ЛАБОРАТОРНА РОБОТА № 1

**Вивчення інтегрованого середовища автоматизованого**

**проектування Active-HDL**

**Мета роботи:** ознайомитись з принципами автоматизованого проектування ПЛІС за допомогою Active-HDL. Вивчити структуру VHDL-проекту.Навчитися працювати з засобами управління проектом (Майстер проекту, Вікно перегляду проекту). Навчитися описувати об’єкти та їх інтерфейси за допомогою конструкції entity.

**Теоретичні відомості:**

**Програмована логічна інтегральна схема, ПЛІС** (англ. programmable logic device, PLD) — електронний компонент, що використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС не визначається при виготовленні, а задається за допомогою програмування.   
  
У відповідності до сучасних вимог системи автоматизованого проектування ПЛІС повинні забезпечувати:   
- реалізацію однієї або більше HDL-мов з можливістю введення, редагування та відлагодження вихідного тексту програм;   
- реалізацію засобів графічного введення проектної схеми, наприклад за допомогою редактора скінчених автоматів, та засобів компіляції графічного представлення в HDL-код;   
- реалізацію засобів моделювання поведінки описаного об’єкта;   
- реалізацію засобів синтезу бітового потоку з підтримкою широкого класу серій ІМС;   
- реалізацію засобів моделювання об’єкта на рівні вентилів;   
- реалізацію засобів програмування ІМС.   
  
При виборі засобів автоматизованого проектування перевагу слід надавати програмним пакетам, що реалізують процес проектування ПЛІС якомога повніше. Одним з таких пакетів є Active-HDL корпорації Aldec Inc. (США).   
  
Програмний продукт Active-HDL компанії Aldec Inc. є сучасною системою проектування і моделювання проектів з підтримкою всіх сімейств програмованих інтегральних схем ПЛІС. Дана система проектування забезпечує гнучкість підходу до розробки проекту і надає високофункціональну службу підтримки для проектування складних багаторівневих FPGA проектів. FPGA - програмована користувачем вентильна матриця.   
Середовище проектування Active - HDL включає в себе інтерфейси з усіма провідними програмними продуктами, надаючи розробникам свободу у використанні тих систем проектування, які найбільш повно відповідають вимогам для кожного конкретного проекту.   
  
Синтаксис опису портів:   
identifier1, identifier2 – імена портів;   
mode – параметр, що вказує на напрям порту і може приймати одне з трьох значень:   
in – вхідний, out – вихідний та inout – двонаправлений;   
type – тип порту;   
expression – початкове значення порту.   
  
Сигнали в VHDL - це гнучкий і потужний засіб для моделювання реальних каналів обміну інформацією як всередині електронної системи, так і між електронною системою та зовнішнім середовищем.   
  
Головними частинами Active-HDL є:   
Design Browse   
вікно перегляду проекту, призначене для для відображення інформації про складові елементи (компоненти) проекту: - файли опису пристрою, використані в проекті бібліотеки, - допоміжні файли проекту, до яких належать макроси та скрипт-файли, файли результатів симуляції роботи пристрою, допоміжні текстові файли, - структуру проекту, - сигнали та змінні, декларовані в проекті.   
HDL Editor   
редактор HDL-тексту з можливістю відображення заданих синтаксичних конструкцій мови різними кольорами; завдяки інтеграції редактора із симулятором компонент дозволяє виконувати зручне покрокове відлагодження пристрою і швидко виявляти помилки.   
Console   
вікно призначене для інтерактивного виводу текстової інформації, зокрема повідомленнь середовища; компонент також призначений для вводу команд середовища (Active-HDL commands).   
Waveform Editor   
редактор, призначений для графічного відображення та редагування результатів симуляції - часових діаграм.   
Language Assistant   
компонент є зручним засобом, який дозволяє розробнику використовувати бібліотеку шаблонів опису стандартних примітивних конструкцій та функціональних блоків; Language Assistant дозволяє розміщувати вибрані шаблони безпосередньо в редагованому файлі та створювати свої власні шаблони.   
  
Інтерфейс (від англ. Interface — поверхня розділу, перегородка) — сукупність засобів, методів і правил взаємодії (управління, контролю і т. д.) між елементами системи.

**Порядок виконання роботи:**

1.Вивчити теоретичні відомості про програмну конструкцію entity

2.Ознайомитись з складом, інтерфейсом та можливостями інтегрованого середовища Active-HDL за допомогою Help.

3. В наборі VHDL-прикладів відкрити проект Modulator та вивчити його склад і структуру за допомогою Design Browser.

4.Скласти повний перелік об’єктів, що входять до складу проектуModulator, описати їх інтерфейси.

5.Вивчити порядок застосування та функціональні можливості Майстра Нового Проекту (New Design Wisard).

6.Створити за допомогою Майстра Нового Проекту (New Design Wisard) порожній проект.

7. Описати на VHDL об’єкт, що являє собою RS-тригер.

8.Згенерувати такий же об’єкт в окремому файлі за допомогою Майстра. 9.Згенерувати за допомогою Майстра об’єкт, що реалізує 4-бітнийлічильник.

**Хід роботи:**

-------------------------------------------------------------------------------

--

-- Title : No Title

-- Design : lr1

-- Author : KSU

-- Company : NUOS

--

-------------------------------------------------------------------------------

--

-- File : c:\My\_Designs\labaaa1\lr1\compile\sm.vhd

-- Generated : 09/11/17 13:07:28

-- From : c:\My\_Designs\labaaa1\lr1\src\sm.asf

-- By : FSM2VHDL ver. 5.0.7.2

--

-------------------------------------------------------------------------------

--

-- Description :

--

-------------------------------------------------------------------------------

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.std\_logic\_arith.all;

use IEEE.std\_logic\_unsigned.all;

entity sm is

port (

R: in STD\_LOGIC;

S: in STD\_LOGIC;

INV\_U: out STD\_LOGIC;

U: out STD\_LOGIC);

end sm;

architecture sm\_arch of sm is

-- SYMBOLIC ENCODED state machine: Sreg0

type Sreg0\_type is (

S2, S1

);

-- attribute enum\_encoding of Sreg0\_type: type is ... -- enum\_encoding attribute is not supported for symbolic encoding

signal Sreg0, NextState\_Sreg0: Sreg0\_type;

-- Declarations of pre-registered internal signals

begin

----------------------------------------------------------------------

-- Machine: Sreg0

----------------------------------------------------------------------

------------------------------------

-- Next State Logic (combinatorial)

------------------------------------

Sreg0\_NextState: process (R, S, Sreg0)

begin

NextState\_Sreg0 <= S1;

-- Set default values for outputs and signals

U <= '0';

INV\_U <= '1';

case Sreg0 is

when S2 =>

U<='1';

INV\_U<='0';

if S='0' and R='1' then

NextState\_Sreg0 <= S1;

elsif R='0' then

NextState\_Sreg0 <= S2;

end if;

when S1 =>

U<='0';

INV\_U<='1';

if S='1' and R='0' then

NextState\_Sreg0 <= S2;

elsif S='0' then

NextState\_Sreg0 <= S1;

end if;

--vhdl\_cover\_off

when others =>

null;

--vhdl\_cover\_on

end case;

end process;

------------------------------------

-- Current State Logic

------------------------------------

Sreg0\_CurrentState: process (NextState\_Sreg0)

begin

Sreg0 <= NextState\_Sreg0 after 10 ns;

end process;

end sm\_arch;

**Результат:**

