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

Подготовка описаний элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций методами CORDIC, средствами генератора параметризированных модулей CORE Generator

CORDIC (COordinate Rotational DIgital Computer) представляет собой совокупность эффективных методов поворота вектора на заданный угол, реализуемых с помощью операций сложения и сдвига. В состав генератора параметризированных модулей CORE Generator входит одноименное ядро, которое позволяет создавать описания элементов, выполняющих вычисления значений различных тригонометрических и гиперболических функций методами CORDIC. Эти элементы можно использовать в составе проектируемых устройств, реализуемых на основе ПЛИС следующих семейств: Spartan-II; Spartan-IIE; Spartan-3; Spartan-3E; Virtex; QPRO Virtex Rad-Hard; QPRO Virtex Hi-Rel; Virtex-E; QPRO Virtex-E Military; Virtex-II; Virtex-II PRO и Virtex-4.

Генератор параметризированных модулей CORE Generator поддерживает несколько версий ядра *CORDIC*. В настоящее время наиболее актуальной версией рассматриваемого параметризированного модуля является модификация CORDIC v3.0, которая характеризуется следующими особенностями:

- Широкий выбор вариантов выполняемой операции для формируемого элемента:
  1. Вычисление значений функций синуса и косинуса, соответствующих двоичному коду фазы, представленному на входной шине данных.
  2. Формирование значений арктангенса, соответствующих двоичным кодам синуса и косинуса, поступающим на входные шины данных.
  3. Вычисление значений функций гиперболического синуса и косинуса, соответствующих двоичному коду фазы, представленному на входной шине данных.
  4. Вычисление гиперболического арктангенса для пары значений гиперболического синуса и косинуса, присутствующих на входных шинах данных.
  5. Извлечение квадратного корня из значений входных данных.
  6. Поворот вектора на заданный угол.
  7. Преобразования прямоугольных координат вектора в полярные.
- Возможность генерации описаний элементов, предназначенных для вычисления значений одной из указанных выше функции методами CORDIC, разрядность входных и выходных шин которых можно выбирать в диапазоне от 8 до 48 разрядов.
- Поддержка четырех различных методов округления результатов вычислений.
- Возможность формирования описаний элементов, вычисляющих значения выбранной функции методами CORDIC, с последовательной архитектурой, позволяющей минимизировать объем используемых ресурсов ПЛИС.
- Поддержка генерации описаний элементов с параллельной архитектурой, обеспечивающей достижение минимальной задержки формирования результатов вычислений значений выбранной функции.
- Возможность автоматического или ручного выбора точности представления промежуточных результатов, получаемых при выполнении внутренних операций сдвига и сложения/вычитания в процессе вычисления значений целевой функции.
- Наличие опции автоматического определения необходимого количества итераций совокупности операций сдвига и сложения/вычитания в генерируемом элементе с возможностью явного указания требуемого значения данного параметра.
- Возможность применения в составе формируемых элементов блока «грубого» поворота вектора, расширяющего допустимый диапазон изменения аргумента (фазы) вычисляемых тригонометрических функций.
- Поддержка коррекции значений модуля вектора при выполнении операций поворота на заданный угол и преобразования координат.
- Возможность выборочного применения в формируемых элементах входных и выходных регистров.
- Использование (по выбору разработчика) сигналов подтверждения (квитирования) в генерируемых элементах, выполняющих вычисления значений выбранной функции методами CORDIC.
- Поддержка различных форматов представления входных и выходных данных (целочисленного формата без знака, дробного формата со знаком и без знака).
- Реализация формируемых элементов в виде синхронных устройств, тактируемых одним сигналом синхронизации.
- Возможность генерации описаний элементов, предназначенных для выполнения вычислений методами CORDIC, в форме макросов с относительным размещением Relationally Placed Macro (RPM).
- Наличие в создаваемых элементах входов синхронного и асинхронного сброса, а также входа разрешения синхронизации, использование которых определяется разработчиком.
- Возможность выбора различных вариантов конвейерной организации выполнения операций в создаваемых элементах.
- Применение технологии Xilinx Smart-IP, обеспечивающей наилучшую реализацию генерируемого элемента в выбранном кристалле.

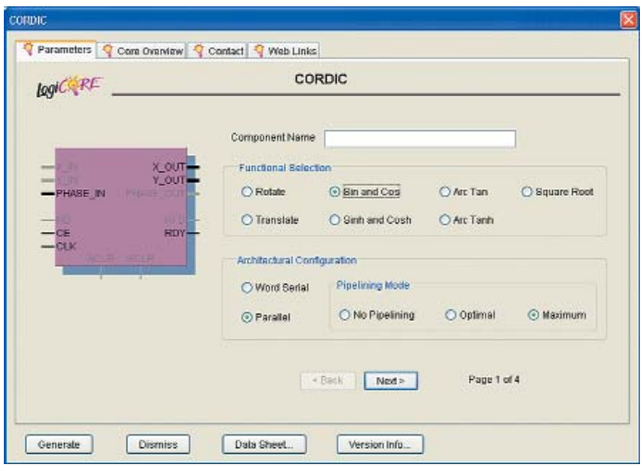

Рис. 105. Стартовая диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0 (страница Parameters)

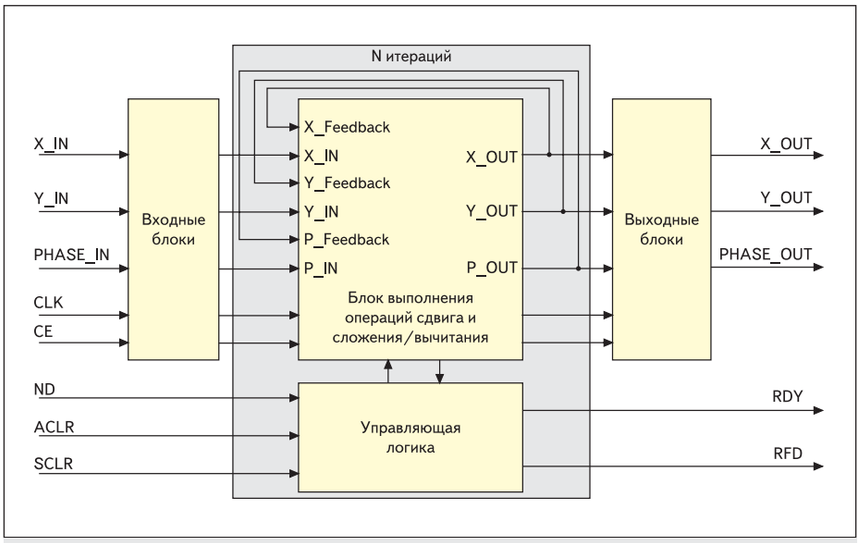

Рис. 106. Структура элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций методами CORDIC, выполняемых на основе архитектуры последовательного типа

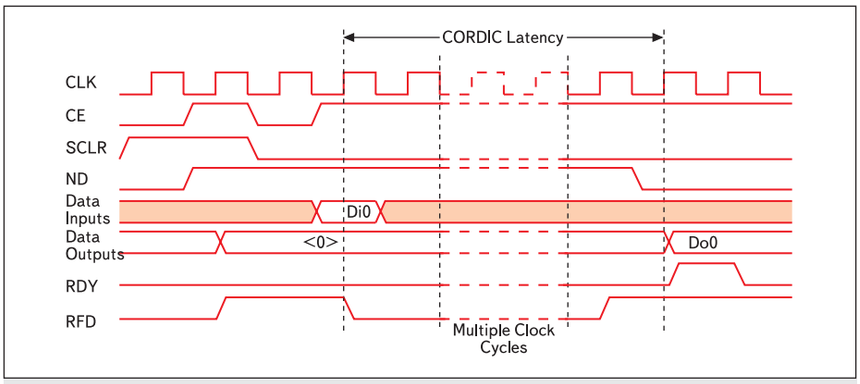

Рис. 107. Временные диаграммы, поясняющие функционирование элементов, выполняющих операции вычисления значений тригонометрических и гиперболических функций методами CORDIC, реализуемых на основе архитектуры последовательного типа

«Мастер» настройки параметров ядра *CORDIC* версии v3.0 содержит четыре диалоговые панели. Вид страницы *Parameters* стартовой диалоговой панели этого «мастера» приведен на рис. 105.

Данная диалоговая панель используется для выбора требуемого варианта выполняемой операции и типа архитектуры формируемого элемента, а также для определения его названия — *Component Name*. Выбор операции, которую должен выполнять создаваемый элемент, осуществляется с помощью группы кнопок с зависимой фиксацией, расположенных во встроенной панели *Functional Selection* (рис. 105). Чтобы сгенерировать описание элемента, предназначенного для вычисления значений тригонометрических функций синуса и косинуса методами CORDIC, соответствующих двоичному коду аргумента (фазы), представленному на входной шине PHASE_IN, следует зафиксировать в нажатом состоянии кнопку *Sin and Cos*. Если необходимо подготовить описание элемента, вычисляющего значения арктангенса, соответствующие двоичным кодам синуса и косинуса, которые представлены на входных шинах Y_IN и X_IN, то нужно нажать кнопку *Arc Tan*. Для формирования описания элемента, выполняющего операции вычисления значений функций гиперболического синуса и косинуса, необходимо нажать кнопку *Sinh and Cosh*. Генерация описания элемента, предназначенного для вычисления значений функции гиперболического арктангенса, производится при нажатой кнопке *Arc Tanh*.

Процесс вычисления значений различных функций с точностью до N-го разряда, выполняемый методами CORDIC, требует приблизительно N итераций совокупности операций сдвига и сложения/вычитания. В параметризированном модуле *CORDIC* версии v3.0 предусмотрено два способа организации выполнения этих итераций, которые реализуются в виде соответствующих вариантов архитектуры создаваемых элементов: последовательного и параллельного. При использовании последовательного типа архитектуры создаваемого элемента для осуществления всех итераций поочередно применяется один аппаратный блок, выполняющий операции сдвига и сложения/вычитания. Такое решение позволяет минимизировать объем аппаратных ресурсов кристалла, требуемых для реализации формируемого элемента, за счет снижения его быстродействия. Структура элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций методами CORDIC, выполняемых на основе архитектуры последовательного типа, представлена на рис. 106.

Временные диаграммы, поясняющие функционирование элементов, выполняющих операции вычисления значений тригонометрических и гиперболических функций методами CORDIC, реализуемых на основе архитектуры последовательного типа приведены на рис. 107.

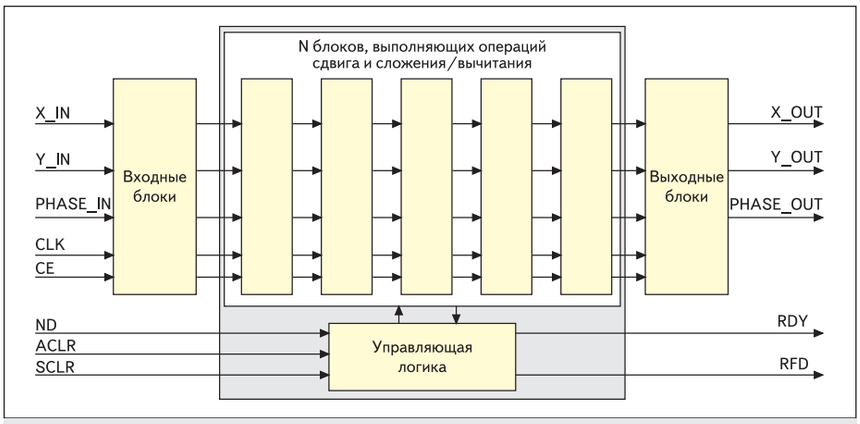

Рис. 108. Структура элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций методами CORDIC, выполняемых на основе архитектуры параллельного типа

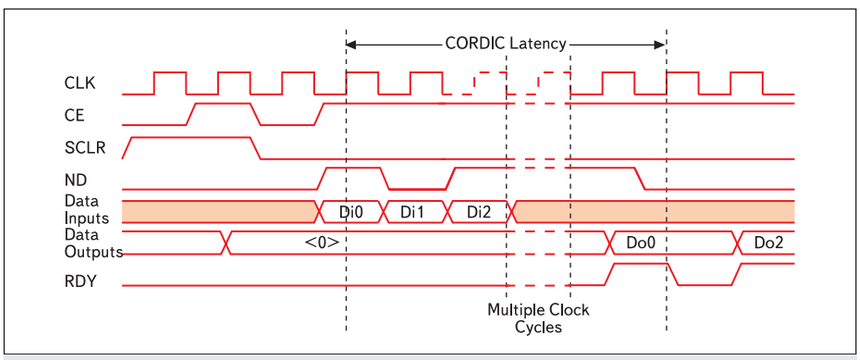

Рис. 109. Временные диаграммы, демонстрирующие работу элементов, выполняющих операции вычисления значений тригонометрических и гиперболических функций методами CORDIC, реализуемых на основе архитектуры параллельного типа

При выборе параллельного типа архитектуры генерируемого элемента для реализации каждой итерации процесса вычислений выделяют отдельный аппаратный блок, выполняющий операции сдвига и сложения/вычитания. Применение данного типа архитектуры обеспечивает повышение быстродействия формируемых элементов, но приводит к существенному увеличению объема аппаратных ресурсов ПЛИС, необходимых для их реализации. На рис. 108 изображена структура элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций методами CORDIC, выполняемых на основе архитектуры параллельного типа.

Временные диаграммы, демонстрирующие работу элементов, выполняющих операции вычисления значений тригонометрических и гиперболических функций методами CORDIC, реализуемых на основе архитектуры параллельного типа, показаны на рис. 109.

Тип архитектуры формируемого элемента указывается с помощью группы кнопок с зависимой фиксацией, которые представлены во встроенной панели *Architectural Configuration* (рис. 105). Для генерации описания элемента с последовательной архитектурой следует нажать кнопку *Word Serial*. Чтобы подготовить описание элемента с параллельной архитектурой, нужно зафиксировать в нажатом положении кнопку *Parallel*.

Группа кнопок с зависимой фиксацией *Pipelining Mode*, находящаяся в этой же встроенной панели, позволяет выбрать параметры конвейерной организации выполнения вычислений в создаваемом элементе. При нажатой кнопке *No Pipelining* конвейерная организация в формируемом элементе не применяется. Если в нажатом положении находится кнопка *Optimal*, то в составе архитектуры генерируемого элемента будет задействовано максимально возможное число ступеней конвейерной организации выполнения вычислений, не требующее для их реализации дополнительных таблиц преобразования Look Up Table (LUT). Чтобы подготовить описание элемента, в котором конвейерные регистры установлены на выходе каждого блока, выполняющего операции сдвига и сложения/вычитания, следует переключить в нажатое состояние кнопку *Maximum*.

Вторая диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0 позволяет выбрать формат представления значений аргумента (фазы) вычисляемых тригонометрических и гиперболических функций, а также указать дополнительные входы и выходы управляющих сигналов в создаваемых элементах. На рис. 110 приведен вид страницы *Parameters* этой диалоговой панели.

Определение формата представления значений аргумента (фазы) тригонометрических и гиперболических функций в генерируемом элементе осуществляется с помощью двух кнопок с зависимой фиксацией, которые расположены во встроенной панели *Phase Format* (рис. 110). Двоичный код фазы на входных и выходных шинах создаваемых элементов может восприниматься как значение, выраженное в радианах или нормированных единицах (π*радианах). Чтобы сформировать описания элементов, предназначенных для вычисления значений тригонометрических и гиперболических функций, в которых

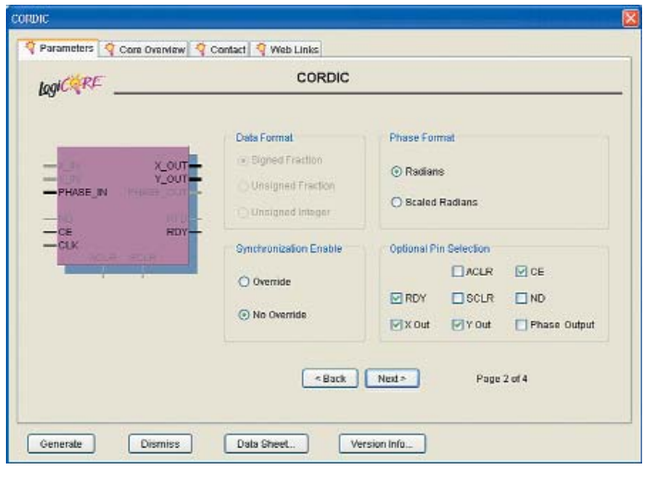

Рис. 110. Вид страницы Parameters второй диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0

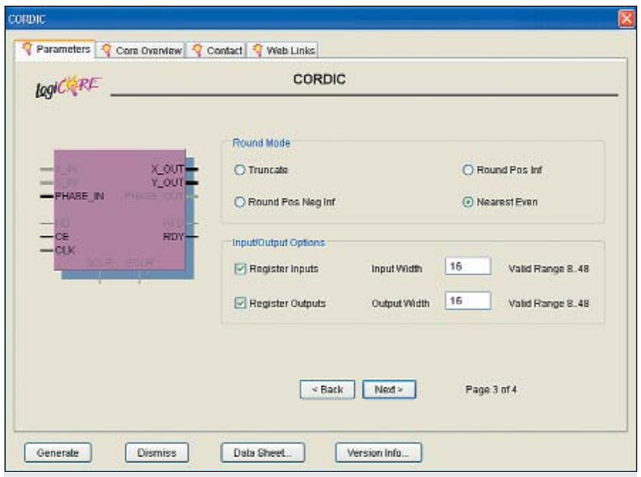

Рис. 111. Вид страницы Parameters третьей диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0

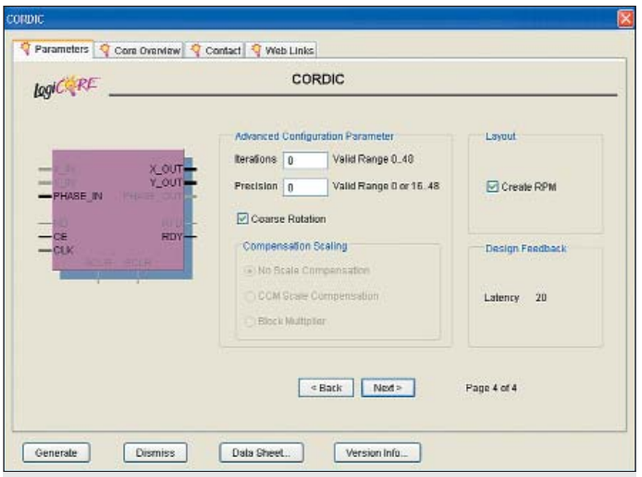

Рис. 112. Вид страницы Parameters заключительной диалоговой панели «мастера» настройки параметров ядра *CORDIC* версии v3.0

входные или выходные значения фазы представляются в радианах, нужно зафиксировать в нажатом состоянии кнопку *Radians*. При нажатой кнопке *Scaled Radians* двоичный код на входных шинах PHASE_IN или выходных шинах PHASE_OUT генерируемых элементов будет соответствовать нормированным единицам измерения фазы.

Для выбора дополнительных входов и выходов управляющих сигналов в формируемых элементах следует воспользоваться индикаторами состояния, которые представлены во встроенной панели *Optional Pin Selection* (рис. 110). Чтобы включить в состав интерфейса генерируемого элемента вход сигнала New Data, активный уровень которого указывает на достоверность новых значений данных, представленных на входных шинах, нужно установить в состояние «Включено» индикатор *ND*. Если в создаваемом элементе предполагается использование выходного сигнала Ready, информирующего о готовности достоверного результата вычислений на соответствующих выходных шинах, то следует установить во включенное состояние индикатор *RDY*. Чтобы задействовать входы сигналов асинхронного и/или синхронного сброса, необходимо перевести в состояние «Включено» индикаторы *ACLR* и *SCLR* соответственно. Для использования в разрабатываемом элементе входа сигнала разрешения синхронизации Clock Enable, нужно перевести во включенное состояние индикатор *CE*. В том случае, когда в создаваемом элементе одновременно присутствуют входы синхронного сброса и разрешения синхронизации, необходимо определить соотношение приоритетов этих сигналов. Для этой цели следует воспользоваться двумя кнопками с зависимой фиксацией, которые находятся во встроенной панели *Synchronization Enable*. Когда в нажатом состоянии находится кнопка *Override*, сигнал на входе разрешения синхронизации будет иметь более высокий приоритет по сравнению с сигналом на входе синхронного сброса. Установка противоположного соотношения приоритетов указанных сигналов в формируемом элементе осуществляется переключением в нажатое положение кнопки *No Override*.

Третья диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0, вид страницы *Parameters* которой представлен на рис. 111, предназначена для выбора метода округления результатов вычислений, а также для определения разрядности входных и выходных шин в генерируемом элементе. В этой же панели задаются значения параметров, определяющих присутствие входных и выходных регистров в составе формируемых элементов.

Параметризированный модуль *CORDIC* версии v3.0 предоставляет разработчику возможность использования в создаваемых элементах одного из четырех поддерживаемых методов округления результатов вычислений. Выбор требуемого метода производится с помощью группы кнопок с зависимой фиксацией, которые расположены во встроенной панели *Round Mode* (рис. 111). Если в нажатом состоянии зафиксирована кнопка Truncate, то выполняется усечение полученных результатов вычислений, при котором младшие неиспользуемые биты значений отбрасываются, а оставшиеся разряды поступают на выходные шины формируемого элемента в неизменном виде. При нажатой кнопке *Round Pos Inf* в генерируемом элементе применяются стандартные правила округления чисел. Когда в нажатом положении находится кнопка *Round Pos Neg Inf*, округление положительных значений осуществляется так же, как и при использовании метода *Round Pos Inf*. Отрицательные результаты вычислений, значения дробной части которых по модулю больше или равны 0,5, округляются до ближайшего меньшего отрицательного числа. Если нажата кнопка *Nearest Even*, то результаты вычислений в формируемом элементе будут округляться до ближайшего четного значения.

Для включения входных и выходных регистров в состав элементов, генерируемых с помощью параметризированного модуля *CORDIC* версии v3.0, нужно воспользоваться индикаторами состояния, которые расположены во встроенной панели *Input/Output Options* (рис. 111). Чтобы добавить входные регистры в состав структуры формируемого элемента, предназначенного для вычисления значений тригонометрических и гиперболических функций методами CORDIC, нужно перевести в состояние «Включено» индикатор *Register Inputs*. При этом будут задействованы регистры в цепях входных сигналов X_IN, Y_IN, PHASE_IN и ND. Если в создаваемом элементе необходимы выходные регистры, то следует установить во включенное состояние индикатор *Register Outputs*. В этом случае будут установлены регистры в цепях выходных сигналов X_OUT, Y_OUT, PHASE_OUT, RFD и RDY.

Определение разрядности входных и выходных шин формируемого элемента осуществляется с помощью полей редактирования, которые также находятся во встроенной панели *Input/Output Options* (рис. 111). Количество разрядов во входных шинах указывают в поле редактирования *Input Width*, а в выходных — в поле редактирования *Output Width*.

Заключительная диалоговая панель «мастера» настройки параметров ядра *CORDIC* версии v3.0 предназначена для определения значений дополнительных параметров генерируемых элементов. Вид страницы *Parameters* этой диалоговой панели изображен на рис. 112.

Необходимое количество итераций совокупности операций сдвига и сложения/вычитания, используемых в создаваемом элементе для вычисления значений выбранной функции, определяют автоматически, исходя из заданной разрядности выходных данных. Кроме того, разработчик может указать требуемое

количество итераций, воспользовавшись полем редактирования *Iterations*, которое расположено во встроенной панели *Advanced Configuration Parameter* (рис. 112). Нулевое значение, представленное по умолчанию в этом поле редактирования, соответствует автоматическому выбору количества итераций в генерируемом элементе. Точность промежуточных результатов вычислений (операций сдвига и сложения/вычитания) в элементах, формируемых на основе параметризированного модуля *CORDIC* версии v3.0, устанавливается автоматически в соответствии с выбранной разрядностью выходных данных и используемым количеством итераций. Пользователь может задать точность представления промежуточных результатов вычислений, используя поле редактирования *Precision*, которое также находится во встроенной панели *Advanced Configuration Parameter*. По умолчанию в данном поле редактирования содержится нулевое значение, которое соответствует режиму автоматического определения точности промежуточных результатов вычислений в создаваемом элементе.

Индикатор состояния *Coarse Rotation* позволяет включить в состав архитектуры формируемого элемента, предназначенного для выполнения вычислений методами CORDIC, блок «грубого» поворота вектора. Применение данного блока позволяет расширить допустимый диапазон изменения значений фазы при вычислении тригонометрических функций до интервала от $-\pi$ до $+\pi$. При отсутствии блока «грубого» поворота вектора формируемый элемент способен вычислять значения тригонометрических функций только в диапазоне от $-\pi/4$ до $+\pi/4$. В элементах, создаваемых для вычисления значений гиперболических функций, указанный блок не используется. Поэтому при подготовке описаний устройств, предназначенных для вычисления значений гиперболического синуса, косинуса и арктангенса, индикатор состояния *Coarse Rotation* находится в недоступном состоянии.

Состояние индикатора *Create RPM*, расположенного во встроенной панели *Layout* (рис. 112), определяет форму представления генерируемого описания для последующей реализации в ПЛИС. Если данный индикатор установлен в состояние «Включено», то описание разрабатываемого элемента будет представлено в форме макроса с относительным размещением RPM.

Сведения о величине задержки формирования результатов вычислений на выходных шинах разрабатываемого элемента отображаются во встроенной панели *Design Feedback* в виде значения параметра *Latency*. Значение задержки, которое соответствует указанным параметрам конфигурации этого элемента, выражается в количестве периодов тактового сигнала.

Система условных обозначений интерфейсных портов в описаниях элементов, создаваемых с помощью параметризированного модуля *CORDIC* версии v3.0, выглядит следующим образом:

- phase_in[N:0] — входная шина данных с разрядностью N+1, на которую подается двоичный код значения аргумента (фазы) вычисляемых тригонометрических и гиперболических функций или угла поворота вектора;
- x_in[N:0] — входная шина данных с разрядностью N+1, на которую поступает двоичный код значения косинусной составляющей аргумента вычисляемой функции;
- y_in[N:0] — входная шина данных с разрядностью N+1, на которую подается двоичный код значения синусной составляющей аргумента вычисляемой функции;
- x_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления функции косинуса или гиперболического косинуса, соответствующие текущему входному двоичному коду аргумента (фазы);
- y_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления функции синуса или гиперболического синуса, соответствующие текущему входному двоичному значению аргумента (фазы);
- phase_out[M:0] — выходная шина данных с разрядностью M+1, на которой отображаются результаты вычисления значений арктангенса или гиперболического арктангенса;
- clk — вход тактового сигнала;
- ce — вход сигнала разрешения синхронизации;
- aclr — вход сигнала асинхронного сброса выходного регистра;
- sclr — вход сигнала синхронного сброса выходного регистра;
- nd — вход сигнала New Data, подтверждающего достоверность сигналов на входных шинах данных, определяющих значения аргумента вычисляемой тригонометрической или гиперболической функции;
- rfd — выход сигнала Ready for Data (RFD), информирующего о готовности вычисления значения выбранной тригонометрической функции для новых входных данных (аргумента), представленных на соответствующих шинах;
- rdy — выход сигнала Ready, сообщающего о готовности достоверных результатов вычислений, представленных на соответствующих выходных шинах.

Пример описания элемента, предназначенного для вычисления значений функций синуса и косинуса методами CORDIC, сформированного средствами генератора параметризированных модулей CORE Generator

В качестве примера использования параметризированного модуля *CORDIC* версии v3.0 в настоящем разделе приводится текст VHDL-описания элемента *sine_and_cosine_cordic*, который выполняет операции вычисления значений функций синуса и косинуса. Аргумент (фаза) вычисляемых функций может принимать любые значения в диапазоне от $-\pi$ до $+\pi$. В данном элементе значения аргумента указанных тригонометрических функций и результатов вычислений представлены в виде 32-разрядного двоичного кода:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY sine_and_cosine_cordic IS
   port (
         phase_in: IN std_logic_VECTOR(31 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(31 downto 0);
         y_out: OUT std_logic_VECTOR(31 downto 0);
         phase_out: OUT std_logic_VECTOR(31 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 sine_and_cosine_cordic;
--
ARCHITECTURE sine_and_cosine_cordic_a OF
sine_and_cosine_cordic IS
-- synthesis translate_off
component wrapped_sine_and_cosine_cordic
   port (
         phase_in: IN std_logic_VECTOR(31 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(31 downto 0);
         y_out: OUT std_logic_VECTOR(31 downto 0);
         phase_out: OUT std_logic_VECTOR(31 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_sine_and_cosine_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 => 1,
                  c_input_width => 32,
                  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 => 1,
                  c_has_rfd => 1,
                  c_cordic_function => 2,
                  c_has_ce => 1,
                  c_mif_file_prefix => «cor1»,
                  c_round_mode => 0,
                  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 => 0,
                  c_pipeline_mode => -2,
                  c_output_width => 32
                  );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_sine_and_cosine_cordic
         port map (
                  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 sine_and_cosine_cordic_a;
```

Структура элемента *sine_and_cosine_cordic* соответствует последовательному типу архитектуры. В рассматриваемом элементе присутствуют входные и выходные регистры, а также максимально возможное количество конвейерных регистров. Элемент *sine_and_cosine_cordic* позволяет использовать в процессе вычислений значений тригонометрических функций всю совокупность сигналов подтверждения, поддерживаемых параметризированным модулем *CORDIC* версии v3.0. В этом элементе задействованы также входы разрешения синхронизации, асинхронного и синхронного сброса. При этом сигнал на входе разрешения синхронизации имеет более низкий приоритет по сравнению с сигналом на входе синхронного сброса.

Декларация компонента *sine_and_cosine_cordic* в описании разрабатываемого устройства осуществляется с помощью представленной далее последовательности выражений:

```
component sine_and_cosine_cordic
   port (
         phase_in: IN std_logic_VECTOR(31 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(31 downto 0);
         y_out: OUT std_logic_VECTOR(31 downto 0);
         phase_out: OUT std_logic_VECTOR(31 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 sine_and_cosine_cordic: component is true;
```

Для описания конкретных экземпляров компонента *sine_and_cosine_cordic* в составе архитектуры проектируемого устройства следует использовать оператор, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_элемента_sine_and_cosine_cordic>:
sine_and_cosine_cordic
         port map (
                    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. 113 приведена информационная панель, содержащая подробные сведения

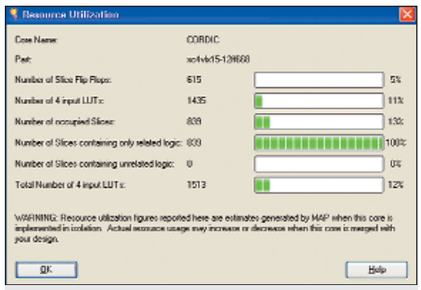

Рис. 113. Вид информационной панели, содержащей сведения о ресурсах ПЛИС, используемых для реализации элемента sine_and_cosine_cordic

о количестве различных ресурсов ПЛИС, которые необходимы для автономной реализации элемента *sine_and_cosine_cordic*.

Пример описания элемента, предназначенного для вычисления значений функции арктангенса методами CORDIC, сформированного средствами генератора параметризированных модулей CORE Generator

Примером устройства, сформированного на основе параметризированного модуля *CORDIC* версии v3.0 для вычисления значений функции арктангенса, является элемент *arc_tan_cordic*, текст описания которого имеет следующий вид:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY arc_tan_cordic IS
   port (
         x_in: IN std_logic_VECTOR(19 downto 0);
         y_in: IN std_logic_VECTOR(19 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(19 downto 0);
         y_out: OUT std_logic_VECTOR(19 downto 0);
         phase_out: OUT std_logic_VECTOR(19 downto 0);
         rdy: OUT std_logic;
         clk: IN std_logic;
         ce: IN std_logic;
         aclr: IN std_logic;
         sclr: IN std_logic
         );
END arc_tan_cordic;
--
ARCHITECTURE arc_tan_cordic_a OF arc_tan_cordic IS
-- synthesis translate_off
component wrapped_arc_tan_cordic
   port (
         x_in: IN std_logic_VECTOR(19 downto 0);
         y_in: IN std_logic_VECTOR(19 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(19 downto 0);
         y_out: OUT std_logic_VECTOR(19 downto 0);
         phase_out: OUT std_logic_VECTOR(19 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_arc_tan_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 => 2,
                    c_input_width => 20,
                    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 => 3,
                    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 => 20
                    );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_arc_tan_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,
                    clk => clk,
                    ce => ce,
                    aclr => aclr,
                    sclr => sclr
                    );
-- synthesis translate_on
--
END arc_tan_cordic_a;
```

В сформированном элементе используются 20-разрядные входные и выходные шины данных. Элемент *arc_tan_cordic* имеет архитектуру параллельного типа, в состав которой включены входные и выходные регистры, а также блок «грубого» поворота вектора. В этом элементе используется оптимальный вариант конвейерной организации выполнения операций сдвига и сложения/вычитания. В состав интерфейса рассматриваемого элемента входят те же входы и выходы сигналов управления и подтверждения, что и в элементе *sine_and_cosine_cordic*, который был представлен в предыдущем разделе. Сигнал на входе синхронного сброса в элементе *arc_tan_cordic* имеет более низкий приоритет, чем сигнал на входе разрешения синхронизации.

При использовании элемента *arc_tan_cordic* в качестве одного из компонентов в составе описания разрабатываемого устройства необходимо поместить в раздел декларации следующую конструкцию:

```
component arc_tan_cordic
   port (
         x_in: IN std_logic_VECTOR(19 downto 0);
         y_in: IN std_logic_VECTOR(19 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(19 downto 0);
         y_out: OUT std_logic_VECTOR(19 downto 0);
         phase_out: OUT std_logic_VECTOR(19 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 arc_tan_cordic: component is true;
```

Конкретные экземпляры компонента *arc_tan_cordic* описываются с помощью оператора, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_элемента_arc_tan_cordic>:
arc_tan_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,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
```

Количество конфигурируемых логических блоков CLB, таблиц преобразования LUT и секций Slices, используемых для автономной реализации элемента *arc_tan_cordic*, указано в информационной панели, вид которой показан на рис. 114.

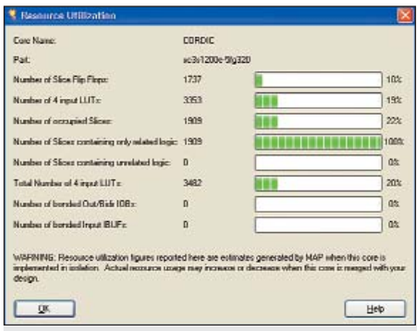

Рис. 114. Вид информационной панели, содержащей сведения о ресурсах кристалла, используемых для реализации элемента arc_tan_cordic

Пример описания элемента, предназначенного для вычисления значений функций гиперболического синуса и косинуса методами CORDIC, сформированного средствами генератора параметризированных модулей CORE Generator

Результат применения параметризированного модуля *CORDIC* версии v3.0 для подготовки описаний устройств, выполняющих операции вычисления значений функций гиперболического синуса и косинуса, продемонстрирован в настоящем разделе на примере элемента *sinh_and_cosh_cordic*. Значения аргумента указанных гиперболических функций и результаты вычислений в этом элементе представлены в виде 36-разрядного двоичного кода. Сформированный текст VHDL-описания элемента *sinh_and_cosh_cordic* имеет следующий вид:

```
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY sinh_and_cosh_cordic IS
  port (
        phase_in: IN std_logic_VECTOR(35 downto 0);
        nd: IN std_logic;
        x_out: OUT std_logic_VECTOR(35 downto 0);
        y_out: OUT std_logic_VECTOR(35 downto 0);
        phase_out: OUT std_logic_VECTOR(35 downto 0);
        rdy: OUT std_logic;
        clk: IN std_logic;
        ce: IN std_logic;
        aclr: IN std_logic;
        sclr: IN std_logic
        );
END sinh_and_cosh_cordic;
--
ARCHITECTURE sinh_and_cosh_cordic_a OF sinh_and_cosh_cordic IS
-- synthesis translate_off
component wrapped_sinh_and_cosh_cordic
  port (
        phase_in: IN std_logic_VECTOR(35 downto 0);
        nd: IN std_logic;
        x_out: OUT std_logic_VECTOR(35 downto 0);
        y_out: OUT std_logic_VECTOR(35 downto 0);
        phase_out: OUT std_logic_VECTOR(35 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_sinh_and_cosh_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 => 36,
                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 => 1,
                c_has_rfd => 0,
                c_cordic_function => 4,
                c_has_ce => 1,
                c_mif_file_prefix => «cor1»,
                c_round_mode => 3,
                c_has_aclr => 1,
                c_sync_enable => 0,
                c_has_y_out => 1,
                c_data_format => 0,
                c_reg_outputs => 1,
                c_coarse_rotate => 0,
                c_phase_format => 0,
                c_has_phase_out => 1,
                c_has_x_in => 0,
                c_pipeline_mode => -2,
                c_output_width => 36
                );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_sinh_and_cosh_cordic
        port map (
                phase_in => phase_in,
                nd => nd,
                x_out => x_out,
                y_out => y_out,
                phase_out => phase_out,
                rdy => rdy,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
-- synthesis translate_on
--
END sinh_and_cosh_cordic_a;
```

Элемент *sinh_and_cosh_cordic* построен на основе архитектуры параллельного типа. Применение в его составе входных и выходных регистров в сочетании с максимально возможным количеством конвейерных регистров обеспечивает высокий уровень производительности этого элемента. В состав интерфейса рассматриваемого элемента включены те же входы и выходы сигналов управления и подтверждения, что и в элементе *sine_and_cosine_cordic*, но при этом сигнал на входе разрешения синхронизации имеет более высокий приоритет по сравнению с сигналом синхронного сброса.

Блок декларации компонента *sinh_and_cosh_cordic* в составе описания разрабатываемого устройства включает приведенную далее последовательность выражений:

```
component sinh_and_cosh_cordic
  port (
        phase_in: IN std_logic_VECTOR(35 downto 0);
        nd: IN std_logic;
        x_out: OUT std_logic_VECTOR(35 downto 0);
        y_out: OUT std_logic_VECTOR(35 downto 0);
        phase_out: OUT std_logic_VECTOR(35 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 sinh_and_cosh_cordic: component is true;
```

Для создания конкретных экземпляров компонента *sinh_and_cosh_cordic* в составе структурного или смешанного описания архитектуры проектируемого устройства нужно использовать оператор, шаблон которого выглядит следующим образом:

```
<идентификатор_экземпляра_элемента_sinh_and_cosh_cordic>:
sinh_and_cosh_cordic
        port map (
                phase_in => phase_in,
                nd => nd,
                x_out => x_out,
                y_out => y_out,
                phase_out => phase_out,
                rdy => rdy,
                clk => clk,
                ce => ce,
                aclr => aclr,
                sclr => sclr
                );
```

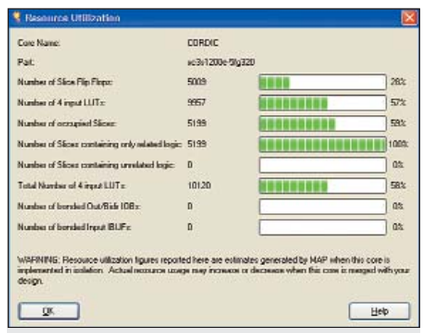

Рис. 115. Вид информационной панели, содержащей сведения о ресурсах ПЛИС, используемых для реализации элемента sinh_and_cosh_cordic

Объем ресурсов кристалла, необходимых для реализации элемента *sinh_and_cosh_cordic*, указан в информационной панели, вид которой представлен на рис. 115.

Пример описания элемента, предназначенного для вычисления значений функции гиперболического арктангенса методами CORDIC, сформированного средствами генератора параметризированных модулей CORE Generator

В качестве примера описания устройства, предназначенного для вычисления значений функции гиперболического арктангенса методами CORDIC, сформированного средствами генератора параметризированных модулей CORE Generator, в настоящем разделе приводится VHDL-описание элемента *arc_tanh_cordic*. В этом элементе применяется архитектура последовательного типа в сочетании с оптимальным вариантом конвейерной организации выполнения операций сдвига и сложения/вычитания. Значения входных и выходных данных в элементе *arc_tanh_cordic* представлены в виде 24-разрядного двоичного кода:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY arc_tanh_cordic IS
   port (
         x_in: IN std_logic_VECTOR(23 downto 0);
         y_in: IN std_logic_VECTOR(23 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(23 downto 0);
         y_out: OUT std_logic_VECTOR(23 downto 0);
         phase_out: OUT std_logic_VECTOR(23 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 arc_tanh_cordic;
--
ARCHITECTURE arc_tanh_cordic_a OF arc_tanh_cordic IS
-- synthesis translate_off
component wrapped_arc_tanh_cordic
   port (
         x_in: IN std_logic_VECTOR(23 downto 0);
         y_in: IN std_logic_VECTOR(23 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(23 downto 0);
         y_out: OUT std_logic_VECTOR(23 downto 0);
         phase_out: OUT std_logic_VECTOR(23 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_arc_tanh_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 => 24,
                  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 => 1,
                  c_cordic_function => 5,
                  c_has_ce => 1,
                  c_mif_file_prefix => «cor1»,
                  c_round_mode => 3,
                  c_has_aclr => 1,
                  c_sync_enable => 1,
                  c_has_y_out => 1,
                  c_data_format => 0,
                  c_reg_outputs => 1,
                  c_coarse_rotate => 0,
                  c_phase_format => 0,
                  c_has_phase_out => 1,
                  c_has_x_in => 1,
                  c_pipeline_mode => -1,
                  c_output_width => 24
                  );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_arc_tanh_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 arc_tanh_cordic_a;
```

В сформированном элементе присутствует та же совокупность входов и выходов сигналов управления и подтверждения, что и в элементе *sine_and_cosine_cordic*.

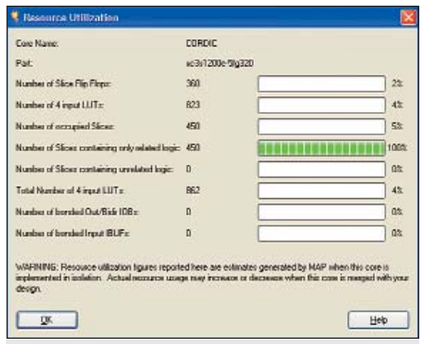

Рис. 116. Вид информационной панели, содержащей сведения о ресурсах кристалла, применяемых для автономной реализации элемента arc_tanh_cordic

При включении элемента *arc_tanh_cordic* в качестве одного из компонентов в состав архитектуры проектируемого устройства необходимо добавить в раздел декларации формируемого описания следующую конструкцию:

```
component arc_tanh_cordic
   port (
         x_in: IN std_logic_VECTOR(23 downto 0);
         y_in: IN std_logic_VECTOR(23 downto 0);
         nd: IN std_logic;
         x_out: OUT std_logic_VECTOR(23 downto 0);
         y_out: OUT std_logic_VECTOR(23 downto 0);
         phase_out: OUT std_logic_VECTOR(23 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 arc_tanh_cordic: component is true;
```

Описание конкретного экземпляра компонента *arc_tanh_cordic* в составе блока определения архитектуры разрабатываемого устройства выполняется с помощью оператора, который имеет следующий формат:

```
<идентификатор_экземпляра_элемента_arc_tanh_cordic>:
arc_tanh_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
                  );
```

Оценка объема различных ресурсов кристалла, применяемых для автономной реализации элемента *arc_tanh_cordic*, представлена в информационной панели, вид которой показан на рис. 116.

Продолжение следует