Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по лабораторной работе № 7**

Дисциплина: Языки описания аппаратных средств вычислительных систем.

Выполнил студент гр. 5130901/10101 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Т. Непомнящий

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Федотов

(подпись)

Санкт-Петербург

2023

**Оглавление**

[1. Lab1\_1: Basic Simulation Flow 3](#_Toc150106610)

[1.1. Создание библиотек и открытие файлов 3](#_Toc150106611)

[1.2. Компиляция проекта 4](#_Toc150106612)

[1.3. Симуляция проекта 5](#_Toc150106613)

[1.4. Создание точки останова и пошаговый запуск 6](#_Toc150106614)

[1.5. Добавление файлов в Waveform 8](#_Toc150106615)

[1.6. Масштабирование временных диаграмм 9](#_Toc150106616)

[1.7. Управление курсором 12](#_Toc150106617)

[1.8. Вывод 12](#_Toc150106618)

[2. Lab1\_2: Projects Flow 13](#_Toc150106619)

[2.1. Создание нового проекта 14](#_Toc150106620)

[2.2. Компиляция проекта 14](#_Toc150106621)

[2.3. Добавление папок в проект 15](#_Toc150106622)

[2.4. Переместить файлы в проект 16](#_Toc150106623)

[2.5. Simulate Configuration 17](#_Toc150106624)

[2.6. Вывод 17](#_Toc150106625)

[3. Lab1\_3: Working with Multiple Libraries 18](#_Toc150106626)

[3.1. Создание ресурсной библиотеки 19](#_Toc150106627)

[3.2. Создание проекта 20](#_Toc150106628)

[3.3. Загрузка без привязки библиотек 21](#_Toc150106629)

[3.4. Привязка к библиотеке ресурсов 22](#_Toc150106630)

[3.5. Вывод 22](#_Toc150106631)

[4. Lab1\_4: Automating Simulation 23](#_Toc150106632)

[4.1. Создание простого DO файла 23](#_Toc150106633)

[4.2. Запуск в режиме командной строки 24](#_Toc150106634)

[4.3. Запуск сценария Tcl script 26](#_Toc150106635)

[4.4. Вывод 29](#_Toc150106636)

[5. Lab1\_5: Viewing and Initializing Memories 30](#_Toc150106637)

[5.1. Компиляция проекта и загрузка дизайна 30](#_Toc150106638)

[5.2. Просмотр памяти и её содержание 30](#_Toc150106639)

[5.3. Навигация по памяти 33](#_Toc150106640)

[5.4. Экспорт поля памяти в файл 34](#_Toc150106641)

[5.5. Инициализация памяти 35](#_Toc150106642)

[5.6. Команды интерактивной отладки 36](#_Toc150106643)

[5.7. Вывод 37](#_Toc150106644)

# Lab1\_1: Basic Simulation Flow

В данной лабораторной работе используются Verilog файлы *counter.v* и *tcounter.v*. Они находятся в рабочей папке (**D:\Intel\_trn\ModelSim\Lab1\_1**) для Lab1\_1.

## Создание библиотек и открытие файлов

Перед тем, как смоделировать проект, нужно сначала создать библиотеку и скомпилировать исходный код в эту библиотеку

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рис.1.1 – Создание новой библиотеки

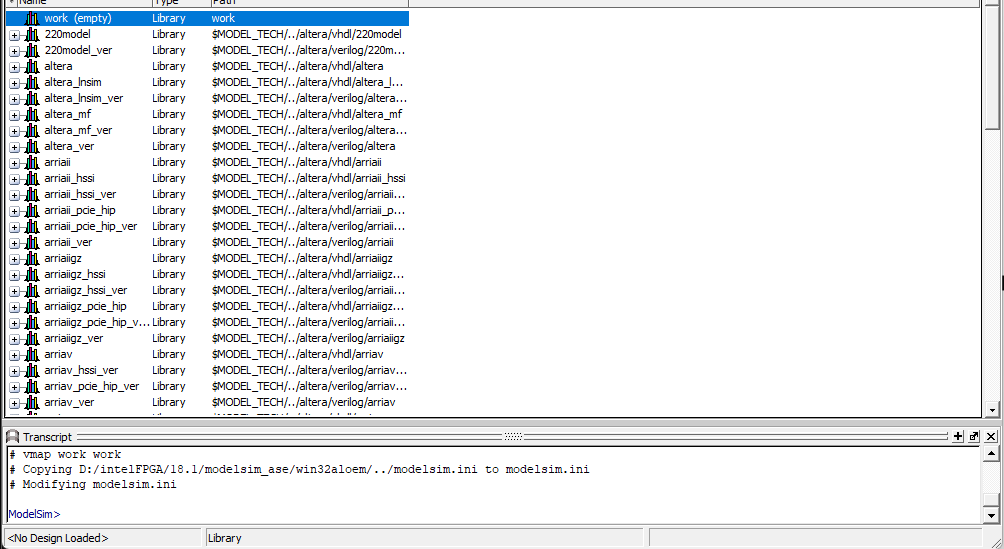


Рис.1.3 – Создана новая библиотека в окне Library window

## Компиляция проекта

После этого была проведена компиляция проектов *counter.v* и *tcounter.v* (рис. 1.4 – 1.5):

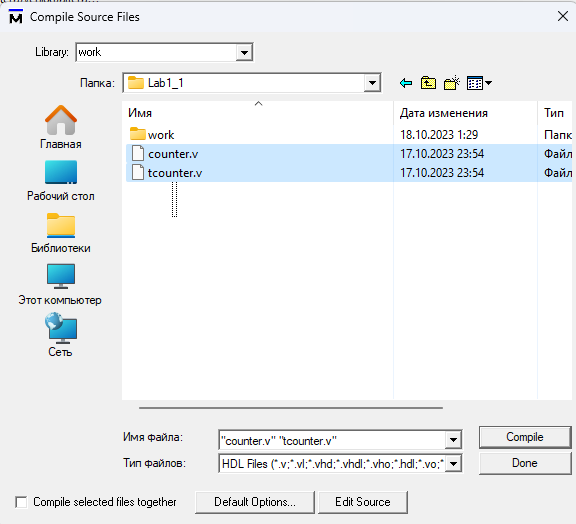


Рис.1.4 – Выбор проектов для компиляции

Изображение выглядит как текст, Шрифт, программное обеспечение, снимок экрана

Автоматически созданное описание

Рис. 1.5 – Компиляция проекта

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Рис. 1.6. – Structure window

## Симуляция проекта

Запустим симуляцию проекта. Изначально длина этой симуляции составляет 100ns (рис. 1.7):

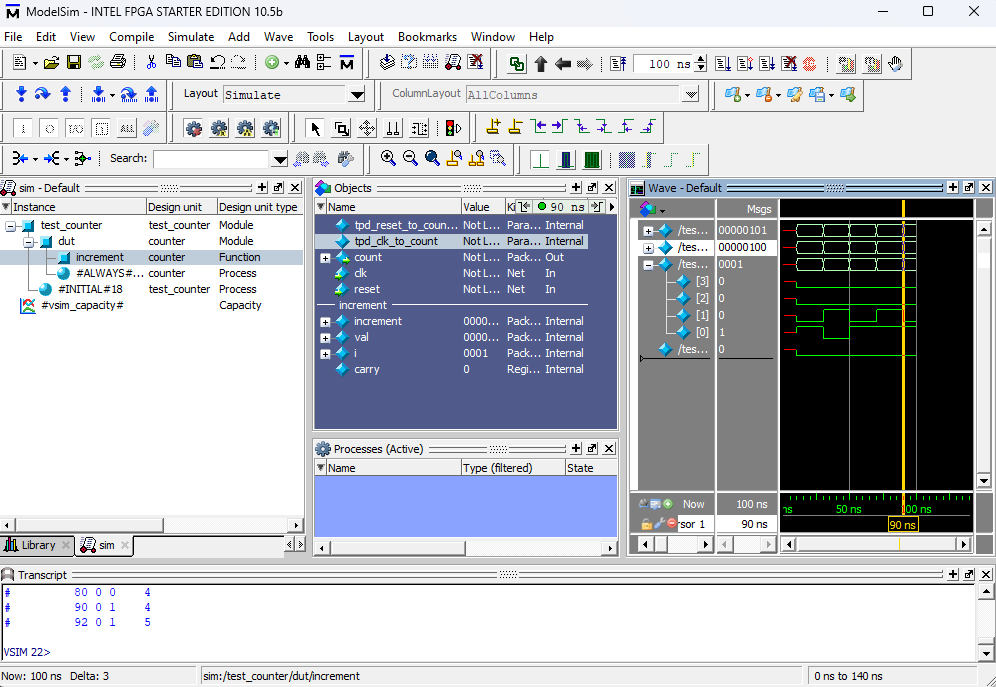


Рис. 1.7 – Симуляция проекта (100 ns)

Введём в окно команд **run 500**. Это создаст дополнительные 500 ns на временной диаграмме.

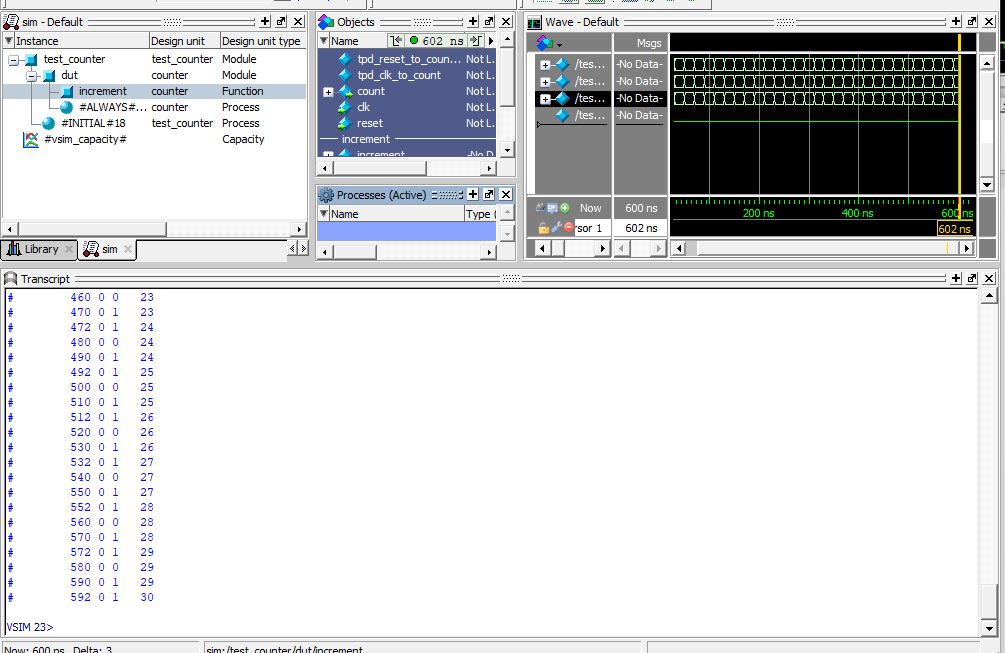


Рис. 1.8 – Симуляция проекта (100 ns)

## Создание точки останова и пошаговый запуск

Установим точку останова (break point) в файле counter.v на 36 строке:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рис. 1.9 – Создание точки останова

Запустим компиляцию при наличии точки останова:



Рис. 1.10 – Запуск файла при наличии точка останова

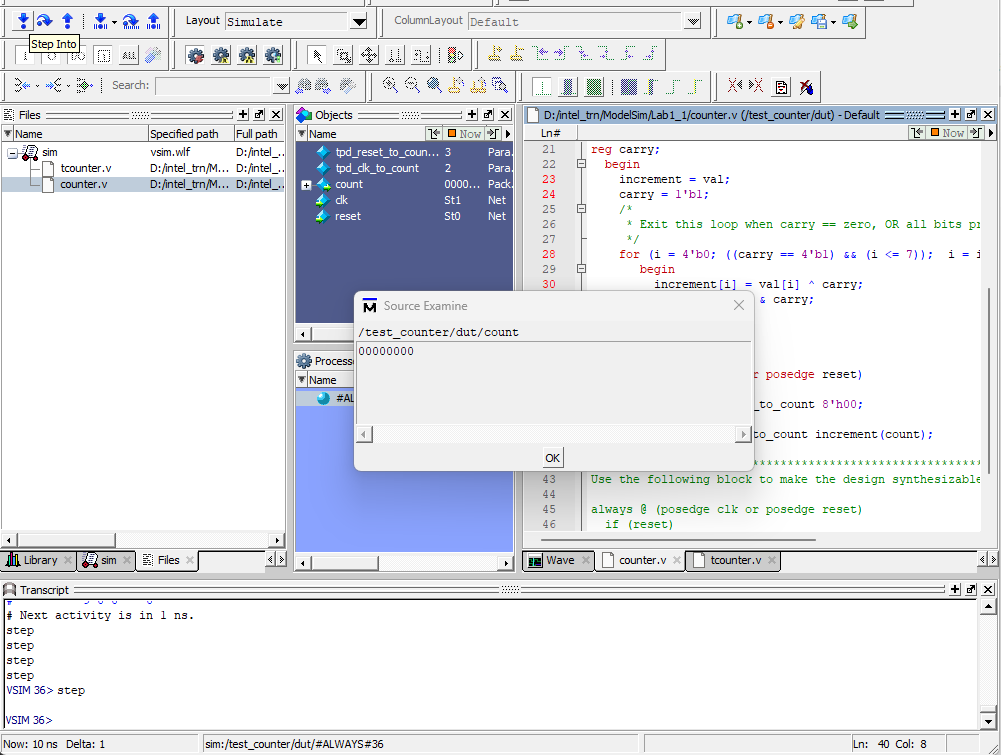


Рис. 1.11 – Пошаговый запуск

Проведём пошаговый запуск несколько раз:

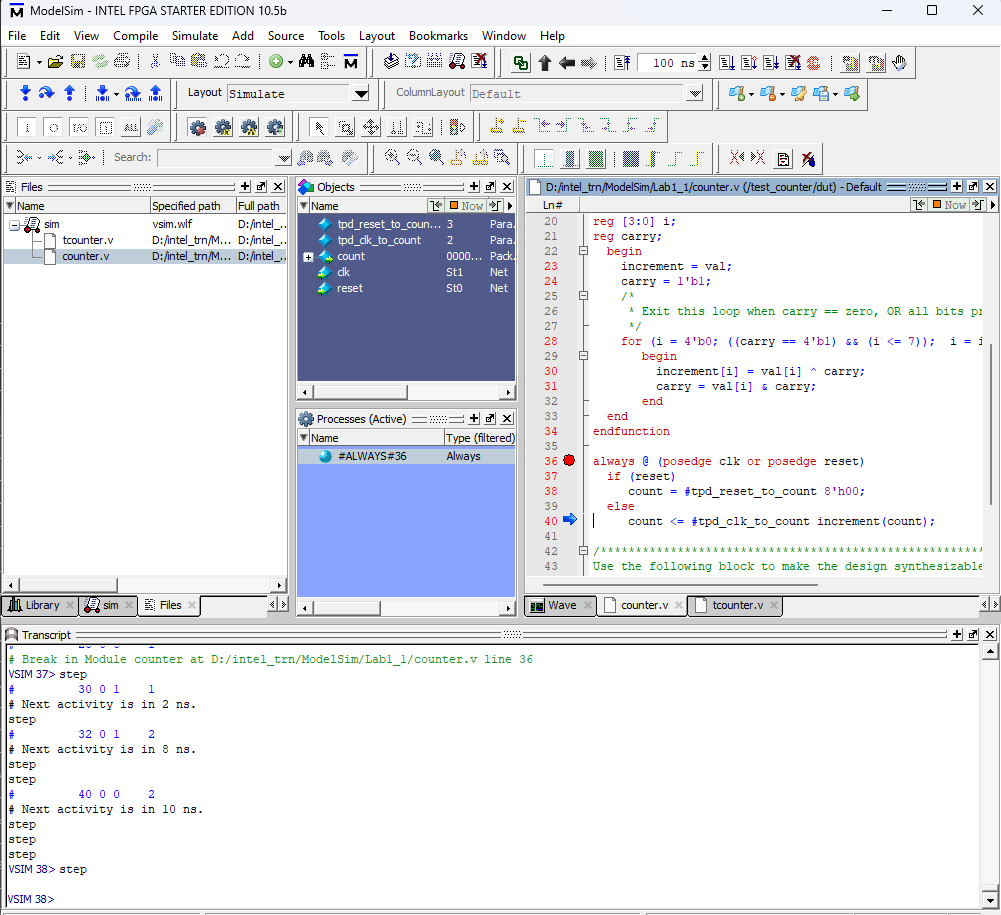


Рис. 1.12 – Выводы пошаговых команд в консоль

## Добавление файлов в Waveform

Воспользуемся различными способами добавления файлов в Waveform (рис. 1.12 – 1.13)

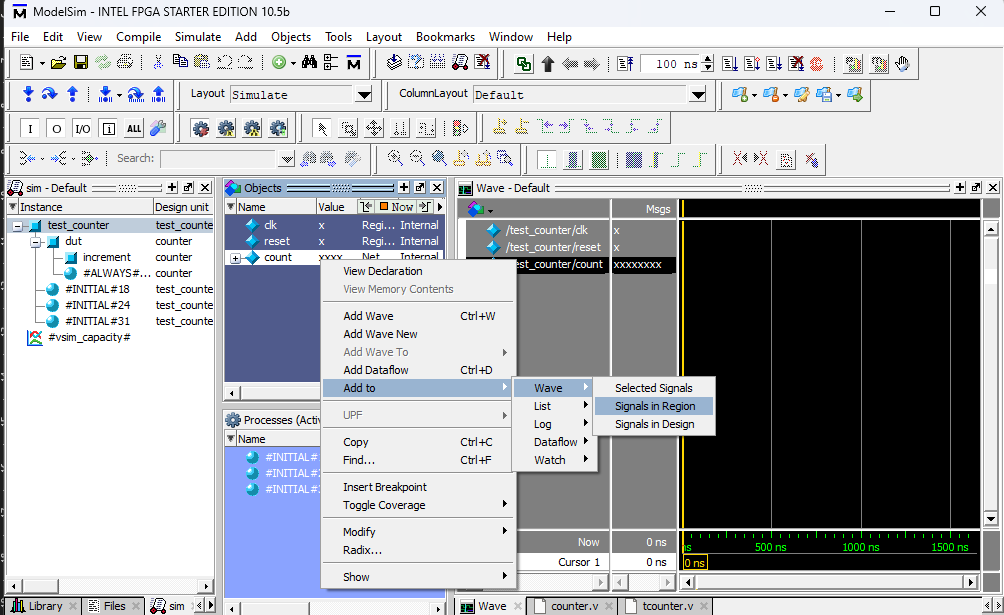


Рис. 1.13 – Добавление файлов с использованием окна Objects

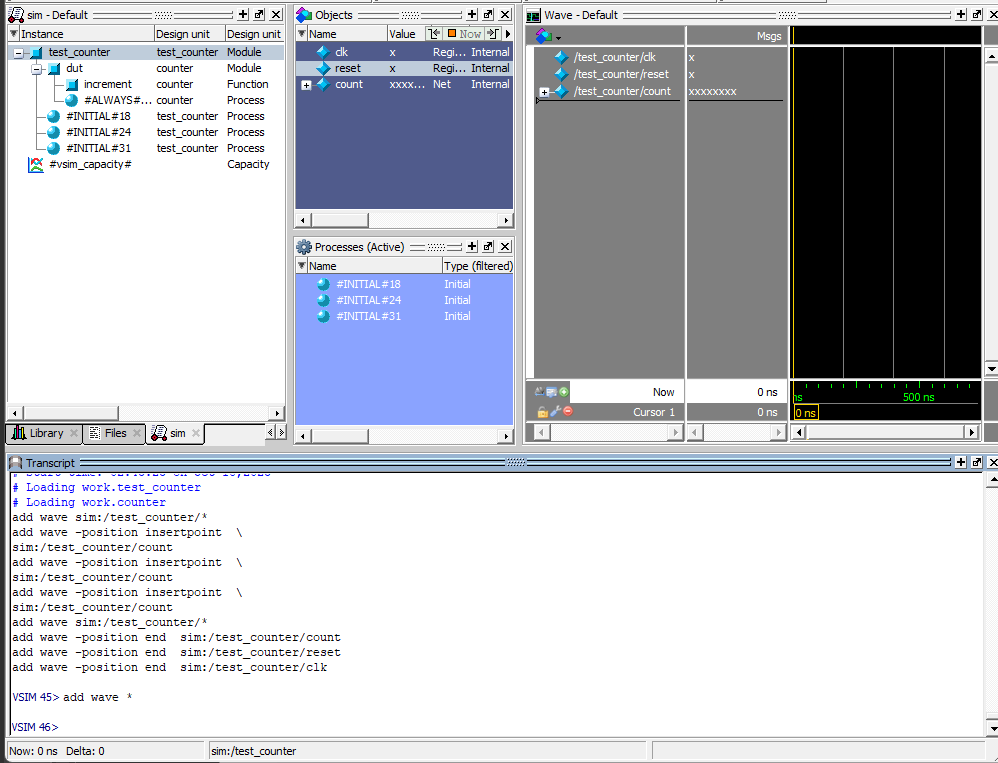


Рис. 1.14 – Добавление файлов при помощи командной строки

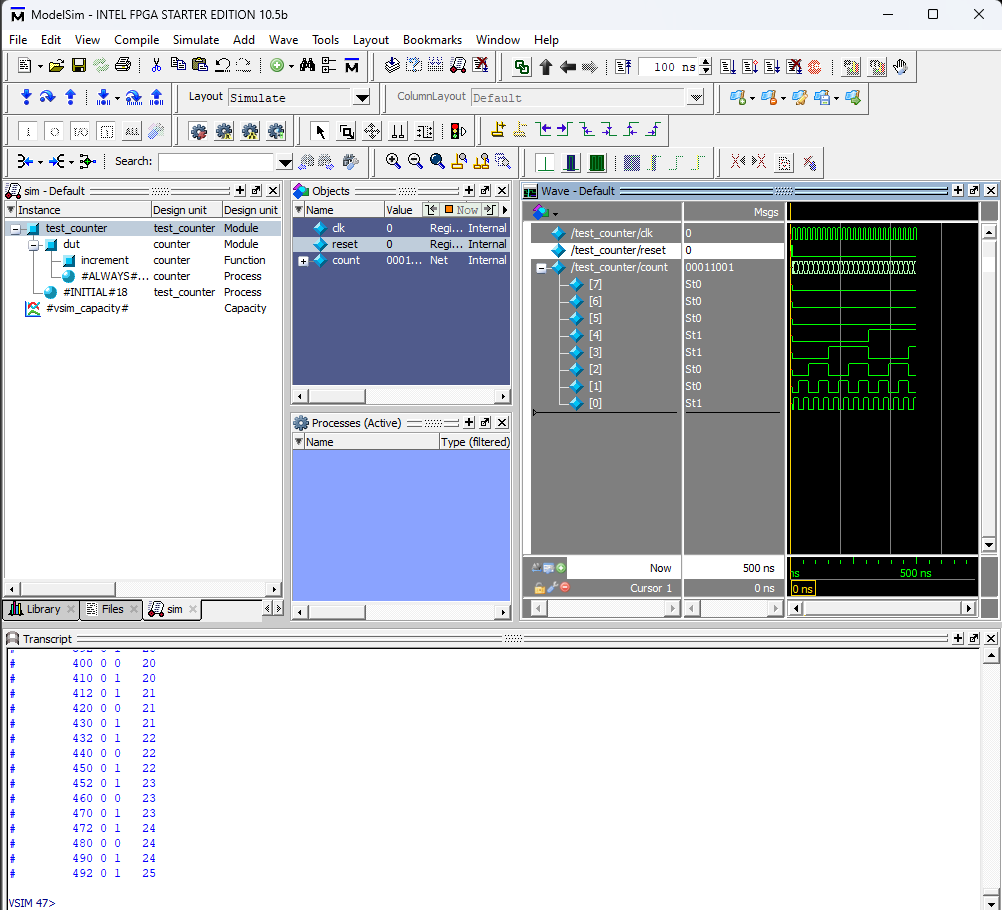
е

Рис. 1.15 – Запуск проекта

## Масштабирование временных диаграмм

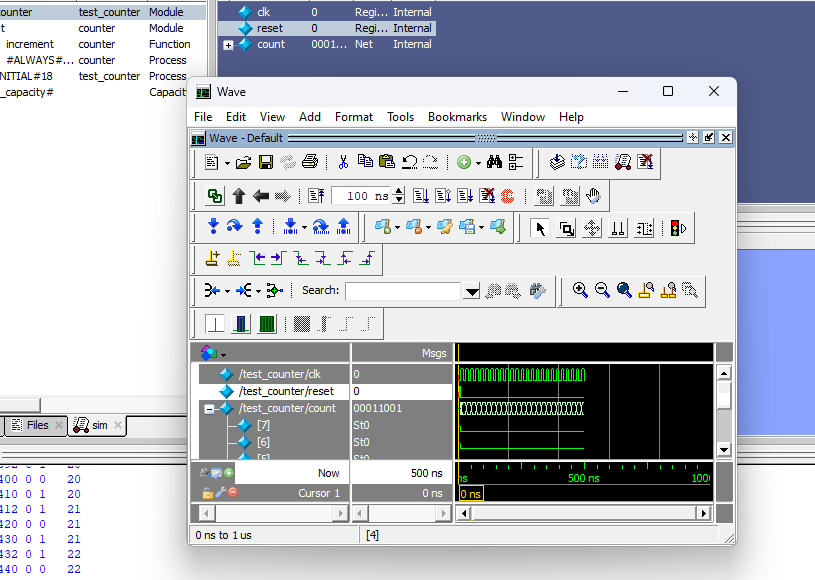


Рис. 1.16 – Запуск в отдельном окне

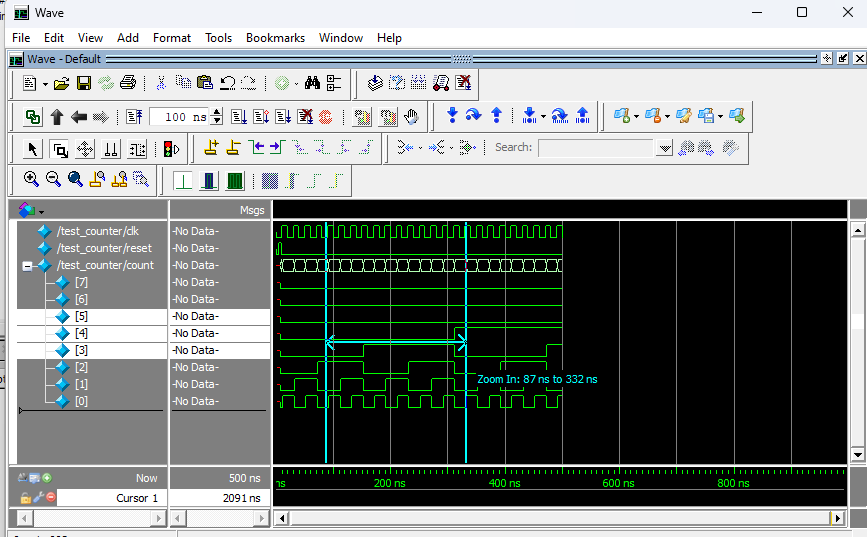


Рис. 1.17 – Масштабирование с помощью Zoom Mode

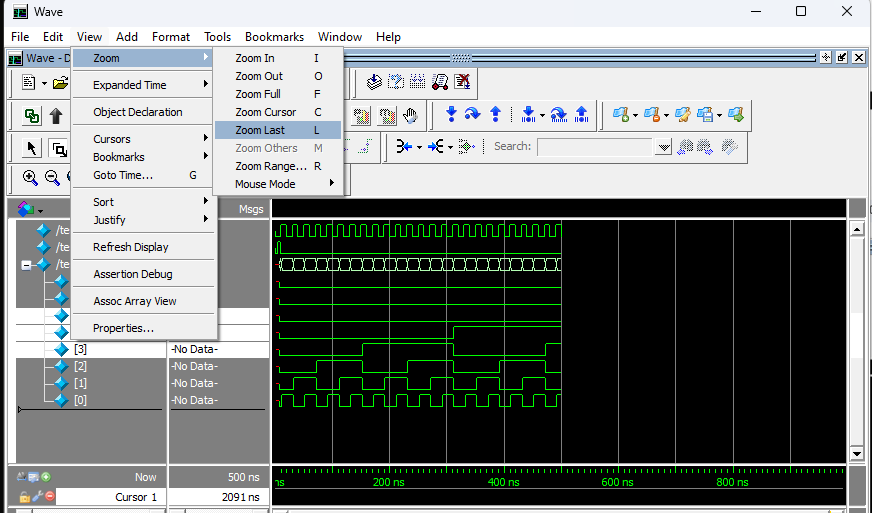


Рис. 1.17 – Масштабирование при помощи Zoom Last

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 1.17 – Масштабирование при помощи инструмента Zoom In

Изменение формата чисел

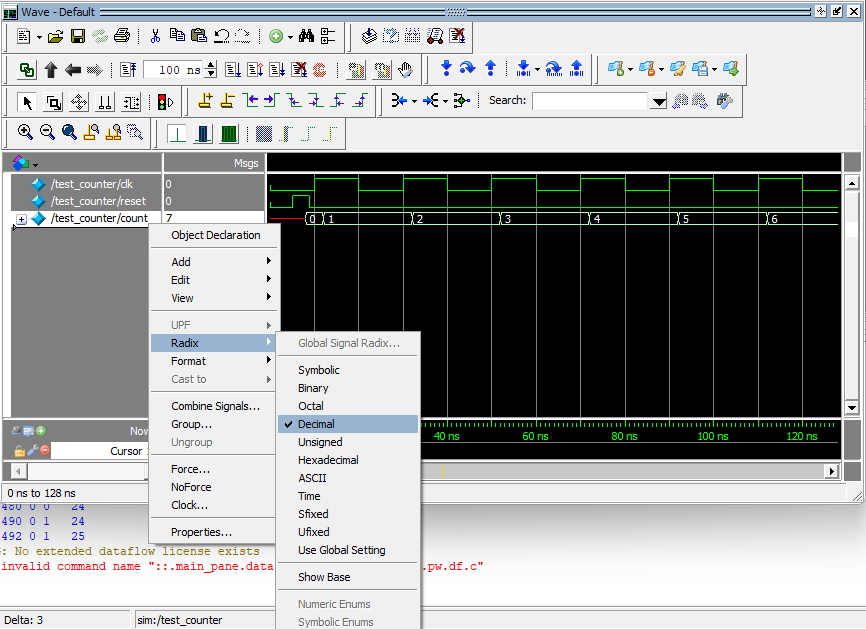


Рис. 1.18 – Изменение отображаемой системы счисления

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рис. 1.19 – Изменение представления формата чисел

Изображение выглядит как программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение, снимок экрана

Автоматически созданное описание

Рис. 1.20 – Изменение формата обратно на Literal

## Управление курсором

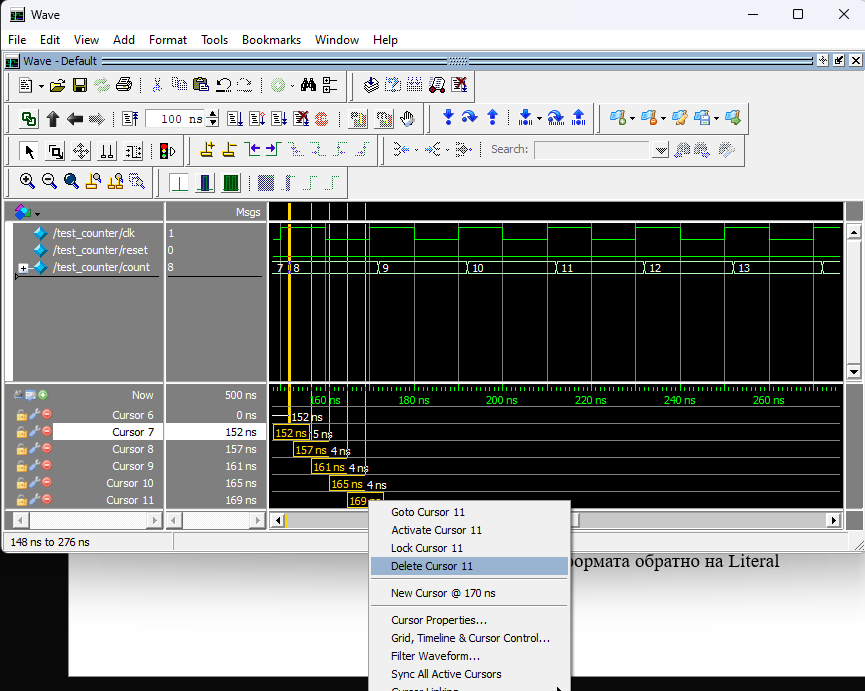


Рис. 1.21 – Insert/Output Cursor

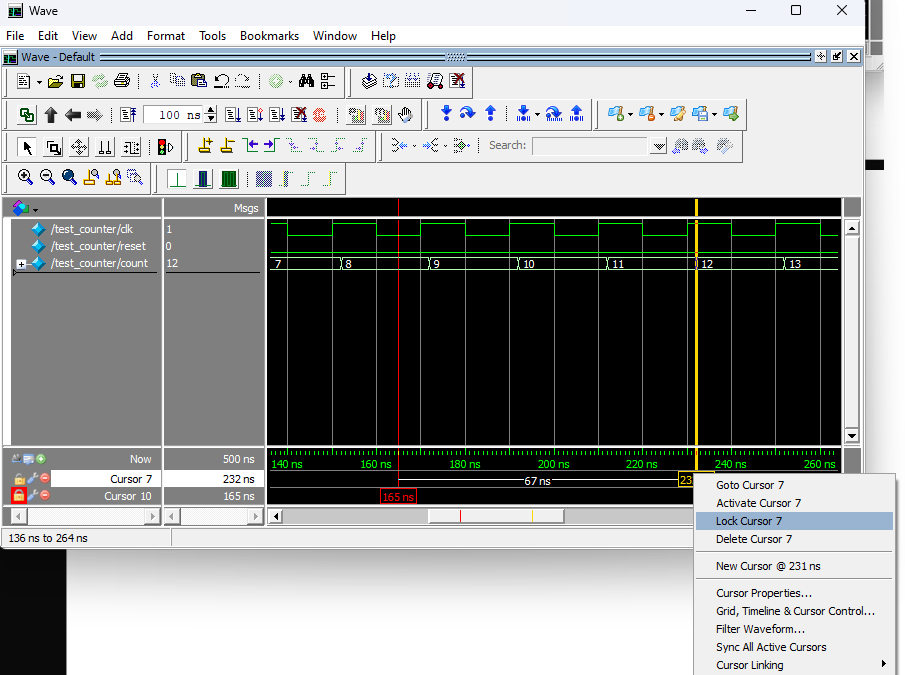


Рис. 1.22 – Block Cursor

## Вывод

В данной лабораторной работе были рассмотрены базовые примеры работы ModelSim, а также применение их на практике для исходной библиотеки

# Lab1\_2: Projects Flow

В данной лабораторной работе будет изучено на практике то, как создавать проект.

Как минимум, проекты содержат рабочую библиотеку и состояние сеанса, хранящееся в файле .mpf. Также, в проекте могут содержаться:

* исходные файлы HDL или ссылки на исходные файлы
* другие файлы, такие как README или другая проектная документация.
* местные библиотеки
* ссылки на глобальные библиотеки

Проект — это механизм сбора HDL-проектов, находящихся в стадии разработки или тестирования. Несмотря на то, что вам не обязательно использовать проекты в ModelSim, проекты могут упростить взаимодействие с инструментом и полезны для организации файлов и указания настроек моделирования.

На диаграмме ниже представлены основные шаги для моделирования конструкции в проекте ModelSim.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис. 2.1 - Основные шаги моделирования конструкции в проекте ModelSim.

Процесс аналогичен базовому процессу моделирования, однако есть два важных различия:

• Не нужно создавать рабочую библиотеку в ходе проекта, это делается автоматически.

• Проекты постоянны. Другими словами, они будут открываться каждый раз, когда вы вызываете ModelSim, если вы специально не закроете их.

## Создание нового проекта

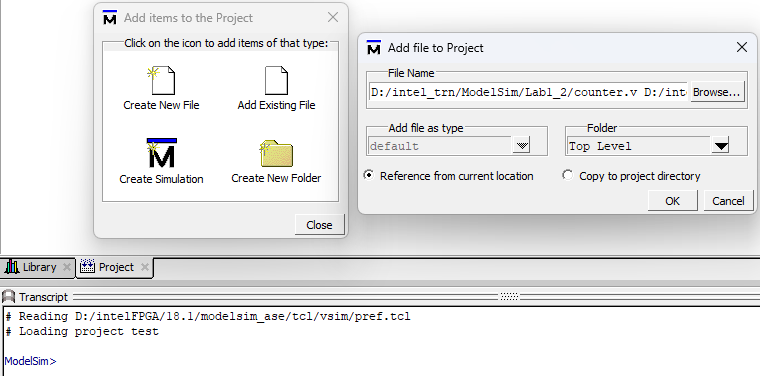


Рис. 2.2 – Добавление файлов в проект

Значки вопросительного знака в столбце «Состояние» указывают на то, что файл не был скомпилирован или что исходный файл изменился с момента последней успешной компиляции. Другие столбцы идентифицируют тип файла (например, Verilog или VHDL), порядок компиляции и дату изменения.

## Компиляция проекта

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рис. 2.3 – Подготовка к компиляции обоих файлов

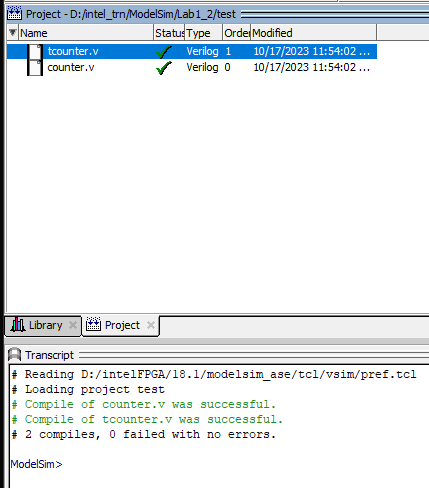


Рис. 2.4 – Конец компиляции

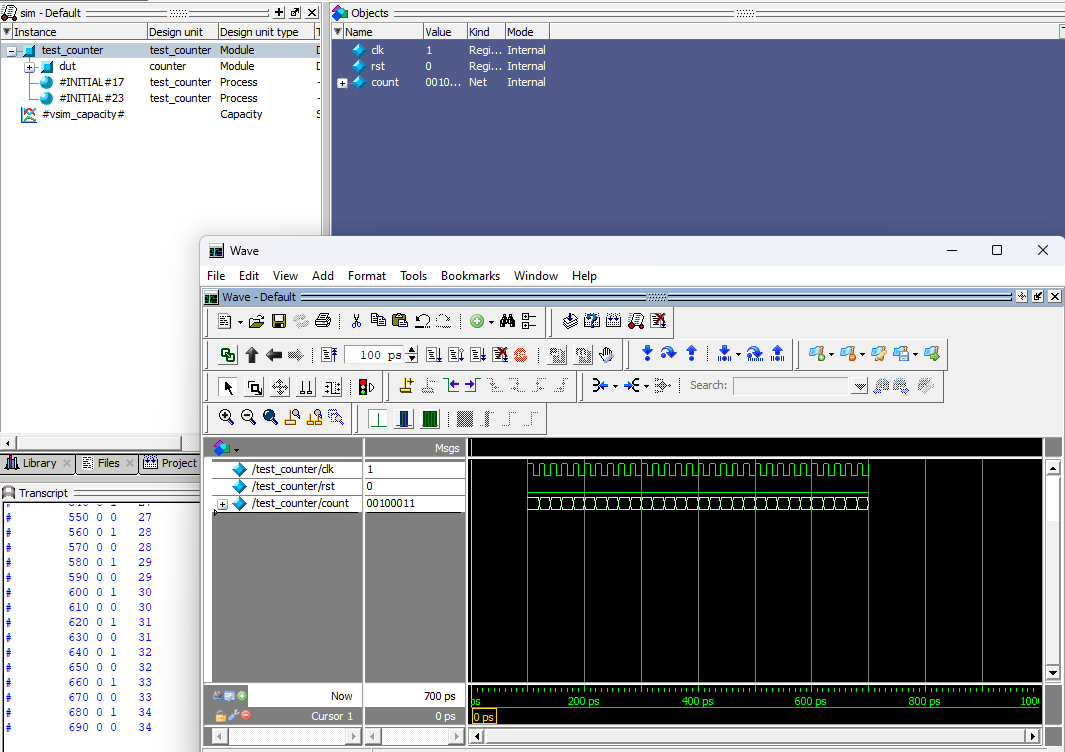


Рис. 2.5 – Компиляция проекта

## Добавление папок в проект

Добавим папку в проект, используя Pop\_up menu => Add to Project => Folder…

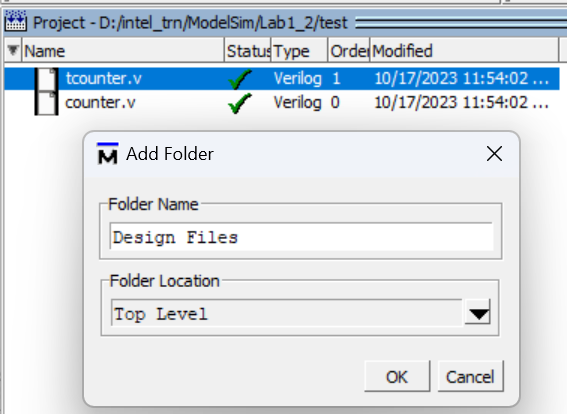


Рис. 2.6 – Создание папки

Теперь добавим новую папку HDL в только что созданную папку Design Files (рис. 2.6 -2.7):

Изображение выглядит как текст, программное обеспечение, Шрифт, число

Автоматически созданное описание

Рис. 2.7 – Подготовка к созданию вложенной папки

Изображение выглядит как текст, программное обеспечение, число, Значок на компьютере

Автоматически созданное описание

Рис. 2.8 – Создание папки HDL

## Переместить файлы в проект

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Рис. 2.9 – Перемещение файлов в папку HDL

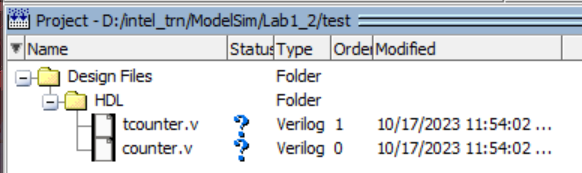


Рис. 2.10 – Новое расположение файлов

Конфигурация моделирования связывает проектные единицы и их параметры моделирования. Например, предположим, что каждый раз, когда вы загружаете tcounter.v, вы хотите установить разрешение симулятора в пикосекундах (пс) и включить проверку опасностей порядка событий. Обычно вам придется указывать эти параметры каждый раз, когда вы загружаете проект. С помощью конфигурации моделирования вы указываете параметры проекта, а затем сохраняете «конфигурацию», которая связывает проект и его параметры. Затем конфигурация отображается в окне «Проект», и вы можете дважды щелкнуть ее, чтобы загрузить tcounter.v вместе с его параметрами.

## Simulate Configuration

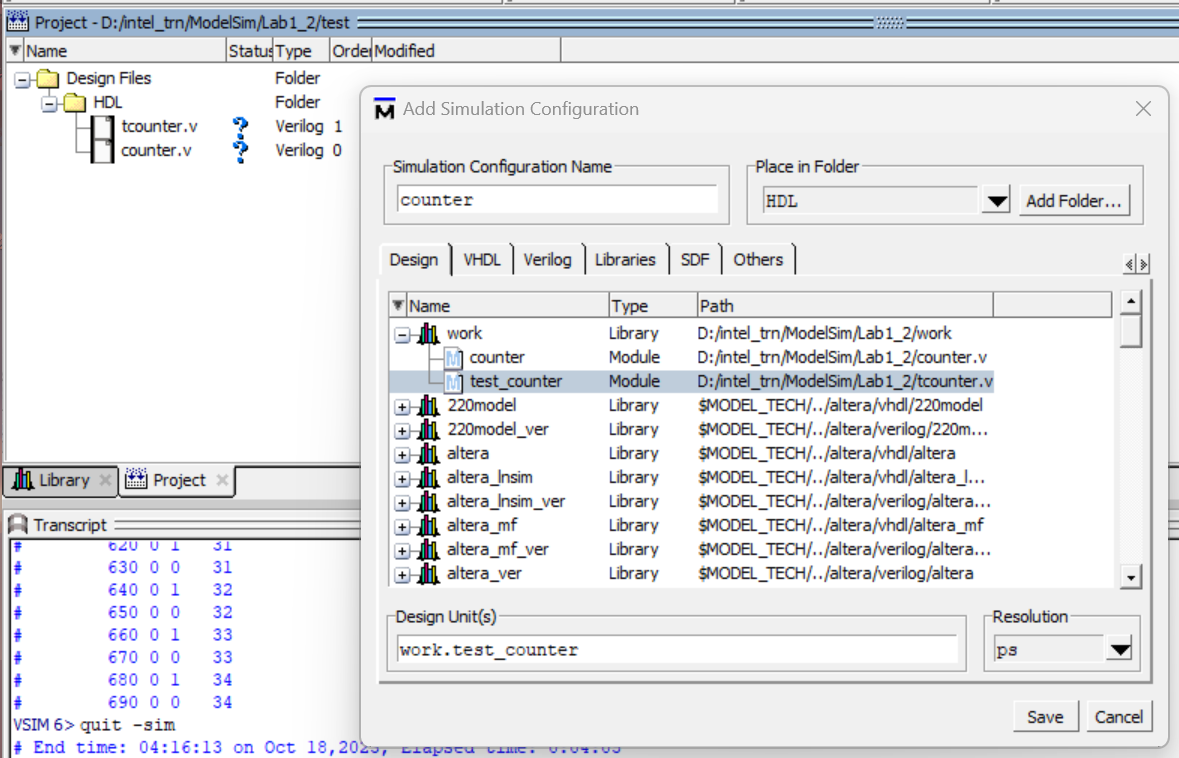


Рис. 2.11 – Изменение конфигурации

Файлы tcounter.v и counter.v отображают значки вопросительных знаков в столбце состояния, поскольку их местоположение изменилось с момента последней компиляции, и их необходимо перекомпилировать.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рис. 2.12 – Компиляция всех файлов

В окне «Transcript» главного окна вызов vsim (симулятора ModelSim) отображает переключатели -hazards и -t ps. Это эквиваленты командной строки параметров, указанных в диалоговом окне «Simulate».

## Вывод

В данной лабораторной работе была изучена работа с файлами, перемещение их внутрь других папок, а также работа с конфигурациями.

# Lab1\_3: Working with Multiple Libraries

В этой работе будет рассмотрена работа с несколькими библиотеками. Есть несколько для организации проекта, доступа к IP из стороннего источника или совместного использования общих частей между симуляциями.

ModelSim использует библиотеки двумя способами:

* в качестве локальной рабочей библиотеки, содержащей скомпилированную версию проекта;
* как библиотека ресурсов.

Содержимое рабочей библиотеки будет меняться по мере обновления проекта и перекомпиляции. Библиотека ресурсов обычно является статической и служит источником деталей для вашего проекта. Можно создавать свои собственные библиотеки ресурсов или они могут быть предоставлены другой командой разработчиков или третьей стороной (например, поставщиком микросхем).

Мы сами указываем, какие библиотеки ресурсов будут использоваться при компиляции проекта, и существуют правила, определяющие порядок их поиска. Типичным примером использования как рабочей библиотеки, так и библиотеки ресурсов является случай, когда проект уровня шлюза и испытательный стенд компилируются в рабочую библиотеку, а проект ссылается на модели уровня шлюза в отдельной библиотеке ресурсов.

На диаграмме ниже показаны основные шаги моделирования с использованием нескольких библиотек.

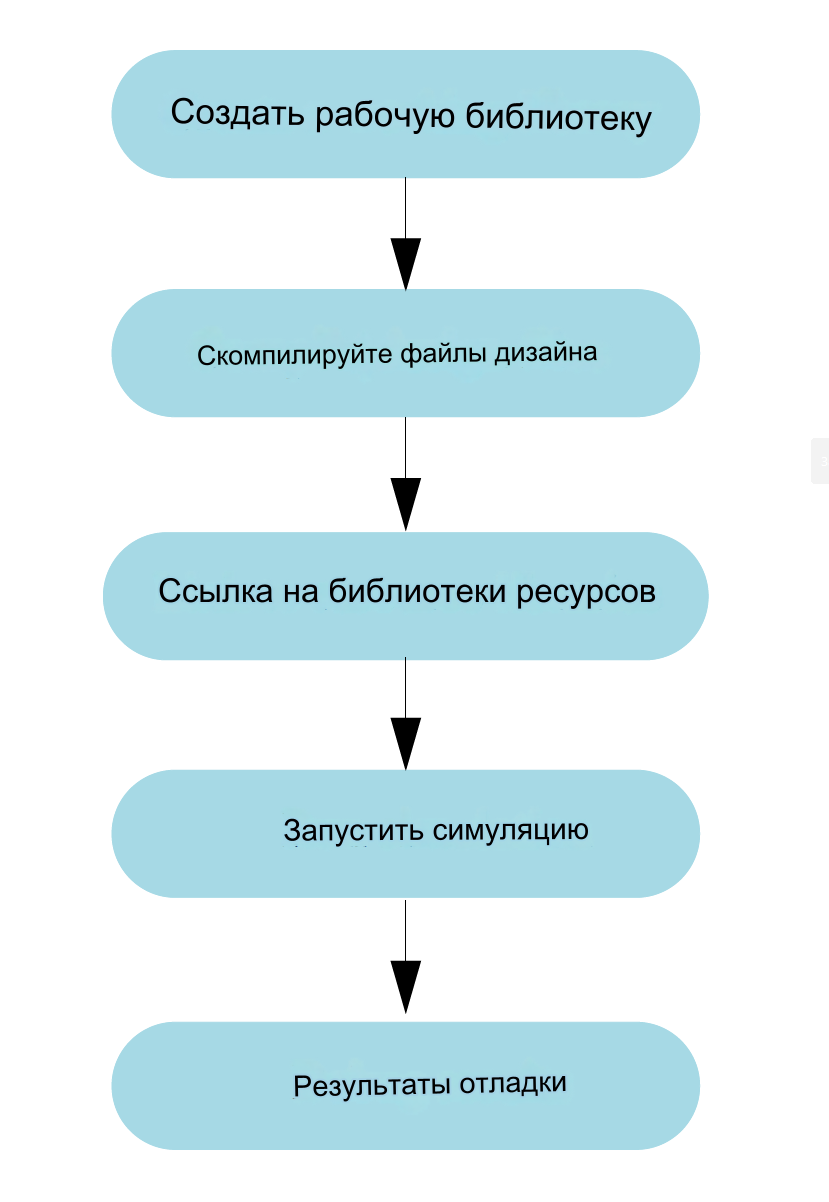


Рис. 3.1 – Схема моделирования нескольких библиотек

Проверим, что файл modelsim.ini доступен только для чтения. Это предотвратит постоянное сопоставление библиотек ресурсов с главным файлом modelsim.ini:

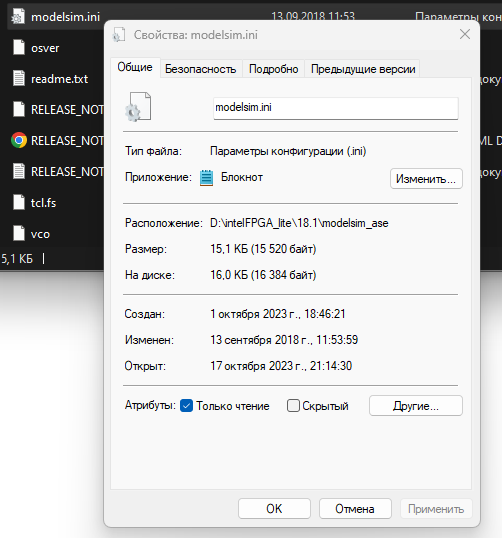


Рис. 3.2 – Проверка, что файл modelsim.ini доступен только для чтения

## Создание ресурсной библиотеки

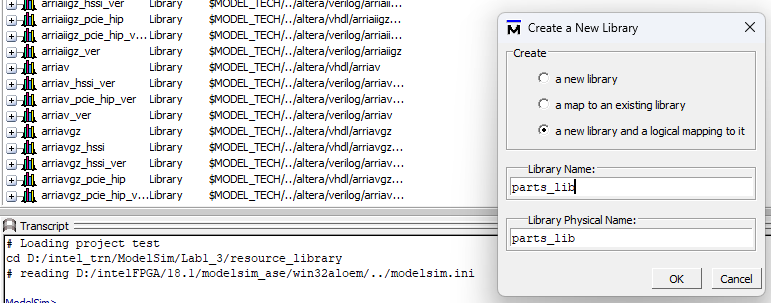


Рис. 3.3 – Изменение директории

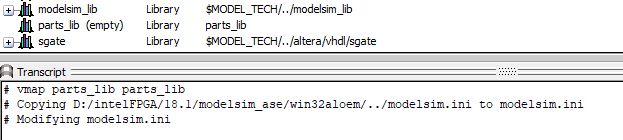


Рис. 3.4 –

ModelSim создал каталог для библиотеки, отобразил его в окне «Библиотека» и изменил локальный файл modelsim.ini для записи этой новой библиотеки на будущее.

Создана библиотека ресурсов, содержащая скомпилированную версию модуля проектирования счетчика (рис. 3.5)

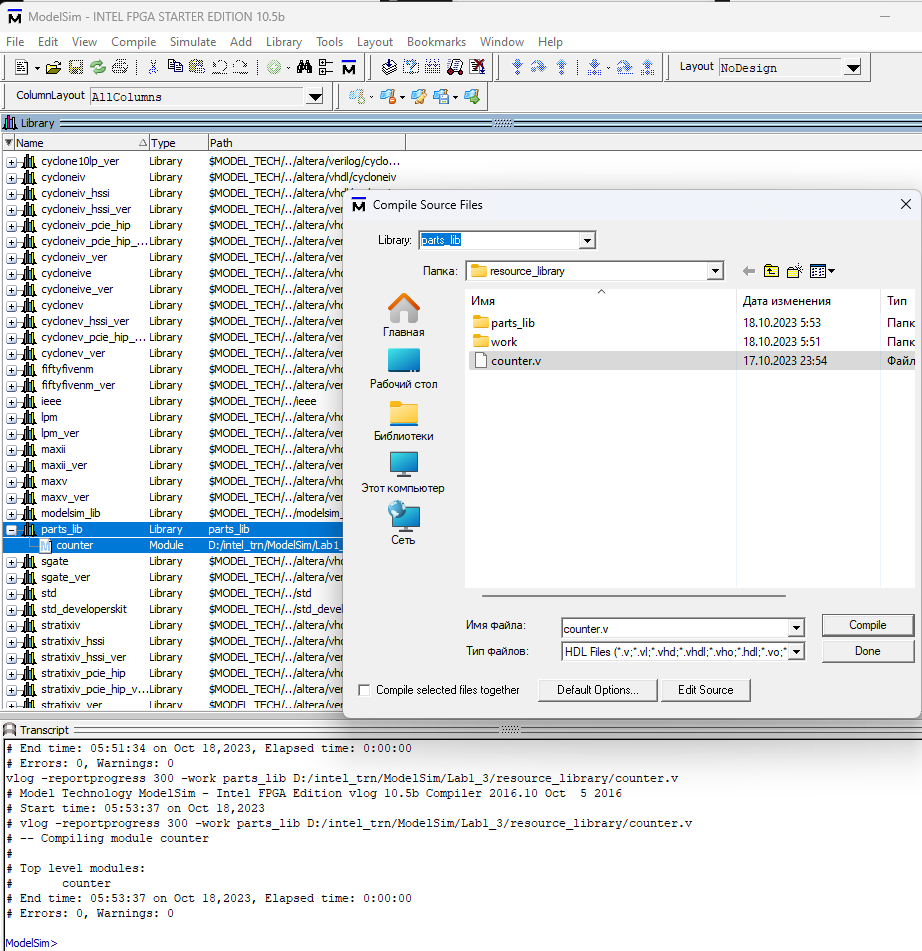


Рис. 3.5 – Создана библиотека ресурсов

## Создание проекта

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 3.6 – Создание проекта

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 3.7 – Добавление test bench

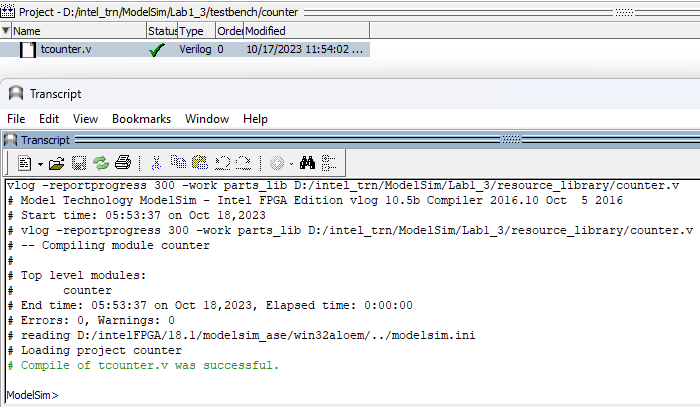


Рис. 3.8 – Компилирование файла

## Загрузка без привязки библиотек

В этой часте лабораторной работы будет осуществляться соединение с библиотекой parts\_lib, которая была создана ранее.

Но сначала попробуем загрузить тестовый стенд без ссылки и посмотрим, что произойдет. Загрузим в симулятор тестовый стенд Verilog:

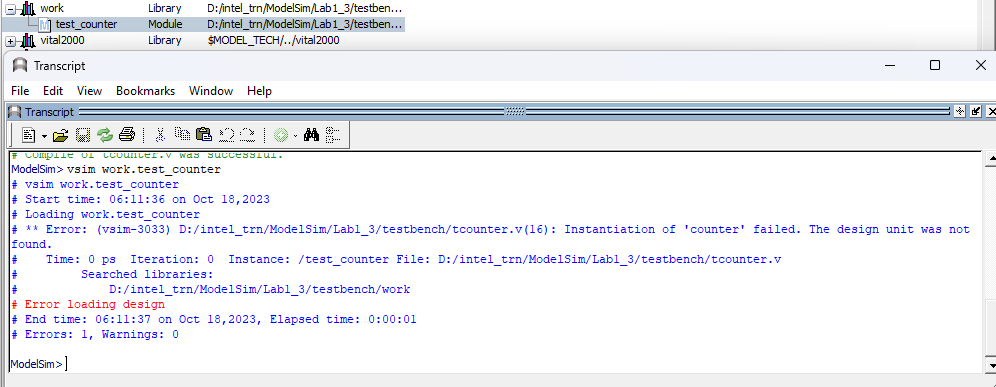


Рис. 3.9 – Загрузка тестового стенда

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 3.10 – Информация об ошибке

## Привязка к библиотеке ресурсов

Для связи с библиотекой ресурсов необходимо указать «библиотеку поиска» при вызове симулятора.

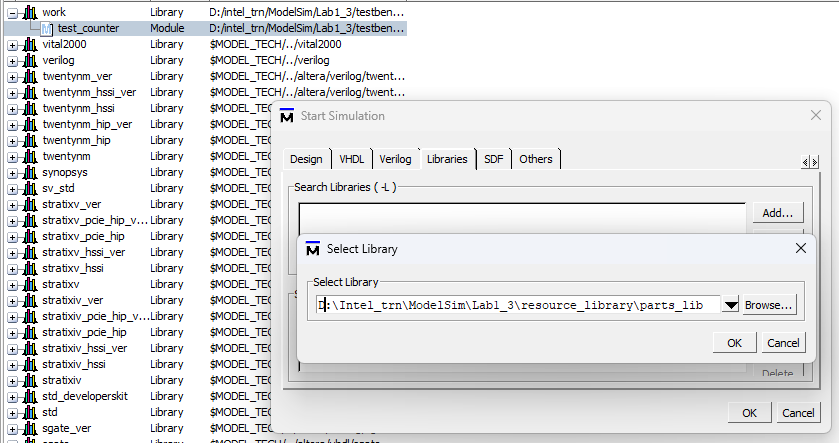


Рис. 3.11 – Привязка к библиотеке

Изображение выглядит как текст, снимок экрана, линия, программное обеспечение

Автоматически созданное описание

Рис. 3.12 – Конечный результат привязки

## Вывод

Работа с несколькими библиотеками в цифровом дизайне, таком как FPGA (программируемые логические интегральные схемы), включает в себя управление и использование заранее разработанных компонентов. Эти компоненты находятся в библиотеках и могут включать логические элементы, триггеры, блоки памяти и другие. При работе с несколькими библиотеками, важно правильно управлять зависимостями, следить за версиями, тщательно документировать и тестировать дизайн перед его внедрением в оборудование. В данной лабораторной работе были исследованы различные способы работы с несколькими библиотеками. На практике была осуществлена привязка к библиотеке ресурсов.

# Lab1\_4: Automating Simulation

Помимо выполнения пары существующих файлов DO, предыдущие уроки были сосредоточены на использовании ModelSim в интерактивном режиме: выполнении отдельных команд одну за другой через меню графического интерфейса или командную строку главного окна. В ситуациях, когда нужно выполнить повторяющиеся задачи, можно повысить свою производительность с помощью файлов DO.

Файлы DO представляют собой сценарии, которые позволяют выполнять множество команд одновременно. Сценарии могут быть такими же простыми, как серия команд ModelSim со связанными аргументами, или они могут быть полноценными программами Tcl с переменными, условным выполнением и т. д. Файлы DO можно запускать из графического пользовательского интерфейса или из командной строки системы, даже не вызывая графический интерфейс.

Файлы с примерами хранятся по следующей директории: D:\intel\_trn\ModelSim\Lab1\_4

## Создание простого DO файла

В этой лабораторной работе нужно будет создать файл DO, который загружает проект, добавляет сигналы в окно Wave, обеспечивает стимул для этих сигналов, а затем продвигает моделирование. Также, нужно будет создать файл DO из сохраненного файла стенограммы.

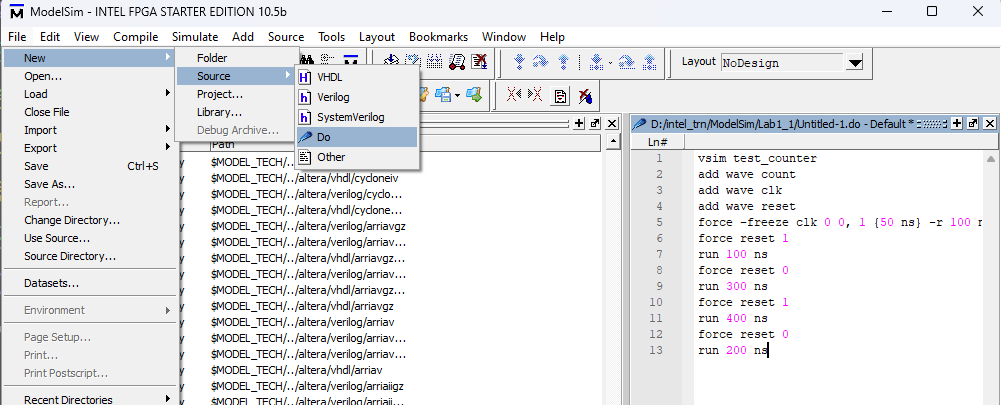


Рис. 4.1 – Создание Do файла

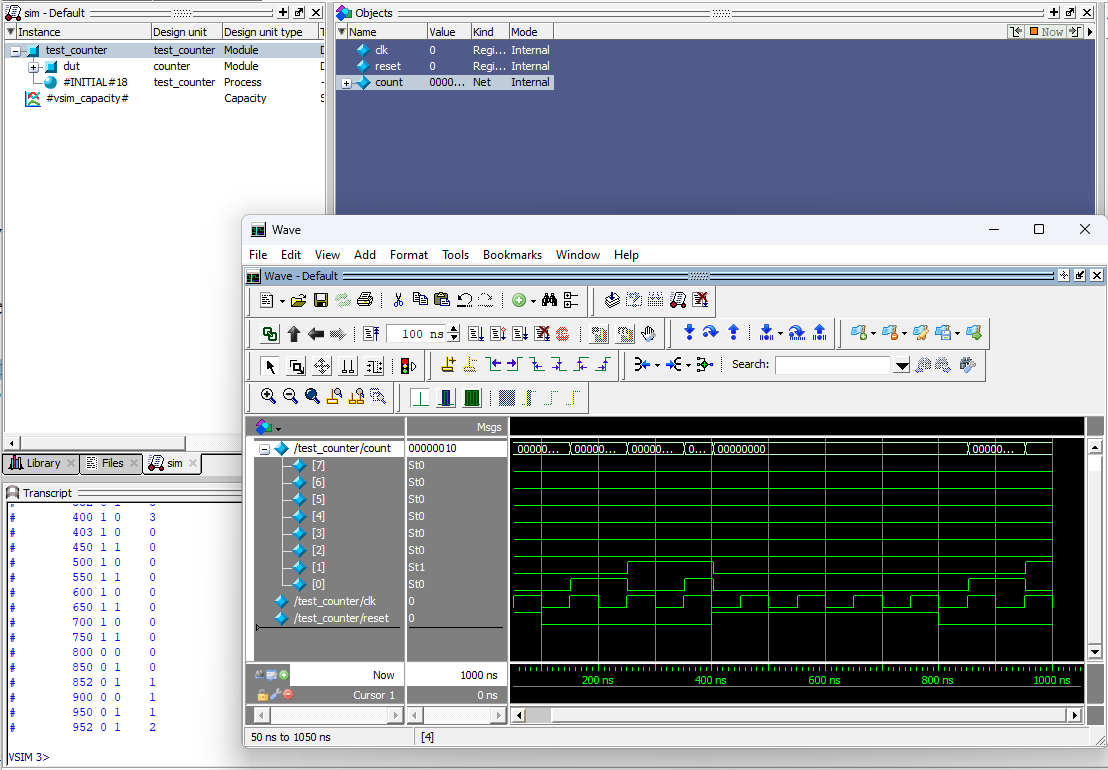


Рис. 4.2 – Компиляция файла sim.do

## Запуск в режиме командной строки

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рис. 4.3 – Добавление библиотеки work в папку



Рис. 4.4 – Запуск в командной строке

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рис. 4.5 – Создание файла sim.do

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 4.6 – Запуск файла sim.do в cmd

* Аргумент -c указывает ModelSim не вызывать графический интерфейс.
* Аргумент -wlf сохраняет результаты моделирования в файле WLF. Это позволяет просматривать результаты моделирования в графическом интерфейсе в целях отладки.

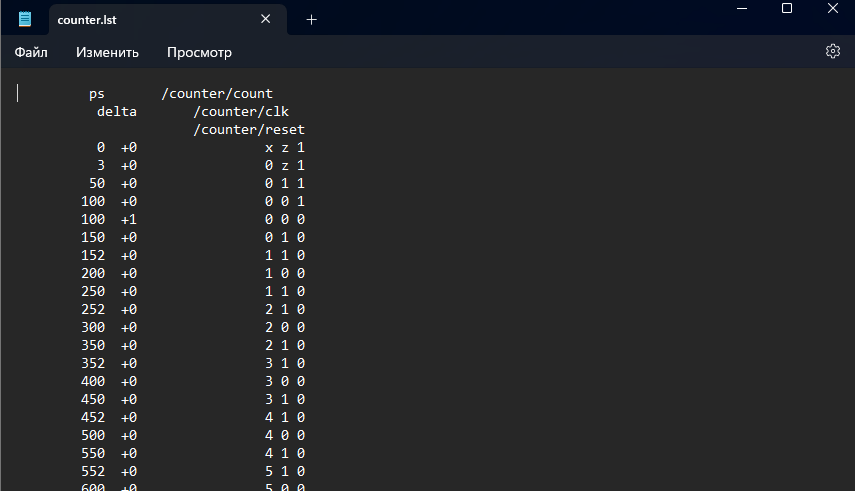


Рис. 4.7 – Созданный файл counter.lst

В окне ModelSim в командную строку впишем **vsim -view counter.wlf**. Откроется графический интерфейс, и отобразится вкладка набора данных с именем «счетчик».



Рис. 4.8 – Временная диаграмма для counter

## Запуск сценария Tcl script

Файлы DO, использованные на предыдущих шагах, содержали только команды ModelSim. Однако файлы DO — это всего лишь сценарии Tcl. Это означает, что они могут включать целый ряд конструкций Tcl, таких как процедуры, условные операторы, математические и триггерные функции, регулярные выражения и т. д.

Создадим простой сценарий, который проверяет определенные значения сигнала, а затем добавляет закладки, которые масштабируют окно Wave, когда это значение существует. Закладки позволяют сохранить определенный диапазон масштабирования и положение прокрутки в окне Wave.

Изображение выглядит как снимок экрана, Шрифт, текст, линия

Автоматически созданное описание

Рис. 4.9 – Создание файла со сценарием (1 часть)

Команды из кода, изображённого на рисунке 4.9 выполняют следующее:

* Создаётся новая процедура под названием «add\_wave\_zoom», имеющая два аргумента: stime и num.
* Создаётся закладка с диапазоном масштабирования: от текущего времени моделирования минус 100 единиц времени до текущего времени моделирования плюс 50 единиц времени.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 4.10 – Создание файла со сценарием (2 часть)

Команды из кода, изображённого на рисунке 4.10 выполняют следующее:

* Добавление количество сигналов в окно Wave в десятичной системе счисления.
* Добавление сигналов clk и сброса в окно Wave.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 4.11 – Создание файла со сценарием (3 часть)

Команды из кода, изображённого на рисунке 4.10 выполняют следующее:

* Использование оператора if, чтобы определить, когда clk переходит в 1.
* Проверка значения count на этих переходах и добавьте закладку, если это определенное значение.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 4.12 – Итоговый вид файла со сценарием

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рис. 4.13 – Подготовка к компиляции

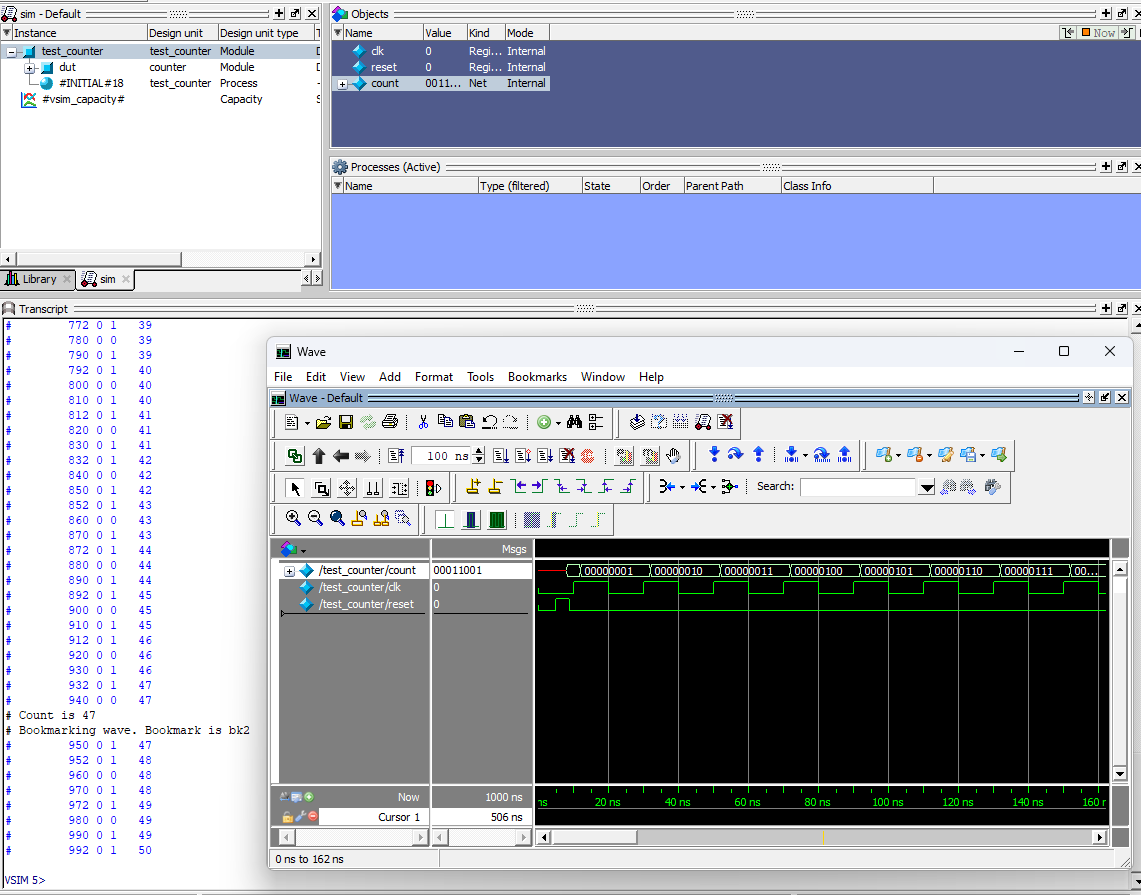


Рис. 4.14 – Результаты компиляции

Были созданы закладки (Bookmarks). Результаты приведены на рисунках 4.15 – 4.16

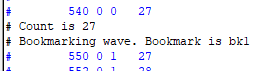


Рис. 4.15 – Bookmarks (1)

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рис. 4.16 – Bookmarks (2)

Выполним переход по только что созданным закладкам:

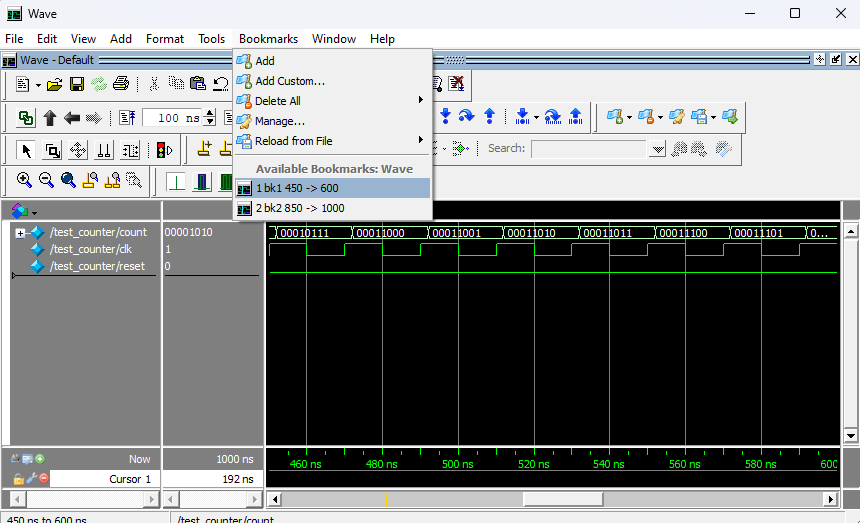


Рис. 4.17 – Переход по закладке bk1 450 -> 600

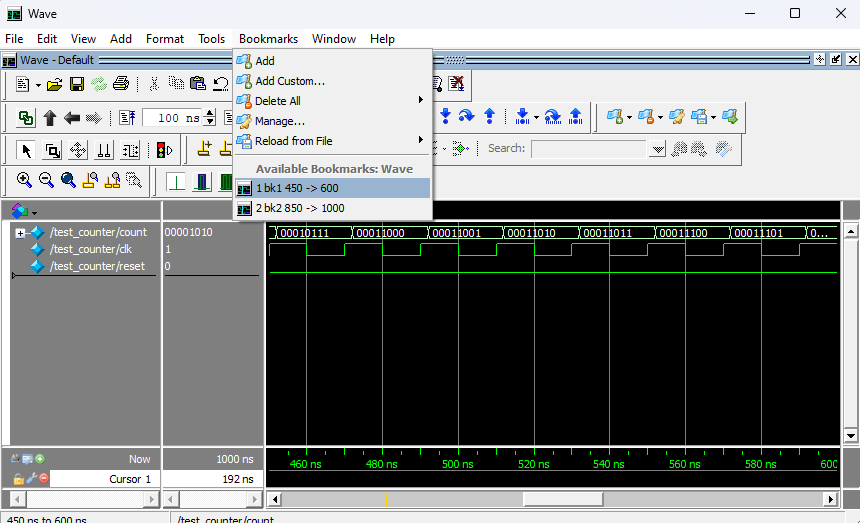


Рис. 4.18 – Переход по закладке bk1 850 -> 1000

## Вывод

Данная работа была посвящена автоматизации симуляции в цифровом проектировании. Она подразумевает использование сценариев или скриптов для упрощения и ускорения процесса симуляции, оптимизации процесса тестирования и анализа результатов, а также улучшения общей производительности при разработке цифровых систем.

# Lab1\_5: Viewing and Initializing Memories

В этой лабораторной работе нужно будет просматривать и инициализировать памяти. ModelSim определяет и перечисляет любое из следующего как память:

* reg, wire arrays
* Integer arrays

Исходные файлы *ram\_tb.v; sp\_syn\_ram.v; dp\_syn\_ram.v* расположены по следующему пути: **D:\Intel\_trn\ModelSim\Lab1\_5**

## Компиляция проекта и загрузка дизайна

Подготовимся к работе с проектом, выбрав нужную директорию, а затем вбив определённые команды в командную строку ModelSim:

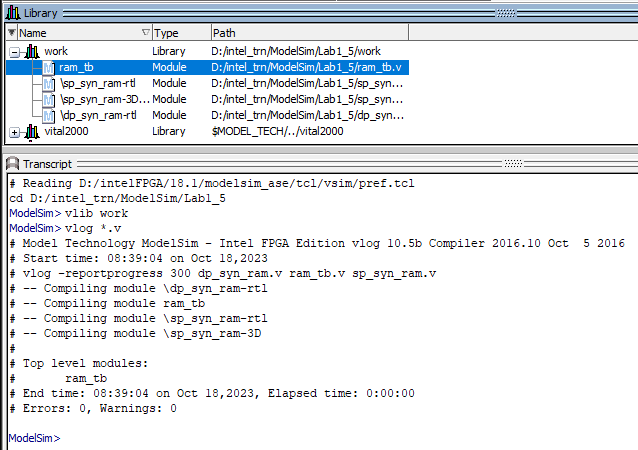


Рис. 5.1 – Подготовка проекта

## Просмотр памяти и её содержание

В окне «Список памяти» перечислены все экземпляры памяти в проекте, показаны для каждого экземпляра диапазон, глубина и ширина. Двойной щелчок по экземпляру открывает окно, отображающее данные памяти.

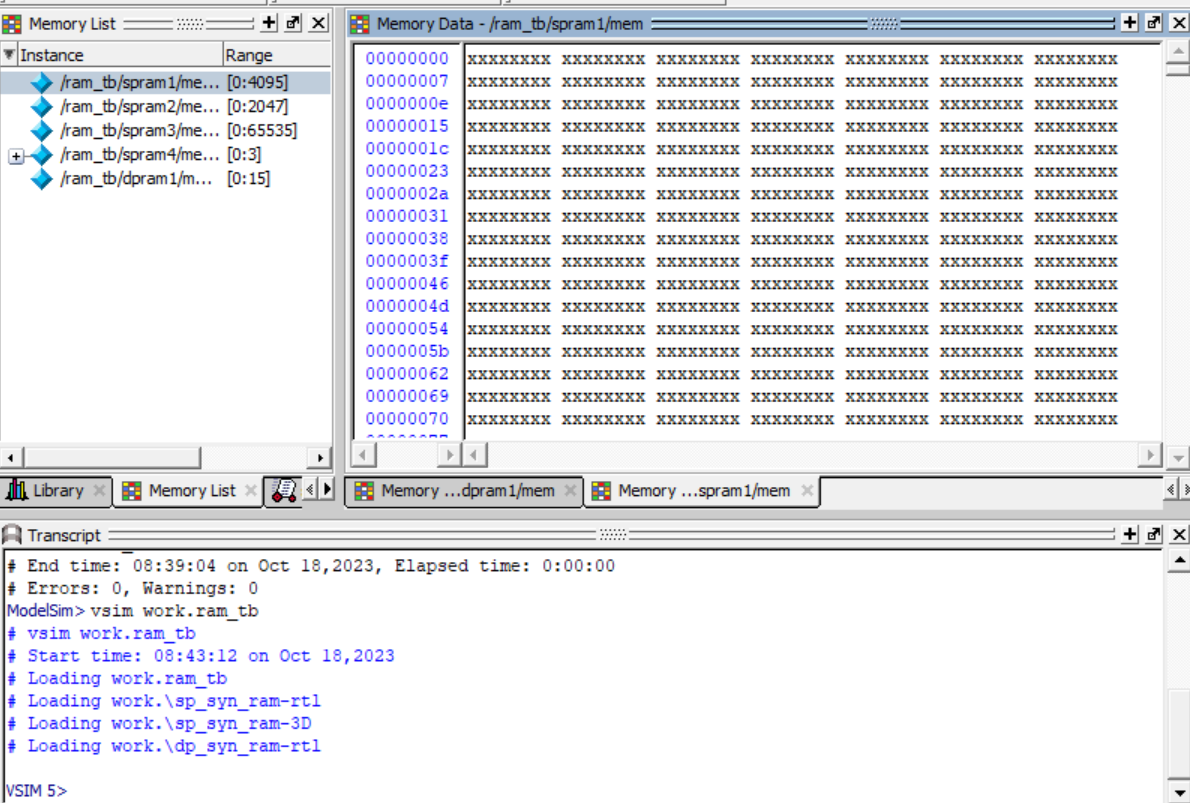


Рис. 5.2 – Содержание памяти для /ram\_tb/spram1/mem

Откроется окно «Данные памяти» (рис. 5.2), отображающее содержимое spram1. В первом столбце (синие шестнадцатеричные символы) указаны адреса, а в остальных столбцах показаны значения данных.

Все данные представляют собой X (рис. 5.2), поскольку вы еще не смоделировали проект.

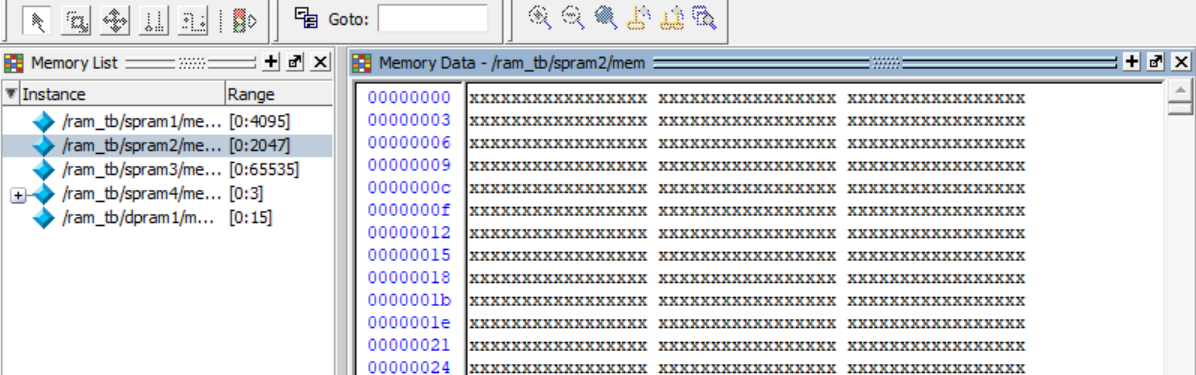


Рис. 5.3 – Содержание памяти для /ram\_tb/spram2/mem



Рис. 5.4 – Исходный код для *ram\_tb*

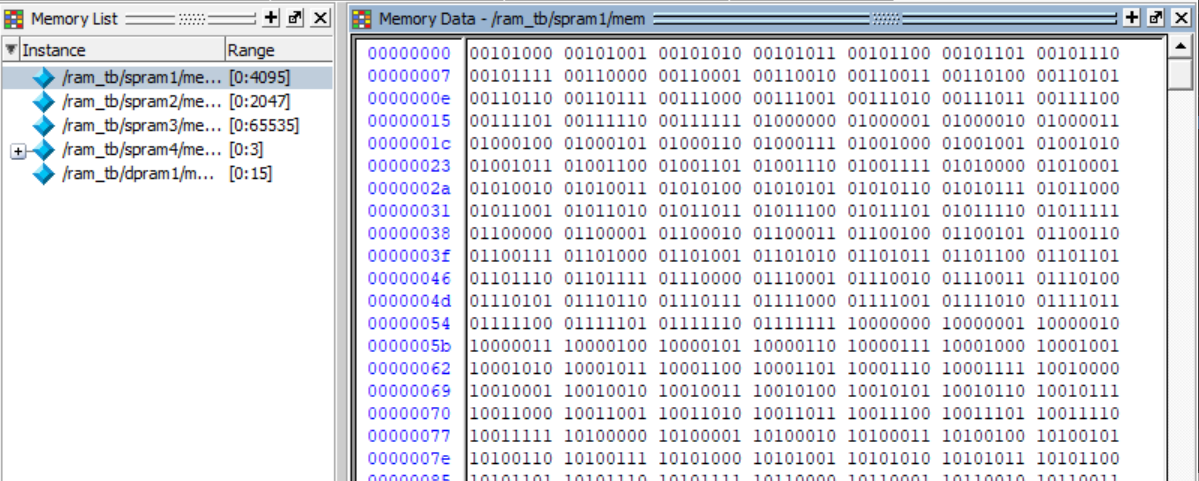


Рис. 5.5 – Обновлённое поле

Изменение системы счисления адреса и количество слов в строке, например /ram\_tb/spram1/mem.

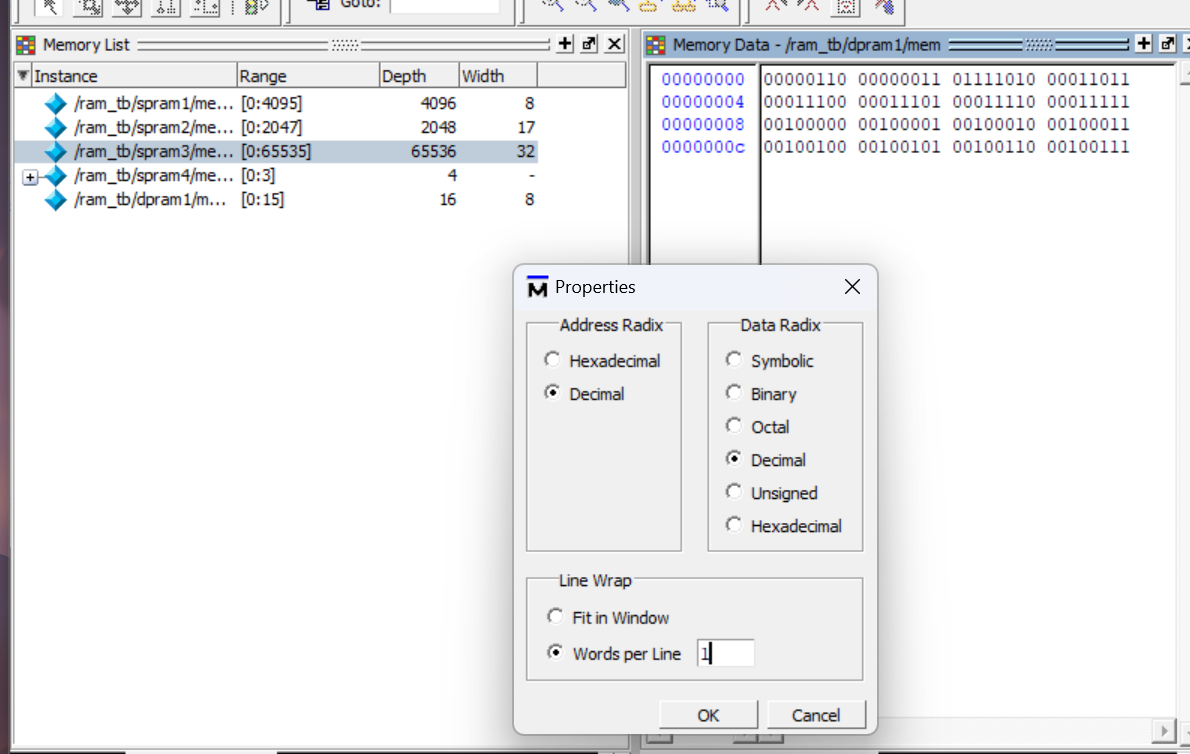


Рис. 5.6 – Изменение значений в окне Properties окна Memory Data

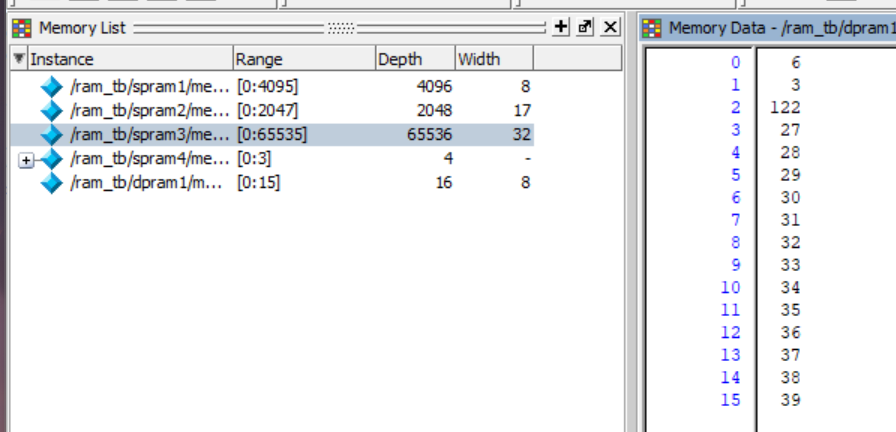


Рис. 5.7 – Обновлённое поле Memory Data

## Навигация по памяти

Существует возможность перейти к определенным адресам памяти или к местам, содержащим определенные шаблоны данных. Для этого используем The Goto dialog box.

Для быстрого перехода по адресу, например, 30, можно внести данный адрес в значение для любого другого поля, переход осуществится автоматически:

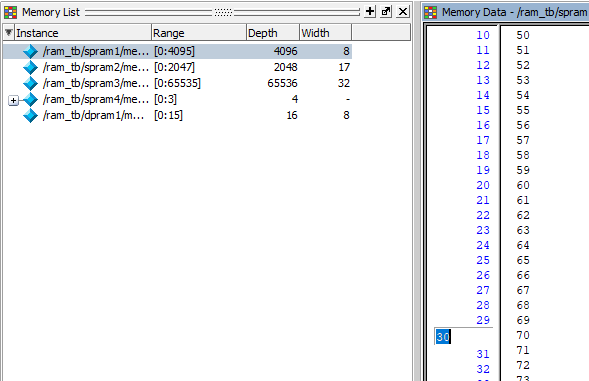


Рис. 5.8 – Быстрый переход по адресу

Поиск ключа по значению можно осуществить с помощью Find data field, кликнув Find Next:

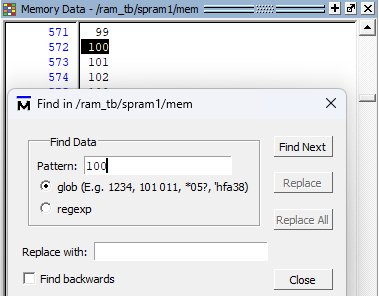


Рис. 5.9 – Поиск с помощью Find data field

## Экспорт поля памяти в файл

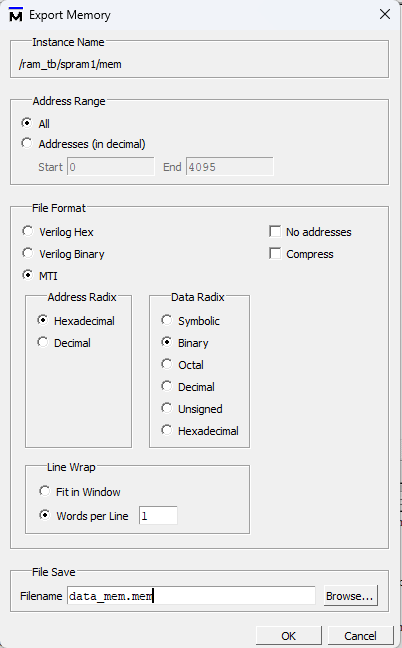


Рис. 5.10 – Окно Export memory

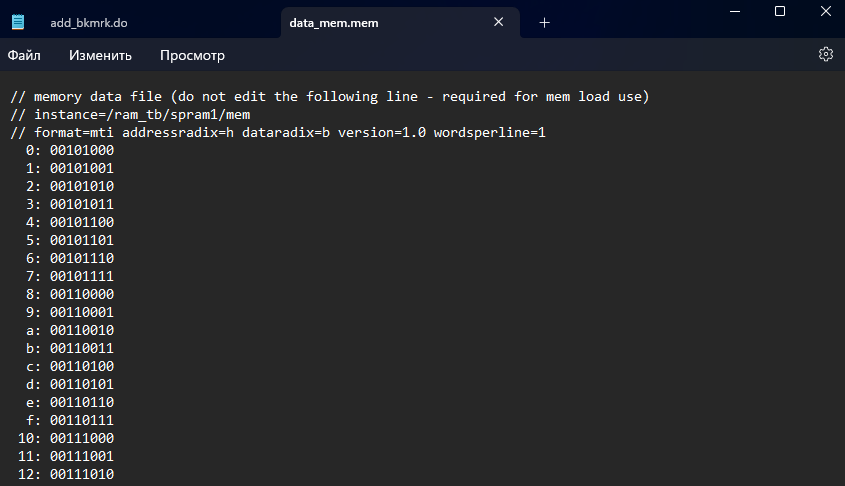


Рис. 5.11 – Файл data\_mem.mem

Файлы шаблонов памяти можно экспортировать как перемещаемые файлы, просто опустив информацию об адресе. Перемещаемые файлы памяти можно загружать в любое место памяти, поскольку адреса не указаны.

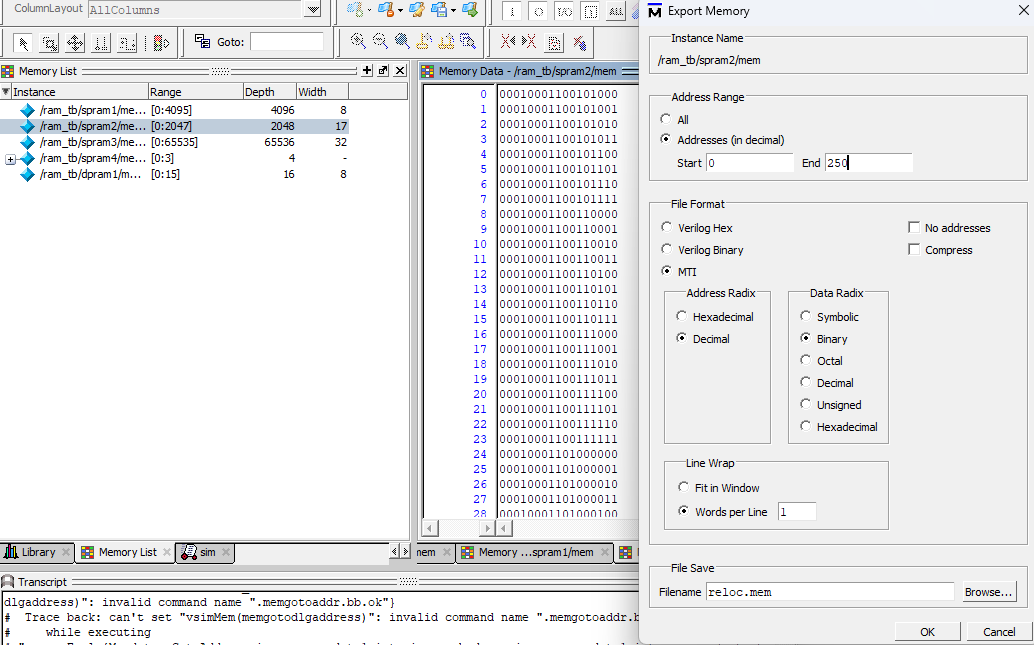


Рис. 5.12 – Настройка Export Memory для */ram\_tb/spram2/mem*

Этот файл будет использован для инициализации в следующей секции

## Инициализация памяти

Инициализацию памяти возможно реализовать 3мя методами: из экспортированного файла памяти, из образца заливки или из того и другого

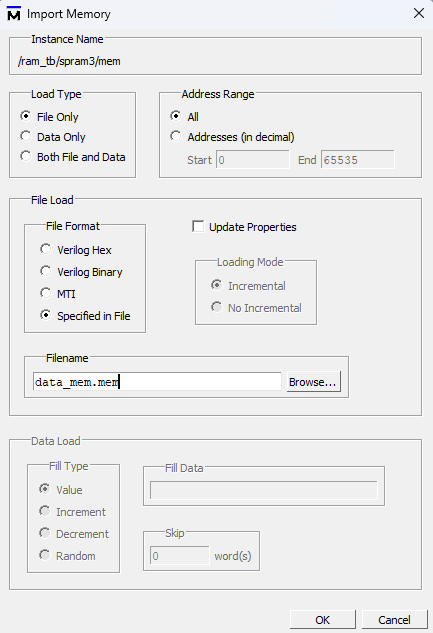


Рис. 5.13 – Заполнение окна Import Memory

Адреса (от 0 до 4095) в экземпляре /ram\_tb/spram3/mem обновляются данными из data\_mem.mem, который содержит только 4096 записей адресов.

На следующем шаге нужно будет поэкспериментировать с импортом как из файла, так и из образца заливки. Проводится инициализация spram3 с помощью 250 адресов данных, которые ранее экспортировались в перемещаемый файл reloc.mem. Также, происходит инициализация 50 дополнительных записей адресов с помощью шаблона заполнения.

Производим импорт из */ram\_tb/spram3/mem* с перемещаемым шаблоном памяти (reloc.mem) и шаблоном заполнения:

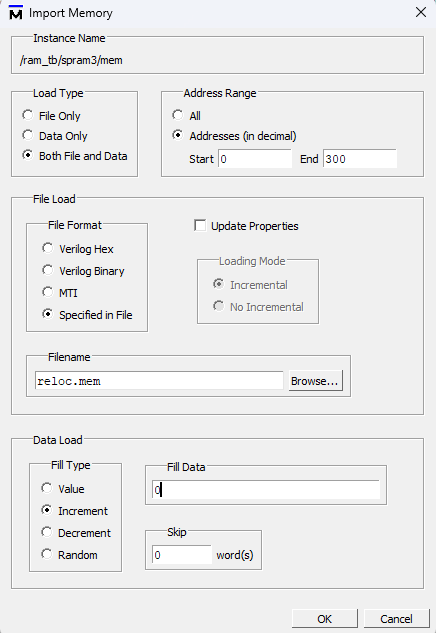


Рис. 5.14 – Настройка Import Memory

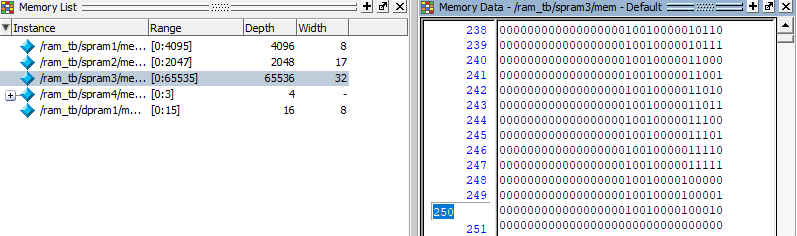


Рис. 5.15 – Выходные значения

## Команды интерактивной отладки

Окна данных памяти также можно использовать в интерактивном режиме для различных целей отладки. Для этой цели полезны функции, описанные в этом разделе.

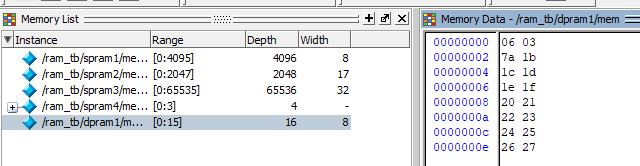


Рис. 5.16 – Окно Memory Data после изменения окна Properties

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рис. 5.17 – Настройка окна Change Memory

Данные в указанном диапазоне заменяются сгенерированным случайным шаблоном заполнения.

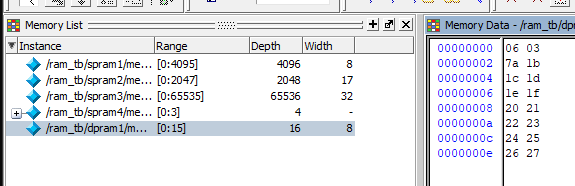


Рис. 5.18 – Замена данных в указанном диапазоне

## Вывод

"Просмотр и инициализация памяти" - это процессы, связанные с управлением памятью в цифровых системах. Просмотр памяти предполагает анализ и проверку содержимого памяти, что может быть важно для отладки и анализа работы системы. Инициализация памяти заключается в установке начальных значений в памяти, что может быть необходимо при запуске системы или при восстановлении ее состояния. Эти процессы играют важную роль в разработке цифровых систем и обеспечивают их корректное и надежное функционирование.