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

Формирование описаний элементов, предназначенных для выполнения операций поворота вектора на заданный угол и преобразования координат вектора методами CORDIC, средствами генератора параметризированных модулей CORE Generator

Поворот вектора на заданный угол — это основная операция, применяемая в элементах, создаваемых на основе параметризированного модуля *CORDIC*. Результатом поворота вектора с координатами (X, Y) на некоторый угол θ является вектор, координаты которого (X', Y') в общем случае определяются следующей совокупностью выражений:

$$X' = (\cos(\theta)\times X - \sin(\theta)\times Y), \quad (17)$$

$$Y' = (\cos(\theta)\times Y + \sin(\theta)\times X). \quad (18)$$

На рис. 117 приведена векторная диаграмма, демонстрирующая выполнение операции поворота вектора на заданный угол.

При использовании методов CORDIC поворот вектора на угол θ выполняется в виде последовательности микроповоротов, которая включает в себя N итераций. На каждой из этих итераций осуществляется поворот вектора на угол i, значение которого определяется по формуле:

$$\theta_i = \text{atan}(2^{-i}), \quad (19)$$

где i — порядковый номер итерации. Количество итераций N, на которое разбивается процесс поворота вектора на заданный угол, выбирается в соответствии с требуемой точностью представления вычисляемых координат результирующего вектора.

При итерационной реализации процесса поворота вектора приведенные выражения, определяющие значения координат результирующего вектора, преобразуются к следующему виду:

$$X' = \prod_{i=1}^{N}\cos(\text{atan}(2^{-i}))(X_i - \alpha_i\times Y_i\times 2^{-i}), \quad (20)$$

$$Y' = \prod_{i=1}^{N}\cos(\text{atan}(2^{-i}))(Y_i + \alpha_i\times X_i\times 2^{-i}), \quad (21)$$

$$\theta' = \sum_{i=1}^{N}\theta - (\alpha_i\times\text{atan}(2^{-i})), \quad (22)$$

где X_i и Y_i — координаты вектора на i-й итерации, α_i — коэффициент, определяющий знак (α_i = (+ или –)1).

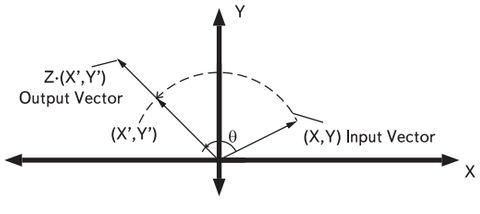

Рис. 117. Векторная диаграмма, иллюстрирующая выполнение операции поворота вектора на заданный угол

Каждая итерация в процессе поворота вектора может быть осуществлена с помощью операций сдвига и сложения/вычитания. Для вычисления координат вектора, получаемого в результате выполнения очередной итерации, можно воспользоваться следующими формулами:

$$x_{i+1} = x_i - \alpha_i\times y_i\times 2^{-i}, \quad (23)$$

$$y_{i+1} = y_i + \alpha_i\times x_i\times 2^{-i}, \quad (24)$$

$$\theta_{i+1} = \theta_i + \alpha_i\times\text{atan}(2^{-i}). \quad (25)$$

Значения коэффициента, определяющего знак, при выполнении операции поворота вектора методами *CORDIC* выбираются таким образом, чтобы значение θ' сходилось к нулю. Иначе говоря, если $\theta_{i-1} \geq 0$, то $\alpha_i = -1$, а если $\theta_{i-1} < 0$, то $\alpha_i = +1$.

Когда операция поворота вектора на заданный угол осуществляется в соответствии с методами *CORDIC*, выражения, определяющие значения координат результирующего вектора, приобретают следующий вид:

$$X' = Z\times(\cos(\theta)\times X - \sin(\theta)\times Y), \quad (26)$$
$$Y' = Z\times(\cos(\theta)\times Y + \sin(\theta)\times X), \quad (27)$$

где Z — коэффициент масштабирования *CORDIC*. Значение этого коэффициента зависит только от выбранного количества итераций и вычисляется по формуле:

$$Z = 1/(\prod_{i=1}^{N}\text{acos}(\text{atan}(2^{-i}))). \quad (28)$$

Элементы, выполняющие операцию поворота вектора на заданный угол, можно применять также для преобразования полярных координат вектора в ортогональные. При этом в качестве исходного используется

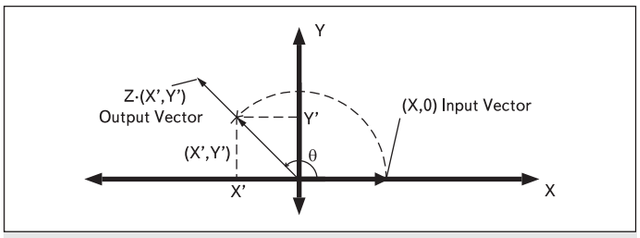

Рис. 118. Векторная диаграмма, иллюстрирующая преобразование полярных координат вектора в ортогональные с помощью операций поворота вектора на заданный угол

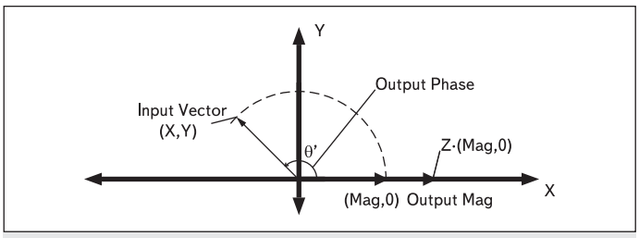

Рис. 119. Векторная диаграмма, поясняющая выполнение преобразования ортогональных координат вектора в полярные с помощью операций поворота вектора на некоторый угол

вектор с координатами (Mag, 0), где Mag представляет значение длины (модуля) вектора. Для преобразования полярных координат в ортогональные необходимо выполнить операцию поворота этого вектора на угол, равный значению фазы (угла) θ. Результатом преобразования являются ортогональные координаты вектора (X', Y'), который получается в итоге выполнения операции поворота. На рис. 118 представлена векторная диаграмма, которая в наглядной форме демонстрирует применение операции поворота вектора на заданный угол для преобразования полярных координат вектора в ортогональные.

Процесс преобразования ортогональных координат вектора (X, Y) в полярные можно представить как поворот исходного вектора на некоторый угол θ', при котором координата Y' результирующего вектора приобретает нулевое значение. Векторная диаграмма, показанная на рис. 119, поясняет осуществление преобразования ортогональных координат вектора в полярные с помощью операций поворота вектора на некоторый угол.

При этом значение координаты X' результирующего вектора соответствует длине (модулю) исходного вектора, умноженному на коэффициент масштабирования CORDIC, и определяется выражением (29):

$$X' = Z \times \sqrt{(X^2 + Y^2)}. \qquad (29)$$

Угол поворота θ', на который нужно повернуть вектор для достижения нулевого значения координаты Y', представляет собой значение фазы исходного вектора и вычисляется по следующей формуле:

$$\theta' = \text{atan}(X/Y). \qquad (30)$$

Значения коэффициента, определяющего знак в выражениях (23)–(25), при выполнении операции преобразования ортогональных координат вектора в полярные методами *CORDIC* выбирают таким образом, чтобы значение координаты Y' сходилось к нулю. Таким образом, если $y_{i-1} \geq 0$, то $\alpha_i = -1$, а если $y_{i-1} < 0$, то $\alpha_i = +1$.

В процессе вычисления коэффициента масштабирования *CORDIC* для определения значения выражения $\text{acos}(\text{atan}(2^{-i}))$ можно воспользоваться результатом разложения этой функции в ряд Тейлора:

$$\text{acos}(\text{atan}(2^{-i})) \approx (1+2^{-2i})^{-1/2}. \qquad (31)$$

При этом выражение, определяющее значение коэффициента масштабирования *CORDIC*, преобразуется к следующему виду:

$$Z = \prod_{i=1}^{N} (1+2^{-2i})^{1/2}. \qquad (32)$$

Для формирования описания элемента, выполняющего операцию поворота вектора на заданный угол, необходимо в стартовой диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0 во встроенной панели *Functional Selection* нажать кнопку *Rotate*. Чтобы сгенерировать описание элемента, предназначенного для преобразования ортогональных координат вектора в полярные, нужно зафиксировать в нажатом положении кнопку *Translate*. Далее, поочередно переходя к следующим диалоговым панелям «мастера» настройки, нужно выполнить практически ту же последовательность действий, что и при создании элементов, осуществляющих операции вычисления значений тригонометрических и гиперболических функций методами *CORDIC*, которая была рассмотрена в предыдущей части данной статьи (КиТ № 2 '2008). Основное отличие проявляется при выборе значений дополнительных параметров формируемого элемента, которые определяются в заключительной диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0. Вид страницы *Parameters* этой диалоговой панели при подготовке описаний элементов, выполняющих операции поворота вектора и преобразования значений ортогональных координат вектора в полярные, изображен на рис. 120.

Кроме возможности явного указания требуемого количества итераций, включающих в себя совокупность операций сдвига и сложения/вычитания, в поле редактирования *Iterations* и точности представления промежуточных результатов вычислений в поле редактирования *Precision*, данная панель позволяет добавить в состав архитектуры создаваемого элемента блок компенсации масшта-

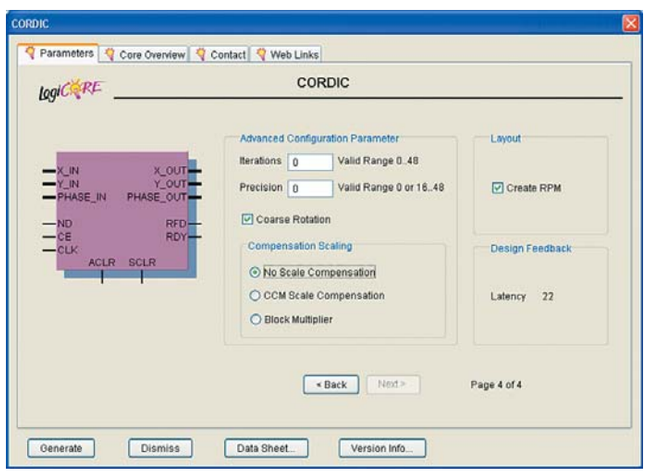

Рис. 120. Вид страницы Parameters заключительной диалоговой панели «мастера» настройки параметров ядра CORDIC версии v3.0 при подготовке описания элементов, выполняющих операции поворота вектора и преобразования значений ортогональных координат вектора в полярные

бирующего коэффициента *CORDIC*. При использовании этого блока в формируемом элементе значения координат результирующего вектора (X', Y'), полученного при выполнении операции поворота исходного вектора на заданный угол методами *CORDIC*, дополнительно умножаются на коэффициент, равный 1/Z. В случае применения блока компенсации масштабирующего коэффициента *CORDIC* разработчик может выбрать также способ реализации этого блока. Для этой цели предназначена группа кнопок с зависимой фиксацией, которая представлена во встроенной панели *Compensation Scaling* (рис. 120).

Параметризированный модуль *CORDIC* версии v3.0 позволяет формировать описания элементов, предназначенных для выполнения операций поворота вектора на заданный угол или преобразования ортогональных координат вектора в полярные, в которых блок компенсации масштабирующего коэффициента CORDIC может быть выполнен на основе ресурсов таблиц преобразования LUT или аппаратных умножителей Multiplier Blocks. Чтобы включить в состав создаваемого элемента блок компенсации, выполняемый в виде умножителя на константу (Constant Coefficient Multiplier), реализуемого на базе ресурсов таблиц преобразования LUT, следует переключить в нажатое состояние кнопку *CCM Scale Compensation*. Для генерации описания элемента с блоком компенсации, реализуемым на основе аппаратных умножителей ПЛИС Multiplier Blocks, нужно установить в нажатое положение кнопку *Block Multiplier*. Создание описания элементов, не содержащих блока компенсации масштабирующего коэффициента CORDIC, осуществляется при нажатой кнопке *No Scale Compensation*.

В описаниях элементов, предназначенных для выполнения операций поворота вектора на заданный угол и преобразования значений координат вектора из ортогональной системы в полярную или наоборот, используется следующая система условных обозначений входных и выходных портов:

- phase_in[N:0] — входная шина данных с разрядностью N+1, на которую подается двоичный код значения угла поворота вектора;
- x_in[N:0] — входная шина данных с разрядностью N+1, на которую поступает двоичный код значения координаты X исходного вектора;
- y_in[N:0] — входная шина данных с разрядностью N+1, на которую подается двоичный код значения координаты Y исходного вектора;
- x_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления значения координаты X' результирующего вектора;
- y_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления значения координаты Y' результирующего вектора;
- phase_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления значения фазы (угла) результирующего вектора.

Входные порты clk, ce, aclr, sclr, nd и выходные порты rfd и rdy имеют то же назначение, что и в элементах, формируемых для вычисления тригонометрических и гиперболических функций, которые были рассмотрены в предыдущей части статьи.

Пример описания элемента, предназначенного для выполнения операций поворота вектора на заданный угол, сформированного средствами генератора параметризированных модулей CORE Generator

Примером устройства, сформированного на основе параметризированного модуля CORDIC версии v3.0 для выполнения операций поворота вектора на заданный угол, является элемент *rotate_vector_cordic*, текст описания которого приведен далее в этом разделе. Значения координат входного (исходного) вектора и выходного вектора, являющегося результатом выполнения операции поворота, а также значения угла поворота в этом элементе представлены в виде 18-разрядного двоичного кода. Вычисление координат результирующего вектора в элементе *rotate_vector_cordic* осуществляется с применением блока компенсации масштабирующего коэффициента CORDIC, который реализуется на основе аппаратных умножителей Multiplier Blocks ПЛИС:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY rotate_vector_cordic IS
      port (
            x_in: IN std_logic_VECTOR(17 downto 0);
            y_in: IN std_logic_VECTOR(17 downto 0);
            phase_in: IN std_logic_VECTOR(17 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(17 downto 0);
            y_out: OUT std_logic_VECTOR(17 downto 0);
            phase_out: OUT std_logic_VECTOR(17 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
END rotate_vector_cordic;
--
ARCHITECTURE rotate_vector_cordic_a OF rotate_vector_cordic IS
-- synthesis translate_off
component wrapped_rotate_vector_cordic
      port (
            x_in: IN std_logic_VECTOR(17 downto 0);
            y_in: IN std_logic_VECTOR(17 downto 0);
            phase_in: IN std_logic_VECTOR(17 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(17 downto 0);
            y_out: OUT std_logic_VECTOR(17 downto 0);
            phase_out: OUT std_logic_VECTOR(17 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
end component;
--
-- Configuration specification
    for all : wrapped_rotate_vector_cordic use entity
XilinxCoreLib.cordic_v3_0(behavioral)
      generic map(
                  c_has_clk => 1,
                  c_has_x_out => 1,
                  c_has_y_in => 1,
                  c_reg_inputs => 1,
                  c_architecture => 1,
                  c_input_width => 18,
                  c_iterations => 0,
                  c_precision => 0,
                  c_has_rdy => 1,
                  c_has_sclr => 1,
                  c_has_nd => 1,
                  c_scale_comp => 2,
                  c_enable_rlocs => 1,
                  c_has_phase_in => 1,
                  c_has_rfd => 1,
                  c_cordic_function => 0,
                  c_has_ce => 1,
                  c_mif_file_prefix => «cor1»,
                  c_round_mode => 1,
                  c_has_aclr => 1,
                  c_sync_enable => 0,
                  c_has_y_out => 1,
                  c_data_format => 0,
                  c_reg_outputs => 1,
                  c_coarse_rotate => 1,
                  c_phase_format => 0,
                  c_has_phase_out => 1,
                  c_has_x_in => 1,
                  c_pipeline_mode => -1,
                  c_output_width => 18
                  );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_rotate_vector_cordic
      port map (
                  x_in => x_in,
                  y_in => y_in,
                  phase_in => phase_in,
                  nd => nd,
                  x_out => x_out,
                  y_out => y_out,
                  phase_out => phase_out,
                  rdy => rdy,
                  rfd => rfd,
                  clk => clk,
                  ce => ce,
                  aclr => aclr,
                  sclr => sclr
                  );
-- synthesis translate_on
--
END rotate_vector_cordic_a;
```

В элементе *rotate_vector_cordic* используется архитектура последовательного типа в сочетании с максимальным количеством ступеней конвейерной организации выполнения операций сдвига и сложения/вычитания. В состав структуры сформированного элемента включены входные и выходные регистры, а также блок «грубого» поворота вектора. Элемент *rotate_vector_cordic* позволяет осуществлять операции поворота вектора с использованием совокупности сигналов подтверждения, поддерживаемых параметризированным модулем CORDIC версии v3.0. Кроме того, в этом элементе задействованы входы сигналов синхронного и асинхронного сброса, а также вход сигнала разрешения синхронизации. При этом сигнал на входе синхронного сброса имеет более низкий приоритет по сравнению с сигналом на входе разрешения синхронизации.

Чтобы использовать элемент *rotate_vector_cordic* в качестве компонента в описании проектируемого устройства, необходимо добавить в раздел декларации этого описания следующую конструкцию:

```
component rotate_vector_cordic
      port (
            x_in: IN std_logic_VECTOR(17 downto 0);
            y_in: IN std_logic_VECTOR(17 downto 0);
            phase_in: IN std_logic_VECTOR(17 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(17 downto 0);
            y_out: OUT std_logic_VECTOR(17 downto 0);
            phase_out: OUT std_logic_VECTOR(17 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
end component;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of rotate_vector_cordic: component is true;
```

Для описания конкретных экземпляров компонента *rotate_vector_cordic* в блоке определения архитектуры разрабатываемого устройства нужно использовать оператор, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_элемента_rotate_vector_cordic>:
rotate_vector_cordic
      port map (
                  x_in => x_in,
                  y_in => y_in,
                  phase_in => phase_in,
                  nd => nd,
                  x_out => x_out,
                  y_out => y_out,
                  phase_out => phase_out,
                  rdy => rdy,
                  rfd => rfd,
                  clk => clk,
                  ce => ce,
                  aclr => aclr,
                  sclr => sclr
                  );
```

На рис. 121 представлена информационная панель, в которой отражены подробные сведения о количестве различных ресурсов ПЛИС, в том числе аппаратных умножителей Multiplier Blocks, используемых при автономной реализации рассмотренного элемента.

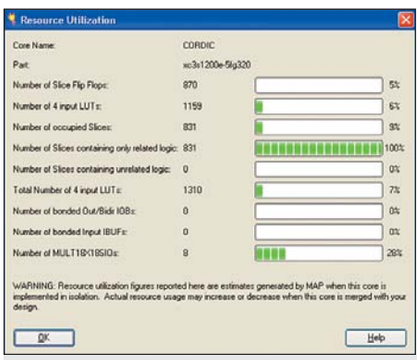

Рис. 121. Вид информационной панели, содержащей сведения о количестве различных ресурсов ПЛИС, используемых для реализации элемента rotate_vector_cordic

Пример описания элемента, предназначенного для преобразования значений ортогональных координат вектора в полярные, сформированного средствами генератора параметризированных модулей CORE Generator

В качестве примера использования параметризированного модуля CORDIC версии v3.0 для создания элементов, выполняющих преобразование значений ортогональных координат вектора в полярные, в настоящем разделе приводится текст VHDL-описания элемента *translate_vector_cordic*. В данном элементе значения исходных координат и результаты преобразования представлены в виде 28-разрядного двоичного кода. В процессе трансляции значений ортогональных координат вектора в полярные используется блок компенсации масштабирующего коэффициента CORDIC, реализация которого осуществляется на базе ресурсов таблиц преобразования LUT:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY translate_vector_cordic IS
      port (
            x_in: IN std_logic_VECTOR(27 downto 0);
            y_in: IN std_logic_VECTOR(27 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(27 downto 0);
            y_out: OUT std_logic_VECTOR(27 downto 0);
            phase_out: OUT std_logic_VECTOR(27 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
END translate_vector_cordic;
--
ARCHITECTURE translate_vector_cordic_a OF translate_vector_cordic
IS
-- synthesis translate_off
component wrapped_translate_vector_cordic
      port (
            x_in: IN std_logic_VECTOR(27 downto 0);
            y_in: IN std_logic_VECTOR(27 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(27 downto 0);
            y_out: OUT std_logic_VECTOR(27 downto 0);
            phase_out: OUT std_logic_VECTOR(27 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
end component;
--
-- Configuration specification
      for all : wrapped_translate_vector_cordic use entity
XilinxCoreLib.cordic_v3_0(behavioral)
      generic map(
                  c_has_clk => 1,
                  c_has_x_out => 1,
                  c_has_y_in => 1,
                  c_reg_inputs => 1,
                  c_architecture => 1,
                  c_input_width => 28,
                  c_iterations => 0,
                  c_precision => 0,
                  c_has_rdy => 1,
                  c_has_sclr => 1,
                  c_has_nd => 1,
                  c_scale_comp => 1,
                  c_enable_rlocs => 1,
                  c_has_phase_in => 0,
                  c_has_rfd => 1,
                  c_cordic_function => 1,
                  c_has_ce => 1,
                  c_mif_file_prefix => «cor1»,
                  c_round_mode => 2,
                  c_has_aclr => 1,
                  c_sync_enable => 1,
                  c_has_y_out => 1,
                  c_data_format => 0,
                  c_reg_outputs => 1,
                  c_coarse_rotate => 1,
                  c_phase_format => 0,
                  c_has_phase_out => 1,
                  c_has_x_in => 1,
                  c_pipeline_mode => -1,
                  c_output_width => 28
                  );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_translate_vector_cordic
      port map (
                  x_in => x_in,
                  y_in => y_in,
                  nd => nd,
                  x_out => x_out,
                  y_out => y_out,
                  phase_out => phase_out,
                  rdy => rdy,
                  rfd => rfd,
                  clk => clk,
                  ce => ce,
                  aclr => aclr,
                  sclr => sclr
                  );
-- synthesis translate_on
--
END translate_vector_cordic_a;
```

В сформированном элементе *translate_vector_cordic* применяется архитектура последовательного типа, в состав которой дополнительно включены входные и выходные регистры. Использование оптимального варианта конвейерной организации выполнения операций позволяет повысить производительность этого элемента без привлечения дополнительных таблиц преобразования LUT кристалла. В состав интерфейса рассматриваемого элемента входят те же входы и выходы сигналов управления и подтверждения, что и в элементе *rotate_vector_cordic*, который был представлен в предыдущем разделе. Но при этом сигнал на входе разрешения синхронизации элемента *translate_vector_cordic* имеет более низкий приоритет, чем сигнал на входе синхронного сброса.

Если элемент *translate_vector_cordic* применяется в качестве компонента в описании разрабатываемого устройства, то для его декларации необходимо поместить в соответствующий раздел следующую совокупность выражений:

```
component translate_vector_cordic
      port (
            x_in: IN std_logic_VECTOR(27 downto 0);
            y_in: IN std_logic_VECTOR(27 downto 0);
            nd: IN std_logic;
            x_out: OUT std_logic_VECTOR(27 downto 0);
            y_out: OUT std_logic_VECTOR(27 downto 0);
            phase_out: OUT std_logic_VECTOR(27 downto 0);
            rdy: OUT std_logic;
            rfd: OUT std_logic;
            clk: IN std_logic;
            ce: IN std_logic;
            aclr: IN std_logic;
            sclr: IN std_logic
            );
end component;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of translate_vector_cordic: component is true;
```

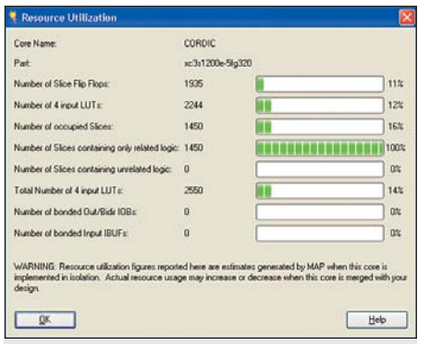

Рис. 122. Вид информационной панели, содержащей сведения о ресурсах кристалла, используемых для реализации элемента translate_vector_cordic

Для создания конкретных экземпляров компонента *translate_vector_cordic* в составе описания проектируемого устройства нужно использовать оператор, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_элемента_translate_vector_cordic>:
translate_vector_cordic
    port map (
                x_in => x_in,
                y_in => y_in,
                nd => nd,
                x_out => x_out,
                y_out => y_out,
                phase_out => phase_out,
                rdy => rdy,
                rfd => rfd,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
```

Сведения о количестве триггеров, таблиц преобразования LUT и секций Slices, которое необходимо для автономной реализации элемента *translate_vector_cordic*, приведены в информационной панели, показанной на рис. 122.

Подготовка описаний элементов, предназначенных для выполнения операции извлечения квадратного корня, на основе параметризированного модуля CORDIC версии v3.0 средствами генератора CORE Generator

При подготовке описаний элементов, предназначенных для выполнения операции извлечения квадратного корня, состав определяемых параметров конфигурации ядра CORDIC версии v3.0 отличается по сравнению с процессом генерации устройств, осуществляющих вычисления значений тригонометрических и гиперболических функций, поворот вектора на заданный угол и преобразование координат вектора из одной системы в другую. Следствие этого — изменение структуры диалоговых панелей «мастера» настройки параметров ядра *CORDIC* версии v3.0, которые были представлены в предыдущей части статьи. На рис. 123 показан вид страницы *Parameters* стартовой диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0 при подготовке описаний элементов, предназначенных для выполнения операции извлечения квадратного корня из значений, представленных на входной шине данных.

Для генерации описания элемента, вычисляющего значение квадратного корня, следует во встроенной панели *Functional Selection* нажать кнопку *Square Root* (рис. 123). Далее с помощью группы кнопок с зависимой фиксацией *Pipelining Mode* нужно выбрать требуемый вариант конвейерной организации выполнения вычислений в создаваемом элементе. При этом следует обратить внимание на то, что параметризированный модуль CORDIC версии v3.0 позволяет формировать описания элементов, предназначенных для выполнения операции извлечения квадратного корня, которые могут быть реализованы только на основе архитектуры параллельного типа. Поэтому кнопки выбора типа архитектуры генерируемого элемента, представленные во встроенной панели *Architectural Configuration*, находятся в недоступном состоянии (рис. 123).

Вторая диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0 при создании описаний элементов, предназначенных для выполнения операции извлечения квадратного корня из значений, представленных на входной шине данных, приобретает вид, изображенный на рис. 124.

В данной диалоговой панели нужно выбрать формат представления входных и выходных данных, а также указать дополнительные входы и выходы управляющих сигналов в формируемых элементах. В устройствах, разрабатываемых на основе параметризированного модуля *CORDIC* версии v3.0 для вычисления значений тригонометрических и гиперболических функций, поворота вектора на заданный угол и преобразования значений ортогональных координат вектора в полярные, входные и выходные данные неизменно интерпретировались как дробные числа со знаком. При создании описаний элементов, предназначенных для выполнения операции извлечения квадратного корня, значения исходных данных и результатов вычисления могут быть представлены в дробном или целочисленном формате без знака. Выбор формата представления входных и выходных данных осуществляется с помощью группы кнопок с зависимой фиксацией, расположенных во встроенной панели *Data Format* (рис. 124). Чтобы сформировать описание элемента, в котором значения исходных данных и результатов вычислений интерпретируются как дробные числа без знака, следует зафиксировать в нажатом положении кнопку *Unsigned Fraction*. Для генерации описания элемента, использующего беззнаковый целочисленный формат представления значений на входной и выходной шине, необходимо пере-

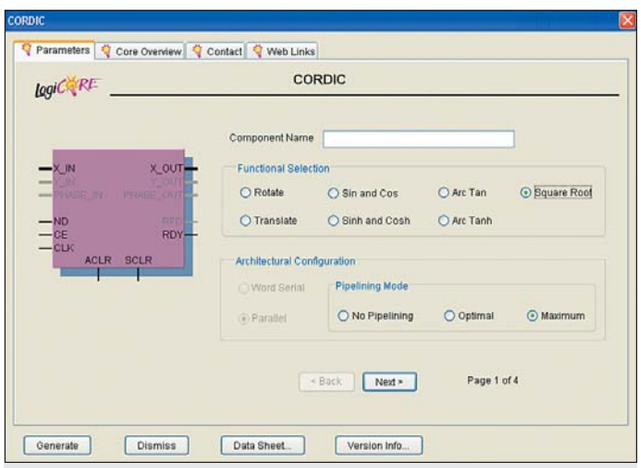

Рис. 123. Вид страницы Parameters стартовой диалоговой панели «мастера» настройки параметров ядра CORDIC версии v3.0 при подготовке описаний элементов, предназначенных для выполнения операции извлечения квадратного корня из значений, представленных на входной шине данных

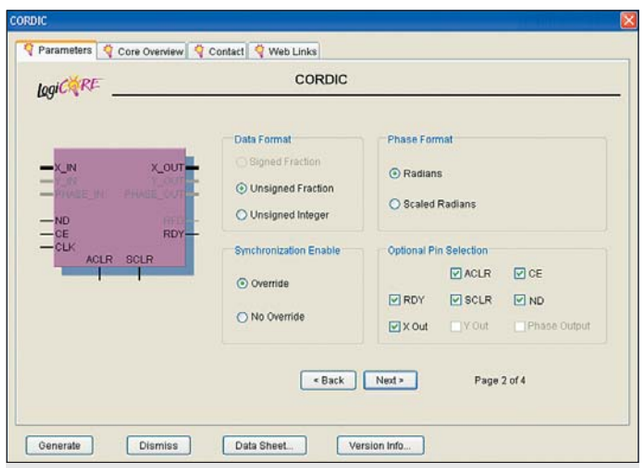

Рис. 124. Вид страницы Parameters второй диалоговой панели «мастера» настройки параметров ядра CORDIC версии v3.0 при подготовке описаний элементов, предназначенных для выполнения операции извлечения квадратного корня из значений, представленных на входной шине данных

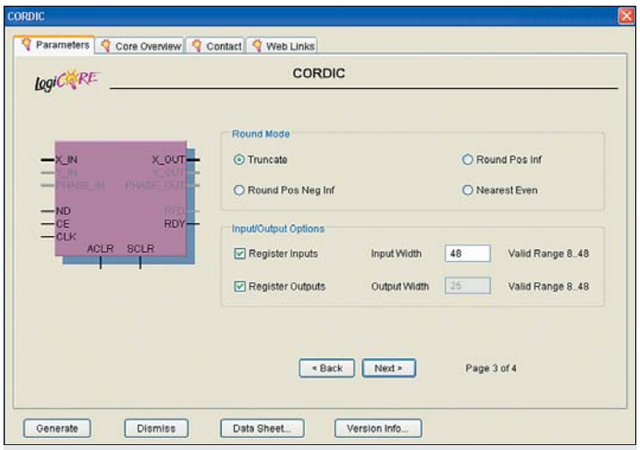

Рис. 125. Вид страницы Parameters третьей диалоговой панели «мастера» настройки параметров ядра CORDIC версии v3.0 при подготовке описаний элементов, предназначенных для выполнения операции извлечения квадратного корня из значений, представленных на входной шине данных

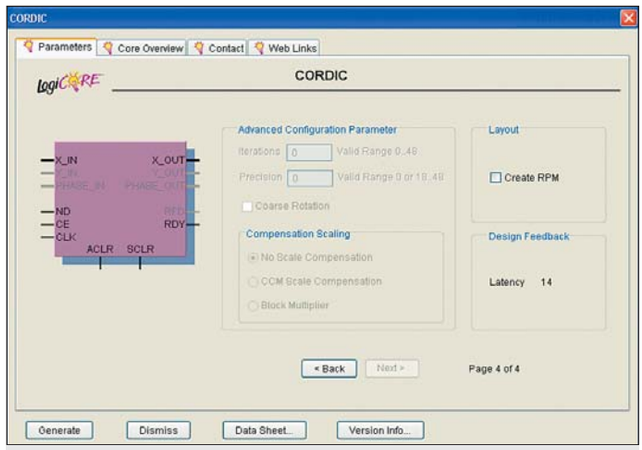

Рис. 126. Вид страницы Parameters заключительной диалоговой панели «мастера» настройки параметров ядра CORDIC версии v3.0 при подготовке описания элементов, предназначенных для выполнения операции извлечения квадратного корня

ключить в нажатое состояние кнопку *Unsigned Integer*.

Включение дополнительных входов и выходов сигналов управления и подтверждения в состав интерфейса создаваемых элементов производится с помощью индикаторов состояния, которые расположены во встроенной панели *Optional Pin Selection* (рис. 124). При этом следует обратить внимание на то, что не все дополнительные входы и выходы сигналов управления и подтверждения, предусмотренные в параметризированном модуле *CORDIC* версии v3.0, могут быть задействованы в элементах, предназначенных для выполнения операции извлечения квадратного корня. Если в составе интерфейса генерируемого элемента предполагается одновременное присутствие входов сигнала разрешения синхронизации и синхронного сброса, то следует уточнить соотношение приоритетов этих сигналов, воспользовавшись группой кнопок с зависимой фиксацией, которая находится во встроенной панели *Synchronization Enable*. Выбор доступных дополнительных входов и выходов сигналов управления и подтверждения, а также определение соотношения приоритетов сигналов разрешения синхронизации и синхронного сброса производится так же, как и при подготовке описаний элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций, поворота вектора на заданный угол и преобразования координат вектора из ортогональной системы в полярную.

Основное отличие в конфигурации третьей диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0 при подготовке описаний элементов, выполняющих операцию извлечения квадратного корня, проявляется в способе определения разрядности выходной шины, на которой отображаются результаты вычислений. Вид страницы *Parameters* данной диалоговой панели при создании описаний указанных элементов представлен на рис. 125.

В этой диалоговой панели следует выбрать метод округления результатов вычислений с помощью группы кнопок с зависимой фиксацией *Round Mode* и, при необходимости, добавить в состав архитектуры формируемого элемента входные и выходные регистры, используя индикаторы состояния, расположенные во встроенной панели *Input/Output Options* (рис. 125). Возможные значения этих параметров конфигурации создаваемых элементов были рассмотрены при подготовке описаний устройств, выполняющих операции вычисления значений тригонометрических и гиперболических функций, поворота вектора на заданный угол и преобразования координат вектора из ортогональной системы в полярную.

Разрядность двоичного кода входных значений в создаваемом элементе указывается в поле редактирования *Input Width*. При генерации описаний элементов, в которых применяется беззнаковый целочисленный формат представления значений на входной и выходной шинах (в нажатом состоянии зафиксирована кнопка *Unsigned Integer*), разрядность формируемого результата вычислений определяется автоматически в соответствии с указанным значением параметра *Input Width*. Автоматически установленное значение разрядности выходной шины отображается в поле *Output Width*, как показано на рис. 125, и разработчик его не может изменить. Если в создаваемом элементе выбран беззнаковый дробный формат представления входных и выходных значений (нажата кнопка *Unsigned Fraction*), то требуемое количество разрядов выходной шины определяется вручную в поле редактирования *Output Width*.

Заключительная диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0 в процессе подготовки описаний элементов, выполняющих операцию извлечения квадратного корня, используется только для указания формы представления генерируемого описания для последующей реализации в ПЛИС и получения информации о величине задержки отображения результатов вычислений на выходных шинах этих элементов. Вид страницы *Parameters* этой диалоговой панели при формировании описаний элементов, предназначенных для вычислений значений квадратного корня, показан на рис. 126.

В этом случае необходимое количество итераций, включающих в себя совокупность операций сдвига и сложения/вычитания, используемых в создаваемом элементе, а также точность промежуточных результатов вычислений устанавливаются автоматически в соответствии с выбранной разрядностью выходных данных. При этом разработчик не может изменить значения этих параметров, так как поля редактирования *Iterations* и *Precision*, расположенные во встроенной панели *Advanced Configuration Parameter* (рис. 126), находятся в недоступном состоянии.

Форма представления создаваемого описания для последующей реализации в ПЛИС выбирается с помощью индикатора состояния *Create RPM*, который находится во встроенной панели *Layout* (рис. 126). Информация о задержке формирования результатов вычислений квадратного корня приводится во встроенной панели *Design Feedback*.

Система условных обозначений интерфейсных портов в описаниях элементов, формируемых на основе параметризированного модуля *CORDIC* версии v3.0 для выполнения операции извлечения квадратного корня, выглядит следующим образом:

- x_in[N:0] — входная шина данных с разрядностью N+1, на которую поступает двоичный код, определяющий значение подкоренного числа;
- x_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты извлечения квадратного кор-

ня из значений, представленных на входной шине данных;

- clk, ce, aclr, sclr, nd и rdy — имеют то же назначение, что и в описаниях элементов, выполняющих вычисления значений тригонометрических и гиперболических функций, которые были рассмотрены в предыдущей части статьи.

Пример описания элемента, предназначенного для выполнения операции извлечения квадратного корня, сформированного на основе параметризированного модуля CORDIC версии v3.0

Иллюстрацией результата применения параметризированного модуля *CORDIC* версии v3.0 для подготовки описаний устройств, выполняющих операции извлечения квадратного корня из значений, представленных на входной шине данных, является VHDL-описание элемента *square_root_cordic*, текст которого имеет следующий вид:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY square_root_cordic IS
        port (
                x_in: IN std_logic_VECTOR(47 downto 0);
                nd: IN std_logic;
                x_out: OUT std_logic_VECTOR(47 downto 0);
                rdy: OUT std_logic;
                clk: IN std_logic;
                ce: IN std_logic;
                aclr: IN std_logic;
                sclr: IN std_logic
                );
END square_root_cordic;
--
ARCHITECTURE square_root_cordic_a OF square_root_cordic IS
-- synthesis translate_off
component wrapped_square_root_cordic
        port (
                x_in: IN std_logic_VECTOR(47 downto 0);
                nd: IN std_logic;
                x_out: OUT std_logic_VECTOR(47 downto 0);
                rdy: OUT std_logic;
                clk: IN std_logic;
                ce: IN std_logic;
                aclr: IN std_logic;
                sclr: IN std_logic
                );
end component;
--
-- Configuration specification
    for all : wrapped_square_root_cordic use entity
XilinxCoreLib.cordic_v3_0(behavioral)
    generic map(
                c_has_clk => 1,
                c_has_x_out => 1,
                c_has_y_in => 0,
                c_reg_inputs => 1,
                c_architecture => 2,
                c_input_width => 48,
                c_iterations => 0,
                c_precision => 0,
                c_has_rdy => 1,
                c_has_sclr => 1,
                c_has_nd => 1,
                c_scale_comp => 0,
                c_enable_rlocs => 1,
                c_has_phase_in => 0,
                c_has_rfd => 0,
                c_cordic_function => 6,
                c_has_ce => 1,
                c_mif_file_prefix => «cor1»,
                c_round_mode => 1,
                c_has_aclr => 1,
                c_sync_enable => 1,
                c_has_y_out => 0,
                c_data_format => 1,
                c_reg_outputs => 1,
                c_coarse_rotate => 0,
                c_phase_format => 0,
                c_has_phase_out => 0,
                c_has_x_in => 1,
                c_pipeline_mode => -1,
                c_output_width => 48
                );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_square_root_cordic
    port map (
                x_in => x_in,
                nd => nd,
                x_out => x_out,
                rdy => rdy,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
-- synthesis translate_on
--
END square_root_cordic_a;
```

В сформированном элементе *square_root_cordic* используются 48-разрядные входная и выходная шины. При этом входные данные и результаты вычисления квадратного корня представлены в виде дробных чисел без знака. В состав архитектуры элемента *square_root_cordic* включены входные и выходные регистры. Выбор оптимального варианта конвейерной организации выполнения операций в этом элементе обеспечивает достаточно высокую производительность вычислений квадратного корня без привлечения дополнительных таблиц преобразования LUT ПЛИС. В элементе *square_root_cordic* задействованы входы сигналов синхронного и асинхронного сброса, а также вход сигнала разрешения синхронизации. Сигнал на входе синхронного сброса имеет более высокий приоритет, чем сигнал на входе разрешения синхронизации. В состав интерфейса рассматриваемого элемента также включен вход сигнала New Data, подтверждающего достоверность входных данных, и выход сигнала Ready, сообщающего о готовности достоверных результатов вычислений квадратного корня.

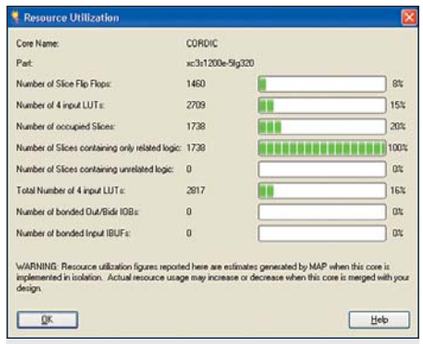

Рис. 127. Вид информационной панели, содержащей сведения о ресурсах ПЛИС, используемых для реализации элемента square_root_cordic

При использовании элемента *square_root_cordic* в качестве одного из компонентов в составе структурного или смешанного описания проектируемого устройства необходимо дополнить раздел декларации формируемого описания следующими выражениями:

```
component square_root_cordic
        port (
                x_in: IN std_logic_VECTOR(47 downto 0);
                nd: IN std_logic;
                x_out: OUT std_logic_VECTOR(47 downto 0);
                rdy: OUT std_logic;
                clk: IN std_logic;
                ce: IN std_logic;
                aclr: IN std_logic;
                sclr: IN std_logic
                );
end component;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of square_root_cordic: component is true;
```

Описание каждого экземпляра компонента *square_root_cordic* в составе блока определения архитектуры разрабатываемого устройства выполняется с помощью оператора, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_элемента_square_root_cordic>:
square_root_cordic
        port map (
                x_in => x_in,
                nd => nd,
                x_out => x_out,
                rdy => rdy,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
```

Оценка объема различных ресурсов кристалла, необходимых для автономной реализации элемента *square_root_cordic*, приведена в информационной панели, вид которой представлен на рис. 127.

Продолжение следует