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

Пример описания элемента запоминающего устройства FIFO с раздельными сигналами синхронизации портов записи и чтения данных, сформированного с помощью параметризированного модуля FIFO Generator версии v4.1 для последующей реализации на основе блочной памяти ПЛИС Block SelectRAM

Результат использования параметризированного модуля FIFO Generator версии v4.1 для подготовки описаний элементов памяти, функционирующих по принципу «первым вошел–первым вышел», с раздельными сигналами синхронизации портов записи и чтения данных, которые предназначены для последующей реализации на основе модулей блочной памяти ПЛИС Block SelectRAM, демонстрируется в настоящем разделе на примере запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram*. Данный элемент памяти обладает информационной емкостью 32 кбит. При этом входной порт (порт записи данных) имеет организацию 4096 слов×8 разрядов. Разрядность выходного порта (порта чтения данных) запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram* составляет 32 бита. Сформированный элемент FIFO-памяти может использоваться для преобразования слова данных, передаваемого в виде последовательности из четырех байт, в 32-разрядный параллельный двоичный код.

Текст описания сгенерированного запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram* выглядит следующим образом:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY fifo_generator_v4_1_independent_clocks_block_ram IS
   port (
         din: IN std_logic_VECTOR(7 downto 0);
         prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
         prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
         prog_full_thresh_assert: IN std_logic_VECTOR(11 downto 0);
         prog_full_thresh_negate: IN std_logic_VECTOR(11 downto 0);
         rd_clk: IN std_logic;
         rd_en: IN std_logic;
         rst: IN std_logic;
         wr_clk: IN std_logic;
         wr_en: IN std_logic;
         almost_empty: OUT std_logic;
         almost_full: OUT std_logic;
         dout: OUT std_logic_VECTOR(31 downto 0);
         empty: OUT std_logic;
         full: OUT std_logic;
         overflow: OUT std_logic;
         prog_empty: OUT std_logic;
         prog_full: OUT std_logic;
         valid: OUT std_logic;
         rd_data_count: OUT std_logic_VECTOR(10 downto 0);
         underflow: OUT std_logic;
         wr_ack: OUT std_logic;
         wr_data_count: OUT std_logic_VECTOR(12 downto 0)
         );
END fifo_generator_v4_1_independent_clocks_block_ram;
--
ARCHITECTURE fifo_generator_v4_1_independent_clocks_block_ram_a
OF fifo_generator_v4_1_independent_clocks_block_ram IS
-- synthesis translate_off
component wrapped_fifo_generator_v4_1_independent_clocks_block_ram
   port (
         din: IN std_logic_VECTOR(7 downto 0);
         prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
         prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
         prog_full_thresh_assert: IN std_logic_VECTOR(11 downto 0);
         prog_full_thresh_negate: IN std_logic_VECTOR(11 downto 0);
         rd_clk: IN std_logic;
         rd_en: IN std_logic;
         rst: IN std_logic;
         wr_clk: IN std_logic;
         wr_en: IN std_logic;
         almost_empty: OUT std_logic;
         almost_full: OUT std_logic;
         dout: OUT std_logic_VECTOR(31 downto 0);
         empty: OUT std_logic;
         full: OUT std_logic;
         overflow: OUT std_logic;
         prog_empty: OUT std_logic;
         prog_full: OUT std_logic;
         valid: OUT std_logic;
         rd_data_count: OUT std_logic_VECTOR(10 downto 0);
         underflow: OUT std_logic;
         wr_ack: OUT std_logic;
         wr_data_count: OUT std_logic_VECTOR(12 downto 0)
         );
end component;
--
-- Configuration specification
   for all : wrapped_fifo_generator_v4_1_independent_clocks_block_ram
use entity XilinxCoreLib.fifo_generator_v4_1(behavioral)
         generic map(
                     c_has_int_clk => 0,
                     c_rd_freq => 1,
                     c_wr_response_latency => 1,
                     c_has_srst => 0,
                     c_has_rd_data_count => 1,
                     c_din_width => 8,
                     c_has_wr_data_count => 1,
                     c_full_flags_rst_val => 0,
                     c_implementation_type => 2,
                     c_family => «virtex5»,
                     c_use_embedded_reg => 1,
                     c_has_wr_rst => 0,
                     c_wr_freq => 1,
                     c_underflow_low => 1,
                     c_has_meminit_file => 0,
                     c_has_overflow => 1,
                     c_preload_latency => 0,
                     c_dout_width => 32,
                     c_rd_depth => 1024,
                     c_default_value => «BlankString»,
                     c_mif_file_name => «BlankString»,
                     c_has_underflow => 1,
                     c_has_rd_rst => 0,
                     c_has_almost_full => 1,
                     c_has_rst => 1,
                     c_data_count_width => 12,
                     c_has_wr_ack => 1,
                     c_use_ecc => 0,
                     c_wr_ack_low => 1,
                     c_common_clock => 0,
                     c_rd_pntr_width => 10,
                     c_use_fwft_data_count => 1,
                     c_has_almost_empty => 1,
                     c_rd_data_count_width => 11,
                     c_enable_rlocs => 0,
                     c_wr_pntr_width => 12,
                     c_overflow_low => 1,
                     c_prog_empty_type => 4,
                     c_optimization_mode => 0,
                     c_wr_data_count_width => 13,
                     c_preload_regs => 1,
                     c_dout_rst_val => «0»,
                     c_has_data_count => 0,
                     c_prog_full_thresh_negate_val => 4094,
                     c_wr_depth => 4096,
                     c_prog_empty_thresh_negate_val => 5,
                     c_prog_empty_thresh_assert_val => 4,
                     c_has_valid => 1,
                     c_init_wr_pntr_val => 0,
                     c_prog_full_thresh_assert_val => 4095,
                     c_use_fifo16_flags => 0,
                     c_has_backup => 0,
                     c_valid_low => 1,
                     c_prim_fifo_type => «4kx9»,
                     c_count_type => 0,
                     c_prog_full_type => 4,
                     c_memory_type => 1
                     );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_fifo_generator_v4_1_independent_clocks_block_ram
         port map (
                  din => din,
                  prog_empty_thresh_assert => prog_empty_thresh_assert,
                  prog_empty_thresh_negate => prog_empty_thresh_negate,
                  prog_full_thresh_assert => prog_full_thresh_assert,
```

```
                prog_full_thresh_negate => prog_full_thresh_negate,
                rd_clk => rd_clk,
                rd_en => rd_en,
                rst => rst,
                wr_clk => wr_clk,
                wr_en => wr_en,
                almost_empty => almost_empty,
                almost_full => almost_full,
                dout => dout,
                empty => empty,
                full => full,
                overflow => overflow,
                prog_empty => prog_empty,
                prog_full => prog_full,
                valid => valid,
                rd_data_count => rd_data_count,
                underflow => underflow,
                wr_ack => wr_ack,
                wr_data_count => wr_data_count
                );
-- synthesis translate_on
--
END fifo_generator_v4_1_independent_clocks_block_ram_a;
```

В представленном элементе FIFO-памяти применяется режим чтения с функцией предварительного просмотра данных First-word fall-through (FWFT), а также режим синхронного сброса. В модулях блочной памяти Block SelectRAM, используемых для реализации запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram*, задействованы встроенные выходные регистры. Интерфейс этого элемента, помимо входных и выходных шин данных, включает в себя выходы сигналов, информирующих о том, что в запоминающее устройство может быть записано или извлечено из памяти не более одного слова данных. Кроме того, в запоминающем устройстве *fifo_generator_v4_1_independent_clocks_block_ram* задействованы дополнительные, программируемые пользователем, выходы сигналов состояния, условия установки и сброса которых могут определяться в процессе функционирования запоминающего устройства. Для этого в составе рассматриваемого элемента предусмотрены две пары входных шин, предназначенные для записи значений, определяющих условия переключения соответствующих дополнительных программируемых сигналов состояния.

Сформированное запоминающее устройство *fifo_generator_v4_1_independent_clocks_block_ram* предоставляет возможность использования совокупности сигналов подтверждения для портов записи и чтения данных, которые поддерживаются параметризированным модулем FIFO Generator версии v4.1. Активным уровнем для этих сигналов подтверждения является низкий логический уровень. В составе интерфейса сгенерированного элемента FIFO-памяти присутствуют две выходные шины, на которых в полном объеме отображается состояние выходов счетчиков записанных слов данных, относящихся к порту записи и чтения. В порте записи запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram* применяется 13-разрядный счетчик записанных слов данных. Выходная шина счетчика записанных слов данных, относящегося к порту чтения этого элемента, содержит 11 разрядов.

Для использования элемента FIFO-памяти *fifo_generator_v4_1_independent_clocks_block_ram* в качестве одного из компонентов в составе описания разрабатываемого устройства необходимо поместить в раздел декларации VHDL-описания следующую конструкцию:

```
component fifo_generator_v4_1_independent_clocks_block_ram
   port (
         din: IN std_logic_VECTOR(7 downto 0);
         prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
         prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
         prog_full_thresh_assert: IN std_logic_VECTOR(11 downto 0);
         prog_full_thresh_negate: IN std_logic_VECTOR(11 downto 0);
         rd_clk: IN std_logic;
         rd_en: IN std_logic;
         rst: IN std_logic;
         wr_clk: IN std_logic;
         wr_en: IN std_logic;
         almost_empty: OUT std_logic;
         almost_full: OUT std_logic;
         dout: OUT std_logic_VECTOR(31 downto 0);
         empty: OUT std_logic;
         full: OUT std_logic;
         overflow: OUT std_logic;
         prog_empty: OUT std_logic;
         prog_full: OUT std_logic;
         valid: OUT std_logic;
         rd_data_count: OUT std_logic_VECTOR(10 downto 0);
         underflow: OUT std_logic;
         wr_ack: OUT std_logic;
         wr_data_count: OUT std_logic_VECTOR(12 downto 0)
         );
end component;
```

Создание конкретных экземпляров компонента запоминающего устройства FIFO, представленного в настоящем разделе, в VHDL-описании проектируемого устройства осуществляется с помощью оператора, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_компонента_fifo_generator_v4_1_
independent_clocks_block_ram>: fifo_generator_v4_1_independent_
clocks_block_ram
   port map (
                din => din,
                prog_empty_thresh_assert => prog_empty_thresh_assert,
                prog_empty_thresh_negate => prog_empty_thresh_negate,
                prog_full_thresh_assert => prog_full_thresh_assert,
                prog_full_thresh_negate => prog_full_thresh_negate,
                rd_clk => rd_clk,
                rd_en => rd_en,
                rst => rst,
                wr_clk => wr_clk,
                wr_en => wr_en,
                almost_empty => almost_empty,
                almost_full => almost_full,
                dout => dout,
                empty => empty,
                full => full,
                overflow => overflow,
                prog_empty => prog_empty,
                prog_full => prog_full,
                valid => valid,
                rd_data_count => rd_data_count,
                underflow => underflow,
                wr_ack => wr_ack,
                wr_data_count => wr_data_count
                );
```

Информационная панель, вид которой приведен на рис. 183, содержит исчерпывающие сведения о количестве триггеров (Flip Flop), таблиц преобразования (LUT) секций (Slices) и модулей блочной памяти кристалла Block SelectRAM, которое необходимо для автономной реализации запоминающего устройства *fifo_generator_v4_1_independent_clocks_block_ram*.

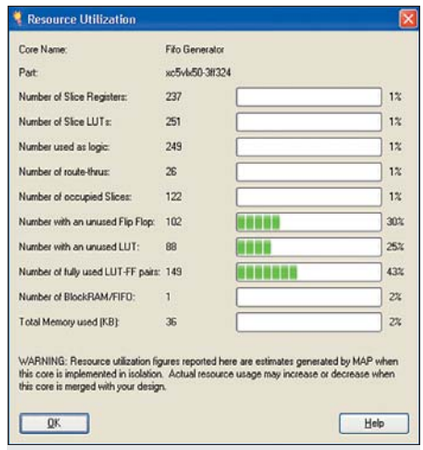

Рис. 183. Вид информационной панели, содержащей сведения о количестве различных ресурсов ПЛИС, используемых для реализации элемента FIFO-памяти *fifo_generator_v4_1_independent_clocks_block_ram*

Пример описания элемента запоминающего устройства FIFO с двумя различными сигналами синхронизации портов записи и чтения данных, сформированного с помощью параметризированного модуля FIFO Generator версии v4.1 для последующей реализации на базе ресурсов распределенной памяти ПЛИС

В качестве примера описания запоминающего устройства FIFO с двумя различными сигналами синхронизации портов записи и чтения данных, сформированного с помощью параметризированного модуля FIFO Generator версии v4.1 для последующей реализации на основе ресурсов распределенной памяти ПЛИС Block SelectRAM, ниже представлен текст VHDL-описания элемента *fifo_generator_v4_1_independent_clocks_distributed_ram*. Информационная емкость данного запоминающего устройства составляет 48 кбит. Порты записи и чтения данных в этом элементе FIFO-памяти имеют организацию 1024 слова×48 разрядов:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY fifo_generator_v4_1_independent_clocks_distributed_ram IS
   port (
         din: IN std_logic_VECTOR(47 downto 0);
         prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
         prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
         prog_full_thresh_assert: IN std_logic_VECTOR(9 downto 0);
         prog_full_thresh_negate: IN std_logic_VECTOR(9 downto 0);
         rd_clk: IN std_logic;
         rd_en: IN std_logic;
         rst: IN std_logic;
         wr_clk: IN std_logic;
         wr_en: IN std_logic;
         almost_empty: OUT std_logic;
         almost_full: OUT std_logic;
         dout: OUT std_logic_VECTOR(47 downto 0);
         empty: OUT std_logic;
         full: OUT std_logic;
```

```
        overflow: OUT std_logic;
        prog_empty: OUT std_logic;
        prog_full: OUT std_logic;
        valid: OUT std_logic;
        rd_data_count: OUT std_logic_VECTOR(9 downto 0);
        underflow: OUT std_logic;
        wr_ack: OUT std_logic;
        wr_data_count: OUT std_logic_VECTOR(9 downto 0)
        );
END fifo_generator_v4_1_independent_clocks_distributed_ram;
--
ARCHITECTURE fifo_generator_v4_1_independent_clocks_distributed_
ram_a OF fifo_generator_v4_1_independent_clocks_distributed_ram IS
-- synthesis translate_off
component wrapped_fifo_generator_v4_1_independent_clocks_dis-
tributed_ram
   port (
        din: IN std_logic_VECTOR(47 downto 0);
        prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
        prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
        prog_full_thresh_assert: IN std_logic_VECTOR(9 downto 0);
        prog_full_thresh_negate: IN std_logic_VECTOR(9 downto 0);
        rd_clk: IN std_logic;
        rd_en: IN std_logic;
        rst: IN std_logic;
        wr_clk: IN std_logic;
        wr_en: IN std_logic;
        almost_empty: OUT std_logic;
        almost_full: OUT std_logic;
        dout: OUT std_logic_VECTOR(47 downto 0);
        empty: OUT std_logic;
        full: OUT std_logic;
        overflow: OUT std_logic;
        prog_empty: OUT std_logic;
        prog_full: OUT std_logic;
        valid: OUT std_logic;
        rd_data_count: OUT std_logic_VECTOR(9 downto 0);
        underflow: OUT std_logic;
        wr_ack: OUT std_logic;
        wr_data_count: OUT std_logic_VECTOR(9 downto 0)
        );
end component;
--
-- Configuration specification
   for all : wrapped_fifo_generator_v4_1_independent_clocks_distrib-
uted_ram use entity XilinxCoreLib.fifo_generator_v4_1(behavioral)
        generic map(
                c_has_int_clk => 0,
                c_rd_freq => 1,
                c_wr_response_latency => 1,
                c_has_srst => 0,
                c_has_rd_data_count => 1,
                c_din_width => 48,
                c_has_wr_data_count => 1,
                c_full_flags_rst_val => 1,
                c_implementation_type => 2,
                c_family => «virtex5»,
                c_use_embedded_reg => 0,
                c_has_wr_rst => 0,
                c_wr_freq => 1,
                c_underflow_low => 0,
                c_has_meminit_file => 0,
                c_has_overflow => 1,
                c_preload_latency => 1,
                c_dout_width => 48,
                c_rd_depth => 1024,
                c_default_value => «BlankString»,
                c_mif_file_name => «BlankString»,
                c_has_underflow => 1,
                c_has_rd_rst => 0,
                c_has_almost_full => 1,
                c_has_rst => 1,
                c_data_count_width => 10,
                c_has_wr_ack => 1,
                c_use_ecc => 0,
                c_wr_ack_low => 0,
                c_common_clock => 0,
                c_rd_pntr_width => 10,
                c_use_fwft_data_count => 0,
                c_has_almost_empty => 1,
                c_rd_data_count_width => 10,
                c_enable_rlocs => 0,
                c_wr_pntr_width => 10,
                c_overflow_low => 0,
                c_prog_empty_type => 4,
                c_optimization_mode => 0,
                c_wr_data_count_width => 10,
                c_preload_regs => 0,
                c_dout_rst_val => «1»,
                c_has_data_count => 0,
                c_prog_full_thresh_negate_val => 1020,
                c_wr_depth => 1024,
                c_prog_empty_thresh_negate_val => 3,
                c_prog_empty_thresh_assert_val => 2,
                c_has_valid => 1,
                c_init_wr_pntr_val => 0,
                c_prog_full_thresh_assert_val => 1021,
                c_use_fifo16_flags => 0,
                c_has_backup => 0,
                c_valid_low => 0,
                c_prim_fifo_type => «1kx36»,
                c_count_type => 0,
                c_prog_full_type => 4,
                c_memory_type => 2
                );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_fifo_generator_v4_1_independent_clocks_distributed_ram
        port map (
                din => din,
                prog_empty_thresh_assert => prog_empty_thresh_assert,
                prog_empty_thresh_negate => prog_empty_thresh_negate,
                prog_full_thresh_assert => prog_full_thresh_assert,
                prog_full_thresh_negate => prog_full_thresh_negate,
                rd_clk => rd_clk,
                rd_en => rd_en,
                rst => rst,
                wr_clk => wr_clk,
                wr_en => wr_en,
                almost_empty => almost_empty,
                almost_full => almost_full,
                dout => dout,
                empty => empty,
                full => full,
                overflow => overflow,
                prog_empty => prog_empty,
                prog_full => prog_full,
                valid => valid,
                rd_data_count => rd_data_count,
                underflow => underflow,
                wr_ack => wr_ack,
                wr_data_count => wr_data_count
                );
-- synthesis translate_on
--
END fifo_generator_v4_1_independent_clocks_distributed_ram_a;
```

В сформированном запоминающем устройстве *fifo_generator_v4_1_independent_clocks_distributed_ram* используется стандартный режим чтения данных. В состав интерфейса этого элемента включены дополнительные выходы сигналов состояния, информирующих о том, что в запоминающее устройство можно записать или считать из памяти не более одного слова данных. Кроме того, в элементе *fifo_generator_v4_1_independent_clocks_distributed_ram* предусмотрены программируемые пользователем выходы сигналов состояния, условия переключения каждого из которых определяются двумя параметрами. Значения этих параметров задаются в процессе функционирования сгенерированного запоминающего устройства (в режиме сброса) в виде двоичных кодов, подаваемых на соответствующие входные шины.

Рассматриваемый элемент памяти, функционирующий по принципу «первым вошел–первым вышел», позволяет организовать выполнение операций записи и чтения данных с использованием всех сигналов подтверждения, поддерживаемых параметризированным модулем FIFO Generator версии v4.1. В качестве активного уровня сигналов на этих выходах выбран высокий логический уровень. В интерфейсе запоминающего устройства *fifo_generator_v4_1_independent_clocks_distributed_ram* присутствуют выходные шины счетчиков записанных слов данных, относящихся к портам записи и чтения. Каждая из этих выходных шин счетчиков имеет разрядность, равную десяти. В этом элементе задействован также вход сигнала асинхронного сброса.

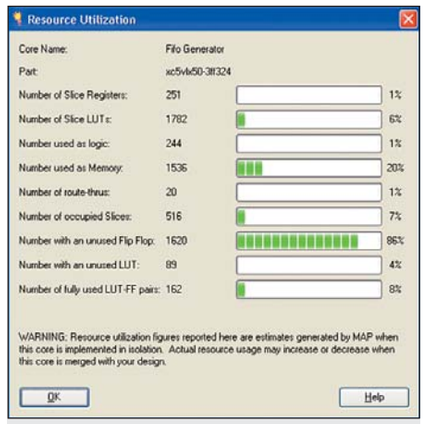

Рис. 184. Вид информационной панели, содержащей сведения о количестве различных ресурсов ПЛИС, используемых для автономной реализации элемента запоминающего устройства *fifo_generator_v4_1_independent_clocks_distributed_ram*

Чтобы использовать элемент запоминающего устройства *fifo_generator_v4_1_independent_clocks_distributed_ram* в качестве компонента VHDL-описания разрабатываемого устройства, нужно добавить в раздел декларации этого описания приведенную ниже последовательность выражений:

```
component fifo_generator_v4_1_independent_clocks_distributed_ram
   port (
        din: IN std_logic_VECTOR(47 downto 0);
        prog_empty_thresh_assert: IN std_logic_VECTOR(9 downto 0);
        prog_empty_thresh_negate: IN std_logic_VECTOR(9 downto 0);
        prog_full_thresh_assert: IN std_logic_VECTOR(9 downto 0);
        prog_full_thresh_negate: IN std_logic_VECTOR(9 downto 0);
        rd_clk: IN std_logic;
        rd_en: IN std_logic;
        rst: IN std_logic;
        wr_clk: IN std_logic;
        wr_en: IN std_logic;
        almost_empty: OUT std_logic;
        almost_full: OUT std_logic;
        dout: OUT std_logic_VECTOR(47 downto 0);
        empty: OUT std_logic;
        full: OUT std_logic;
        overflow: OUT std_logic;
        prog_empty: OUT std_logic;
        prog_full: OUT std_logic;
        valid: OUT std_logic;
        rd_data_count: OUT std_logic_VECTOR(9 downto 0);
        underflow: OUT std_logic;
        wr_ack: OUT std_logic;
        wr_data_count: OUT std_logic_VECTOR(9 downto 0)
        );
end component;
```

Конкретные экземпляры данного компонента в составе блока определения архитектуры проектируемого устройства формируются с помощью оператора, шаблон которого выглядит следующим образом.

```
<идентификатор_экземпляра_компонента_fifo_generator_
v4_1_independent_clocks_distributed_ram>: fifo_generator_v4_1_
independent_clocks_distributed_ram
   port map (
                din => din,
                prog_empty_thresh_assert => prog_empty_thresh_assert,
                prog_empty_thresh_negate => prog_empty_thresh_negate,
                prog_full_thresh_assert => prog_full_thresh_assert,
                prog_full_thresh_negate => prog_full_thresh_negate,
                rd_clk => rd_clk,
                rd_en => rd_en,
                rst => rst,
                wr_clk => wr_clk,
```

```
        wr_en => wr_en,
        almost_empty => almost_empty,
        almost_full => almost_full,
        dout => dout,
        empty => empty,
        full => full,
        overflow => overflow,
        prog_empty => prog_empty,
        prog_full => prog_full,
        valid => valid,
        rd_data_count => rd_data_count,
        underflow => underflow,
        wr_ack => wr_ack,
        wr_data_count => wr_data_count
        );
```

Подробный отчет об объеме различных ресурсов кристалла, который требуется для автономной реализации элемента запоминающего устройства *fifo_generator_v4_1_independent_clocks_distributed_ram*, представлен в информационной панели, вид которой показан на рис. 184.

Пример описания элемента запоминающего устройства, функционирующего по принципу «первым вошел–первым вышел», с раздельными сигналами синхронизации портов записи и чтения данных, сформированного с помощью параметризированного ядра FIFO Generator версии v4.1 для последующей реализации на основе встроенных модулей FIFO

Примером запоминающего устройства, функционирующего по принципу «первым вошел–первым вышел», с двумя различными сигналами синхронизации портов записи и чтения данных, которое сформировано с помощью параметризированного ядра FIFO Generator версии v4.1 для последующей реализации на основе встроенных модулей FIFO в ПЛИС семейств Virtex-4 и Virtex-5, является элемент *fifo_generator_v4_1_independent_clocks_built_in_fifo*. Данный элемент FIFO-памяти обладает информационной емкостью 128 кбит с организацией входного и выходного информационных портов (портов записи и чтения данных) 2048 слов×64 разряда. Текст сгенерированного VHDL-описания запоминающего устройства *fifo_generator_v4_1_independent_clocks_built_in_fifo* выглядит следующим образом:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY fifo_generator_v4_1_independent_clocks_built_in_fifo IS
   port (
        din: IN std_logic_VECTOR(63 downto 0);
        rd_clk: IN std_logic;
        rd_en: IN std_logic;
        rst: IN std_logic;
        wr_clk: IN std_logic;
        wr_en: IN std_logic;
        dout: OUT std_logic_VECTOR(63 downto 0);
        empty: OUT std_logic;
        full: OUT std_logic;
        overflow: OUT std_logic;
        prog_empty: OUT std_logic;
        prog_full: OUT std_logic;
        valid: OUT std_logic;
        underflow: OUT std_logic;
        wr_ack: OUT std_logic;
        sbiterr: OUT std_logic;
        dbiterr: OUT std_logic
        );
END fifo_generator_v4_1_independent_clocks_built_in_fifo;
--
ARCHITECTURE fifo_generator_v4_1_independent_clocks_built_in_fifo_a
OF fifo_generator_v4_1_independent_clocks_built_in_fifo IS
-- synthesis translate_off
component wrapped_fifo_generator_v4_1_independent_clocks_built_in_fifo
   port (
        din: IN std_logic_VECTOR(63 downto 0);
        rd_clk: IN std_logic;
        rd_en: IN std_logic;
        rst: IN std_logic;
        wr_clk: IN std_logic;
        wr_en: IN std_logic;
        dout: OUT std_logic_VECTOR(63 downto 0);
        empty: OUT std_logic;
        full: OUT std_logic;
        overflow: OUT std_logic;
        prog_empty: OUT std_logic;
        prog_full: OUT std_logic;
        valid: OUT std_logic;
        underflow: OUT std_logic;
        wr_ack: OUT std_logic;
        sbiterr: OUT std_logic;
        dbiterr: OUT std_logic
        );
end component;
--
-- Configuration specification
   for all : wrapped_fifo_generator_v4_1_independent_clocks_built_
in_fifo use entity XilinxCoreLib.fifo_generator_v4_1(behavioral)
        generic map(
                c_has_int_clk => 0,
                c_rd_freq => 100,
                c_wr_response_latency => 1,
                c_has_srst => 0,
                c_has_rd_data_count => 0,
                c_din_width => 64,
                c_has_wr_data_count => 0,
                c_full_flags_rst_val => 0,
                c_implementation_type => 4,
                c_family => «virtex5»,
                c_use_embedded_reg => 0,
                c_has_wr_rst => 0,
                c_wr_freq => 200,
                c_underflow_low => 1,
                c_has_meminit_file => 0,
                c_has_overflow => 1,
                c_preload_latency => 0,
                c_dout_width => 64,
                c_rd_depth => 2048,
                c_default_value => «BlankString»,
                c_mif_file_name => «BlankString»,
                c_has_underflow => 1,
                c_has_rd_rst => 0,
                c_has_almost_full => 0,
                c_has_rst => 1,
                c_data_count_width => 11,
                c_has_wr_ack => 1,
                c_use_ecc => 1,
                c_wr_ack_low => 1,
                c_common_clock => 0,
                c_rd_pntr_width => 11,
                c_use_fwft_data_count => 0,
                c_has_almost_empty => 0,
                c_rd_data_count_width => 11,
                c_enable_rlocs => 0,
                c_wr_pntr_width => 11,
                c_overflow_low => 1,
                c_prog_empty_type => 1,
                c_optimization_mode => 0,
                c_wr_data_count_width => 11,
                c_preload_regs => 1,
                c_dout_rst_val => «0»,
                c_has_data_count => 0,
                c_prog_full_thresh_negate_val => 2039,
                c_wr_depth => 2048,
                c_prog_empty_thresh_negate_val => 11,
                c_prog_empty_thresh_assert_val => 10,
                c_has_valid => 1,
                c_init_wr_pntr_val => 0,
                c_prog_full_thresh_assert_val => 2040,
                c_use_fifo16_flags => 0,
                c_has_backup => 0,
                c_valid_low => 1,
                c_prim_fifo_type => «512x72»,
                c_count_type => 0,
                c_prog_full_type => 1,
                c_memory_type => 4
                );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_fifo_generator_v4_1_independent_clocks_built_in_fifo
        port map (
                din => din,
                rd_clk => rd_clk,
                rd_en => rd_en,
                rst => rst,
                wr_clk => wr_clk,
                wr_en => wr_en,
                dout => dout,
                empty => empty,
                full => full,
                overflow => overflow,
                prog_empty => prog_empty,
                prog_full => prog_full,
                valid => valid,
                underflow => underflow,
                wr_ack => wr_ack,
                sbiterr => sbiterr,
                dbiterr => dbiterr
                );
-- synthesis translate_on
--
END fifo_generator_v4_1_independent_clocks_built_in_fifo_a;
```

В элементе *fifo_generator_v4_1_independent_clocks_built_in_fifo* используется режим чтения информации с возможностью предварительного просмотра данных FWFT. Применение в этом запоминающем устройстве функции коррекции ошибок ECC (Error Correction Checking) позволяет исправлять одиночные ошибки (ошибки в одном разряде слова данных) и обнаруживать двойные (ошибки в двух разрядах слова данных).

В составе интерфейса данного элемента FIFO-памяти присутствует вход сигнала синхронного сброса. Сформированное запоминающее устройство *fifo_generator_v4_1_independent_clocks_built_in_fifo* поддерживает возможность выполнения операций записи и чтения данных с использованием всех сигналов подтверждения, предоставляемых параметризированным модулем FIFO Generator версии v4.1. В качестве активного уровня этих сигналов выбран низкий логический уровень. В рассматриваемом элементе FIFO-памяти предусмотрены также программируемые пользователем выходы сигналов предупреждения PROG_EMPTY и PROG_FULL, которые информируют о достижении заданных верхней и нижней границ массива слов данных, записанных в данное запоминающее устройство. Активный уровень на выходе сигнала предупреждения PROG_EMPTY формируется в том случае, если количество слов данных, записанных в элемент *fifo_generator_v4_1_independent_clocks_built_in_fifo*, меньше или равно десяти. Сигнал предупреждения PROG_FULL переключается в активное состояние тогда, когда число записанных слов данных в указанном элементе FIFO-памяти больше или равно 2048.

Декларация представленного выше элемента запоминающего устройства *fifo_generator_v4_1_independent_clocks_built_in_fifo* при использовании его в качестве компонента в составе описания разрабатываемого устройства выполняется с помощью конструкции, которая имеет следующий вид:

```
component fifo_generator_v4_1_independent_clocks_built_in_fifo
   port (
         din: IN std_logic_VECTOR(63 downto 0);
         rd_clk: IN std_logic;
         rd_en: IN std_logic;
         rst: IN std_logic;
         wr_clk: IN std_logic;
         wr_en: IN std_logic;
         dout: OUT std_logic_VECTOR(63 downto 0);
         empty: OUT std_logic;
         full: OUT std_logic;
         overflow: OUT std_logic;
         prog_empty: OUT std_logic;
         prog_full: OUT std_logic;
         valid: OUT std_logic;
         underflow: OUT std_logic;
         wr_ack: OUT std_logic;
         sbiterr: OUT std_logic;
         dbiterr: OUT std_logic
         );
end component;
```

Для создания конкретного экземпляра компонента *fifo_generator_v4_1_independent_clocks_built_in_fifo* нужно включить в состав блока определения архитектуры проектируемого устройства оператор, шаблон которого выглядит следующим образом.

```
<идентификатор_экземпляра_компонента_fifo_generator_v4_1_
independent_clocks_built_in_fifo>: fifo_generator_v4_1_independ-
ent_clocks_built_in_fifo
   port map (
                  din => din,
                  rd_clk => rd_clk,
                  rd_en => rd_en,
                  rst => rst,
                  wr_clk => wr_clk,
                  wr_en => wr_en,
                  dout => dout,
                  empty => empty,
                  full => full,
                  overflow => overflow,
                  prog_empty => prog_empty,
                  prog_full => prog_full,
                  valid => valid,
                  underflow => underflow,
                  wr_ack => wr_ack,
                  sbiterr => sbiterr,
                  dbiterr => dbiterr
                  );
```

Подробные сведения о количестве различных ресурсов ПЛИС (включая встроенные модули FIFO, конфигурируемые на основе ресурсов блочной памяти кристалла Block SelectRAM), используемом для автономной реализации элемента запоминающего устройства *fifo_generator_v4_1_independent_clocks_built_in_fifo*, отражены в отчете, содержащемся в информационной панели, вид которой представлен на рис. 185.

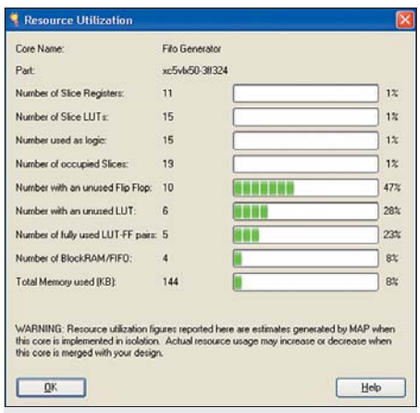

Рис. 185. Вид информационной панели, содержащей сведения о количестве различных ресурсов ПЛИС, используемых для автономной реализации элемента запоминающего устройства *fifo_generator_v4_1_independent_clocks_built_in_fifo*

Описание интерфейса элементов запоминающих устройств, функционирующих по принципу «первым вошел–первым вышел», которые формируются на основе параметризированного модуля FIFO Generator версии v4.1

В описаниях элементов запоминающих устройств, работающих по принципу «первым вошел–первым вышел», которые создаются с помощью параметризированного модуля FIFO Generator версии v4.1, используется следующая система условных обозначений входных и выходных портов:

- clk — вход сигнала синхронизации в элементах FIFO-памяти, в которых порты записи и чтения данных тактируются одним сигналом;
- wr_clk — вход сигнала синхронизации для порта записи данных в элементах запоминающих устройств FIFO, в которых входной и выходной порты управляются двумя различными тактовыми сигналами;
- rd_clk — вход сигнала синхронизации для порта чтения данных в элементах FIFO-памяти, в которых входной и выходной порты тактируются раздельными сигналами;
- rst — вход сигнала асинхронного сброса;
- srst — вход сигнала синхронного сброса;
- din[N:0] — входная шина с разрядностью N+1, на которую поступают информационные данные, записываемые в запоминающее устройство FIFO;
- wr_en — вход сигнала разрешения (или запрета) выполнения операции записи данных в элемент FIFO-памяти;
- rd_en — вход сигнала, разрешающего или запрещающего выполнение операции чтения (извлечения) данных из запоминающего устройства FIFO;
- almost_full — выход сигнала ALMOST_FULL, предупреждающего о том, что в элемент памяти может быть записано еще не более одного слова данных;
- almost_empty — выход сигнала ALMOST_EMPTY, информирующего о том, что не более одного слова данных может быть еще считано из сформированного запоминающего устройства;
- empty — выход сигнала EMPTY, активный уровень которого уведомляет об отсутствии записанной информации в элементе FIFO-памяти и невозможности корректного выполнения операции чтения данных;
- full — выход сигнала FULL, активный уровень которого предупреждает о заполнении всего объема запоминающего устройства FIFO (всех ячеек FIFO-памяти) и невозможности осуществления операции записи новых данных;
- dout[M:0] — выходная шина данных с разрядностью M+1, на которую выводится информация, считываемая из элемента FIFO-памяти;
- wr_ack — выход сигнала Write Acknowledge, информирующего об успешном выполнении операции записи данных, осуществляемой в течение предыдущего периода тактового сигнала;
- overflow — выход сигнала Overflow, сообщающего об ошибке в процессе выполнения операции записи, производимой во время предыдущего периода тактового сигнала, из-за переполнения емкости запоминающего устройства;
- valid — выход сигнала Valid, активный уровень которого информирует о достоверности информации, представленной на выходной шине данных;
- underflow — выход сигнала Underflow, извещающего об ошибке в процессе выполнения операции чтения данных, осуществляемой в течение предыдущего периода тактового сигнала, из-за отсутствия записанной информации в запоминающем устройстве FIFO;
- prog_full — выход дополнительного, программируемого пользователем, сигнала состояния PROG_FULL, активный уровень которого сообщает о записи определенного числа слов данных (заполнении заданного пользователем числа ячеек FIFO-памяти);
- prog_empty — выход дополнительного, программируемого пользователем, сигнала состояния PROG_EMPTY, информирующего о достижении заданного минимально допустимого числа записанных слов данных в процессе операции чтения;
- prog_full_thresh[K:0] — входная шина с разрядностью K+1, предназначенная для записи двоичного кода значения, которое определяет условие переключения уровня (сброса и установки) программируемого пользователем сигнала состояния PROG_FULL, сообщающего о записи определенного числа слов данных в элемент памяти FIFO;
- prog_full_thresh_assert[K:0] — входная шина с разрядностью K+1, на которую подается двоичный код значения, определяющего условие установки программируемого пользователем сигнала состояния PROG_FULL, информирующего о заполнении заданного числа ячеек запоминающего устройства FIFO;
- prog_full_thresh_negate[K:0] — входная шина с разрядностью K+1, на которую поступает двоичный код значения, определяющего условие сброса программируемого пользователем сигнала состояния PROG_FULL, сообщающего о записи определенного числа слов данных в элемент памяти FIFO;
- prog_empty_thresh[K:0] — входная шина с разрядностью K+1, предназначенная для загрузки двоичного кода значения, которое устанавливает условие переключения

уровня программируемого пользователем сигнала состояния PROG_EMPTY, предупреждающего о достижении заданного минимально допустимого числа записанных слов данных в процессе операции чтения;

- prog_empty_thresh_assert[K:0] — входная шина с разрядностью K+1, на которую подается двоичный код значения, определяющего условие установки программируемого пользователем сигнала состояния PROG_EMPTY, уведомляющего о достижении заданного минимально допустимого числа записанных слов данных в процессе операции чтения;
- prog_empty_thresh_negate[K:0] — входная шина с разрядностью K+1, на которую поступает двоичный код значения, определяющего условие сброса программируемого пользователем сигнала состояния PROG_EMPTY, предупреждающего о достижении заданного минимально допустимого числа записанных слов данных в процессе операции чтения;
- data_count[L:0] — выходная шина с разрядностью L +1, на которой отображается текущее значение счетчика числа слов данных, записанных в запоминающее устройство FIFO, тактируемое одним сигналом синхронизации;
- wr_data_count [L:0] — выходная шина с разрядностью L +1, на которой отображается текущее значение счетчика числа записанных слов данных, относящегося к порту записи в запоминающем устройстве FIFO с двумя различными сигналами синхронизации входного и выходного портов;
- rd_data_count [L:0] — выходная шина с разрядностью L +1, на которую выводится текущее значение счетчика числа записанных слов данных, относящегося к порту чтения в запоминающем устройстве FIFO с раздельными сигналами синхронизации входного и выходного портов;
- sbiterr — выход сигнала, активный уровень которого информирует о наличии скорректированной ошибки в одном разряде слова данных в элементах FIFO-памяти, реализуемых на базе ресурсов блочной памяти ПЛИС Block SelectRAM или встроенных модулей FIFO в кристаллах семейства Virtex-5;
- dbiterr — выход сигнала, активный уровень которого уведомляет об обнаружении ошибок в двух разрядах слова данных в запоминающих устройствах, выполняемых на основе ресурсов блочной памяти кристалла Block SelectRAM или встроенных модулей FIFO в ПЛИС семейства Virtex-5.

Следует обратить внимание на то, что наличие активного уровня сигнала на выходе dbiterr свидетельствует об искажении соответствующего слова данных, записанного в сформированный элемент FIFO-памяти. При выполнении операций записи нужно учитывать, что в случае возникновения переполнения емкости запоминающего устройства (при появлении активного уровня сигнала на выходе overflow) информация, загруженная ранее в этот элемент, сохраняется.

Окончание следует