Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Воронежский государственный лесотехнический университет

имени Г.Ф. Морозова»

Базовая кафедра технического и программного обеспечения вычислительных и информационных систем

(название кафедры)

**Пояснительная записка**

Основы языка VHDL

(тема)

Вариант 3

09.03.02 Информационные системы и технологии

(код и наименование направления подготовки)

По дисциплине «Программирование на языке VHDL и проектирование микроэлектронных устройств»

|  |  |  |
| --- | --- | --- |
| Студент группы ИС2-191-ОБ  (номер группы)  Руководитель, доцент, к.ф-м.н  (ученая степень, ученое звание) | \_\_\_\_\_\_\_\_\_\_\_  (подпись)  \_\_\_\_\_\_\_\_\_\_  (подпись) | Д.Р. Брославский  (инициалы и фамилия)  А.А. Караваев  (инициалы и фамилия) |

Воронеж 2022

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc124888843)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc124888844)

[1.1 Архитектура языка VHDL 4](#_Toc124888845)

[1.2 Типы данных 8](#_Toc124888846)

[1.3 Описание интерфейса устройства 11](#_Toc124888847)

[1.4 Операции языка VHDL 13](#_Toc124888848)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 16](#_Toc124888849)

[ЗАКЛЮЧЕНИЕ 17](#_Toc124888850)

[СПИСОК ЛИТЕРАТУРЫ 18](#_Toc124888851)

# ВВЕДЕНИЕ

Язык VHDL служит для описания модели цифрового устройства (прибора, системы). Описание на языке VHDL определяет внешние связи устройства (“вид снаружи” или интерфейс) и один или несколько “видов изнутри” (рисунок 1.1). Вид снаружи задает интерфейс устройства, набор сигналов, которыми устройство обменивается с внешним миром. Этот вид описывает абстрактное представление устройства “в целом” и обозначается английским термином entity, что в дословном переводе означает «сущность» и наиболее точно отражает смысл представления. Однако в литературе термин «сущность» не нашел широкого распространения, для обозначения внешнего описания объекта используются термины «интерфейс объекта», «декларативная часть» и другие. В настоящем пособии будет использоваться термин «интерфейс объекта» или просто «интерфейс».



Рисунок 1.1 - Цифровое устройство и его модель

Вид изнутри определяет функциональные возможности устройства или его структуру. Внутреннее строение объекта определяет архитектура.

Как и в языках программирования, язык VHDL имеет свои правила, в том числе правила описания имен переменных, объектов, типов данных и других параметров. Основные правила языка VHDL описаны в последующих разделах.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# 1.1 Архитектура языка VHDL

Архитектура моделирует взгляд на устройство “изнутри”. Устройство может рассматриваться под разными углами и описываться по-разному. Оно может предстать либо как композиция более простых модулей (структурный стиль моделирования), либо как набор параллельно исполняемых алгоритмов, либо описано как процесс последовательного исполнения операций (стиль “поведенческий”), либо может быть раскрыто сочетанием указанных стилей.

С одним интерфейсом типа entity может быть связано несколько архитектур. Рассмотрение архитектурных стилей начнем с поведенческого.

Поведенческое описание

Функционирование устройства рассматривается как процесс последовательного вычисления выражений, входящих в процесс. Толчком к запуску процесса является изменение какого-либо сигнала, из входящих в список “воспринимаемых” сигналов. Этот список внешне очень похож на список параметров в других языках высокого уровня. Рассмотрим поведенческую модель декодера dec2х4. Декларация интерфейса его уже приводилась, а теперь она будет пополнена архитектурой поведенческого типа.

Архитектура имеет свое имя dec\_seq. О принадлежности архитектуры к описанию интерфейса dec2х4 говорит выражение: dec\_seq of dec2x4. Список сигналов, которыми запускается процесс, дан в скобках: process (a, b, enable). До начала процесса, если необходимо, декларируются переменные variable abar, bbar: BIT. Зона действия их ограничена данным процессом (от begin до end process). Предположим, что в момент T произошло изменение сигнала а. Новое значение переменной abar (выражение 1) вычисляется и присваивается без задержки – мгновенно (в качестве знака присваивания в данном случае используется =). Без задержки будут вычислены выражения 2 и 3.

Далее в ходе процесса одно за другим вычисляются выражения от 4 до 7, а новые значения сигналам z(0), z(1), z(2), z(3) будут присвоены (в качестве символа оператора присвоения используется <= ) лишь через некоторую, пусть даже фиктивную (бесконечно малую), задержку Δ. Это существенное отличие процедур присвоения для переменных и для сигналов.

Рассмотрим пример мультиплексора 4х1 (рисунок 1.2). Четыре входных сигнала A, B, C и D выводятся через один порт Z. Коммутация осуществляется сигналом CTRL.



Рисунок 1.2 - Мультиплексор 4х1.

Архитектура получила название MUX\_BEHAVIOR, процесс имеет ярлык PMUX. Переменной TEMP в зависимости от значения сигнала CTRL присваивается значение одного из входных портов: A, когда CTRL имеет значение “00”; B, когда CTRL имеет значение “01”; A, когда CTRL имеет значение “10”; D, когда CTRL имеет значение “11”.

Для описания синхросигналов обычно используется атрибут EVENT. Он истинен, когда происходит “событие” - изменение сигнала. Например, поведение D-триггера, показанного на рисунке 1.3, можно описать так:



Рисунок 1.3 - D-триггер

Потоковое описание (dataflow)

Dataflow - означает поток данных. Описание в стиле dataflow использует в архитектуре параллельно вычисляемые выражения. Количество параллельно вычисляемых выражений может быть любым. Так как вычисления происходят параллельно, то порядок записи выражений не имеет значения. Толчком к началу вычислений является изменение любого из сигналов, входящих в выражения. Рассмотрим модель одноразрядного сумматора, показанного на рисунке 1.4.



Рисунок 1.4 - Одноразрядный сумматор

Для описания потока данных здесь используются два выражения. Всегда, когда происходят изменения сигналов a, b или cin, оба выражения вычисляются и сигналам sum и cout присваиваются новые значения через 15ns и 10ns, соответственно. Знак <= оператор присваивания. Выражение after 15ns означает задержку, вносимую логикой, реализующей вычисление выражения для sum. Если задержка не введена явно, то предполагается наличие бесконечно малой задержки Δ. Такой прием позволяет упорядочить события модели dataflow. Рассмотрим схему рисунок 1.5 и ее модель.



Рисунок 1.5 - Цепочка инверторов

В выражениях 1, 2 и 3, присваивающих значения переменным z, c, b, подразумевается задержка Δ. Например, в момент T происходит изменение сигнала a. Запускается процесс вычисления выражения 3 и через задержку Δ переменной b будет присвоено новое значение (в момент T+Δ). В свою очередь изменение сигнала b запустит процесс вычисления выражения 2 и значение сигнала с будет обновлено в момент T+2Δ. Изменение сигнала с приведет к запуску процесса вычисления выражения 1 и значение сигнала z будет изменено в момент T+3Δ. Временные соотношения в схеме иллюстрирует рисунок 1.6.



Рисунок 1.6 - Временные соотношения в цепочке инверторов.

Структурное описание.

Структурное описание интерпретирует устройство как набор компонентов, связанных между собой сигналами. Грубо говоря - это таблица соединений (netlist). Рассмотрим простую схему управления (рисунок 1.7) и ее структурную модель.



Рисунок 1.7 - Устройство местного управления

# 1.2 Типы данных

Каждый объект в языке VHDL может хранить значения, относящиеся к определенному набору. Это множество значений декларируется с помощью объявления типа (type declaration). Тип – это имя, которое связывается с определенным набором значений и набором операций. Некоторые типы предопределены языком VHDL. Например, BOOLEAN имеет набор значений FALSE, TRUE и набор операторов: and, or, nor, nand, not. В языке имеется возможность создавать новые типы с использованием деклараций и задания набора операций.

Все возможные типы в VHDL распадаются на четыре больших категории:

- scalar (скалярные),

- composite (композитные) – они состоят из элементов одного типа (массивы) или различного типа (записи),

- access type (типы доступа) – обеспечивают доступ к данному типу через указатели,

- file types (тип – файл) – обеспечивает доступ к объектам, содержащим последовательности значений данного типа.

В свою очередь скалярные типы подразделяются на четыре вида:

-enumeration (перечислимый тип),

-integer (целый тип),

-physical(физический тип),

-floating point (тип “с плавающей запятой”).

Перечислимый тип

В декларации определяется набор определенных пользователем значений, например:

type MVL is ('U','0','1','Z);

type MICRO\_OP is (LOAD, STORE, ADD, SUB, MUL, DIV);

MVL – перечислимый тип с упорядоченным набором значений: 'U', '0', '1', и 'Z'. MICRO\_OP имеет набор значений: LOAD, STORE, ADD, SUB, MUL, DIV. Порядок записи значений в декларации определяет лексику, т.е. значение справа всегда больше значения слева: STORE<DIV is true, SUB > MUL is false. Значения в перечислимых типах имеют позиционный номер. Позиционный номер самого левого элемента 0. Значения в перечислимых типах еще называют enumeration literals (литералы перечислимого типа). Например, в декларации:

type CAR\_STATE is (STOP, SLOW, MEDIUM, FAST);

литералами являются STOP, SLOW, MEDIUM, FAST и только они могут присваиваться переменной CAR\_STATE.

Целый тип

Integer – целое, задает тип, набор значений которого находится в заданном целочисленном диапазоне, например:

type INDEX is range 0 to 15;

type WORD\_LENGTH is range 31 downto 0;

subtype DATA\_WORD is WORD\_LENGTH range 15 downto 0;

type MY\_WORD is range 4 to 6;

INDEX – это переменная целочисленного типа, набор значений которой размещен в диапазоне целых от 0 до 15. DATA\_WORD – подтип WORLD\_LENGTH в диапазоне от 15 до 0. В отличие от перечислимых в целочисленных типах позиционный номер равен величине значения, например, для значения 31 переменной WORD\_LENGTH позиция равна 31.

Тип “с плавающей запятой”

Тип floating point обладает набором значений в заданном диапазоне вещественных чисел, например:

type TTL\_VOLTAGE is range 1.4 to 5.5

type REAL\_DATA is range 0.0 to 31.9;

Литералы типа floating point отличаются от целочисленных присутствием точки ( . ). В результате 0 – это целочисленный литерал, а 0.0 – это литерал типа с плавающей запятой.

Физический тип

Physical type хранит значения, которые представляют собой результаты измерений физических величин: времени, длины, напряжения, тока и т.п. Значения этого типа выражаются целыми, умноженными на базовую единицу, например:

type CURRENT is range 0 to 1 E9

units

nA; -- (base unit) nano-ampere

uA = 1000 nA; -- micro-ampere

mA = 1000 μA; --milli-ampere

Amp = 1000 mA; -- ampere

end units;

subtype FILTER\_CURRENT is CURRENT range 10 μA to 5 mA;

Здесь CURRENT определен как физический тип, имеющий значения в диапазоне от 0 nA до 10^9 nA. Базовой единицей является наноАмпер, а все остальные являются производными. Позиционный номер значения равен числу базовых единиц, представленных данным значением, например, 2 μA имеют позиционный номер 2000, в то время как 100 nA занимают позицию 100. Физический литерал записывается как целое, за которым следует название единицы измерения (пробел обязателен).

Тип “массив”

Объект типа array состоит из элементов одного типа. Ниже даны примеры массивов:

type ADDRESS\_WORD is array (0 to 63) of BIT;

type DATA\_WORD is array (7 downto 0) of MVL;

type ROM is array (0 to 125) of DATA\_WORD;

ADDRESS\_WORD – одноразмерный массив из 64 элементов типа BIT. DATA\_WORD – одноразмерный массив из 8 элементов типа MVL. ROM – массив из 126 элементов типа DATA\_WORD, т.е. в данном случае имеем дело с массивом массивов.

Доступ к элементам массива осуществляется с помощью индексов, например, ADDRESS\_WORD(26) ссылается на 27-ой элемент массива ADDRESS\_WORD.

# 1.3 Описание интерфейса устройства

В начале главы уже говорилось, что на моделируемое устройство в первую очередь следует посмотреть со стороны (“вид снаружи”) и для этого “черного ящика” определить интерфейс с внешним миром. Такая задача решается в VHDL декларацией интерфейса entity. Она задает имя описания, имена интерфейсных портов, направление передачи, тип портов. Порт – это сигнальная линия (шина), с помощью которой устройство (модель) взаимодействуют с окружающей средой. Например, для полусумматора (рисунок 1.8) декларация будет выглядеть следующим образом:



Рисунок 1.8 - Полусумматор

entity half\_adder is

port(a, b: in BIT; sum, cur: out BIT);

end half\_adder;

Здесь описание с именем half\_adder имеет два входных порта, a и b (in – означает входной порт), и два выходных порта, sum и carry (out – означает выходной порт). Тип портов определен как BIT – означает, что сигналы на линиях портов могут принимать значения: ‘0’ или ‘1’.

Другим примером может послужить дешифратор, показанный на рис. 1.9.

entity dec2x4 is

port(a, b, enable: in BIT; z: out BIT\_VECTOR (0 to 3));

end dec2x4;

Здесь описание с именем dec2x4 имеет три входных порта и четыре выходных. Выходные порты описаны как массив. BIT\_VECTOR – это одномерный массив, диапазон задается параметром Z (0 to 3).



Рисунок 1.9 - Декодер 2х4

Каждый интерфейсный порт может функционировать в следующих режимах:

in – значение порта только считывается для использования внутри модели,

out – значение порта может только обновляться моделью, но не считывается,

inout – двунаправленный порт, значение считывается и обновляется моделью,

buffer – буферный порт, значение считывается и обновляется моделью, но источником сигнала может быть либо буфер, либо одиночный источник.

# 1.4 Операции языка VHDL

В языке VHDL имеются операции следующих категорий:

1. Логические операции

2. Операции отношений

3. Операции сложения/вычитания

4. Операции умножения/деления

5. Прочие

Приоритет операций возрастает от категории 1 к категории 5. Операции одной категории имеют одинаковый приоритет и выполняются в последовательности: слева направо. Скобки используются для изменения последовательности исполнения.

Операции логические

Существует шесть логических операций: and, or, nand, nor, xor, not.

Операции применимы к типам BIT, BOOLEAN, к одноразмерным массивам BIT и BOOLEAN. При выполнении битовые значения ‘0’ и ‘1’ интерпретируются как булевские FALSE и TRUE. Результат имеет тот же тип, что и операнды. Операция not – унарная операция, она имеет приоритет категории 5.

Операции отношений

В языке VHDL имеются следующие операции отношений:

=, /=, <, <=, >, >=

Результатом выполнения любой операции отношений является булевское выражение BOOLEAN. Операции равенства ( = ) и неравенства ( /= ) допустимы со всеми типами, за исключением типа “файл”. Остальные четыре операции допустимы над скалярными типами (целочисленными, перечислимыми) или над массивами дискретного типа. Когда операндами являются массивы, то сравнение выполняется слева направо по одному элементу, например при следующем сравнении:

BIT\_VECTOR'('0', '1', '1') < BIT\_VECTOR'('1', '0', '1')

получен результат TRUE, т.к. первый элемент вектора слева меньше первого элемента вектора справа. Другой пример, если декларирован тип:

type MVL is ('U', '0', '1', 'Z' );

то результат сравнения:

MVL'( 'U' ) < MVL'( 'Z' )

будет TRUE, т.к. ‘U’ находится левее ‘Z’.

Операции суммирования, вычитания, конкатенации

Операции имеют обозначения:

+, -, &.

Операнды, участвующие в операциях сложения ( + ) и вычитания ( - ) должны быть одного числового типа, результат оказывается того же типа. Операндами в конкатенации ( & ) могут быть или отдельные элементы или одномерные массивы. Результат выдается в виде массива, например при выполнении конкатенации:

‘0’ & ‘1’

образуется массив символов “01”, или еще пример:

‘C’ & ‘A’ & ‘T’

дает “CAT”.

Операции умножения, деления

К этой группе оператций относятся: \*, /, mod, rem.

Операнды умножения ( \* ) и деления ( / ) должны быть одновременно либо целочисленного типа (integer) либо типа с плавающей запятой (floating point). Результат всегда того типа, что и операнды. Операция умножения может иметь один операнд физического типа, а другой – либо целочисленного, либо натурального типа. Результат выдается в виде физического типа.

В операции деления допустимо делить объект физического типа на целочисленный или натуральный. Результат - всегда физического типа. Деление физического типа на физический дает целочисленный результат.

Операции остатка ( rem ) и “деления по модулю” ( mod ) в качестве операндов могут иметь целочисленные типы и результат - целочисленного типа. Результат rem имеет знак первого операнда и определяется следующим образом:

A rem B = A - ( A / B ) \* B

Результат mod имеет знак второго операнда и определяется следующим образом:

A mod B = A – B \* N,

где N – некоторое целое.

Прочие операции

К их числу следует отнести: Abs, \*\* и другие. Операция выделения абсолютного значения ( abs ) совместима с любым числовым типом операнда. Операция возведения в степень ( \*\* ) операндом слева имеет целое или тип с плавающей точкой, а в качестве правого операнда (степень) - только целое.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

Задание: для заданной нерегулярной логической схемы составить структурное VHDL-описание, выполнить моделирование на разных наборах значений входных переменных, построить систему логических функций, реализуемую схемой как на рисунке 2.1.

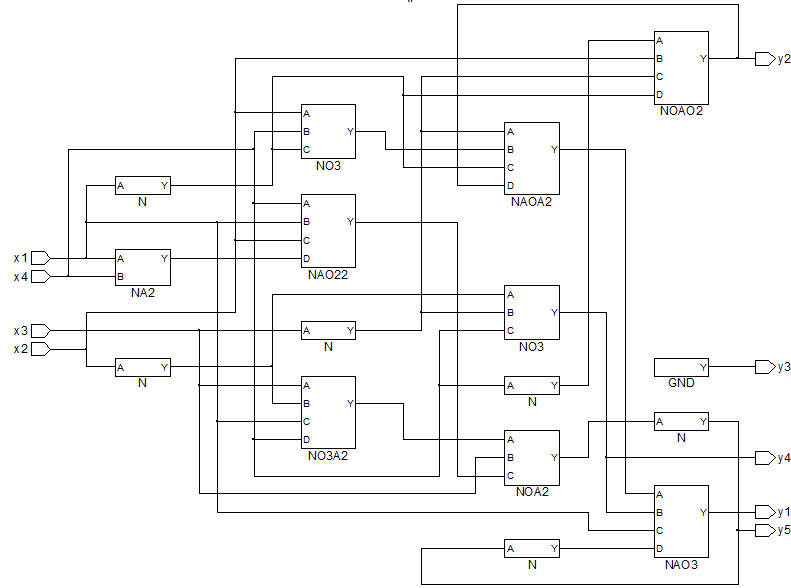


Рисунок 2.1 – Система логических функций

Практическая часть выполняется в специальной среде разработки Active-HDL Student Edition. Она построена на основе ядра логического моделирования. В таблице 2.1 представлены логические элементы, используемые в построении заданной схемы.

Таблица 2.1 – Используемые логические элементы

|  |  |  |
| --- | --- | --- |
| N |  | 1 |
| NA2 |  | 2 |
| NO3 |  | 4 |
| NAO22 |  | 3 |
| NO3A2 |  | 5 |
| NAOA2 |  | 4 |
| NOA2 |  | 3 |
| NOAO2 |  | 4 |
| NAO3 |  | 5 |

Каждый логический элемент создается в виде исходного файла с расширением vhd. Библиотека логических элементов представлена на рисунке 2.2.

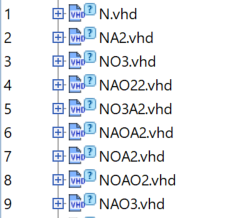


Рисунок 2.2 – Библиотека файлов логических элементов

Файл содержит код, который реализует логический элемент. Как пример, элемент NA2 () представляет логический и не, он имеет 2 входа и 1 выход. Также элементу NA2 прописывается задержка в 2ns согласно заданию. Код элемента NA2.vhd представлен на рисунке 2.3.

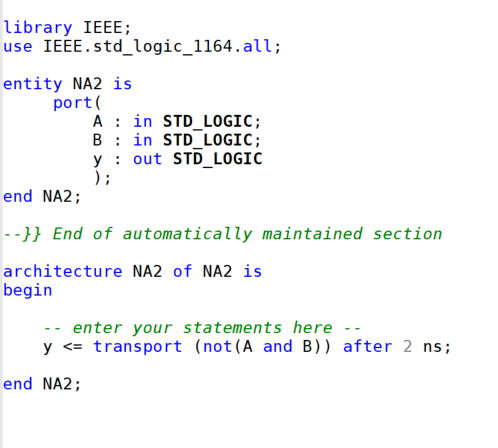


Рисунок 2.3 – Код на VHDL логического элемента NA2

После написания всех логических элементов строится схема, как показано на рисунке 2.4

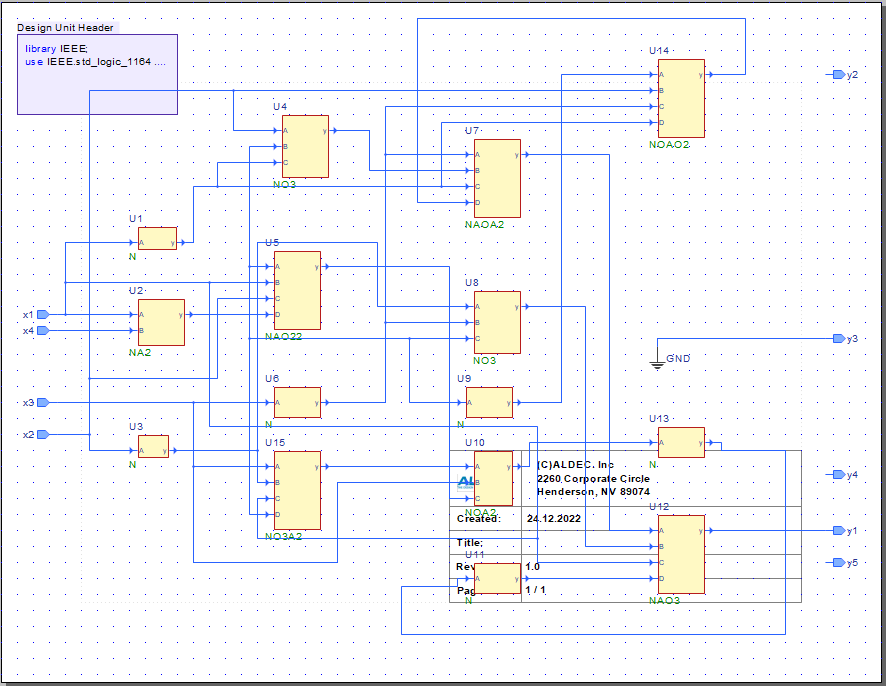


Рисунок 2.4 – Схема системы логических функций

Для работы программы строится график при помощи меню New Waveform и инициализируется симуляция (Initialize Simulation) в меню Simulation. Затем указываются значения входных параметров: x1, x2, x3, x4. Значение каждого из них изменяется через 100 ns. Запускается симуляция при помощи кнопки Run и выводится график как на рисунке 2.5.

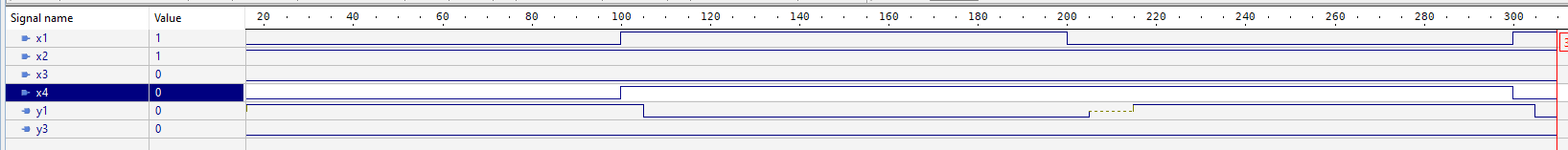


Рисунок 2.5 – График работы программы

Схема после старта симуляции также наглядно показывает работу логических элементов. Схема представлена на рисунке 2.6.

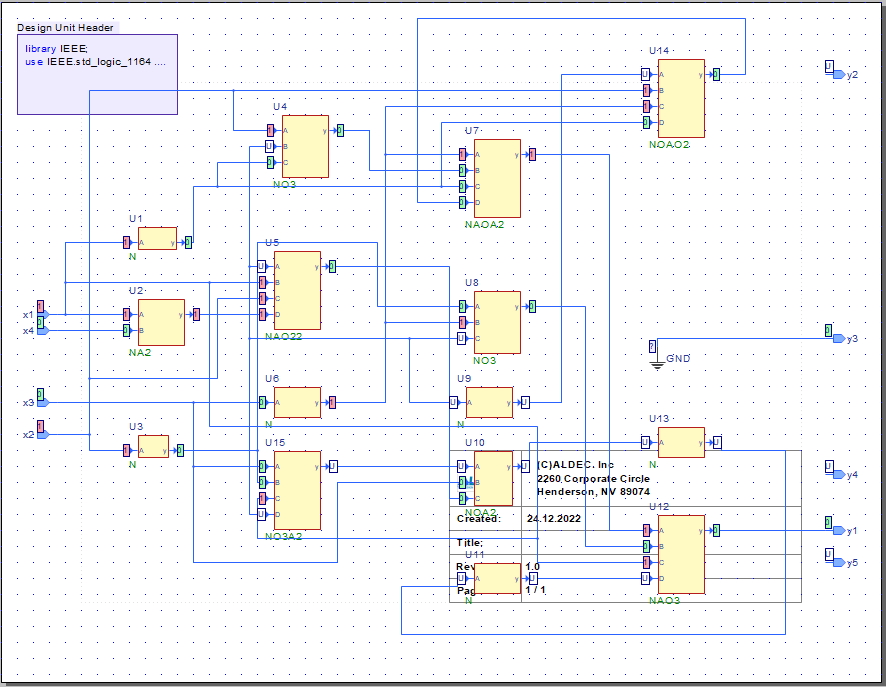


Рисунок 2.6 – Схема после запуска симуляции

# ЗАКЛЮЧЕНИЕ

В ходе выполнения теоретической части курсовой работы был рассмотрен язык проектирования VHDL. Также была рассмотрена его архитектура, типы данных, описание интерфейса устройства и операции языка VHDL.

В практической части курсовой работы была создана VHDL модель типов элементов и рабочая VHDL схема.

# СПИСОК ЛИТЕРАТУРЫ

1. Бибило П.Н. [Основы языка VHDL. Второе издание.](http://vhdl.bas-net.by/literature/bibilo/#basics) — М.: Солон-Р, 2002. — 224 с.

2. Сергиенко А.М. VHDL для проектирования вычислительных устройств. — К.: ЧП “Корнейчук”, ООО “ТИД “ДС”, 2003. — 208 с.

3. VHDL’92. Новые свойства языка описания аппаратуры VHDL / Пер с англ. — М.: Радио и связь, 1995. — 256 с.

4. Уэйкерли Дж. Ф. Проектирование цифровых устройств. Т. 2. — М.: Постмаркет, 2002. — 528 с.

5. Палагин А.В., Опанасенко В.Н., Сахарин В.Г. Опыт проектирования цифровых устройств на базе ПЛИС с использованием HDL-технологии. // Управляющие системы и машины. — 2004. — N 6. – С. 11 – 20.