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 сумматоров и вычитающих устройств с помощью генератора параметризированных модулей CORE Generator

Для подготовки описаний сумматоров и вычитающих устройств в среде генератора параметризированных модулей CORE Generator применяется ядро *Adder/Subtractor*. Данный параметризированный модуль относится к группе ядер, предназначенных для реализации математических функций *Math Functions*. На основе ядра *Adder/Subtractor* можно формировать описания сумматоров и вычитающих устройств с различной разрядностью для последующей реализации в кристаллах большинства современных семейств ПЛИС: Spartan-II; Spartan-IIE; Spartan-3; Spartan-3E; Spartan-3A; Virtex; QPRO Virtex Rad-Hard; QPRO Virtex Hi-Rel; Virtex-E; QPRO Virtex-E Military; Virtex-II; Virtex-II PRO и Virtex-4.

К особенностям параметризированного модуля *Adder/Subtractor* относятся:

- расширенная функциональность, позволяющая выбирать различные варианты выполняемой операции для формируемого устройства (суммирования, вычитания, суммирования/вычитания);
- возможность генерации сумматоров и вычитающих устройств, выполняющих соответствующие операции над входным словом данных и константой, значение которой определяется пользователем;
- поддержка широкого диапазона разрядности используемых операндов (входных шин данных) от 1 до 256 двоичных разрядов;
- возможность выбора разрядности выходных данных в диапазоне от 1 до 258 разрядов;
- поддержка различной формы представления входных данных и полученного результата (в форме значения со знаком или без знака);
- возможность генерации описаний сумматоров и вычитающих устройств с поддержкой режима непосредственной (прямой) загрузки данных;
- поддержка выбора активного уровня сигнала на входе управления непосредственной загрузкой данных;
- применение конвейерного метода выполнения операций (по выбору пользователя);
- возможность создания сумматоров и вычитающих устройств с выходами различного типа (комбинационным и регистровым);
- возможность выборочного использования в выходном регистре создаваемых сумматоров и вычитающих устройств синхронных и асинхронных входов сброса и/или установки, входов инициализации, а также входа разрешения синхронизации.

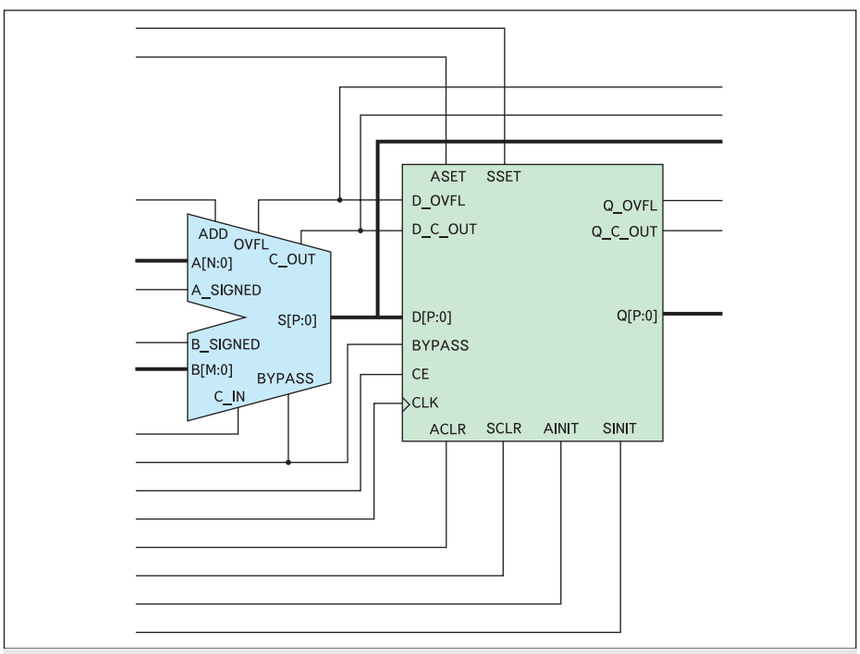

Рис. 46. Обобщенная структура сумматоров и вычитающих устройств, формируемых на основе параметризированного модуля Adder/Subtractor

Обобщенная структура сумматоров и вычитающих устройств, формируемых на основе параметризированного модуля *Adder/Subtractor*, показана на рис. 46.

Установка требуемых значений параметров создаваемых устройств, выполняющих операции сложения и вычитания, осуществляется с помощью соответствующего «мастера» настройки, в состав которого входят две основные диалоговые панели и одна дополнительная. Стартовая диалоговая панель данно-

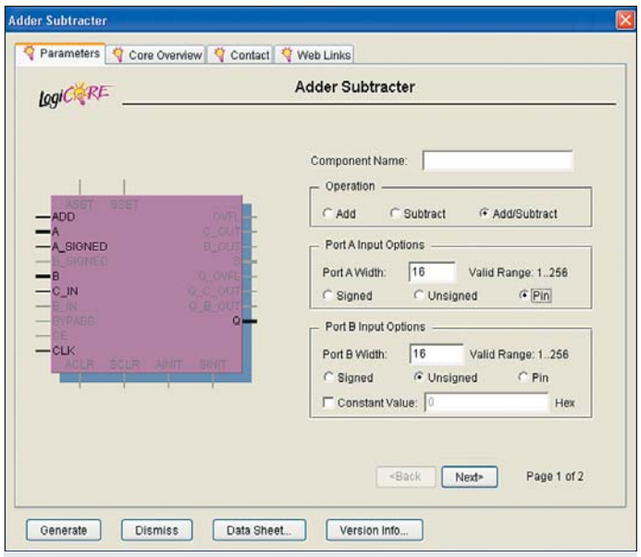

Рис. 47. Страница Parameters стартовой диалоговой панели «мастера» настройки параметров ядра сумматора/вычитающего устройства Adder/Subtractor

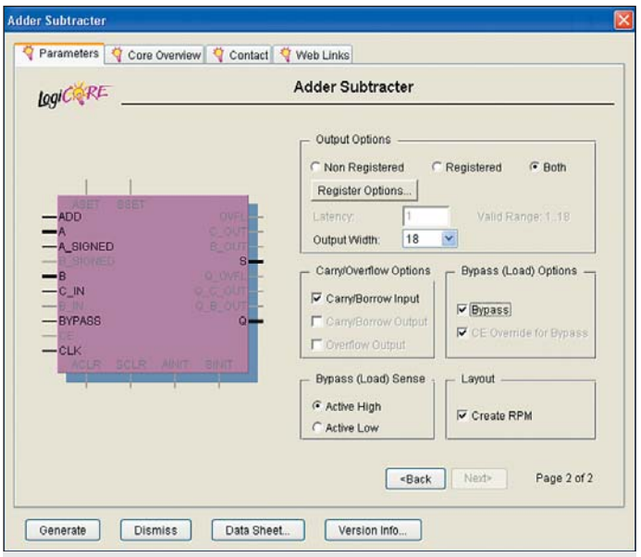

Рис. 48. Страница Parameters заключительной диалоговой панели «мастера» настройки параметров ядра сумматора/вычитающего устройства Adder/Subtractor

го «мастера» предназначена для определения значений главных параметров формируемого сумматора или вычитающего устройства. Страница *Parameters* этой диалоговой панели, вид которой приведен на рис. 47, позволяет выбрать тип выполняемой операции и разрядность генерируемого элемента.

Процесс разработки нового сумматора или вычитающего устройства целесообразно начинать с ввода названия типа создаваемого элемента в поле редактирования *Component Name*. Затем необходимо выбрать один из трех вариантов операции, которую должен выполнять формируемый элемент. С этой целью нужно воспользоваться группой кнопок с зависимой фиксацией, которая находится во встроенной панели *Operation* (рис. 47). Для генерации описания сумматора необходимо зафиксировать в нажатом состоянии кнопку *Add*. Чтобы сформировать описание вычитающего устройства, следует переключить в нажатое положение кнопку *Subtract*. Если создаваемый элемент должен выполнять операции сложения и вычитания, то нужно нажать кнопку *Add/Subtract*.

Далее в этой же диалоговой панели следует указать требуемые параметры операндов (соответствующих входных портов) генерируемого сумматора или вычитающего устройства. Установка необходимых значений этих параметров осуществляется с помощью кнопок с зависимой фиксацией и полей редактирования, расположенных во встроенных панелях *Port A Input Options* и *Port B Input Options* для порта A и порта B соответственно (рис. 47). Выбор формата представления входных данных для каждого порта производится с помощью группы кнопок с зависимой фиксацией, которые находятся в этих встроенных панелях. Чтобы входные данные были представлены в форме значения со знаком, следует установить в нажатое положение кнопку *Signed*. Если входные данные должны восприниматься как значения без знака, то необходимо переключить в нажатое состояние кнопку *Unsigned*. При нажатой кнопке *Pin* форма представления данных может выбираться в процессе работы с помощью соответствующего уровня сигнала на дополнительных входах A_SIGNED и B_SIGNED. Разрядность входных данных (операндов) для порта A и порта B указывается в поле редактирования *Port A Width* и *Port B Width* соответственно.

Для порта B входное слово данных может быть задано в виде постоянного значения. Такой режим работы указанного порта устанавливается при переключении индикатора *Constant Value*, расположенного во встроенной панели *Port B Input Options*, в состояние «Включено» (рис. 47). Значение константы, которое будет использоваться в качестве входного слова данных порта B в сгенерированном сумматоре или вычитающем устройстве, определяется в одноименном поле редактирования. Требуемое значение константы задается в форме шестнадцатеричного числа, количество разрядов которого должно соответствовать разрядности входного порта данных *Port B Width*, которая была указана ранее.

Заключительная диалоговая панель «мастера» настройки параметров ядра сумматора/вычитающего устройства *Adder/Subtractor* позволяет выбрать тип и разрядность выходных портов создаваемого элемента, а также определить необходимость использования входного переноса и режима прямой загрузки входных данных. Вид страницы *Parameters* этой диалоговой панели представлен на рис. 48.

Тип выходов в создаваемом сумматоре или вычитающем устройстве указывается с помощью группы кнопок с зависимой фиксацией, которые находятся во встроенной панели *Output Options* (рис. 48). При нажатой кнопке *Non-Registered* в формируемом элементе будут использоваться только комбинационные выходы. Если в создаваемом элементе необходимы только регистровые выходы, то следует переключить в нажатое состояние кнопку *Registered*. Для того чтобы в генерируемом сумматоре или вычитающем устройстве присутствовали как комбинационные выходы, так и регистровые, следует установить в нажатое положение кнопку *Both*. В том случае, если выбран вариант с использованием выходного регистра, то автоматически переводится в доступное состояние клавиша *Register Options…*, которая находится в этой же встроенной панели. Указанная клавиша предназначена для открытия дополнительной диалоговой панели «мастера» настройки параметризированного модуля *Adder/Subtractor*, которая позволяет выбрать входы управления (сброса, установки, инициализации) в выходном регистре и определить соотношение приоритетов сигналов на этих входах. Вид дополнительной диалоговой панели аналогичен виду вспомогательной диалоговой панели «мастера» настройки параметров двоичного дешифратора, который показан на рис. 7 (см. КиТ № 2` 2007, стр. 63). Процесс выбора управляющих входов выходного регистра осуществляется так же, как и при формировании двоичных дешифраторов, рассмотренном в первой части данной статьи.

Разрядность выходной шины данных указывается в поле редактирования *Output Width*, расположенном во встроенной панели *Output Options* (рис. 48). Для применения конвейерного способа выполнения операций в создаваемом сумматоре или вычитающем устройстве нужно воспользоваться полем редакти-

рования *Latency*. В нем указывается значение, соответствующее числу ступеней конвейера. Это значение должно находиться в пределах допустимого диапазона — от 1 до 18.

Параметризированный модуль *Adder/Subtractor* позволяет создавать сумматоры и вычитающие устройства с входами и выходами сигналов переноса (займа) и переполнения. Для включения входов и/или выходов переноса (займа) и переполнения в состав формируемого элемента нужно воспользоваться индикаторами состояния, которые находятся во встроенной панели *Carry/Overflow Options* (рис. 48). Чтобы использовать вход сигнала переноса в генерируемом сумматоре или вход сигнала займа в вычитающем устройстве, следует перевести в состояние «Включено» индикатор *Carry/Borrow Input*. Если необходимо сформировать сумматор с выходом сигнала переноса или вычитающее устройство с выходом сигнала займа, то следует установить в активное состояние индикатор *Carry/Borrow Output*. Для генерации на основе модуля *Adder/Subtractor* элементов с выходом переполнения, необходимо установить во включенное состояние индикатор *Overflow Output*. При выборе выходов переноса (займа) и переполнения нужно учитывать, что индикаторы состояния *Carry/Borrow Output* и *Overflow Output* находятся в доступном положении только при таких соотношениях разрядности входных и выходных данных, когда возможен перенос из самого старшего разряда полученного результата или переполнение.

Чтобы в формируемом сумматоре или вычитающем устройстве можно было использовать режим прямой загрузки входных данных непосредственно в выходной регистр, следует переключить индикатор *Bypass*, находящийся во встроенной панели *Bypass (Load) Options* (рис. 48), в состояние «Включено». В этом случае в состав интерфейсного блока генерируемого описания сумматора или вычитающего устройства будет включен входной порт сигнала управления непосредственной загрузкой. При одновременном использовании входа разрешения синхронизации и входа управления прямой загрузкой данных необходимо определить соотношение приоритетов сигналов, подаваемых на эти входы. Требуемое соотношение приоритетов указанных сигналов задается с помощью индикатора состояния *CE Overrides for Bypass*, который расположен в этой же встроенной панели. Если в создаваемом сумматоре или вычитающем устройстве сигнал разрешения синхронизации должен иметь более высокий приоритет, чем сигнал управления загрузкой данных, то данный индикатор нужно установить в состояние «Включено». Выключенное состояние индикатора *CE Overrides for Bypass* указывает на то, что в формируемом элементе более высоким приоритетом будет обладать сигнал на входе разрешения прямой записи данных.

В параметризированном модуле *Adder/Subtractor* пользователю предоставляется возможность выбора активного уровня сигнала на входе управления прямой загрузкой данных. Для этой цели предназначена группа кнопок с зависимой фиксацией *Bypass (Load) Sense*. Если в нажатом состоянии находится кнопка *Active High*, то активным будет считаться высокий логический уровень сигнала на входе управления прямой загрузкой данных. Чтобы установить в качестве активного уровня сигнала разрешения прямой записи данных низкий логический уровень, следует зафиксировать в нажатом состоянии кнопку *Active Low*. По умолчанию в качестве активного предлагается высокий логический уровень сигнала (в нажатом положении находится кнопка *Active High*).

В описаниях сумматоров и вычитающих устройств, формируемых на основе параметризированного модуля *Adder/Subtractor*, применяется следующая система условных обозначений входных и выходных портов:

- A[M:0] — входная шина данных A с разрядностью M+1;
- B[M:0] — входная шина данных B с разрядностью M+1;
- BYPASS — вход управления прямой загрузкой данных, представленных на входной шине B;
- C_IN — вход сигнала переноса;
- OVFL — комбинационный выход сигнала переполнения;
- Q_OVFL — регистровый выход сигнала переполнения;
- S[P:0] — комбинационные выходы генерируемого элемента (асинхронная выходная шина данных с разрядностью P+1);
- Q[P:0] — регистровые выходы формируемого элемента (синхронная выходная шина данных с разрядностью P+1);
- ADD — вход выбора выполняемой операции (суммирование или вычитание);
- B_IN — вход сигнала займа (присутствует только в описаниях устройств, выполняющих операцию вычитания);
- C_OUT — комбинационный выход сигнала переноса (присутствует только в описаниях устройств, выполняющих операцию сложения);
- Q_C_OUT — регистровый выход сигнала переноса (присутствует только в описаниях устройств, выполняющих операцию сложения);
- B _OUT — асинхронный выход сигнала займа (используется только в описаниях устройств, выполняющих операцию вычитания);
- Q_B_OUT — синхронный (регистровый) выход сигнала займа(используется только в описаниях устройств, выполняющих операцию вычитания);
- CLK — вход тактового сигнала выходного регистра;
- CE — вход сигнала разрешения синхронизации выходного регистра;
- ACLR — вход сигнала асинхронного сброса выходного регистра;
- ASET — вход сигнала асинхронной установки выходного регистра;
- SCLR — вход сигнала синхронного сброса выходного регистра;
- SSET — вход сигнала синхронной установки выходного регистра;
- AINIT — вход сигнала асинхронной инициализации выходного регистра;
- SINIT — вход сигнала синхронной инициализации выходного регистра.

Примером сумматора, сгенерированного на основе ядра *Adder/Subtractor*, является элемент add_64, выполняющий операции сложения двух 64-разрядных значений входных данных. В сформированном сумматоре используются комбинационный и регистровый выходы данных и сигнала переполнения, а также входы сигнала переноса и управления прямой записью данных. В выходном регистре задействованы асинхронные входы сброса и установки данных, вход сигнала разрешения синхронизации, а также вход синхронной инициализации. Сформированный текст VHDL-описания сумматора add_64 выглядит следующим образом:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY add_64 IS
   port (
      A: IN std_logic_VECTOR(63 downto 0);
      B: IN std_logic_VECTOR(63 downto 0);
      BYPASS: IN std_logic;
      C_IN: IN std_logic;
      OVFL: OUT std_logic;
      Q_OVFL: OUT std_logic;
      S: OUT std_logic_VECTOR(63 downto 0);
      Q: OUT std_logic_VECTOR(63 downto 0);
      CLK: IN std_logic;
      CE: IN std_logic;
      ACLR: IN std_logic;
      ASET: IN std_logic;
      SINIT: IN std_logic
      );
END add_64;
--
ARCHITECTURE add_64_a OF add_64 IS
-- synopsys translate_off
component wrapped_add_64
   port (
      A: IN std_logic_VECTOR(63 downto 0);
      B: IN std_logic_VECTOR(63 downto 0);
      BYPASS: IN std_logic;
      C_IN: IN std_logic;
      OVFL: OUT std_logic;
      Q_OVFL: OUT std_logic;
      S: OUT std_logic_VECTOR(63 downto 0);
      Q: OUT std_logic_VECTOR(63 downto 0);
      CLK: IN std_logic;
      CE: IN std_logic;
      ACLR: IN std_logic;
      ASET: IN std_logic;
      SINIT: IN std_logic
      );
end component;
--
   Configuration specification
   for all : wrapped_add_64 use entity
XilinxCoreLib.C_ADDSUB_V7_0(behavioral)
      generic map(
                  c_has_bypass_with_cin => 0,
                  c_a_type => 0,
                  c_has_sclr => 0,
                  c_sync_priority => 1,
                  c_has_aset => 1,
                  c_has_b_out => 0,
                  c_has_s => 1,
                  c_has_q => 1,
                  c_bypass_enable => 0,
```

```
                c_b_constant => 0,
                c_has_ovfl => 1,
                c_high_bit => 63,
                c_latency => 1,
                c_sinit_val                                          =>
«0000000000000000000000000000000000000000000000000000000000000001»,
                c_has_bypass => 1,
                c_pipe_stages => 1,
                c_has_sset => 0,
                c_has_ainit => 0,
                c_has_a_signed => 0,
                c_has_q_c_out => 0,
                c_b_type => 0,
                c_has_add => 0,
                c_has_sinit => 1,
                c_has_b_in => 0,
                c_has_b_signed => 0,
                c_bypass_low => 0,
                c_enable_rlocs => 0,
                c_b_value => «0»,
                c_add_mode => 0,
                c_has_aclr => 1,
                c_out_width => 64,
                c_ainit_val => «0000»,
                c_low_bit => 0,
                c_has_q_ovfl => 1,
                c_has_q_b_out => 0,
                c_has_c_out => 0,
                c_b_width => 64,
                c_a_width => 64,
                c_sync_enable => 1,
                c_has_ce => 1,
                c_has_c_in => 1
                );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_add_64
        port map (
                A => A,
                B => B,
                BYPASS => BYPASS,
                C_IN => C_IN,
                OVFL => OVFL,
                Q_OVFL => Q_OVFL,
                S => S,
                Q => Q,
                CLK => CLK,
                CE => CE,
                ACLR => ACLR,
                ASET => ASET,
                SINIT => SINIT
                );
-- synopsys translate_on
--
END add_64_a;
```

Для декларации сумматора add_64 в состав описания разрабатываемого устройства должны быть включены следующие выражения:

```
component add_64
   port (
        A: IN std_logic_VECTOR(63 downto 0);
        B: IN std_logic_VECTOR(63 downto 0);
        BYPASS: IN std_logic;
        C_IN: IN std_logic;
        OVFL: OUT std_logic;
        Q_OVFL: OUT std_logic;
        S: OUT std_logic_VECTOR(63 downto 0);
        Q: OUT std_logic_VECTOR(63 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SINIT: IN std_logic
        );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of add_64: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of add_64: component is true;
```

Описание конкретных экземпляров сумматоров вида add_64 выполняется с помощью соответствующего оператора, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_сумматора>: add_64
        port map (
                A => A,
                B => B,
                BYPASS => BYPASS,
                C_IN => C_IN,
                OVFL => OVFL,
                Q_OVFL => Q_OVFL,
                S => S,
                Q => Q,
                CLK => CLK,
                CE => CE,
                ACLR => ACLR,
                ASET => ASET,
                SINIT => SINIT
                );
```

В качестве примера вычитающего устройства, сформированного с помощью параметризированного модуля *Adder/Subtractor*, ниже приводится VHDL-описание элемента sub_36, предназначенного для вычисления разности двух значений 36-разрядных данных без знака. В состав вычитающего устройства sub_36 включен выходной регистр со входами разрешения синхронизации, синхронного и асинхронного сброса и установки. В сформированном элементе используются входы сигнала займа и управления прямой записью данных, а также выходы сигнала займа:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY sub_36 IS
   port (
        A: IN std_logic_VECTOR(35 downto 0);
        B: IN std_logic_VECTOR(35 downto 0);
        BYPASS: IN std_logic;
        B_IN: IN std_logic;
        B_OUT: OUT std_logic;
        Q_B_OUT: OUT std_logic;
        S: OUT std_logic_VECTOR(35 downto 0);
        Q: OUT std_logic_VECTOR(35 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
END sub_36;
--
ARCHITECTURE sub_36_a OF sub_36 IS
-- synopsys translate_off
component wrapped_sub_36
   port (
        A: IN std_logic_VECTOR(35 downto 0);
        B: IN std_logic_VECTOR(35 downto 0);
        BYPASS: IN std_logic;
        B_IN: IN std_logic;
        B_OUT: OUT std_logic;
        Q_B_OUT: OUT std_logic;
        S: OUT std_logic_VECTOR(35 downto 0);
        Q: OUT std_logic_VECTOR(35 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
end component;
--
-- Configuration specification
   for all : wrapped_sub_36 use entity
XilinxCoreLib.C_ADDSUB_V7_0(behavioral)
        generic map(
                c_has_bypass_with_cin => 0,
                c_a_type => 1,
                c_has_sclr => 1,
                c_sync_priority => 1,
                c_has_aset => 1,
                c_has_b_out => 1,
                c_has_s => 1,
                c_has_q => 1,
                c_bypass_enable => 0,
                c_b_constant => 0,
                c_has_ovfl => 0,
                c_high_bit => 35,
                c_latency => 1,
                c_sinit_val => «0»,
                c_has_bypass => 1,
                c_pipe_stages => 1,
                c_has_sset => 1,
                c_has_ainit => 0,
                c_has_a_signed => 0,
                c_has_q_c_out => 0,
                c_b_type => 1,
                c_has_add => 0,
                c_has_sinit => 0,
                c_has_b_in => 1,
                c_has_b_signed => 0,
                c_bypass_low => 0,
                c_enable_rlocs => 1,
                c_b_value => «0»,
                c_add_mode => 1,
                c_has_aclr => 1,
                c_out_width => 36,
                c_ainit_val => «0000»,
                c_low_bit => 0,
                c_has_q_ovfl => 0,
                c_has_q_b_out => 1,
                c_has_c_out => 0,
                c_b_width => 36,
                c_a_width => 36,
                c_sync_enable => 1,
                c_has_ce => 1,
                c_has_c_in => 0
                );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_sub_36
        port map (
                A => A,
                B => B,
                BYPASS => BYPASS,
                B_IN => B_IN,
                B_OUT => B_OUT,
                Q_B_OUT => Q_B_OUT,
                S => S,
                Q => Q,
                CLK => CLK,
                CE => CE,
                ACLR => ACLR,
                ASET => ASET,
                SCLR => SCLR,
                SSET => SSET
                );
-- synopsys translate_on
--
END sub_36_a;
```

Декларация элемента sub_36 в составе описания проектируемого устройства осуществляется следующим образом:

```
component sub_36
   port (
        A: IN std_logic_VECTOR(35 downto 0);
        B: IN std_logic_VECTOR(35 downto 0);
        BYPASS: IN std_logic;
        B_IN: IN std_logic;
        B_OUT: OUT std_logic;
        Q_B_OUT: OUT std_logic;
        S: OUT std_logic_VECTOR(35 downto 0);
        Q: OUT std_logic_VECTOR(35 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of sub_36: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of sub_36: component is true;
```

Для описания каждого экземпляра вычитающего устройства вида sub_36 использу-

ется оператор, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_вычитающего_устройства>: sub_36
        port map (
                        A => A,
                        B => B,
                        BYPASS => BYPASS,
                        B_IN => B_IN,
                        B_OUT => B_OUT,
                        Q_B_OUT => Q_B_OUT,
                        S => S,
                        Q => Q,
                        CLK => CLK,
                        CE => CE,
                        ACLR => ACLR,
                        ASET => ASET,
                        SCLR => SCLR,
                        SSET => SSET
                        );
```

Результат применения параметризированного модуля *Adder/Subtractor* для формирования описания устройства, выполняющего операции сложения и вычитания, демонстрируется на примере элемента add_sub_48. Данное устройство предназначено для сложения и вычитания двух значений 48-разрядных данных без знака, поступающих на его входы, с учетом входного переноса (займа). В сформированном элементе предусмотрена поддержка режима прямой загрузки данных, представленных на входной шине B. Результат выполнения операций сложения и вычитания отображается в виде соответствующих сигналов на комбинационных и регистровых выходах, включая сигнал выходного переноса (займа). Выходной регистр содержит асинхронные и синхронные входы сброса и установки данных, а также вход сигнала разрешения синхронизации:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synopsys translate_off
Library XilinxCoreLib;
-- synopsys translate_on
ENTITY add_sub_48 IS
   port (
        A: IN std_logic_VECTOR(47 downto 0);
        B: IN std_logic_VECTOR(47 downto 0);
        BYPASS: IN std_logic;
        C_IN: IN std_logic;
        C_OUT: OUT std_logic;
        Q_C_OUT: OUT std_logic;
        ADD: IN std_logic;
        S: OUT std_logic_VECTOR(47 downto 0);
        Q: OUT std_logic_VECTOR(47 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
END add_sub_48;
--
ARCHITECTURE add_sub_48_a OF add_sub_48 IS
-- synopsys translate_off
component wrapped_add_sub_48
   port (
        A: IN std_logic_VECTOR(47 downto 0);
        B: IN std_logic_VECTOR(47 downto 0);
        BYPASS: IN std_logic;
        C_IN: IN std_logic;
        C_OUT: OUT std_logic;
        Q_C_OUT: OUT std_logic;
        ADD: IN std_logic;
        S: OUT std_logic_VECTOR(47 downto 0);
        Q: OUT std_logic_VECTOR(47 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
end component;
--
-- Configuration specification
   for all : wrapped_add_sub_48 use entity
XilinxCoreLib.C_ADDSUB_V7_0(behavioral)
        generic map(
                        c_has_bypass_with_cin => 0,
                        c_a_type => 1,
                        c_has_sclr => 1,
                        c_sync_priority => 1,
                        c_has_aset => 1,
                        c_has_b_out => 0,
                        c_has_s => 1,
                        c_has_q => 1,
                        c_bypass_enable => 0,
                        c_b_constant => 0,
                        c_has_ovfl => 0,
                        c_high_bit => 47,
                        c_latency => 1,
                        c_sinit_val => «0»,
                        c_has_bypass => 1,
                        c_pipe_stages => 1,
                        c_has_sset => 1,
                        c_has_ainit => 0,
                        c_has_a_signed => 0,
                        c_has_q_c_out => 1,
                        c_b_type => 1,
                        c_has_add => 1,
                        c_has_sinit => 0,
                        c_has_b_in => 0,
                        c_has_b_signed => 0,
                        c_bypass_low => 0,
                        c_enable_rlocs => 0,
                        c_b_value => «0»,
                        c_add_mode => 2,
                        c_has_aclr => 1,
                        c_out_width => 48,
                        c_ainit_val => «0000»,
                        c_low_bit => 0,
                        c_has_q_ovfl => 0,
                        c_has_q_b_out => 0,
                        c_has_c_out => 1,
                        c_b_width => 48,
                        c_a_width => 48,
                        c_sync_enable => 1,
                        c_has_ce => 1,
                        c_has_c_in => 1
        );
-- synopsys translate_on
BEGIN
-- synopsys translate_off
U0 : wrapped_add_sub_48
        port map (
                        A => A,
                        B => B,
                        BYPASS => BYPASS,
                        C_IN => C_IN,
                        C_OUT => C_OUT,
                        Q_C_OUT => Q_C_OUT,
                        ADD => ADD,
                        S => S,
                        Q => Q,
                        CLK => CLK,
                        CE => CE,
                        ACLR => ACLR,
                        ASET => ASET,
                        SCLR => SCLR,
                        SSET => SSET
                        );
-- synopsys translate_on
--
END add_sub_48_a;
```

При использовании элемента add_sub_48 в качестве компонента проектируемого устройства необходимо поместить в раздел декларации VHDL-описания следующие выражения:

```
component add_sub_48
   port (
        A: IN std_logic_VECTOR(47 downto 0);
        B: IN std_logic_VECTOR(47 downto 0);
        BYPASS: IN std_logic;
        C_IN: IN std_logic;
        C_OUT: OUT std_logic;
        Q_C_OUT: OUT std_logic;
        ADD: IN std_logic;
        S: OUT std_logic_VECTOR(47 downto 0);
        Q: OUT std_logic_VECTOR(47 downto 0);
        CLK: IN std_logic;
        CE: IN std_logic;
        ACLR: IN std_logic;
        ASET: IN std_logic;
        SCLR: IN std_logic;
        SSET: IN std_logic
        );
end component;
--
-- FPGA Express Black Box declaration
attribute fpga_dont_touch: string;
attribute fpga_dont_touch of add_sub_48: component is «true»;
--
-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of add_sub_48: component is true;
```

Конкретные экземпляры компонента add_sub_48, используемые в составе описания разрабатываемого устройства, создаются с помощью оператора, шаблон которого имеет следующий вид:

```
<идентификатор_экземпляра_устройства_сложения_вычитания>: add_sub_48
        port map (
                        A => A,
                        B => B,
                        BYPASS => BYPASS,
                        C_IN => C_IN,
                        C_OUT => C_OUT,
                        Q_C_OUT => Q_C_OUT,
                        ADD => ADD,
                        S => S,
                        Q => Q,
                        CLK => CLK,
                        CE => CE,
                        ACLR => ACLR,
                        ASET => ASET,
                        SCLR => SCLR,
                        SSET => SSET
                        );
```

Подготовка описаний умножителей на основе параметризированных модулей с помощью средств CORE Generator

В рассматриваемой версии генератора параметризированных модулей CORE Generator представлено три модификации ядра *Multiplier Generator* (помимо устаревших модификаций), предназначенного для формирования умножителей. Эти модификации входят в состав группы ядер *Math Functions*, предназначенных для реализации математических функций. Каждый из трех вариантов ядра *Multiplier Generator* предоставляет пользователю различные возможности, поэтому далее поочередно рассматриваются актуальные версии данного параметризированного модуля.

Версия v7.0 ядра *Multiplier Generator* предназначена для генерации описаний умножителей с различной разрядностью, реализуемых в ПЛИС следующих семейств: Spartan-II; Spartan-IIE; Spartan-3; Spartan-3E; Spartan-3A; Virtex; QPRO Virtex Rad-Hard; QPRO Virtex Hi-Rel; Virtex-E; QPRO Virtex-E Military; Virtex-II; Virtex-II PRO и Virtex-4. Отличительными особенностями данной версии рассматриваемого параметризированного модуля являются:

- возможность формирования умножителей различного типа (параллельного или последовательного);
- наличие опции генерации элементов, выполняющих умножение на заданное зна-

чение (коэффициент), указываемое пользователем при определении параметров создаваемого умножителя, или загружаемое в процессе работы;
- возможность индивидуального выбора разрядности перемножаемых данных (входных шин данных) в диапазоне от 1 до 64 двоичных разрядов;
- поддержка разрядности выходных данных в диапазоне от 1 до 129 разрядов;
- поддержка операций умножения, выполняемых над входными данными, представленными в виде значений со знаком или без знака;
- выборочное использование сигналов подтверждения (квитирования) при выполнении операций умножения;
- возможность генерации описаний умножителей, реализуемых на базе различных ресурсов ПЛИС (таблиц преобразования LUT, аппаратных блоков умножения Multiplier Blocks или секций XtremeDSP);
- применение конвейерного метода выполнения операций умножения;
- использование входных регистров данных в составе разрабатываемого умножителя (по выбору пользователя);
- возможность генерации умножителей с асинхронными (комбинационными) и синхронными (регистровыми) выходами;
- возможность выборочного использования в выходном регистре создаваемых умножителей синхронного и асинхронного входов сброса, а также входа сигнала разрешения синхронизации;
- поддержка генерации описаний умножителей в виде макросов с относительным размещением Relationally Placed Macro (RPM) различной формы.

«Мастер» настройки параметров ядра *Multiplier Generator* версии v7.0, в зависимости от типа создаваемого умножителя, содержит от четырех до шести диалоговых панелей. Стартовая диалоговая панель данного «мастера» позволяет выбрать тип формируемого умножителя, вид ресурсов кристалла, используемых для его реализации, и форму представления его описания. Вид страницы

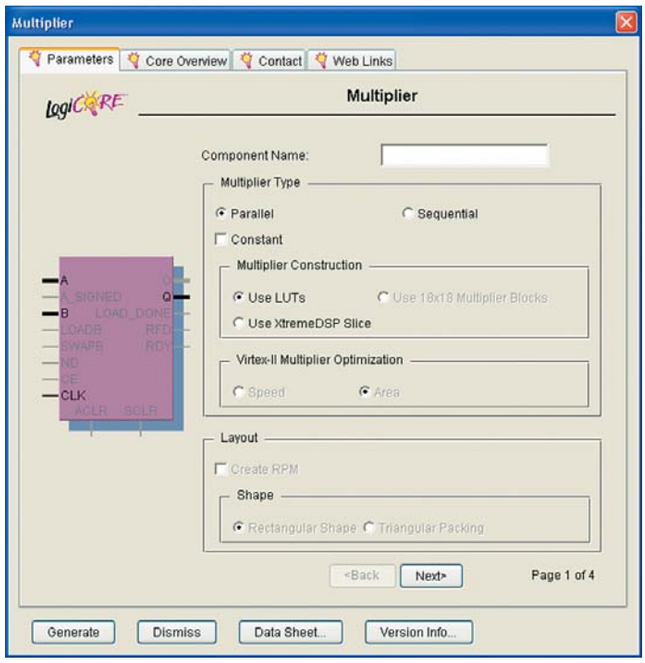

Рис. 49. Страница Parameters стартовой диалоговой панели «мастера» настройки параметров ядра умножителя Multiplier Generator версии v7.0

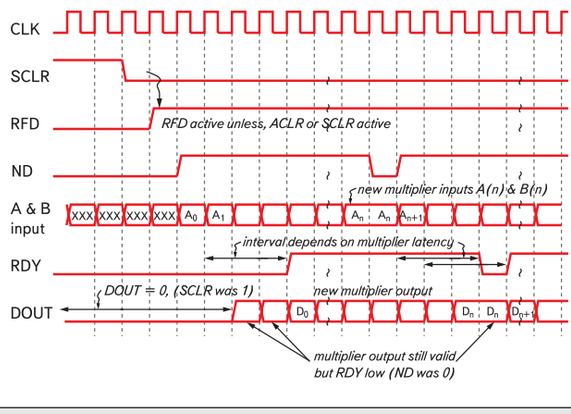

Рис. 50. Временные диаграммы, поясняющие функционирование параллельных умножителей, формируемых на основе параметризированного модуля Multiplier Generator версии v7.0

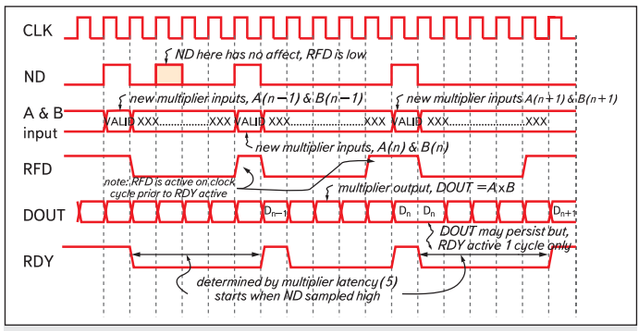

Рис. 51. Временные диаграммы, демонстрирующие работу последовательных умножителей, генерируемых на основе параметризированного модуля Multiplier Generator версии v7.0, с минимальным количеством ступеней конвейера

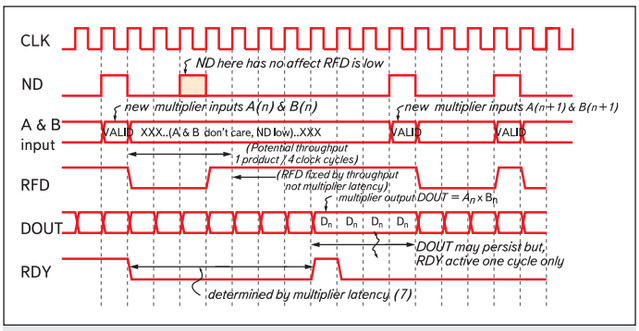

Рис. 52. Временные диаграммы, поясняющие функционирование последовательных умножителей, формируемых на основе параметризированного модуля Multiplier Generator версии v7.0, с максимальным количеством ступеней конвейера

Parameters этой диалоговой панели показан на рис. 49.

Выбор типа генерируемого умножителя осуществляется с помощью двух кнопок с зависимой фиксацией, которые расположены во встроенной панели *Multiplier Type* на странице *Parameters* стартовой диалоговой панели. Для создания описания параллельного умножителя нужно зафиксировать в нажатом состоянии кнопку *Parallel*. При этом будет сформировано описание умножителя, в котором вычисление всех частичных произведений выполняется параллельно. Такая архитектура умножителя позволяет сократить суммарное время вычисления произведения. На рис. 50 приведены временные диаграммы, поясняющие функционирование параллельных умножителей, формируемых на основе параметризированного модуля *Multiplier Generator* версии v7.0.

В последовательных умножителях процесс формирования окончательного результата представляет собой цепочку поочередных вычислений частичных произведений с их последующим суммированием. Таким образом, в суммарное время вычисления произведения вносится дополнительная задержка, длительность которой зависит от количества вычисляемых частичных произведений и используемых при этом конвейерных регистров. Временные диаграммы, демонстрирующие работу последовательных умножителей, генерируемых на основе параметризированного модуля *Multiplier Generator* версии v7.0, с минимальным и максимальным количеством конвейерных регистров (ступеней конвейера) показаны на рис. 51, 52 соответственно. Чтобы сформировать описание последовательного умножителя, следует установить в нажатое положение кнопку *Sequential*.

Для генерации описаний элементов, выполняющих умножение входных данных на некоторое заданное значение (коэффициент), следует установить в состояние «Включено» индикатор *Constant*, который также находится во встроенной панели *Multiplier Type* (рис. 49). Умножители входных данных на заданные коэффициенты часто применяются при разработке устройств цифровой обработки сигналов.

Вид ресурсов ПЛИС, на основе которых реализуется создаваемый умножитель, выбирается с помощью группы кнопок с зависимой фиксацией *Multiplier Construction*. Для реализации формируемого умножителя на базе таблиц преобразования LUT необходимо зафиксировать в нажатом состоянии кнопку *Use LUTs*. Данный вариант реализации целесообразно выбирать, если умножитель разрабатывается для использования в кристаллах, не содержащих аппаратных блоков умножения. Кроме того, на основе таблиц преобразования LUT реализуются последовательные умножители и элементы, выполняющие умножение входных значений на заданные коэффициенты. Чтобы сгенерировать описание умножителя, реализуемого на основе аппаратных 18-разрядных блоков умножения Multiplier Blocks, следует переключить в нажатое положение кнопку *Use 18×18 Multiplier Blocks*. Этот вариант реализации рекомендуется выбирать при создании параллельных умножителей, предназначенных для использования в ПЛИС, которые содержат указанные аппаратные блоки. В этом случае достигается повышенная производительность разрабатываемого устройства при оптимальном использовании аппаратных ресурсов кристалла. Если формируемый умножитель предназначен для применения в кристаллах семейства Virtex-4, то для его реализации можно использовать блоки XtremeDSP. Для этого нужно нажать кнопку *Use XtremeDSP Slice*.

При использовании аппаратных 18-разрядных блоков умножения Multiplier Blocks для реализации создаваемого умножителя можно выбрать один из двух критериев дополнительной оптимизации. Выбор осуществляется с помощью двух кнопок с зависимой фиксацией *Virtex-II Multiplier Optimization*, которые также расположены во встроенной панели *Multiplier Type* (рис. 49). При нажатой кнопке *Speed* критерием оптимизации является достижение максимального быстродействия умножителя. Если в нажатом состоянии находится кнопка *Area*, то в качестве критерия оптимизации выбирается минимизация используемой области кристалла.

Выбор формы представления описания создаваемого умножителя для последующей реализации производится с помощью индикатора состояния *Create RPM* и группы кнопок с зависимой фиксацией *Shape*, расположенных во встроенной панели *Layout* (рис. 49). Чтобы сгенерировать описание умножителя в виде макроса с относительным размещением, нужно установить индикатор *Create RPM* в состояние «Включено». При этом становится доступной группа кнопок *Shape*, которые позволяют выбрать форму топологии элементов создаваемого макроса в кристалле. Когда в нажатом состоянии зафиксирована кнопка *Rectangular Shape*, форма топологии элементов генерируемого макроса близка к прямоугольной. В такой форме элементы макроса располагаются достаточно свободно. Более высокую плотность размещения элементов макроса можно получить, переключив в нажатое состояние кнопку *Triangular Packing*. Следует обратить внимание на то, что при формировании описаний последовательных умножителей в виде макросов с относительным размещением во включенное состояние автоматически переводится кнопка *Triangular Packing*.

Если в стартовой диалоговой панели выбран вариант умножителя, в котором в качестве второго сомножителя используется заданное значение, то второй диалоговой панелью «мастера» настройки параметров ядра умножителя *Multiplier Generator* версии v7.0 будет являться панель, предназначенная для определения параметров коэффициента. Вид этой диалоговой панели представлен на рис. 53.

Значение коэффициента, используемого в умножителе, указывается в десятичной форме в поле редактирования *Constant Value*, которое находится во встроенной панели *Coefficient* (рис. 53). Для формирования описания умножителя, в котором новое значение коэффициента может загружаться в процессе работы, следует установить в состояние «Включено» индикатор *Reloadable*, расположенный в этой же встроенной панели. В таком умножителе новые значения коэффициента задаются в виде совокупности сигналов на входной шине данных B, с которой затем записываются во внутреннюю память при активном уровне сигнала на входе управления загрузкой LOADB. На рис. 54 приведены временные диаграммы, поясняющие функционирование умножителей входных данных на коэффициенты, значения которых загружаются во внутреннюю память в процессе функционирования устройства.

Если в формируемом умножителе необходим выход сигнала, информирующего о состоянии процесса загрузки нового значения коэффициента, то следует установить во включенное состояние индикатор LOAD_DONE Output, который также представлен во встроенной панели *Coefficient*. Высокий уровень сигнала на этом выходе указывает на окончание процесса загрузки нового значения коэффициента.

Группа кнопок с зависимой фиксацией *Reload Options*, расположенная в этой же встроенной панели (рис. 53), предназначена для выбора режима работы создаваемого умножителя при загрузке нового значения коэффициента. Если нажата кнопка *Stop During Reload*, то будет сгенерирован умножитель с одним банком внутренней памяти, и во время загрузки нового значения коэффициента выполнение текущей операции умножения прекращается. При этом данные на выходной шине умножителя (результаты текущей выполняемой операции) становятся недостоверными. Временные диаграммы, приведенные на рис. 54, соответствуют режиму *Stop During Reload*. Чтобы сформировать умножитель с двумя банками внутренней памяти, нужно переключить в нажатое состояние кнопку *Continue During Reload*. При этом в создаваемом умножителе будет присутствовать дополнительный вход SWAPB, сигнал на котором используется для переключения банков памяти. Применение двух банков памяти позволяет выполнять загрузку новых значений коэффициента, не прерывая выполнение текущей операции умножения. В этом случае в процессе загрузки новых значений коэффициента на выходе умножителя будут представлены достоверные данные. На рис. 55 показаны временные диаграммы, которые демонстрируют работу умножителей значений входных данных на коэффициенты, использующих два переклю-

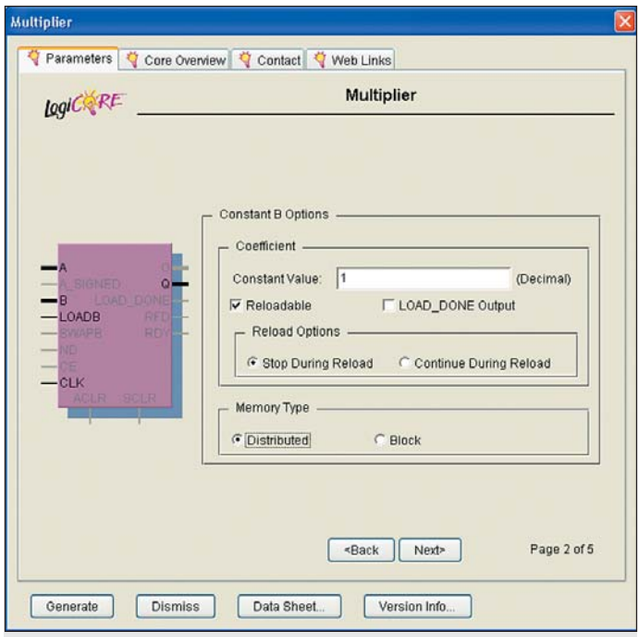

Рис. 53. Диалоговая панель Constant B Options «мастера» настройки параметров ядра умножителя Multiplier Generator версии v7.0, предназначенная для определения параметров задаваемого коэффициента (страница Parameters)

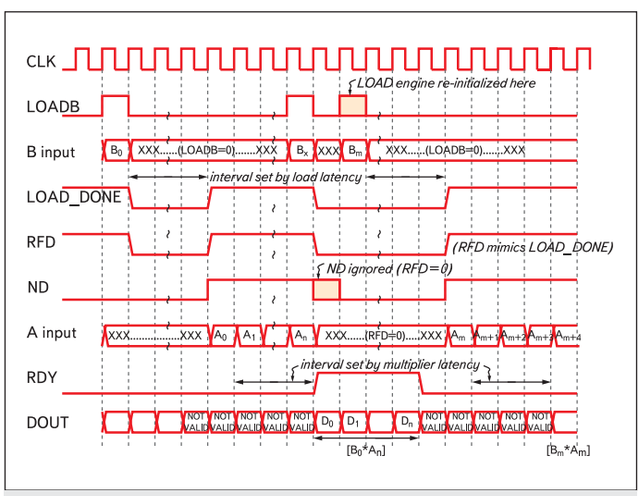

Рис. 54. Временные диаграммы, демонстрирующие работу умножителей входных данных на коэффициенты, значения которых загружаются во внутреннюю память в процессе функционирования устройства

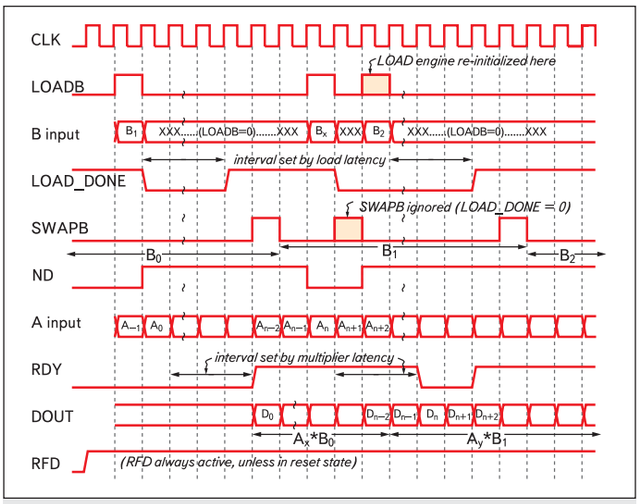

Рис. 55. Временные диаграммы, демонстрирующие работу умножителей значений входных данных на коэффициенты, использующих два переключаемых банка внутренней памяти для загрузки значений коэффициентов

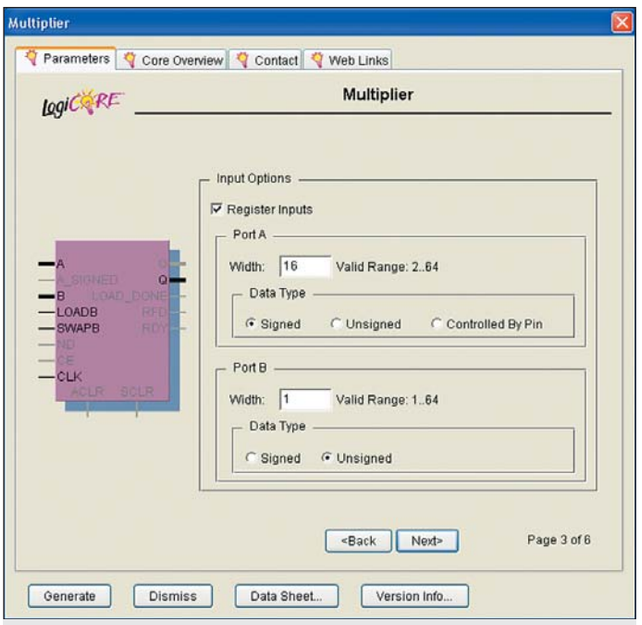

Рис. 56. Диалоговая панель Input Options «мастера» настройки параметров ядра Multiplier Generator версии v7.0, предназначенная для определения конфигурации входных шин данных формируемого умножителя (страница Parameters)

чаемых банка внутренней памяти для загрузки новых значений коэффициентов.

Тип памяти, применяемой в создаваемом умножителе, выбирается с помощью группы кнопок с зависимой фиксацией *Memory Type* (рис. 53). При нажатой кнопке *Distributed* используется распределенная память ПЛИС, реализуемая на базе таблиц преобразования LUT. Чтобы использовать ресурсы блочной памяти кристалла Block RAM, следует переключить в нажатое состояние кнопку *Block*. Этот вариант доступен только при создании умножителей, предназначенных для реализации в ПЛИС, обладающих блочной памятью.

Следующая диалоговая панель *Input Options* «мастера» настройки параметров ядра *Multiplier Generator* версии v7.0, вид которой приведен на рис. 56, предназначена для определения конфигурации входных шин данных формируемого умножителя.

Для включения в состав разрабатываемого умножителя входных регистров нужно перевести во включенное состояние индикатор *Register Inputs*. В этом случае регистры будут установлены на входных шинах данных A и B, а также на входах A_SIGNED и ND. Разрядность входных шин данных A и B указывается в полях редактирования Width, расположенных во встроенных панелях *Port A* и *Port B* соответственно. Форма представления значений для каждой входной шины данных выбирается с помощью соответствующей группы кнопок с зависимой фиксацией *Data Type*. Если в нажатом состоянии зафиксирована кнопка *Signed*, то входные данные интерпретируются как значения со знаком. При нажатой кнопке *Unsigned* входные данные воспринимаются как операнды без знака. Для входной шины A форма представления данных может выбираться в процессе работы с помощью соответствующего уровня сигнала на дополнительном входе A_SIGNED.

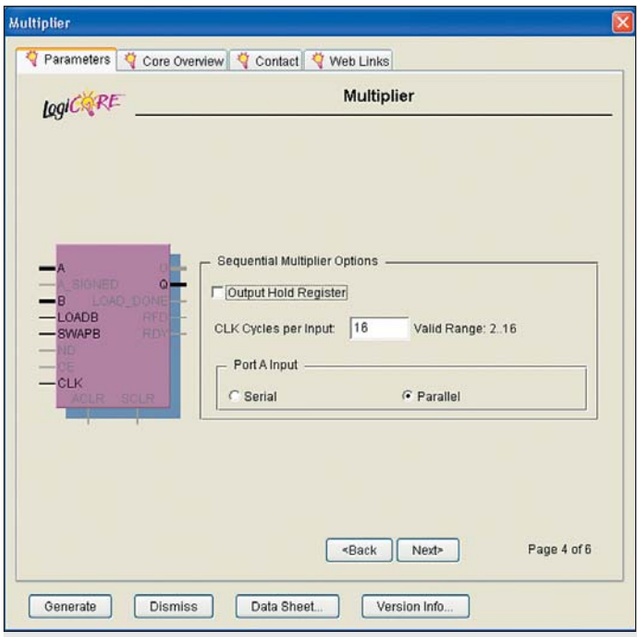

Рис. 57. Диалоговая панель Sequential Multiplier Options «мастера» настройки параметров ядра Multiplier Generator версии v7.0, предназначенная для выбора специальных опций последовательного умножителя (страница Parameters)

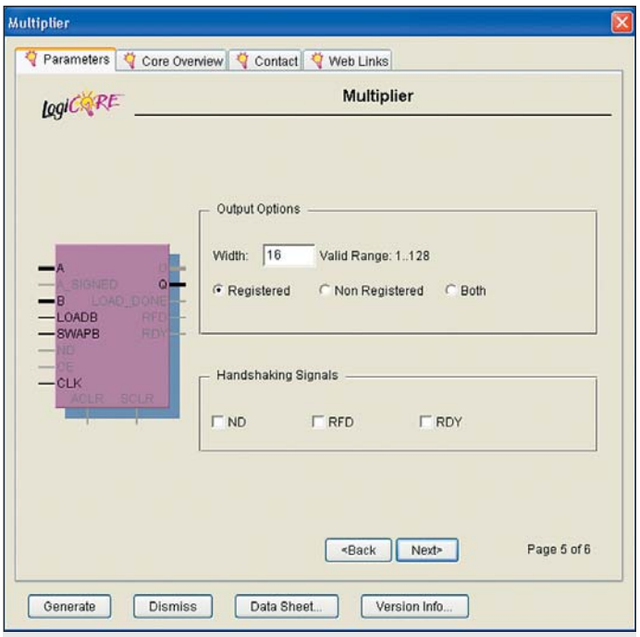

Рис. 58. Диалоговая панель «мастера» настройки ядра Multiplier Generator версии v7.0, предназначенная для определения параметров выходных шин формируемого умножителя и выбора сигналов подтверждения (страница Parameters)

При высоком логическом уровне сигнала на этом входе данные, представленные на входной шине A, воспринимаются как значения со знаком. Если на входе A_SIGNED присутствует низкий логический уровень сигнала, то данные на входной шине A интерпретируются как значения без знака. Для создания умножителя с динамически изменяемой формой представления данных входной шины A нужно перевести во включенное состояние кнопку *Controlled by Pin*.

Если в стартовой диалоговой панели «мастера» настройки параметров ядра *Multiplier Generator* был выбран последовательный тип создаваемого умножителя, то очередной будет открыта диалоговая панель *Sequential Multiplier Options*, вид которой изображен на рис. 57.

Данная диалоговая панель позволяет установить значения параметров, которые используются только при генерации описаний последовательных умножителей. Значение параметра *Output Hold Register* определяет состояние выходного регистра последовательного умножителя в процессе вычисления произведения. Если одноименный индикатор установлен в состояние «Включено», то состояние выходного регистра не изменяется в процессе вычислений до получения окончательного результата произведения. При выключенном состоянии индикатора *Output Hold Register* в выходной регистр будут записываться промежуточные результаты вычислений, которые отображаются на комбинационных выходах умножителя.

В поле редактирования *CLK Cycles per Input* нужно указать количество периодов тактового сигнала, выделяемых для вычисления одного частичного произведения (результата перемножения входных данных с меньшей разрядностью). Значение данного параметра косвенно определяет количество шин с меньшей разрядностью, на которое разделяются входные шины данных в последовательном умножителе.

Группа кнопок с зависимой фиксацией *Port A Input* предназначена для выбора типа входного порта A последовательного умножителя. Для использования последовательного порта данных A в создаваемом умножителе нужно зафиксировать в нажатом состоянии кнопку *Serial*, а для параллельного — кнопку *Parallel*.

Предпоследняя диалоговая панель «мастера» настройки ядра *Multiplier Generator* версии v7.0 предназначена для определения параметров выходных шин формируемого умножителя и выбора сигналов подтверждения (квитирования). Вид этой панели представлен на рис. 58.

Разрядность выходных шин данных указывается в поле редактирования *Width*, которое представлено во встроенной панели *Output Options*. Тип выходных портов генерируемого умножителя указывается с помощью трех кнопок с зависимой фиксацией, расположенных в этой же встроенной панели (рис. 58). Если во включенном состоянии находится кнопка *Non-Registered Output*, то в формируемом умножителе будут использоваться только комбинационные (асинхронные) выходы. Для создания умножителя с регистровыми (синхронными) выходами нужно зафиксировать в нажатом состоянии кнопку *Registered Output*. При включенной кнопке *Both* в генерируемом умножителе будут представлены как комбинационные, так и регистровые выходы.

Выбор входов и выходов сигналов подтверждения осуществляется с помощью трех индикаторов состояния, расположенных во встроенной панели *Handshaking Signals*. При включенном состоянии индикатора *ND* в создаваемом умножителе будет задействован вход сигнала New Data (ND), высокий логический уровень которого сообщает о присутствии новых входных данных. Для использования в формируемом умножителе выхода сигнала Ready for Data (RFD), информирующего о готовности приема новых данных, следует установить индикатор *RFD* в состояние «Включено». Чтобы в разрабатываемом умножителе присутствовал выход сигнала Ready, который подтверждает готовность (достоверность) выходных данных, нужно установить во включенное состояние индикатор *RDY*.

Заключительная диалоговая панель «мастера» настройки параметров ядра *Multiplier Generator* версии v7.0, вид которой приведен на рис. 59, позволяет выбрать требуемые опции выходного регистра и конвейерной организации формируемого умножителя.

Группа кнопок с зависимой фиксацией *Pipeline Options* предназначена для выбора параметров конвейерной организации создаваемого умножителя. При нажатой кнопке *Maximum Pipelining* на каждой ступени умножителя устанавливаются дополнительные регистры. Когда в нажатом состоянии находится кнопка *Minimum Pipelining*, в структуре формируемого умножителя используется минимальное количество регистров.

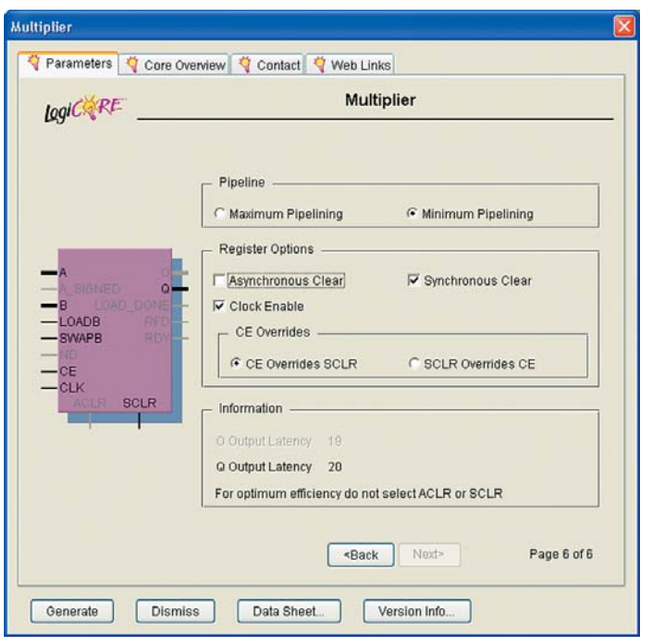

Рис. 59. Заключительная диалоговая панель «мастера» настройки параметров ядра Multiplier Generator версии v7.0, предназначенная для определения опций выходного регистра и конвейерной организации формируемого умножителя (страница Parameters)

Определение параметров выходного регистра создаваемого умножителя производится с помощью индикаторов состояния и кнопок, расположенных во встроенной панели *Register Options* (рис. 59). Если в выходном регистре требуются входы асинхронного и/или синхронного сброса, то следует установить в состояние «Включено» индикаторы *Asynchronous Clear* и/или *Synchronous Clear* соответственно. Чтобы включить в состав умножителя выходной регистр с входом разрешения синхронизации, следует воспользоваться индикатором состояния *Clock Enable*, находящимся в этой же встроенной панели. При наличии в выходном регистре входов синхронного сброса и разрешения синхронизации необходимо определить приоритеты соответствующих сигналов с помощью группы кнопок с зависимой фиксацией *CE Overrides*. Чтобы сигнал на входе разрешения синхронизации имел более высокий приоритет по сравнению с сигналом на входе синхронного сброса, следует зафиксировать в нажатом состоянии кнопку *CE Overrides SCLR*. Для установки обратного соотношения приоритетов указанных сигналов нужно переключить в нажатое состояние кнопку *SCLR Overrides CE*.

В заключительной диалоговой панели «мастера» настройки параметров ядра *Multiplier Generator* версии v7.0 представлена также встроенная информационная панель *Information*, в которой отображаются сведения о задержках выходных сигналов и рекомендации, позволяющие повысить производительность разрабатываемого умножителя.

Продолжение следует