**Лабораторна робота №3**

**Моделювання режиму очікування в цифрових пристроях за**

**допомогою оператора wait**

**Мета:** отримати практичні навички застосування оператора wait при моделюванні різноманітних режимів очікування в Active-HDL. Навчитися виконувати порівняння часових діаграм.

**Теоретичні відомості:**

Аналіз основних класів електронних пристроїв показує, що вони мають безперервний цикл роботи. Тобто після ініціалізації вони виконують визначені для них операції, а потім переходять в режим очікування відповідної події нового запуску. Пристрої припиняють свою роботу після завершення поточних задач і поновлюють її знову, як тільки виникає необхідна подія. Дана властивість систем описується в VHDL за допомогою службового слова wait, яке визначає не тільки момент розриву послідовності операцій, але і необхідні умови їх поновлення. Аналогічні функції виконує список чутливості процесу, тому один і той же процес не може містити одночасно список чутливості і оператор wait. При цьому процес, в який входить вираз, що містить wait, має вигляд:

process is

begin

sequential\_statement ;

wait\_statement;

sequential\_statement;

wait\_statement;

...

end process;

Даний процес виконує послідовність дій sequential\_statement доти, доки не зустріне оператор wait (wait\_statement), який припиняє виконання процесу до виникнення події, що зазначена в параметрах оператора wait. Існує три основних і один додатковий тип оператора wait:

***Основний оператор 1.***

wait on sensitivity\_list – очікування здійснюється доти, доки не зміниться який-небудь з сигналів в списку чутливості (sensitivity\_list).

**wait on** CLK;

**wait on** Ain, Bout;

***Основний оператор 2.***

wait for time\_expression – очікування здійснюється поки не мине час, що вказаний в time\_expression.

**wait for** 5 ns;

**wait for** CLKperiod/2;

***Основний оператор 3.***

wait until boolean\_expression – очікування виконання умови, що вказана в boolean\_expression.

**wait until** CLK='1';

**wait until** IntData>15;

***Додатковий тип*** – комбінований тип запису припинення процесу, який може складатися із двох або трьох різних форм оператора wait.

**wait on** А **until** CLK='1';

**wait until** CLK='1' **for** 10 ns;

Окрім наведених вище типів запису оператора wait, він може використовуватись і без параметрів.

process is

begin

list\_of\_statements;

wait;

end process;

В останньому випадку процес почне свою роботу відразу після початку моделювання і зупиниться, коли будуть виконані всі операції з list\_of\_statements. Такий оператор wait без параметрів корисний при створенні випробувальних стендів і використовується для формування задаючих або збурюючих впливів.

**Порядок виконання:**

1. Ознайомиться з матеріалом, наведеним у розділі 4.7 теоретичної частини.

2. Відкрити базовий проект LabWait, текст якого наведено на рис. Л3.1.

3. Промоделювати роботу LabWait. Отримати часові діаграми для сигналів CLK (задається як джерело з синхронізуючим імпульсом частотою 50 МГц), Aout, Bout.

4. Зберегти результати моделювання, як файл з ім'ям Wform1, у поточному каталозі.

5. За допомогою **wait for** сформувати синхронізуючий імпульс CLK самостійно (попередньо виключивши цей сигнал з інтерфейсу системи) в тілі процесу Pr\_CLK. Рівень цього сигналу повинен змінюватись з високого на низький через кожні 10 ns.

6. Вивчити роботу процесу Pr\_А та переписати його з використанням оператора **wait on**.

7. Вивчити роботу процесу Pr\_В та переписати його з використанням оператора **wait until**.

8. Промоделювати роботу створених за допомогою оператора wait процесів, отримати часову діаграму.

9. Порівняти поточну часову діаграму та базову часову діаграму Wform1.

10. Вийти з режиму порівняння.

11. Додати до послідовності операцій процесу Pr\_А вираз wait for 30 ns.

12. Промоделювати роботу процесу, записати отриману часову діаграму як файл Wform2.awf.

13. У виразі wait for, який був доданий у п.10, змінити час затримки на 20 ns.

14. Промоделювати роботу процесу. Отриману часову діаграму порівняти з Wform2.awf.

**Хід роботи:**

**Пункт 2:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity LabWait is

port(

CLK: in STD\_LOGIC;

Bout: out STD\_LOGIC:='0';

Aout: inout STD\_LOGIC:='0'

);

end LabWait;

architecture LabWait of LabWait is

begin

Pr\_A: process (CLK) is

begin

if CLK'event and CLK='1' then Aout<='1' after 5 ns;

elsif CLK'event and CLK='0' then Aout<='0' after 5 ns;

end if;

end process Pr\_A;

Pr\_B: process (Aout) is

begin

if Aout'event then Bout<=not Aout;

end if;

end process Pr\_B;

end LabWait;

**Пункт 5-7:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity LabWait is

port(

--CLK: in STD\_LOGIC;

Bout: out STD\_LOGIC:='0';

Aout: inout STD\_LOGIC:='0'

);

end LabWait;

architecture LabWait of LabWait is

signal CLK: STD\_LOGIC;

begin

Pr\_CLK: process is

begin

CLK<='1';

CLK<='0' after 10 ns;

**wait for** 20 ns;

end process Pr\_CLK;

Pr\_A: process is

begin

if CLK='1' then Aout<='1' after 5 ns;

elsif CLK='0' then Aout<='0' after 5 ns;

end if;

**wait on** CLK;

end process Pr\_A;

Pr\_B: process is

begin

**wait until** Aout'event;

Bout <= not Aout;

end process Pr\_B;

end LabWait;

**Пункт 11,13:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity LabWait is

port(

Bout: out STD\_LOGIC:='0';

Aout: inout STD\_LOGIC:='0'

);

end LabWait;

architecture LabWait of LabWait is

signal CLK: STD\_LOGIC;

begin

Pr\_CLK: process is

begin

CLK<='1';

CLK<='0' after 10 ns;

**wait for** 20 ns;

end process Pr\_CLK;

Pr\_A: process is

begin

if CLK='1' then Aout<='1' after 5 ns;

elsif CLK='0' then Aout<='0' after 5 ns;

end if;

wait for 30 ns;

--wait for 20 ns;

**wait on** CLK;

end process Pr\_A;

Pr\_B: process is

begin

**wait until** Aout'event;

Bout<=not Aout;

end process Pr\_B;

end LabWait;

**Результат:**

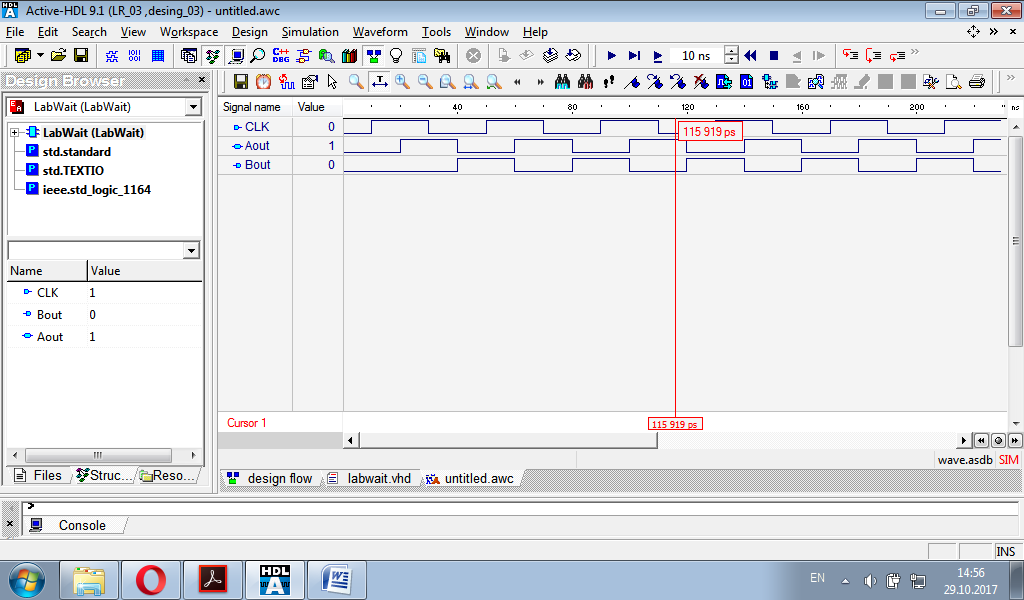


Рис. 1. Часова діаграма (пункт 3)

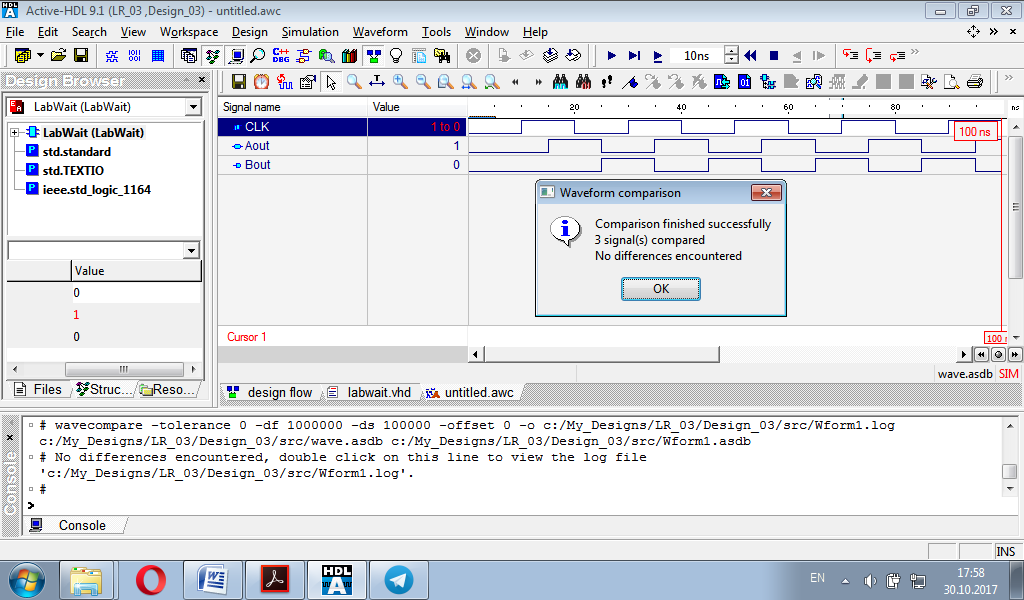


Рис. 2. Часова діаграма (пункт 9)

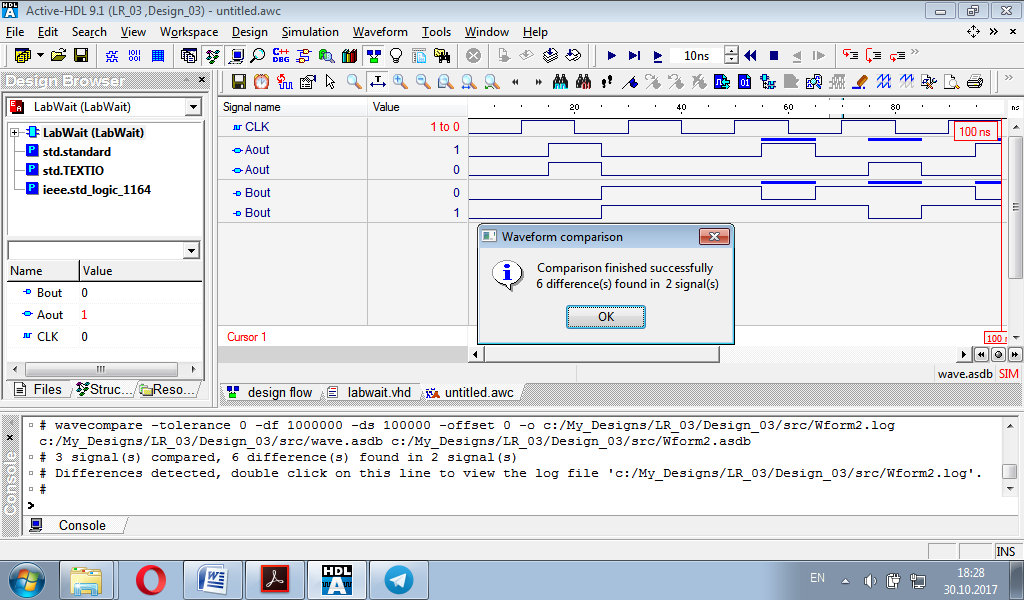


Рис. 3. Часова діаграма (пункт 14)

**Висновок.** Під час виконання лабораторної роботи я отримав практичні навички застосування оператора wait при моделюванні різноманітних режимів очікування в Active-HDL. Також навчився виконувати порівняння часових діаграм.