**Лабораторна робота №5**

**Розробка програми і моделювання паралельного та зсувного регістрів**

**Мета:** засвоїти принципи роботи паралельного та зсувного регістрів; навчитися описувати синхронізовані процеси та використовувати циклічні оператори при описанні поведінки об’єктів за допомогою VHDL.

**Теоретичні відомості:**

Цикли. Цикли застосовуються для програмування фрагментів, що повторюються. Базовим для всіх циклічних конструкцій в VHDL є нескінчений цикл.

Нескінчений цикл. Формат запису нескінченого циклу має вигляд:

Loop\_Label: loop

послідовність\_операторів

end loop Loop\_Label;

Тут Loop\_Label – мітка циклу (може бути пропущена).

Оператор Exit. Хоч нескінчені цикли і зустрічаються на практиці, однак набагато частіше виникає необхідність в циклах, які закінчуються при виконанні якоїсь умови. Для реалізації таких циклів в VHDL використовується оператор Exit. Формат запису оператора Exit:

exit loop\_label when boolean\_expression;

Його найпростіша форма має вигляд:

exit;

За цією командою переривається виконання поточного циклу і управління передається на оператор, що є наступним після end loop. В реальних програмах часто виникає необхідність в застосуванні фрагмента:

if boolean\_expression then

exit;

end if;

який перериває виконання поточного циклу при досягненні boolean\_expression значення TRUE. Існує спрощена форма запису такого фрагмента:

exit when boolean\_expression;

Цикл з параметром (for). Крім циклу “Поки” в мові VHDL існує також спеціальна конструкція для реалізації циклу з параметром. Його синтаксис:

loop\_label: for parameter in diapason loop

послідовність\_операторів

end loop;

Суттєва відмінність циклу з параметром у VHDL від аналогічних конструкцій в мовах Pascal та С++ полягає в тому, що параметр тут не являється змінною. Параметр не потрібно попередньо об’являти, а всередині циклу параметр може розглядатися як константа (тобто його можна використовувати в виразах, але не можна присвоювати параметру значення), після закінчення циклу параметр зникає.

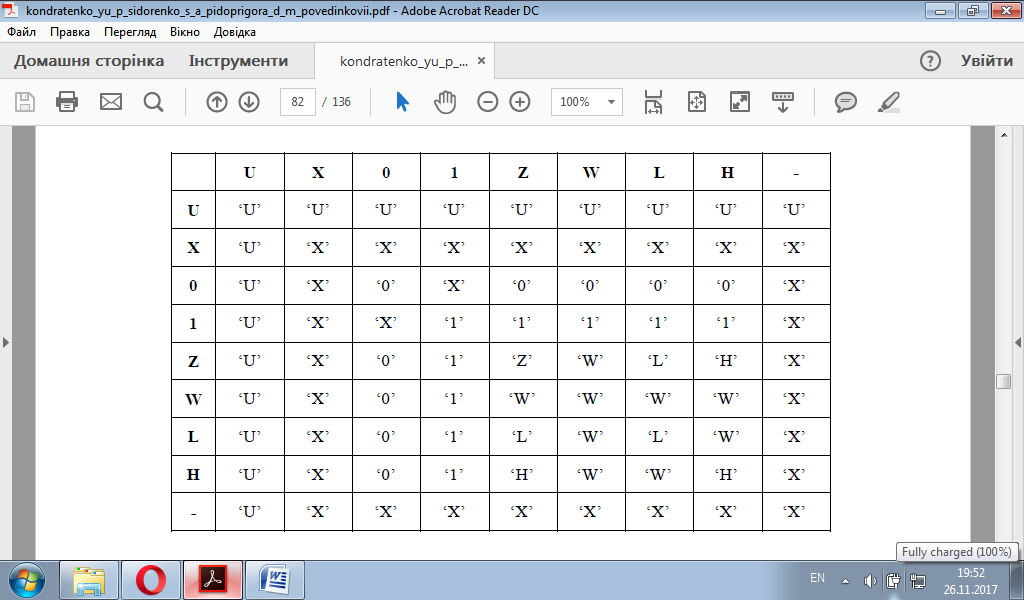
Перекриття сигналів. Мова йде про паралельну роботу процесів. Стоїть задача визначення сигналу, який має два або більше джерел. Для цього в VHDL існує так званий метод перекриття сигналів, принцип якого полягає в “змішенні” сигналів різного рівня відповідно до функції перекриття. Функція перекриття задається при описанні підтипу і визначає закон формування результуючого значення сигналу для всіх можливих комбінацій сигналів джерел.

Для моделювання реальних багаторівневих сигналів цифрових пристроїв в стандартному VHDL створено тип STD\_LOGIC. Сигнал такого типу може набувати одного з дев’яти рівнів, а саме:

‘**U**’ – невизначений; ‘**X**’ – сильний невідомий (або ‘0’, або ‘1’); ‘**0**’ – сильний нуль; ‘**1**’ – сильна одиниця; ‘**Z**’ – високий імпеданс; ‘**W**’ – слабкий невідомий; ‘**L**’ – слабкий нуль; ‘**H**’ – слабка одиниця; ‘**-**’ – байдужий стан.

Відмінності між “сильними” та “слабкими” логічними значеннями полягають у способі їх накладання. Якщо перекриваються між собою значення ’H’ і ’0’, то результатом функції перекриття буде ’0’. Відповідно, перекриття ’1’ і ’L’ дасть ’1’. У більшості випадків для моделювання реальних цифрових пристроїв та їх реалізації на основі ПЛІС достатньо значень ’0’, ’1’, ‘Z’ та ‘X’. Значення ‘X’ та 'W' з’явитися при моделюванні у тих випадках, коли значення сигналу неможливо обчислити. Типи STD\_LOGIC та STD\_LOGIC\_VECTOR (масив елементів типу STD\_LOGIC) описано у пакеті STD\_LOGIC\_1164 бібліотеки IEEE, що обумовлює необхідність звернення до нього перед початком опису об’єкта, який буде використовувати такі типи.

Таблиця 1 – Перекриття сигналів типу STD\_LOGIC



**Порядок виконання:**

1. Вивчити розділи 3.4, 4.13 теоретичної частини.

2. Розробити VHDL-модель паралельного 8-розрядного регістру.

3. Промоделювати роботу паралельного регістру, розробленого в попередньому пункті, в режимах запису інформації, збереження байта та зчитування інформації.

4. Проаналізувати на основі одержаних часових діаграм відповідність роботи паралельного регістра заданому алгоритму.

5. Розробити VHDL-модель зсувного 8-розрядного регістру.

6. Промоделювати роботу зсувного регістра, розробленого в попередньому пункті, в режимах запису інформації, збереження байта та зчитування інформації.

7. Проаналізувати на основі одержаних часових діаграм відповідність роботи зсувного регістра заданому алгоритму.

**Хід роботи:**

**Пункт 2:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity Registr is

port(

CLK : in STD\_LOGIC;

WE : in STD\_LOGIC;

RE : in STD\_LOGIC;

DATA\_IN : in STD\_LOGIC\_VECTOR(7 downto 0);

DATA\_OUT : out STD\_LOGIC\_VECTOR(7 downto 0)

);

end Registr;

architecture Registr of Registr is

begin

process (CLK) is

variable T: STD\_LOGIC\_VECTOR (7 downto 0);

begin

if (WE = '1') and (RE ='0') then T:=DATA\_IN;

elsif (WE = '0') and (RE = '1') then DATA\_OUT <= T;

else DATA\_OUT <= "ZZZZZZZZ";

end if;

end process;

end Registr;

**Пункт 5:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity registr\_posl is

port(

DATA\_IN : in STD\_LOGIC;

CLK : in STD\_LOGIC;

WE : in STD\_LOGIC;

RE : in STD\_LOGIC;

DATA\_OUT : out STD\_LOGIC\_VECTOR(7 downto 0)

);

end registr\_posl;

architecture registr\_posl of registr\_posl is

begin

process (CLK) is

variable T: STD\_LOGIC\_VECTOR (7 downto 0);

begin

if CLK = '1' then

if (WE = '1') and (RE ='0') then

for i in 7 downto 1 loop

T(i):=T(i-1);

end loop;

T(0):=DATA\_IN;

elsif (WE = '0') and (RE = '1') then DATA\_OUT <= T;

else DATA\_OUT <= "ZZZZZZZZ";

end if;

end if;

end process;

end registr\_posl;

**Результат:**

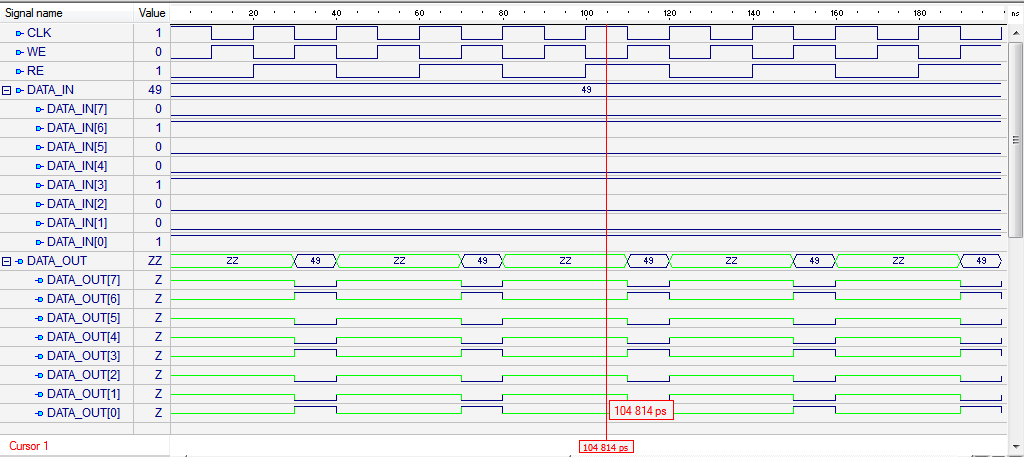


Рис. 1. Часова діаграма (пункт 3)

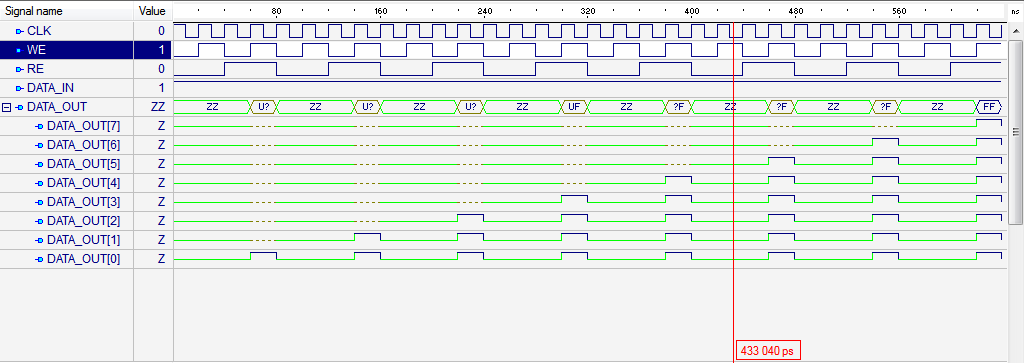


Рис. 2. Часова діаграма (пункт 6)

**Висновок.** Під час виконання лабораторної роботи я засвоїв принципи роботи паралельного та зсувного регістрів, навчився описувати синхронізовані процеси та використовувати циклічні оператори при описанні поведінки об’єктів за допомогою VHDL.