Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Рязанский государственный радиотехнический университет  
имени В.Ф. Уткина

Кафедра ЭВМ

К защите

Руководитель работы:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ**

по дисциплине

**«Проектирование цифровых устройств»**

Тема:

«Проектирование микропроцессорных систем на основе ПЛИС»

Выполнил студент группы 045

Анохин В.А.

дата сдачи на проверку, подпись

Руководитель работы

доцент кафедры ЭВМ

Устюков Д.И.

оценка дата защиты, подпись

**Задание на курсовой проект**

**Задание 1.** Разработайте иерархический проект синтезированной микропроцессорной системы на основе ПЛИС по описанию, приведенному в теоретической части. В отчете опишите процесс и результаты разработки.

**Технические требования**

1. Требование по быстродействию. Любая команда должна выполняться за один период синхроимпульсов.

2. Разрядность команд - 16 бит, разрядность данных – 8 бит.

3. Методы адресации и состав системы команд.

3.1. Команды с непосредственной адресацией. Операнды - содержимое регистра и константа из команды. Состав команд: пересылка, суммирование, суммирование с учетом переноса, логические операции И, ИЛИ, сумма по модулю два.

3.2. Двухадресные команды с регистровой адресацией. Операнды - содержимое двух регистров. Состав команд подобен п.3.1.

3.3. Одноадресные команды циклических сдвигов.

3.4. Команды обращения к памяти с косвенной адресацией.

3.5. Команды безусловных и условных переходов по признакам нуля - zf и переноса - cf с прямой адресацией.

4. Тип ПЛИС - семейство FLEX 10K.

**Задание 2.** Разработайте программу для тестирования логических операций с непосредственной адресацией. Выполните моделирование. Определите временные задержки формирования адреса команд, чтения кода команды, а также формирования результата операции на шине данных.

**Задание 3.** Создайте в памяти, начиная с адреса 00, массив из 8 чисел W0 – W7, которые вычисляются в соответствии с формулой: W =2×k +3;

Определите экспериментально максимальную частоту синхронизации.

**Задание 4.** Включите логический сдвиг влево содержимого регистра в систему команд, представьте в отчете результаты тестирования.

**Задание 5.** Добавить в данную систему трёхадресные команды.

**Содержание**

[Введение 4](#_Toc163268559)

[1 Выбор архитектуры микропроцессорной системы и конфигурации процессорного ядра 5](#_Toc163268560)

[2 Разработка системы команд 7](#_Toc163268561)

[2.1 Команды с непосредственной адресацией 7](#_Toc163268562)

[2.2 Двухадресные команды с регистровой адресацией 8](#_Toc163268563)

[2.3 Одноадресные команды с регистровой адресацией 9](#_Toc163268564)

[2.4 Команды обращения к памяти с косвенной адресацией 9](#_Toc163268565)

[2.5 Команды ветвления с прямой адресацией 10](#_Toc163268566)

[3 Разработка системы синхронизации 11](#_Toc163268567)

[4 Разработка проекта микропроцессорной системы 13](#_Toc163268568)

[4.1 Модуль Control 13](#_Toc163268569)

[4.2 Модуль РОН 13](#_Toc163268570)

[4.3 Модуль АЛУ 14](#_Toc163268571)

[4.4 Устройство синхронизации записи данных 15](#_Toc163268572)

[4.5 Регистр состояния 15](#_Toc163268573)

[6. Экспериментальная часть 18](#_Toc163268574)

[Заключение 20](#_Toc163268575)

[Список использованной литературы 21](#_Toc163268576)

# Введение

Проектирование современных цифровых вычислительных устройств тесно связано с использованием программируемых логических интегральных схем (ПЛИС). Эти универсальные компоненты открывают широкие возможности для создания интегрированных систем на кристалле (SoC), которые объединяют в себе различные функциональные блоки, такие как процессорные ядра, память, модули обработки данных и интерфейсные средства.

Синтезированные процессоры, разработанные с помощью средств САПР, могут значительно превосходить универсальные микропроцессоры в компактности и эффективности, особенно при выполнении специализированных задач. Использование таких решений позволяет оптимизировать затраты на аппаратное обеспечение и улучшить производительность системы.

Синтез микропроцессоров на ПЛИС представляет собой одну из наиболее интересных задач в области цифрового проектирования.

В данной курсовой работе основной целью является разработка микропроцессорной системы на базе ПЛИС, а также проверка её возможностей и работоспособности.

# Выбор архитектуры микропроцессорной системы и конфигурации процессорного ядра

Архитектура микропроцессорной системы отражает логическое построение системы, устройства и связей между ними.

Существуют две архитектуры для разработки МП. Джона фон Неймана и Гарвардская. Их основное различие в том, что в первой связь между процессором, устройствами памяти, ввода и вывода выполняет общая системная шина, которая содержит группы проводников - шины адреса, данных и управления. А во второй для хранения команд и данных используются независимые адресные пространства, полученные в результате использования отдельных устройств памяти и отдельных системных шин. В гарвардской архитектуре для хранения команд и данных можно использовать блоки памяти различных типов, емкости и разрядности. В микроконтроллерах для хранения программы, которая в процессе работы изменяться не должна, используют ПЗУ, а для хранения данных – ОЗУ

Для микропроцессорной системы выбрана гарвардская архитектура (рисунок 1). С устройствами памяти команд и данных процессор связан посредством четырех отдельных шин. Это шина адреса команд **ak**, шина команды **k**, шина адреса данных **ad** и шина данных **d-bus**.

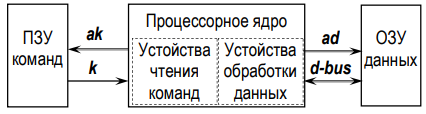


Рисунок 1 - Гарвардская архитектура микропроцессорной системы

Конфигурация процессорного ядра основана на совместном использовании арифметико-логическое устройства (АЛУ) и блока регистров общего назначения (РОН). Блок РОН выполнен как двухадресная (двухпортовая) память, которая позволяет одновременно выдать в АЛУ содержимое двух регистров - rx и ry, в которых хранятся первый и второй операнды двухадресной команды. Для одноадресных и двухадресных команд регистр первого операнда rx впоследствии используется как получатель результата и исходное значение первого операнда теряется. Для части двухадресных команд доступны трёхадресные варианты команд, в которых указывается регистр для записи. Содержимое регистра второго операнда ry проходит через сдвигающее устройство для реализации возможности использования команд с масштабированием. Код операции и адреса регистров заданы в команде k. Результат операции, полученный в АЛУ, через шину данных d\_bus передается на блок РОН для записи в rx и на другие устройства процессора.

Пример использования архитектуры АЛУ и блока РОН представлен на рисунке 2.

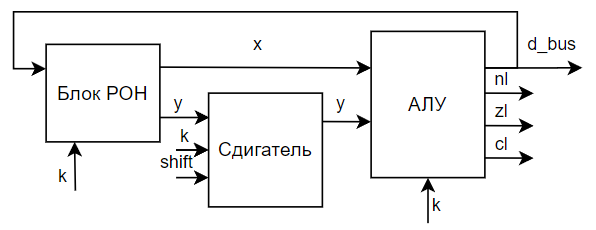


Рисунок 2 - Конфигурация процессора с блоком РОН

# Разработка системы команд

Благодаря тому, что мы используем гарвардскую архитектуру у нас есть возможность без сложностей реализовать RISС систему команд. В RISC все команды имеют одинаковую разрядность и выполняются за один машинный цикл.

Для разрабатываемого процессора выберем два варианта деления команды на поля. Первый вариант – два 4-разрядных и одно 8-разрядное поле (например, команды с непосредственной адресацией). При этом код команды в 16-ричной системе счисления содержит 4 цифры, 4-разрядному полю соответствует одна цифра, а байт кодируют две цифры. Второй вариант – команда содержит 4 поля, по 4 разряда (пример – команды с регистровой адресацией).

## 2.1 Команды с непосредственной адресацией

Структура команд с непосредственной адресацией представлена на рисунке 3.



Рисунок 3 - структура команд с непосредственной адресацией

k1 - код операции;

rx - номер регистра первого операнда, в который после выполнения операции записывается результат;

d8 –байт данных - второй операнд, поступающий из команды.

Поле k1 должно иметь признак команд с непосредственной адресацией, например, старший бит, равный нулю. В этом случае значения кода k1 будут принадлежать диапазону от 0 до 7, а возможное количество команд с непосредственной адресацией составит 8

В таблице 1 указан список команд с непосредственной адресацией.

Таблица 1 – Команды с непосредственной адресацией.

|  |  |  |  |
| --- | --- | --- | --- |
| k1 | Команда | Операция | Пояснение |
| 0 | movi rx, #d8 | rx = d8 | Пересылка в регистр константы |
| 1 | addi rx, #d8 | rx = rx + d8 | Суммирование |
| 2 | subi rx, #d8 | rx = rx + d8 | Вычитание |
| 3 | andi rx, #d8 | rx = rx \*d8 | Логическая операция И |
| 4 | ori rx, #d8 | rx = rx ∧ d8 | Логическая операция ИЛИ |
| 5 | xori rx, #d8 | rx = rx ⊕ d8 | Логическая операция Исключающее ИЛИ |

## 2.2 Дву**хадресные команды с регистровой адресацией**

Структура двухадресных команд с регистровой адресацией представлена на рисунке 4.



Рисунок 4 - структура двухадресных команд с регистровой адресацией

k1 - код операции равный 8;

rx - номер регистра первого операнда, в который после выполнения операции записывается результат;

ry – номер регистра второго операнда

k2 – код исполняемой операции

Таблица 2 – Двухадресные команды с регистровой адресацией.

|  |  |  |  |
| --- | --- | --- | --- |
| k2 | Команда | Операция | Пояснение |
| 0 | add rx, ry | rx = rx + ry | Суммирование |
| 1 | sub rx, ry | rx = rx - ry | Вычитание |
| 2 | mul rx, ry | rx = rx \* ry | Умножение |
| 3 | div rx, ry | rx = rx / ry | Деление |
| 4 | and rx, ry | rx = rx \*ry | Логическая операция И |
| 5 | or rx, ry | rx = rx ∧ ry | Логическая операция ИЛИ |

Таблица 2 – Двухадресные команды с регистровой адресацией (продолжение).

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | xor rx, ry | rx = rx ⊕ ry | Логическая операция Исключающее ИЛИ |
| 8 | mov rx, ry | rx = ry | Пересылка в регистр из регистра |

## 2.3 Трёхадресные команды

Для двухадресных команд с регистровой адресацией с кодами k1 0-7 доступны трёхадресные команды и команды с масштабированием (таблица 3).

Структура трёхадресных команд и команд с масштабированием представлена на рисунке 4.

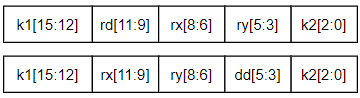


Рисунок 5 – Структура трехадресных команд

k1 – код операции (таблица 3);

rd – номер регистра получателя;

rx – номер регистра первого операнда;

ry – номер регистра второго операнда

dd – величина сдвига от 0 до 7

k2 – код исполняемой операции

Таблица 3 – Дополнительные вариации для двухадресных команд.

|  |  |  |  |
| --- | --- | --- | --- |
| k2 | k1 = b | k1 = c | k1 = d |
| 0 | add rd, rx, ry | add rx, ry, lsl d3 | add rx, ry, lsr d3 |
| 1 | sub rd, rx, ry | sub rx, ry, lsl d3 | sub rx, ry, lsr d3 |
| 2 | mul rd, rx, ry | mul rx, ry, lsl d3 | mul rx, ry, lsr d3 |
| 3 | div rd, rx, ry | div rx, ry, lsl d3 | div rx, ry, lsr d3 |
| 4 | and rd, rx, ry | and rx, ry, lsl d3 | and rx, ry, lsr d3 |
| 5 | or rd, rx, ry | or rx, ry, lsl d3 | or rx, ry, lsr d3 |
| 6 | xor rd, rx, ry | xor rx, ry, lsl d3 | xor rx, ry, lsr d3 |

## 2.4 Одноадресные команды с регистровой адресацией

Структура одноадресных команд с регистровой адресацией представлена на рисунке 6.



Рисунок 6 - одноадресных команд с регистровой адресацией

k1 - код операции равный 9;

rx - номер регистра первого операнда

k2 – код исполняемой операции

Таблица 4 – Одноадресные команды с регистровой адресацией.

|  |  |  |  |
| --- | --- | --- | --- |
| k1 | Команда | Операция | Пояснение |
| 0 | inc rx | rx = rx + 1 | Инкремент |
| 1 | dec rx | rx = rx - 1 | Декремент |
| 2 | not rx | rx = !rx | Отрицание |
| 3 | asr rx | rx(n) = rx(n + 1) C = r(0) n = 0..6 | Логическая сдвиг вправо |
| 4 | lsl rx | rx(n + 1) = rx(n), rd(0) = 0 | Логический сдвиг влево |

## 2.5 Команды обращения к памяти с косвенной адресацией

Структура команд обращения к памяти представлена на рисунке 7.



Рисунок 7 – структура команд обращения к памяти

k1 - код операции равный 0xA;

rx - номер регистра, в котором находится операнд;

ra - номер регистра, в котором находится адрес ячейки;

k2 – код исполняемой операции;

Таблица 5 – Команды обращения к памяти

|  |  |  |  |
| --- | --- | --- | --- |
| k1 | Команда | Операция | Пояснение |
| 0 | ld rx, [ra] | rx = mem[ra] | Загрузка в rx из памяти |
| 1 | st rx, [ra] | mem[ra] = rx | Сохранение в памяти |

## 2.6 Команды ветвления с прямой адресацией

Структура команд ветвления с прямой адресацией представлена на рисунке 8.

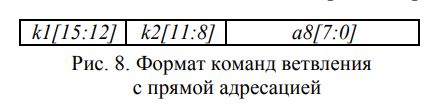


Рисунок 8 - Структура команд ветвления с прямой адресацией

k1 - код операции равный 0xF;

a8 – адрес перехода;

k2 – код исполняемой операции;

Таблица 6 – Одноадресные команды с регистровой адресацией.

|  |  |  |  |
| --- | --- | --- | --- |
| k1 | Команда | Операция | Условие |
| 0 | b a8 | PC = a8 | Безусловный переход |
| 1 | beq a8 | Z = 1, то PC = a8 | Результат 0 |
| 2 | bne a8 | Z = 0, то PC = a8 | Результат 1 |
| 3 | bcs a8 | C = 1, то PC = a8 | Был перенос |
| 4 | bcc | C = 0, то PC = a8 | Не было переноса |
| 5 | bpl a8 | N = 0, то PC = a8 | Результат > 0 |
| 6 | bmi a8 | N = 1, то PC = a8 | Результат < 0 |

# Разработка системы синхронизации

Система синхронизации обеспечивает устранение ошибок, обусловленных гонками, возникающими при выполнении операций. Работа процессора сводится к циклическому исполнению определенных операций, которым должны соответствовать определенные событиям и состояниям синхросигнала.

Период синхросигнала содержит два состояния (clk = 0 и clk = 1) и два изменения состояний – события - (спад и фронт импульса) (Рисунок 9). Примем за начало цикла команды (машинного цикла), который должен составлять один период синхросигнала, спад синхроимпульса. Выполнение команды во времени представим в виде последовательности следующих действий.

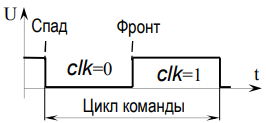


Рисунок 9 – Импульсы синхронизации

1. Спад синхросигнала clk - запись адреса следующей команды с выхода программного счетчика в регистр адреса ПЗУ. С этого момента начинается цикл выполнения команды, при этом команда k содержится на шине команд в течение всего периода синхросигнала и подается в блоки АЛУ, РОН, устройства управления и синхронизации.

2. Состояние clk = 0. Устройства и блоки обработки данных выполняют операцию, предусмотренную в команде под управлением кодов, содержащиеся в определенных полях команды, которые выбираются при разработке форматов команд и учитываются в последствии при разработке принципиальной схемы.

На блок РОН поступают адреса регистров, используемых в данной команде, в соответствии с которой выдаются операнды, содержащиеся в rx и в ry.

На входы АЛУ будут поступать все данные, необходимые для выполнения операции – операнды из блока РОН и код операции из команды. На выходе АЛУ, и на шине «d\_bus», будет сформирован результат выполнения команды.

Схема управления переходами формирует адреса новой команды для записи в программный счетчик.

Схема синхронизации записи сформирует сигналы разрешения записи результата операции в те устройства памяти и регистры, которые предусмотрены в данной команде.

Обработку данных выполняют устройства комбинационного типа (АЛУ, мультиплексоры, дешифраторы), которые не связаны с импульсами синхронизации, работают асинхронно. Формирование результата будет сопровождаться задержками сигналов в логических элементах.

3. Фронт синхросигнала clk - момент записи результата в регистр, или память, запись в программный счетчик адреса следующей команды, запись признаков результата в регистр состояния. Модуль разрешения записи формирует сигналы разрешения записи данных, полученных на шине «d\_bus» при выполнении текущей команды, только в те устройства, для которых эта запись предусмотрена при проектировании модуля.

4. Интервал С=1. Запить адреса следующей команды, поступающего с выхода программного счетчика в регистр адреса ПЗУ.

Выполнение команды за один такт синхроимпульса становится возможным, если выполнять операции записи синхронно с фронтом или спадом импульсов, а все остальные операции выполнять посредством комбинационных схем, которые не содержат элементов памяти и работают асинхронно.

# Разработка проекта микропроцессорной системы

Функциональная схема представлена на рисунке 10.

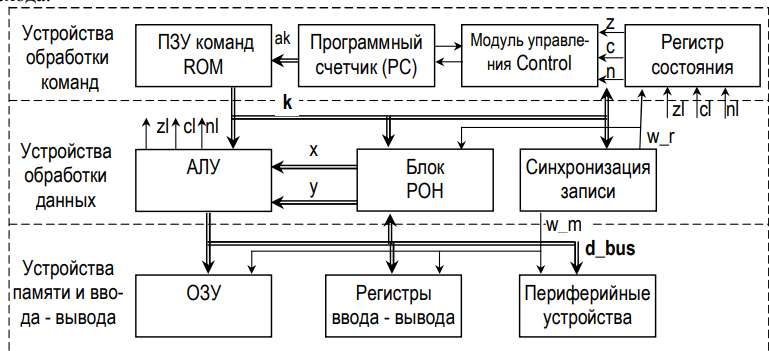


Рисунок 10 – Функциональная схема

## 4.1 Модуль Control

Модуль Control формирует адрес следующей команды, в зависимости от признаков и сигналов от модуля прерываний о сохранении и загрузке состояния.

module control(clk, k, c, z, n, branch, q\_pc);

input clk, c, z, n;

input [15:0] k;

output [7:0] q\_pc; reg [7:0] q\_pc;

output branch;

assign branch =

(k[15:12]==4'hf)&((k[11:8]==4'h0)|

(k[11:8]==1)& z| (k[11:8]==2)& ~z|

(k[11:8]==3)& c| (k[11:8]==4)&~c|

(k[11:8]==5)& n|(k[11:8]==6)& ~n);

always @ (posedge clk)

if (branch) q\_pc = k;

else q\_pc = q\_pc+1;

endmodule

## 4.2 Модуль РОН

В разрабатываемом микропроцессоре используется двухадресный (двухпортовый) блок регистров общего назначения (РОН), имеющий режим одновременного чтения содержимого двух регистров.

module blok\_ron(c,wreg,d\_bus,k, x,y);

input c,wreg;

input [7:0] d\_bus;

input [15:0] k;

output [7:0] x,y;

reg [7:0] x,y;

reg [7:0] ron [15:0];

always begin

if (k[15:12] <= 4'ha) begin

x <= ron[k[11:8]];

y <= (k[15]==0 | k[15:12]==9) ? 8'h00 : ron[k[7:4]];

end

else if (k[15:12]==4'hb) begin

x <= ron[k[8:6]];

y <= ron[k[5:3]];

end

else if (k[15:12]==4'hc | k[15:12]==4'hd) begin

x <= ron[k[11:9]];

y <= ron[k[8:6]];

end

else begin

x <= 0;

y <= 0;

end

end

always @(posedge c)

if (wreg) begin

if (k[15:12]==4'hb | k[15:12]==4'hc | k[15:12]==4'hd) ron[k[11:9]] = d\_bus;

else ron[k[11:8]] = d\_bus;

end

endmodule

## 4.3 Модуль shifter

Данный модуль выполняет сдвиг регистра y для команд с масштабированием.

module shifter(in,k,shift,out);

input [7:0] in;

input [3:0] k;

input [2:0] shift;

output [7:0] out;

assign out = (k==4'hc) ? in << shift :

(k==4'hd) ? in >> shift :

in;

endmodule

## 4.3 Модуль АЛУ

Модуль АЛУ выполняет арифметические и логические команды обработки данных с непосредственной и регистровой адресацией. Кроме того, через данный модуль выполняются команды обращения к ОЗУ.

module alu (k,x,y,dm, d\_bus, cl,zl,nl);

input [15:0] k;

input [7:0] x, y, dm;

output [7:0] d\_bus; reg [7:0] d\_bus;

output cl; reg cl; output zl, nl;

always if (k[15]==0) case(k [14:12])

0: {cl,d\_bus} = {1'b0,k [7:0]};

1: {cl,d\_bus} = x + k [7:0];

2: {cl,d\_bus} = x - k [7:0] ;

3: {cl,d\_bus} = {1'b0,( x & k [7:0])};

4: {cl,d\_bus} = {1'b0,( x | k [7:0])};

5: {cl,d\_bus} = {1'b0,( x ^ k [7:0])};

default: {cl,d\_bus} = 0;

endcase else

if (k[15:12] == 8 & k[3]==1) case (k [2:0])

0: d\_bus = y;

default: {cl,d\_bus} = 0;

endcase else

if (k[15:12] == 8 | k[15:12]==4'hb | k[15:12]==4'hc | k[15:12]==4'hd) case (k [2:0])

0: {cl,d\_bus} = x + y;

1: {cl,d\_bus} = x - y;

2: {cl,d\_bus} = {1'b0, x \* y};

3: {cl,d\_bus} = {1'b0, x / y};

4: {cl,d\_bus} = {1'b0,( x & y)};

5: {cl,d\_bus} = {1'b0,( x | y)};

6: {cl,d\_bus} = {1'b0,( x ^ y)};

default: {cl,d\_bus} = 0;

endcase else

if (k[15:12] == 9) case (k [2:0])

0: {cl,d\_bus} = {1'b0,( x+1)};

1: {cl,d\_bus} = {1'b0,( x-1)};

2: {cl,d\_bus} = {1'b0,( ~x)};

3: {d\_bus,cl} = {x[7], x[7:0]};

4: {cl,d\_bus} = {x,1'b0};

default: {cl,d\_bus} = 0;

endcase else

if (k[15:12]==4'ha) case (k[0])

0: d\_bus = dm;

1: d\_bus = x;

endcase

else d\_bus = 0;

assign zl = (d\_bus==0);

assign nl = d\_bus[7];

endmodule

## 4.4 Устройство синхронизации записи данных

Результат операции, выполненной, а АЛУ, выдается на шину выходных данных d\_bus, к которой подключаются различные устройства памяти, принимающие данные, это РОН, ОЗУ, другие периферийные устройства. Данные подаются на все устройства. Однако, запись результата выполняется только в определенное устройство, в соответствии с кодом команды. Устройство синхронизации записи подобно дешифратору, на вход которого поступают коды команд, а выходами являются сигналы разрешения записи в регистры (wreg), и в память (wmem).

module sync\_wr (k, wreg, wmem);

input [15:0] k ;

output wreg, wmem ;

assign wreg = k[15:12] <= 4'hd & ~wmem;

assign wmem = k[15] & ~k[14] & k[13] &~ k[12]& k[0];

endmodule

## 4.5 Регистр состояния

Хранит признаки нуля (z), отрицательного результата (n) и переноса (z) а также по команде от контроллера прерываний сохраняет все признаки.

module status\_reg(clk, wreg,zl,cl,nl,z,c,n);

input clk, wreg, nl, zl, cl;

output n; reg n;

output z; reg z;

output c; reg c;

always @ (posedge clk)

if (wreg==1) begin

n=nl; z=zl; c=cl;

end

endmodule

На рисунке 11 показана схема микропроцессорной системы в среде Quartus II.

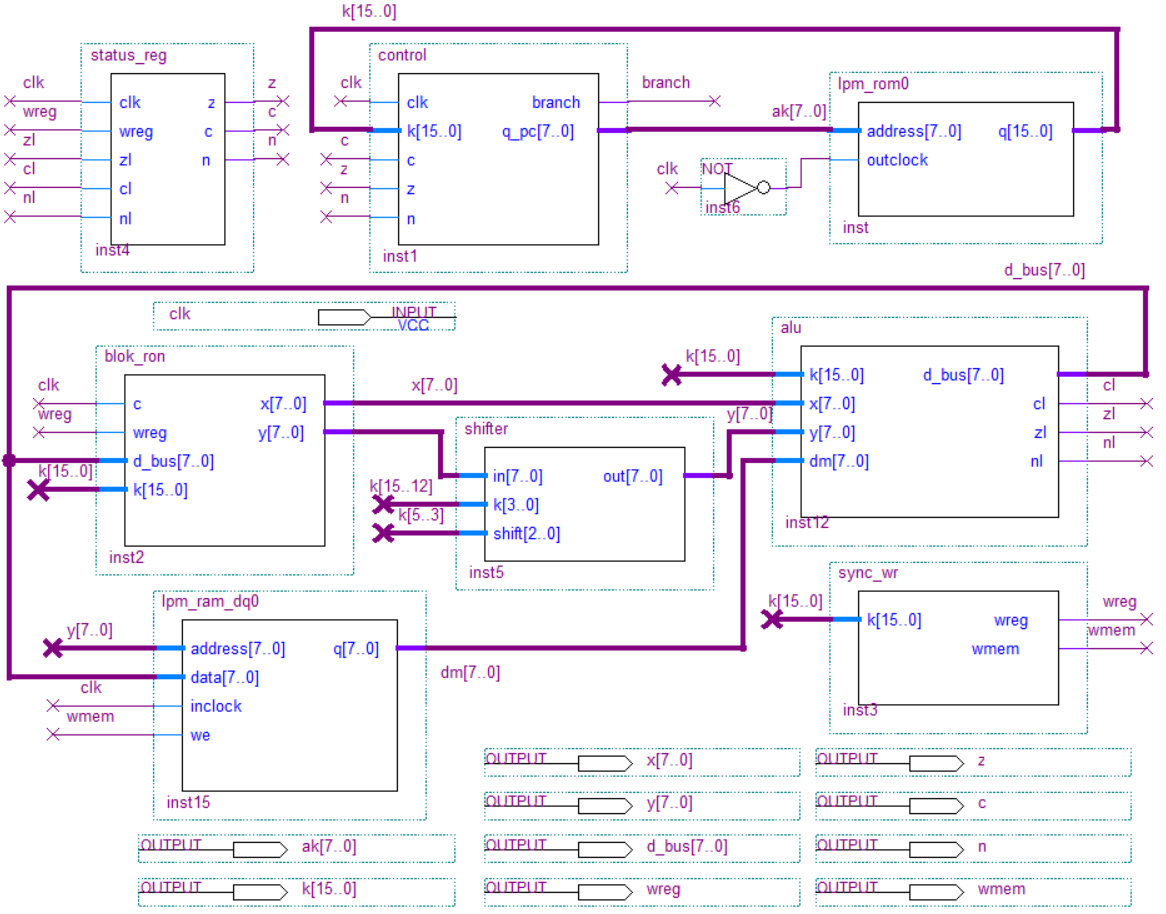


Рисунок 11 – Схема микропроцессорной системы

# 5. Экспериментальная часть

**Задание 1.** Разработайте иерархический проект синтезированной микропроцессорной системы на основе ПЛИС по описанию, приведенному в теоретической части.

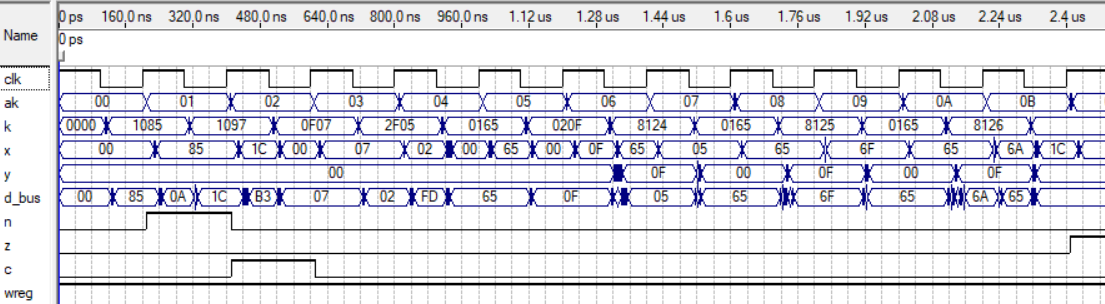


Рисунок 12 – Результат выполнения тестовой программы

**Задание 2.** Разработайте программу для тестирования логических операций с непосредственной адресацией. Выполните моделирование. Определите временные задержки формирования адреса команд, чтения кода команды, а также формирования результата операции на шине данных.

Таблица 7 – Программа тестирования логических операций с непосредственной адресацией.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ak | Команда | k | d\_bus | nzc |
| 00 | movi r0, 0F | 000F | 0F | 000 |
| 01 | andi r0, 3C | 303C | 0C | 000 |
| 02 | ori r0, 68 | 4068 | 6C | 000 |
| 03 | xori r0, 6C | 506C | 00 | 010 |

На рисунках 13 – 16 показаны результаты работы:

1. Формирование адреса: 9,8 нс
2. Чтение команды: 13,8 нс
3. Получение данных на шине: 26.1 – 13,8 = 12,3 нс

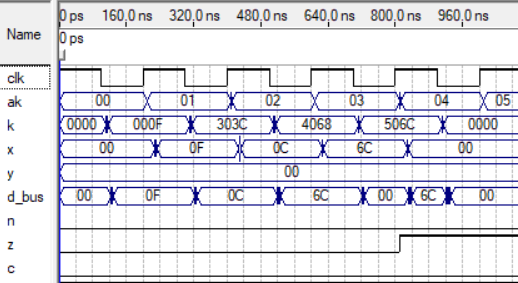


Рисунок 13 – Результат моделирования

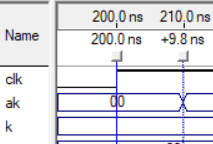


Рисунок 14 – Задержка формирования адреса команд

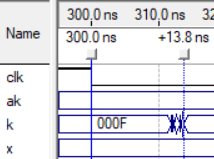


Рисунок 15 – Время чтения команды

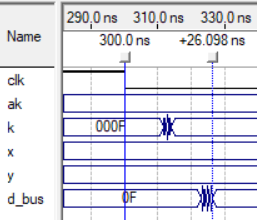


Рисунок 16 – Время формирования данных на шине d\_bus

**Задание 3.** Создайте в памяти, начиная с адреса 00, массив из 8 чисел W0 – W7, которые вычисляются в соответствии с заданной формулой:

W =2×k +3;

Определите экспериментально максимальную частоту синхронизации.

Пусть r0 – счетчик циклов. r1 – аккумулятор. r2 – разность прогрессии, r3 – указатель адреса в памяти.

Таблица 8 – Программа тестирования записи в память

|  |  |  |
| --- | --- | --- |
| ak | Команда | k |
| 00 | movi r0, #8 | 0008 |
| 01 | movi r1, #3 | 0103 |
| 02 | movi r2, #2 | 0202 |
| 03 | movi r3, #0 | 0300 |
| 04 | st r1, [r3] | a131 |
| 05 | add r1, r2 | 8120 |
| 06 | inc r3 | 9300 |
| 07 | dec r0 | 9001 |
| 08 | beq 0a | f10a |
| 09 | b 04 | f004 |
| 0a | b 0a | f00a |

Установим период равный 100 нс (рисунки 17 – 18).

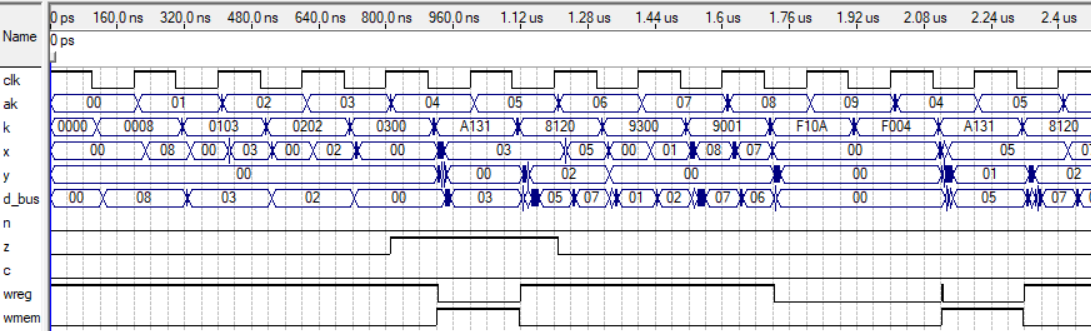


Рисунок 17 – Моделирование программы при 100нс

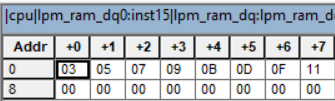


Рисунок 18 – Результат выполнения программы при 100 нс

Результат совпадает с ожидаемым. Уменьшим до 50 нс (рисунки 19 – 20).

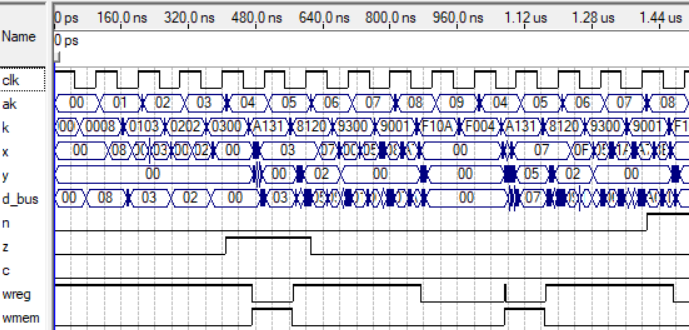


Рисунок 19 – Моделирование программы при 50нс

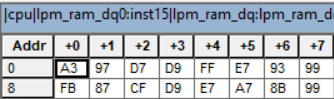


Рисунок 20 – Результат выполнения программы при 50 нс

Результат не совпадает с ожидаемым. Увеличим до 60 нс (рисунки 21 – 22).

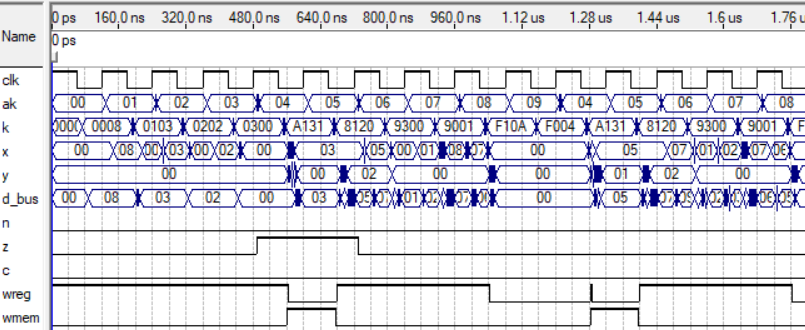


Рисунок 21 – Моделирование программы при 60 нс

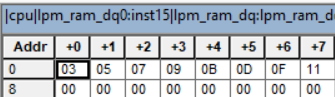


Рисунок 22 – Результат выполнения программы при 60 нс

Экспериментальным путем выяснили максимальную частоту работы микропроцессора при выполнении этой программы. Она составляет 16МГц.

**Задание 4.** Включите логический сдвиг влево содержимого регистра в систему команд, представьте в отчете результаты тестирования.

Для реализации логического сдвига влево была добавлена команда 4: {cl,d\_bus} = {x,1'b0}; в модуль АЛУ в блок команд под номером 9.

Таблица 9 – Тестирование логического сдвига влево.

|  |  |  |
| --- | --- | --- |
| ak | Команда | k |
| 00 | movi r0, 03 | 0003 |
| 01 | lsl r0 | 9004 |
| 02 | b 1 | f001 |

Ожидаемый результат: 3, 6, С, 18, 30, 60, С0.

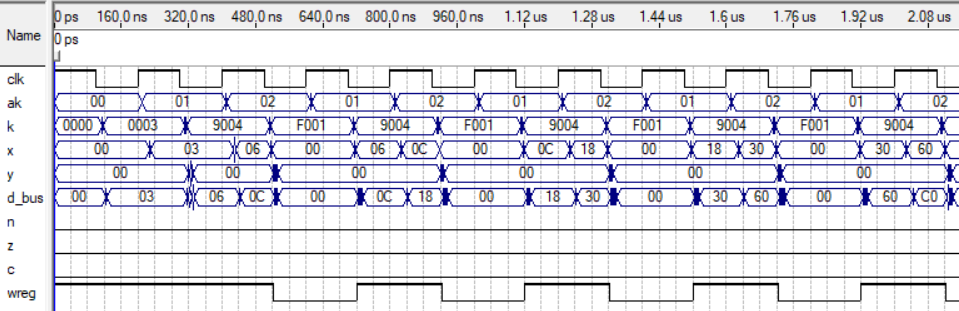


Рисунок 23 – Результат выполнения программы

Результат выполнения программы совпадает с ожидаемым, операция работает корректно.

**Задание 5.**

Для проверки работы трёхадресных команд напишем программу (таблица 10), в которой содержатся основные добавленные команды. На рисунке 24 показан результат выполнения программы.

Таблица 10 – Программа проверки трёхадресных команд.

|  |  |  |  |
| --- | --- | --- | --- |
| ak | Команда | k | d\_bus |
| 00 | movi r0, 05 | 0005 | 05 |
| 01 | movi r1, 02 | 0102 | 02 |
| 02 | add r2, r0, r1 | b408 | 07 |
| 03 | sub r3, r0, r1 | b609 | 03 |
| 04 | mul r4, r2, r3 | b89a | 15 |
| 05 | div r5, r4, r0 | bb03 | 04 |
| 06 | add r0, r1, lsl 2 | c050 | 0d |
| 07 | sub r0, r4, lsr 1 | d109 | 3 |

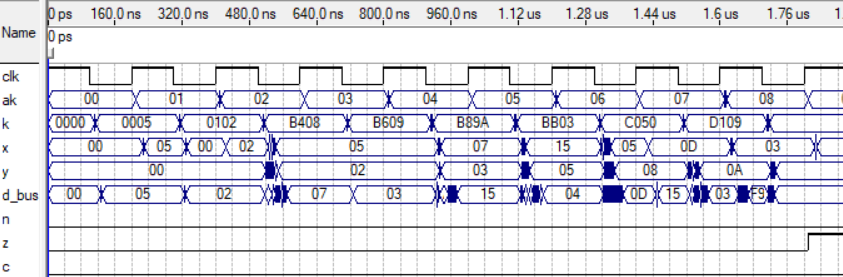


Рисунок 24 – Результат работы программы

Результат выполнения программы совпадает с ожидаемым, трёхадресные операции работают корректно.

# Заключение

В ходе курсовой работы была разработана и проверена микропроцессорная система на основе ПЛИС. В полученном микропроцессоре используется гарвардская архитектура, RISC команды с разрядностью данных 8 бит и разрядностью команд 16 бит. Дополнительно были добавлены трёхадресные команды.

# Список использованной литературы

1.Корнеев В. В., Киселев А. В. Современные микропроцессоры. — 3-е изд., перераб, и доп. — СПб.: БХВ - Петербург, 2003. - 448 е.: ил.

2.Стешенко В.Б. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания. \_М.: Издательский дом «Додэка-XXI» 2007.-124c.

4.Зотов В. Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы XILINX. - М.: Горячая линия - Телеком, 2006. - 520 с, ил.

5.В. В. Соловьев. Основы языка проектирования цифровой аппаратуры VERILOG. — М.: Горячая линия-Телеком, 2014. — 205 с: ил.