МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Рязанский государственный радиотехнический университет
Кафедра ЭВМ
А. В. Кистрин

ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ НА ОСНОВЕ ПЛИС

Методические указания к курсовому проектированию,
и к лабораторным работам
Рязань 2023

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 команд, выполняющих заданные операции. Выполните моделирование. Определите временные задержки формирования адреса команд, чтения кода команды, а также формирования результата операции на шине данных. Номер задачи – из таблицы вариантов.

0. Арифметические операции с непосредственной адресацией.
1. Арифметические операции с регистровой адресацией.
2. Логические операции с непосредственной адресацией.
3. Логические операции с регистровой адресацией.
4. Команды обращения к памяти.
5. Команды условных и безусловных переходов.
6. Команды сдвигов.

Задание 3. Создайте в памяти, начиная с адреса 00, массив из 8 чисел $W_0 – W_7$, которые вычисляются в соответствии с заданной формулой:

0. $W_k = 2 \cdot k + 5$; 1. $W_k = 2 \cdot k + 4$; 2. $W_k = 2 \cdot k + 3$; 3. $W_k = 3 \cdot k + 1$; 4. $W_k = 3 \cdot k + 2$.

Определите экспериментально максимальную частоту синхронизации.

Задание 4. Включите дополнительные команды в систему команд, представьте в отчете результаты тестирования.

0. Инкремент содержимого регистра ***inc rx***.
1. Декремент содержимого регистра ***dec rx***.
2. Логический сдвиг влево содержимого регистра ***lsl rx***.
3. Логический сдвиг вправо содержимого регистра ***lsr rx***.
4. Арифметический сдвиг вправо содержимого регистра ***asr rx***.
5. Перестановка тетрад в байте заданного регистра ***swap rx***.

Варианты заданий 2, 3, 4 определяется номером студента в журнале группы. Содержание отдельных заданий определяются указанными далее номерами.

Вариант по журналу	1	2	3	4	5	6	7	8	9	10	11	12
Вариант задачи 2	0	1	2	3	4	5	6	0	1	2	3	4
Вариант задачи 3	0	1	2	3	4	0	1	2	3	4	1	2
Вариант задачи 4	0	1	2	3	4	5	0	1	2	3	4	5

Вариант по журналу	13	14	15	16	17	18	19	20	21	22	23	24
Вариант задачи 2	5	6	0	1	2	3	4	5	6	0	1	2
Вариант задачи 3	3	4	0	1	2	3	4	0	1	2	3	4
Вариант задачи 4	0	1	2	3	4	5	0	1	2	3	4	5

Теоретическая часть

Современный уровень проектирования цифровых вычислительных устройств характеризуется использованием программируемых логических интегральных схем (ПЛИС). Сложные микропроцессорные системы, содержащие на одном кристалле процессорное ядро, устройства памяти и обработки данных, а также интерфейсные модули называют «система на кристалле» (*System On Chip*). Процессоры, разработанные в виде проектов в САПР, называются синтезированными. Для решения конкретных задач синтезированный процессор оказываются гораздо компактнее универсального процессора, что позволяет получить выигрыш в аппаратных затратах и в быстродействии. Кроме того, синтезируемые процессоры абсолютно синхронны со всеми остальными устройствами, размещенными микросхеме, что упрощает обмен данными в системе. Проектирование процессора на основе ПЛИС является одной из наиболее интересных задач, возникающей при разработке комплексных устройств обработки данных. Рассмотрим осиновые этапы проектирования синтезированного процессора.

Выбор архитектуры микропроцессорной системы.

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

Рис. 1. Система на кристалле

В архитектуре Джона фон Неймана (1949 год) связь между процессором, устройствами памяти, ввода и вывода выполняет общая системная шина, которая содержит группы проводников - шины адреса, данных и управления (рис. 1). Для устройств памяти команд и данных используется общее адресное пространство , емкость которого определяет разрядность шины адреса.

Достоинство архитектуры Джона фон Неймана – экономия аппаратных средстнизкое быстродействие при обменен данными через об быстродействия вследствие обмена данными через общую шину.

Применение данной архтиектуры целесообразно при проектировании несложных устройств обработки данных

Гарвардская архитектура микропроцессорной системы (рис. 2) содержит независимые адресные пространства и отдельные устройства памяти для хранения команд и данных.

С устройствами памяти команд и данных процессор связан посредством отдельных шин. Это шина адреса команд ***ak***, шина команды ***k***, шина адреса данных ***ad*** и шина данных ***d_bus***. Гарвардская архитектура сложнее архитектуры Джона фон Неймана, но она обладает очевидными достоинствами.

1. 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.

2. Гарвардская архитектура позволяет простыми техническими средствами организовать двухфазную синхронизацию и конвейер операций.

Выбор конфигурации процессора

Конфигурация процессора отображает взаимодействие арифметико-логическое устройства (АЛУ) и элементов собственной быстродействующей памяти процессора. Для ее выбора конфигурации выполняется анализ заданных параметров разрабатываемой системы (быстродействие, объем данных, сложность вычислений) и известных вариантов построения процессора.

Для простых систем выбирается процессор аккумуляторного типа, подобный используемому в калькуляторах.

Для быстродействующих систем выбирается стековый процессор с безадресной системой команд.

Для задач средней сложности используется конфигурация процессора с блоком регистров общего назначения (РОН), которая обеспечивает выполнение двухадресных команд за один такт частоты синхронизации и получение при этом высокого быстродействия.

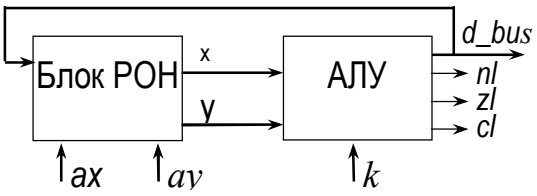

Рис. 3. Конфигурация процессора с блоком РОН

Блок РОН выполнен как двухадресная (двухпортовая) память, которая позволяет одновременно выдать в АЛУ содержимое двух регистров - rx и ry, в которых хранятся первый (x) и второй (y) операнды двухадресной команды. Регистр первого операнда rx впоследствии используется как получатель результата, при этом исходное значение первого операнда теряется. Содержимое регистра второго операнда y сохраняется. Код операции k и адреса регистров *ax* и *ay* поступают из команды. Результат операции, полученный в АЛУ, передается через шину данных d_bus на блок РОН для записи в регистр *rx,* а также на для записи в другие устройства памяти и регистры процессора.

Для проектируемого процессора использована гарвардская архитектура микропроцессорной системы и процессор с блоком РОН.

В блоке АЛУ содержатся комбинационные схемы lk\ формирования логических сигналов (nl zl, cl), определяющих признаки результата выполненной операции.

2. Разработка системы команд

В зависимости от типа системы команд различают: процессоры со сложной системой команд *(Complex Instruction Set Computer)* и процессоры с сокращенной системой команд *(Reduce Instruction Set Computer).*

В *CISC* – процессорах используются команды различной разрядности (например, 1, 2 и более байт), для выполнения команд требуются различные интервалы времени. Проектирование *CISC*-процессоров основано на их представлении в виде совокупности двух

автоматов – операционного и управляющего и на использовании микропрограммного управления. В Данный тип имеет система команд процессоров Intel 8080, 8086, и т.д.

В *RISC* - процессорах все команды имеют одинаковую разрядность. Каждая команда выполняется за один машинный цикл. Для проектируемого процессора будет использована гарвардская *RISC*-архитектура, подобная микроконтроллеру AVR.

Команда *RISC*- процессора должна содержать всю информацию, необходимую для ее выполнения. По этой причине при разрядности данных 1 байт команда должна содержать 16, или более разрядов.

Разработка системы команд содержит следующие этапы.

1) Выбор метода адресации.

2) Разработка формата команды, который отображает деление кода команды на поля, содержащие информацию определенного типа.

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).

3) Выбор функционального состава проектируемой группы команд

4) Кодирование команд.

2.1. Команды с непосредственной адресацией (k[15] = 0)

k1[15:12]	*rx[11:8]*	*dd[7:0]*

Рис. 4. Непосредственная адресация

При непосредственной адресации в коде команды содержатся данные. Формат команды с непосредственной адресацией содержит 3 поля:

k1 - код операции;

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

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

Поле k1должно 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.

После выполнения операции результат записывается в регистр rx, при этом исходное значение операнда теряется.

Один из способов обозначения мнемоник команд с непосредственной адресацией - добавление буквы i (от immediate - непосредственно). Примем для проектируемого процессора систему команд, содержащую

В таблице 1 приведены арифметические и логические команды с непосредственной адресацией, выбранные для проектируемого процессора. Таблица содержит коды команд (k1) и описания выполняемых функций на языке Vtrilog, показывающие возможность выполнения операций устройствами комбинационного типа, что особенно важно для RISC-процессора.. Использованы обозначения, приведенные на рис. 2 и 3..

Таблица 1. Команды с непосредственной адресацией

k1	Команда	Описание на Verilog	Пояснение
0	*movi rx, d8*	*d_bus = d8*	Пересылка
1	*addi rx, d8*	*d_bus =rx + d8*	Суммирование
2	*subi rx, d8*	*d_bus = rx - d8*	Вычитание
3	*andi rx, d8*	*d_bus = rx & d8*	Операция И
4	*ori rx, d8*	*d_bus = rx | d8*	Операция ИЛИ

5	*xori rx, d8*	*d_bus = rx ^ d8*	Исключ. ИЛИ

Пример. Ккоманда ***movi r3, 0x2f***, выполнит пересылку числа $2F_{16}$ в регистр ***r3,*** будет иметь код $032F_{16}$.

2.2. Двухадресные ккоманды с регистровой адресацией (k1=8)

k1[15:12]	*rx[11:8]*	*ry[7:4]*	*K2[3:0]*

3

Рис. 5. Регистровая адресациея

При регистровой адресации данные находятся в регистрах. Формат двухадресных коман с регистровой адресацией содержит 4 поля:

k1 – код метода адресации;
rx -номер регистра первого операнда;
ry -номер регистра второго операнда;
k2 - код исполняемой операции.

Результат операции записывается в регистр ***rx***, исходное содержимое которого теряется, содержимое регистра *ry* сохраняется. В таблице 2 приведены команды арифметических и логических операций с регистровой адресацией, выбранные для проектируемого процессора. Таблица содержит коды команд (k1, k2) и описания выполняемых функций на языке Vtrilog, в которых выходной сигнал АЛУ записан как конкатенация сигнала переноса cl и выходного сигнала, выдаваемого на шину d_bus. Для логических операций cl = 0.

Таблица 2. Двухадресныеоманды с регистровой адресацией

k 2	Команда	Описание на Verilog	Пояснение
0	*mov rx, ry*	*{cl,d_bus} = {1'b0,y}*	Пересылка
1	*add rx, ry*	*{cl,d_bus} x+ y*	Суммирование
2	*sub rx, ry*	*{cl,d_bus} =x- y*	Вычитание
3	*and rx, ry*	*{cl,d_bus} = {1'b0,x&y}*	Операция И
4	*or rx, ry*	*{cl,d_bus} = {1'b0,x|y}*	Операция ИЛИ
5	*xor rx, ry*	*{cl,d_bus} = {1'b0,x^y}*	Исключающее ИЛИ

Пример. Команда *add r1, r2* будет иметь 16-ричный код: 8121, к содержимому регистра r1дудет прибавлено содержимое регистра r2, результат запишется в r1. При переполнении будет выдан сигнал cl = 1

2.3. Одноадресные команды с регистровой адресацией (k1=9)

k1[15:12]	*rx[11:8]*	*k2[7:0]*

Рис. 6. Одноадресные команды с регистровой адресацией

Формат одноадресных команд с регистровой адресацией (рис.8). содержит три поля:

k1 – код метода адресации, выбираем его равным 9;

rx – номер регисста, который содержит исходное значение операнда, а после выполнения операции – результат;

k2, код исполняемой операции

Данный формат позволяет закодировать большое количество команд различного функционального назначения. Это сдвиги, инкремент и декремент, вычисление обратного, дополнительного, смещенного кода, обращения к стеку и другие.

Таблица 3 содержит команды, выбранные для проекта.

Таблица 3. Одноадресныеоманды с регистровой адресацией

k 2	Команда	Операция	Пояснение
0	*inc rx*	*d_bus= x+1*	инкремент
1	*dec rx*	*d_bu= x-1*	декремент
2	*not rx*	*d_bus=~x*	Обратный код

3	*asr rx*	*d_bus={x[7] x[7] x[6:1]}*	Арифметический сдвиг вправо
4	*lsl rx*	*d_bus:= { x[6:1], 1'b0}*	Логический сдвиг влево

2.4. Команды обращения к памяти с косвенной адресацией k1=0xa

k1[15:12]	*rx[11:8]*	*ra[7:4]*	*K2[3:0]*

Рис. 7. Косвенная адресация

При косвенной адресации в команде содержится указатель адреса ячейки памяти, в которой находится операнд. Формат этих команд содержит 4 поля (рис. 9):

k1 – метод адресации, примем для команд с косвенной адресацией *k1* = a_{16};

rx - номер регистра, в котором находится операнд, участвующий в обмене,.

ra - указатель адреса ячейки памяти, в которой находится операнд;

k2 – код, определяющий направление пересылки, для загрузки операнда в регистр из памяти примем ***k2*** = 0, а для записи в память из регистра ***k2*** = 1 (табл. 4).

При записи команд используются мнемоники:

ld (от слова *load*) - загрузка регистра, или пересылка в регистр содержимого ячейки памяти;

st (от слова store) – запись в ячейку памяти содержимого регистра.

[ra] - имя регистра в квадратных скобках обозначает указатель адреса ячейки памяти.

Таблица 4.Команды обращения к памяти с косвенной адресацией

k2	Мнемоника	Пояснение
0	ld rx, [ra]	Загрузка в rx операнда из памяти по адресу из ra
1	st rx, [ra]	Запись в память по адресу из ra, операнда из rx

Пример. Команда ***ld r1, [r2]***, например, имеет 16-ричный код *a*120, загружает в регистр ***r1*** содержимое ячейки памяти, адрес которой заблаговременно был записан в регистр ***r2***.

Команда ***st r3, [r4]*** имеет 16-ричный код a341, запишет в ячейку памяти, адрес которой записан в регистр ***r4,*** содержимое регистра ***r3***

2.5. Команды ветвления с прямой адресацией (k1=b16)

k1[15:12]	*k2[11:8]*	*a8[7:0]*

Рис. 8. Формат команд ветвления с прямой адресацией

При прямой (или абсолютной) адресации команда содержит адрес. Формат команд передачи управления с прямой адресацией (рис.10). содержит три поля:

k1 – код метода адресации, выбираем его равным b_{16};

k2, код исполняемой операции

a8 – адрес перехода

Таблица 5. Команды ветвления с прямой адресацией

k 2	Команда	Условие ветвления	Операция
0	b a8	*d_bus= x+1*	если z=1, то PC=a8
1	beq a8	если результат равен нулю (z=1)	если z=0, то PC=a8
2	bne a8	если результат не равен нулю (z=0)	если c=1,то PC=a8
3	bcs a8	если был перенос (c=1)	если c=0, то PC=a8
4	bcc a8	если не было переноса (c=0)	Если n=0, PC=a8
5	bpl a8	если знак результата плюс (N=0)	Если n=1, PC=a8
6	bьш a8	если знак результата плюс (N=0)	Если n=1, PC=a8

Пример. Команда ***jz 20***, например, имеющая код *f*120_{16} или 1111 0001 0010 0000_2 ,

выполняет условный переход к команде, адрес которой в ПЗУ команд равен 20_{16}, если после выполнения предыдущей команды получен флаг ***fz*** = 1.

3. Разработка системы синхронизации

Система синхронизации обеспечивает устранение ошибок, обусловленных гонками, возникающими при выполнении операций.

Работа процессора сводится к циклическому исполнению определенных операций.

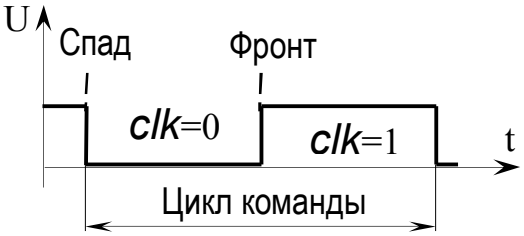

Рис. 9. Импульсы синхронизации

В соответствии с рис. 4 это вычисление адреса команжы, запись его в ПС, вычисление результата операции, запись результата. Целесообразно выбрать синхронное выполнение операций записи, в моменты спада и фронта импульса, которые определены достаточно строго. Для операций чтения и выполнения данных достаточно выделить интервал, не накладывая жестких требований на момент выполнения.

Период синхросигнала содержит 4 события. Это два состояния (*clk* = 0 и *clk* = 1) и два изменения состояний (спад и фронт импульса). Примем за начало машинного цикла команды, который должен составлять один период синхросигнала, спад синхроимпульса. Получим выполнение команды в виде последовательности следующих действий.

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

2. Состояние *clk = 0.* В АЛУ - обработка данных в соответствии с кодом команды, поступающим из определенных полей команды. На блок РОН поступают адреса регистров, используемых в данной команде, а на АЛУ – код операции.

Обработка данных выполняется устройствами комбинационного типа (это АЛУ, мультиплексоры, дешифраторы), поэтому формирование результата на шине d_bus выполнчется асинхронно и сопровождается задержками сигналов в логических элементах.

Схема синхронизации записи в это же время сформирует сигналы разрешения записи результата операции в те устройства памяти и регистры, которые указаны в данной команде.

Для команды ветвления на данном интервале определяется адрес перехлда.

3. Фронт синхросигнала *clk.* Выполняется запись результата из шины *d_bus* в регистр rx, или память, на которые подается сигнал разрешения записи. Для команды ветвления выполняется запись адреса следующей команды в программный счетчик.

адреса следующей команды, запись признаков результата в регистр состояния.

Сигналы разрешения записи данных, полученных на шине «*d_bus*» при выполнении текущей команды формирует схема разрешения записи.

4. Интервал С=1. Адрес следующей команды подается на ПЗУ, он подготовлен для записи по спаду синхроимпульса. Заметим, сигнал с выхода АЛУ – с шины d_bus - не используется, это «мусор».

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.

4. Разработка функциональной схемы микропроцессорной системы

В основе разрабатываемой схемы гарвардская архитектура (оис.2), содержащая элементы, выполняющие обработку команд, данных, хранение данных(Рис. 3).

Устройства обработки команд – это ПЗУ команд, программный счетчик и схема управления программным счетчиком, регистр 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]==1'hb)&((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
```

Программа, выполняемая процессором, хранится в **ПЗУ команд**. Выполняемая команда выдается с выхода ПЗУ на шину команд «k» и содержит всю информацию, необходимую для ее выполнения за один такт синхросигнала.

Программный счетчик (*Program Counter - PC*) содержит адрес выполняемой команды, который по шине *ak* поступает на адресные входы ПЗУ.

Схема управления формирует адрес следующей команды, которую процессор должен выполнять, и записывает его в программный счетчик. Если программа не содержит команд условных или безусловных переходов, то адрес следующей команды получается из адреса текущей команды прибавлением единицы. Если выполняется команда перехода (ветвления), то схема управления учитывает признаки результата (z, c, …), хранящиеся в регистре состояния (Status *Register*), и записывает в программный счетчик адрес следующей команды, указанный в команде перехода.

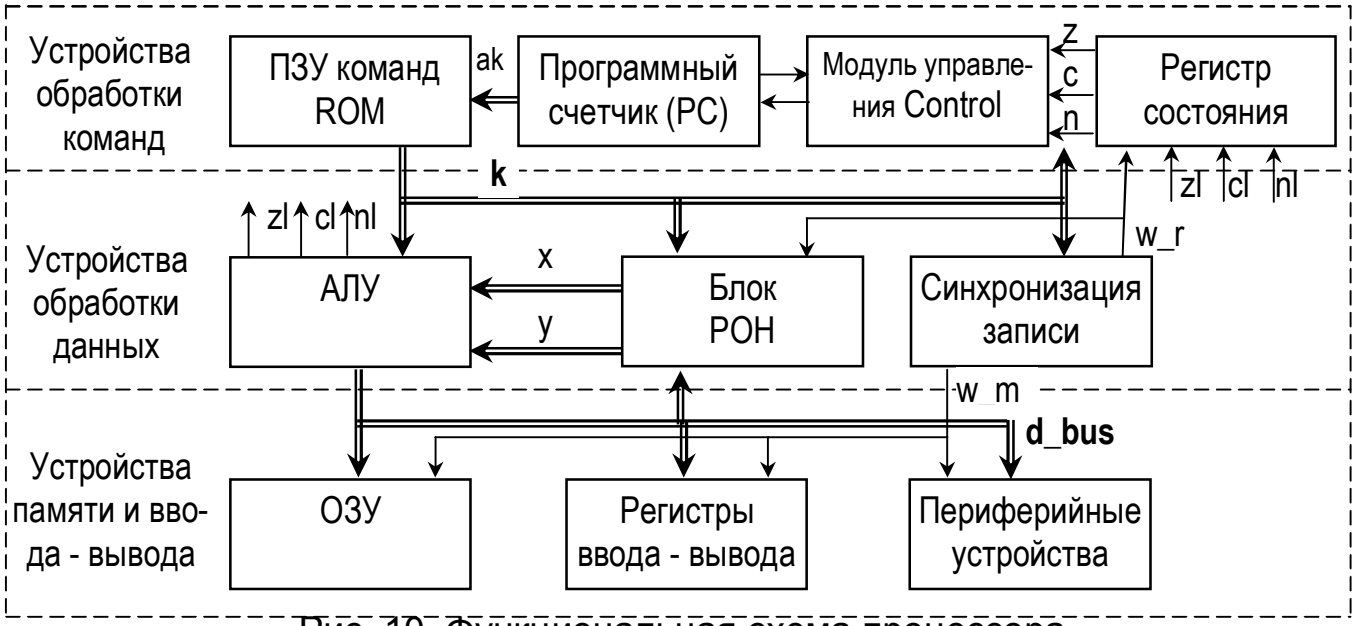

Рис. 10. Функциональная схема процессора

Устройства обработки данных – это АЛУ и блок РОН.

Заданная в команде операция выполняется в АЛУ, операнды поступают из РОН. Из кода команды (k) в АЛУ подается код операции, а в блок РОН адреса регистров *rd* и *rs,* в которых содержатся операнды. Результат операции с выхода АЛУ подается на шину данных *d_bus*, к которой параллельно подключены входы всех устройств, которые сохраняют результат операции. Это регистр rd блока РОН, ОЗУ, другие периферийные устройства. Команды с регистровой адресацией записывают результат в регистр *rd,* а команды обращения к памяти записывают результат в ОЗУ. Запись результата выполняется под управлением устройства синхронизации записи только в определенное устройство, указанное в выполняемой команде .

5. Разработка модулей микропроцессорной системы в САПР

```
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])};
endcase else
        if (k[15:12] == 8)  case (k [2:0])
0:d_bus=y;
1:{cl,d_bus} = x + y;
2:{cl,d_bus} = 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)};
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};
endcase else
if (k[15:12]==4'b1010)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
```

Проект микропроцессорной системы в САПР разрабатывается как иерархическая структура, содержащая модули в соответствии с функциональной схемой (рис.10). Для каждого модуля создайте проект, имч которого совпадает с именем в заголовке описании., выполните компиляцию, создайте символ.

5.1. Устройство управления программным счетчиком - control

Модуль **control** формирует адрес следующей команды.

Вописании приняты обозначения: clk – синхросигнал;
k - команда (16-разрядный вектор);
c –признак переноса
 z - признак нуля;
 n - признак отрицательного числа
branch – признак команды ветвления
q_pc – выходной код программного счетчика, является адресом следующей команды

. Признаки результата получены по результату выполнения предыдущей команды и записаны в регистр состояния sreg.

Выходным сигналом является адрес следующей команды, содержащийся в программном счетчике *pc*. Для сигнала pc указан тип *reg*.

Модуль control работает в соответствии с выбранными командами ветвления. Если текущая команда является командой ветвления и выполняется условие ветвления, то формируется сигнал ветвления branch = 1,в соответствии с которым в программный счетчик загружается адрес перехода из команды (см. рис.8). При отсутствии переходов адрес программного счетчика инкрементируется.

```
module blok_ron (c,wreg,
d_bus, ax,ay,x,y);
input c,wreg ;
input [7:0] d_bus;
input [3:0] ax, ay;
output [7:0] x,y;
 reg [7:0] x,y;
reg [7:0] ron [3:0];        //1
 always   begin              //2
 x=ron[ax]; y=ron[ay];   end
 always @(posedge c)     //3
if (wreg) ron[ax] = d_bus;
 endmodule
```

В описании записано выражение для сигнала branch с учетои формата и таблицы команд ветвления (рис. 8). Формат содержит признак команд ветвления k1=k[]15:12], и код типа команды k2 = k[]11:8]. Сигнал branch=1, если k1 = 11 (b_{16}), и при этом k2 = 0 (безусловное ветвление), или k2=1 и z=1 (ветвление по z), или k2=2 и z=0, и т.д..

Новое значение адреса записывается в программный счетчик по фронту синхросигнала и поступает на вход адреса ПЗУ. Впоследствиии, по спаду команда, хранящаяся в ПЗУ, будет присутствовать на шине команд k и выполняться в следующем цикле.

5.2. Блок РОН. В разрабатываемом процессоре использует двухадресный (двухпортовый) блок регистров общего назначения (РОН), имеющий режим одновремен-

ного чтения содержимого двух регистров. Для записи используют адрес первого регистра.

В описании блока РОН на языке Verilog используется описание блока памяти. Строка описания (1) содержит ключевое слово (reg), индексы, определяющие разрядность ячеек, имя блока памяти (ron) и индексы, определяющие разрядразрядность адреса ячейки в блоке. Обращение к ячейке содержит имя блока и адрес в квадратных скобках. Приняты обозначения: wreg - разрешение записи в регистр; d_bus – входной сигнал с шины данных; ax,ay – адреса первого и второго операндов; x,y – первый и второй операнды.

Одновременное чтение из диух ячеек выполняет последовательный оператор (строка 2) без списка чувствительности, поэтому чтение будет выполняться асинхронно, сразу же, как только изменятся данные, подобно параллельному оператору.

Запись ранных выполниет последовательный оператор, содержащий список чувствительности (строка 3), поэтому запись - синхронная по фронту сигнала, выполняется по перврму адресу (ax), если запись предварительно была разрешена (wreg = 1).

5.3. Модуль АЛУ. Выполняет арифметические и логические команды обработки данных с непосредственной и регистровой адресацией. Кроме того через данный модуль выполняются команды обращения к ОЗУ. В описании приняты обозначения:

k – код команды;

,x, y – первый и второй опреанды из блока РОН

dm – данные с выхода оперативной памяти (RAM);

d_bus – шина данных, входные данные для ФЛУ, записываются вместо x;

cl; ,zl, nl – признаки результата текущей команды полученные в форме логических сигналов, будут записаны в регистр состояния по фронту синхросигнала.

Модуль АЛУ выполнен в виде комбинационной схемы, которая формирует результат в соответствии с текущими значениями входных сигналов. Для описания подобных схем предназначены операторы параллельного типа (assign).

Второй вариант - использование последовательного оператора (always), в которых список чувствительности отсутствует. Такой оператор имеет более широкие функциональные возможности, позволяет использовать операторы case, if и скобки begin – end. Он будет срабатывать после изменения любого из входных сигналов подобно параллельному оператору. Выходной сигнал, формируемый данным оператором, необходимо описать как reg.

Данный вариант использован для описания АЛУ, где вначале, посредством оператора if, определяется тип адресации выполняемой команды, а затем оператором варианта case выполняется заданная команда.

Первый оператор if выполняет условный переход к выполнению команд с непосредственной адресацией; второй – к выполнению двухадресных команд с регистровой адресацией; третий– к выполнению одноадресных команд с регистровой адресацией; четвертый описывает выполнение команд обращения к памяти. По текущему результату формируются логические сигналя признаков результата zl, cl, nl.

```
module sync_wr (k, wreg, wmem);
input [15:0] k ;
output wreg,  wmem ;
assign wreg = ~k[15] |( k[15] &
 ~k[14] & ~k[13] );
 assign wmem = k[15] & ~k[14] &
 k[13] &~ k[12]& k[0];
 endmodule
```

5.4. Устройство синхронизации записи данных(sync_wr)

В описании приняты обозначения: k – код команды; wreg - разрешение записи в РОН и в регистр состочния; wmem – разрешение записи в оперативную памятью

Результат операции, выполненной а АЛУ, выдается на шину выходных данных d_bus, к которой под-

ключаются 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 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
```

5.5. Регистр состояния status_reg. В описании приняты обозначения:

Clk – входной синхросигнал, wreg входной сигнал разрешения записи в регистр zl, cl nl – входные сигналы признаков с выхода комбинационных логических схем z, c, n – признаки, хранящиеся в status_reg.

В регистре состояния если разрешена запись в регистры, то по фронту синхросигнала новые значения признаков (zl,cl,nl) записываются вместо старых значений (z,c,n).

5.4. Разработка тестовой программы. В схеме микропроцессорной используется ПЗУ. При вводе символа lpm_rom требуется указать файл инициализации памяти. Его необходимо создать заблаговременно, Рассмотрим разработку файла инициализации памяти test_0.mif для тестирования команд с непосредственной и регистровой адресацией.

Для записи программы используется ассемблер с произвольно выбранным алфавитом (столбец 1 таблицы), команды которого позволяют записать машинные коды. Таблица программы test_0 содержит столбцы: программа – исходный тек на ассемблере; ak – адрес команды в ПЗУ, или номер строки таблицы; k – код команды; d_bus – код на выходе АЛУ при выполнении команды; nzc – признаки результата на, записанные в регистр sreg.

Таблица 6. Программа test_0

программа	ak	k	*q_bus*	nzc
movi r0, 00	*00*	*0001*	*01*	*010*
addi r0, 85	*01*	*1085*	*85*	*100*
addi r0, 97	*02*	*1097*	*1c*	*001*
mvi rf, 07	*03*	*0f07*	*07*	*000*
subi rf, 85	*04*	*2f05*	*0A*	*101*
movi r1, 56	*05*	*0165*		
movi r2, 0f	*06*	*020f*		
and r1, r2	*07*	*8123*		
movi r1, 55	*08*	*0165*		
or r1, r2	*09*	*8124*		
movi r1,55	*0a*	*0165*		
xor r1, r2	*0 b*	*8125*		

Первая строка – пересылка нуля в r0. Должен появиться z = 1.Вторая строка прибавление к содержимому r0 числа, старший бит которого равен 1, появится n = 1. Строка 3. Прибавление к r0 числа 97 (в 16-ричной системе), произойдет переполнение, c = 1. В строках 4,5 проверяется работа регистра rf –это максимальный номер. При вычитании получится отрицательный результат. Признаки в остальных строках определите самостоятельно.

Для заполнения столбца k используются выбранные ранее коды из системы команд.

Создайте файл инициализации памяти: Eile/Nev/ Memory Initialization File. Введите количество отсчетов 16, разрядность 16, система счисления 16-ричная. Фийл должен содержать кщды выделенных столбцов таблицы 6. Содержимое остальных столбцов уточняется при моделировании. Сохраните файл с именем test.mif.

6. Экспериментальная часть

Выполнение задания 1

Принципиальная схема проекта микропроцессорной системы (рис. 11) разрабатывается в следующей последовательности.

6.1. Назначение типа микросхемы и ввол символов модулкй.

Создайте графический файл для схемы верхнего уровня иерархического проекта микропроцессорной системы. Создайте проект.

В разрабатываемом проекте необходимо назначить ПЛИС достаточно большой информационной емкости, имеющую блочную память. Необходимо выбрать из меню Assign/Device, в открывшемся окне указать семейство, например, ACEX 1K и тип устройства AUTO. Введите символы всех разработанных модулей

6.2. Ввод и настройка модуля ROM.

Введите символ модуля ПЗУ, указывая после двойного щелчка по схеме из библиотеки: megafuinctions / Storage / lpm_rom. В первом окне мастера установки (Mega Wizard Plug-In Manager) проверьте каталог установки, во втором окне установите разрядность 16 бит и количество слов 16. Это предполагаемое количество ломанд. В третьем окне укажите асинхронный (unregistered) режим для выходного сигнала ПЗУ, для этого в разделе Which ports should be registered? снимите галочку в строке 'q' output port. В четвертом окне нажмите кнопку Browse, укажите тип файлов (MIF Files) и путь к ранее созданному файлу test0.mif.

После ввода символа следует открывается окно редактирования портов и параметров Edit Port/Parameters, в разделе Ports которого предусмот

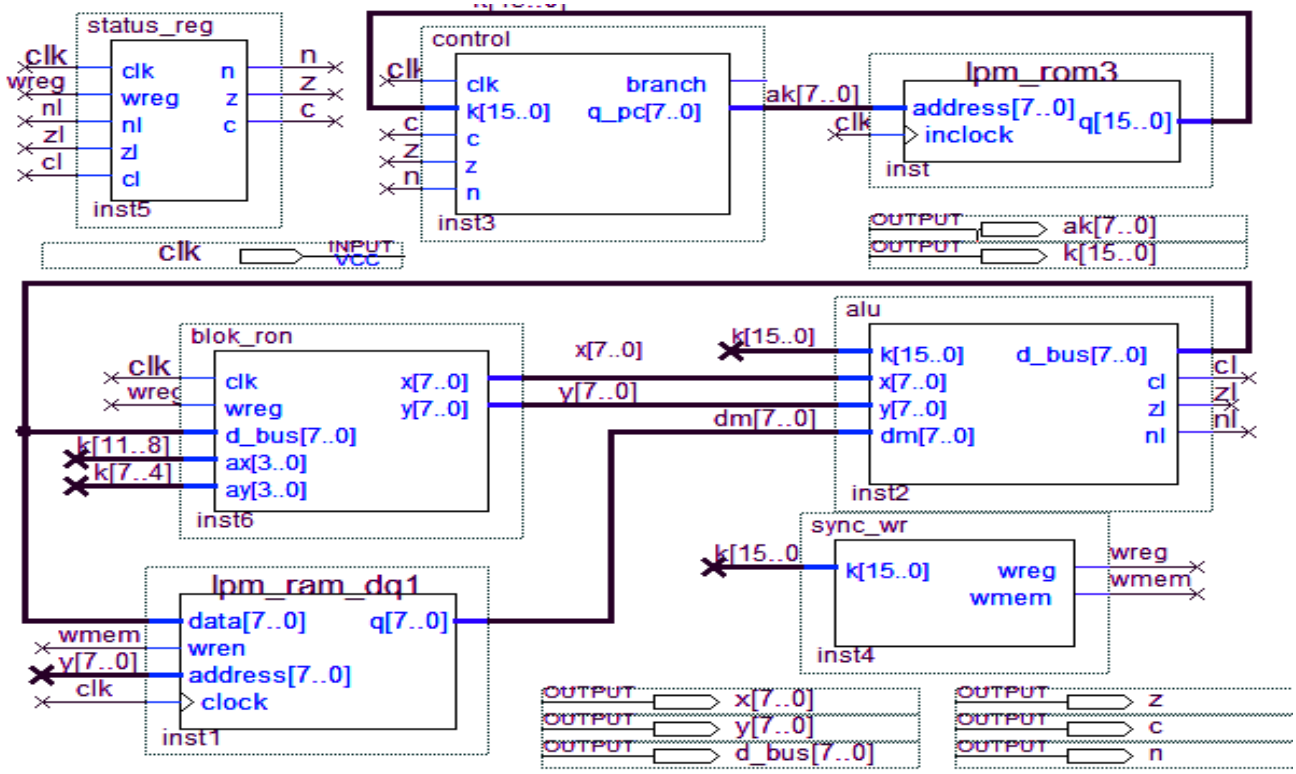

Рис.11 Схема микропроцессорной системы в графическом редактору

6.3. Ввод и настройка модуля RAM.

Результат выполнения всех команд формируется на выходе АЛУ и подается по шине данных d_bus на блок РОН и на модуль ОЗУ - LPM_RAM_DQ. Выбранное устройство имеет отдельные шины входных (q) и выходных (d) данных.Запись данных выполняется по фронту синхросигнала clk при наличии сигнала разрешения записи wmem.

Параметры модуля : рахрядность – 8, количество ячеек – 16, вход – синхронный, выход – асинхронный (подобно ПЗУ).

6.4. Моделирование и анализ результатов.

Для схемы (рис. 11)выполните компиляцию, создайте файл временных диаграмм. Из меню Edit / End Time установите время моделирования End Time = 5 us, чтобы отобразить процессы выполнения программ, содержащих несколько операторов.

Выполните анализ результата моделирования (Рис. 12).

1. Определите моменты срабатывания устройств памяти с динамическим управле-

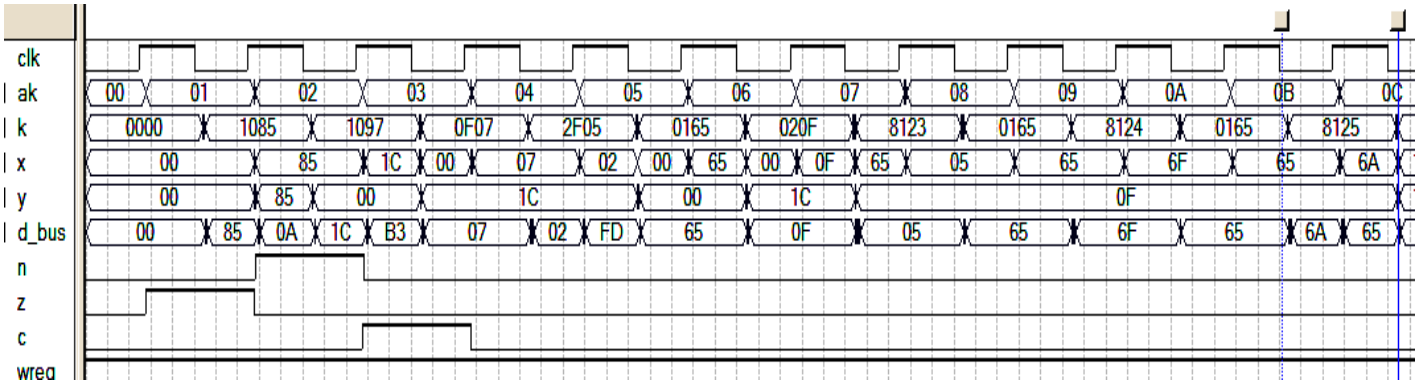

Рис. 12 Работа программы test_0

нием (РОН, программный счетчик, регистр признаков, ПЗУ, ОЗУ). Результат приведите в отчете.

2. Выделите циклы команд – интервалы, соответствующие наличию кода команды на шине k.

3. Опишите результат выполнения каждой команды, приведите признаки.

6.5. Пример выполнения задания 2

Пусть задано разработать программу для тестирования заданной группы команд - команд сдвигов и привести в отчете результаты моделирования.

В разработанной системе команд содержится две команды сдвига: *rrc rx* - циклический сдвиг вправо через перенос с кодом операции *9rx0* и *rlc rx* - циклический сдвиг влево через перенос с кодом операции 9 *rx* 1.

В программе тестирования данной группы команд необходимо загрузить в регистр константу, а затем выполнять сдвиги. Для полного тестирования необходимо проверить сдвиги для различных значений 8-разрядного операнда (2^8 вариантов), которые могут выполняться в одном из 8 регистров (2^3 вариантов), в одном из двух направлений. Всего 4096 вариантов команд сдвига.

Проверим правильность выполнения операции сдвига константы $0e_{16}$ влево и вправо в регистре ***r5***. При моделировании для анализа результата в данном случае целесообразно использовать двоичную систему счисления. При сдвиге вправо содержимое младшего разряда операнда будет записываться в триггер признака переноса ***cf***, а исходное значение ***cf*** будет поступать в старший разряд операнда, а при сдвиге влево старший бит операнда будет записываться в триггер признака переноса ***cf***, а исходное значение ***cf*** будет поступать в младший разряд операнда. Перед выполнением сдвига необходимо установить признак ***cf*** в определенное состояние, например сбросить. Программа приведена в табл. 6.

Таблица 6 Программа по аданию 2

Операция	Адрес	Мнемоника	Код
Загрузить в r5 константу 0e	00	mov r5,# 0e	050e
Циклический сдвиг вправо через перенос	02	rrc r5	9500
Циклический сдвиг вправо через перенос	03	rrc r5	9500
Циклический сдвиг вправо через перенос	04	rrc r5	9500
Циклический сдвиг вправо через перенос	05	rrc r5	9500

Циклический сдвиг влево через перенос	06	rlc r5	9501
Циклический сдвиг влево через перенос	07	rlc r5	9501
Циклический сдвиг влево через перенос	08	rlc r5	9501
Циклический сдвиг влево через перенос	09	rlc r5	9501
Циклический сдвиг влево через перенос	0a	rlc r5	9501
Циклический сдвиг влево через перенос	0b	rlc r5	9501

Временные диаграммы, отражающие результаты моделирования (рис. 12), позволяют выполнить анализ работы программы и определить временные задержки выполнения отдельных операций относительно спада синхроимпульса. Начало измеряемого интервала отмечают маркером, а конец – указателем мыши.

1. Программа не содержит переходов. В каждом цикле команды адрес команды (шина ***ak***) инкрементируется по спаду синхроимпульса с задержкой 11 нс.

2. На шине команд появляются коды команд в соответствии с программой с задержкой 31 нс.

3. По адресу 00 записана команда загрузки константы $0E_{16}$ в регистр ***r5***. Код $0E_{16}$ с задержкой 52 нс появился на выходе АЛУ и шине ***d_bus*** с задержкой 52 нс.

4. По адресу 01 записана команда, не изменяющая содержимого регистра, но устанавливающая флаги. В результате установился ***cf***=0.

5. По адресу 02 записана тестируемая команда сдвига. По спаду синхроимпульса (момент отмечен маркером) с задержкой 11 нс установился адрес команды, затем с задержкой 31 нс установилась команда, а с задержкой 50 нс появился результат выполнения операции на шине данных ***q_alu***. По фронту синхросигнала ***c*** задержкой 40 нс результат операции записан в регистр и появился на шине ***x***.

6. Анализ временных диаграмм показывает, что запись данных производится в моменты времени, когда все изменения сигналов завершились. Тактовая частота процессора может быть увеличена без потери работоспособности.

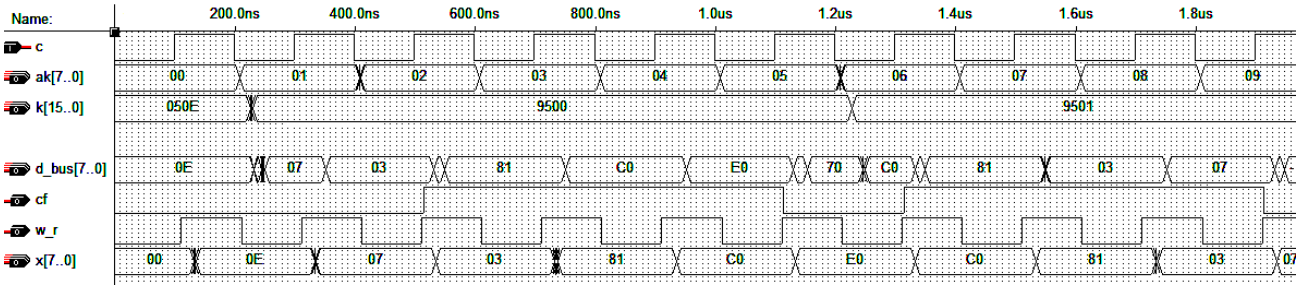

Рис. 12. Временные диаграммы работы программы , задание 2

6.6. Пример выполнения задания 3

В памяти, начиная с адреса 00, запишите массив из 8 чисел W_0 – W_7, которые вычисляются в соответствии с заданной формулой: $W_k = 3 \cdot k + 2.$

При решении задачи на ассемблере (или в машинных кодах) необходимо предусмотреть выполнение всех машинных команд на уровне регистров. Поэтому первый этап решения данной задачи - назначение регистров для хранения определенных переменных. Пусть регистр r0 – счетчик циклов, регистр r1 выполняет функцию аккумулятора, r2 – разность прогрессии, а r3 используемый как указатель адреса при записи в память, в котором хранится адрес элемента массива. Затем составляется алгоритм и программа (табл. 7).

Таблица 7 Программа по аданию 3

Адрес	Операция	Мнемоника	Код
00	**Инициализация.** Счетчика циклов *r0:=0*	*mov r0, #08*	0008

01	Сброс аккумулятора *r1:=0*	*mov r1, #00*	0100
02	Запись разности прогрессии в r2:=2	*mov r2, #02*	0202
03	Сброс указателя *r7:=0*	*mov r7, #00*	0700
04	**Цикл:** В аккумуляторе очередное число	*add r1, r2*	8121
05	Запись очередного числа в память	*st r1, [r7],*	a171
06	Инкремент указателя *r7:=r7+1*	*add r7, #01*	1701
07	Декремент счетчика циклов *r0:=r0-1*	*add r0, #ff*	10ff
08	Условный переход на завершение цикла	*jz 00a*	f10a
09	Безусловный переход на повторение цикла	*jmp 04*	f004
0a	Бесконечный цикл	*jmp 0a*	f00a

Программа содержит цикл. Счетчиком цикла является сигнал ***y*** (содержимое регистра ***ry***). После выполнения нулевого цикла на шине ***q_alu*** получен результат $W_0 = 2$. Результаты записываются в память по фронту сигнала ***w_m***. Работу программы отображают временные диаграммы, приведенные на рис. 13.

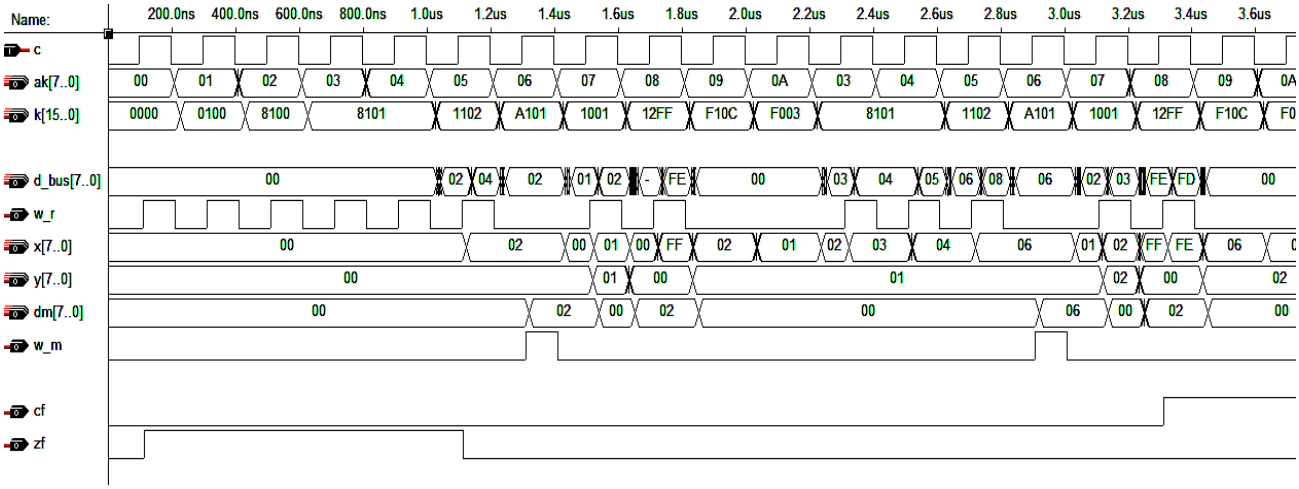

Рис. 13. Временные диаграммы работы программы, задание 3

Период синхроимпульсов, устанавливаемый по умолчанию, равен 200 нс. При этом в ОЗУ данных при параметре *End Time*, равном 10 мкс, формируется массив из 7 элементов. Измерения временных задержек формирования сигналов, выполненные а задании А, показывают, что задержка формирования данных составляет приблизительно 50 нс, следовательно период синхросигнала можно установить равным 100 нс в окне «Overwrite Clock».

На диаграммах маркером отмечен момент записи данных в регистр (по фронту сигнала ***w_r***). Диаграммы показывают, что установленная частота ($F = 1/T = 10$МГц) близка к предельной. Для нахождения предельной частоты следует уменьшать период синхросигнала до появления ошибок.

Массив чисел в ОЗУ данных, показанный на рис. 14, соответствует теоретическим данным.

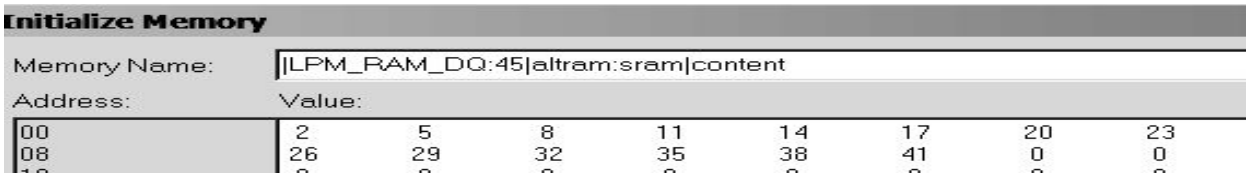

Рис. 14. Массив данных в ОЗУ, задание 3

6.7. Пример выполнения задания 4

Включите в систему команд дополнительную команду перестановки тетрад в байте заданного регистра swap rx, представьте результаты тестирования.

В задании В выполняется модификация модулей проекта, поэтому необходимо создать копию проекта с другим именем, для этого открыть файл *processor.gdf* и командой *Save As* сохранить его с именем *processor_m.gdf*. Затем открыть модуль, подлежащий модификации, например АЛУ, и также сохранить его с другим именем *alu_m*. Это будет модифицированный модуль АЛУ, в котором необходимо заменить имя в описании, выполнить модификацию, компиляцию, создать символ, вставить этот символ, имя которого будет alu_m, в модифицированный проект процессора processor_m.

Для включения в систему команд заданной команды *swap rx* необходимо в описании АЛУ изменить раздел одноадресных команд, для которого выбран код К1 = 9, следующим образом. После описания команд сдвига, которым соответствуют значения селектора *(k [2:0]), равные 0 и 1*)

if (k[15:12] == 9) case (k [2:0]) 0:{q,cl}={cf,x}; 1:{cl,q}={x,cf};

необходимо добавить строку с описанием заданной команды

2: {cl,q}={1'b0, x[3:0], x[7:4]};

Для заданной команды *swap rx* выберем машинный код 9 *rx* 0 2.

Для тестирования команды составлена программа, содержащая загрузку в регистр *r5* константы 0*F* (код команды 050*F*), а затем несколько команд *swap rx* (код команды 9502), которые меняют местами тетрады в байте. Ввиду простоты программы подробная таблица с пояснениями не приводится.

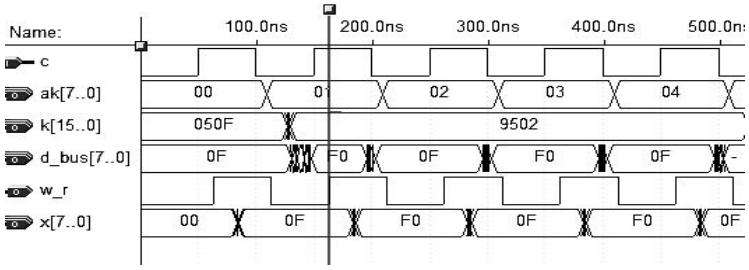

Рис 15. Работа команды *swap r5*

Временные диаграммы (рис. 15) отображают выполняемые команды и подтверждают правильность работы заданной команды. В каждом машинном цикле меняются местами татрады в байте.

Библиографический список

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.-124с.
3. Зотов В. Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы XILINX. - М.: Горячая линия - Телеком, 2006. - 520 с, ил.
4. В. В. Соловьев.. Основы языка проектирования цифровой аппаратуры VERILOG. — М.: Горячая линия-Телеком, 2014. — 205 с: ил.