

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО**

П. В. Кустарев, С. В. Быковский

Функциональная схемотехника

Методические указания к лабораторной работе №1



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург

2019

Содержание

1 Лабораторная работа №1. «Введение в проектирование цифровых интегральных схем»	4
1.1 Цели работы	4
1.2 Указания к выполнению работы	4
1.3 Часть 1. Порядок выполнения	5
1.4 Часть 2. Порядок выполнения	5
1.5 Задания по вариантам	6
2 Требования к оформлению отчета	7
3 Основы работы в среде LTspice	9
3.1 О моделировании в SPICE	9
3.2 Ограничения области применимости SPICE	9
3.3 О программе LTspice	10
3.4 Создание инвертора	10
3.4.1 Подготовка к работе	10
3.4.2 Создание и симуляция схемы	11
3.5 Создание иерархических элементов	14
4 Основы работы в Vivado Design Suite	17
4.1 Схема сумматора	17
4.2 Проектирование схемы сумматора на вентильном уровне с использованием Verilog HDL	18
4.3 Создание проекта	22
4.4 Моделирование схемы	29
Рекомендуемая литература	33

Лабораторная работа №1. «Введение в проектирование цифровых интегральных схем»

1.1 Цели работы

- Получить базовые знания о принципах построения цифровых интегральных схем с использованием технологий КМОП.
- Познакомиться с технологией SPICE моделирования схем на транзисторах.
- Получить навыки описания схем базовых операционных элементов (БОЭ) комбинационного типа на вентильном уровне с использованием языка описания аппаратуры Verilog HDL.

1.2 Указания к выполнению работы

Лабораторная работа имеет две части.

Первая часть посвящена проектированию цифровых вентилей на полевых транзисторах, построению схем на базе вентилей и знакомству с технологией SPICE моделирования. Первая часть работы выполняется в программном пакете LTspice. При построении схем вентилей необходимо использовать КМОП-транзисторы с параметрами из файла 90mn_bulk.txt (см. раздел «Основы работы в среде LTspice»).

Вторая часть посвящена знакомству с языком описания аппаратуры Verilog HDL, особенностью использования его для описания схем на вентильном уровне и приобретению навыков их тестирования. Вторая часть работы выполняется с использованием Vivado Simulator, входящим в пакет Vivado Design Suite.

По результатам выполнения обеих частей лабораторной работы составляется отчета в соответствии с требованиями, приведенными в разделе «Требования к оформлению отчета».

1.3 Часть 1. Порядок выполнения

1. Постройте в LTspice схему вентиля на транзисторах, являющегося основой логического базиса, указанного в варианте задания.
2. Постройте символ для разработанного вентиля.
3. С использованием построенного символа постройте схему тестирования вентиля.
4. Проведите моделирование работы схемы и определите задержку распространения сигнала через тестируемый вентиль.
5. Определите максимальную частоту изменения входных сигналов, при которой построенная схема сохраняет работоспособность.
6. Постройте на базе созданного вентиля БОЭ, согласно варианту задания.
7. Создайте символ для построенного БОЭ.
8. Проведите моделирование работы схемы и определите задержку распространения сигнала через БОЭ.
9. Определите максимальную частоту изменения входных сигналов, при которой построенная схема сохраняет работоспособность.
10. Составьте отчет по результатам выполнения заданий первой части лабораторной работы.

1.4 Часть 2. Порядок выполнения

1. Опишите на Verilog HDL на вентильном уровне модуль, реализующий функцию БОЭ в указанном логическом базисе согласно варианту задания.
2. Разработайте тестовое окружение для созданного модуля.
3. Проведите моделирование работы схемы.
4. Составьте отчет по результатам выполнения заданий второй части лабораторной работы.

1.5 Задания по вариантам

№ Варианта	Логический базис	БОЭ
1	NOR	Демультиплексор «1 в 4»
2	NAND	Полный четырехразрядный компаратор
3	NOR	Схема мажоритарного контроля с 5-ю входами
4	NAND	Позиционный шифратор «8 в 3»
5	NOR	Преобразователь BCD-кода в двоичный код (числа от 0 до 99)
6	NAND	Позиционный дешифратор «3 в 8»
7	NOR	Четырехразрядный двоичный сумматор с переносом
8	NAND	Шифратор кода Грея для трехразрядного двоичного числа
9	NOR	Мультиплексор «4 в 1»
10	NAND	Дешифратор трехразрядного кода Грея

Требования к оформлению отчета

Отчет выполняется в виде самодостаточного документа. Материал, изложенный в отчете, должен быть понятным без дополнительных комментариев со стороны исполнителей.

На защиту предоставляется только распечатанный титульный лист. Электронная версия отчета высылается на почту преподавателю.

Отчет должен содержать:

- Титульный лист, на котором указываются:
 - название университета;
 - название факультета;
 - название дисциплины;
 - номер и тема лабораторной работы;
 - вариант лабораторной работы;
 - фамилия, инициалы и номер группы каждого исполнителя;
 - фамилия и инициалы преподавателя;
 - текущий год.
- Содержание.
- Цели работы.
- Задание в соответствии с вариантом.
- Отчет о выполнении заданий части 1:
 - Схема разработанного вентиля.
 - Символ вентиля и схема тестирования.
 - Временная диаграмма процесса тестирования вентиля. Недопускается изображение нескольких сигналов друг на друге. Сигналы должны быть разделены на диаграмме.

- Результат измерения задержки распространения сигнала через вентиль.
 - Максимальная частота работы вентиля.
 - Схема разработанного БОЭ.
 - Символ разработанного БОЭ и схема тестирования.
 - Временная диаграмма процесса тестирования БОЭ. Недопускается изображение нескольких сигналов друг на друге. Сигналы должны быть разделены на диаграмме.
 - Результат измерения задержки распространения сигнала через БОЭ.
 - Максимальная частота работы БОЭ.
- Отчет о выполнении заданий части 2:
 - Код разработанного модуля БОЭ.
 - Код разработанного тестового окружения БОЭ.
 - Временная диаграмма процесса тестирования БОЭ.

Требования к оформлению:

- отчет выполняется как текстовый документ в соответствии с ГОСТ 2.105-95;
- шрифт Times New Roman 12-14 pt, межстрочный интервал 1-1,5, поля с краев листа – не менее 2 см;
- сквозная нумерация страниц;
- обязательны нумерация и подписи к рисункам и таблицам, а также ссылки на них в тексте отчета;
- схемы и временные диаграммы должны быть темными на светлом фоне; если наложение временных диаграмм нескольких сигналов мешает их однозначному восприятию, они должны разноситься на отдельные координатные сетки;
- в распечатанном отчете линии на схемах и временных диаграммах должны быть четко видны.

Основы работы в среде LTspice

3.1 О моделировании в SPICE

SPICE (Simulation Program with Integrated Circuit Emphasis) – программа для моделирования поведения интегральных схем и схем на печатных платах. Изначально разработана в Калифорнийском университете в Беркли в начале 1970-х гг. В дальнейшем на ее основе были созданы другие программы, одной из которых является LTspice.

Необходимость моделирования вызвана тем, что создать дешевый прототип интегральной схемы практически невозможно – для этого ее надо фактически изготовить, что является очень затратным мероприятием. SPICE позволяет промоделировать работу схемы в разных условиях и выявить потенциальные проблемы до того, как она будет выпущена.

SPICE содержит модели различных электронных компонентов и элементов схем – транзисторов, резисторов, конденсаторов, источников напряжения, усилителей и т.п. Соединяя компоненты, можно построить произвольную схему, а затем подвергнуть ее анализу одним из поддерживаемых способов: передаточная функция, анализ по постоянному току, анализ по переменному току и др.

3.2 Ограничения области применимости SPICE

Несмотря на то, что SPICE во многих случаях может заменить прототипирование, часто результаты моделирования могут оказаться недостаточно точными или даже совсем не совпадать с работой реальной схемы. Реальные электронные компоненты являются сложными устройствами, работа которых зависит от множества параметров. Результаты моделирования точны настолько, на сколько точны модели компонентов.

Модели схем в SPICE свободны от шума, перекрёстных помех, паразитных емкостей и др., если только они явно не внесены в модель проектировщиком. Например, сигнал в несколько пикоампер может корректно управлять схемой в SPICE-модели, но в физическом мире будет полностью забит шумом и паразитными цепями.

SPICE-моделирование не подходит для исследования излучающих и принимающих схем (антенн). Для задач такого рода используются системы, построенные на численном решении уравнений Максвелла.

SPICE не является лучшим средством для предсказания поломки компонентов. Моделирование не выдаст никакого сигнала предупреждения в случае, если будет превышено максимально допустимое значение тока или напряжения для какого-то из компонентов. Например, проектировщик может не заметить тока в несколько ампер на одной из цепей при моделировании схемы, но после изготовления такая схема при включении сгорит.

3.3 О программе LTspice

Загрузить программу LTspice можно с сайта компании Analog Devices по адресу: <https://www.analog.com/en/design-center/design-tools-and-calculators/lmspice-simulator.html>. Программа является бесплатной. Доступны версии под Windows и Mac OS, под Linux LTspice можно запустить в Wine. Версия для Mac OS имеет некоторые отличия от версии для Windows.

LTspice имеет встроенную справку с описанием основных приемов работы. Также, в интернете можно найти множество уроков и статей по использованию LTspice.

В рамках данного курса будут использоваться модели транзисторов для техпроцесса 90 нм, полученные путем моделирования. Цифра в названии технологии, как правило, означает минимальную длину канала транзистора. Т.к. увеличение длины канала транзистора в цифровых схемах не дает никакого положительного эффекта, все транзисторы в рамках этого курса будут иметь длину канала равную 90 нм. Минимальная допустимая ширина канала обычно в два раза больше минимальной допустимой длины, т.е. для техпроцесса 90 нм минимальная ширина канала равна 180 нм. Увеличение ширины канала увеличивает его проводимость и емкости между затвором и каналом.

3.4 Создание инвертора

Рассмотрим основные принципы работы в среде LTspice и познакомимся с КМОП-схемотехникой на примере простейшего цифрового вентиля – инвертора.

3.4.1 Подготовка к работе

Создайте рабочий каталог для сохранения файлов проекта. Загрузите файл с параметрами транзисторов 90nm_bulk.txt (предоставляется преподавателем) и со-

храните его в своем рабочем каталоге. В этом же каталоге следует сохранять схемы (файлы *.asc, *.asy), выполняемые в рамках курса.

Запустите программу LTspice. Окно программы представлено на рис. 3.1.

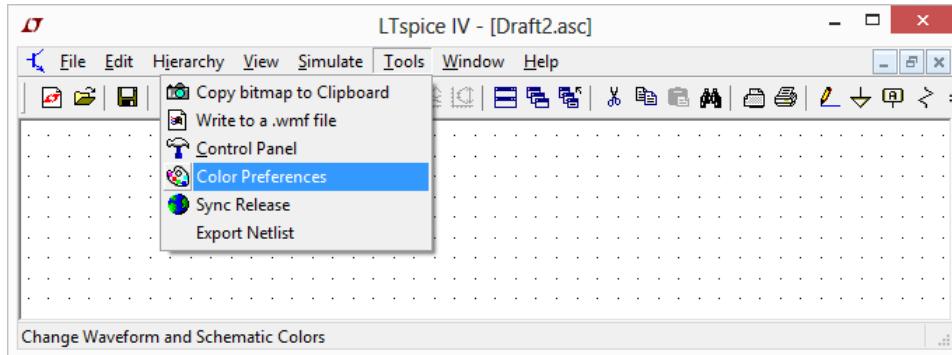


Рисунок 3.1: Окно программы LTspice

В меню Tools → Color Preferences (рис. 3.1) можно настроить цвета, в которых изображаются рабочее поле и схема.

Включить отображение сетки в рабочем поле можно с помощью View → Show Grid.

3.4.2 Создание и симуляция схемы

Создайте новую схему (File → New Schematic) и сохраните ее файл в своем рабочем каталоге. Добавьте к схеме SPICE-директиву, подключающую параметры для транзисторов (Edit → SPICE Directive, рис. 3.2). Помните, что директивы SPICE начинаются с точки. Разместите директиву на схеме (рис. 3.3).

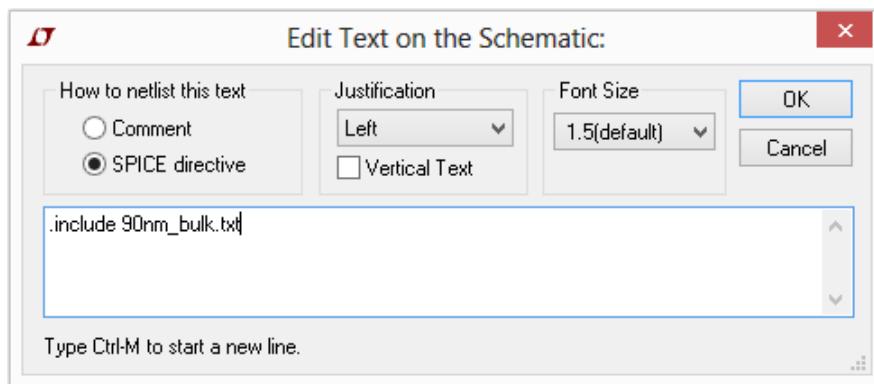


Рисунок 3.2: Окно ввода директивы

На рис. 3.4 показана схема тестирования инвертора – простейшего цифрового вентиля. Схема включает: КМОП-инвертор из двух транзисторов (PMOS и NMOS), источник питания инвертора (VDD1), источник тестового входного напряжения (V1)

.include 90nm_bulk.txt

Рисунок 3.3: Директива, вставленная в схему

и нагрузку выхода инвертора (R_1 и C_1). Нагрузка необходима, чтобы моделировать реальную ситуацию, когда выход инвертора управляет другими вентилями.

Постройте схему. Для этого добавьте на схему транзисторы pmos4 и nmos4, компоненты voltage, res, cap (Edit → Component) и землю (Edit → Place GND). Сориентируйте элементы нужным образом (Edit → Rotate и Edit → Mirror), а затем соедините проводами (Edit → Draw Wire). Проводам можно присваивать имена с помощью Edit → Label Net (просто ввести имя, тип порта оставить None). Если присвоить двум проводам в разных местах схемы одинаковое имя, это будет эквивалентно их соединению. На схеме тестирования инвертора таким образом соединены все фрагменты провода с именем VDD.

Отредактируйте параметры компонентов в соответствии со схемой на рис. 3.4. Для редактирования параметров необходимо щелкнуть на компоненте правой кнопкой мыши.

Для транзисторов необходимо ввести длину и ширину (l и w) канала в нанометрах. Заметьте, что ширина канала PMOS-транзистора в два раза больше, чем ширина канала NMOS-транзистора. Это связано с тем, что мобильность дырок примерно в два раза ниже мобильности электронов. Следовательно, чтобы обеспечить одинаковую проводимость каналов, ширину PMOS-канала нужно сделать в два раза больше. Чтобы вывести параметры транзистора рядом с компонентом (как на рис. 3.4), необходимо настроить видимость параметров компонента: при зажатой клавише Ctrl щелкнуть правой кнопкой мыши на компоненте и отметить столбец Visible для поля Value2.

Для источника питания инвертора VDD1 задайте постоянное значение напряжения 1 В. Для источника напряжения V1 задайте параметры, как показано на рис. 3.5 (для этого в окне параметров необходимо нажать кнопку Advanced). Источник V1 генерирует прямоугольные импульсы (чертежование «0» и «1»), которые будут поданы на вход инвертора для его тестирования. Заметьте, что буквы после чисел обозначают приставки кратных и дольных величин в системе СИ (к – кило-, н –nano-, р – пико-, ф – фемто- и т.д.).

Добавьте на схему настройки анализа переходного процесса (Edit → SPICE Analysis) с параметрами, как показано на рис. 3.6. Настройки задают время моделирования 8 нс и шаг моделирования 1 пс.

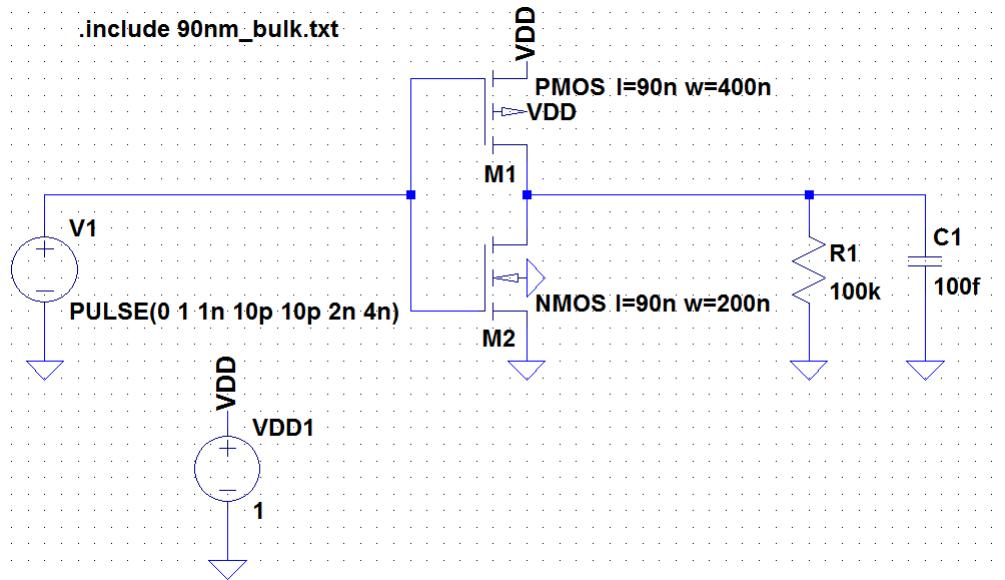


Рисунок 3.4: Схема тестирования инвертора

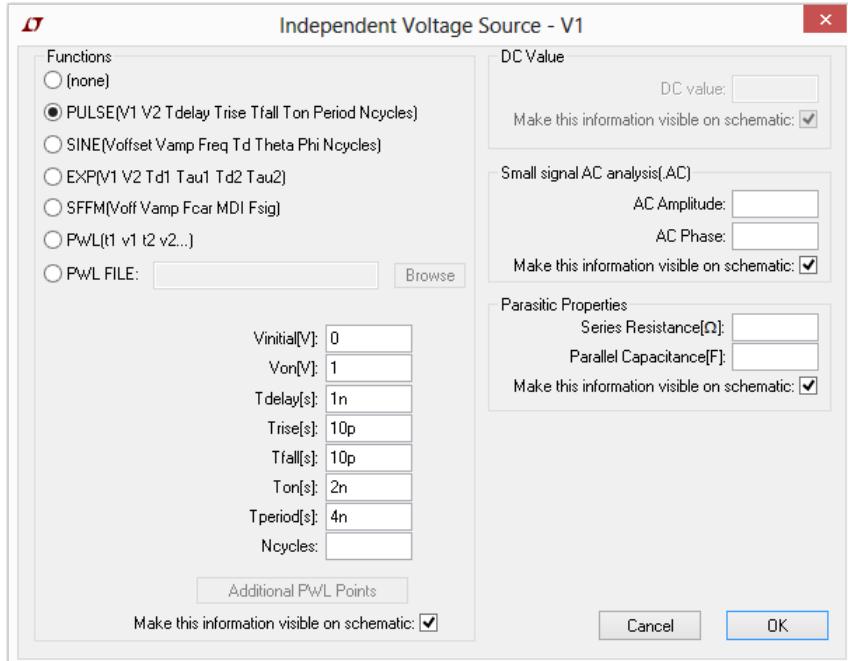


Рисунок 3.5: Окно настроек источника напряжения

Запустите симуляцию (Simulate → Run). Появится окно для вывода временных диаграмм. Наведите курсор мыши на провод перед инвертором; курсор примет вид щупа осциллографа. Щелкните на проводе, чтобы добавить временную диаграмму входного напряжения инвертора на график. Аналогично добавьте диаграмму выходного напряжения. График должен иметь вид, как на рис. 3.7. Выходное напряжение инвертора противоположно входному, а значит, инвертор работает корректно.

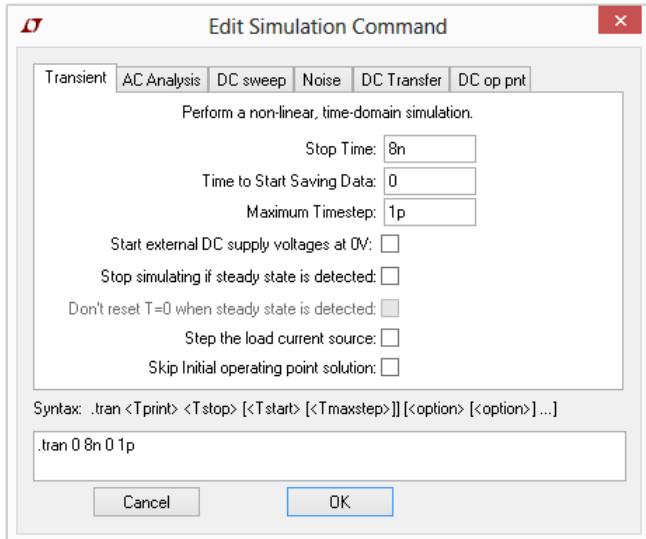


Рисунок 3.6: Окно настроек процедуры анализа переходного процесса

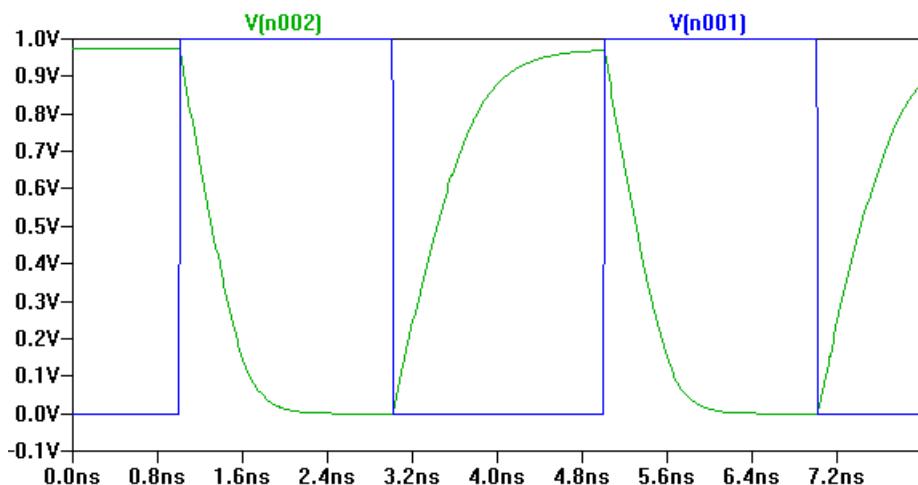


Рисунок 3.7: Результаты моделирования переходного процесса в инверторе

3.5 Создание иерархических элементов

Для разработки более сложных схем полезно будет иметь готовый инвертор в «библиотеке» и вставлять его в новые схемы как иерархический элемент.

Для создания библиотечного элемента требуется создать схему (File → New Schematic) и символ (File → New Symbol) с одинаковыми именами, например, inverter.

Постройте схему инвертора, как показано на рис. 3.8. У этого инвертора будет настраиваемый параметр – множитель для ширины канала транзисторов. Это позволит создавать инверторы разного размера.

Для создания параметра поместите на схему SPICE-директиву «.param», как показано на рис. 3.8. У параметра задано значение по умолчанию, равное 1. Имя

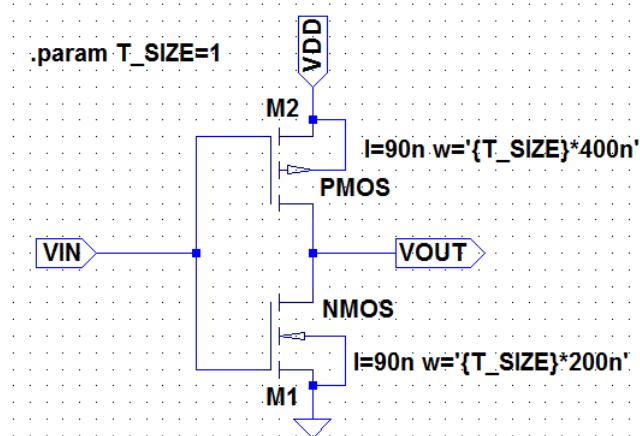


Рисунок 3.8: Схема инвертора

параметра можно использовать при задании параметров компонентов схемы. В нашем случае он использован при вводе ширины канала транзисторов. Заметьте, что выражение для ширины канала транзистора должно быть заключено в одиночные кавычки.

В схеме необходимо создать именованные порты ввода и вывода с помощью Edit → Label Net и выбора типа порта.

В качестве символов элементов следует использовать общепринятые обозначения элементов по стандартам ГОСТ, IEC или ANSI. Нарисуйте ANSI-символ инвертора, как показано на рис. 3.9. Для рисования используйте команды из меню Draw. Добавьте в символ порты (Edit → Add Pin/Port) с такими же названиями, как и в схеме элемента. Для порта выберите, надо ли отображать название, и если да, то с какой стороны от него будет отображаться точка электрического соединения.

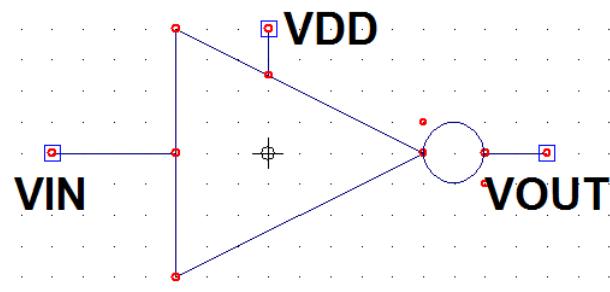


Рисунок 3.9: Символ инвертора

Теперь инвертор можно вставлять в новые схемы как библиотечный элемент. Постройте схему тестирования инвертора с применением иерархического символа (рис. 3.10). Промоделируйте схему. Результаты должны совпасть с результатами моделирования предыдущей схемы.

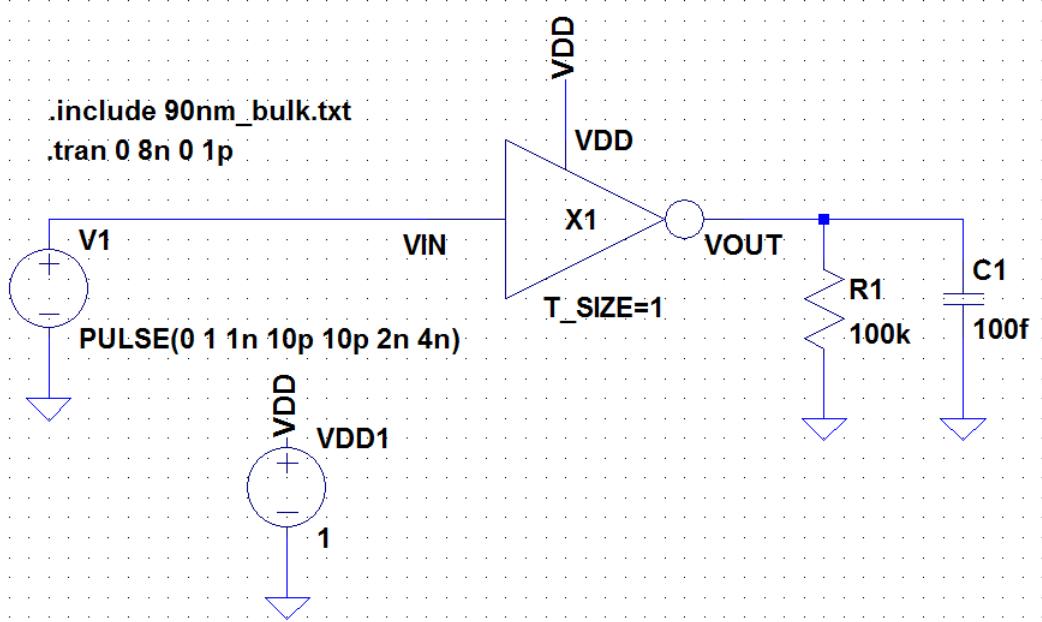


Рисунок 3.10: Схема тестирования инвертора с использованием разработанного символа

Отредактируйте параметры инвертора. Для этого щелкните по символу правой кнопкой мыши и в строке PARAMS введите «T_SIZE=4». Промоделируйте схему еще раз. Длительность переходных процессов при переключении инвертора должна уменьшиться.

Основы работы в Vivado Design Suite

В данном разделе описаны принципы работы в системе автоматизированного проектирования (САПР) Vivado Design Suite (далее Vivado) фирмы Xilinx. Версия WebPACK является бесплатной и доступна для скачивания с официального сайта Xilinx (<https://www.xilinx.com/>).

Vivado предназначена для проектирования программного обеспечения микросхем программируемой логики (ПЛИС) фирмы Xilinx. В разделе описан пример проектирования в Vivado версии 2017.4. Описанные рекомендации также применимы и для более ранних и поздних версий с учетом небольших различий версий в элементах пользовательского интерфейса.

4.1 Схема сумматора

Чтобы ознакомиться с принципами проектирования цифровых схем в Vivado, разберем пример построения схемы неполного одноразрядного сумматора. Таблица истинности рассматриваемой схемы сумматора представлена в таблице 4.1.

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 4.1: Таблица истинности неполного одноразрядного сумматора

Построим данную схемы в базисе И-НЕ. Для этого сначала запишем совершенную дизъюнктивную нормальную форму (СДНФ) функции сумматора:

$$a + b = \bar{a} \cdot b \vee a \cdot \bar{b}. \quad (4.1)$$

Возьмем двойное отрицание и преобразуем СДНФ по правилам де Моргана:

$$\overline{\overline{\overline{a} \cdot b \vee a \cdot \bar{b}}} = \overline{\overline{\bar{a} \cdot b}} \cdot \overline{\overline{a \cdot \bar{b}}} \quad (4.2)$$

В итоге для выполнения суммирования двух одноразрядных числа нам понадобится 5 двухвходовых элемента И-НЕ. Схема сумматора на И-НЕ представлена на рисунке 4.1.

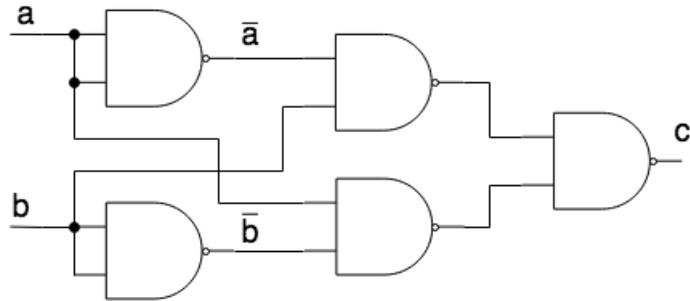


Рисунок 4.1: Схема сумматора в базисе И-НЕ

4.2 Проектирование схемы сумматора на вентильном уровне с использованием Verilog HDL

Рассмотрим пример описания схемы, изображенной на рисунке 4.1, на языке Verilog HDL, а также пример описания тестового окружения для этой схемы.

В данном разделе мы будем описывать схему на вентильном уровне, то есть используя только примитивы вентилей, доступные в Verilog HDL.

Дадим всем линиям схемы обозначения, которые затем будем использовать в текстовом описании схемы. Аннотированная схема представлена на рисунке 4.2.

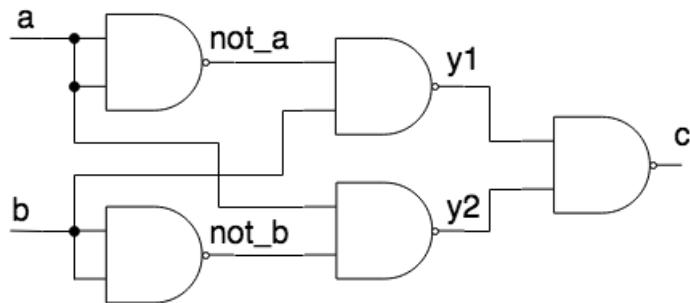


Рисунок 4.2: Аннотированная схема сумматора в базисе И-НЕ

Полное описание модуля, описывающий функциональность схемы, на Verilog HDL представлен в листинге 4.1. В Verilog HDL базовой единицей схемы является модуль. Вся схема строится из модулей. Вся логика описывается внутри модулей. Рекомендуется описывать каждый модуль в виде отдельного файла, хотя это не является обязательным.

Модули могут входить в состав других модулей, образуя при этом иерархическую структуру целевого устройства. Модули в Verilog обрамляются ключевыми словами **module** и **endmodule** в начале и конце соответственно.

```
'timescale 1ns / 1ps

module adder (
    input  a,
    input  b,
    output c
);

    wire not_a, not_b, y1, y2;

    nand(not_a, a, a);
    nand(not_b, b, b);

    nand(y1, not_a, b);
    nand(y2, a, not_b);

    nand(c, y1, y2);

endmodule
```

Листинг 4.1: Описание модуля сумматора на Verilog HDL

В самом верху исходного файла располагается директива **timescale**, которая задает шаг (1ns) и точность (1ps) продвижения модельного времени. Данная директива не является обязательной и, чтобы она работала, необходимо, чтобы она была одинакова во всех файлах проекта. При этом данная настройка может быть переопределена в настройках симулятора.

После ключевого слова **module** записывается уникальное имя модуля. В нашем случае имя модуля **adder**. Далее идет определение портов с указанием направления передачи данных: входной (**input**) или выходной (**output**) порт.

По умолчанию тип переменной порта, если он не указан, задается как **wire**. В Verilog также существует ещё один синтезируемый тип переменных и портов - это **reg**. Когда говорится, что данная конструкция языка синтезируется, то подразумевается, что всё, что описано с помощью таких синтезируемых конструкций может быть транслировано в конкретную цифровую схему. Несинтезируемое подмножество языка используются только для моделирования схемы, например задания входных и анализа выходных значений и и не может быть транслировано в элементы цифровой схемы.

Переменные **wire** используются для описания только комбинационной логики (схем без памяти). Переменные **reg** могут быть использованы для описания как комбинационной, так и последовательностной (схем с памятью) логики. При этом тип **reg** может быть только у выходных портов.

В разбираемом примере мы используем только элементы комбинационной логики, поэтому можно ограничиться использованием типа **wire**. В листинге 4.1 определены линии not_a, not_b, y1, y2 в соответствии с рисунком 4.2.

Далее следует описание набора вентилей И-НЕ **nand**. В Verilog доступны для использования и другие вентили такие, как nog, og, and, not и xor. При описании вентилей первый аргумент задает связь выходного значения с локальной переменной, остальные аргументы - это входные значения.

При описании вентилей порядок их задания не имеет значения. Стоит понимать, что при таком описании описывается структура взаимосвязей компонентов, а не последовательность действий.

После того, как модуль сумматора готов, необходимо удостоверится в его работоспособности. Для проверки работоспособности создается специальный модуль тестового окружения, в котором описывается последовательность изменения входных и анализ выходных значений нашего модуля. Описание тестового окружения сумматора представлено в листинге 4.2.

Модуль тестового окружения не взаимодействует с внешним миром, поэтому не имеет портов ввода/вывода. В первой части модуля описано подключение разработанного модуля сумматора и определено подключение его портов. Порты можно подключать простым перечислением, а можно с явным указанием имен портов, как и сделано в приведенном листинге. Второй вариант подключения является более безопасным, так как изменение порядка следования портов в модуле сумматора не приведет к ошибке в соединении портов в тестовом окружении.

Переменные, задающие входные значения, имеют тип **reg**, так как мы должны иметь возможность удерживать тестовые значения на входах модуля, чтобы проанализировать его выход. Условно можно считать, что входные порты сумматора подключаются к регистрам, значения которых мы будем менять в процессе моделирования. Переменная, к которой, подключается выходной порт модуля может иметь только тип **wire**.

Последовательность изменения входных значений задается в процедурном блоке **initial**. Он вызывается в самом начале процесса моделирования и после выполнения всех действий повторно не выполняется. Если определено несколько блоков **initial**, то их выполнение начинается параллельно и невозможно предсказать точную последовательность действий, привязанных к одному и тому же моменту.

```
'timescale 1ns / 1ps

module adder_tb;

reg a_in , b_in;
wire c_out;

adder adder_1(
    .a(a_in),
    .b(b_in),
    .c(c_out)
);

integer i;
reg [1:0] test_val;
reg expected_val;

initial begin

    for(i = 0; i < 4; i = i+1) begin
        test_val = i;
        a_in = test_val[0];
        b_in = test_val[1];
        expected_val = ^test_val;

        #10

        if(c_out == expected_val) begin
            $display("The adder output is correct!!! a_in=%b, b_in=%b
                     c_out = %b", a_in, b_in, c_out);
        end else begin
            $display("The adder output is wrong!!! a_in=%b, b_in=%b
                     c_out = %b, expected %b", a_in, b_in, c_out,
                     expected_val);
        end
    end
    #10 $stop;
end

endmodule
```

Листинг 4.2: Описание модуля тестового окружения сумматора на Verilog HDL

Если процедурный блок **initial** описывает выполнение последовательности действий, то эта последовательность заключается между ключевыми словами **begin** и **end**. Это верно для всех процедурных блоков.

Продвижение модельного времени осуществляется с помощью явного указания задержек, предваряемых символом **#**. Задержки, по умолчанию, указываются в наносекундах (нс). Это может быть переопределено либо с помощью директивы **timescale**, либо с помощью настроек симулятора.

В листинге 4.2 в теле блока **for** задается последовательность изменения входных значений и анализ выходных. Результат анализа выходных значений печатается в консоль симулятора с помощью встроенной функций **display**. Встроенные функции в Verilog предваряются символом доллара **\$**. В конце блока **initial** вызывается функция **stop**, которая останавливает моделирование.

4.3 Создание проекта

Создадим проект в Vivado для схемы сумматора, описанной в предыдущих разделах. Для запуска Vivado находим соответствующий значок на рабочем столе (рисунок 4.3) и щелкаем два раза.



Рисунок 4.3

В открывшемся окружении выбираем *File->New Project* (рисунок 4.4). В открывшемся окне нажимаем *Next* (рисунок 4.5). Задаем имя проекта и нажимаем *Next* (рисунок 4.6). Выбираем *RTL Project* и нажимаем *Next* (рисунок 4.7).

В качестве микросхемы программируемой логики, под которую будет вестись разработка устройства, выбираем XC7A100TCG324-1 и нажимаем *Next* (рисунок 4.8). Данная микросхема установлена в отладочной плате Nexys A7 (<https://store.digilentinc.com/nexys-a7-fpga-trainer-board-recommended-for-ece-curriculum/>), которая будет в дальнейшем использоваться в работах.

В последнем диалоговом окне нажимаем кнопку *Finish* (рисунок 4.9). После создания проекта откроется окружение, показанное на рисунке 4.10.

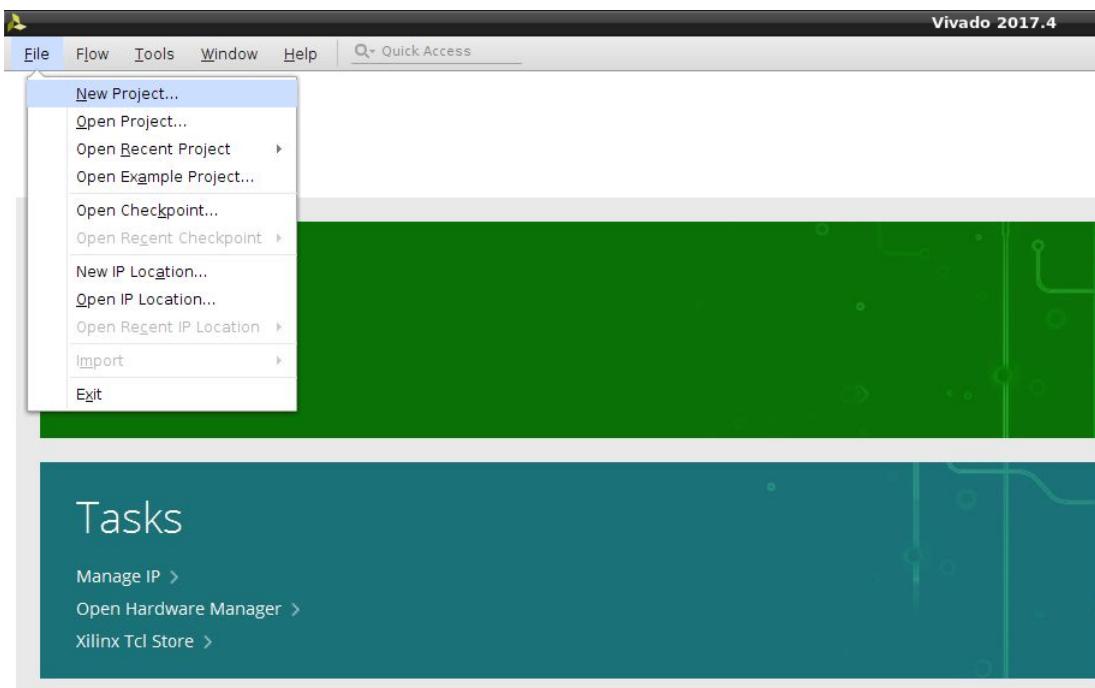


Рисунок 4.4



Рисунок 4.5

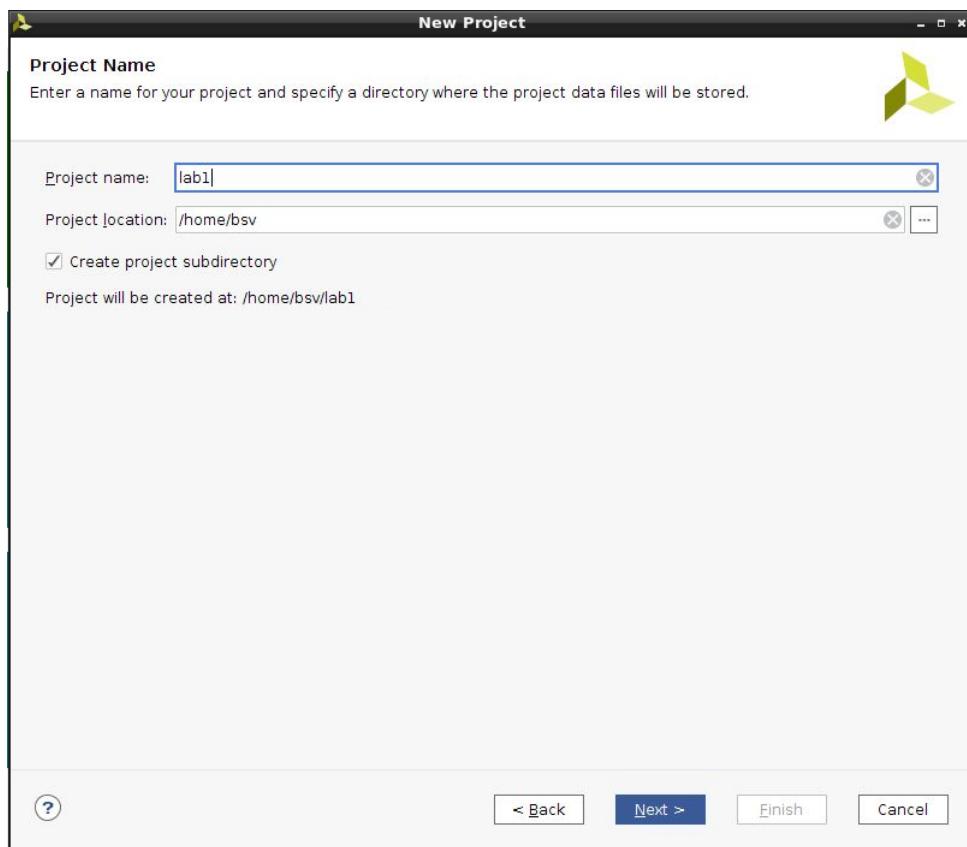


Рисунок 4.6

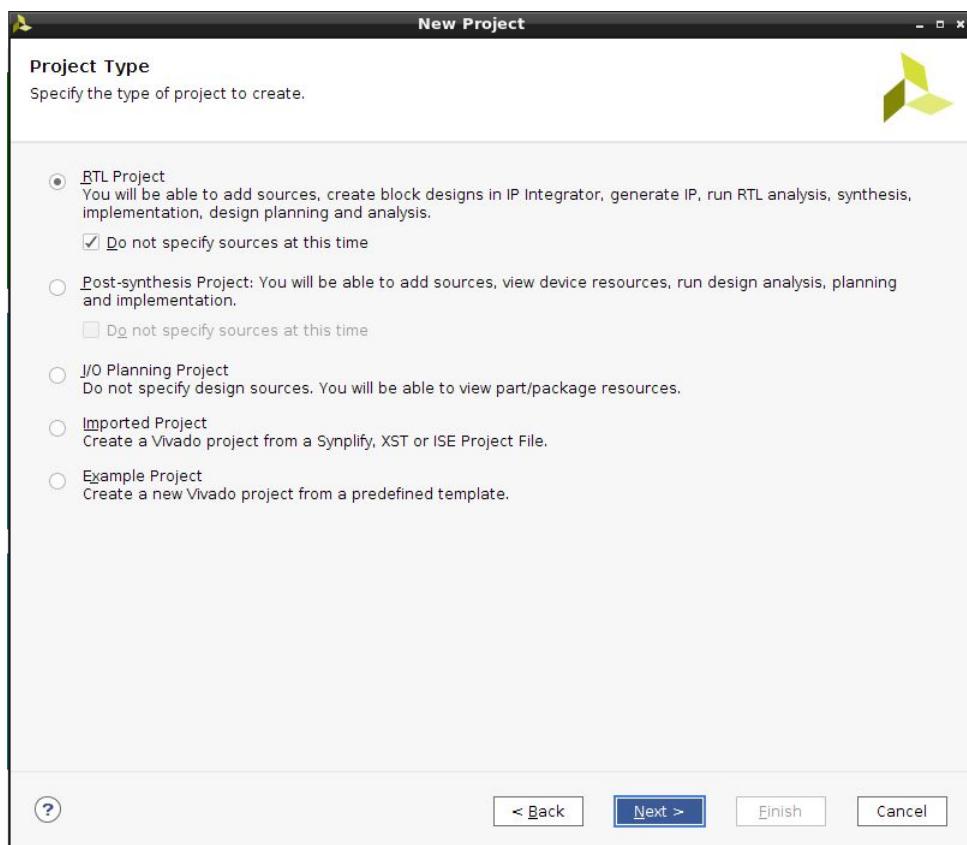


Рисунок 4.7

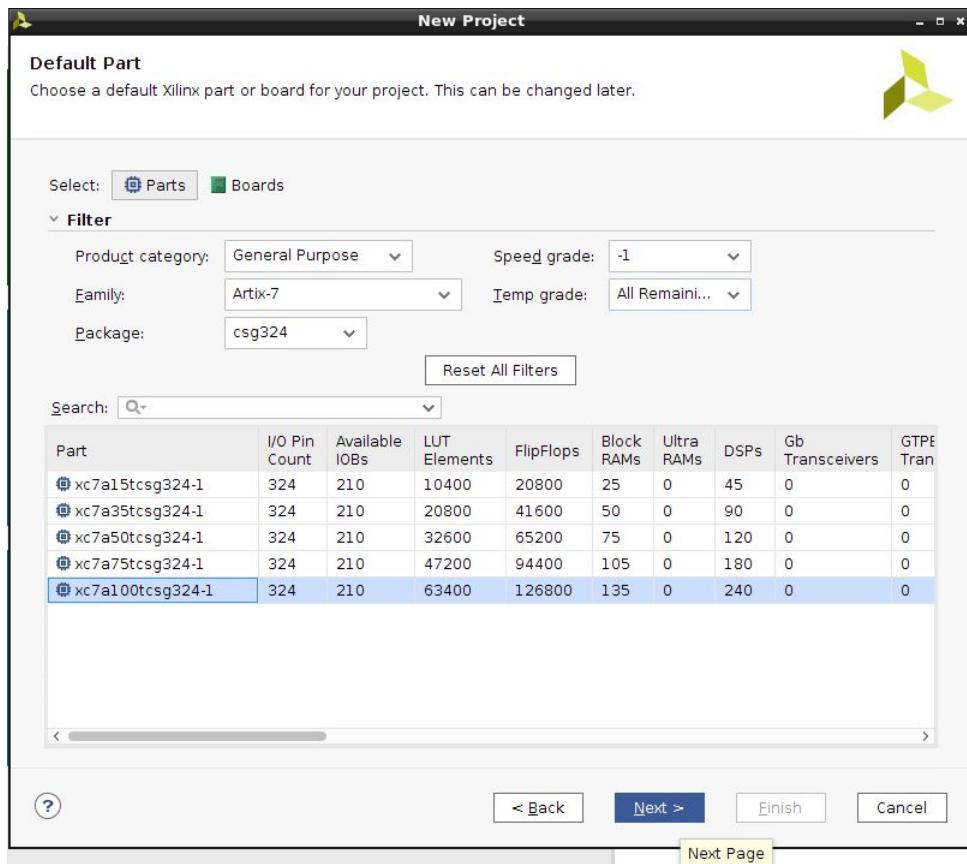


Рисунок 4.8

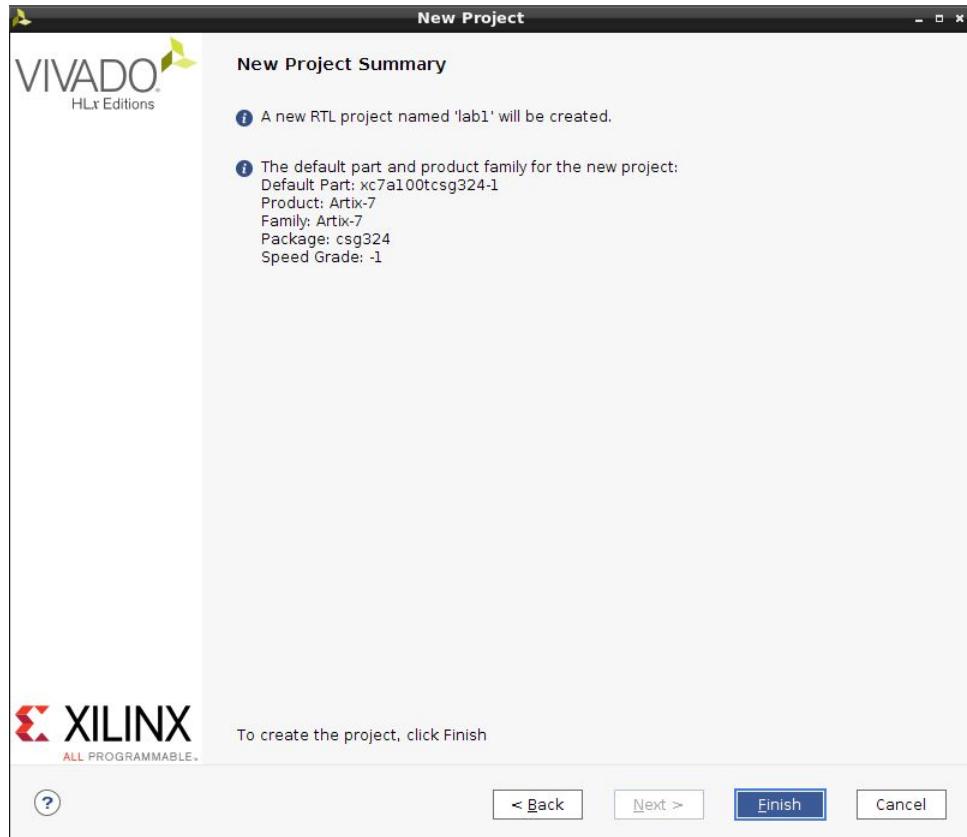


Рисунок 4.9

Для добавления новых исходных файлов нажимаем на *Add Sources* в левом верхнем углу рабочего окружения (подчеркнуто красным на рисунке 4.10).

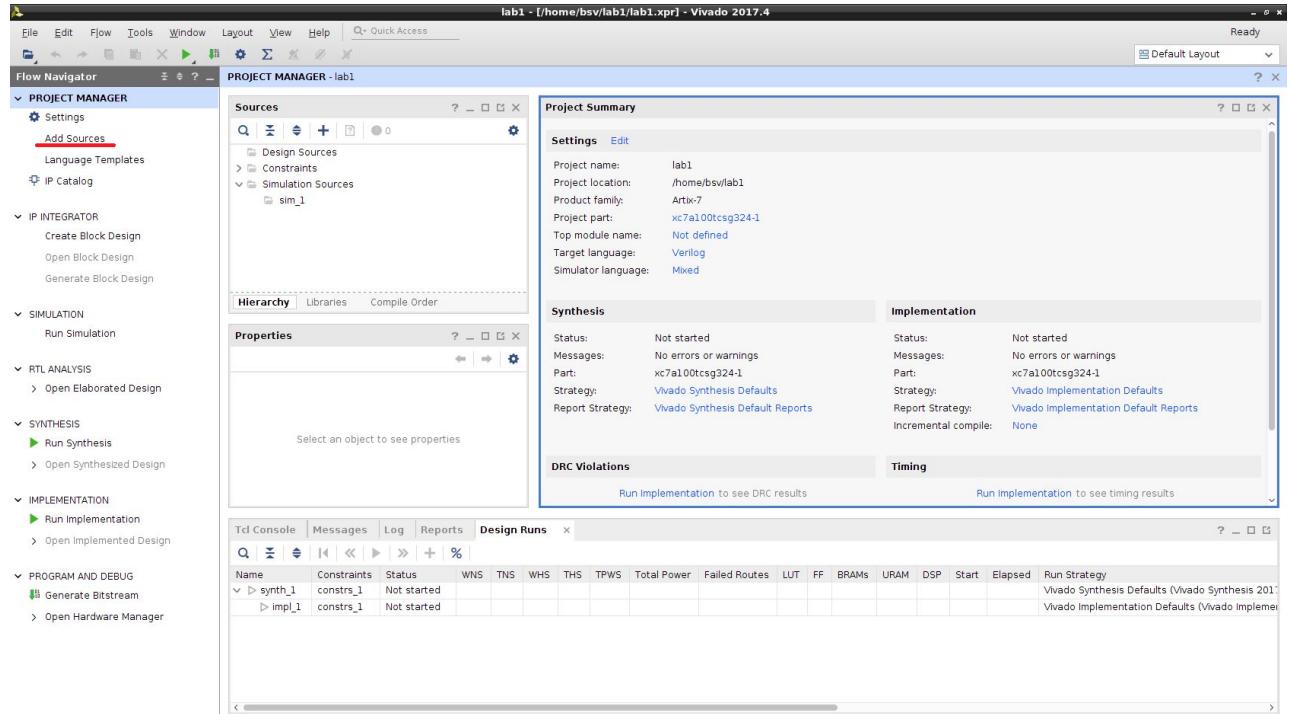


Рисунок 4.10

В открывшемся окне выбираем *Add or create design sources* и нажимаем *Next* (рисунок 4.11).

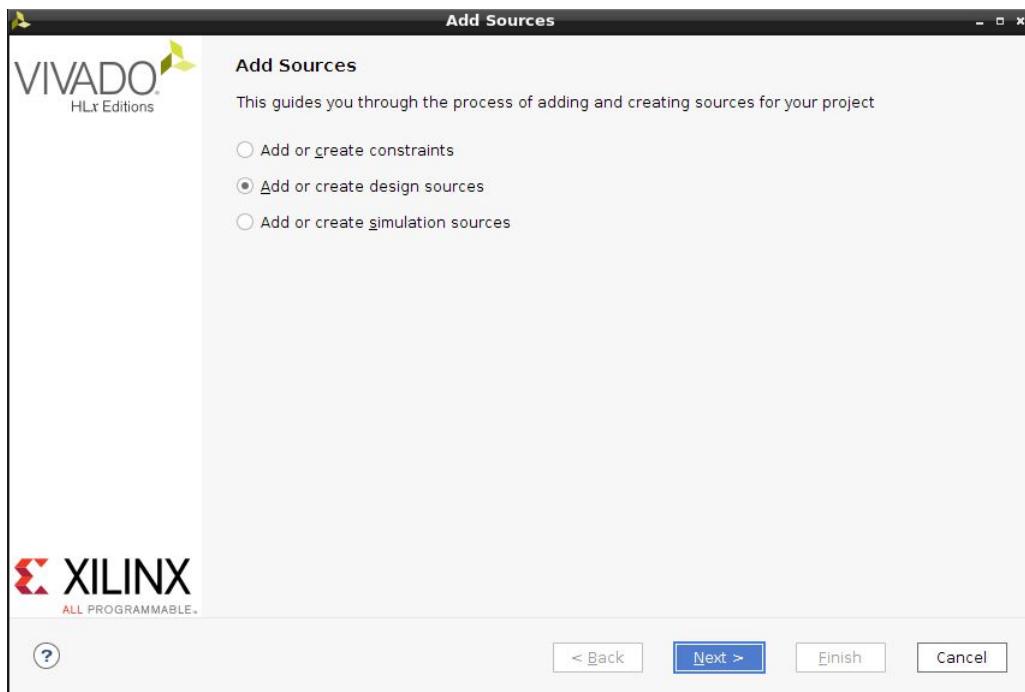


Рисунок 4.11

В открывшемся окне выбираем *Create File* и задаем имя файлу, в котором будем описывать функциональность сумматора (рисунок 4.12).

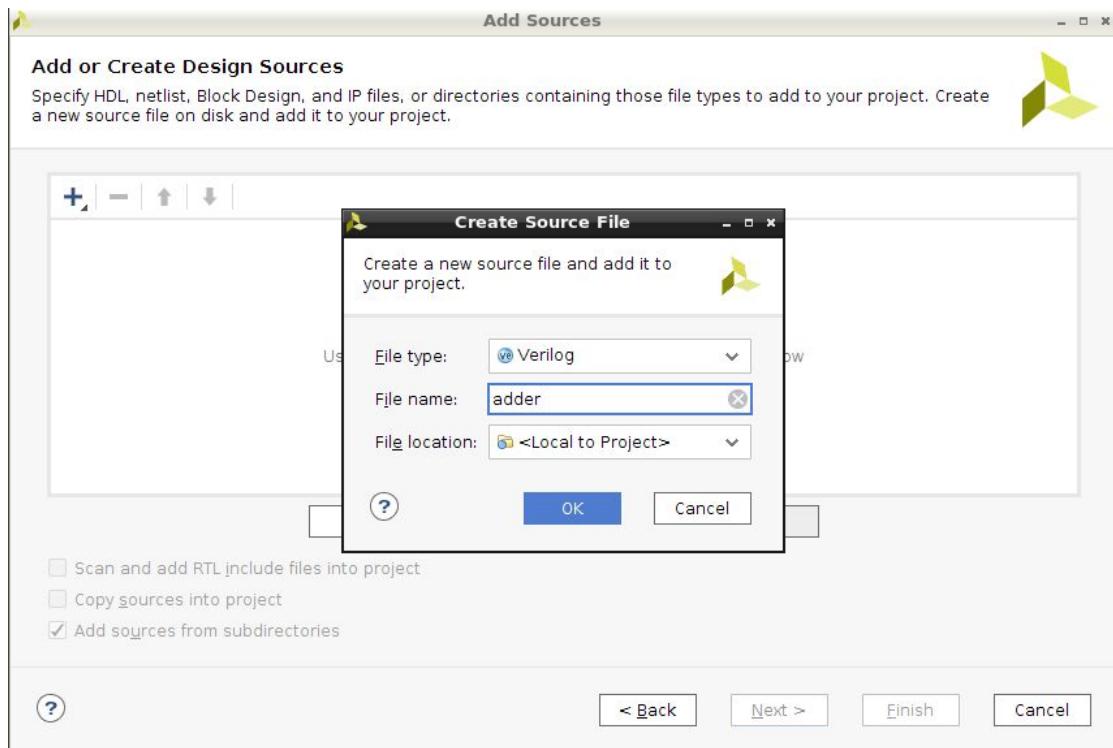


Рисунок 4.12

Нажимаем *Next* и переходим к описанию портов модуля (рисунок 4.13). Данный шаг можно пропустить, нажав *Ok*, и определить интерфейс модуля явно в файле модуля.

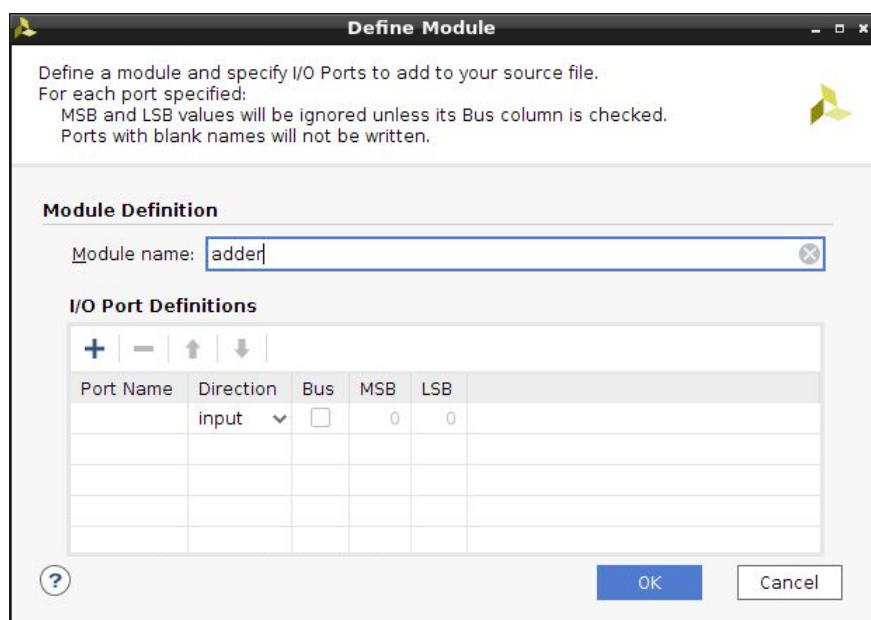


Рисунок 4.13

После создания нового файла он откроется для редактирования (рисунок 4.14).

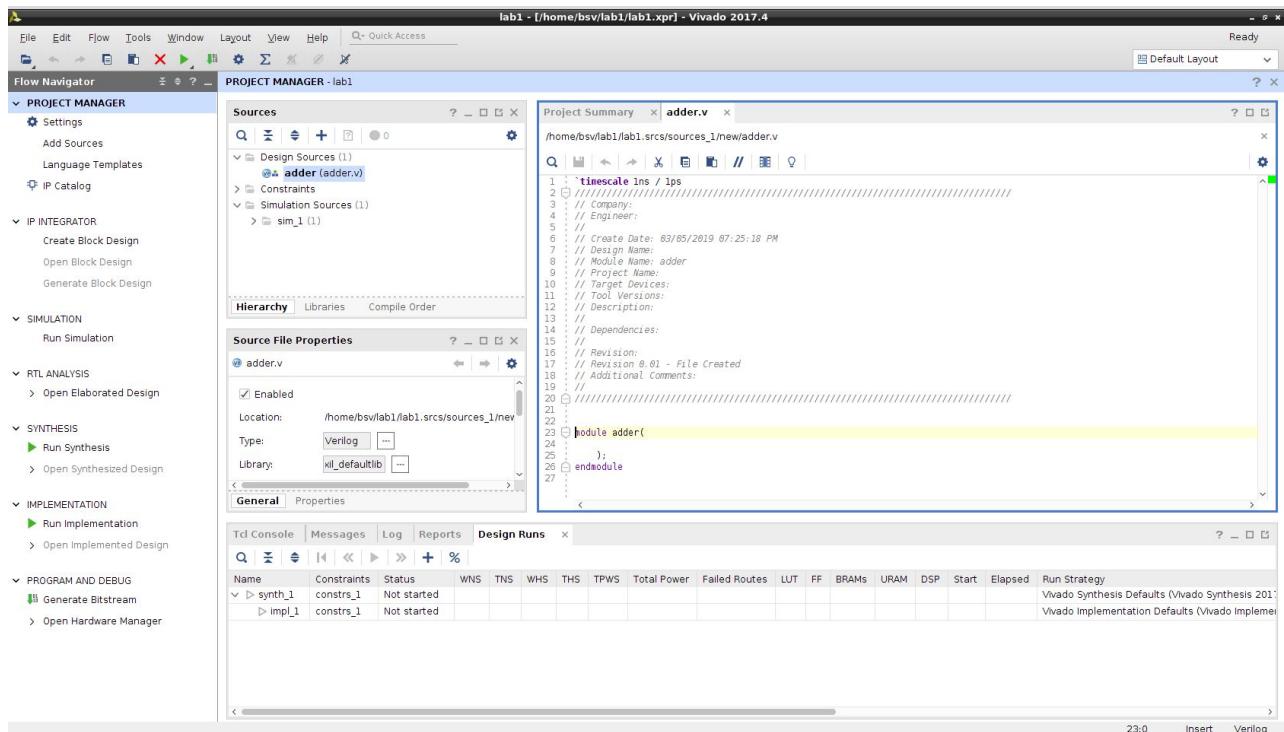


Рисунок 4.14

Добавим описание модуля (рисунок 4.15)

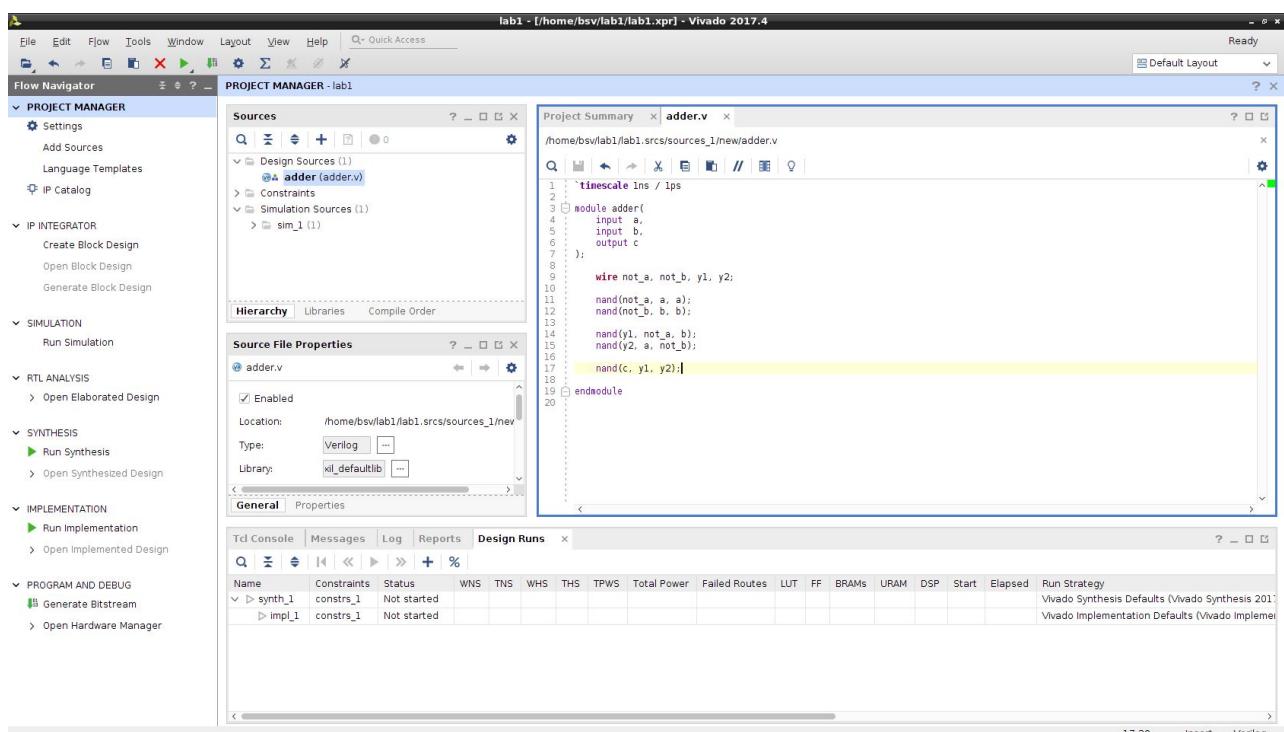


Рисунок 4.15

Теперь у нас есть проект и описание модуля сумматора. Далее необходимо создать тестовое окружение для разработанного модуля.

4.4 Моделирование схемы

Для создания файла тестового окружения вновь нажимаем на *Add Sources* в левом верхнем углу рабочего окружения (подчеркнуто красным на рисунке 4.10). Теперь выбираем *Add or create simulation sources* (рисунок 4.16).

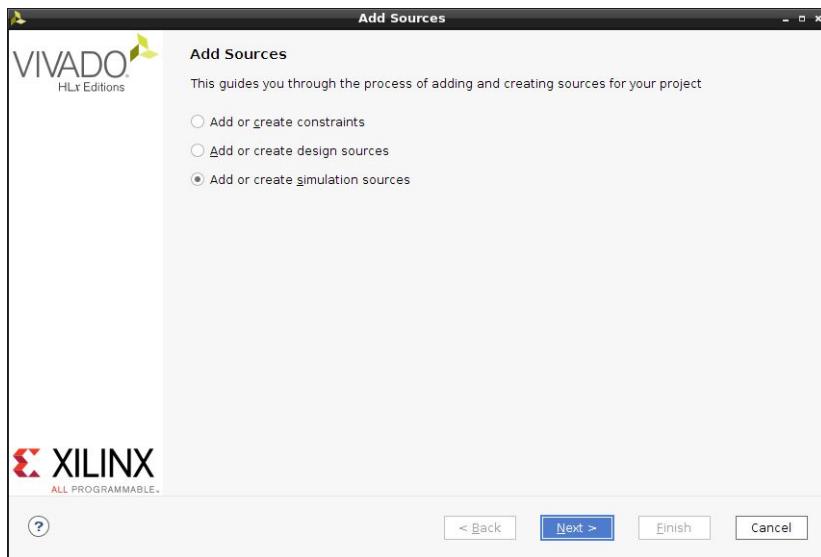


Рисунок 4.16

В открывшемся окне выбираем *Create File* и задаем имя файлу, в котором будем описывать функциональность тестового окружения сумматора (рисунок 4.17).

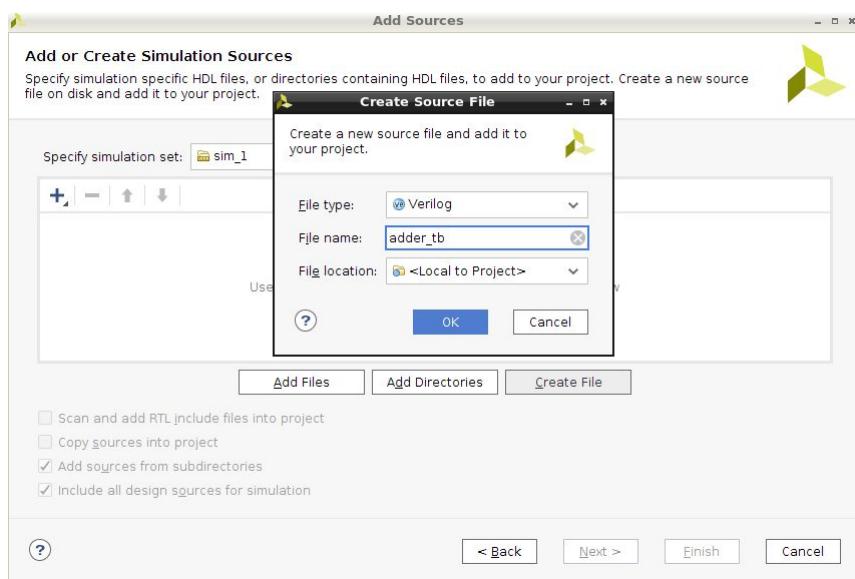


Рисунок 4.17

После создания файла тестового окружения, наполняем его (рисунок 4.18).

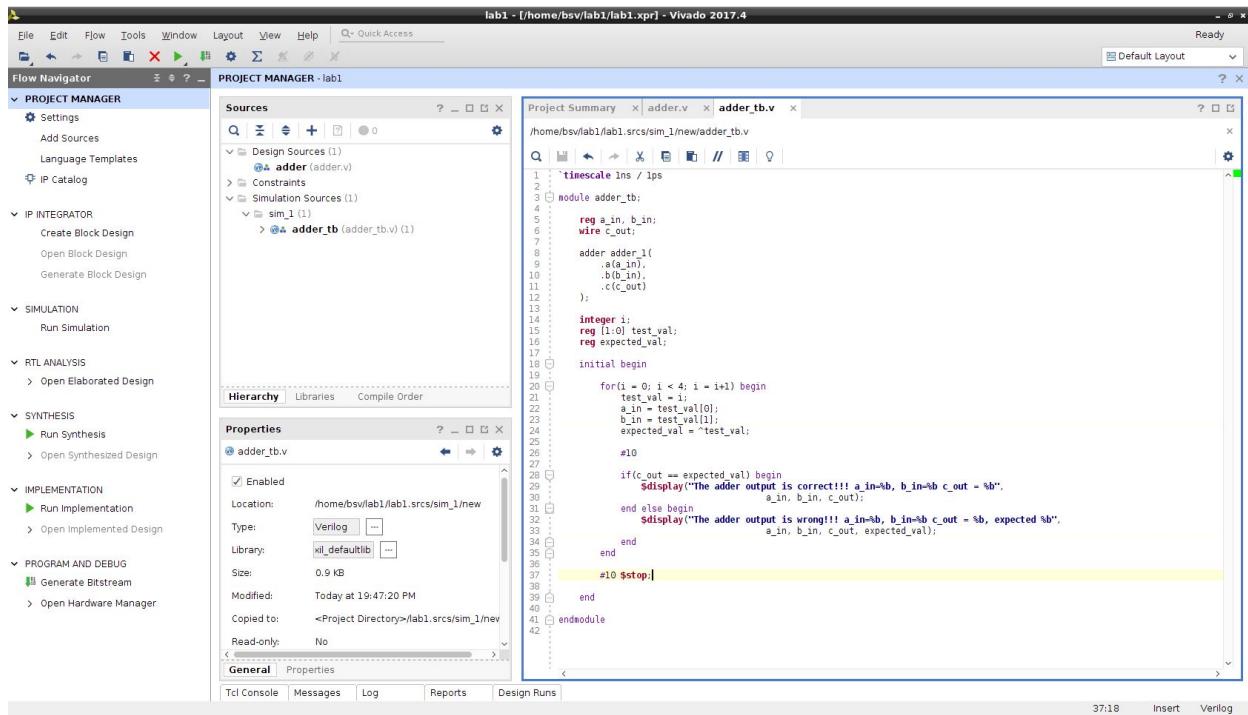


Рисунок 4.18

Для запуска тестового окружения на выполнение необходимо выбрать в левой панели выбрать *Run Simulation* и в открывшемся выпадающем меню *Run Behavioural Simulation* (рисунок 4.19).

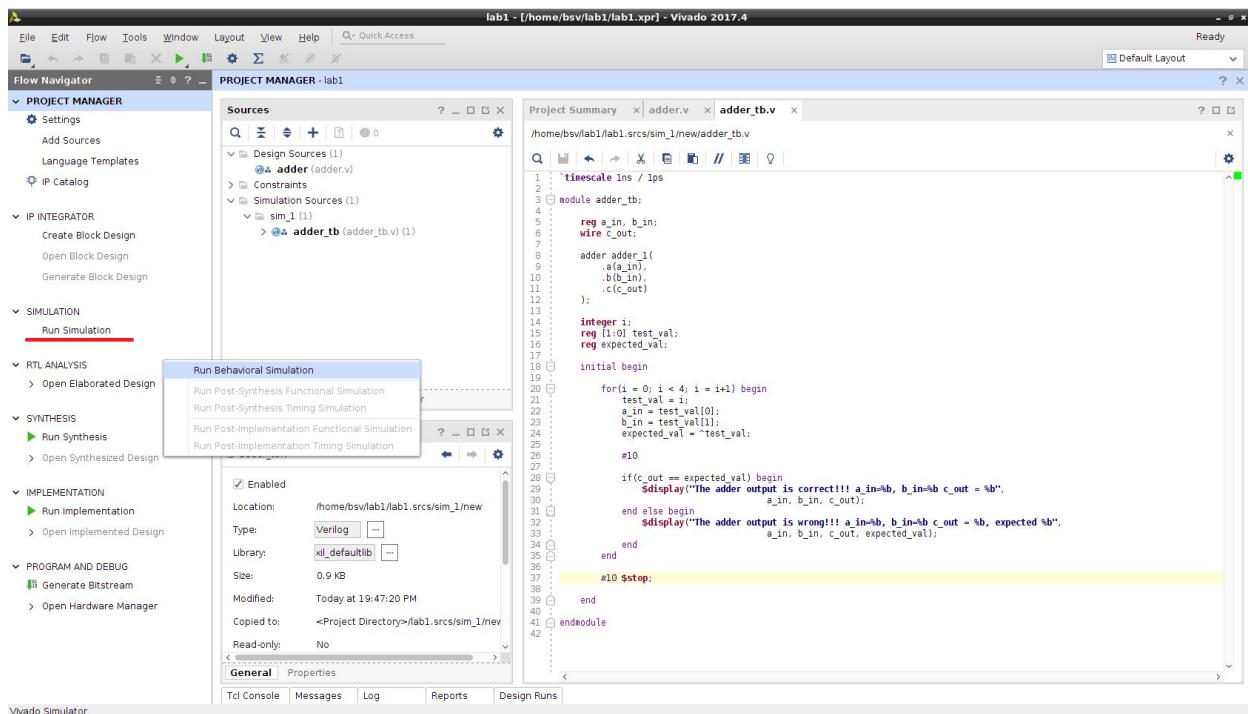


Рисунок 4.19

После запуска процесс моделирования сразу остановится на моменте вызова функции `$stop`. Для просмотра результатов моделирования необходимо переключиться на окно с названием, начинающегося с *Untitled* (рисунок 4.20).

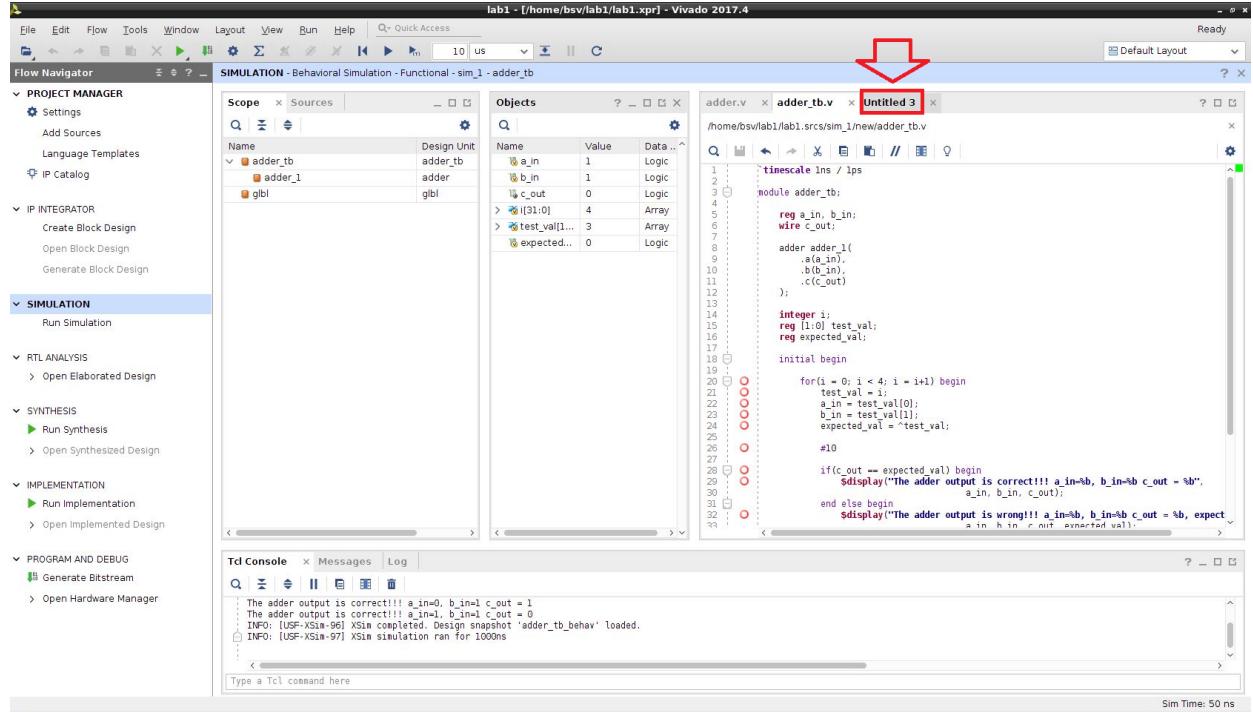


Рисунок 4.20

По умолчанию временная диаграмма имеет масштаб, который не позволяет просмотреть весь процесс моделирования. Для того, чтобы увидеть полную диаграмму необходимо нажать на кнопку *Zoom Fit* (рисунок 4.21).

На рисунке 4.22 представлена временная диаграмма всего процесса моделирования. Можно видеть все комбинации входных сигналов, значение выходного сигнала, а также вывод результатов анализа выходного значения схемы в консоли симулятора.

Стоит отметить, что при проведении поведенческого моделирования схемы, описанного выше, никак не учитываются задержки распространения сигналов через вентили. Задержки, которые явно не определены, не моделируются.

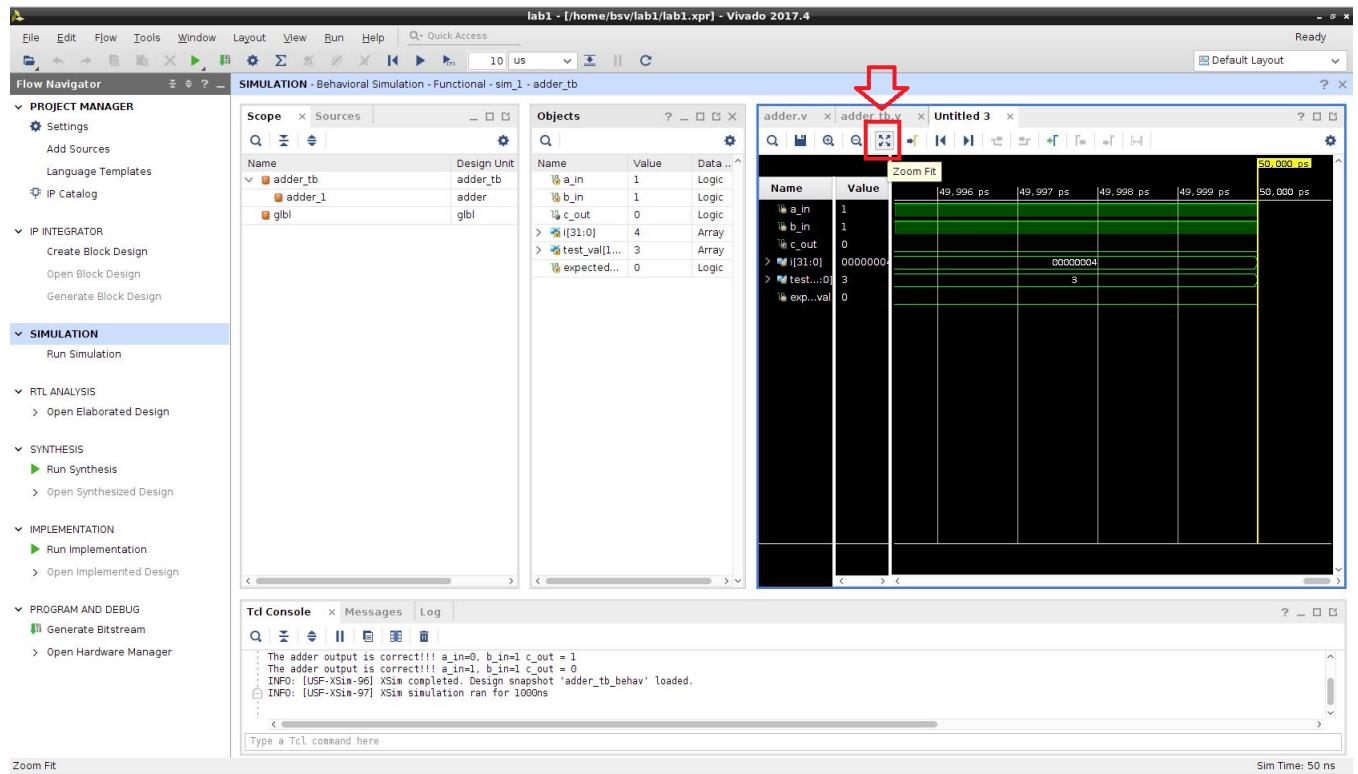


Рисунок 4.21

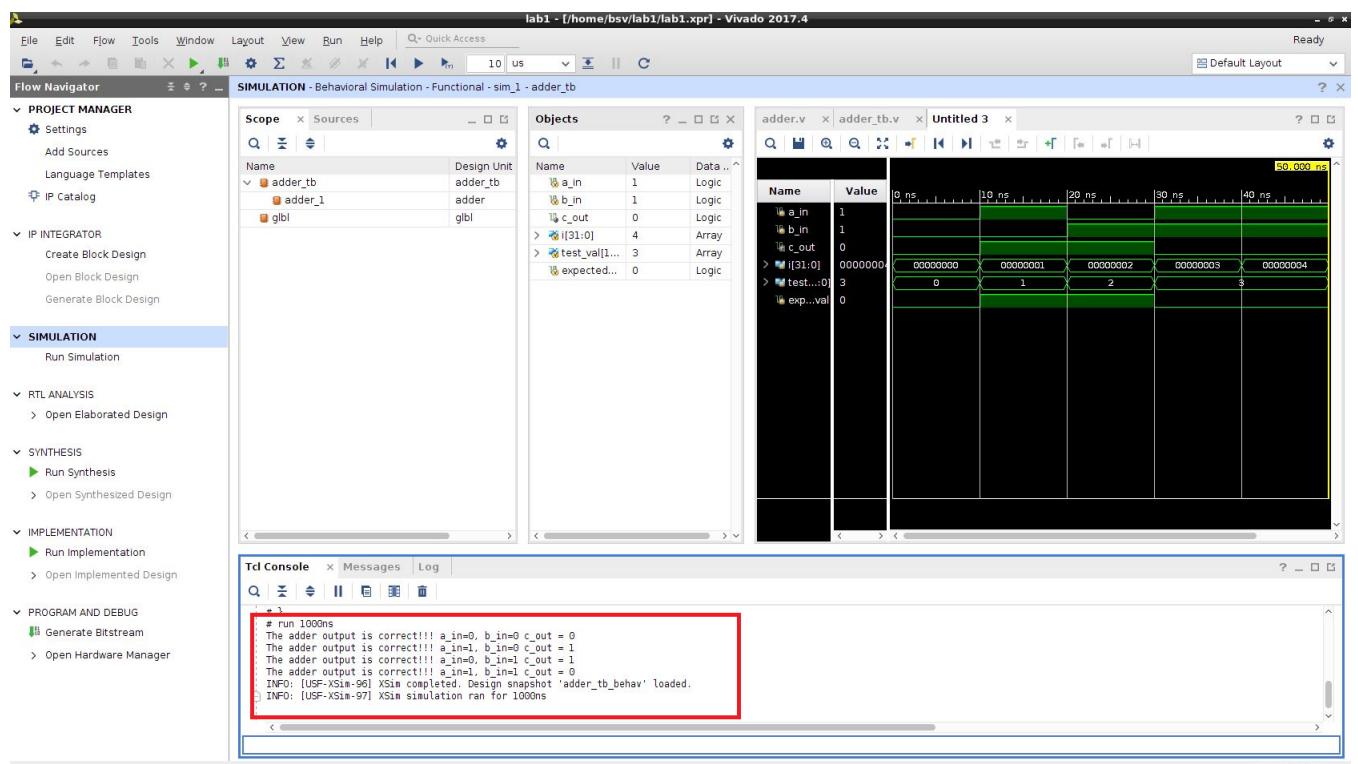


Рисунок 4.22

Рекомендуемая литература

1. Дэвид М. Харрис, Сара Л. Харрис. Цифровая схемотехника и архитектура компьютера. 2-е изд.
2. Жан М. Рабай, Ананта Чандракасан, Боривож Николич. Цифровые интегральные схемы. Методология проектирования. – 2-е изд. – М.:«Вильямс», 2007.
3. David Harris, Sarah Harris. Digital Design and Computer Architecture, 2d edition.
4. N. H.E. Weste, D.M. Harris. CMOS VLSI design: A circuits and systems perspective.
5. Угрюмов Е.П. Цифровая схемотехника. Уч. пособие для ВУЗов. 2-ое изд. – СПб.: БХВ – Петербург, 2007, 800 с.
6. Хоровиц П., Хилл У., Искусство схемотехники./ Пер. с англ. 6-е изд. – М.: Мир, 2003. – 704 с.
7. Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. Digital Integrated Circuits Prentice Hall; Prentice Hall 2 edition (January 3, 2003)
8. Baker, R. Jacob. CMOS: Circuit Design, Layout, and Simulation, Third Edition. Wiley-IEEE, 2010. <http://CMOSedu.com>
9. Weste, Neil H. E. and Harris, David M. CMOS VLSI Design: A Circuits and Systems Perspective, Fourth Edition. Boston: Pearson/Addison-Wesley, 2010.
10. Точки, Рональд, Дж., Уидмер, Нил, С. Цифровые системы. Теория и практика. – 8-е изд.. – М.: «Вильямс», 2004.
11. Сохор Ю.Н. Моделирование устройств в пакете LTspice/SwCAD. Учебно-методическое пособие. Псковск. гос. политехн. ин-т. – Псков: Издательство ППИ, 2008. – 165 с.