Лабораторная работа 4

Синтаксические конструкции

Библиотеки (library) и пакеты (package)

Библиотеки и пакеты необходимы для обеспечения удобной работы с проектом. Библиотеки состоят из пакетов, а пакеты, в свою очередь, содержат объявления часто используемых типов данных, констант, сигналов, компонентов, процедур и функций. Так, например, тип данных **std_logic** определен в пакете **std_logic_1164** библиотеки **ieee**.

Для того чтобы использовать необходимый пакет VHDL необходимо в VHDL файле указать, какую библиотеку и пакет(ы) в ней использовать, с помощью ключевых слов **library** и **use**.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee.numeric_std.all;
```

Расширение **all** показывает, что пакет std_logic_1164 необходимо использовать полностью. Есть возможность использовать только отдельные составляющие пакета.

В настоящей лабораторной работе используется пакет **numeric_std** библиотеки **ieee**. В этом пакете определены знаковые и беззнаковые целые числа и операции над ними. Также можно применять пакет **std_logic_arith**, который не является частью стандарта IEEE, но де факто – промышленный стандарт, который появился до пакета **numeric_std**.

После подключения пакета **numeric_std** или **std_logic_arith** появляется возможность выполнять, например, сложение или умножение двух чисел, имеющих тип std_logic_vector. Для этого при выполнении операции необходимо явно указать, как интерпретируются сигналы: со знаком (signed) или без знака (unsigned).

```
res1 <= unsigned(s1) + unsigned(s2);
res2 <= signed(s1) * signed(s2);
```

Изменение разрядности чисел в дополнительном коде

Пусть в результате на 4 бита больше, чем в исходном числе X. Возможны следующие варианты:

1. $X \geq 0$. Необходимо добавить четыре нуля слева от X. Синтаксис операции будет следующий:

```
res_p <= "0000" & x;
```

Знаком `&` выполняется объединение двух векторов в один, разрядность которого равна сумме разрядностей исходных векторов.

2. $X < 0$. Необходимо добавить четыре единицы слева от X. Синтаксис операции будет следующий:

```
res_n <= "1111" & x;
```

3. Заранее ничего не известно о знаке X: число может быть либо положительным, либо отрицательным. Необходимо добавить четыре копии знаковых разряда слева от X. Синтаксис операции будет следующий:

```
res_pn <= x(3) & x(3) & x(3) & x(3) & x;
```

Для сокращения записи можно воспользоваться функцией **resize()** (или **sxt()** из пакета **std_logic_arith**), которая выполняет аналогичную операцию. Первый параметр функции – преобразуемое число, второй – необходимое количество разрядов в результате:

```
res_pn <= sxt(x,8);-- std_logic_arith
res_pn <= resize(unsigned(x),8);-- numeric_std
```

```
res_pn <= resize(signed(x),8);-- numeric_std
```

Задание к лабораторной работе 4

Реализовать операции суммы, разности, сдвига, циклического сдвига и умножения на логике общего назначения и на DSP элементах (используя Xilinx IP Core Generator) в ПЛИС. В качестве входов использовать кнопки и слайдеры на плате, в качестве выходов – светодиодные индикаторы.

Необходимо реализовать:

1. Сумматор двух неотрицательных трехразрядных чисел.
   Цифровое устройство должно иметь два трехразрядных входа (слайдеры) и четыре выхода (светодиоды). При выполнении операции сложения, необходимо предусмотреть возможность переполнения разрядности. Для этого – увеличить число разрядов в слагаемых на 1.

   Использовать следующий шаблон для объявления интерфейса модуля:

```
----------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity lab41 is
    Port ( x1 : in STD_LOGIC_VECTOR (2 downto 0);
           x2 : in STD_LOGIC_VECTOR (2 downto 0);
           y : out STD_LOGIC_VECTOR (3 downto 0));
end lab41;
----------------------------------
```

2. Сумматор знакового семиразрядного числа и константы (cnst = 3).
   Цифровое устройство должно иметь один 7-разрядный вход *x* для задания входного числа (слайдеры) и один 8-разрядный выход *y* (светодиоды). Сложение должно выполняться с учетом знака, т.е. старший разряд каждого слагаемого в сумме – знаковый. Разрядности обоих операндов необходимо увеличить до 8 при выполнении операции сложения, причем увеличение разрядности должно быть выполнено с учетом знака.
   Создать тестбенч для разработанного модуля, выполнить моделирование его работы.

   Использовать следующий шаблон для модуля:

```
----------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;

entity lab42 is
    Port ( x : in STD_LOGIC_VECTOR (6 downto 0);
           y : out STD_LOGIC_VECTOR (7 downto 0));
end lab42;

architecture a of lab42 is
    constant cnst : std_logic_vector(7 downto 0) := X"03"; -- Знаковая
константа 3
begin
    y <= signed(x(6) & x) + signed(cnst);
--    y <= signed(sxt(x,8)) + signed(cnst);
end a;
----------------------------------
```

   Использовать следующий шаблон для тестбенча:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench1 is
--  Port ( );
```

```
end test_bench1;

architecture a of test_bench1 is
    signal x        : std_logic_vector(6 downto 0) := (others => '0');
    signal y        : std_logic_vector(7 downto 0) := (others=> '0');

    component lab42 is
    Port ( x : in STD_LOGIC_VECTOR (6 downto 0);
           y : out STD_LOGIC_VECTOR (7 downto 0));
    end component lab42;
begin
  ----------------------------------------------------------------------
  -- Instantiate the DUT
  ----------------------------------------------------------------------
  dut : lab42
    port map (
        -- Inputs
        x => x,
        -- Outputs
        y  => y);

  ----------------------------------------------------------------------
  -- Generate inputs
  ----------------------------------------------------------------------
  process begin
    x <= "0000001", "1000001" after 200 ns, "1111101" after 400 ns, "0001111"
after 600 ns, "1001111" after 800 ns;
    wait;
  end process;
end a;
--------------------------------------
```

3. Блок, выполняющий циклический сдвиг четырехразрядного числа влево на два разряда.
Цифровое устройство должно иметь один четырехразрядный вход для задания входного числа (слайдеры) и два четырехразрядных выхода (светодиоды). Первый выход – для отображения исходного числа, а второй – для отображения результата циклического сдвига на два разряда влево. Исходное число должно выводиться в младших разрядах выходного сигнала.

Использовать следующий шаблон для объявления интерфейса модуля:

```
--------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab43 is
  Port (x : in  std_logic_vector(3 downto 0);
        y : out std_logic_vector(7 downto 0));
end lab43;
architecture a of lab43 is
begin
    y(3 downto 0) <= x;
...
--------------------------------------
```

4. Умножитель двух неотрицательных двухразрядных чисел.
Цифровое устройство должно иметь два двухразрядных входа для задания входных чисел (слайдеры) и один семиразрядный выход (семисегментный индикатор) для отображения результата умножения.

Использовать следующий шаблон для объявления интерфейса модуля:

```
--------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;

entity lab44 is
    Port ( x1 : in STD_LOGIC_VECTOR (1 downto 0);
           x2 : in STD_LOGIC_VECTOR (1 downto 0);
```

```
           an : out STD_LOGIC_VECTOR (3 downto 0);
           y  : out STD_LOGIC_VECTOR (6 downto 0));
end lab44;
-------------------------------------
```

5. *Блок, по команде выполняющий операции логического сдвига, арифметического сдвига и циклического сдвига входного 6-разрядного числа. Все операции должны выполняться в двух направлениях (влево и вправо) на один разряд.

   Все операции должны выполняться в двух направлениях: влево и вправо на один разряд. Цифровое устройство должно иметь одноразрядный вход, определяющий направление сдвига (lr, слайдер), двухразрядный вход, определяющий тип сдвига (cmd, слайдеры), 6-разрядный вход для задания входного (сдвигаемого) числа (din, слайдеры), а также 6-разрядный выход для отображения входного числа (dout_din, светодиоды) и 6-разрядный выход для отображения результата сдвига (dout_shifted, светодиоды). Неиспользуемые комбинации cmd должны оставлять входное число неизменным.

   Использовать следующий шаблон для объявления интерфейса модуля и архитектуры:

```
-------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab45 is
    Port ( cmd : in STD_LOGIC_VECTOR (1 downto 0);
           lr : in STD_LOGIC;
           din : in STD_LOGIC_VECTOR (5 downto 0);
           dout_din     : out STD_LOGIC_VECTOR (5 downto 0);
           dout_shifted : out STD_LOGIC_VECTOR (5 downto 0));
end lab45;

architecture a of lab45 is

begin
    dout_din <= din;

    process (cmd,lr,din) begin
        if lr='0' then -- Сдвиг влево
            case cmd is
                when "00"   => -- Логический
                    dout_shifted <=
                when "01"   => -- Арифметический
                    dout_shifted <=
                when "10" => -- Циклический
                    dout_shifted <=
                when others => -- Нет сдвига
                    dout_shifted <=
            end case;
        else-- Сдвиг вправо
            case cmd is
                when "00"   => -- Логический
                    dout_shifted <=
                when "01"   => -- Арифметический
                    dout_shifted <=
                when "10" => -- Циклический
                    dout_shifted <=
                when others => -- Нет сдвига
                    dout_shifted <=
            end case;
        end if;
    end process;
end a;
-------------------------------------
```

6. *Используя Xilinx IP Core Generator (IP catalog) умножитель двух 16-разрядных чисел: на логике общего назначения и на DSP элементах.

   Без запуска проекта в отладочной плате. Отдельный проект для каждого типа умножителя. Для создания собственно умножителя использовать Xilinx IP Core Generator (IP catalog). Использовать документ *"Работа с Vivado.docx*". В данном подзадании не нужно генерировать файл прошивки, но необходимо выполнить этап синтеза (synthesis). Результатом выполнения задания является количество таблиц истинности и встроенных DSP элементов для каждой из версий умножителя.

7. **Реализовать умножитель 8-битного беззнакового целого на рациональную константу 5/7.

   Результатом операции должно являться 10-битное число с двумя младшими разрядами, выделенными под дробную часть. Необходимо найти такое двоичное представление числа 5/7, что при умножении на него любого 8-битного беззнакового целого будет получаться корректный результат в 10-битном представлении с двумя битами дробной части. Округления выполнять путем отбрасывания битов (truncation).

   Использовать следующий шаблон для объявления интерфейса модуля:

```
---------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;

entity lab47 is
    Port ( x : in STD_LOGIC_VECTOR (7 downto 0);
           y : out STD_LOGIC_VECTOR (9 downto 0));
end lab47;
---------------------------------------
```