

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни
“Технології проектування комп’ютерних систем”
для студентів спеціальностей 8.05010201 "Комп'ютерні системи і
мережі", 8.05010203 "Спеціалізовані комп'ютерні
системи" всіх форм навчання

Частина 3

2015

Методичні вказівки до лабораторних робіт з дисципліни “Технології проектування комп’ютерних систем” для студентів спеціальностей 8.05010201 “Комп’ютерні системи і мережі”, 8.05010203 “Спеціалізовані комп’ютерні системи” всіх форм навчання. Частина 3. /Укл. А.В. Сирота - Запоріжжя: ЗНТУ, 2015. 44 с.

Рекомендовано НМК як методичні вказівки до лабораторних робіт з дисципліни „Технології проектування комп’ютерних систем” всіх форм навчання.

Укладач: А.В. Сирота, ст. викладач.

Рецензент: Р.К. Кудерметов, доцент, к.т.н.

Відповідальний за випуск: А.В. Сирота, ст. викладач.

Затверджено
на засіданні кафедри КСМ

Протокол № 6
від 02. 02.2015

ЗМІСТ

1 ПРОСТА МІКРОПРОЦЕСОРНА СИСТЕМА	4
1.1 Постановка задачі	4
1.2 Структура мікропроцесорної системи.....	4
1.3 Інтерфейс мікропроцесора і зовнішньої пам'яті.....	5
2 ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ.....	7
2.1 Створення проекту.....	7
2.1.1 Навчальний VHDL-проект містить п'ять текстових документів мовою VHDL, одну логічну схему та один командний файл запуску теста:	7
2.2 Модель зовнішньої пам'яті.....	10
2.3 Модель генератора синхроімпульсів.....	16
2.4 Проектування імітатора мікропроцесора	18
2.5 Файл верхнього рівня проекту	21
2.6 Майстер Code2Graphics	27
2.7 Функціональне моделювання	30
2.8 Створення випробувального стенда	35
2.9 Послідовність виконання роботи	41
2.10 Зміст звіту	42
2.11 Контрольні питання	42
ЛІТЕРАТУРА	44

Лабораторна робота № 5

1 ПРОСТА МІКРОПРОЦЕСОРНА СИСТЕМА

Мета роботи. На прикладі простої мікропроцесорної системи навчитись розробляти проекти цифрових систем, створені мовою VHDL. Отримати навички виконання функціонального моделювання та аналізу стану об'єкта, що проектується.

1.1 Постановка задачі

У лабораторній роботі вивчаються наступні питання.

1. Вихідна стратегія створення моделі засобами САПР Active-HDL.
2. Методика створення програм мовою VHDL.
3. Методика створення логічних схем.
4. Методика створення випробувального стенду.
5. Методика виконання аналізу стану об'єкта проектування.

Виконується проектування та симуляція поведінки блока зовнішньої пам'яті на прикладі етапу розробки компонентів мікропроцесорної системи, основу якої становить гіпотетичний мікропроцесор.

1.2 Структура мікропроцесорної системи

Невід'ємною частиною будь-якої мікропроцесорної системи є *постійна пам'ять* (ROM), в якій зберігається програма мікропроцесора та *оперативна пам'ять* (RAM), яка використовується для тимчасового зберігання програм та даних. Побудова моделі пам'яті під час проектування мікропроцесора зумовлена потребою в симуляції роботи всієї мікропроцесорної системи, що виконує програму.[10]

Розглянемо структуру нескладної мікропроцесорної системи (рис. 1), що складається з *мікропроцесора* **cpu**, *тактового генератора* **clk_gen** та *блока зовнішньої пам'яті* **mem**.

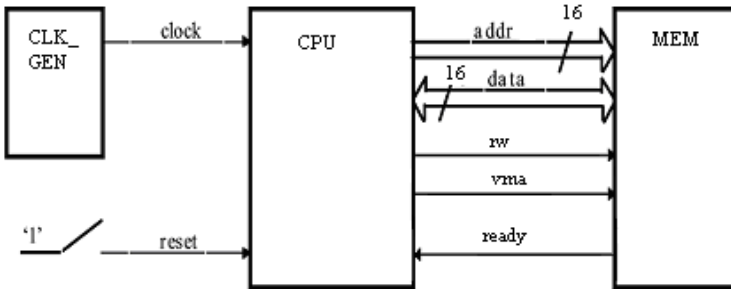


Рисунок 1.1 – Структурна схема мікропроцесорної системи

Сигнал **reset**, що надходить ззовні, призначений для скидання поточно-го стану керуючого пристрою мікропроцесора (МП) і початкового налаштування системи.

Наведена структура мікрокомп'ютера не є функціонально завершеною, оскільки відсутній інформаційний зв'язок поміж мікрокомп'ютером та зовнішнім оточенням. З погляду навчального проектування таке обмеження не є критичним. До того ж, запропоновану структуру мікропроцесора можна доповнити додатковими компонентами з метою розширення функціональних можливостей мікрокомп'ютера.

1.3 Інтерфейс мікропроцесора і зовнішньої пам'яті

Мікропроцесор має необхідну кількість зовнішніх сигналів (портів), що створюють *мікропроцесорний інтерфейс*.

Таблиця 1.1 – Інтерфейс мікропроцесора

clock	вх.	(1 біт)	тактовий сигнал
reset	вх.	(1 біт)	сигнал скидання
addr	вих.	(16 біт)	зовнішня магістраль адреси
data	вх./вих.	(16 біт)	загальна магістраль даних
rw	вих.	(1 біт)	сигнал роботи з пам'яттю: читання / запис
vma	вих.	(1 біт)	сигнал дозволу роботи з пам'яттю
ready	вх.	(1 біт)	сигнал готовності даних пам'яті

Інтерфейс модуля пам'яті наведений в табл.2.

Таблиця 1.2 - Інтерфейс зовнішньої пам'яті

addr	вх.	(16 біт)	адреса, що ідентифікує потрібну комірку пам'яті
data	вх./ вих	(16 біт)	вхідні / вихідні данні
rw	вх.	(1 біт)	задає тип операції з пам'яттю: читання (rw=0), або запис (rw=1)
vma	вх.	(1 біт)	сигнал дозволу читання / запису даних пам'яті
ready	вих.	(1 біт)	сигнал готовності модуля пам'яті надати дані

Обмін даними між МП та зовнішньою пам'яттю виконують 16-розрядними словами. Максимальний об'єм пам'яті обмежений розрядністю магістралі адреси МП: $2^{16} = 65536 = 64\text{К}$ 16-розрядних слів або 128 Кбайт.

Запис / читання слова мікропроцесором виконується впродовж двох *машиинних циклів* (МЦ). У програмній моделі процесора тривалість МЦ дорівнює періоду тактового сигналу **clock**, що виробляє генератор **clk_gen**.

Читання слова пам'яті (рис.1.2,а) виконується у такій послідовності:

МЦ1: після надходження зростаючого фронту тактового сигналу **clock** МП формує адресу комірки пам'яті на магістралі **addr (15:0)**, підтверджує адресу (**vma = '1'**) і вказує тип операції з пам'яттю (**rw = '0'**);

МЦ2: після надходження сигналу підтвердження даних (**ready = '1'**) МП зчитує данні із зовнішньої магістралі **data (15:0)** і скидає сигнали **addr (15:0)**, **rw** і **vma**.

Запис слова пам'яті (рис. 1.2,б) виконується у такій послідовності:

МЦ1: після надходження зростаючого фронту тактового сигналу **clock** МП формує адресу комірки пам'яті на магістралі **addr (15:0)**, виводить слово даних на магістраль **data (15: 0)**, підтверджує адресу (**vma = '1'**) і вказує тип операції з пам'яттю (**rw = '1'**);

МЦ2: у синхронних системах, до яких належить більшість обчислювачів, цей цикл може не виконуватись. В асинхронних системах він потрібен. Після надходження сигналу підтвердження запису даних від модуля пам'яті (**ready = '1'**) МП скидає сигнали **addr (15:0)**, **data (15:0)**, **rw** і **vma**. Якщо використовується синхронний обмін, ці сигнали МП скидає самостійно, після фіксованого часу затримки (один або більше періодів тактового сигналу **clock**).

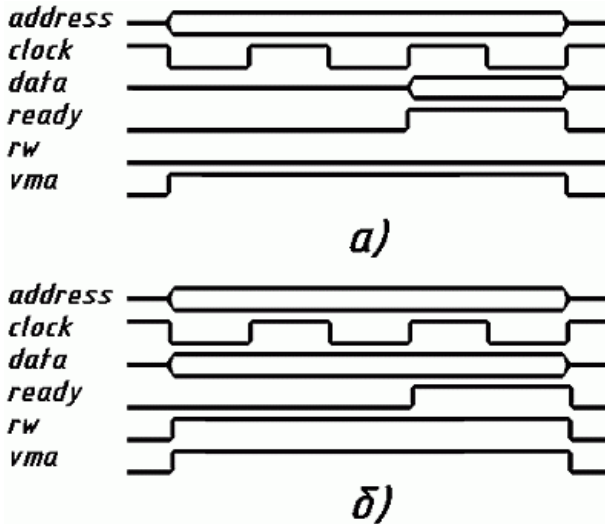


Рисунок 1.2 – Часові діаграми читання / запису даних зовнішньої пам'яті
а – цикл читання; б – цикл запису

Між початком циклу читання / запису і надходженням сигналу підтвердження даних **ready** існує *часова затримка*, яку можна вимірювати у кількох періодах тактового сигналу **clock**.

Зовнішня пам'ять забезпечує мікропроцесор командами і вхідними даними. Програма мікропроцесора зберігається у зовнішній пам'яті, починаючи з адреси 0000h. Після скидання МП автоматично завантажує першу інструкцію програми до регістру команд.

2 ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Створення проекту

2.1.1 Навчальний VHDL-проект містить п'ять текстових документів мовою VHDL, одну логічну схему та один командний файл запуску теста:

- **top.vhd** – структурний опис системи верхнього рівня;
- **icpu.vhd** – поведінковий опис імітатора мікропроцесора;
- **mem.vhd** – поведінковий опис модуля зовнішньої пам'яті;
- **clk_gen.vhd** – поведінковий опис генератора тактових сигналів;
- **top_TB.vhd** – файл випробувального стенду;

- **top_TB_runtest.do** – командний файл для запуску теста.

Логічна схема **top.bde** – це додатковий документ, ілюстрація, яка пояснює зміст файлу верхнього рівня ієрархії **top.vhd**. Файли **top_TB.vhd** та **top_TB_runtest.do** потрібні для автоматизації тестування проекту.

Після запуску програми Active-HDL з'явиться вікно САПР. У головному меню програми оберіть команду **File**. Можна або відкрити один з існуючих робочих просторів (англ. «workspace»), або створити новий. Для створення нового робочого простору оберіть **New -> Workspace**.

- 2.1.2 З'явиться вікно майстра **New Workspace wizard** (рис. 2.1). Введіть ім'я **робочого простору** (наприклад, **comp**), в якому буде знаходитись новий **проект** – поведінкова модель зовнішнього модуля пам'яті. Активізуйте опцію **Add New Design to Workspace** (англ. «додати новий проект до робочого простору»). Клацніть **OK**.

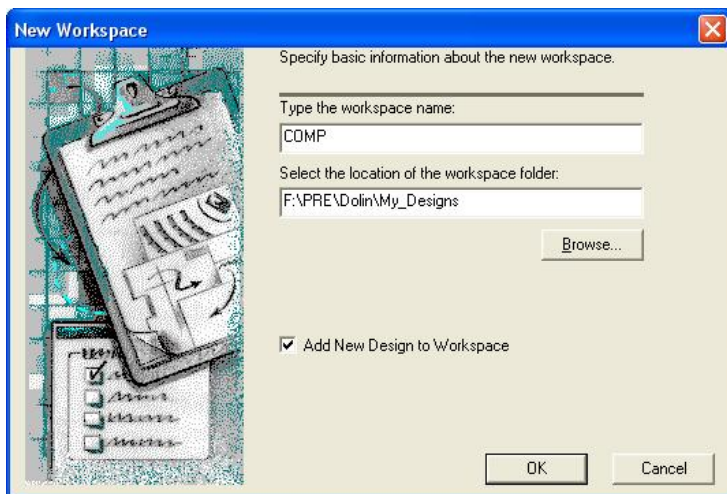


Рисунок 2.1 – Майстер New Workspace

Якщо робочий простір існує, то в меню **File** оберіть команду **Open New Workspace / Design Explorer**.

- 2.1.3 З'явиться вікно майстра нового проекту **New Design Wizard**, в якому визначається спосіб створення ресурсів проекту. Виберіть **Create an empty design** (англ. «створити порожній проект») і клацніть **Next**.
- 2.1.4 У наступному вікні **New Design Wizard** вказується інформація про інструменти синтезу і реалізації проекту, а також цільове сімейство мікро-

схем програмованої логіки. Перевірте, що обрана мова опису моделі є **VHDL**. Решту налаштувань залиште без змін (**none**). Клацніть **Next**.

- 2.1.5 У наступному вікні **New Design Wizard** (рис.2.2) треба вказати назву проекту (**MEM**) і встановити його місцезнаходження. Майстер створить робочу бібліотеку з тією ж самою назвою. Ця бібліотека міститиме всі компільовані компоненти VHDL-проекту. Клацніть **Next**.



Рисунок 2.2 – Введення назви проекту

- 2.1.6 В останньому вікні **New Design Wizard** виводиться інформація про налаштування, що були зроблені за допомогою майстра. Підготовчий етап проектування закінчений. Клацніть **Finish**.
- 2.1.7 З'явиться вікно **Design Browser**, в якому відображається ієрархічна, деревоподібна структура порожнього проекту **MEM**. Якщо клацнути на піктограмі  перед назвою проекту, то з'явиться піктограма **Add New File** (рис.2.3).

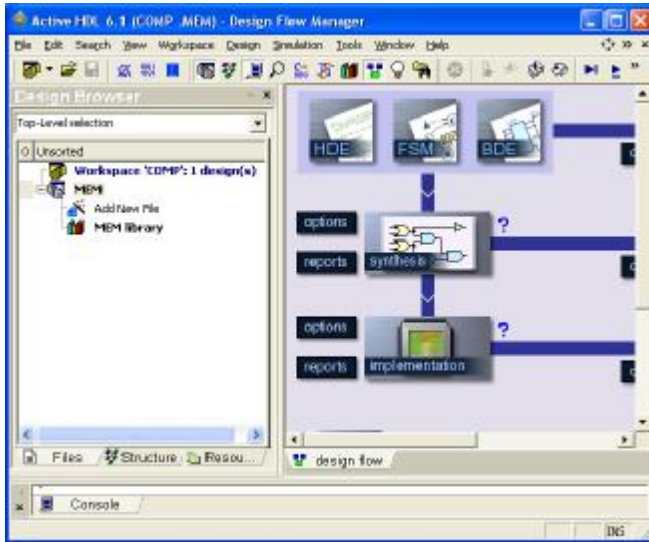


Рисунок 2.3– Вікно Design Browser

2.2 Модель зовнішньої пам'яті

2.2.1 Перейдемо до розробки файлів VHDL-проекту. Двічі клацніть на піктограмі **Add New File** у вікні **Design Browser**. У вікні **Add New File** оберіть сторінку **Wizards** і в ній оберіть **VHDL Source Code Wizard** (рис. 2.4). Клацніть **OK**.

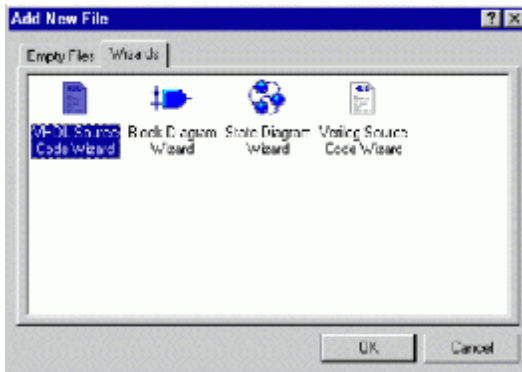


Рисунок 2.4 – Запуск майстра VHDL Source Code Wizard

2.2.2 З'явиться вікно майстра **New Source File Wizard**. Переконайтеся, що настройка **Add generated file to design** (англ. «додати створюваний файл до поточного проекту») активна і клацніть **Next**. У наступному вікні (рис. 2.5) треба вказати ім'я створюваного файлу, *сутності* (entity) і *архітектури* (architecture) моделі. Введіть **MEM** в полі імені файлу і клацніть **Next**.

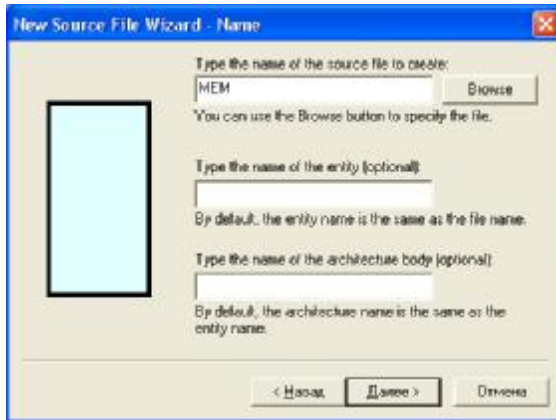


Рисунок 2.5 – Вибір імені файлу

2.2.3 Пам'ять повинна мати порти, що були описані в табл.1.2. Щоб додати новий *порт* (сигнал) в модель, клацніть кнопку **New**, надрукуйте ім'я сигналу в полі імені і оберіть напрям його розповсюдження (**in** – для входів, **out** – для виходів, **inout** – для двонаправлених портів). Якщо порт багаторозрядний (з'єднаний із магістраллю), треба вказати його розрядність за допомогою опцій **Array Indexes**, як показано на рис. 2.6. Кнопка **Type** дає змогу змінювати тип сигналу (залиште типи **STD_LOGIC** і **STD_LOGIC_VECTOR**). За допомогою кнопки **Delete** можна видаляти помилковий запис.

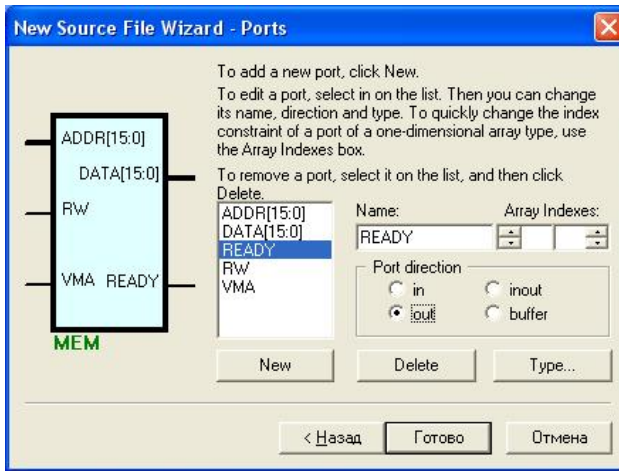


Рисунок 2.6 – Створення портів

2.2.4 Після клацання на кнопці **Finish** буде автоматично створений шаблон файлу опису моделі мовою VHDL (рис..2.7).

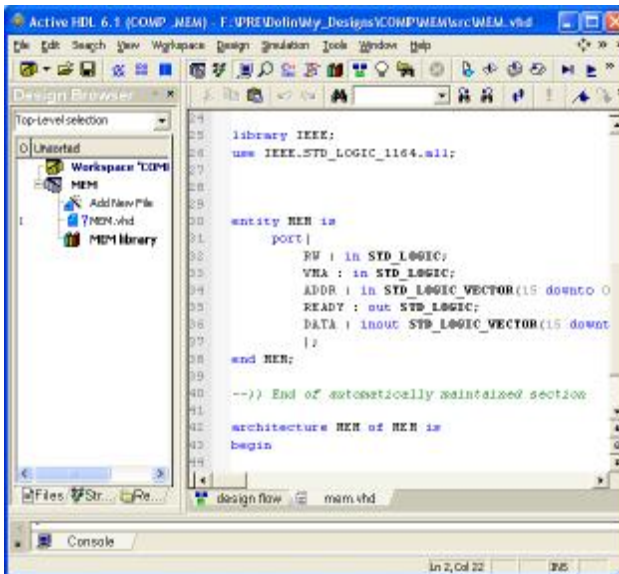


Рисунок 2.7. – Шаблон опису моделі в текстовому редакторі

2.2.5 Шаблон містить оголошення бібліотеки підпрограм багаторівневої логіки **USE IEEE.std_logic_1164.ALL**, опис портів і порожню область опису архітектури. Встановіть курсор у розділі опису архітектури (у рядку *-- enter your statements here --*) і наберіть текст опису архітектури (на лістингу 2.1 виділений жирним шрифтом).

Лістинг 2.1 – Файл mem.vhd

```
-----
-- Модуль зовнішньої пам'яті
-----
-- Декларація використаних в проекті бібліотек
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
-- Опис інтерфейсу об'єкта MEM
ENTITY mem IS
  PORT (
    |---|
    addr : IN STD_LOGIC_VECTOR (15 DOWNT0 0);    --addr-
    -|  |
    vma : IN STD_LOGIC;                          --vma -
    -|  |
    rw : IN STD_LOGIC;                          --rw -
    -|  |
    data : INOUT STD_LOGIC_VECTOR (15 DOWNT0 0);--
    |  |--data
    ready : OUT STD_LOGIC                        --
    |  |--ready
    );                                           --
    |---|
END mem ;
-- Опис поведінки об'єкта MEM
ARCHITECTURE mem OF mem IS
  TYPE t_mem IS ARRAY (0 TO 63) OF STD_LOGIC_VECTOR (15
DOWNT0 0);
  SIGNAL mem_data : t_mem :=
    (
      "0000000000000000", -- 0000h -- масив програми
      "0000000000000000", -- 0001h
      "0000000000000111", -- 0002h

```

```

"0000000000000000", -- 0003h
"0000000000000000", -- 0004h
"0000000000000000", -- 0005h
"0000000000000000", -- 0006h
"0000000000000000", -- 0007h
"0000000000000000", -- 0008h
"0000000000000000", -- 0009h
"0000000000000000", -- 000Ah
"0000000000000000", -- 000Bh
"0000000000000000", -- 000Ch
"0000000000000000", -- 000Dh
"0000000000000000", -- 000Eh
"0000000000000000", -- 000Fh

```

```

-----
"0000000000000000", -- 0010h - массив-джерело
"0000000000000000", -- 0011h
"0000000000000000", -- 0012h
"0000000000000000", -- 0013h
"0000000000000000", -- 0014h
"0000000000000000", -- 0015h
"0000000000000000", -- 0016h
"0000000000000000", -- 0017h
"0000000000000000", -- 0018h
"0000000000000000", -- 0019h
"0000000000000000", -- 001Ah
"0000000000000000", -- 001Bh
"0000000000000000", -- 001Ch
"0000000000000000", -- 001Dh
"0000000000000000", -- 001Eh
"0000000000000000", -- 001Fh

```

```

-----
"0000000000000000", -- 0020h
"0000000000000000", -- 0021h
"0000000000000000", -- 0022h
"0000000000000000", -- 0023h
"0000000000000000", -- 0024h
"0000000000000000", -- 0025h
"0000000000000000", -- 0026h
"0000000000000000", -- 0027h
"0000000000000000", -- 0028h

```

```

"0000000000000000", -- 0029h
"0000000000000000", -- 002Ah
"0000000000000000", -- 002Bh
"0000000000000000", -- 002Ch
"0000000000000000", -- 002Dh
"0000000000000000", -- 002Eh
"0000000000000000", -- 002Fh

```

```

-----
"0000000000000000", -- 0030h - місце призначення
"0000000000000000", -- 0031h
"0000000000000000", -- 0032h
"0000000000000000", -- 0033h
"0000000000000000", -- 0034h
"0000000000000000", -- 0035h
"0000000000000000", -- 0036h
"0000000000000000", -- 0037h
"0000000000000000", -- 0038h
"0000000000000000", -- 0039h
"0000000000000000", -- 003Ah
"0000000000000000", -- 003Bh
"0000000000000000", -- 003Ch
"0000000000000000", -- 003Dh
"0000000000000000", -- 003Eh
"0000000000000000" -- 003Fh

```

```

);
BEGIN
PROCESS (vma,rw)
BEGIN
IF vma = '1' THEN
IF rw = '1' THEN -- запис
mem_data (CONV_INTEGER(addr)) <= data;
ready <= '1';
ELSIF rw = '0' THEN -- читання
data <= mem_data (CONV_INTEGER(addr));
ready <= '1'; -- для асинхронного обміну
END IF;
ELSE
data <= "ZZZZZZZZZZZZZZZZ";
ready <= '0';
END IF;
END PROCESS;

```

END mem;

Стан зовнішньої пам'яті відображається за допомогою сигналу **mem_data**, що оголошений усередині розділу опису поведінки об'єкта MEM. Цей масив містить 64 16-розрядних слова. Це комірки модуля пам'яті. Під час звернення до пам'яті номер комірки визначається за допомогою функції **CONV_INTEGER**, що перетворює 16-розрядний бінарний вектор **addr** в індекс елементу масива – ціле число в діапазоні від 0 до 63. Функція перетворення **CONV_INTEGER** знаходиться у пакеті **STD_LOGIC_UNSIGNED**, який необхідно приєднати до опису моделі у розділі декларації використаних бібліотек.

2.2.6 Програма мовою VHDL створена і готова до трансляції. Щоб запустити компіляцію файла, клацніть правою кнопкою миші на його імені у вікні **Design Browser**, і оберіть команду **Compile** контекстного меню. Зелений значок поряд з ім'ям файла і відповідне повідомлення у вікні **Console** сигналізують про те, що компіляція виконана успішно. Компоненти моделі, що успішно відкомпілювалися, автоматично додаються до робочої бібліотеки проекту.

2.3 Модель генератора синхроімпульсів

- 2.3.1 Щоб додати новий файл до проекту, двічі клацніть на піктограмі **Add New File** у вікні **Design Browser**.
- 2.3.2 У вікні **Add New File** оберіть **VHDL Source Code Wizard** і клацніть **OK**.
- 2.3.3 Переконайтеся, що у вікні **New Source File Wizard** встановлена настройка **Add generated file to design**. Клацніть **Next** і в наступному вікні надрукуйте ім'я файла моделі – **clk_gen**. Клацніть **Next**.
- 2.3.4 Генератор **clk_gen** має лише один вихідний порт – **clock : OUT STD_LOGIC**. Задамо його так само, як це робили в п 2.3.3. Клацніть **Finish**.
- 2.3.5 У вікні **Design Browser** з'явиться новий файл **clk_gen.vhd** (рис. 2.8). Як у попередньому випадку, створений майстром **New Source File Wizard** файл автоматично відкривається у вікні текстового редактора.

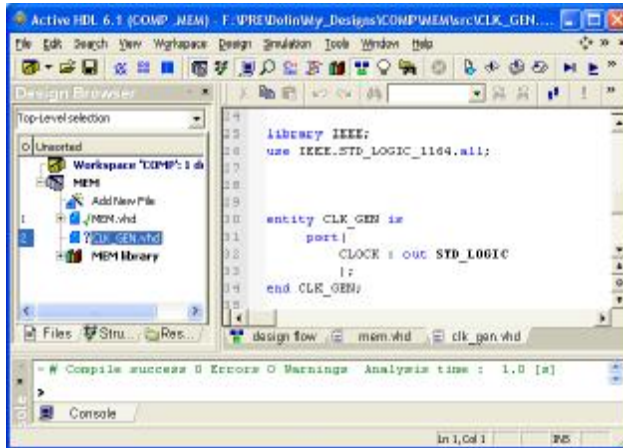


Рисунок 2.8– Додавання файлу clk_gen до проекту

Нижче рядка -- *enter your statements here* -- надрукуйте інструкції мовою VHDL, які описують поведінку модуля **clk_gen** (на лістингу 2.2 виділені жирним шрифтом).

Лістинг 2.2 – Файл clk_gen.vhd

 -- Генератор тактового сигналу

LIBRARY IEEE;
 USE IEEE.STD_LOGIC_1164.ALL;

ENTITY CLK_GEN IS

PORT(	--	---
CLOCK : OUT STD_LOGIC	--	--clock
);	--	---

END CLK_GEN;

--enter your statements here --

ARCHITECTURE CLK_GEN OF CLK_GEN IS

 SIGNAL clk : std_logic := '0';

BEGIN

 clk <= NOT clk AFTER 10 ns;

```
clock <= clk;
END CLK_GEN;
```

Генератор тактових сигналів МП (лістинг 2.2) автоматично формує сигнал **clock**, період якого визначає довжину МЦ: 20 ns. Він синхронізує роботу керуючого пристрою мікропроцесора. Від частоти сигналу **clock** залежить швидкодія МП.

2.3.6 Файл **clk_gen.vhd** повинен відкомпілюватися без помилок. Запустити компілятор можна за допомогою кнопки  **Compile** панелі інструментів, або так, як описано в п. 2.3.6. Результат трансляції буде автоматично доданий до бібліотеки VHDL-проекту.

2.4 Проектування імітатора мікропроцесора

2.4.1 Імітатор блока CPU формує сигнали запису даних до модуля пам'яті, а потім, через 50 ns, – відповідні сигнали читання. Двічі клацніть на піктограмі **Add New File** у вікні **Design Browser**. На панелі вікна **Add New File** оберіть не **Wizards**, а **Empty Files** (порожній файл). У новому вікні оберіть **VHDL Source Code**, а у вікні **Name** надрукуйте ім'я нового файлу **icpu** (рис. 2.9). Клацніть **OK**.

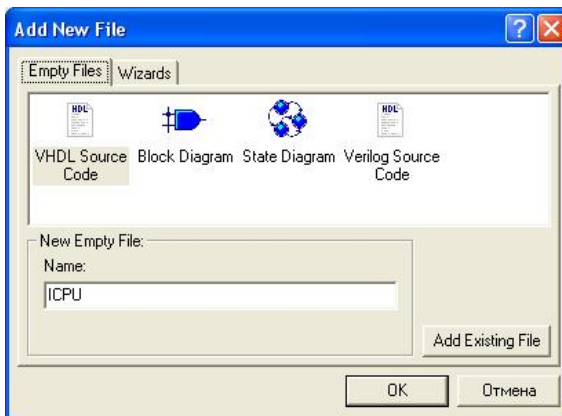


Рисунок 2.9 – Запуск майстра VHDL Source Code

2.4.2 У вікні текстового редактора введіть опис блока **icpu** (лістинг 2.3) і виконайте компіляцію файла **icpu.vhd**.

Лістинг 2.3 – Файл icpu.vhd

```

-----
-- Імітатор процесора
-----
-----
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
ENTITY icpu IS
  PORT
    (
      |---|
      (clock : IN STD_LOGIC;           --clock-
    -|  |
      reset : IN STD_LOGIC;           --reset-
    -|  |
      ready : IN STD_LOGIC;           --ready-
    -|  |
      addr  : OUT  STD_LOGIC_VECTOR (15 DOWNT0 0); --
      |  |--addr
      vma   : OUT  STD_LOGIC;           --
      |  |--vma
      rw    : OUT  STD_LOGIC;           --
      |  |--rw
      data  : INOUT STD_LOGIC_VECTOR (15 DOWNT0 0)--
      |  |--data
      );
    -|  |
    |---|
END icpu;

  architecture icpu of icpu is
    TYPE st_type is (st1,st2,st3,st4,st5);
    signal icpu_data : STD_LOGIC_VECTOR(15
downto 0):=
      "00000000000000011";
    signal st: st_type:= st1;
  begin
    process(reset,clock)
    begin
      if reset='1' then
        data <= "ZZZZZZZZZZZZZZZZ";

```

```

addr <= "000000000000000000";
vma <= '0';
rw <= '0';
    st <= st1;
elsif clock'event and clock = '1' then
    case st is
        when st1 =>
            vma <= '1';
            rw <= '1';    -- запис
            data <= icpu_data ;
            addr <= "000000000000000000";
            st <= st2;
        when st2 =>
            vma <= '0';
            rw <= '0';
            data <= "ZZZZZZZZZZZZZZZZ";
            st <= st3;
        when st3 =>
            vma <= '1';    -- читання
            addr <= "000000000000000010";
            st <= st4;
        when st4 =>
            vma <= '1';
            if ready = '1' then
                icpu_data <= data;
                st <= st5;
            else
                st <= st4;
            end if;
        when st5 =>
            vma <= '0';

-----
-----

        when others => null;
    end case;
end if;
end process;
end icpu;

```

Робота (**reset=1**) імітатора ісру виконується відповідно до його стану st. Зміна стану відбувається послідовно під час зростаючого фронту сигналу **clock**. В стані **st1** імітатор виставляє сигнали запису в пам'ять **mem** (**vma=1**,

rw=1), а дані **icpu_data =0002h** - на шину даних **data**, а в стані **st3** під час зростаючого фронту сигналу **clock** виставляє сигнали читання з пам'яті (**vma=1, rw=0**) і в стані **st4** після надходження з пам'яті сигналу **ready=1** зчитує дані з шини **data (icpu_data<= data)**. Якщо **ready≠1**, то **icpu** буде чекати поки пам'ять не виставить на шину **data** дані (**ready=1**). Якщо з'являється сигнал скидання (**reset=1**), то сигнали імітатора **icpu** скидаються через період сигналу **clock** в стані **st5** в початковий стан і пам'ять відключається (**vma=0**).

2.5 Файл верхнього рівня проекту

Тепер можна приступити до проектування моделі мікрокомп'ютера. Файл верхнього рівня проекту (опис моделі мікрокомп'ютера) можна ввести або у вікні текстового редактора (**HDL Editor**), або за допомогою редактора логічних схем (**Block Diagram Editor**).

У першому випадку опис мікропроцесорної системи складається з декларацій компонентів системи, внутрішніх сигналів та з'єднань компонентів, виконаних за допомогою операторів **PORT MAP** (лістинг 2.4). Другий варіант створення моделі верхнього рівня проекту наведений нижче.

Лістинг 2.4 – Файл top.vhd

```
-----
-- файл верхнього рівня проекту
-----
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
-----
ENTITY top IS
    PORT
    (
        reset : IN STD_LOGIC
    );
END top;
ARCHITECTURE top OF top IS
-----
    COMPONENT clk_gen -- компонент CLK_GEN
    PORT
    (
        clock : OUT STD_LOGIC
    );
    END COMPONENT;
-----
    COMPONENT mem -- компонент MEM
    PORT
    (
        addr : IN STD_LOGIC_VECTOR (15 DOWNT0 0);
```

```

vma : IN STD_LOGIC;
rw : IN STD_LOGIC;
data : INOUT STD_LOGIC_VECTOR (15 DOWNT0 0);
ready: OUT STD_LOGIC
);
END COMPONENT;
-----
COMPONENT icpu -- компонент ICPU
PORT
(
clock : IN STD_LOGIC;
ready : IN STD_LOGIC;
reset : IN STD_LOGIC;
data : INOUT STD_LOGIC_VECTOR(15 DOWNT0 0);
rw : OUT STD_LOGIC;
vma : OUT STD_LOGIC;
addr : OUT STD_LOGIC_VECTOR(15 DOWNT0 0)
);
END COMPONENT;
-----
SIGNAL clock: STD_LOGIC; -- Тактовий сигнал
SIGNAL addr : STD_LOGIC_VECTOR (15 DOWNT0 0);-- Шина адреси
SIGNAL data : STD_LOGIC_VECTOR (15 DOWNT0 0);-- Шина даних
SIGNAL vma : STD_LOGIC; -- Фіксація адреси
SIGNAL rw : STD_LOGIC; -- 0 - читання;1 - запис
SIGNAL ready : STD_LOGIC; -- Підтвердження даних
BEGIN
-----
g1 : clk_gen -- Підключення CLK_GEN
PORT MAP
(
clock => clock
);
-----
m1 : mem -- Підключення MEM
PORT MAP
(
addr => addr ,
vma => vma ,
rw => rw ,
ready => ready,
data => data
);
-----
u1 : icpu -- Підключення ICPU
PORT MAP
(
clock => clock,
reset => reset,
ready => ready,
addr => addr ,
rw => rw,
vma => vma,

```

```
data => data
);
```

```
-----
PROTOCOL: PROCESS (clock)
BEGIN
  IF clock = '1' THEN
    IF rw = '0' THEN
      REPORT "READ RAM" SEVERITY WARNING;
    ELSIF rw = '1' THEN
      REPORT "WRITE RAM" SEVERITY ERROR;
    END IF;
  END IF;
END PROCESS;
END top;
```

- 2.5.1 Логічна схема є наочним документом. Тому порядок її складання є корисним для навчального проектування. Для створення логічної схеми двічі клацніть на піктограмі **Add New File** у вікні **Design Browser**. У новому вікні (рис. 2.10) оберіть спосіб опису моделі – **Block Diagram Wizard**. Модель мікрокомп'ютера буде представлена у вигляді логічної схеми. Клацніть **OK**.

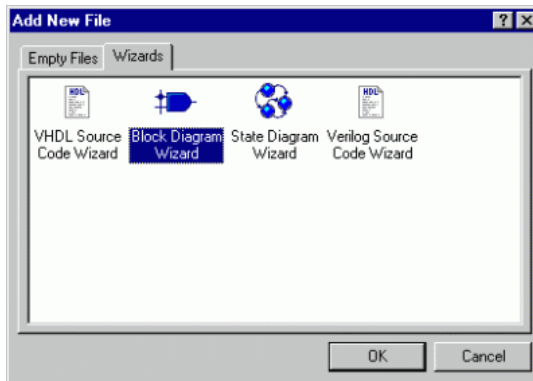


Рисунок 2.10 – Запуск майстра Block Diagram Wizard

- 2.5.2 Переконайтеся, що у вікні **New Source File Wizard** активізована настройка *Add generated file to design* і клацніть **Next**.
- 2.5.3 Логічна схема, яку створить майстер, може бути перетворена в один з трьох форматів опису: EDIF, VHDL і Verilog. Оберіть мову **VHDL** і натисніть **Next**.
- 2.5.4 Як раніше це робилося для компонентів, у наступному вікні вкажіть ім'я файлу **top** клацніть **Next**.

- 2.5.5 На наступному кроці треба задати порти мікрокомп'ютера, користуючись кнопкою **New**. МП має тільки один вхідний порт: **reset : IN STD_LOGIC** . Введіть його і клацніть **Finish**.
- 2.5.6 У вікні **Design Browser** з'явиться новий файл – **top.bde**. Шаблон нової логічної схеми буде автоматично завантажений у вікно редактора BDE (див.рис. 2.11). Він містить тільки один раніше оголошений порт **reset**.
- 2.5.7 Інструментальна панель редактора BDE містить кнопку  **Symbol toolbox**, яка дозволяє поміщати у логічну схему раніше згенеровані моделі – *блоки*. Клацніть на кнопці **Symbol toolbox**. Зверніть увагу, що у вікні, яке з'явилося (рис. 2.12), вже є блоки **clk_gen**, **cpu** і **mem**.

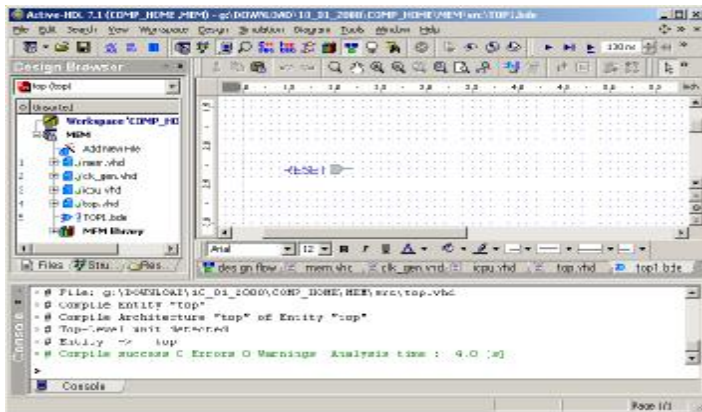


Рисунок 2.11 – Шаблон логічної схеми

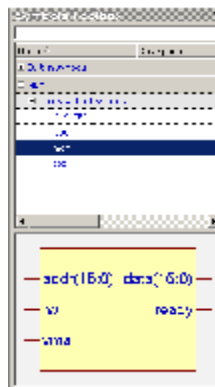


Рисунок 2.12– Вікно Symbol Toolbox

2.5.8 Перетягніть блоки **clk_gen**, **cpu** і **mem** на робоче поле редактора **BDE** і розташуйте, як показано на рис. 2.13. Закрийте вікно **Symbol Toolbox**.

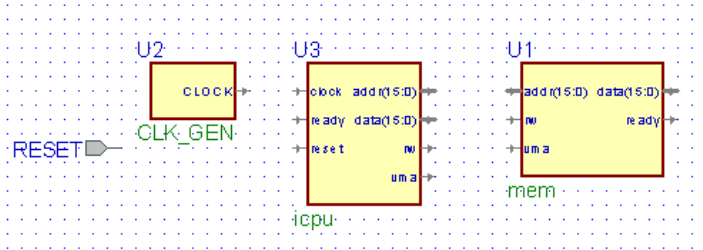


Рисунок 2.13 – Розміщення компонентів логічної схеми

2.5.9 Виконаємо з'єднання компонентів логічної схеми. Один провідник прокладається за допомогою інструменту **Wire** редактора BDE. Для виконання з'єднання треба активізувати кнопку **Wire** панелі інструментів, клацнути на контакт-джерелі, перемістити курсор і клацнути на контакт-одержувачі. До логічної схеми буде додано з'єднання. Щоб видалити помилково виконане з'єднання, треба використовувати інструмент **Select**. За його допомогою виділіть компонент логічної схеми і, натиснувши кнопку **DELETE** клавіатури, видаліть його. Зробіть з'єднання, як показано на рис. 2.14.

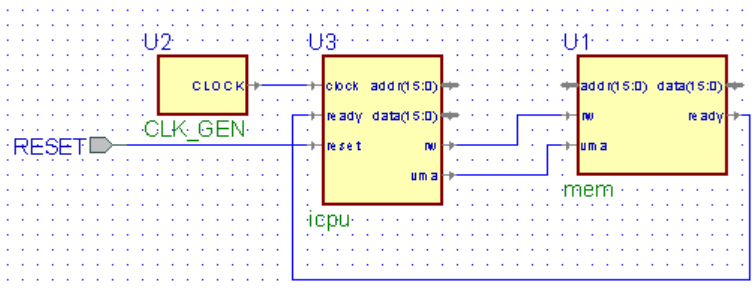


Рисунок 2.14 – З'єднання компонентів логічної схеми

2.5.10 На логічній схемі можна розміщати *глобальні сигнали*. Глобальним називається сигнал, який присутній на всіх однойменних вузлах схеми. Тому немає необхідності формувати відповідні з'єднання. Достатньо,

щоб пов'язані вузли логічної схеми мали однакову назву. Додати глобальний сигнал до схеми можна за допомогою інструменту  **Global Wire**. Щоб дати назву глобального сигналу оберіть інструмент **Select** і двічі клацніть на імені **Global0**. Змініть цю назву на необхідну. На рис. 2.15 наведений варіант еквівалентної логічної схеми, у якій замість з'єднань використовуються глобальні сигнали.

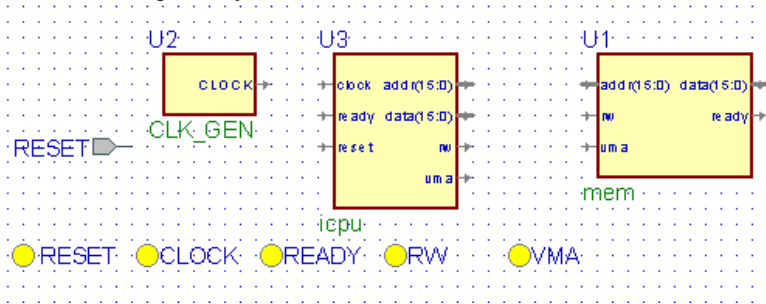


Рисунок 2.15 – Глобальні сигнали логічної схеми

2.5.11 Для з'єднання багаторозрядних портів логічної схеми треба використовувати магістралі відповідної розрядності. Використовують спеціальний інструмент –  **Bus** (шина), правила роботи з яким такі самі, як і з інструментом **Wire**.

Під час складання логічної схеми треба було виконати наступні з'єднання:

- вихід **clock** генератора **clk_gen** з'єднати з входом **clock** імітатора мікропроцесора **icpu** (інструмент **Wire**);
- вихід **addr (15:0)** імітатора мікропроцесора **icpu** з'єднати з входом **addr (15:0)** модуля пам'яті **mem** (інструмент **Bus**);
- вхід / вихід **data (15:0)** імітатора мікропроцесора **icpu** з'єднати з портом **data (15:0)** модуля пам'яті **mem** (інструмент **Bus**);
- вихід **rw** імітатора мікропроцесора **icpu** з'єднати з входом **rw** модуля пам'яті **mem** (інструмент **Wire**);
- вихід **vma** імітатора мікропроцесора **icpu** з'єднати з входом **vma** модуля пам'яті **mem** (інструмент **Wire**);
- вихід **ready** модуля пам'яті **mem** з'єднати з входом **ready** імітатора мікропроцесора **icpu** (інструмент **Wire**);
- порт **reset** логічної схеми з'єднати з входом **reset** імітатора мікропроцесора **icpu** (інструмент **Wire**).

Закінчена логічна схема повинна мати вигляд, як показано на рис. 2.16.



Рисунок 2.17 – Додавання нового файлу до проекту

- 2.6.3 У вікні **Design Browser** з'явиться новий файл **top.vhd** (рис. 2.8). Його зміст автоматично відкриється у вікні редактора.
- 2.6.4 Скопіюйте зміст лістинга 1.4 до створеного файлу і виконайте компіляцію проекту у режимі **Compile All with File Reorder**. Трансляція повинна виконатись успішно. Біля імені файлу з'явиться зелена піктограма.
- 2.6.5 Оберіть команду **Code2Graphics Conversion Wizard** пункту **Tools** головного меню САПР. З'явиться перше вікно майстра (рис. 2.18).

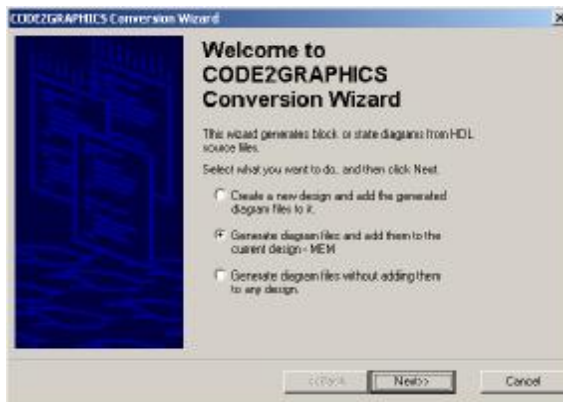


Рисунок 2.18 – Перше вікно майстра перетворення структурного опису

- 2.6.6 Залиште налаштування **Generate diagram files and add them to the current design** (англ. «згенерувати логічну схему і додати її до поточно-го проекту») і клацніть **Next**.

2.6.7 У наступному вікні клацніть кнопку **Add**, щоб у діалозі відкриття файла обрати опис **top.vhd**. Після того, як обраний файл структурного опису був доданий до списку вікна (див. рис. 2.19), клацніть **Finish**.



Рисунок 2.19 – Друге вікно майстра Code2Graphics

2.6.8 У вікні **Design Browser** з'явилась піктограма нового документа – **top.bde**. Зображення створеної логічної схеми можна побачити, як і звичайні документи, у вікні редактора САПР (див. рис. 2.20). Якщо є бажання ознайомитись з протоколом, який був створений майстром **Code2Graphics** під час роботи, клацніть на рядку «# CODE2GRAPHICS: Double click on this line to view the generated log file» у вікні **Console**

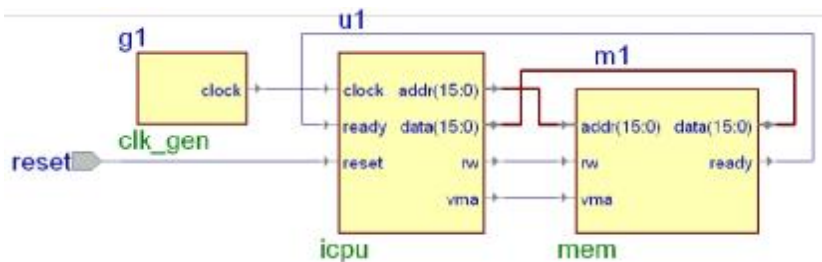


Рисунок 2.20 – Логічна схема об'єкта, що проектується

2.6.9 Модель мікропроцесорної системи створена. VHDL-проект повинен містити файли-описи чотирьох компонентів:

- **top (top)** – структурний опис системи верхнього рівня;

- **icpu (icpu)** – компонент імітатора мікропроцесора;
- **mem (mem)** – модуль зовнішньої пам'яті;
- **clk_gen (clk_gen)** – генератор тактових сигналів.

2.6.10 Щоб запобігти подвійному визначенню файла верхнього рівня проекту, у вікні **Design Browser** необхідно виконати «чистку» – видалити непотрібні копії компонента **top**. Пам'ятайте, що документ логічної схеми об'єкта є лише «ілюстрацією», яка у наочній формі дозволяє ознайомитися із структурним описом мовою VHDL цього ж самого об'єкта. Для видалення об'єкта з проекту необхідно клацнути на його піктограмі **правою** кнопкою миші і в контекстному меню вибрати команду **Delete**.

2.6.11 Відновіть зміст бібліотеки проекту. Спочатку, клацнувши **правою** кнопкою миші на піктограмі бібліотеки, оберіть команду **Delete simulation data** контекстного меню. Потім знову виконайте компіляцію проекту у режимі **Compile All with File Reorder**. У випадку безпомилкової трансляції бібліотека проекту буде створена знову. Вона міститиме усі необхідні дані для функціонального моделювання створеної мікропроцесорної системи.

2.7 Функціональне моделювання

2.7.1 На початку функціонального моделювання треба визначити *об'єкт верхнього рівня* (Top-Level) – пару описів сутності (entity) і архітектури (architecture), відносно якої це моделювання буде виконуватися. У даному випадку моделюється поведінка всієї мікропроцесорної системи, якій відповідає об'єкт **top (top)**. У вікні **Design Browser** розкрийте зміст компоненту **top**, клацнувши символ  перед його ім'ям. Оберіть пару **top (top)** і клацніть **правою** кнопкою миші. У контекстному меню оберіть команду **Set as Top-Level** (англ. «встановити в якості файлу верхнього рівня»).

2.7.2 Виконайте ініціалізацію моделювання. Для цього оберіть команду **Simulation** → **Initialize Simulation** головного меню САПР. Після ініціалізації стане активною сторінка **Structure** у вікні **Design Browser**.

2.7.3 Відкрийте вікно редактора *часових діаграм* **Waveform Editor**. Для цього оберіть команду **File** → **New** → **Waveform**, або клацніть на кнопці  **New Waveform** панелі інструментів Active-HDL. Клацніть на імені **top** у верхній частині вікна **Design Browser** сторінки **Structure**. Внизу сторінки можна побачити сигнали вибраного модуля (див. рис. 2.21). Таким же чином на цій сторінці можна побачити підпорядковані обраному компоненту модулі (**clk_gen(clk_gen)**, **mem(mem)**, **icpu(icpu)**) і їх сигнали.

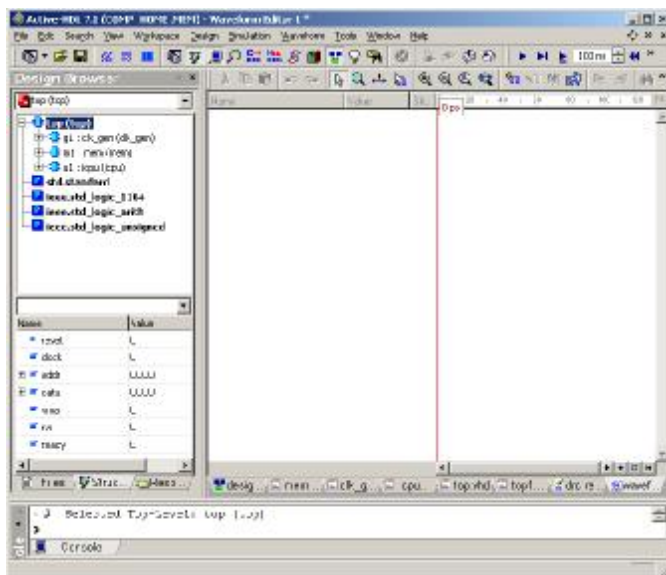


Рисунок 2.21 – Додавання сигналів до часової діаграми

2.7.4 Додайте сигнали із нижньої частини вікна **Design Browser** у вікно часової діаграми (див. табл. 2.1).

Таблиця 2.1 – Сигнали часової діаграми

Сигнал	Вх./вих./сигнал	Біт	Компонент
<i>reset</i>	Вхід	1 біт	<i>top (top)</i>
<i>clock</i>	сигнал	1 біт	<i>top (top)</i>
<i>rw</i>	сигнал	1 біт	<i>top (top)</i>
<i>st</i>	сигнал	st_type	<i>icpu (icpu)</i>
<i>vma</i>	сигнал	1 біт	<i>top (top)</i>
<i>ready</i>	сигнал	1 біт	<i>top (top)</i>
<i>icpu_data</i>	сигнал	(16 біт)	<i>icpu (icpu)</i>
<i>data</i>	сигнал	(16 біт)	<i>top (top)</i>
<i>mem_data (0)</i>	сигнал	(16 біт)	<i>mem (mem)</i>

Додавання сигналів до часової діаграми виконується за допомогою технології «перетягування» (drag and drop). Наприклад, щоб додати

до часової діаграми сигнал **reset**, клацніть на ньому і перетягніть виділений сигнал у вікно **Waveform Editor**.

2.7.5 Результати моделювання можуть відображатися не тільки у вигляді часової діаграми. Щоб виводити результати моделювання у символьному вигляді, використовуються *таблиці перемикань* (вікно **List**). Для створення таблиці перемикань клацніть на кнопці  **New List** панелі інструментів САПР. Так само, як це робилося під час додавання сигналів до **Waveform Editor**, додайте до таблиці **List** сигнали, які перелічені у табл.2.1.

2.7.6 Поверніться до вікна редактора часових діаграм. Оберіть сигнал **reset** і, клацнувши на ньому **правою** кнопкою миші, у контекстному меню оберіть команду **Stimulators...** З'явиться вікно **Stimulators**, де можна задавати тестові послідовності для обраних сигналів (рис. 2.22).

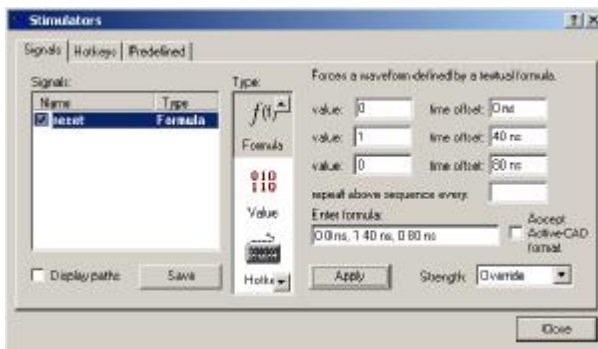


Рисунок 2.22 – Настроювання стимулятора для сигналу reset

2.7.7 В полі *Type* вікна **Stimulators** для сигналу **reset** оберіть тип стимулятора – **Formula**. У відповідне вікно введіть формулу **0 0 ns, 1 100 ns**. Цей запис означає, що у час моделювання **0 ns** сигнал **reset** має рівень лог.0. В момент часу **100 ns** він змінюється на лог.1 Для підтвердження внесених змін клацніть **Apply**.

2.7.8 Для виконання моделювання в САПР Active-HDL використовуються п'ять кнопок, розташованих на панелі інструментів (рис. 2.23). Зверніть увагу на комбінований список поряд з кнопкою  **Run for**. Необхідно змінити заданий за умовчанням крок моделювання **100 ns** на нове значення – **150 ns**. Для цього клацніть на комбінованому списку і надрукуйте нове значення.

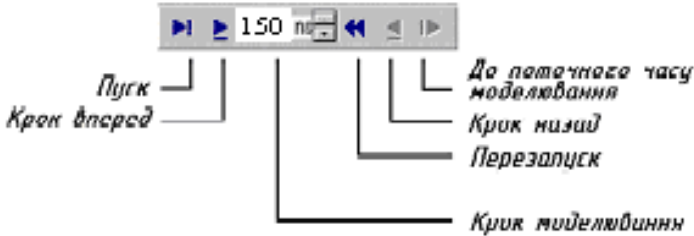


Рисунок 2.23 – Кнопки управління моделюванням

2.7.9 Клацнувши кнопку  **Run for**, виконайте моделювання при заданих настройках вхідних сигналів протягом 150 наносекунд. Результати моделювання будуть відображені у вікні **Waveform Editor** (рис. 2.24). На відрізках часу моделювання (10 ns .. 50 ns) виконується цикл запису зовнішньої пам'яті. На протязі 50 ns .. 90 ns часу моделювання – цикл її читання.

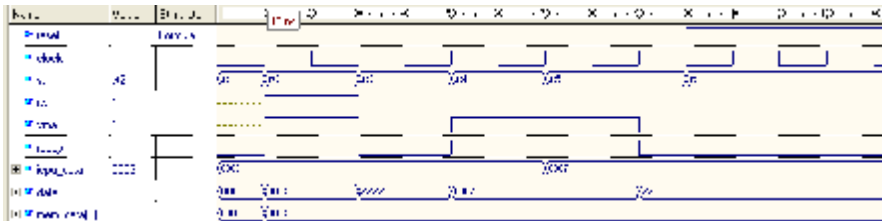


Рисунок 2.24 – Результат моделювання у ручному режимі

2.7.10 Якщо перейти до вікна **List**, можна побачити інший формат опису результатів моделювання – таблицю перемикань (див. рис. 2.25).

Time (ns)	reset	clk_gen	cpu	mem	data	mem	data	mem	data
0.000	0	0	0	0	0000	0000	0000	0000	0000
10.000	0	1	1	0	0000	0000	0000	0000	0000
20.000	0	0	0	0	0000	0000	0000	0000	0000
30.000	0	1	1	0	0000	0000	0000	0000	0000
40.000	0	0	0	0	0000	0000	0000	0000	0000
50.000	0	1	1	0	0000	0000	0000	0000	0000
60.000	0	0	0	0	0000	0000	0000	0000	0000
70.000	0	1	1	0	0000	0000	0000	0000	0000
80.000	0	0	0	0	0000	0000	0000	0000	0000
90.000	0	1	1	0	0000	0000	0000	0000	0000
100.000	0	0	0	0	0000	0000	0000	0000	0000
110.000	0	1	1	0	0000	0000	0000	0000	0000
120.000	0	0	0	0	0000	0000	0000	0000	0000
130.000	0	1	1	0	0000	0000	0000	0000	0000
140.000	0	0	0	0	0000	0000	0000	0000	0000
150.000	0	1	1	0	0000	0000	0000	0000	0000
160.000	0	0	0	0	0000	0000	0000	0000	0000
170.000	0	1	1	0	0000	0000	0000	0000	0000
180.000	0	0	0	0	0000	0000	0000	0000	0000
190.000	0	1	1	0	0000	0000	0000	0000	0000
200.000	0	0	0	0	0000	0000	0000	0000	0000
210.000	0	1	1	0	0000	0000	0000	0000	0000
220.000	0	0	0	0	0000	0000	0000	0000	0000
230.000	0	1	1	0	0000	0000	0000	0000	0000
240.000	0	0	0	0	0000	0000	0000	0000	0000
250.000	0	1	1	0	0000	0000	0000	0000	0000
260.000	0	0	0	0	0000	0000	0000	0000	0000
270.000	0	1	1	0	0000	0000	0000	0000	0000
280.000	0	0	0	0	0000	0000	0000	0000	0000
290.000	0	1	1	0	0000	0000	0000	0000	0000
300.000	0	0	0	0	0000	0000	0000	0000	0000
310.000	0	1	1	0	0000	0000	0000	0000	0000
320.000	0	0	0	0	0000	0000	0000	0000	0000
330.000	0	1	1	0	0000	0000	0000	0000	0000
340.000	0	0	0	0	0000	0000	0000	0000	0000
350.000	0	1	1	0	0000	0000	0000	0000	0000
360.000	0	0	0	0	0000	0000	0000	0000	0000
370.000	0	1	1	0	0000	0000	0000	0000	0000
380.000	0	0	0	0	0000	0000	0000	0000	0000
390.000	0	1	1	0	0000	0000	0000	0000	0000
400.000	0	0	0	0	0000	0000	0000	0000	0000
410.000	0	1	1	0	0000	0000	0000	0000	0000
420.000	0	0	0	0	0000	0000	0000	0000	0000
430.000	0	1	1	0	0000	0000	0000	0000	0000
440.000	0	0	0	0	0000	0000	0000	0000	0000
450.000	0	1	1	0	0000	0000	0000	0000	0000
460.000	0	0	0	0	0000	0000	0000	0000	0000
470.000	0	1	1	0	0000	0000	0000	0000	0000
480.000	0	0	0	0	0000	0000	0000	0000	0000
490.000	0	1	1	0	0000	0000	0000	0000	0000
500.000	0	0	0	0	0000	0000	0000	0000	0000
510.000	0	1	1	0	0000	0000	0000	0000	0000
520.000	0	0	0	0	0000	0000	0000	0000	0000
530.000	0	1	1	0	0000	0000	0000	0000	0000
540.000	0	0	0	0	0000	0000	0000	0000	0000
550.000	0	1	1	0	0000	0000	0000	0000	0000
560.000	0	0	0	0	0000	0000	0000	0000	0000
570.000	0	1	1	0	0000	0000	0000	0000	0000
580.000	0	0	0	0	0000	0000	0000	0000	0000
590.000	0	1	1	0	0000	0000	0000	0000	0000
600.000	0	0	0	0	0000	0000	0000	0000	0000
610.000	0	1	1	0	0000	0000	0000	0000	0000
620.000	0	0	0	0	0000	0000	0000	0000	0000
630.000	0	1	1	0	0000	0000	0000	0000	0000
640.000	0	0	0	0	0000	0000	0000	0000	0000
650.000	0	1	1	0	0000	0000	0000	0000	0000
660.000	0	0	0	0	0000	0000	0000	0000	0000
670.000	0	1	1	0	0000	0000	0000	0000	0000
680.000	0	0	0	0	0000	0000	0000	0000	0000
690.000	0	1	1	0	0000	0000	0000	0000	0000
700.000	0	0	0	0	0000	0000	0000	0000	0000
710.000	0	1	1	0	0000	0000	0000	0000	0000
720.000	0	0	0	0	0000	0000	0000	0000	0000
730.000	0	1	1	0	0000	0000	0000	0000	0000
740.000	0	0	0	0	0000	0000	0000	0000	0000
750.000	0	1	1	0	0000	0000	0000	0000	0000
760.000	0	0	0	0	0000	0000	0000	0000	0000
770.000	0	1	1	0	0000	0000	0000	0000	0000
780.000	0	0	0	0	0000	0000	0000	0000	0000
790.000	0	1	1	0	0000	0000	0000	0000	0000
800.000	0	0	0	0	0000	0000	0000	0000	0000
810.000	0	1	1	0	0000	0000	0000	0000	0000
820.000	0	0	0	0	0000	0000	0000	0000	0000
830.000	0	1	1	0	0000	0000	0000	0000	0000
840.000	0	0	0	0	0000	0000	0000	0000	0000
850.000	0	1	1	0	0000	0000	0000	0000	0000
860.000	0	0	0	0	0000	0000	0000	0000	0000
870.000	0	1	1	0	0000	0000	0000	0000	0000
880.000	0	0	0	0	0000	0000	0000	0000	0000
890.000	0	1	1	0	0000	0000	0000	0000	0000
900.000	0	0	0	0	0000	0000	0000	0000	0000
910.000	0	1	1	0	0000	0000	0000	0000	0000
920.000	0	0	0	0	0000	0000	0000	0000	0000
930.000	0	1	1	0	0000	0000	0000	0000	0000
940.000	0	0	0	0	0000	0000	0000	0000	0000
950.000	0	1	1	0	0000	0000	0000	0000	0000
960.000	0	0	0	0	0000	0000	0000	0000	0000
970.000	0	1	1	0	0000	0000	0000	0000	0000
980.000	0	0	0	0	0000	0000	0000	0000	0000
990.000	0	1	1	0	0000	0000	0000	0000	0000
1000.000	0	0	0	0	0000	0000	0000	0000	0000

Рисунок 2.25 – Таблица перемикань

2.7.11 Ще більш наочно результати моделювання відображаються на документі логічної схеми **top.bde** (рис. 2.26). Якщо змінити крок часу моделювання на **10 ns** (півперіоду сигналу **clock**) і послідовно натискати кнопку **Run for**, можна спостерігати зміну станів компонентів мікропроцесорної системи на документі логічної схеми.

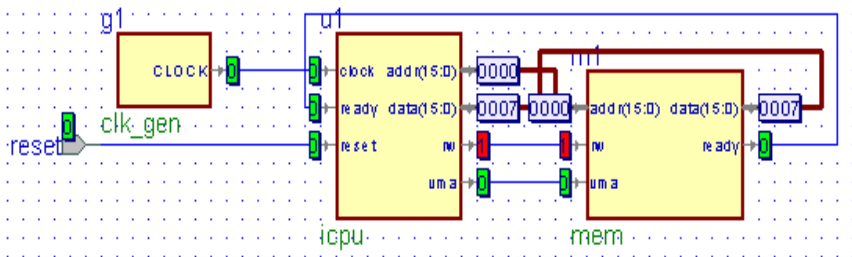


Рисунок 2.26 – Результати моделювання на логічній схемі

2.7.12 Користуючись вбудованими засобами візуалізації результатів моделювання, вивчіть принцип функціонування створеної моделі мікропроцесорної системи.

2.8 Створення випробувального стенда

Результати моделювання можуть бути збережені у «рідному» форматі часових діаграм САПР Active-HDL – файлі, що має розширення **.awf**. Для цього треба вибрати команду **File** → **Save** головного меню. Також є можливість експортувати результати моделювання у файл *VHDL-процесу*, який має розширення **.vhs**. Ці файли можна використовувати для створення *випробувального стенду*.

Випробувальний стенд VHDL (англ. «test bench») – це програма мовою VHDL, в якій описані *тестові послідовності* – закон зміни вхідних сигналів. У САПР Active-HDL тестові дані для моделювання можна або читати з текстового файлу, або експортувати з файлу *VHDL-процесу*.

Фактично **випробувальний стенд** – це файл мовою VHDL, який додається до створеного проекту і займає найвищий ступінь його ієрархії. За рахунок цього поведінка усіх компонентів VHDL-проекту підпорядковується тестовій послідовності, яку створює випробувальний стенд.

Вихідний VHDL-проект, роботу якого перевіряють за допомогою випробувального стенда, зветься **UUT** – *Unit Under Test* (англ. «пристрій, що тестується»).

Використання випробувального стенда дозволяє поліпшити якісні показники функціонального моделювання за рахунок його автоматизації. Наприклад, за його допомогою можна виконувати аналіз сигналів на виході UUT. Ця методика може забезпечити найповнішу перевірку проекту з мінімальною взаємодією із користувачем, тобто виключити з результатів перевірки проекту помилки, пов'язані з «людським фактором».

2.8.1 Зробіть активним вікно часової діаграми і видаліть на ній усі внутрішні та вихідні сигнали, що тестуються. На часовій діаграмі повинен залишитися лише графік зміни сигналу **reset**.

2.8.2 Оберіть команду **File** → **Export** → **Waveforms** головного меню. У діалозі **Save As**, оберіть формат експорту – **VHDL Process (*.VHS)**, надрукуйте ім'я файлу – **vectors** і збережіть файл, клацнувши **Save**. Завдяки тому, що закон зміни тестової послідовності був експортований з часової діаграми, випробувальний стенд можна згенерувати автоматично, за допомогою відповідного майстра САПР Active-HDL.

2.8.3 У вікні **Design Browser** виділіть пару **top (top)** верхнього рівня VHDL-проекту. Клацніть на ній правою кнопкою миші і у контекстному меню оберіть команду **Generate Testbench** (англ. «згенерувати випробувальний стенд»).

2.8.4 З'явиться перше вікно майстра **Testbench Generation Wizard** (рис. 2.27). У даному випадку тип випробувального стенда повинен бути *Single*

Process (англ. «окремий процес»). Клацнувши **Next**, перейдіть до наступного вікна майстра.

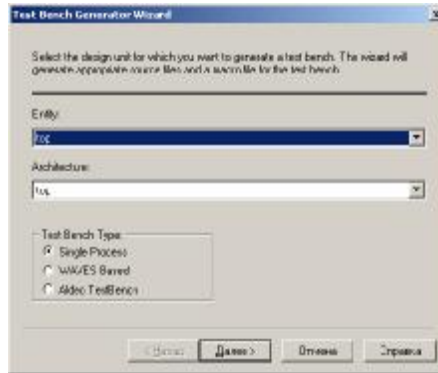


Рисунок 2.27 – Вибір типу випробувального стенда

- 2.8.5 У наступному вікні майстра треба задати сигнали тестових послідовностей випробувального стенда. Оскільки тестова послідовність береться з VHS-файлу, то включіть опцію *Test vectors from file* (рис. 2.28) і клацніть на кнопці **Browse**.
- 2.8.6 З'явиться стандартне діалогове вікно **Open**. У полі *Files type:* оберіть формат файлу, що відкривається, – *stimulus files (*.vhs)*. Знайдіть раніше створений файл **vectors.vhs** і клацніть на кнопці **Open**. У нижній частині вікна майстра розміщені два списки *Signals found in file* та *UUT ports*. Переконайтеся, що їх зміст однаковий. В обох списках повинен міститися один запис – сигнал **reset**. Для продовження роботи клацніть **Next**.

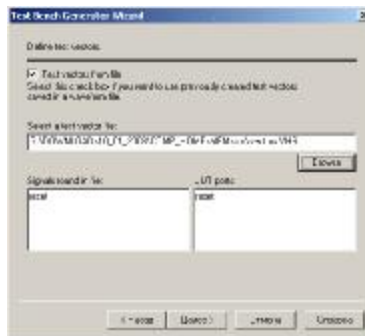


Рисунок 2.28 – Створення тестової послідовності

2.8.7 У наступному вікні майстра можна задати імена entity, architecture і VHDL-файлу випробувального стенда. Тут же задається назва директорії для його зберігання. Залиште настройки, що використовуються за замовчанням і, клацнувши **Next**, перейдіть до наступного вікна.

2.8.8 Останнє вікно майстра **Testbench Generation Wizard** містить інформацію про результати своєї роботи. До проекту будуть додані два файли, що створені автоматично:

- **top_TB.vhd** – файл випробувального стенду;
- **top_TB_runtest.do** – командний файл для запуску теста.

Тут можна вказати файл конфігурації для часового моделювання. Оскільки нашою задачею є не часове, а функціональне моделювання, то настройку *Generate* залиште невибраною (рис. 2.29). У кінці роботи з майстром **Testbench Generation Wizard** клацніть на кнопці **Finish**. Дерево проекту, що відображається на сторінці **Files** вікна **Design Browser**, з'явиться директорія **Testbench**, що містить файли випробувального стенда.

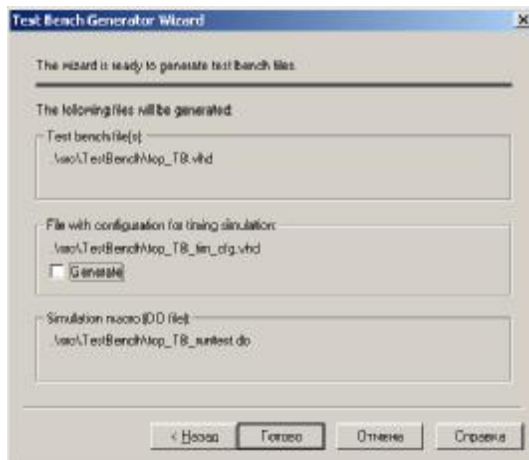


Рисунок 2.29 – Вікно Testbench Generation Wizard

2.8.9 Якщо у вікні **Design Browser** обрати файл **top_TB_runtest.do** і, клацнувши на ньому правою кнопкою миші, вибрати у контекстному меню команду **Execute**, то буде запущений процес трансляції і моделювання тестової послідовності випробувального стенда (рис. 2.30). Вікно часових діаграм містить закон зміни одного сигналу – **reset**.

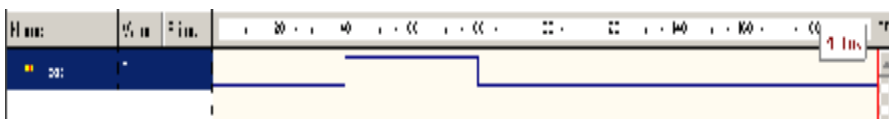


Рисунок 2.30 – Закон зміни тестової послідовності

2.8.10 У вікні **Design Browser** знайдіть файл **top_TB_runtest.do** і двічі клацніть на ньому. Текст командного файлу буде відкритий у вікні текстового редактора. Щоб доповнити часову діаграму результатів тестування сигналами, які перелічені у табл. 2.1, додайте до тексту файлу **top_TB_runtest.do** рядки, які на лістингу 2.5 виділені жирним шрифтом.

Лістинг 2.5 – Файл top_TB_runtest.do

```
#-----
#- командний файл запуску теста
#-----
SetActiveLib -work
comp -include "$DSN\src\icpu.vhd"
comp -include "$DSN\src\mem.vhd"
comp -include "$DSN\src\clk_gen.vhd"
comp -include "$DSN\src\top.vhd"
comp -include "$DSN\src\TestBench\top_TB.vhd"
asim TESTBENCH_FOR_top
wave
wave -noreg reset
wave -noreg UUT/clock
wave -noreg UUT/u1/x
wave -noreg UUT/rw
wave -noreg UUT/vma
wave -noreg UUT/ready
wave -noreg UUT/u1/icpu_data
wave -noreg UUT/data
wave -noreg UUT/m1/mem_data(0)
run 200.00 ns
```

2.8.11 Якщо повторити запуск командного файлу, буде отримана нова часова діаграма (рис. 2.31), яка відображає закон зміни не одного, а усіх дев'ятох сигналів, що тестуються. Переконайтеся, що результати автоматичного тестування співпадають з раніше отриманими (див. рис.2.24).

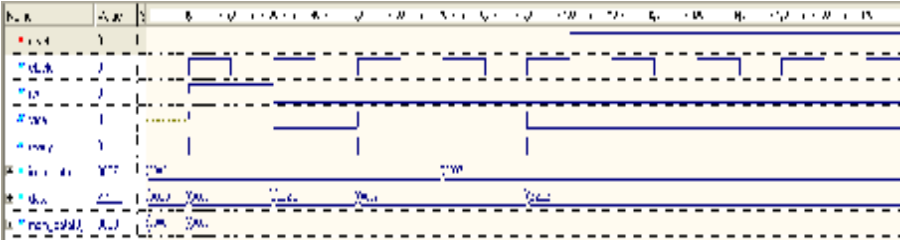


Рисунок 2.31 – Результат автоматичного тестування

Лістинг 2.6 – Файл top_TB.vhd

```

-----
-- випробувальний стенд
-----
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-----
ENTITY top_tb IS
END top_tb;
-----
ARCHITECTURE TB_ARCHITECTURE OF top_tb IS
    COMPONENT top
        PORT( reset : IN std_logic );
    END COMPONENT;
    SIGNAL reset : std_logic;
BEGIN
    UUT : top
    PORT MAP ( reset => reset );
    STIMULUS: PROCESS
    BEGIN
        reset <= '0';
        WAIT FOR 100 ns; --0 fs
        reset <= '1';
        WAIT FOR 100 ns; --100 ns
        reset <= '0';
        WAIT FOR 150 ns; --150 ns
        WAIT;
    END PROCESS STIMULUS;
END TB_ARCHITECTURE;
-----
CONFIGURATION TESTBENCH_FOR_top OF top_tb IS

```

```

FOR TB_ARCHITECTURE
  FOR UUT : top
    USE ENTITY work.top(top);
  END FOR;
END FOR;
END TESTBENCH_FOR_top;

```

2.8.12 Крім опису цифрових систем будь-якої складності, засоби мови VHDL можуть бути вжиті для аналізу поточного стану системи. Наприклад, лістинг 1.4 містить текст файлу *top.vhd*. Він виводить до вікна консолі САПР повідомлення про виконання циклів читання / запису зовнішньої пам'яті. Жирним шрифтом виділений текст циклу опитування стану сигналу **rw**. Якщо **rw = '0'**, до консолі виводиться повідомлення **READ RAM**. Якщо **rw = '1'**, буде згенероване повідомлення **WRITE RAM** (див. рис. 2.32).

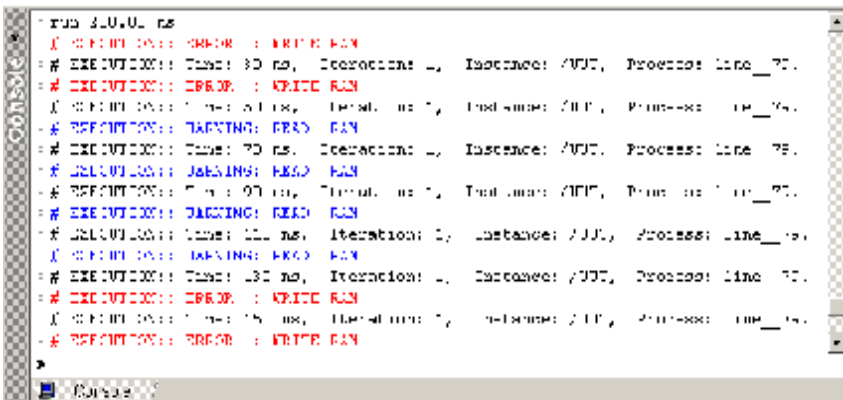


Рисунок 2.32 – Результати аналізу стану системи

2.9 Послідовність виконання роботи

1. Ознайомитись з теоретичним матеріалом.
 2. Спроекувати та виконати компіляцію файлів опису блоків системи.
 3. Створити файл тестової послідовності та провести функціональне моделювання у ручному режимі. Порядок зміни сигналів спостерігати за часовою діаграмою, списком перемикачів і на логічній схемі.
 4. Створити файл випробувального стенду і виконати автоматичне тестування. Порядок зміни сигналів спостерігати за часовою діаграмою.
 5. Скласти звіт про виконання лабораторної роботи та захистити його.
 6. Дати відповіді на контрольні запитання.
- Варіанти індивідуальних завдань наведені в таблиці 2.2.

Таблиця 2.2 – Індивідуальні завдання

Вар.	Формула	Час, нс	Вар.	Формула	Час, нс
0	0 0 ns, 1 220 ns, 0 100 ns	500	13	0 0 ns, 1 250 ns, 0 180 ns	500
1	0 0 ns, 1 340 ns, 0 130 ns	600	14	0 0 ns, 1 270 ns, 0 190 ns	600
2	0 0 ns, 1 60 ns, 0 90 ns	200	15	0 0 ns, 1 40 ns, 0 100 ns	200
3	0 0 ns, 1 110 ns, 0 80 ns	300	16	0 0 ns, 1 70 ns, 0 150 ns	300
4	0 0 ns, 1 180 ns, 0 170 ns	400	17	0 0 ns, 1 120 ns, 0 210 ns	400
5	0 0 ns, 1 240 ns, 0 130 ns	500	18	0 0 ns, 1 200 ns, 0 180 ns	500
6	0 0 ns, 1 280 ns, 0 200 ns	600	19	0 0 ns, 1 220 ns, 0 220 ns	600
7	0 0 ns, 1 50 ns, 0 70 ns	200	20	0 0 ns, 1 40 ns, 0 90 ns	200
8	0 0 ns, 1 170 ns, 0 130 ns	400	21	0 0 ns, 1 150 ns, 0 200 ns	400
9	0 0 ns, 1 220 ns, 0 140 ns	600	22	0 0 ns, 1 230 ns, 0 300 ns	600
10	0 0 ns, 1 100 ns, 0 100 ns	300	23	0 0 ns, 1 60 ns, 0 150 ns	300
11	0 0 ns, 1 130 ns, 0 200 ns	500	24	0 0 ns, 1 100 ns, 0 250 ns	500
12	0 0 ns, 1 120 ns, 0 180 ns	400	25	0 0 ns, 1 80 ns, 0 200 ns	400

2.10 Зміст звіту

1. Мета роботи.
2. Номер варіанту індивідуального завдання.
3. Результати моделювання у вигляді часової діаграми, таблиці перемикачів, логічної схеми.
4. Тексти програм мовою VHDL з коментарями.
5. Висновки.

Звіт до лабораторної роботи повинен містити результати, отримані під час її виконання і тези відповідей на контрольні питання. Звіти в електронному вигляді або ксерокопії до захисту не приймаються.

2.11 Контрольні питання

1. Призначення бібліотек. Що міститься у бібліотеці STD_LOGIC_unsigned? Яким чином підключаються бібліотеки до програми мовою VHDL?
2. Яка різниця між висхідною та низхідною стратегіями проектування?
3. Поясніть часову діаграму роботи пам'яті.
4. Недоліки та переваги різних технологій симуляції роботи системи.
5. Як можна програмувати вміст блоку постійної пам'яті?
6. Як виконується з'єднання компонентів системи на логічній схемі?
7. Поясніть принцип роботи системи, що проектується.
8. З яких файлів складається VHDL- проект ЦС? Для чого потрібно створювати робоче середовище (workspace)? Яку функцію у VHDL- проекті виконує файл верхнього рівня ієрархії? Для чого в описі ЦС використовують модуль entity? Модуль architecture?
9. Побудуйте граф-схему алгоритму програми ісру.vhd (лістинг 1.3).
10. Які засоби редагування має редактор HDL? Поясніть техніку їх використання.
11. Які засоби редагування має редактор BDE? Поясніть техніку їх використання.
12. Які сигнали називаються глобальними? Як задається глобальний сигнал в редакторі BDE?
13. Як використовуються блоки в редакторі BDE? Які є вбудовані блоки в САПР Active-HDL? Наведіть приклади.
14. Які функції виконує майстер Language Assistant? Поясніть техніку його використання.
15. Що таке випробувальний стенд? Тестова послідовність? Порівняйте два способи моделювання ЦС: за участю користувача та автоматичний.

16. Яким чином у САПР Active-HDL можна створити випробувальний стенд для одного процесу?
17. В якій послідовності виконується ініціалізація моделювання ЦС? Як виконати повторний запуск моделювання?
18. Які засоби керування моделюванням і трасуванням VHDL- проекту є в САПР Active-HDL?
19. Які засоби відображення результатів моделювання і трасування можуть використовуватися?
20. Які типи стимуляторів є в САПР Active-HDL? Як вони настроюються?

ЛІТЕРАТУРА

1. 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. Основы языка VHDL.– М.: Солон–Р, 2000.–200с.
3. Бибило П.Н. Синтез логических схем с использованием языка VHDL.– М.: СОЛОН–Р, 2002.– 384 с.
4. Кондратенко Ю.П., Сидоренко С.А., Підпригора Д.М. Поведінковий синтез цифрових пристроїв у середовищі Active–HDL: Навчальний посібник.– Миколаїв: МФ НаУКМА, 2002.– 116 с.
5. Шапо Ф.С., Шапо В.Ф. Введение в VHDL – язык проектирования цифровых систем.– Одесса: Астропринт, 2001.– 220 с.
6. Дж. Ф. Уэйкерли Проектирование цифровых устройств. В 2–х томах: Пер. с англ. М.: Постмаркет, 2002.
7. Угрюмов Е.П. Цифровая схемотехника.–СПб.: БХВ–Петербург, 2001.– 528с.(Гл.9)
8. Методичні вказівки до лабораторних робіт з дисципліни “Основи автоматизованого проектування засобів обчислювальної техніки” для студентів спеціальностей 8.091 501 ”Комп’ютерні системи та мережі” та 8.091 503 ”Спеціалізовані комп’ютерні системи” всіх форм навчання. Частина 2. /Укл. А.К. Тімовський, О.О.Голдобін. - Запоріжжя: ЗНТУ, 2004. – 50 с.
9. Перельройзен Е.З. Проектируем на VHDL – М.: СОЛОН-Пресс, 2004.- 448с.
10. А.К. Тімовський, , О.О.Голдобін Методичні вказівки до лабораторних робіт з дисципліни “Автоматизація проектування комп’ютерних систем”.– ЗДТУ, 2009