|  |  |  |
| --- | --- | --- |
| https://www.mirea.ru/bitrix/templates/unlimtech/images/logo.png | |  |
| МИНОБРНАУКИ РОССИИ | |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | |  |
|  | **Институт** информационных технологий (ИТ) | |
|  | **Кафедра** Вычислительной техники (ВТ) | |

|  |  |
| --- | --- |
| **Практические работы №1-8**  **по дисциплине**  **«Архитектура ВМиС»** | |
|  | |
| Выполнил студент группы |  |
| Принял преподаватель | Гололобов А.А. |

Москва 2023

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc171244796)

[Практическая работа №1 4](#_Toc171244797)

[Практическая работа №2 7](#_Toc171244798)

[Практическая работа №3 9](#_Toc171244799)

[Практическая работа №4 13](#_Toc171244800)

[Практическая работа №5 17](#_Toc171244801)

[Практическая работа №6 20](#_Toc171244802)

[Практическая работа №7 27](#_Toc171244803)

[Практическая работа №8 33](#_Toc171244804)

[8.1 Задание 33](#_Toc171244805)

[ЗАКЛЮЧЕНИЕ 46](#_Toc171244806)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 47](#_Toc171244807)

# ****ВВЕДЕНИЕ****

Широкое внедрение автоматики во все сферы человеческой деятельности, наблюдаемое в настоящее время, предъявляет жесткие требования к изделиям электронной техники. Это связано, с одной стороны, с возрастанием важности и сложности решаемых задач, а с другой стороны, необходимостью улучшения качественных характеристик, таких как: быстродействие, надежность, потребляемая мощность, габариты, стоимость и др. Одним из путей решения данной проблемы является использование новой элементной базы — программируемых логических интегральных схем (ПЛИС — Programmable Logic Device — PLD). ПЛИС представляют собой интегральные схемы, обладающие гибкостью заказных БИС (больших интегральных схем) и доступностью традиционной "жесткой" логики. По существу, разработка устройств на основе ПЛИС представляет собой новую технологию проектирования электронных схем, включая их изготовление и сопровождение. САПР QUARTUS II представляет собой интегрированную среду для разработки цифровых устройств на базе программируемых логических интегральных схем (ПЛИС) фирмы АLTERA и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

* создание проектов устройств;
* синтез структур и трассировку внутренних связей ПЛИС;
* подготовку данных для программирования или конфигурирования ПЛИС (компиляцию);
* верификацию проектов (функциональное моделирование и временной анализ);
* программирование или конфигурирование ПЛИС;

# ****Практическая работа №1****

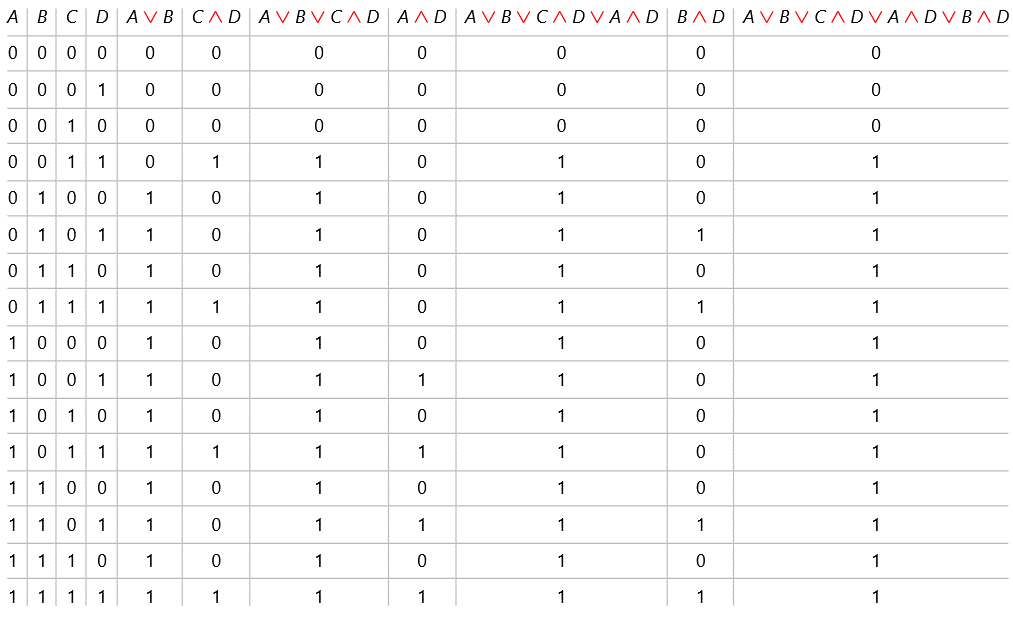
**1.1 Задание**

Графический ввод схемы и симуляция в САПР QUARTUS II. Персональный вариант:

**1.2 Теоретическое введение**

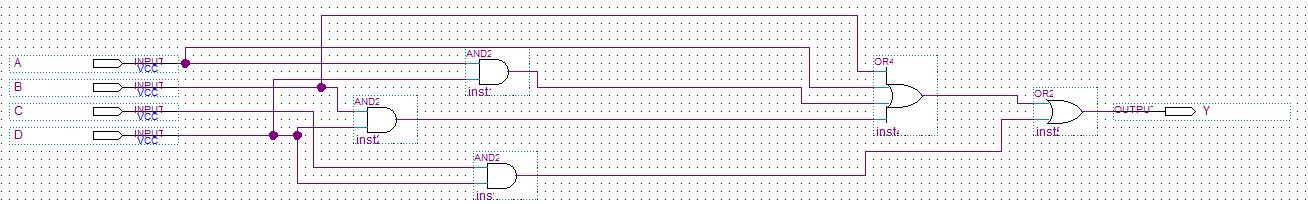
Математической основой цифровой электроники и вычислительной техники является алгебра логики или булева алгебра (по имени английского математика Джона Буля). В булевой алгебре независимые переменные или аргументы (Х) принимают только два значения: «0» или «1». Зависимые переменные или функции (Y) также могут принимать только два значения: «0» или «1». Схемы, реализующие логические функции, называются логическими элементами. Основные логические элементы имеют, как правило, один выход (Y) и несколько входов, число которых равно числу аргументов. На электрических схемах логические элементы рисуют в виде прямоугольников с выводами для входных (слева) и выходных (справа) переменных. В средине прямоугольника изображается символ, обозначающий функциональное назначение элемента. Логические элементы, которые реализуют операции конъюнкции, дизъюнкции, функции Пирса и Шеффера, могут быть, в общем случае, n входными. В таблице истинности такого элемента количество возможных комбинаций входных переменных N, в общем случае равняется: N = 2 n, где n – число входных переменных. Логические элементы используются для построения интегральных микросхем, которые выполняют разнообразные логические и арифметические операции. ФАЛ любой сложности можно реализовать при помощи обозначенных логических элементов.

**1.3 Выполнение работы**



**Рис. 1 – таблица истинности для формулы указанной в варианте**

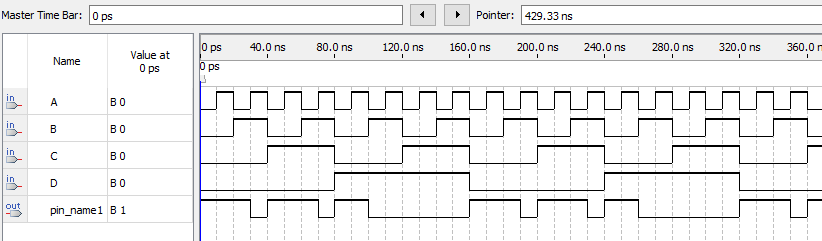
Схема представлена на рисунке 2:



**Рис. 2 – схема логической функции**

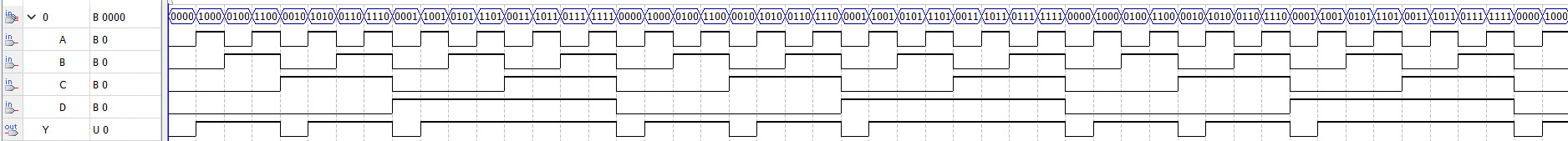
**Диаграмма**

Временная диаграмма представлена на рисунке 3:



**Рис. 3 – Временная диаграмма по схеме**

Временная диаграмма с группировкой представлена на рисунке 4:



**Рис. 4 – Временная диаграмма c группировкой по схеме**

**Таблица истинности**

Таблица истинности, составленная по результатам диаграммы

*Таблица 1 – таблица истинности*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A |  |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Практическая работа №2

**2.1 Задание**

Смоделировать логическую схему при помощи текстового редактора Quartus II.

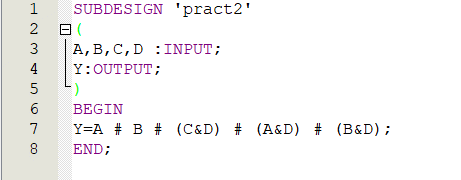
**2.2 Теоретическое введение**

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС QUARTUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение \*.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы QUARTUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera. Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов. При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

**2.3 Выполнение работы**

На языке AHDL составить логическое выражение

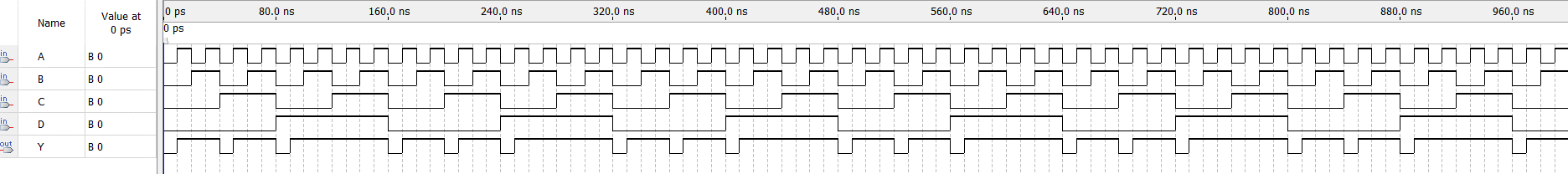
Код программы представлен на рисунке 5:



**Рис. 5 – Код AHDL**

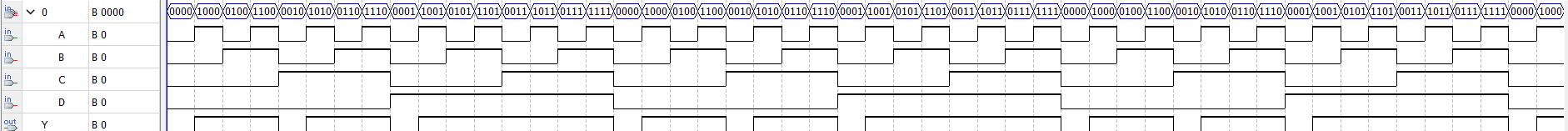
Временная диаграмма представлена на рисунке 6:

**Диаграмма**



**Рис. 6 – Временная диаграмма по коду AHDL**

Временная диаграмма с группировкой представлена на рисунке 7:



**Рис. 7 – Временная диаграмма с группировкой по коду AHDL**

**Таблица истинности**

Таблица истинности, составленная по результатам диаграммы

*Таблица 2 – таблица истинности*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D | C | B | A |  |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Практическая работа №3

**3.1 Задание**

Спроектировать логическую схему при помощи графического редактора Quartus II. Исследовать работу схемы с использованием сигнального редактора Quartus II. **Вариант**: CD 4x2. Описание: составить таблицу истинности (таб. 3) и схему CD 4x2 (рис. 8).

**3.2 Теоретическое введение**

Шифратором (coder CD) М х N называют комбинационное устройство с М входами и N выходами, преобразующее М-разрядный унитарный код в ЛГ-разрядный двоичный код. Шифратор по отношению к дешифратору выполняет обратную функцию. Шифраторы классифицируют по ряду признаков.

По числу входов различают:

• полные шифраторы, число входов которых М = 2^ДГ;

• неполные шифраторы, имеющие число входов М < 2^N.

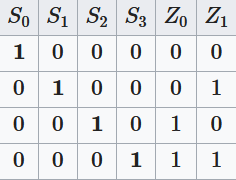
По уровням входных и выходных сигналов выделяют:

• шифраторы высокого уровня, активные сигналы на входах и выходах которых имеют уровень логической единицы; • шифраторы низкого уровня, активные входные и выходные сигналы которых соответствуют уровню логического нуля.

По функциональной значимости входов шифраторы разделяют на две группы:

• шифраторы с равнозначными функциями входов, в которых все входы равноценны и при подаче на любой из них активного уровня сигнала на выходе формируется двоичный код. В таких шифраторах нельзя подавать несколько входных сигналов одновременно от разных источников, т.е. должна соблюдаться очередность подачи сигналов от них. Если на один из входов шифратора подан сигнал, остальные входы шифратора должны быть заблокированы;

• приоритетные шифраторы, в которых возможна одновременная подача на входы сигналов от разных источников, однако только один из них, имеющий больший приоритет, выполнит функцию формирования выходного кода. Как правило, наивысший приоритет назначается входу с самым высоким порядковым номером.



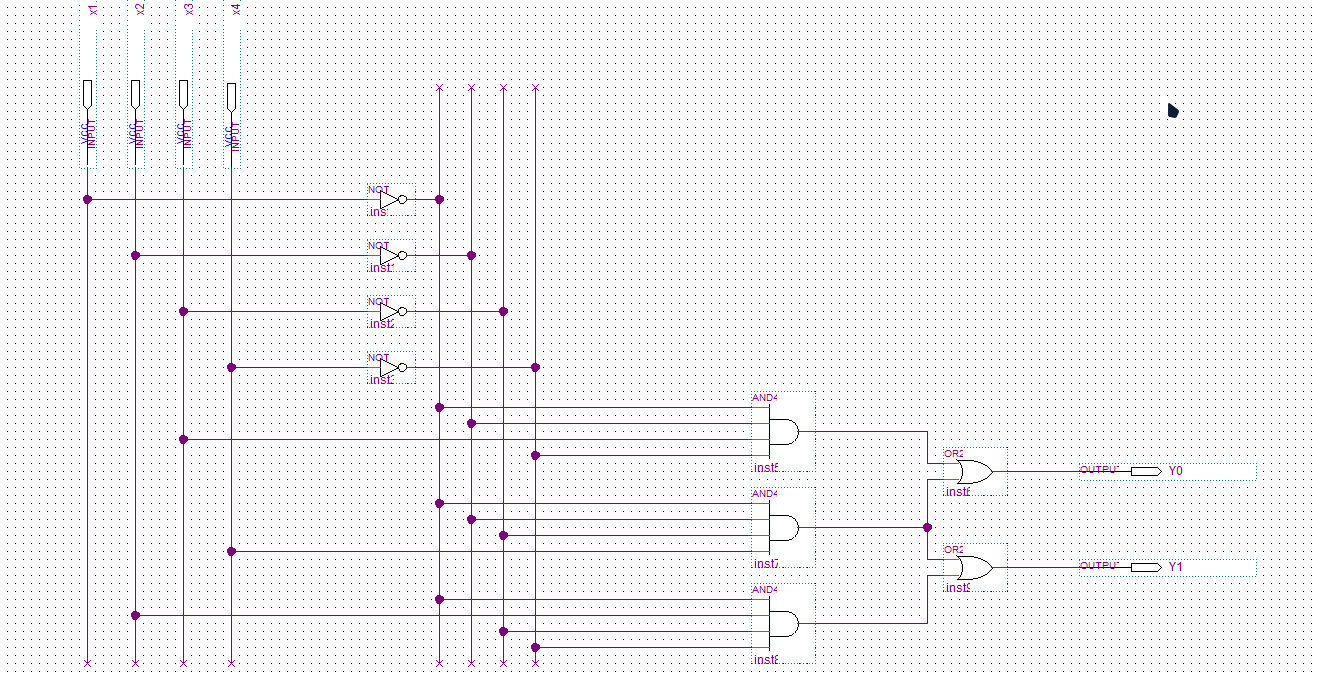
**Рисунок 8 – таблица истинности шифратора CD 4X2**

**3.3 Выполнение работы**

**Постановка задачи и персональный вариант**

**Схема**

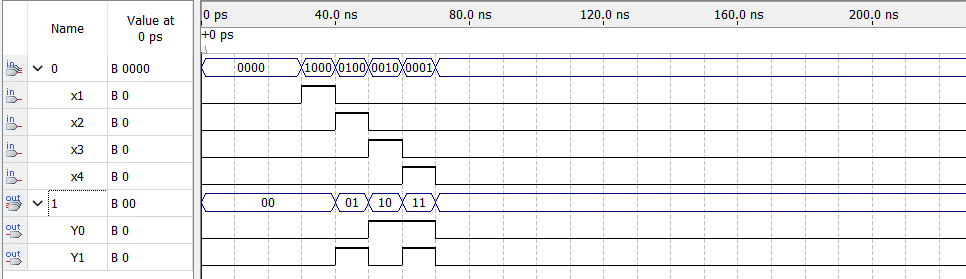
Схема CD 4x2 представлена на рис. 8



**Рисунок 9 – схема CD 4X2**

**Временная диаграмма**

Временная диаграмма с применением группировки, получившаяся по схеме, представлена на рис. 9



**Рисунок 10 – временная диаграмма CD 4X2 с применением группировки**

**Таблица истинности**

По полученному результату восстанавливаем таблицу истинности для D триггера.

*Таблица 3 – таблица истинности на основе схемы*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **x3** | **x4** | **Y0** | **Y1** |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

# Практическая работа №4

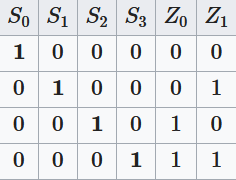
**4.1 Задание**

Приобретение основных навыков описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора Quartus II.

На языке AHDL составить логическое выражение компаратора с тремя выходами, сравнивающего два двухразрядных числа.

**4.2 Теоретическое введение**

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС QUARTUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение \*.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы QUARTUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera. Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов. При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

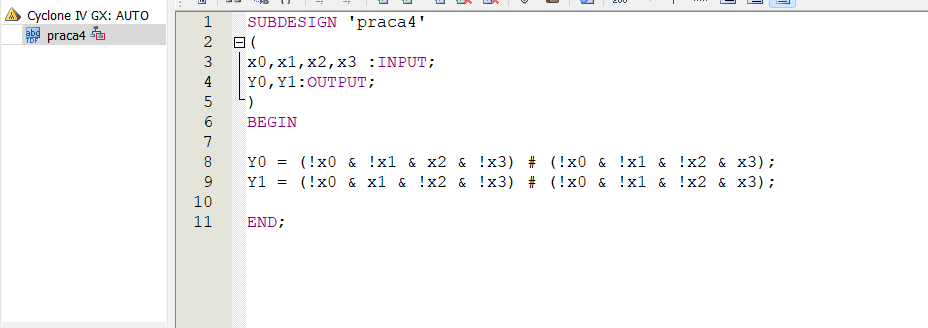


**Рисунок 11 – таблица истинности шифратора CD 4X2**

**4.3 Выполнение работы**

**Код программы**

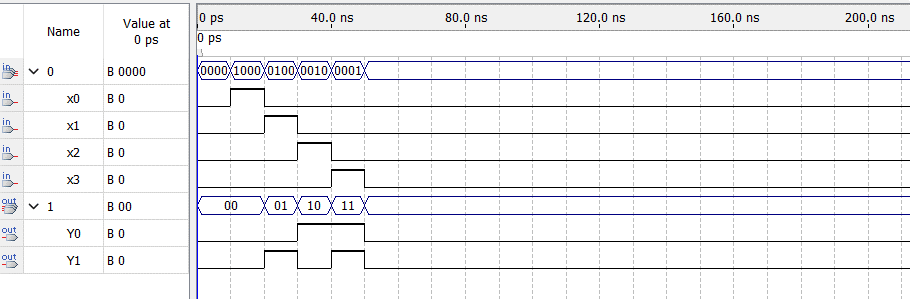
Код программы представлен на рис. 12.



**Рисунок 12 – код программы на языке AHDL составленный для CD 4X2**

**Диаграмма**

Временная диаграмма с применением группировки по коду AHDL CD 4X2 представлена на рис. 13.



**Рисунок 13 – временная диаграмма, составленная по коду AHDL**

**Таблица истинности**

По полученному результату временной диаграммы восстанавливаем таблицу истинности для CD 4X2.

Таблица 4 – таблица истинности на основе схемы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **x3** | **x4** | **Y0** | **Y1** |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 |

# Практическая работа №5

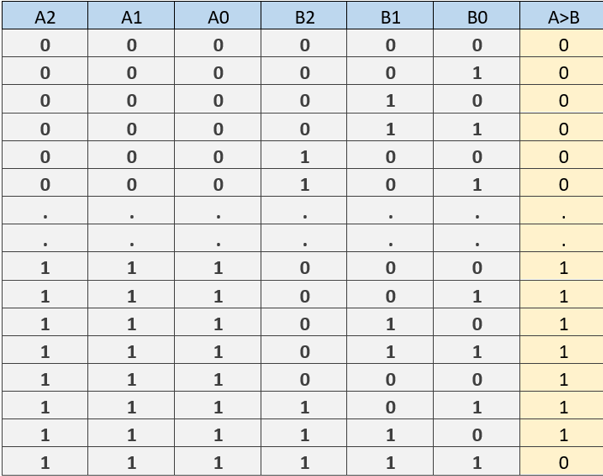
**5.1 Задание**

Приобретение основных навыков описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора САПР QUARTUS II. Персональный вариант: Вариант 23. Составить схему трёхразрядного компаратора A>B.

**5.2 Теоретическое введение**

Цифровой компаратор логическое устройство с двумя словарными входами, на которые подаются два разных двоичных слова равной в битах длины и обычно с тремя двоичными выходами, на которые выдаётся признак сравнения входных слов, — первое слово больше второго, меньше или слова равны. При этом выходы «больше», «меньше» имеют смысл, если входные слова кодируют числа в том или ином машинном представлении.

Часто цифровые компараторы не имеют выходов «больше», «меньше», а только выход «равно».

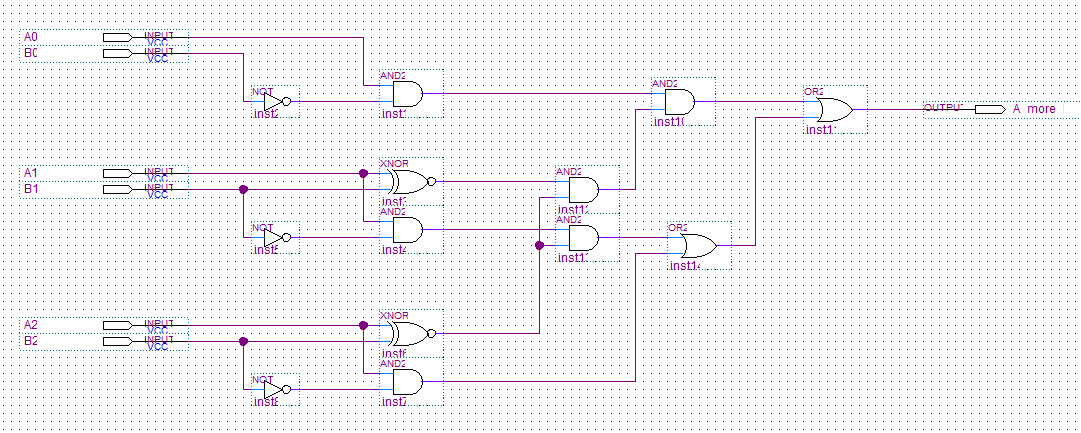


**Рисунок 14 – таблица истинности компаратора A>B**

**5.3 Выполнение работы**

**Схема**

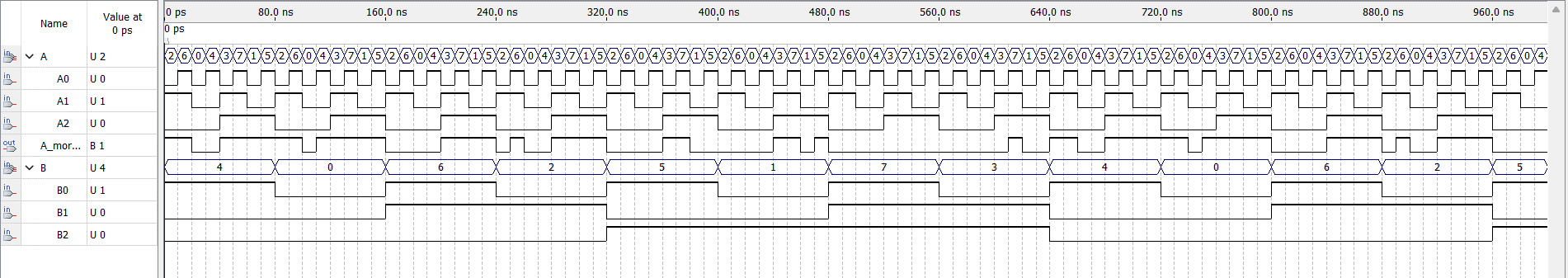
Представлено на рис. 15



**Рисунок 15 – схема трёхразрядного компаратора A>B**

**Временная диаграмма**

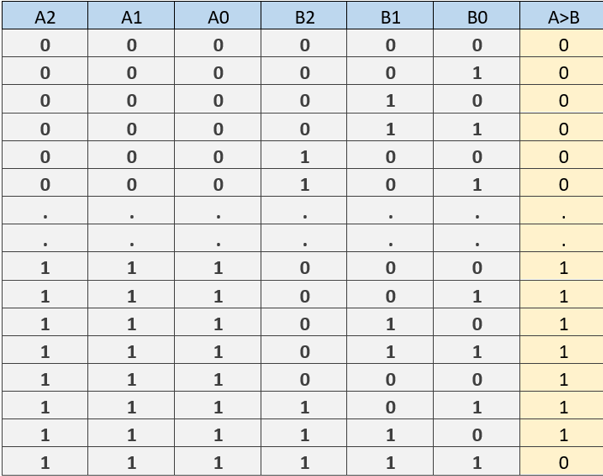
Временная диаграмма представлена на рисунке 16.



**Рисунок 16 – временная диаграмма трёхразрядного компаратора A>B**

**Таблица истинности**

По полученному результату временной диаграммы восстанавливаем таблицу истинности для трёхразрядного компаратора



**Рисунок 17 – таблица истинности компаратора A>B**

# Практическая работа №6

**6.1 Задание**

Ознакомиться с САПР QUARTUS II фирмы Altera, получить практические навыки создания проектов по схемотехнике ЭВМ в САПР (ввод схем, компиляция и моделирование). Индивидуальный вариант: 23.

**6.2 Теоретическое введение**

Для построения счетчиков и регистров используются синхронные триггеры, переключение которых происходит только при наличии синхронизирующего сигнала (синхроимпульса) на входе С. Наиболее часто для построения регистров и счетчиков используется D-триггер, имеющий специальный информационный вход D, и динамический вход С. Устройство, называемое счетчиком, предназначено для подсчета числа поступающих на вход сигналов (импульсов) в произвольной системе счисления. Двоичные счетчики строятся на основе триггеров, работающих в счетном режиме (Т-триггер или счетный триггер). Счетный триггер может быть получен из универсального D - триггера путем соединения его инверсного выхода Q со входом D. У счетного триггера состояние выхода изменяется на противоположное при поступлении на вход С каждого очередного счетного импульса. Для построения счетчика с требуемым коэффициентом пересчета Кс, отличным от величины 2N (N - число двоичных разрядов счетчика), используется принудительный сброс счетчика в исходное состояние при достижении счетчиком числа Кс. Устройство, называемое регистром, служит в основном для хранения чисел в двоичном коде при выполнении над ними различных арифметических и логических операций. С помощью регистров выполняются такие действия над числами, как передача их из одного устройства в другое, арифметический и логический сдвиг в сторону младших или старших разрядов, преобразование кода из последовательного в параллельный и наоборот и т.д.

**6.3 Выполнение работы**

Персональный вариант 23:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вар | Состояние графа | | | | | | | | | | | | | | | | |
| 23 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| 2 | 9 | 1 | 12 | 11 | 8 | 0 | 5 | 15 | 10 | 13 | 4 | 6 | 7 | 8 | 14 |  |

**Таблица перекодировки состояний устройства в десятичном и двоичном коде**

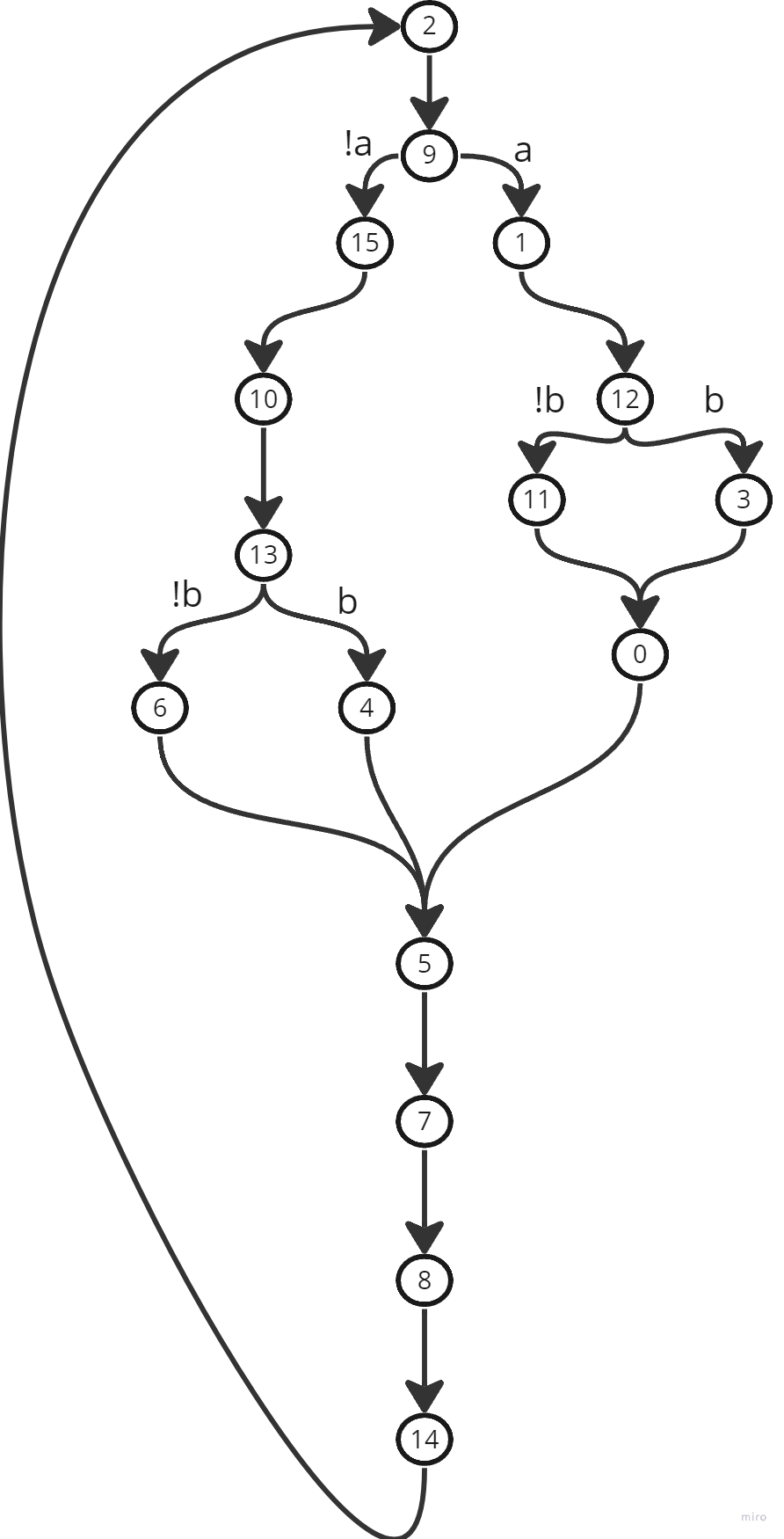
Ниже представлена таблица перекодировки (таблица 5) состояний автомата в десятичном и двоичном коде.

*Таблица 5 - таблица перекодировки состояний автомата и их двоичный код*

|  |  |  |
| --- | --- | --- |
| № состояния | № состояния (2 вариант) | Двоичный код  q3,q2,q1,q0 |
| 0 | 2 | 0010 |
| 1 | 9 | 1001 |
| 2 | 1 | 0001 |
| 3 | 12 | 1100 |
| 4 | 11 | 1011 |
| 5 | 3 | 0011 |
| 6 | 0 | 0000 |
| 7 | 5 | 0101 |
| 8 | 15 | 1111 |
| 9 | 10 | 1010 |
| 10 | 13 | 1101 |
| 11 | 4 | 0100 |
| 12 | 6 | 0110 |
| 13 | 7 | 0111 |
| 14 | 8 | 1000 |
| 15 | 14 | 1110 |

**Граф на основе таблицы перекодировки**

Далее подставим новые значения состояний в исходный граф (рис 18).



**Рисунок 18 – Граф, полученный с учетом таблицы перекодировки**

**Таблица истинности автомата**

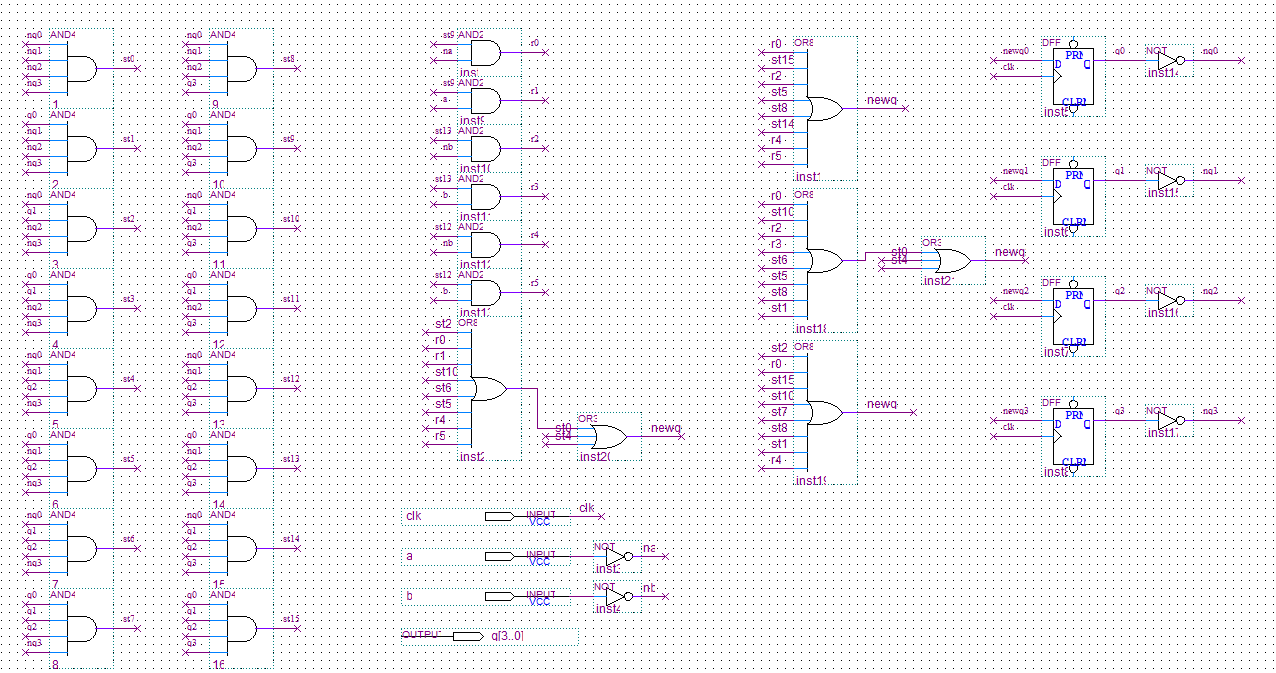
На основании графа состояний составляем таблицу истинности автомата (таблица 6).

*Таблица 6 – Таблица истинности автомата*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старое состояние | | Условие | Новое состояние | |
| № | код | № | Код |
| 2 | 0010 | - | 9 | 1001 |
| 9 | 1001 | A=0 | 15 | 1111 |
| 9 | 1001 | A=1 | 1 | 0001 |
| 15 | 1111 | - | 10 | 1010 |
| 10 | 1010 | - | 13 | 1101 |
| 13 | 1101 | B=0 | 6 | 0110 |
| 13 | 1101 | B=1 | 4 | 0100 |
| 6 | 0110 | - | 5 | 0101 |
| 5 | 0101 | - | 7 | 0111 |
| 7 | 0111 | - | 8 | 1000 |
| 8 | 1000 | - | 14 | 1110 |
| 14 | 1110 | - | 2 | 0010 |
| 1 | 0001 | - | 12 | 1100 |
| 12 | 1100 | B=0 | 11 | 1011 |
| 12 | 1100 | B=1 | 3 | 0011 |
| 11 | 1011 | - | 0 | 0000 |
| 0 | 0000 | - | 5 | 0101 |
| 4 | 0100 | - | 5 | 0101 |
| 3 | 0011 | - | 0 | 0000 |

**Функциональная схема без минимизации**

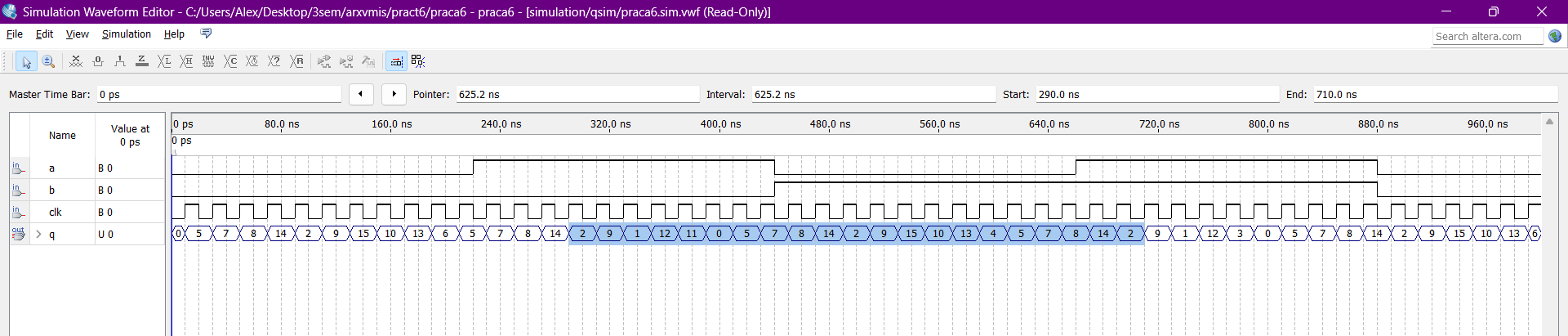
На основе таблицы истинности составим функциональную схему(рис.19)



**Рисунок 19 - Функциональная схема без минимизации**

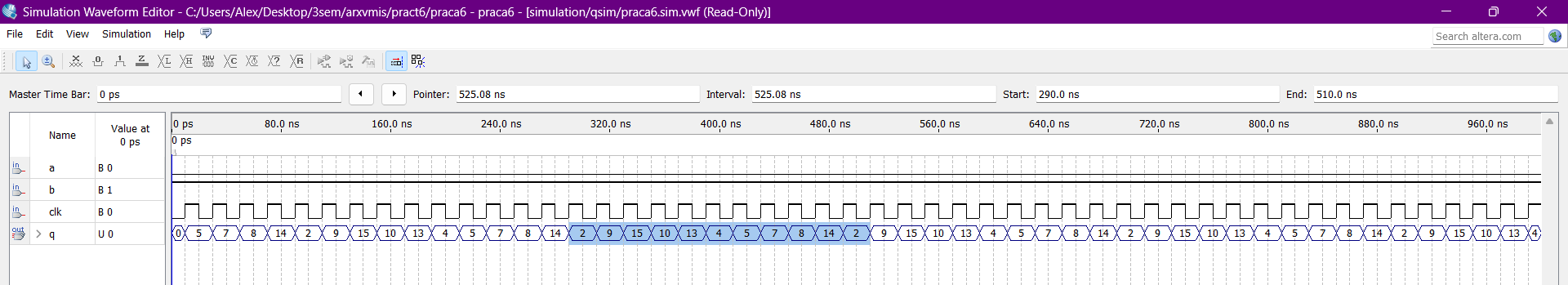
**Временная диаграмма**

Далее запускаем функциональную симуляцию по отрицательным a, b: а = 0 и b = 0 . В результате получаем следующую диаграмму (рис. 20)



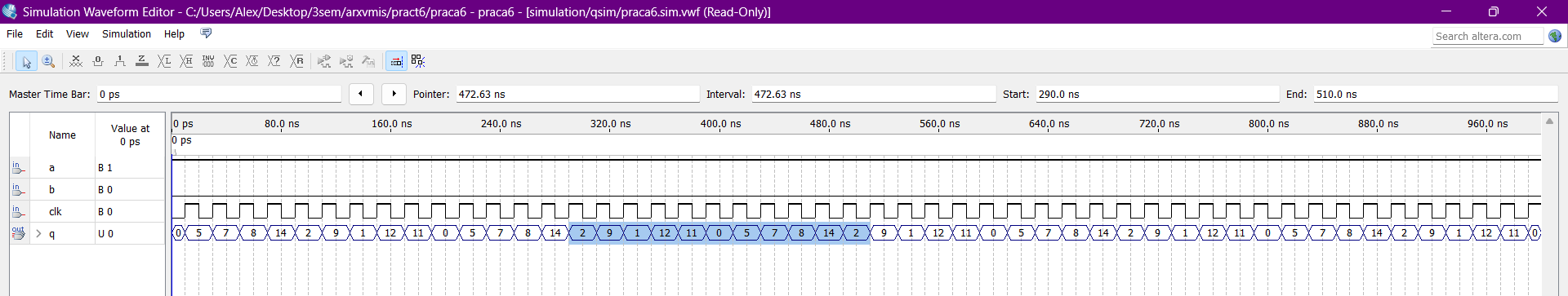
**Рисунок 20 – Временная диаграмма**

Далее запускаем функциональную симуляцию по отрицательной а и положительном b: a=0, b=1. В результате получаем следующую диаграмму (рис. 21)



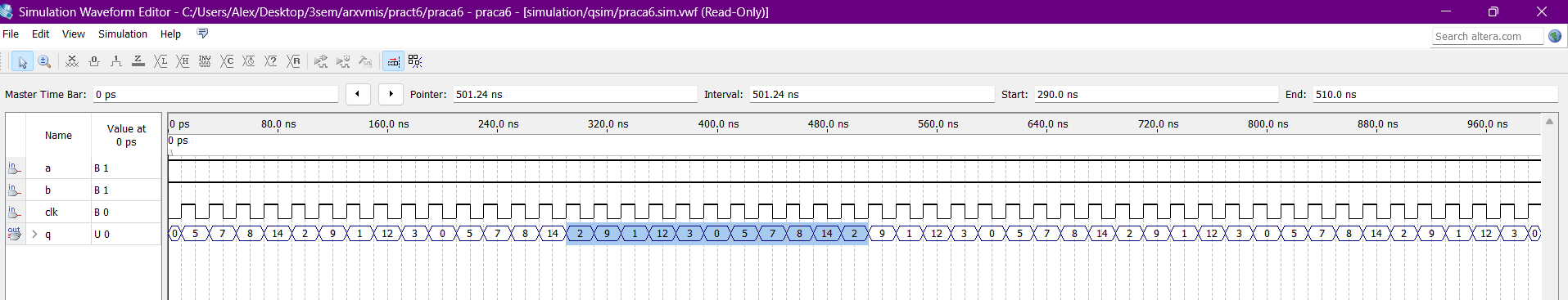
**Рисунок 21 – Временная диаграмма**

Далее запускаем функциональную симуляцию по положительной a и отрицательным b: а = 1 и b = 0. В результате получаем следующую диаграмму (рис. 22)



**Рисунок 22 – Временная диаграмма**

Далее запускаем функциональную симуляцию по положительным a, b: а = 1 и b = 1. В результате получаем следующую диаграмму (рис. 22)



**Рисунок 22 – Временная диаграмма**

# Практическая работа №7

**7.1 Задание**

Приобретение основных навыков описания цифровых схем с помощью языка описания аппаратуры AHDL. Смоделировать логическую схему при помощи текстового редактора САПР QUARTUS II.

**7.2 Теоретическое введение**

Язык описания аппаратуры AHDL разработан фирмой Altera и предназначен для описания комбинационных и последовательностных логических устройств, групповых операций, цифровых автоматов (state machine) и таблиц истинности с учетом архитектурных особенностей ПЛИС фирмы Altera. Он полностью интегрируется с системой автоматизированного проектирования ПЛИС QUARTUS II. Файлы описания аппаратуры, написанные на языке AHDL, имеют расширение \*.TDF (Text design file). Для создания TDF-файла можно использовать как текстовый редактор системы QUARTUS II, так и любой другой. Проект, выполненный в виде TDF-файла, компилируется, отлаживается и используется для формирования файла программирования или загрузки ПЛИС фирмы Altera. Операторы и элементы языка AHDL являются достаточно мощным и универсальным средством описания алгоритмов функционирования цифровых устройств, удобным в использовании. Язык описания аппаратуры AHDL дает возможность создавать иерархические проекты в рамках одного этого языка или же в иерархическом проекте использовать как TDF-файлы, разработанные на языке AHDL, так и другие типы файлов. При распределении ресурсов устройств разработчик может пользоваться командами текстового редактора или операторами языка AHDL для того, чтобы сделать назначения ресурсов и устройств. Кроме того, разработчик может только проверить синтаксис или выполнить полную компиляцию для отладки и запуска проекта. Любые ошибки автоматически обнаруживаются обработчиком сообщений и высвечиваются в окне текстового редактора.

**7.3 Выполнение работы**

**Код программы**

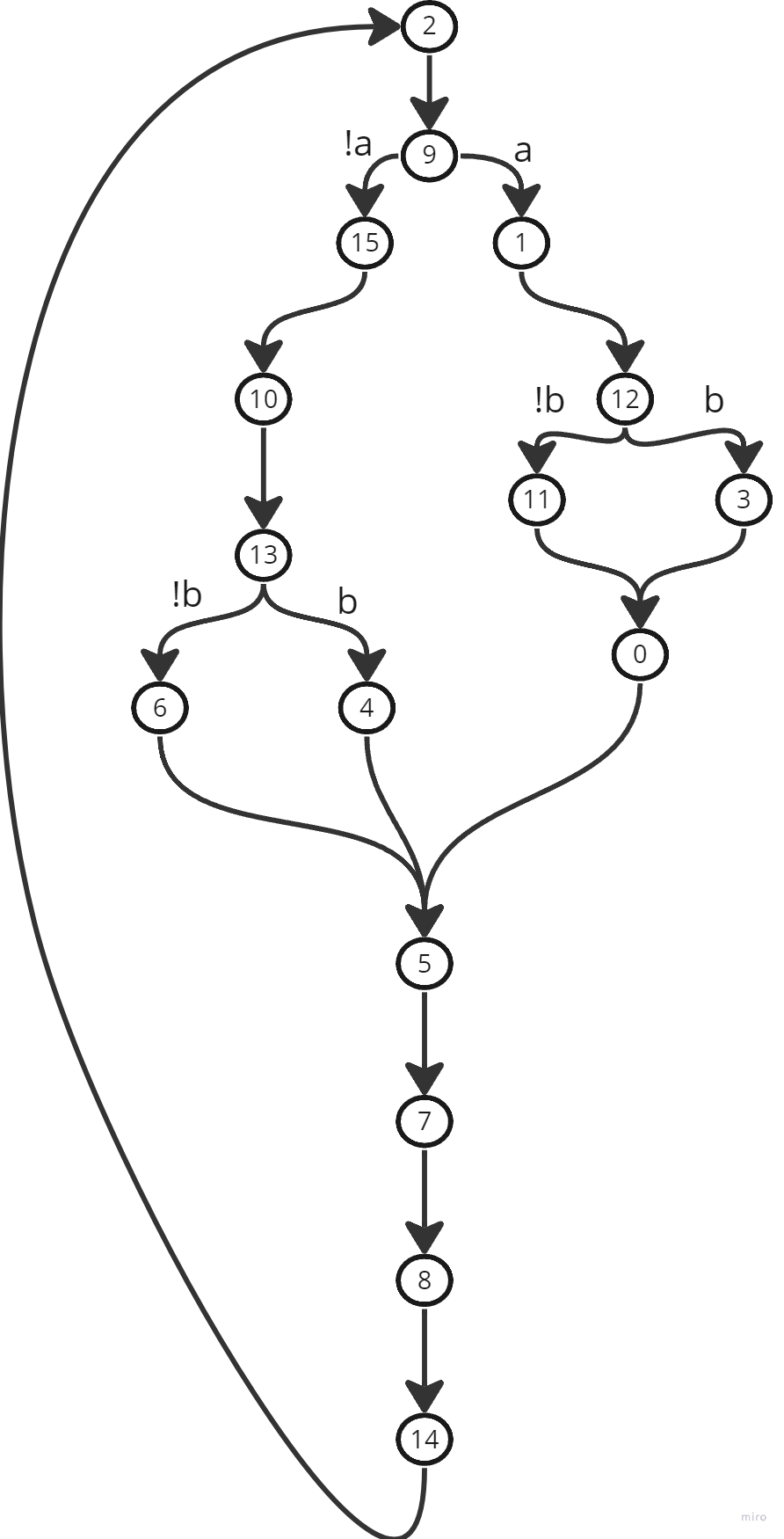
Код программы представлен в листинге 1.

Листинг 1 – код программы на языке AHDL составленный для счётчика с произвольным модулем счёта

|  |
| --- |
| SUBDESIGN 'pract7'  (  a,b,clock:INPUT;  q[3..0]:OUTPUT;  )  VARIABLE  na,nb: NODE;  newq0,newq1,newq2,newq3: NODE;  st0,st1,st2,st3,st4,st5,st6,st7,st8,st9,st10,st11,st12,st13,st14,st15: NODE;  r0,r1,r2,r3,r4,r5: NODE;  trig0,trig1,trig2,trig3: dff;  BEGIN  na = !a;  nb = !b;  st0 = !q0&!q1&!q2&!q3;  st1 = q0&!q1&!q2&!q3;  st2 = !q0&q1&!q2&!q3;  st3 = q0&q1&!q2&!q3;  st4 = !q0&!q1&q2&!q3;  st5 = q0&!q1&q2&!q3;  st6 = !q0&q1&q2&!q3;  st7 = q0&q1&q2&!q3;  st8 = !q0&!q1&!q2&q3;  st9 = q0&!q1&!q2&q3;  st10 = !q0&q1&!q2&q3;  st11 = q0&q1&!q2&q3;  st12 = !q0&!q1&q2&q3;  st13 = q0&!q1&q2&q3;  st14 = !q0&q1&q2&q3;  st15 = q0&q1&q2&q3;  r0 = st9&!a;  r1 = st9&a;  r2 = st13&!b;  r3 = st13&b;  r4 = st12&!b;  r5 = st12&b;  newq0 = st2#r0#r1#st10#st6#st5#r4#r5#st0#st4;  newq1 = r0#st15#r2#st5#st8#st14#r4#r5;  newq2 = r0#st10#r2#r3#st6#st5#st8#st1#st0#st4;  newq3 = st2#r0#st15#st10#st7#st8#st1#r4;  trig0.clk=clock;  trig0.d=newq0;  q[0]=trig0.q;  trig1.clk=clock;  trig1.d=newq1;  q[1]=trig1.q;  trig2.clk=clock;  trig2.d=newq2;  q[2]=trig2.q;  trig3.clk=clock;  trig3.d=newq3;  q[3]=trig3.q;  END; |

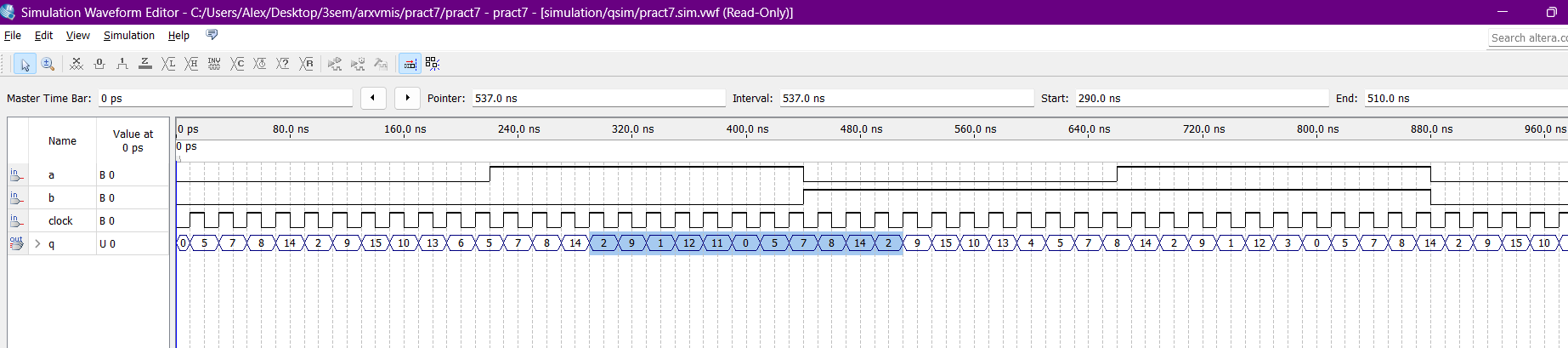
**Граф на основе таблицы перекодировки**

Далее подставим новые значения состояний в исходный граф (рис 23).



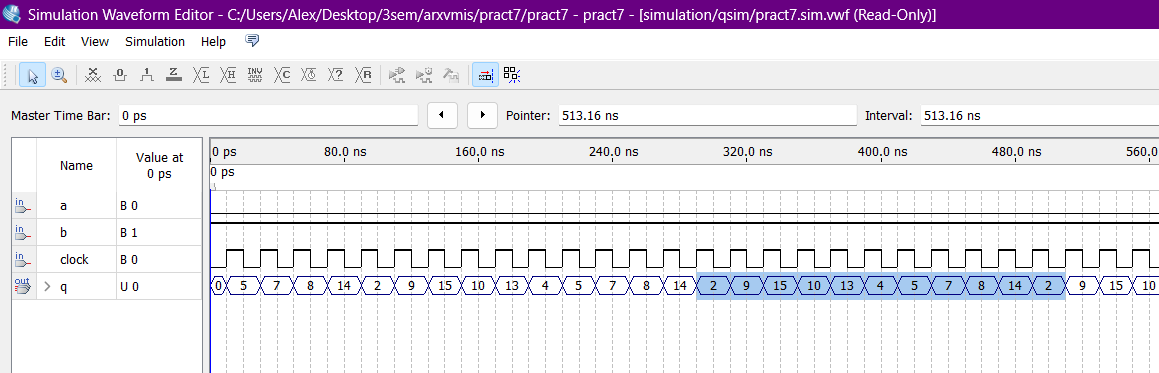
**Временная диаграмма**

Далее запускаем функциональную симуляцию по отрицательным a, b: а = 0 и b = 0 . В результате получаем следующую диаграмму (рис. 24)



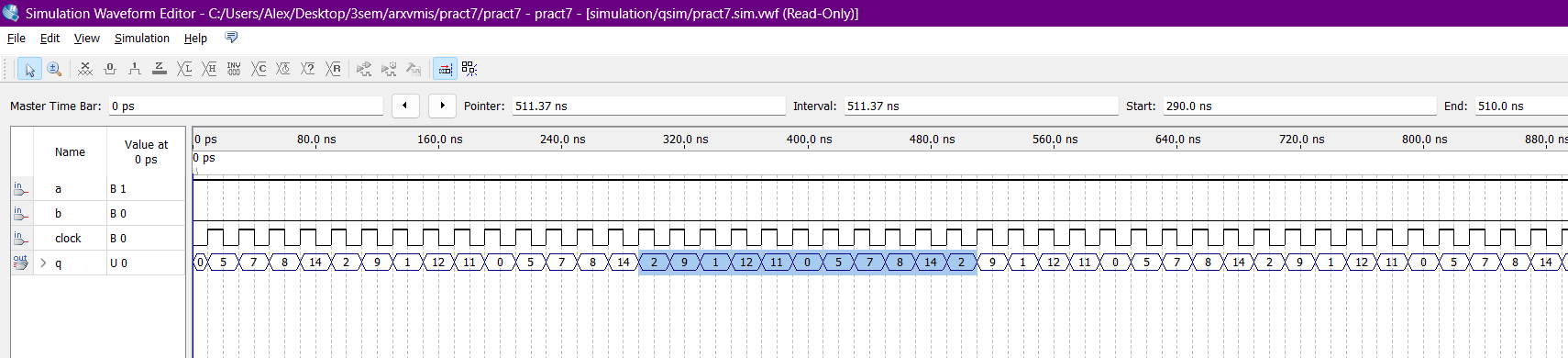
**Рисунок 24 – Временная диаграмма**

Далее запускаем функциональную симуляцию по отрицательной а и положительном b: a=0, b=1. В результате получаем следующую диаграмму (рис. 25)



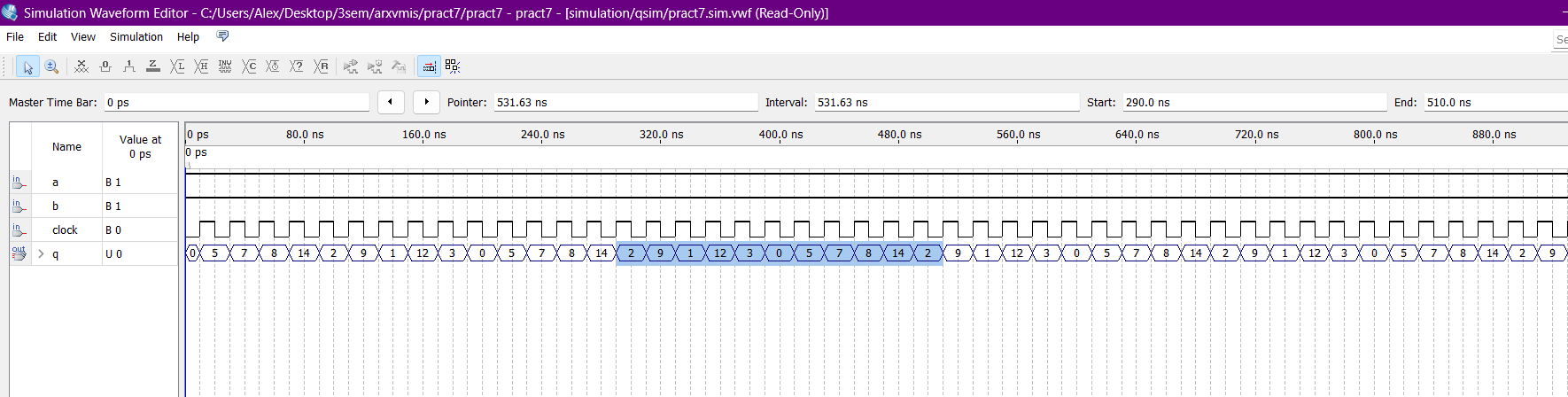
**Рисунок 25 – Временная диаграмма**

Далее запускаем функциональную симуляцию по положительной a и отрицательным b: а = 1 и b = 0. В результате получаем следующую диаграмму (рис. 26)



**Рисунок 26 – Временная диаграмма**

Далее запускаем функциональную симуляцию по положительным a, b: а = 1 и b = 1. В результате получаем следующую диаграмму (рис. 27)



**Рисунок 27 – Временная диаграмма**

**Практическая работа №8**

**8.1 Задание**

Файловая система: изучение команд работы с файлами и каталогами. Индивидуальный вариант: 2 - Библиотека (ФИО автора, название произведения, год издания, издательство). Поиск по издательству. Сортировка по году издания.

**8.2 Теоретическое введение**

Файловая система — порядок, определяющий способ организации, хранения и именования данных на носителях информации в компьютерах, а также в другом электронном оборудовании: цифровых фотоаппаратах, мобильных телефонах и т. п. Файловая система определяет формат содержимого и способ физического хранения информации, которую принято группировать в виде файлов. Конкретная файловая система определяет размер имен файлов (и каталогов), максимальный возможный размер файла и раздела, набор атрибутов файла. Некоторые файловые системы предоставляют сервисные возможности, например, разграничение доступа или шифрование файлов.

**8.3 Выполнение работы**

После запуска системы от имени своей учетной записи и переключения на текстовую консоль, создадим родительский каталог, используя команду *mkdir*. После этого внутри каталога, созданного выше создадим структуру каталогов, представленную на рисунке, для этого также используем команду *mkdir*. Вывести на экран содержимое текущего каталога и убедиться, что все созданные каталоги созданы без ошибок можно с помощью утилиты *tree.*

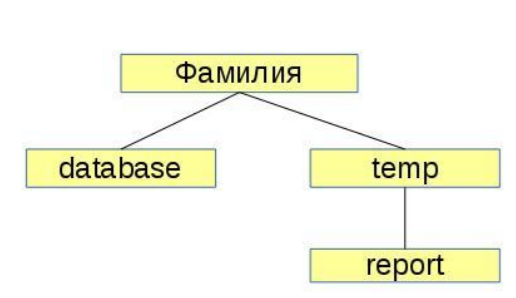


Рисунок 28 – Структура каталогов

Рисунок 29 – Структура каталогов

Перейдем в каталог temp с помощью команды *cd*. Убедимся, что он является текущим с помощью команды *pwd*. Выведем на экран содержимое каталога с помощью команды *ls.*

Внутри каталога temp создадим файл базы данных dataset1.txt с помощью команды *touch*. Заполним файл данными в соответствии с номером варианта задания с помощью команды *nano,* сохраним введенные записи с помощью *Ctrl+O*, вернемся к консоли с помощью *Ctrl+X* . В качестве разделителя столбцов данных в файле используем символ “;” без пробелов. Файл содержит 4 строки.

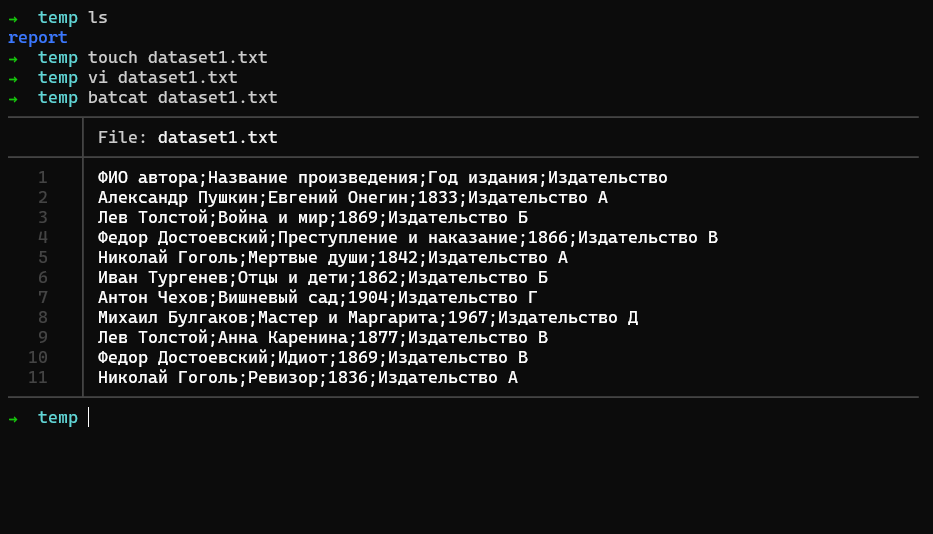


Рисунок 30 – Вывод содержимого каталога, создание и вывод содержимого файла database1

С помощью конвейера команд внутри каталога temp создадим файл базы данных dataset2.txt. Заполним файл данными в соответствии с номером варианта задания. В качестве разделителя столбцов данных в файле также используем символ “;” без пробелов. Файл содержит 5 строк. Данные должны отличаются от введенных ранее (см. Рис. 31).

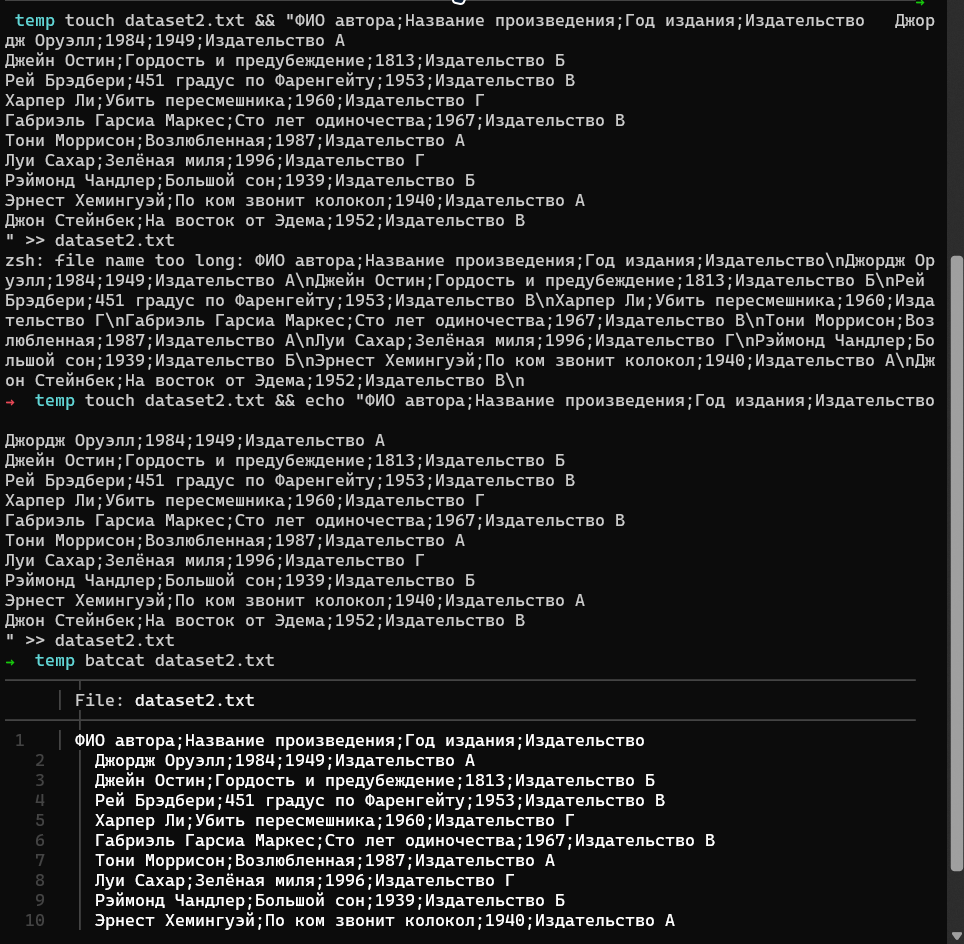


Рисунок 31 – Cоздание, заполнение и вывод содержимого файла database2 с помощью конвеера команд

С помощью перенаправления вывода в файл создадим файл базы данных dataset3.txt. Заполним файл данными в соответствии с номером варианта задания. В качестве разделителя столбцов данных в файле используем символ “;” без пробелов. Файл содержит 3 строки. Данные отличаются от введенных ранее (см.Рис.32).

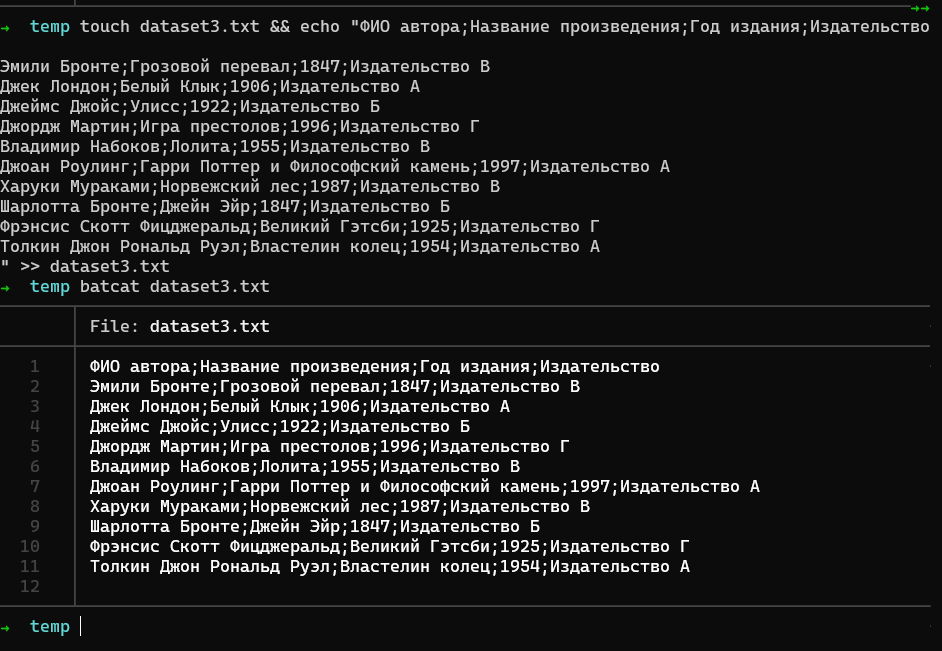


Рисунок 32 – Создание, заполнение и вывод содержимого файла database3

Выведем на экран содержимое всех созданных файлов базы данных (см. Рис.30-32).

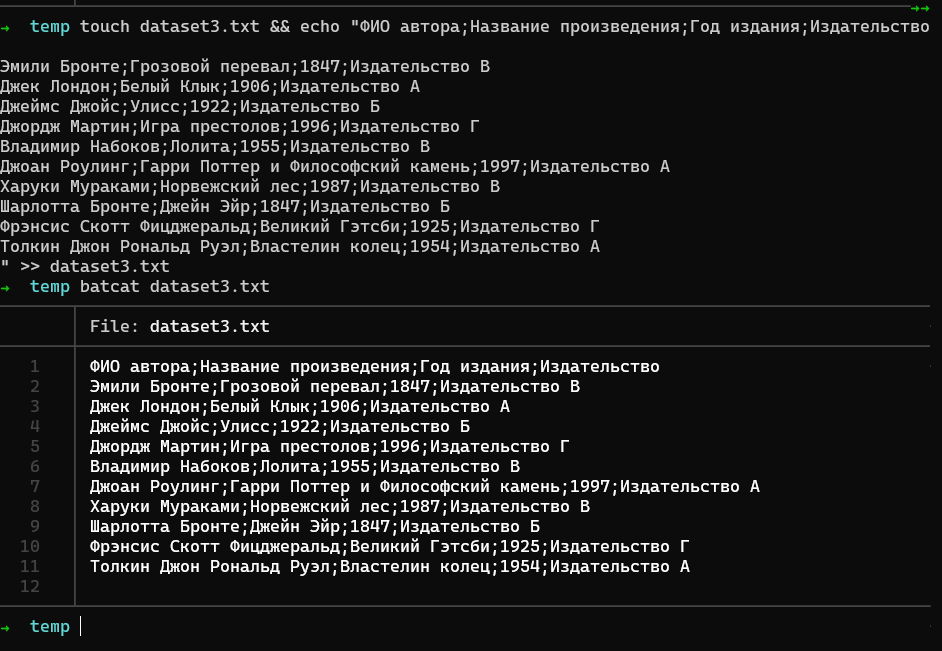


Рисунок 33 – Вывод содержимого файла database1

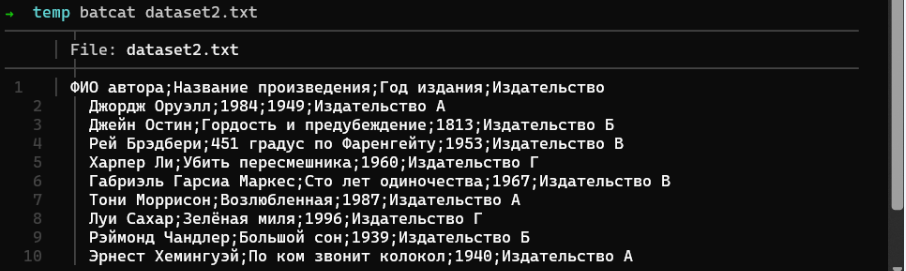


Рисунок 34 – Вывод содержимого файла database2

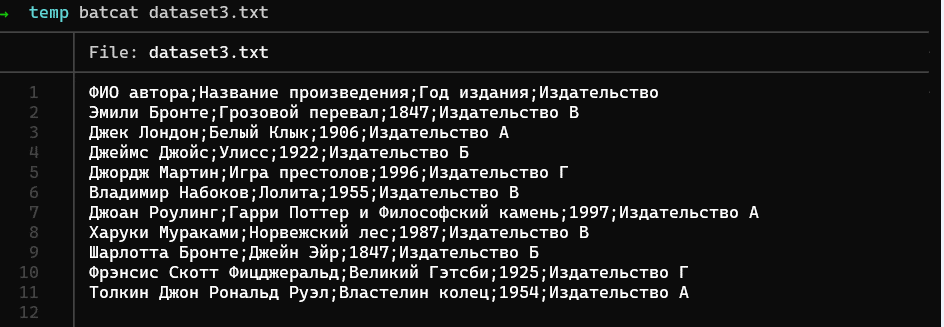


Рисунок 35 – Вывод содержимого файла database3

Объединим содержимое всех созданных файлов базы данных в один файл data.txt и поместим его в каталог /database (см.Рис.36).

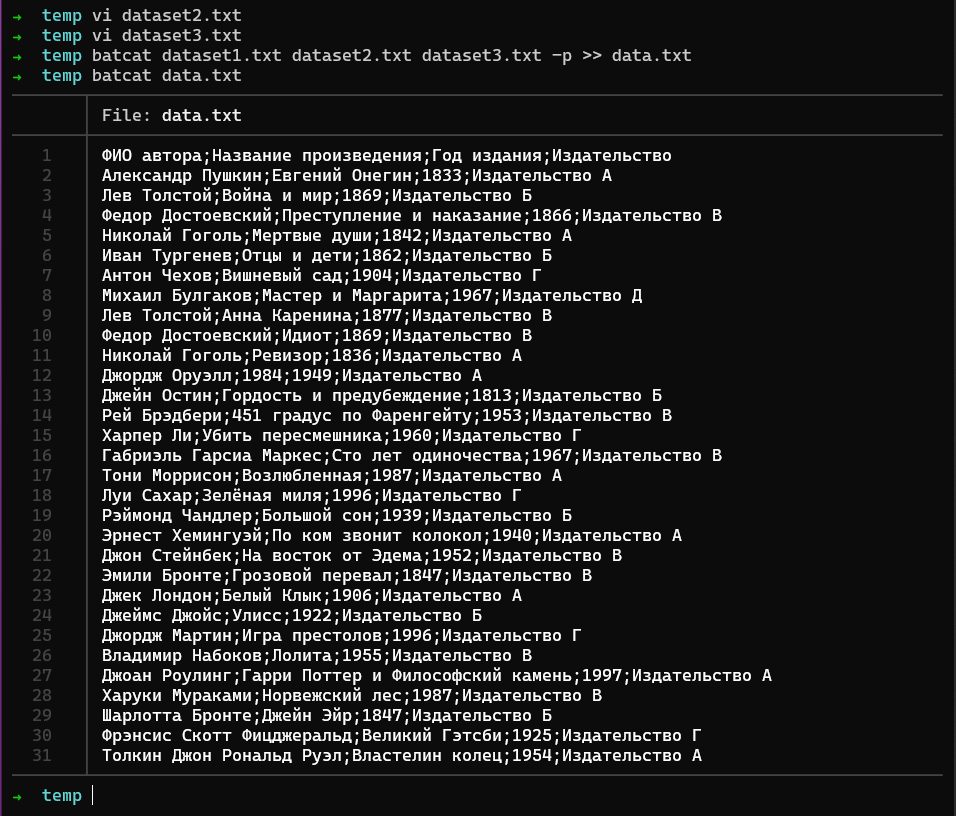


Рисунок 36 – Объединение содержимого в один файл и проверка объединения

Перейдем в каталог /database. Убедимся, что он является текущим. Выведем на экран содержимое каталога (см. Рис.38).

Рисунок 38 – Перенос файла и переход в каталог database

Подсчитаем количество строк файла data.txt. Результат подсчета выведем на экран и в файл отчета output.txt, расположенный в каталоге report (см. Рис.39).

Рисунок 39 – Подсчет количества строк

Дополним файл data.txt 2-я строками данных в соответствии с номером варианта задания. В качестве разделителя столбцов данных в файле используем символ “;” без пробелов (см Рис.40).



Рисунок 40 – Добавление строк

Повторно подсчитаем количество строк файла data.txt. Результат подсчета выведем на экран и допишем в конец файла отчета output.txt, расположенного в каталоге report (см. Рис.41).

Рисунок 41 – Подсчет количества строк и запись в файл

Осуществим фильтрацию данных файла data.txt в соответствии с номером варианта задания. Результат фильтрации выведем на экран и в файл отчета filtered.txt, расположенный в каталоге report. Повторим фильтрацию с различными значениями фильтра. Результаты фильтрации выведем на экран и допишем в файл отчета filtered.txt (см.Рис.42).

Рисунок 42 – Вывод фильтрации в консоль и запись в файл

Выполним сортировку содержимого файла data.txt в соответствии с номером варианта задания. Результат сортировки выведем на экран и в файл отчета sorted.txt, расположенный в каталоге report. (см. Рис.43).

Рисунок 43 – Выполнение сортировки и запись результата в файл sorted.txt

Выполним фильтрацию содержимого файла data.txt с сортировкой результата фильтрации. Фильтрацию и сортировку выполним в соответствии с номером варианта задания. Результат выведем на экран и в файл отчета filteredsorted.txt, расположенный в каталоге report (см. Рис.44).

Рисунок 45 – Фильтрация и сортировка

Выведем календарь на экран и в файл calendar.txt (см. Рис. 46-47).

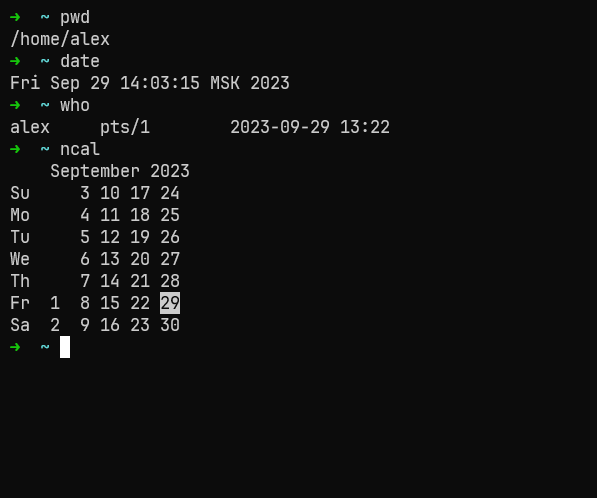


Рисунок 46 – Вывод календаря на экран, выполнение команд pwd, date, who.

Рисунок 47 – Запись календаря в файл.

# ****ЗАКЛЮЧЕНИЕ****

В результате проделанных работ, получены навыки работы в САПР QURTUS II, приобретены навыки проектирования логических схем при помощи графического редактора САПР QUARTUS II, приобретены основные навыки описания цифровых схем с помощью языка описания аппаратуры AHDL, приобретены навыки использования параметрических элементов (LPM function). Изучена работа основных логических элементов таких как: триггеры, дешифраторы, счетчики, компараторы. На практике изучено построение логической схемы счетчика с произвольным модулем счета, а также описание счетчика на языке AHDL. Получены навыки по установке, настройке и использованию операционной системы Linux, запускаемой на виртуальной машине. Изучены команды операционной системы, получены навыки по работы с файловой системой.

# ****СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ****

1. Методические указания к практическим работам (Quartus) https://online-edu.mirea.ru/pluginfile.php?file=%2F964885%2Fmod\_resource%2Fcontent%2F3%2FQuartus.pdf.

2. Методические указания к практическим работам (Виртуальная машина) https://online-edu.mirea.ru/pluginfile.php?file=%2F964880%2Fmod\_resource%2Fcontent%2F1%2F%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B%20%D0%92%D0%9C\_%D0%90%D1%80%D1%85%D0%92%D0%9C%D0%B8%D0%A1\_%D1%80%D0%B5%D0%B4\_%D0%9F%D0%BE%D0%BD%D0%BE%D0%BC%D0%B0%D1%80%D0%B5%D0%B2.pdf.

3. Мусихин А. Г., Смирнов Н. А. Архитектура вычислительных машин и систем [Электронный ресурс]: учебное пособие. - М.: РТУ МИРЭА, 2021. - – Режим доступа: https://ibc.mirea.ru/books/share/4180/

4. Мусихин А. Г., Смирнов Н. А. Архитектура вычислительных машин и систем [Электронный ресурс]: учебное пособие. - М.: РТУ МИРЭА, 2020. - – Режим доступа: https://library.mirea.ru/secret/16022021/2532.iso 2. Мусихин А. Г., Смирнов Н. А. Архитектура вычислительных машин и систем [Электронный ресурс]: методические рекомендации к контрольным работам. - М.: РТУ МИРЭА, 2020. - – Режим доступа: https://ibc.mirea.ru/books/share/3782/

5. Новожилов О. П. Архитектура ЭВМ и систем в 2 ч. Часть 1 [Электронный ресурс]: Учебное пособие для вузов. - Москва: Юрайт, 2021. - 276 с – Режим доступа: https://urait.ru/bcode/474545

6. Новожилов О. П. Архитектура ЭВМ и систем в 2 ч. Часть 2 [Электронный ресурс]: Учебное пособие для вузов. - Москва: Юрайт, 2021. - 246 с – Режим доступа: https://urait.ru/bcode/474546

7. Толстобров А. П. Архитектура ЭВМ [Электронный ресурс]: Учебное пособие для вузов. Москва: Юрайт, 2021. - 154 с – Режим доступа: https://urait.ru/bcode/476512

8. Степина В.В. Архитектура ЭВМ и вычислительные системы [Электронный ресурс]: Учебник. - Москва: ООО "КУРС", 2023. - 384 с. – Режим доступа: https://znanium.com/catalog/document?id=420774