Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
“Автоматизація та інформатизація інженерного проектування електричних та електронних апаратів”

для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка всіх форм навчання

Частина 1

2018

Методичні вказівки до виконання лабораторних робіт з дисципліни “Автоматизація та інформатизація інженерного проектування електричних та електронних апаратів” для студентів спеціальності 141 – Електроенергетика, електротехніка та електромеханіка всіх форм навчання. Частина 1 / Укл. Л.С. Скрупська. - Запоріжжя: ЗНТУ, 2018. – 20 с.

Укладачі: Л.С. Скрупська, ст. викл.

Рецензент: О.В.Близняков, доцент

Відповідальний за випуск: Л.С. Скрупська, ст. викл.

Затверджено
на засіданні кафедри ЕЕА
Протокол № 5
від 25 січня 2018 р.

Затверджено НМК ЕТФ
Протокол № 6
від 25 січня 2018 р.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА № 1
ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ САПР ACTIVE-HDL

Мета роботи: Вивчити структуру VHDL–проекту і функціональні можливості системи автоматизованого проектування Aldec Active–HDL.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ACTIVE-HDL

САПР Aldec Active–HDL є інтерактивним інтегрованим середовищем розробки і функціонально–структурного моделювання проектів цифрових систем (ЦС) на НВІС програмованої логіки (НВІС ПЛ). Її основними компонентами є компілятори мов опису апаратури і засоби функціонального моделювання проектів. Вбудовані додаткові засоби – текстовий редактор з функцією перевірки синтаксису, редактор кінцевих автоматів, бібліотека конструкцій мови, розвинута система допомоги та ін. – дозволяють значно скоротити час проектування і моделювання. Підтримуються мови опису апаратури *VHDL, Verilog* та *EDIF*. Можлива спільна робота Active–HDL і програмних засобів фірм–виробників НВІС ПЛ. Це дозволяє вести проектування цифрової системи від початку до кінця: від тексту програми до готових виробів в базисі FPGA і CPLD.

Користувачу доступні групи засобів розробки VHDL–проекту (рис. 1.1):

- засоби введення проекту (*Design Entry Tools*);
- засоби керування проектом (*Control Tools*);
- засоби виведення результатів моделювання (*Simulation Output Viewers*);
- засоби відлагодження проекту (*Debugging Tools*);
- ядро системи та інструментальні засоби.

До **засобів введення проекту** належать функціональні одиниці САПР Active-HDL: текстовий редактор з перевіркою, графічний редактор логічних схем, графічний редактор кінцевих автоматів, бібліотека конструкцій мови, конвертер HDL–тексту в логічні схеми.

Засоби керування проектом: менеджер проектів, менеджер файлів, ресурсів і структури проектів, менеджер керування напрямом проектування, менеджер бібліотек, вікно команд і повідомлень.

Функціональні одиниці, що є **засобами виведення результатів моделювання:** редактор часових діаграм, вікно таблиці перемикань.

До **засобів відлагодження проекту** належать: ядро системи моделювання, вікно паралельних процесів, вікно сигналів користувача, стек викликів підпрограм, вікно графічного відображення обміну інформацією усередині проекту.

Ядро системи та **інструментальні засоби** Active-HDL складають: генератор VHDL–проектів, генератор Verilog–проектів, генератор документів у форматі EDIF, компілятор мови VHDL, компілятор мови Verilog, компілятор EDIF.

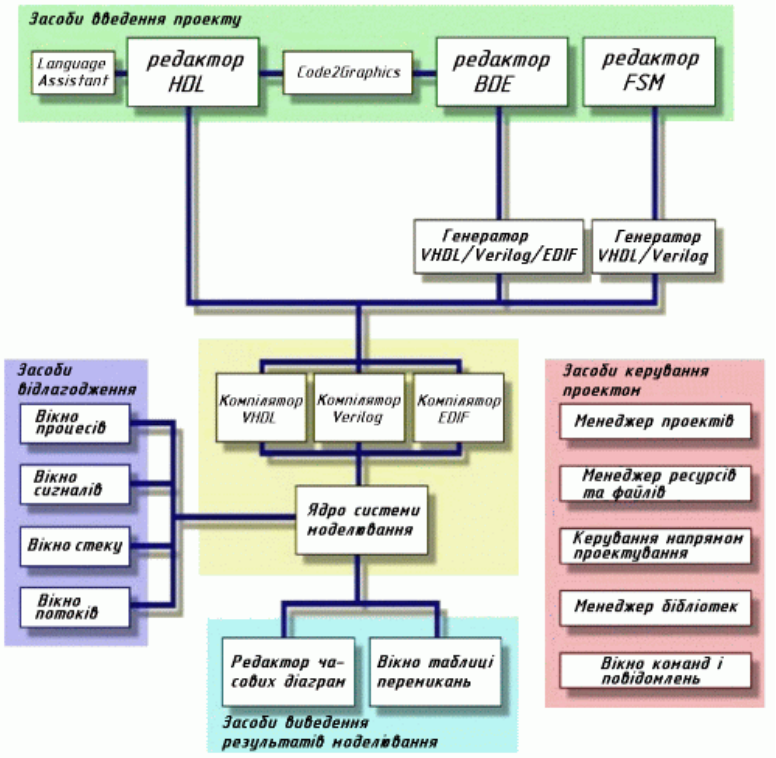

Рисунок 1.1 – Організація САПР Aldec Active–HDL

Часто буває достатньо скласти опис поведінки цифрової системи, алгоритму її роботи. З погляду автоматизації проектування

ЦС, якщо не визначений базис її реалізації, такий підхід є більш раціональним.

Будь–який об'єкт може мати декілька варіантів алгоритмічного опису архітектури. Різні за формою (синтаксисом), але єдині за значенням (семантикою) представлення одного і того ж алгоритму можуть задавати однакову поведінку об'єкту, але мати різну структурну реалізацію.

Можливість за допомогою **мови опису апаратури** (HDL) різними способами задавати один і той же алгоритм дозволяє за допомогою САПР Active–HDL автоматично синтезувати відмінні за складністю (рівнем деталізації) функціонально–структурні описи ЦС. Перевага, яку дає використання САПР Active–HDL, полягає в тому, що користувач, за допомогою конструкцій мови опису апаратури, сам указує рівень деталізації моделі ЦС.

Відповідно до цього складається **VHDL–проект** – формальний опис поведінки ЦС на мові VHDL. Користувач сам вибирає рівень деталізації опису цифрової системи (абстракції) з чотирьох можливих:

- **поведінковий рівень** (опис алгоритму роботи ЦС);
- **рівень регістрових передач** (рівень RTL);
- **функціонально–логічний рівень** (враховуються часові затримки розповсюдження сигналів);
- **рівень логічних елементів** (враховується час спрацьовування логічних елементів).

VHDL–проект складається з **логічних файлів** опису компонентів ЦС, з'єднаних згідно формально визначеного закону функціонування ЦС. Логічні файли організовуються відповідно до ієрархії опису проектованої цифрової системи. Один з логічних файлів VHDL–проекту є файлом верхнього рівня ієрархії описів (*Top–level Design File*). Він грає роль організатора, що визначає архітектуру, склад ЦС і алгоритм її поведінки. Логічні файли нижніх рівнів ієрархії (*Low–level Design files*) містять детальний опис блоків і модулів ЦС. Ім'я проекту завжди співпадає з ім'ям логічного файлу верхнього рівня ієрархії проекту. До складу VHDL–проекту можуть бути залучені різноманітні допоміжні файли (*Ancillary Files*).

В САПР Active–HDL кожний проект зберігається в окремому каталозі, що зветься **каталогом проекту**. Проект і каталог, де він зберігається, мають однакові імена. Кожний каталог проекту містить

файл опису проекту (.ADF) і підкаталоги *SRC*, *GENERIC* і ін. Підкаталог SRC містить текстові файли мовою VHDL (.VHD), файли діаграм автоматів (.ASF) тощо. Підкаталог *GENERIC* містить робочі файли, які використовуються при компіляції і моделюванні.

Бібліотеки Active–HDL діляться на дві групи:

– **системні бібліотеки** поставляються разом з САПР Active–HDL і знаходяться в каталозі *VLIB***.**

– **бібліотеки проектів** автоматично створюються окремо для кожного проекту і зберігаються в підкаталозі *GENERIC* проекту.

Кожна бібліотека складається з двох файлів: бібліотечного індексного файлу (розширення .INI) і головного бібліотечного файлу (розширення .MFG), що містить дані, що використовуються середовищем Active-HDL. В підкаталозі VLIB присутній також бібліотечний файл конфігурації *LIBRARY.CFG*, загальний для всіх бібліотек. Системні бібліотеки з каталогу VLIB наперед скомпільовані і готові до вживання у VHDL-програмах у вигляді *пакетів* (табл. 1.1).

Таблиця 1.1 – Вбудовані стандартні пакети САПР Active–HDL

Пакет	**Призначення**
STD.STANDARD	Стандартний пакет VHDL загального призначення (стандарт IEEE Std. 1076)
STD.TEXTIO	Стандартний пакет VHDL текстових операцій (стандарт IEEE Std. 1076)
IEEE.STD_LOGIC_1164	Підпрограми багаторівневої логіки (стандарт IEEE Std. 1164)
IEEE.VITAL_TIMING	Примітиви VITAL'95 (стандарт IEEE Std. 1076.4)
IEEE.VITAL_PRIMITIVES	Примітиви VITAL'95 (стандарт IEEE Std. 1076.4)
IEEE.STD_LOGIC_ARITH	Математичні підпрограми (стандарт IEEE Std. 1076.2)
IEEE.STD_LOGIC_SIGNED	Підпрограми IEEE (стандарт IEEE Std. 1076.5)
IEEE.STD_LOGIC_UNSIGNED	Підпрограми IEEE (стандарт IEEE Std. 1076.5)

Під час установки середовища Active–HDL пропонується підключити бібліотеки, що призначені для підтримки розробки проектів ЦС в базисі НВІС ПЛ ведучих виробників (Actel, Altera, Atmel, Cypress, Lattice, Lucent, Quicklogic, Synopsys і Xilinx).

Разом з середовищем поставляється докладна контекстна довідка та електронні підручники. Великий обсяг довідкових матеріалів зосереджено в навчальній системі Enhanced VHDL Tutorial (EVITA). Її можна викликати з САПР Active–HDL за допомогою пункту меню *Help → Interactive VHDL Tutorial*.

Розглянемо докладніше **процес розробки VHDL–проекту** нескладної ЦС в САПР Active–HDL. Наприклад, треба виконати функціональне моделювання поведінки демультиплексора DM(3), таблиця дійсності якого наведена в табл. 1.2. Нагадаємо, що демультиплексор – це цифровий пристрій, що перемикає вхідний сигнал W на один з виходів Y. На який саме – залежить від керуючих сигналів X.

Таблиця 1.2 – Таблиця дійсності демультиплексору DM(3)

Входи				**Виходи**							
w	**x(2)**	**x(1)**	**x(0)**	**y(7)**	**y(6)**	**y(5)**	**y(4)**	**y(3)**	**y(2)**	**y(1)**	**y(0)**
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

2 СТВОРЕННЯ НОВОГО ПРОЕКТУ

2.1 Клацнувши на піктограмі на Робочому Столі Windows, запустіть середовище Active–HDL.

2.2 На панелі інструментів Active–HDL клацніть піктограму *New Workspace* (створити нове робоче середовище).

2.3 З‘явиться вікно **New Workspace**. В полі "*Type the workspace name*" введіть ім‘я робочого середовища. Залиште активною опцію

"*Add New Design to Workspace*" (додати новий проект до робочого середовища). Клацніть **OK**.

2.4 Запускається майстер створення проекту **New Design Wizard**.

2.5 Настройка параметрів проекту виконується за чотири кроки. Виберіть режим *Create an empty design* (створити порожній проект). Клацніть кнопку **Next.**

2.6 Друге вікно (другий крок) майстра призначено для вибору інструменту синтезу і імплементації. Необхідно також вказати тип НВІС ПЛ, конфігурацію редактора BDE і мову опису проекту. В полі *Default HDL Language* виберіть **VHDL.**

2.7 Третій крок – визначення імені проекту, місця на диску для його зберігання і, якщо необхідно, настройка імені бібліотеки проекту. За умовчанням ім'я проекту є ім'ям його бібліотеки. В полі *Type the design name* наберіть *Lab5* і клацніть кнопку **Next.**

2.8 Знов створеному проекту привласнено ім'я **Lab5**. В кінці роботи майстер створення проекту виводить інформацію про результати роботи. Файл проекту має ім'я *Lab5.adf.* Клацніть кнопку **Done**. На диску з'явиться каталог VHDL–проекту *Lab5***.** Структура цього каталогу відображається у вікні **Design Browser**. Коли в проект Lab5 будуть вноситись зміни щодо змісту файлів і директорій каталогу Lab5. Файл проекту *Lab5.adf* містить загальний опис проекту, компілятора і бібліотеки *Lab5.LIB*. У файл *console.log* будуть заноситися повідомлення про поточний стан проекту. Отримати доступ до повідомлень консолі можна у будь–який час натисненням на кнопку *Console*  на панелі інструментів або за допомогою комбінації клавіш <ALT+0>.

3 СТВОРЕННЯ НОВОГО ДОКУМЕНТА (ВВЕДЕННЯ ТЕКСТУ ПРОГРАМИ)

3.1 Натисніть кнопку *Design Flow*  панелі інструментів (комбінація клавіш <Alt+3>). Відкриється вікно управління проектом (рис. 5.2).

3.2 Нам потрібен текстовий редактор. Для цього треба клацнути на піктограмі *HDE*. З'явиться вікно вибору мови опису апаратури. Виберіть **VHDL.**

3.3 З'явиться вікно **New Source File Wizard**. Цей майстер автоматично генерує частину тексту програми на мові VHDL – опис

портів в модулі *entity*. На першому кроці пропонується включити створюваний файл в поточний проект (*Add generated file to design*). Клацніть кнопку **Next**.

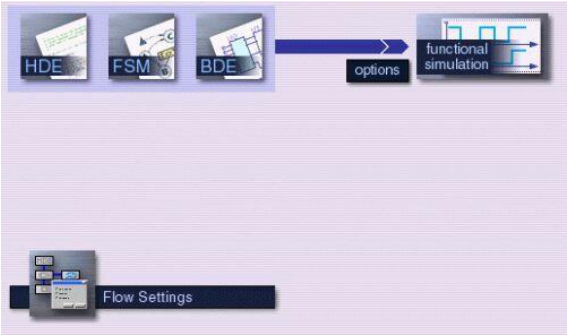

Рисунок 3.1 – Вікно Design Flow. Управління проектом

3.4 На другому кроці майстер пропонує дати імена VHDL–файлу, сутності і опису архітектури (*architecture*) проекту, що генерується. Текстовий файл тут названий *Lab5*, entity – *demux_3*, а опис архітектури має ім'я *beh* ("behavior" – поведінка). Для продовження клацніть **Next**.

3.5 Третій крок найвідповідальніший. Тут задаються порти, їх типи і режими роботи. Наш демультіплексор повинен мати один інформаційний вхід **w**, три входи керування **X** і вісім виходів **У** (рис. 3.2).

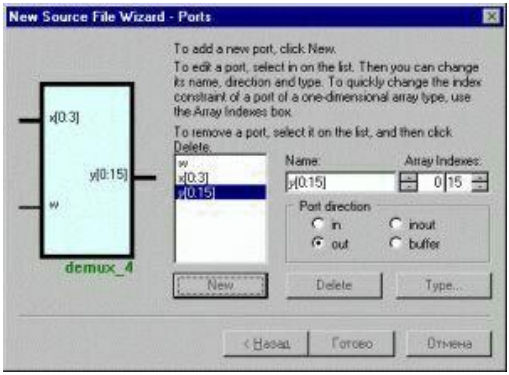

Рисунок 3.2 – New Source File Wizard. Задавання портів

3.6 Список портів редагується за допомогою кнопок **New** і **Delete**. Для створення нового порту натисніть кнопку **New**. В полі *Name* введіть його ім'я. Для вибору напрямку порту використовується група параметрів *Port direction*. Для вибору типу порту треба клацнути кнопку **Type** і відкрити вікно **Port Type**. За умовчанням задаються типи *STD_LOGIS* або *STD_LOGIS_VECTOR***.** В мові VHDL є чотири режиму роботи портів: **in** – вхідний порт, **out** – вихідний порт, **inout** – двонаправлений порт, **buffer** – порт–буфер. Параметр *Array Indexes* використовується для введення діапазонів масивів. Опис портів демультиплексору DM(3) має такий вигляд:

x: **in** STD_LOGIS_VECTOR (0 **to** 2);
w: **in** STD_LOGIS;
y: **out** STD_LOGIS_VECTOR (0 **to** 7);

3.7 На цьому робота з майстром шаблонів VHDL–програм **New Source File Wizard** закінчується. Клацніть **Done**. Буде створений шаблон VHDL–програми – моделі демультиплексора, в каталозі проекту з'явиться новий файл *Lab5.vhd* і відкриється вікно текстового редактора HDE.

Листинг 3.1 – Текст шаблона програми на мові VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity demux_3 is
    port(
            x : in BIT_VECTOR(0 to 3);
            w : in BIT;
            y : out BIT_VECTOR(0 to 15)
        );
end demux_3;
architecture beh of demux_3 is
begin
    -- enter your statements here --
end beh;
```

Шаблон VHDL–програми складається з трьох основних частин:

– перелік стандартних бібліотек, що використовуються (модуль library);

– повністю оформлений опис об'єкту (модуль entity);

– порожні операторні дужки опису архітектури (модуль architecture).

Модуль **library** визначає імена бібліотек проекту. VHDL–бібліотека може містити пакети, об'єкти, описи архітектури і конфігурації. В даному випадку використовується пакет *std_logic_1164* стандартної бібліотеки *IEEE*.

У модулі **entity** оголошується ім'я сутності (*demux_3*), описується інтерфейс між сутністю та середовищем, в якому вона функціонує. Тут визначаються типи і напрями сигналів.

Модуль **architecture** визначає внутрішню організацію, принцип функціонування сутності на мові опису апаратури VHDL.

3.8 В модулі architecture замість надпису — *enter your statements here* — необхідно додати код, який описує поведінку демультиплексора. В листингах 3.2 та 3.3 наведено два еквівалентні варіанти опису ЦС. Введіть програму, що навдено в лістингу 3.2.

Листинг 3.2 – Перший варіант програми

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль
library
library IEEE;
use IEEE.STD_LOGIC_1164.all;
-- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль entity
entity demux_3 is
    port(
        x : in BIT_VECTOR(0 to 3);
        w : in BIT;
        y : out BIT_VECTOR(0 to 15)
        );
end demux_3;
-- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль
architecture
architecture beh of demux_3 is
begin
    GO: process (x,w)
    begin
    L1: if w = '0'
            then  y <= "00000000";
        else
            L2: case x is
                when "000" =>          y <= "10000000";
```

```
                when "100" =>          y <= "01000000";
                when "010" =>          y <= "00100000";
                when "110" =>          y <= "00010000";
                when "001" =>          y <= "00001000";
                when "101" =>          y <= "00000100";
                when "011" =>          y <= "00000010";
                when "111" =>          y <= "00000001";
            end case L2;
        end if L1;
    end process GO;
end beh;
```

Листинг 3.3 – Другий варіант програми

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль
library
library IEEE;
use IEEE.STD_LOGIC_1164.all;
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль entity
entity demux_3 is
    port(
        x : in STD_LOGIC_VECTOR(0 to 2);
        w : in STD_LOGIC;
        y : out STD_LOGIC_VECTOR(0 to 7)
        );
end demux_3;
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- модуль
architecture
architecture beh of demux_3 is
begin
y(7)<= w and x(2) and x(1) and x(0) after 20 ns;
y(6)<= w and x(2) and x(1) and not x(0) after 12 ns;
y(5)<= w and x(2) and not x(1) and x(0) after 24 ns;
y(4)<= w and x(2) and not x(1) and not x(0) after 18 ns;
y(3)<= w and not x(2) and x(1) and x(0) after 15 ns;
y(2)<= w and not x(2) and x(1) and not x(0) after 16 ns;
y(1)<= w and not x(2) and not x(1) and x(0) after 21 ns;
y(0)<= w and not x(2) and not x(1) and not x(0) after 14
ns;
end beh;
```

4 КОМПІЛЯЦІЯ ПРОЕКТУ

Виконаємо компіляцію нашого проекту. Для цього можна натискувати клавішу <F11> або кнопку *Compile* панелі інструментів. На рис. 5.4 показані кнопки запуску компіляції і результат компіляції VHDL–проекту. Якщо при складанні програми були допущені помилки, повідомлення про них можна побачити у вікні **Console.**

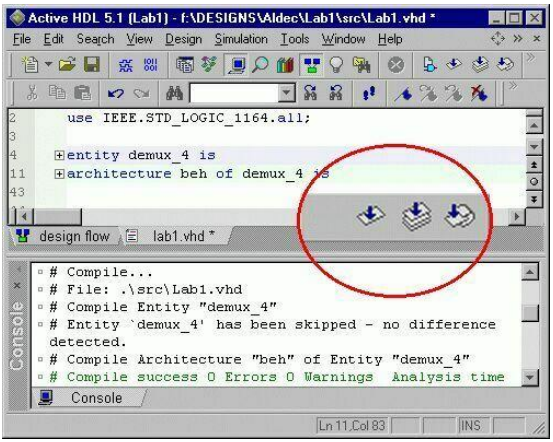

Рисунок 4.1 – Компіляція VHDL–проекту

5 МОДЕЛЮВАННЯ

Для того, щоб перевірити, чи правильно працює синтезований засобами САПР Active–HDL опис поведінки демультиплексора DM(3), виконаємо функціональне моделювання.

Перш ніж приступити до верифікації VHDL–проекту, необхідно скласти **тестову послідовність** вхідних сигналів w і x[0..3]. В найпростішому випадку тестова послідовність – це **вектор**, що містить всі можливі комбінації вхідних сигналів. Для демультиплексора DM(3) це 2^4=16 векторів [w, x0, x1, x2, x3].

5.1 Виберіть команду *File* → *New* → *waveform* або піктограму *New waveform*. З'явиться вікно редактора часових діаграм.

5.2 Додамо в редактор часових діаграм сигнали, що визначені в моделі demux_3. Клацніть правою кнопкою в лівій частині вікна

редактора. В контекстному меню виберіть команду *Add Signals*. З'явиться вікно **Add Signals** (рис. 5.1).

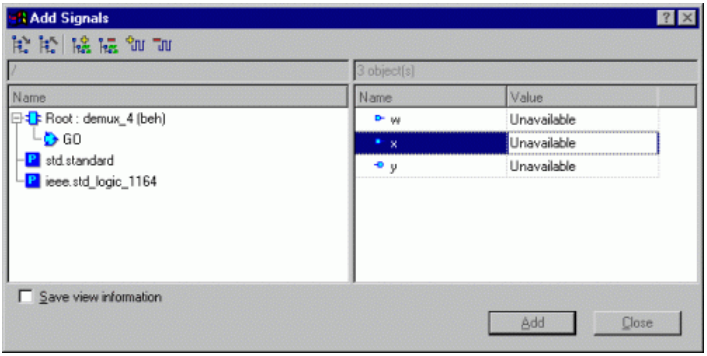

Рисунок 5.1 – Вікно Add Signals

5.3 Вибираючи по черзі сигнали **w**, **x**, **y** і клацаючи на кнопці **Add**, додайте ці сигнали у вікно редактора часових діаграм.

5.4 В головному меню Active–HDL виберіть команду *Simulation* → *Initialize Simulation*.

5.5 Тепер можна задати **стимулятори** моделювання. Щоб створити стимулятор треба виділити потрібний сигнал в редакторі часових діаграм, клацнути правою кнопкою і в контекстному меню вибрати команду **Stimulators**. Є декілька типів стимуляторів (таблиця 5.3).

5.6 Задамо стимулятор для сигналу **w**. Виберіть тип стимулятора **Hotkey**. Встановіть курсор в полі *Press new hotkey* і натисніть клавішу <w> клавіатури. Її будемо використовувати для перемикання рівня сигналу **w**.

5.7 Задамо стимулятор для вектору сигналів **x[0..2]**. Для перебору всіх комбінацій вектора сигналів зручно використовувати стимулятор типу **Counter**. Виділіть вектор **x** і встановіть тип стимулятора **Counter** (рис. 5.6).

5.8 Встановіть настройки лічильника відповідно до таблиці 5.1. Для того, щоб рахунок проводився в зворотному порядку, треба встановити прапорець *Reverse bits order*.

Таблиця 5.1 – Типи стимуляторів

	**Значення** (Value). Дозволяє вибрати значення, що буде надане виділеному сигналу чи групі сигналів, об'єднаних у шину.
	**Формула** (Formula). Дозволяє задати часову діаграму за допомогою формули. Наприклад формула "0 0, 1 12" означає, що сигнал має значення '0' у момент часу 0 і змінюється на '1' після 12 пікосекунд.
	**Гаряча клавіша** (Hotkey). Дозволяє призначити сигналові гарячу клавішу, натискання якої призводитиме до зміни значення сигналу. За замовчуванням сигнал переключатиметься між 0 і 1, проте у цей список можна додати і інші допустимі для даного типу сигналу значення. При натисканні гарячої клавіші сигнал набуватиме наступного значення, включеного у список.
	**Годинник** (Clock). Дозволяє задати періодичний сигнал з визначеними параметрами (частотою, періодом тощо), користуючись зручним графічним редактором.
	**Лічильник** (Counter). Дозволяє моделювати сигнали на виходах лічильника як послідовність чисел в заданому напрямку лічби і в заданому базисі (двійковий код, код Грея та ін.).
	**Наперед визначений** (Predefined). Це стимулятори синхросигналів або формульні, які мають власні імена, що дозволяє назначати їх декільком сигналам, не повторюючи кожен раз вводу параметрів.
	**Стимулятор, що настроюється користувачем** (Custom). Дозволяє вибрати наперед визначений стимулюючий сигнал з заданого переліку та поповнити цей перелік власними сигналами, що часто використовуються.

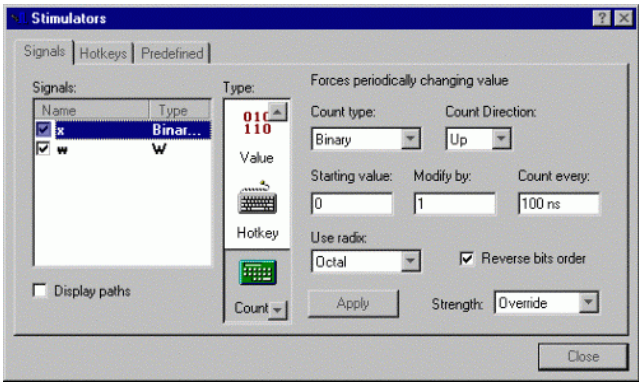

Рисунок 5.2 – Вікно Stimulators. Тип стимулятора – Counter

Таблиця 5.2 – Настройки стимулятора для вектора x[0..2]

Налаштування		**Значення**
Count type	Тип лічильника	Binary
Count direction	Напрям рахунку	Up
Starting value	Початкове значення	0
Modify	Величина зміни	1
Use radix	Підстава	Hexadecimal
Count every	Період рахунку	100 ns

5.9 Перевіримо, як веде себе модель *demux_3*, якщо сигнал **w** має високий логічний рівень. Натисніть клавішу <w> ("гарячу" клавішу, що зв‘язана з сигналом **w**). Натискуючи клавішу <F5> вісім разів, виконайте покрокове моделювання роботи демультиплексора для першої половини тестової послідовності (коли сигнал **w**='1').

5.10 Знову натисніть клавішу <w> та виконайте покрокове моделювання демультиплексора для другої половини тестової послідовності (коли сигнал **w**='0').

5.11 Вибравши команду головного меню *Simulation* → *End Simulation*, зупиніть процес моделювання. На рис. 5.7 показаний результат функціонального моделювання правильно працюючого демультиплексора DM(3).

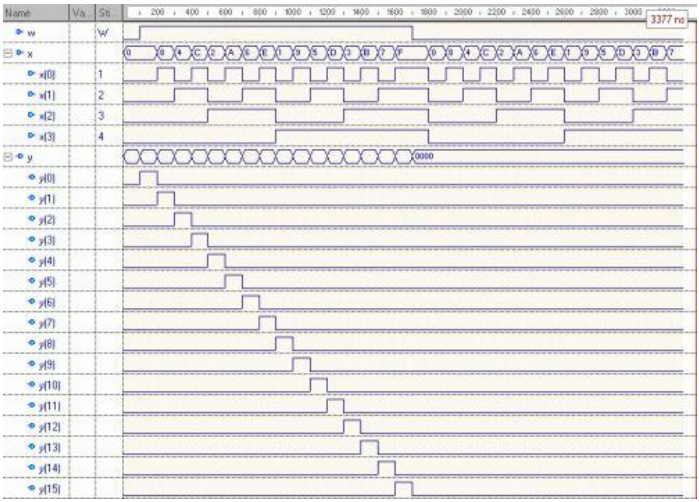

Рисунок 5.3 – Функціональне моделювання демультиплексора DM(3)

6 ПОРЯДОК ВИКОНАННЯ РОБОТИ

6.1 Проект Lab1. Створіть і виконайте моделювання моделі *demux_3*. Для опису поведінки демультиплексора DM(3) використовуйте текст програми, наведений в лістингу 3.2. Текст програми та результати моделювання використовуйте для складання звіту.

6.2 Проект Lab2. Створіть новий проект і виконайте моделювання моделі *demux_3*, для опису якої використовується текст програми, наведений в лістингу 3.3. Текст програми та результати моделювання використовуйте для складання звіту.

6.3 Порівняйте результати моделювання і зробіть висновки.

6.4 Під час моделювання можна помітити, що сигнали перемикаються неодночасно. Цей ефект називається **гонкою сигналів**. Редактор часових діаграм має необхідні інструменти для докладного вивчення часових діаграм. Наприклад, кнопкою *Zoom in* можна збільшити масштаб відображення, а натискання кнопки *Measurement Mode*, вмикає режим вимірювання часових інтервалів. На рис. 5.4 показаний приклад вимірювання часу встановлення сигналу.

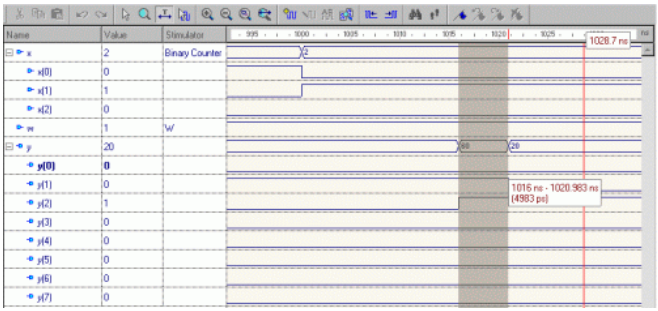

Рисунок 5.4 – Вимірювання часових інтервалів

6.5 Визначте значення максимальної затримки переключення сигналів вектору **y[0..7]**.

7 ЗМІСТ ЗВІТУ

7.1 Мета роботи.

7.2 Таблиця істиності та логічна схема демультиплексора.

7.3 Тексти програм.

7.4 Результати моделювання.

7.5 Результати вимірювання часових інтервалів.

7.6 Висновки.

8 КОНТРОЛЬНІ ПИТАННЯ

8.1 Організація VHDL-проекту цифрової системи.

8.2 Мови опису апаратури. Який зв'язок між ними і мовами програмування високого рівня загального користування.

8.3 Системи автоматизації проектування радіоелектронних пристроїв.

8.4 Організація САПР Active–HDL. Основні компоненти Active–HDL і їх призначення.

8.5 З яких частин складається опис об'єкту проектування?

8.6 Що таке тестова послідовність? Як задаються тестові набори вхідних сигналів? Наведіть приклади.

8.7 Типи стимуляторів. Як вони задаються?

ЛІТЕРАТУРА

1 Семенец В.В. Проектирование цифровых систем с использованием языка VHDL: Уч. Пособие / В.В. Семенец, И.В. Хаханова, В.И. Хаханов.– Харьков: ХНУРЭ, 2003.– 492 с.

2 Бибило П.Н. Синтез логических схем с использованием языка VHDL.– М.: СОЛОН–Р, 2002.– 384 с.

3 Кондратенко Ю.П., Сидоренко С.А., Підопригора Д.М. Поведінковий синтез цифрових пристроїв у середовищі Active–HDL: Навчальний посібник. – Миколаїв: МФ НаУКМА, 2002.– 116 с.

4 Шапо Ф.С. Введение в VHDL – язык проектирования цифровых систем / Ф.С. Шапо, В.Ф. Шапо. – Одесса: Астропринт, 2001.– 220 с.