Лабораторна робота №2

**Тема:** Розробка програми та моделювання декодера для рідкокристалічного індикатора.

**Мета роботи:** Вивчити методи описання поведінки об’єктів за допомогою архітектур та процесів, методи застосування оператора вибору case. Засвоїти методику моделювання поведінки об’єктів в САПР Active-HDL.

### Теоретичні відомості:

**Оператор вибору**

Оператор вибору **case** дозволяє вибрати одне з кількох можливих  
продовжень програми.

Формат запису оператора вибору в VHDL наведено нижче:  
**case** *Селектор* **is  
when** *choices\_1* **=>** послідовність\_операторів\_1;  
**when** *choices\_2* **=>** послідовність\_операторів\_2;  
**when others =>** послідовність\_операторів\_3  
**end case;**де  
*Селектор* – змінна (вираз) порядкового типу;  
*choices\_1*, *choices\_2* – локально-статично визначені вирази того ж типу, що  
й *expression*.  
При цьому *локально-статично визначеним* називається вираз,  
значення якого може бути обчисленим на етапі компіляції.  
Оператор вибору працює таким чином: якщо *choices\_1* містить  
значення змінної *Селектор,* то виконується послідовність\_операторів\_1,  
якщо *choices\_2* містить значення змінної *Селектор,* то виконується  
послідовність\_операторів\_2. У всіх інших випадках виконується  
послідовність\_операторів\_3.  
*Приклад:***variable** *k***:** *integer* **:=** 3;  
**variable** *a***:** *integer***;**…  
**case** *k* **is  
when** 1 **=>** *a* **:=** 5;  
**when** 3 **=>** *a* **:=** 6;  
**when** 2 **=>** *a* **:=** 7;  
**when others** *a* **:=** 8;  
**end case;**

В розглянутому прикладі в ролі *Селектора* виступає змінна *k*, так як  
*k* = 3, то змінна *a* отримає значення 6.  
У випадку, коли альтернативи *choices* не покривають всього  
діапазону можливих значень *Селектора,* наявність альтернативи **when  
others** є обов’язковою.  
Альтернативи *choices* можуть задаватися не тільки одиночними  
значеннями, а й діапазонами, наприклад:  
**when** ‘a’ **to** ‘с’ **=>** …  
**when** 0 **to** 15 **=>** …  
Якщо одна альтернатива складається з декількох виразів, то вони  
записуються через знак **‘|’** (об’єднуються за операцією “**OR**”), наприклад:  
**when** 1|2|5|8 **=>** …  
**when** 1 **to** 7|18 **to** 25 **=>** …  
Альтернативу можна задавати підтипами, *наприклад*:  
**subtype** *New\_Integer* **is** *integer* **range** 0 **to** 10;  
…  
**case** k **is  
when** *New\_Integer* **=>** …  
**when** *New\_Integer***|**11 **=>**…  
**end case**;

**Архітектура**

Після опису інтерфейсу об’єкта необхідно описати його архітектуру.  
*Архітектурою* називається внутрішня структура об’єкта, що визначає його  
поведінку. Саме архітектура визначає яким чином вхідні порти об’єкта  
поєднуються з вихідними. Синтаксис архітектури:  
**architecture** *identifier* **of** *entity\_name* **is***Block\_Declarative\_Item;***begin***Concurrent\_Statements***end architecture** *identifier*;  
де  
*identifier* – ім’я архітектури;  
*entity\_name –* ім’я об’єкта, поведінку якого описує дана архітектура;  
*Block\_Declarative\_Item –* список елементів програми, що будуть доступні в  
межах архітектури; в цьому розділі можна оголошувати константи, типи,  
компоненти та сигнали.  
Поведінка об’єкта описується всередині архітектури за допомогою  
конкурентних операторів (*Concurrent\_Statements*). В послідовних мовах  
програмування, таких як Pascal та С++ поняття конкурентних операторів  
відсутнє. Ці оператори призначаються для реалізації паралельних процесів  
і виконуються одночасно. До конкурентних операторів відносяться  
процеси (**process**), компоненти та деякі інші.

**Процес**

Одним з ключових елементів мови VHDL є процес (**process**). Процес  
є конкурентним оператором, тому поміщається в розділ  
*Concurrent\_Statements* архітектури.  
Синтаксис процесу має вигляд:  
*Мітка\_процесу:* **process** (*Список\_Чутливості*) **is***Розділ\_Оголошень*;  
**begin***Послідовні\_Оператори*;  
**end process** *Мітка\_процесу*;  
В наведеному прикладі позначено:  
*Мітка\_Процесу* – сформований за правилами ідентифікатор процесу;  
*Розділ\_Оголошень* – розділ, в якому можна об’являти локальні (доступні  
лише в поточному процесі) константи, змінні, підтипи, а також  
підпрограми (процедури та функції);  
*Список\_Чутливості* – список сигналів, при виникненні подій на яких  
(зміна сигналу) здійсниться запуск процесу. Список чутливості може бути  
відсутнім, але в цьому випадку обов’язкова присутність у процесі хоча б  
одного оператора **wait** (розглядається нижче);  
*Послідовні\_Оператори* – оператори, що виконуються (на відміну від  
конкурентних) послідовно один за одним. До послідовних операторів  
відносяться *умовний оператор, оператор вибору, циклічні оператори,  
оператор присвоювання, оператор пересилання значення сигналу* та інші.  
Для аналогії зазначимо, що всі оператори таких мов як Pascal та С++ є  
послідовними.  
Розглянемо роботу процесів на прикладі RS – тригера:  
**entity** *RSTrigger* **is  
port**( S*, R*: **in** *Std\_Logic*;  
*D*, *Inv\_D*: **out** *Std\_Logic;***end entity** *RSTrigger*;  
**architecture** *RSTrigger* **of** *RSTrigger* **is  
begin***Flip\_Flop:* **process** (*R,S*) **is  
begin  
if** *S* = ‘1’ **and** *R* = ‘0’ **then***D*<=’1’;  
*Inv\_D*<=’0’;  
**elseif** *S* = ‘0’ **and** *R* = ‘1’ **then***D*<=’0’;  
*Inv\_D*<=’1’;  
**else***D*<=’Z’;  
*Inv\_D*<=’Z’;  
**end if**;  
**end process** *Flip\_Flop*;  
**end architecture** *RSTrigger*;  
Робота цього процесу протікає таким чином: процес очікує появи  
події (зміни сигналу) на сигналах *R* чи *S.* Після цього перевіряються  
значення вхідних сигналів: якщо *R* =’0’, *S* = ‘1’, то на вихід тригера *D*подається сигнал логічної ‘1’, а на інверсний вихід - сигнал логічного ‘0’,  
якщо ж *R* =’1’, *S* = ‘0’, то на вихідний та на інверсний вихідний сигнали  
відповідно подаються логічні ‘0’ та ‘1’. У всіх інших випадках вихідний  
сигнал RS-тригера є невизначеним, тому на його виходи подається сигнал  
‘Z’ (високий імпеданс).

**Опис засобів Active-VHDL для моделювання об’єктів**

1. **Визначення верхнього рівня для моделювання (Top level  
   selection)** – більшість реальних проектів для синтезу цифрових пристроїв  
   складаються з більше ніж одного об’єкта **entity**. Верхній рівень вказує на  
   те, який саме з усіх наявних об’єктів **entity** буде моделюватися в наступній  
   сесії.
2. **Вибір способу відображення процесу моделювання –** хід  
   моделювання в Active-HDL можна візуалізувати різними способами – за  
   допомогою часових діаграм (**waveform editor**) або таблиць істинності  
   (**list**). Вибір способу здійснюється шляхом додавання відповідного файлу  
   до проекту.
3. **Формування тестових векторів** (**stimulators**) – процес  
   моделювання полягає в знаходженні залежностей значень сигналів  
   вихідних портів об’єкта від часу при відомих залежностях значень  
   сигналів вхідних портів від часу. Залежності сигналів вхідних портів від  
   часу у Active-HDL формуються за допомогою так званих **стимуляторів**,  
   що приєднуються до цих портів.
4. **Розрахунок залежностей значень вихідних портів об’єкта  
   від часу –** власне режим імітаційного моделювання.

### Порядок виконання роботи:

1. Вивчити розділи 4.2, 4.6, 3.2 теоретичної частини.
2. Створити новий проект в Active-HDL.
3. Створити об’єкт декодера та описати на VHDL його поведінку.
4. Скомпілювати створений об’єкт (меню Design\Compile або клавіша  
   <F11>).
5. В вікні Design Browser для встановлення верхнього рівня моделювання  
   (Top Level) вибрати з випадаючого списку об’єкт-декодер.
6. Ініціалізувати моделювання об’єкта (меню Simulation\Initialize  
   Simulation).
7. Вставити в проект новий файл Waveform Viewer (меню  
   File\New\Waveform).
8. Вставити в вікно Waveform Viewer сигнали входів та виходів декодера  
   (меню Waveform\Add Signals).
9. Почергово помічаючи вхідні сигнали декодера в вікні Waveform Viewer,  
   призначити їм стимулятори (меню Waveform\Stimulators).
10. Запустити процес моделювання об’єкта (меню Simulation\Run).
11. Вивчити отримані часові діаграми роботи декодера.
12. Змінюючи стимулятори на вхідних портах декодера перевірити  
    правильність його роботи при всіх можливих значеннях входів.

### Код проекту:

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

--

-- Title : Decoder

-- Design : Laba\_2

-- Author : Äèìà

-- Company : ÏÊ

--

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

--

-- File : Src\_2.vhd

-- Generated : Sat Oct 7 13:36:57 2017

-- From : interface description file

-- By : Itf2Vhdl ver. 1.22

--

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

--

-- Description :

--

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

--{{ Section below this comment is automatically maintained

-- and may be overwritten

--{entity {Decoder} architecture {Decoder}}

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity Decoder is

port(

INS : in STD\_LOGIC\_VECTOR(3 downto 0);

LCD\_INDICATOR : out STD\_LOGIC\_VECTOR(6 downto 0)

);

end Decoder;

--}} End of automatically maintained section

architecture Decoding of Decoder is

begin

process (INS) is

begin

case INS is

when "0000" => LCD\_INDICATOR <= "1110111";

when "0001" => LCD\_INDICATOR <= "0010010";

when "0010" => LCD\_INDICATOR <= "1011101";

when "0011" => LCD\_INDICATOR <= "1011011";

when "0100" => LCD\_INDICATOR <= "0111010";

when "0101" => LCD\_INDICATOR <= "1101011";

when "0110" => LCD\_INDICATOR <= "1101111";

when "0111" => LCD\_INDICATOR <= "1010010";

when "1000" => LCD\_INDICATOR <= "1111111";

when "1001" => LCD\_INDICATOR <= "1111011";

when others => LCD\_INDICATOR <= "0000000";

end case;

end process;

end Decoding;

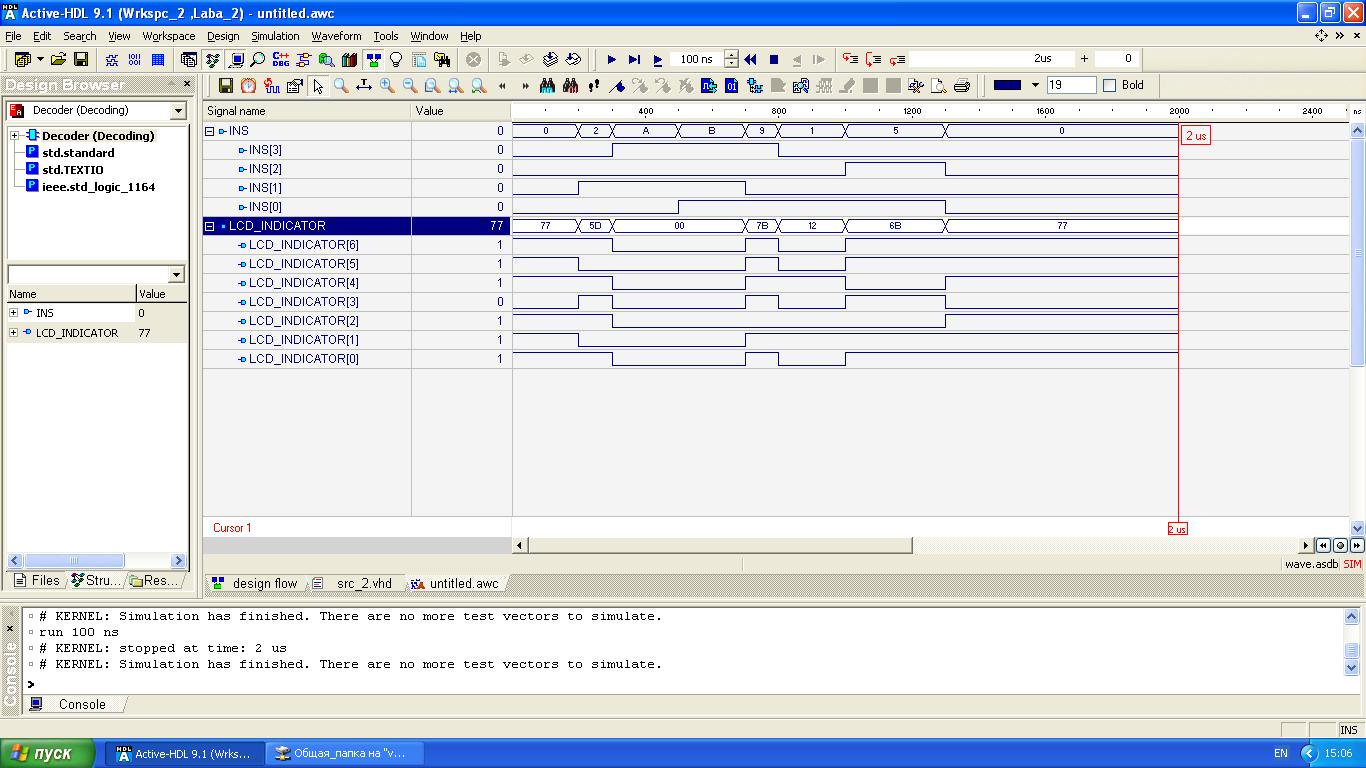


Рис. 1. Моделювання роботи об’єкта

Висновок: виконуючи цю лабораторну роботу я вивчив методи описання поведінки об’єктів за допомогою архітектур та процесів, методи застосування оператора вибору case, а також методику моделювання поведінки об’єктів в САПР Active-HDL.