре создаваемого умножителя кроме выходного регистра присутствуют дополнительные регистры, используемые для организации конвейерной обработки данных. Информация об оптимальном количестве регистров приводится в строке *Optimum pipeline stages*, которая находится справа от рассматриваемого поля выбора.

Если в поле *Pipeline Stages* выбран вариант, при котором в состав формируемого элемента добавляется выходной регистр, то становятся доступными индикаторы состояния, предназначенные для выбора входов управления выходного регистра. Чтобы задействовать в выходном регистре вход сигнала разрешения синхронизации, нужно установить в состояние «Включено» индикатор *Clock Enable*. Для использования режима синхронного сброса (и соответствующего входа) в выходном регистре умножителя необходимо перевести во включенное состояние индикатор *Synchronous Clear*. При совместном использовании входов разрешения синхронизации и синхронного сброса нужно установить соотношение приоритетов сигналов на этих входах с помощью кнопок с зависимой фиксацией *SCLR overrides CE* и *CE overrides SCLR*, расположенных во встроенной панели *SCLR/CE Priority*. Определение требуемого соотношения приоритетов сигналов на указанных входах выполняется так же, как и в предыдущих версиях рассматриваемого параметризированного модуля умножителя *Multiplier Generator*.

Примером применения ядра *Multiplier Generator* версии v9.0 для создания параллельных умножителей, реализуемых на основе встроенных аппаратных блоков ПЛИС, является описание элемента multiplier_v9_0_25, который предназначен для вычисления произведения двух 25-разрядных значений данных, представленных одновременно на входных шинах. В состав этого умножителя включен выходной регистр с входами сигналов разрешения синхронизации и синхронного сброса:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY multiplier_v9_0_25 IS
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(24 downto 0);
      b: IN std_logic_VECTOR(24 downto 0);
      ce: IN std_logic;
      sclr: IN std_logic;
      p: OUT std_logic_VECTOR(49 downto 0)
);
END multiplier_v9_0_25;
--
ARCHITECTURE multiplier_v9_0_25_a OF multiplier_v9_0_25 IS
-- synopsys translate_off
component wrapped_multiplier_v9_0_25
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(24 downto 0);
      b: IN std_logic_VECTOR(24 downto 0);
      ce: IN std_logic;
      sclr: IN std_logic;
      p: OUT std_logic_VECTOR(49 downto 0)
      );
end component;
--
-- Configuration specification
   for all : wrapped_multiplier_v9_0_25 use entity
XilinxCoreLib.mult_gen_v9_0(behavioral)
      generic map(
                  c_a_width => 25,
                  c_b_type => 1,
                  c_ce_overrides_sclr => 1,
                  c_opt_goal => 1,
                  c_has_sclr => 1,
                  c_round_pt => 0,
                  c_out_high => 49,
                  c_mult_type => 1,
                  c_ccm_imp => 0,
                  c_has_load_done => 0,
                  c_pipe_stages => 5,
                  c_has_ce => 1,
                  c_has_zero_detect => 0,
                  c_round_output => 0,
                  c_mem_init_prefix => «mgv9»,
                  c_xdevicefamily => «spartan3e»,
                  c_a_type => 1,
                  c_out_low => 0,
                  c_b_width => 25,
                  c_b_value => «10000001»
                  );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_multiplier_v9_0_25
      port map (
                  clk => clk,
                  a => a,
                  b => b,
                  ce => ce,
                  sclr => sclr,
                  p => p
                  );
-- synopsys translate_on
--
END multiplier_v9_0_25_a;
```

Для декларации компонента multiplier_v9_0_25 необходимо поместить в соответствующий раздел VHDL-описания разрабатываемого устройства приведенную далее совокупность выражений:

```
component multiplier_v9_0_25
   port (
      clk: IN std_logic;
      a: IN std_logic_VECTOR(24 downto 0);
      b: IN std_logic_VECTOR(24 downto 0);
      ce: IN std_logic;
      sclr: IN std_logic;
      p: OUT std_logic_VECTOR(49 downto 0)
      );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of multiplier_v9_0_25: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of multiplier_v9_0_25: component is true;
```

Создание экземпляра умножителя в составе описания проектируемого устройства осуществляется с помощью оператора, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_умножителя_multiplier_v9_0_25>
: multiplier_v9_0_25
      port map (
                  clk => clk,
                  a => a,
                  b => b,
                  ce => ce,
                  sclr => sclr,
                  p => p
                  );
```

Продолжение следует