Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Д.И. Самаль, В.В. Колонов

СТРУКТУРНАЯ И ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ ЭВМ. ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ЦЕНТРАЛЬНОГО ПРОЦЕССОРА

Лабораторный практикум

для студентов специальности

1-40 02 01

“Вычислительные машины, системы и сети“

всех форм обучения

Минск БГУИР 2010

УДК
ББК
С17

Рецензент:
зав. лабораторией логического проектирования
ОИПИ НАН Беларуси,
доктор техн. наук П.Н. Бибило

Самаль Д.И.

С17 Структурная и функциональная организация ЭВМ. Повышение производительности центрального процессора: Лабораторный практикум для студентов специальности 1- 40 02 01“Вычислительные машины, системы и сети“ всех форм обучения./ Д. И. Самаль, В. В. Колонов. – Минск: БГУИР, 2010. – 39 с.: ил.

ISBN

В учебном пособии рассмотрены типовые задания на лабораторные работы, проводимые в рамках дисциплины «Структурная и функциональная организация ЭВМ». Для каждой работы приводится теоретический материал, соответствующий теме работы.

Особое внимание уделено вопросам использования для моделирования схем системы автоматизированного проектирования Quartus 9.0.

УДК
ББК

ISBN

Содержание

Введение

Настоящий лабораторный практикум содержит варианты задания на 5 лабораторных работ, посвящённых изучению таких сложных функциональных блоков центрального процессора ЭВМ как: конвейер команд, кэш-память, предсказатель условных переходов.

Каждая из работ сопровождается теоретическим материалом, который позволяет студентам без использования дополнительных литературных источников разобраться в принципах функционирования изучаемого блока, спроектировать его в САПР Altera Quartus v.7.1 – v. 9.0 и промоделировать его поведение в различных режимах работы.

Помимо лабораторных работ в методическое пособие включены требования к курсовому проекту по дисциплине «Структурная и функциональная организация ЭВМ, а также варианты заданий к нему.

ЛАБОРАТОРНАЯ РАБОТА № 1
"Арбитраж шин"

1.1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В вычислительной технике существуют архитектуры вычислительных систем, содержащие нескольких ведущих устройств, взаимодействующих с ведомыми устройствами посредством одной шины. Некоторые устройства могут быть как ведущими, так и ведомыми в различные моменты времени. Очевидно, что в таком случае неизбежно будут возникать ситуации конфликта интересов ведущих устройств, т.е. попытки получения в своё распоряжение шины одновременно с другими ведущими устройствами. Простейшим решением задачи является присвоение приоритетов всем ведущим устройствам, использующим одну общую шину и решение конфликтов согласно приоритетам. Существует два способа распределения приоритетов по ведущим устройствам:

- статический (приоритеты устанавливаются разработчиком ЭВМ и больше не меняются),
- динамический (приоритеты определяются для каждого из устройств в конкретный момент времени по некоторому алгоритму и, таким образом, дают шанс на пользование шиной каждому ведущему устройству).

Минусом первого способа является высокая вероятность ситуации, в которой устройства с высоким приоритетом могут полностью блокировать (т.е. монополизировать) шину. Минусом второго – усложнение схемы ведущих устройств и процедуры арбитража.

Для динамического способа назначения приоритетов чаще всего используют следующие алгоритмы смены приоритетов устройств:

- Простая циклическая смена приоритетов – после каждого цикла производится изменение приоритета на единицу по кругу для всех ведущих устройств.
- Циклическая смена приоритетов с учётом последнего запроса – последний обслуженный получает самый низкий приоритет, остальные за ним сдвигаются по кругу. Данный алгоритм более распространён, чем предыдущий.
- Смена приоритетов по случайному закону – генератор случайных чисел назначает новые значения приоритетов.
- Схема равных приоритетов – при поступлении нескольких запросов каждый из них имеет шансы на обслуживание, а конфликт решается арбитром. Алгоритм обычно применяется в асинхронных схемах.
- Алгоритм наиболее давнего использования – LRU (Last Recently Used) – после каждого цикла наивысший приоритет получает то устройство, которое дольше всех не использовало шину.

К дополнительным алгоритмам, которые встречаются реже перечисленных выше, относятся:

- Очередь запросов на обслуживание – FIFO. Алгоритм требует сложной аппаратной реализации и по этой причине используется редко.
- Фиксированный квант времени – каждому ведущему отводится фиксированный промежуток времени для захвата шины. Алгоритм хорошо подходит для шин с синхронным протоколом.

Существуют две схемы арбитража - **централизованная** и **децентрализованная.**

В первом случае в ЭВМ существует **центральный арбитр (ЦА)** либо центральный контроллер шины (конструктивно он может быть выполнен в виде самостоятельной микросхемы либо быть интегрированным в ЦП), который полностью контролирует доступ к шине ведущих устройств. Централизованные схемы арбитража могут быть реализованы двумя способами – параллельным или последовательным.

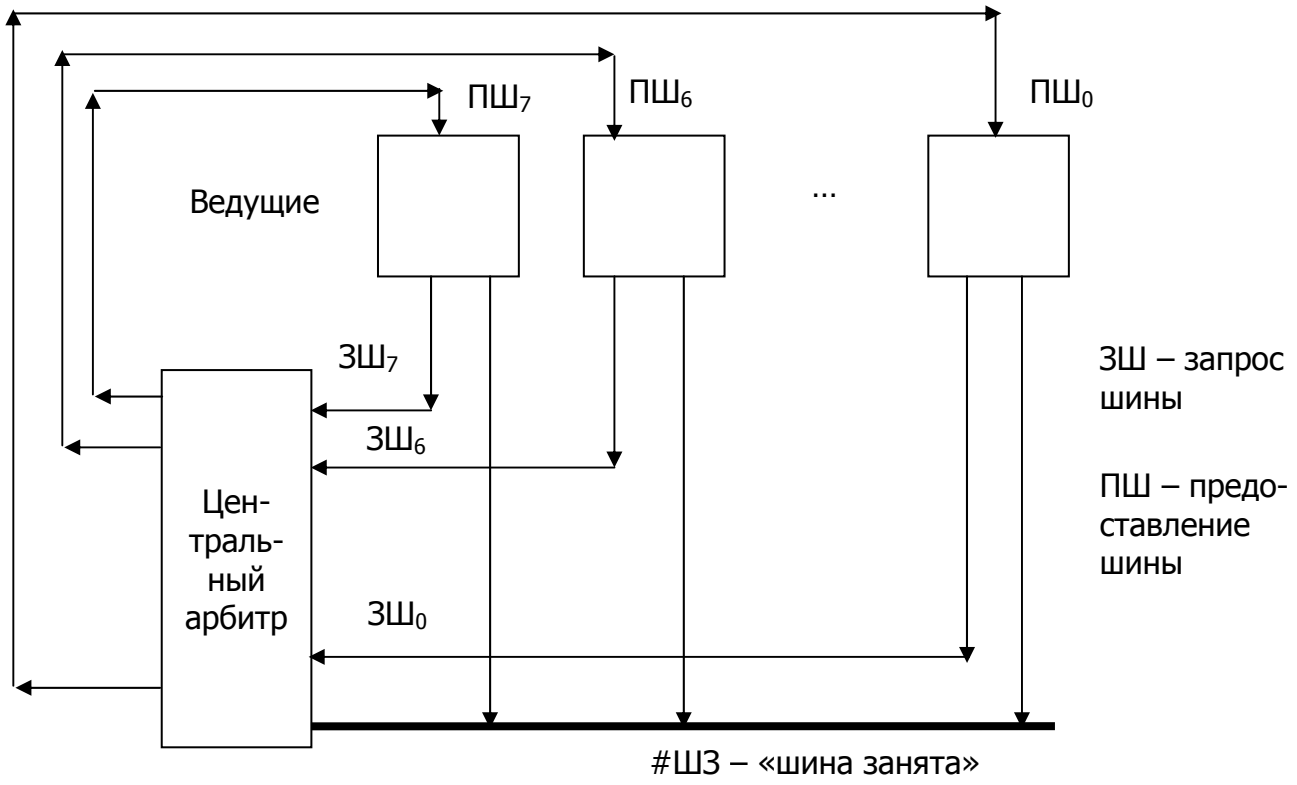


Рис. 1.1. Централизованный параллельный арбитраж (или «централизованный арбитраж независимых запросов)

При параллельном подключении – ЦА связан с каждым потенциальным ведущим индивидуальными двухпроводными трактами. Соответственно, запросы на доступ к шине могут поступать от ведущих параллельно и независимо. При таком подключении ЦА может реализовывать любой из перечислен-

ных выше алгоритмов смены приоритета. Схема параллельного подключения представлена на рис. 1.1.

Централизованный последовательный арбитраж так же называют цепочечным или гирляндным. Существует три вида подобного арбитража – с цепочкой сигнала ЗШ (запрос шины), доп. сигнала разрешения (РШ) и сигнала предоставления шины (ПШ) – наиболее распространённый вариант (рис. 1.2 - 1.3). Данный способ организации арбитража может быть реализован в виде схемы с одной цепочкой сигнала, так и с несколькими (рис. 1.3). В таких случаях говорят, что схема имеет несколько линий приоритета.

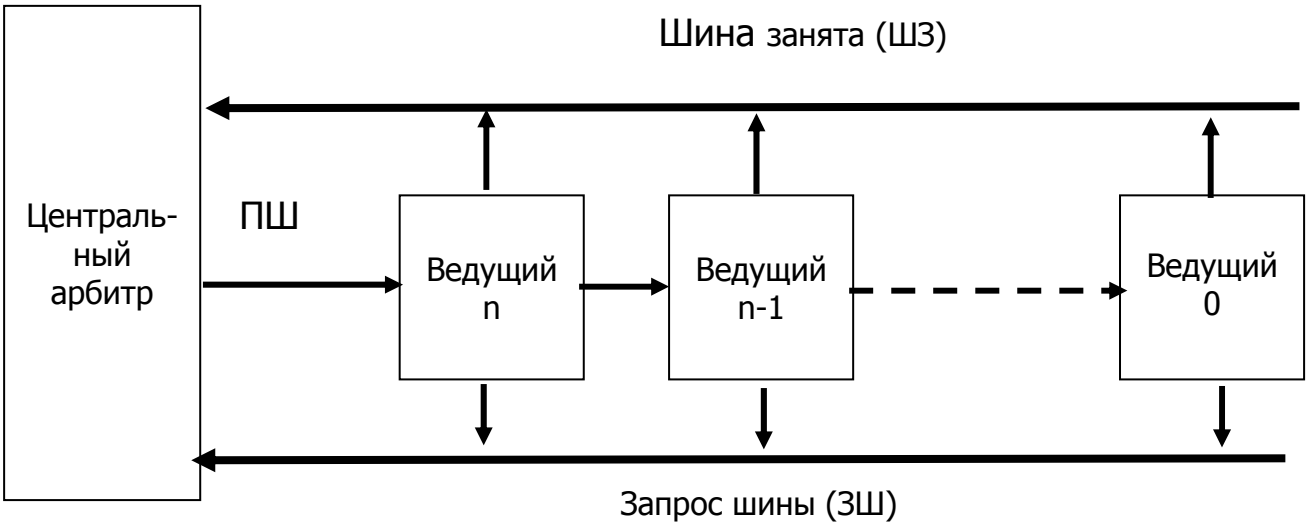


Рис. 1.2. Централизованный последовательный арбитраж

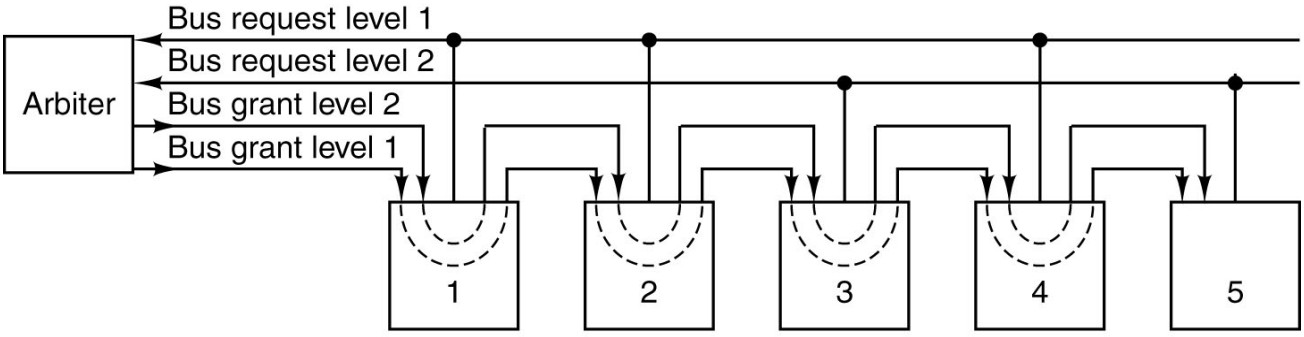


Рис. 1.3. Централизованный последовательный арбитраж с двумя линиями приоритета

Децентрализованный (или распределённый) **арбитраж** основан на принципе самостоятельности анализа каждым из ведущих устройств возможности захвата им шины для выполнения транзакции. Для этого, каждый из ведущих содержит собственный блок управления доступом к шине. Блоки взаимо-

действуют между собой и определяют в каждый момент времени устройство, которое имеет наивысший приоритет. Простейший вариант подобной схемы – параллельный децентрализованный арбитраж представлен на рис. 1.4.

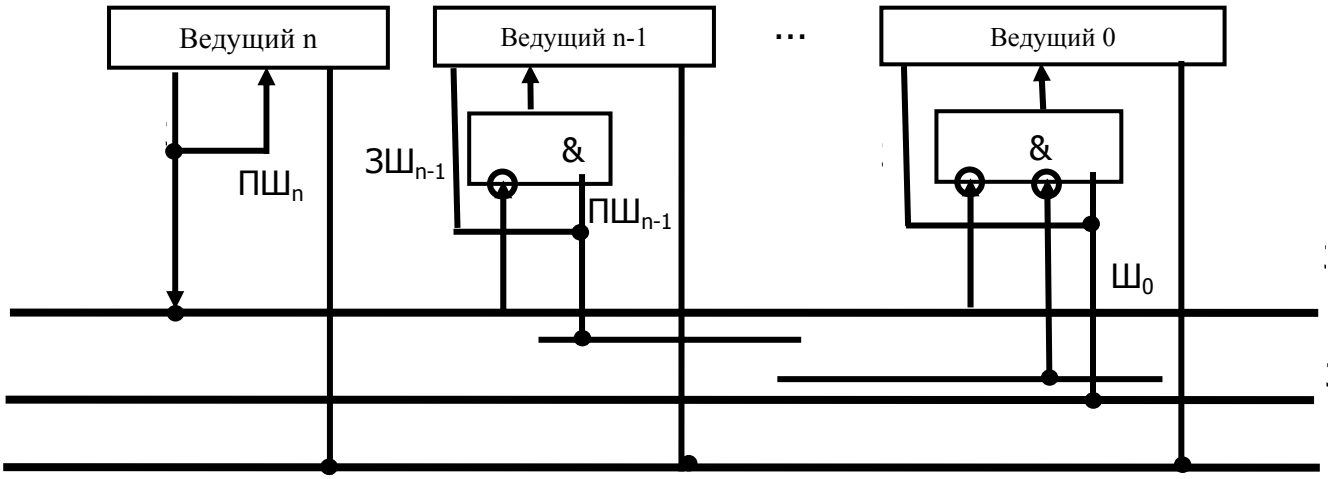


Рис. 1.4. Параллельный децентрализованный арбитраж

Кольцевая схема арбитража с циклической сменой приоритетов представлена на рис 1.5. Переход к следующему ведущему производится со сменой приоритетов. Текущий ведущий в следующем цикле шины будет иметь наименьший приоритет, его сосед справа – наивысший, остальные устройства – на 1 меньше, чем у соседа слева. Текущий ведущий генерирует сигнал ПШ, который проходит через все схемы логики арбитража (ЛА). Если имеется ещё один ведущий, который просит шину (т.е. генерирует запрос ЗШ), то его ЛА не пропускает сигнал ПШ к другим устройствам. Текущий ведущий, «потеряв» сигнал ПШ на своём входе, должен при первой возможности освободить шину. Запрашивающий её займет. Аналогично строится и циклическая смена арбитража с учётом последнего запроса.

Распределённый арбитраж с самостоятельным выбором строится по следующей схеме: арбитражные линии и устройства монтируются по схеме «монтажное ИЛИ». Каждому ведущему присваивается уникальный номер, который соответствует уровню приоритета. Запрашивающие шину выставляют на арбитражные линии свой номер. Каждый из запрашивающих шину ведущих, обнаружив на линиях номер с более высоким приоритетом, снимает младшие биты своего номера. В итоге сеанса арбитража (может понадобиться несколько итераций) на линии остаётся только номер с наиболее высоким приоритетом. Ведущий, распознавший на линиях свой номер – захватывает шину.

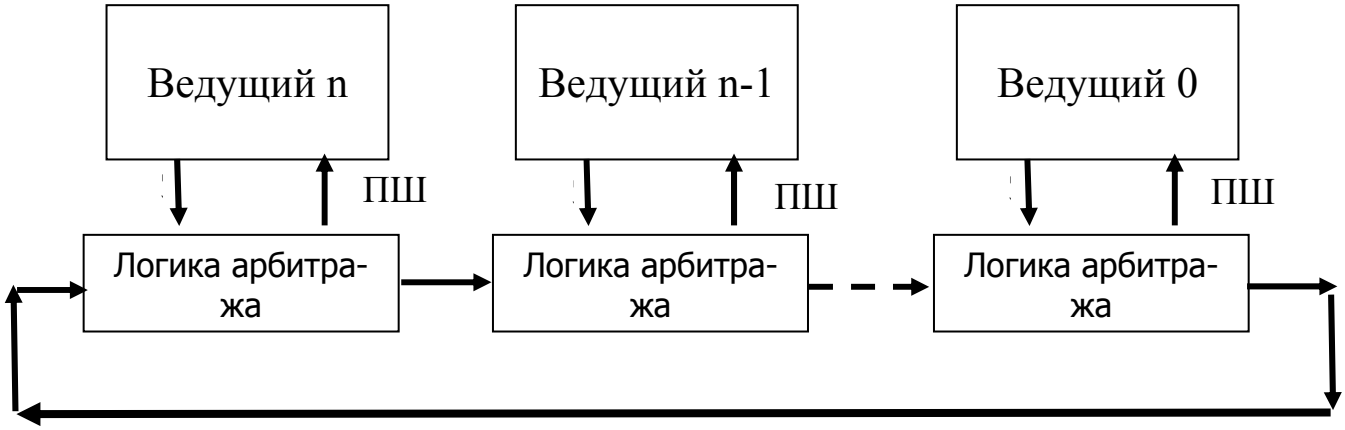


Рис. 1.5. Кольцевой децентрализованный арбитраж

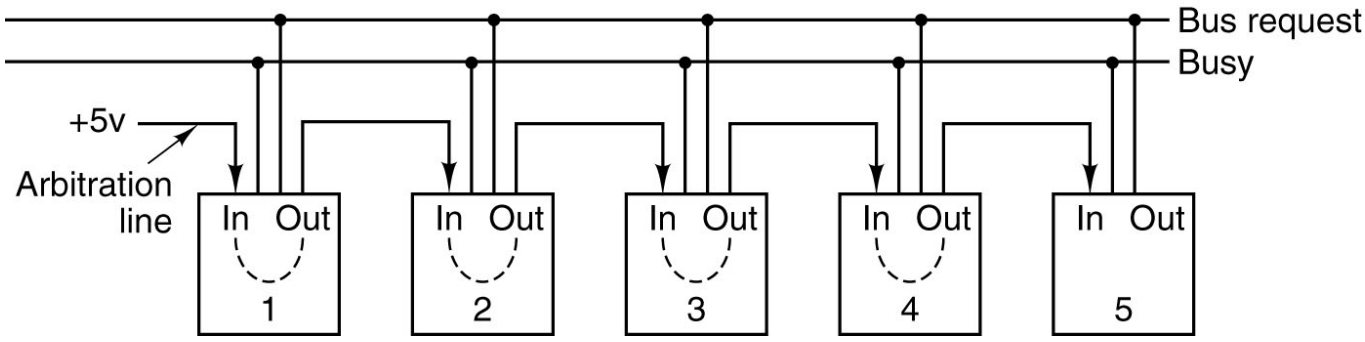


Рис. 1.6. Простейший децентрализованный арбитраж с фиксированными приоритетами

1.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1.2.1. Задание и рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в новом проекте, не используя предыдущих наработок.

Задача состоит в создании нескольких модулей ведущих устройств, количество которых задано в таблице вариантов 1, центрального арбитра (если он задан по варианту), логики арбитража и одного ведомого модуля.

В качестве разделяемого ресурса используется шина данных заданной ширины. Каждое из ведущих устройств пытается выставить для ведомого устройства на эту шину свои уникальные данные (например, порядковый или уникальный номер). В соответствии с вариантом задания необходимо обеспечить арбитраж этой шины данных и прием данных ведомым. Для тех случаев, когда это необходимо можно использовать дополнительно шину управления.

Снятие данных с шины ведомым устройством следует обеспечивать синхронно либо асинхронно в зависимости от значения в поле «Тип шины» таблицы вариантов.

Разрешается использовать функциональное моделирование (без учета задержек). Для этого в диалоге *Processing->Simulator Tool* необходимо переключить тип моделирования *(Simulation Mode)* на функциональное моделирование *(Functional)*. Перед запуском функционального моделирования необходимо сгенерировать таблицу соединений с помощью кнопки *«Generate functional simulation netlist»* (рис.1.7).

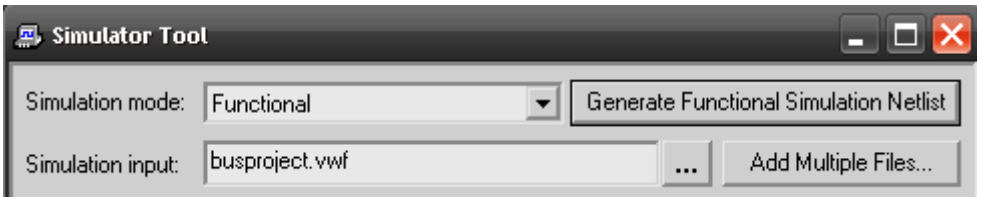


Рис. 1.7. Переключение в режим функционального моделирования

Рассмотрим пример реализации централизованного параллельного арбитража со статическими приоритетами 4-х разрядной шины с двумя ведущими устройствами и одним ведомым.

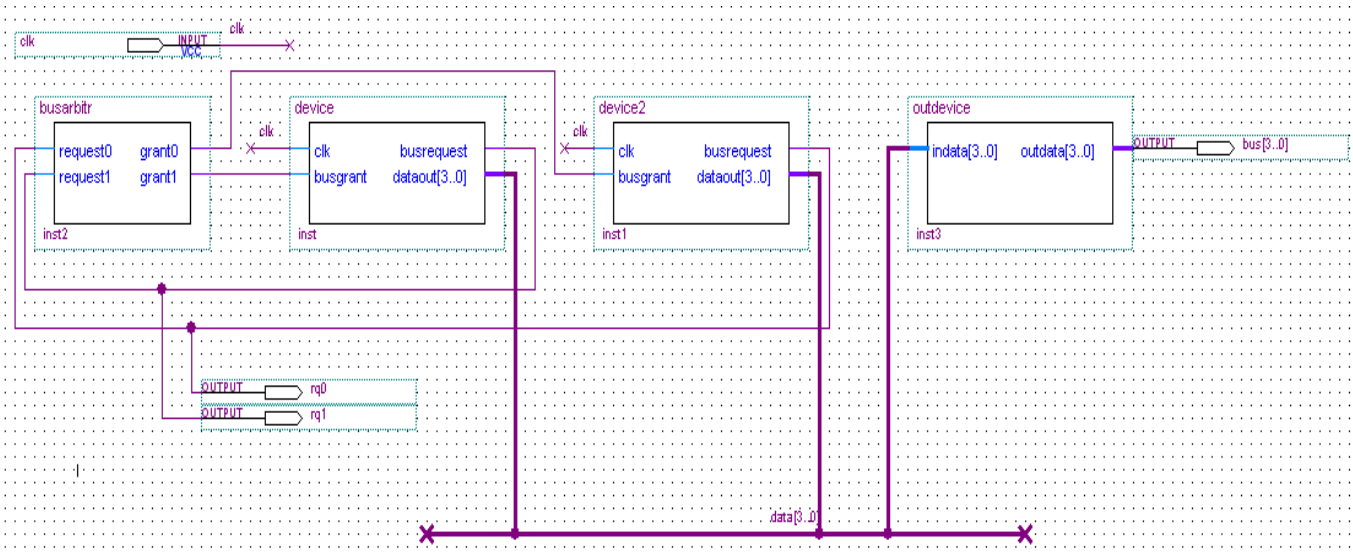


Рис. 1.8. Подключение устройств к шине данных

Подключение устройств к шине показано на рис. 1.8. Ведущие устройства в определенный момент генерируют сигнал запроса шины (ЗШ). Реализация ведущих показана на рис. 1.9. Использование счетчика в схеме ведущих устройств позволяет ведущему в определенный момент выдать сигнал ЗШ на выход. Первый ведущий отличается от второго максимальным числом счета в счетчике 4 и 7 соответственно (значения назначаются произвольно). Кроме того генерируется уникальное 4-х разрядное значение, которое пропускается на выходную шину данных при высоком уровне сигнала ПШ.

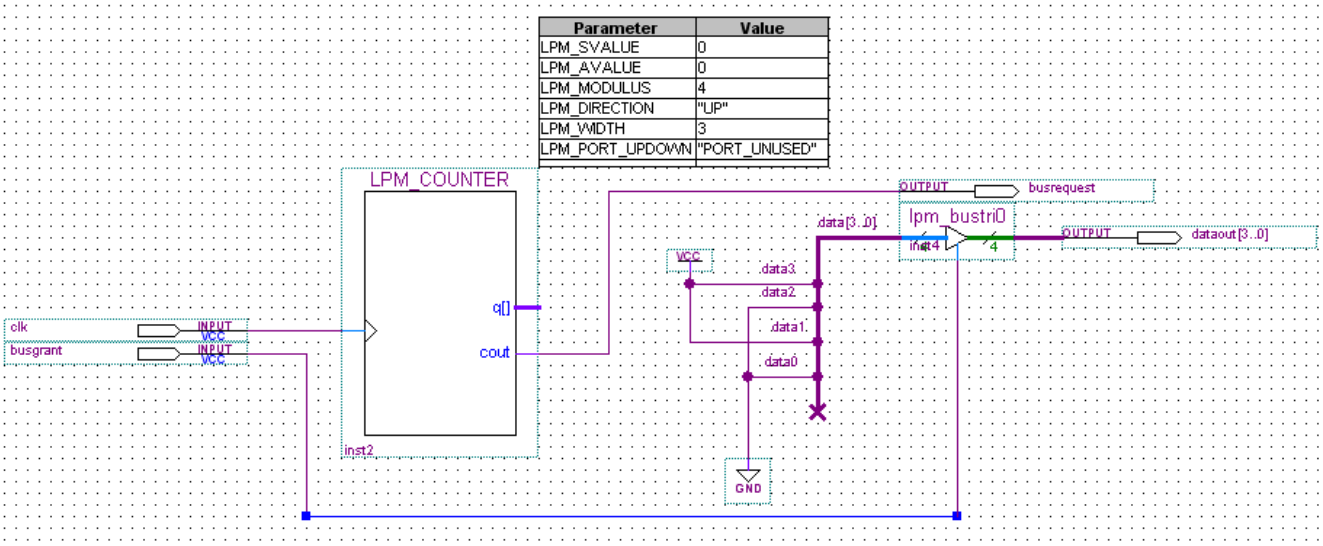


Рис. 1.9. Схема ведущих устройств

Расположение линий на входе центрального арбитра (рис. 1.10) определяет приоритет устройств. В зависимости от приоритета генерируются сигналы предоставления шины (ПШ) для каждого из ведущих устройств.

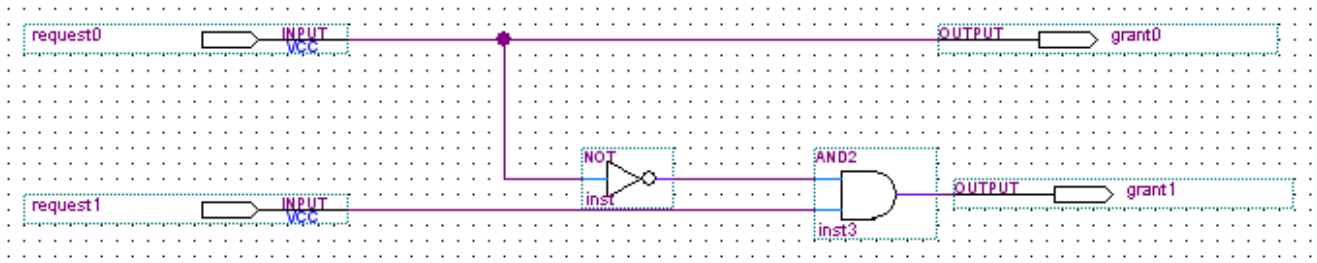


Рис. 1.10. Схема простейшего центрального арбитра для двух устройств

Ведомое устройство снимает приходящие по шине данные и выдает их на выходной порт. В случае, если вариант задания предусматривает арбитраж асинхронной шины, то ведомое устройство должно обеспечивать процедуру квитирования – т.е. обмена сигналами с ведомым устройством о готовности приёма данных и об его завершении.

Пример результатов функционального моделирования проекта приведён на рис. 1.11.

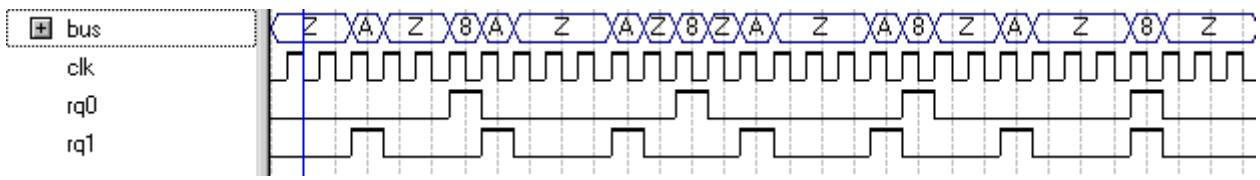


Рис. 1.11. Результаты моделирования

Таблица 1. Варианты заданий к лабораторной работе № 1

№	Вид арбитража	Вид подключения	Алгоритм смены приоритетов	Вид шины	Кол-во ведущих на шине
1	Централизованный	Параллельный	Циклический	Синхр.	4/6/8
2	Централизованный	Параллельный	Циклический с учётом последнего запроса	Асинхр.	4/6/8
3	Централизованный	Параллельный	Наиболее давнего использования	Синхр.	4/6/8
4	Централизованный	Параллельный	Фиксированный квант времени	Асинхр.	4/6/8
5	Централизованный	Последовательный	Статический (с двумя линиями приоритета)	Синхр.	4/6/8
6	Централизованный	Последовательный	С цепочкой сигнала ЗШ	Асинхр.	4/6/8
7	Централизованный	Последовательный	С цепочкой сигнала ПШ	Синхр.	4/6/8
8	Децентрализованный	Параллельный	Статический	Асинхр.	4/6/8
9	Децентрализованный	Кольцевой	Статический	Синхр.	4/6/8
10	Децентрализованный	Параллельный	Распределённый арбитраж с самостоятельным выбором	Асинхр.	4/6/8
11	Децентрализованный	Кольцевой	Наиболее давнего использования	Синхр.	4/6/8
12	Децентрализованный	Кольцевой	Циклический с учётом последнего запроса	Асинхр.	4/6/8

1.2.2. Содержание отчета по лабораторной работе №1

1. Вариант задания на лабораторную работу.
2. Схемы

2.1 Структурная схема подключения ведущих и ведомых устройств к шине.

2.2. Схемы ведущих и ведомого устройства.

2.3. Схема центрального арбитра либо блока арбитража ведомого устройства.

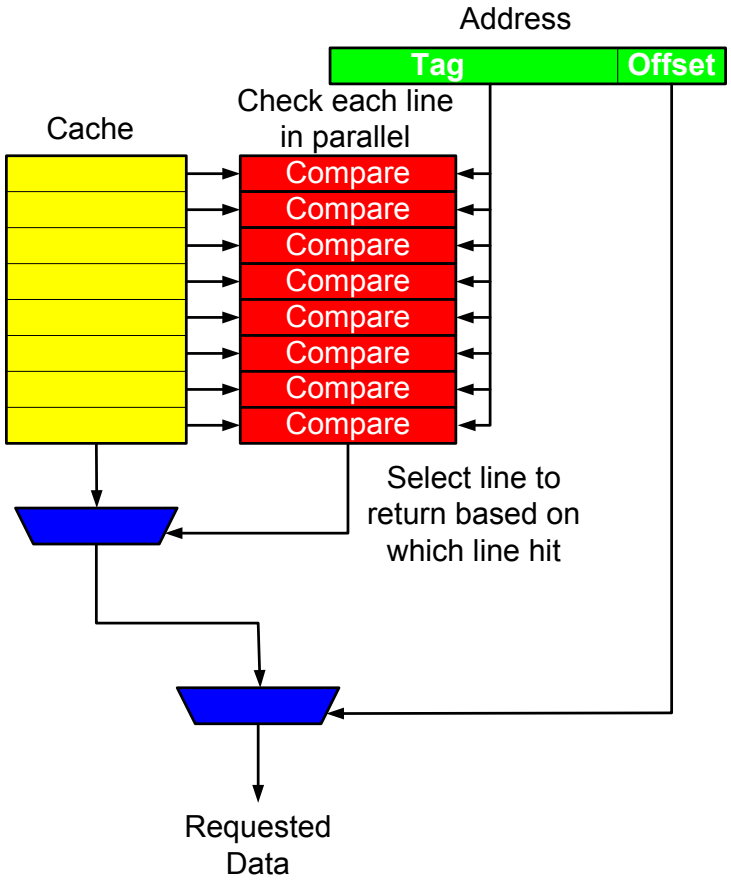
3. Результаты функционального моделирования.

ЛАБОРАТОРНАЯ РАБОТА № 2
"Организация кэш-памяти"

2.1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Быстродействие и цена микросхем памяти являются обратно-пропорциональными величинами. С целью минимизации стоимости вычислительных систем при сохранении их производительности на требуемом уровне используется иерархическая схема организации памяти. Помимо внутренней памяти центрального процессора (регистров) на кристалле ЦП так же располагается кэш-память, которая в зависимости от схемотехнического решения может подразделяться на первый и второй уровни.

При проектировании кэш-памяти учитывается прежде всего принцип локальности данных, который заключается в том, что если к одному слову основной памяти было произведено обращение процессора, то с большой вероятностью на следующих тактах ЦП обратится и к следующему за ним по адресу слову памяти. Это означает, что в кэш-памяти целесообразно хранить блоки



данных, объём которых, как правило, колеблется в диапазоне от 4 до 64 слов.

Рис. 2.1. Организация кэш-памяти с полностью ассоциативным отображением

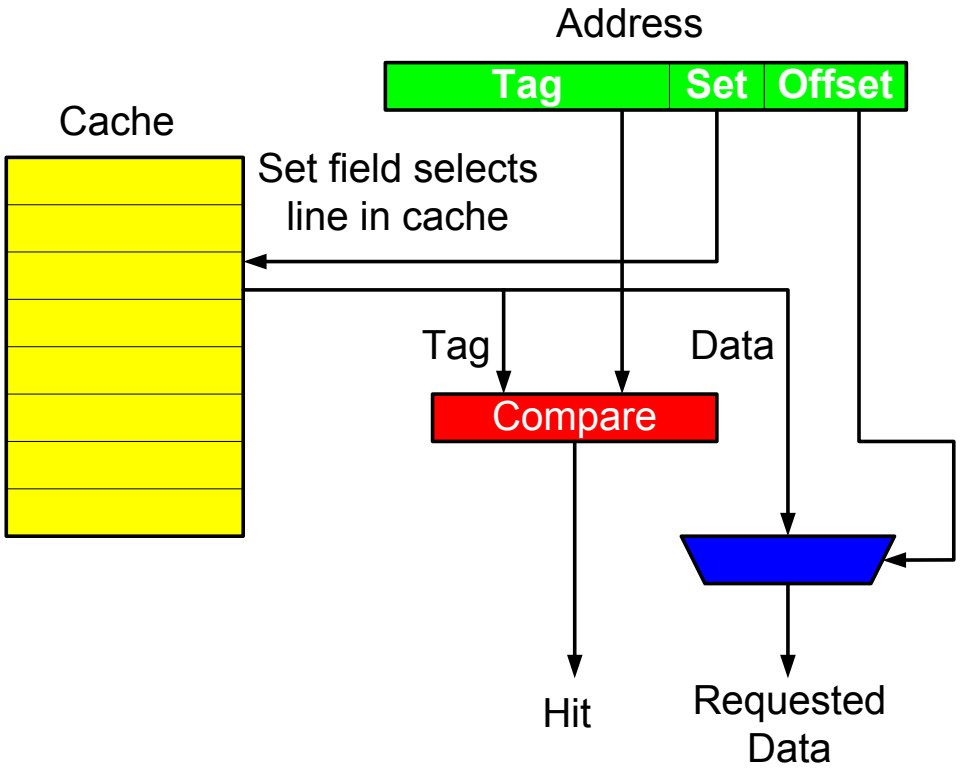
Каждый блок данных, помещаемый в кэш-память, сопровождается ассоциативным признаком, называемым тэгом (tag), по которому производится

идентификация содержимого блока данных. Комбинация тэга и блока данных называется строкой кэш.

По внутренней структуре и, соответственно, способу размещения данных ОЗУ в кэш-памяти, она может быть разделена на три типа:

- кэш с полностью ассоциативным отображением *(fully-associative cache)*;
- кэш с прямым отображением *(direct mapped cache);*
- кэш с множественно-ассоциативным отображением *(k-way associative cache).*

В кэш-памяти с полностью ассоциативным отображением любой блок данных из основной памяти может быть продублирован в любой строке кэш. Таким образом, до тех пор, пока в наличие имеются свободные строки кэш-память данного типа не может производить замещение хранимых в ней данных, что обеспечивает более эффективное заполнение кэш-памяти и снижение вероятности промаха. Схема организации данного типа памяти приведена на рис. 2.1. К сожалению, организация подобного типа кэш-памяти требует дополнительных аппаратных затрат, что удорожает в целом микросхему центрального



процессора.

Рис. 2.2. Организация кэш-памяти с прямым отображением

В кэш-памяти с прямым отображением *(direct mapped cache)*, каждому блоку памяти поставлена в соответствие отдельная строка кэш-памяти. Так как объём кэш-памяти в несколько десятков раз меньше объёма основной памяти, то на размещение в одну строку кэш будут претендовать несколько десятков

блоков памяти. В случае, если требуется записать блок данных из основной памяти в некоторую строку кэш, то запись производится в любом случае – была данная строка занята или нет. При этом запись будет производиться с замещением данных в строке, не смотря на то, что соседние строки могут быть в это время свободными. Таким образом, замещение данных в кэш-памяти подобного типа и, соответственно, использование всей кэш-памяти производится не самым рациональным способом.

Для того, чтобы определить какой блок памяти в какую строку кэш записывать обычно используют несколько бит адреса для задания отображения блоков памяти на кэш. Например, биты 2 и 3 адреса (в зависимости от размера блока и строки) могут быть индексом, задающим номер строки кэш для размещения очередного блока данных. Схема организации данного типа памяти приведена на рис. 2.2.

Достоинства и недостатки обоих типов можно свести к следующим тезисам:

- Кэш с прямым отображением – проще:
  - Занимает меньше места, реализуется меньшим количеством вентилей, требует малое количество бит для хранения тэга;
  - более быстрый в потенциале;
  - но, при этом – б***о***льшее количество промахов и замещений, соответственно – меньший КПД.
- Кэш с полностью ассоциативным отображением:
  - меньше промахов и конфликтов – лучшая производительность в итоге,
  - но требует отдельного компаратора тэгов для каждой строки, что существенно удорожает решение.

Для совмещения преимуществ обоих способов была разработана схема с множественно-ассоциативным отображением *(k-way associative cache)*. В русскоязычных источниках кэш с *n* множественно-ассоциативным отображением называют *n*-канальным [2,3].

В множественно-ассоциативной кэш-памяти несколько строк объединены в один набор *(set)*. В границах набора любой блок данных, который ассоциирован с данным набором может размещаться в любой строке, однако быть размещённым в строке другого набора блок данных не может. Таким образом, данный способ организации кэш-памяти совмещает в себе преимущества как полностью ассоциативной кэш-памяти, так и кэш-памяти с прямым отображением.

При задании множественно-ассоциативной кэш-памяти индекс *k* (либо количество каналов *n*) задаёт количество строк кэш-памяти в каждом из наборов кэш-памяти. В зависимости от значения параметра *k*, данная организация может вырождаться в вышерассмотренные случаи:

- k=1 – полностью ассоциативное отображение.
- k= количеству строк кэш-памяти – прямое отображение *(direct mapped)*.

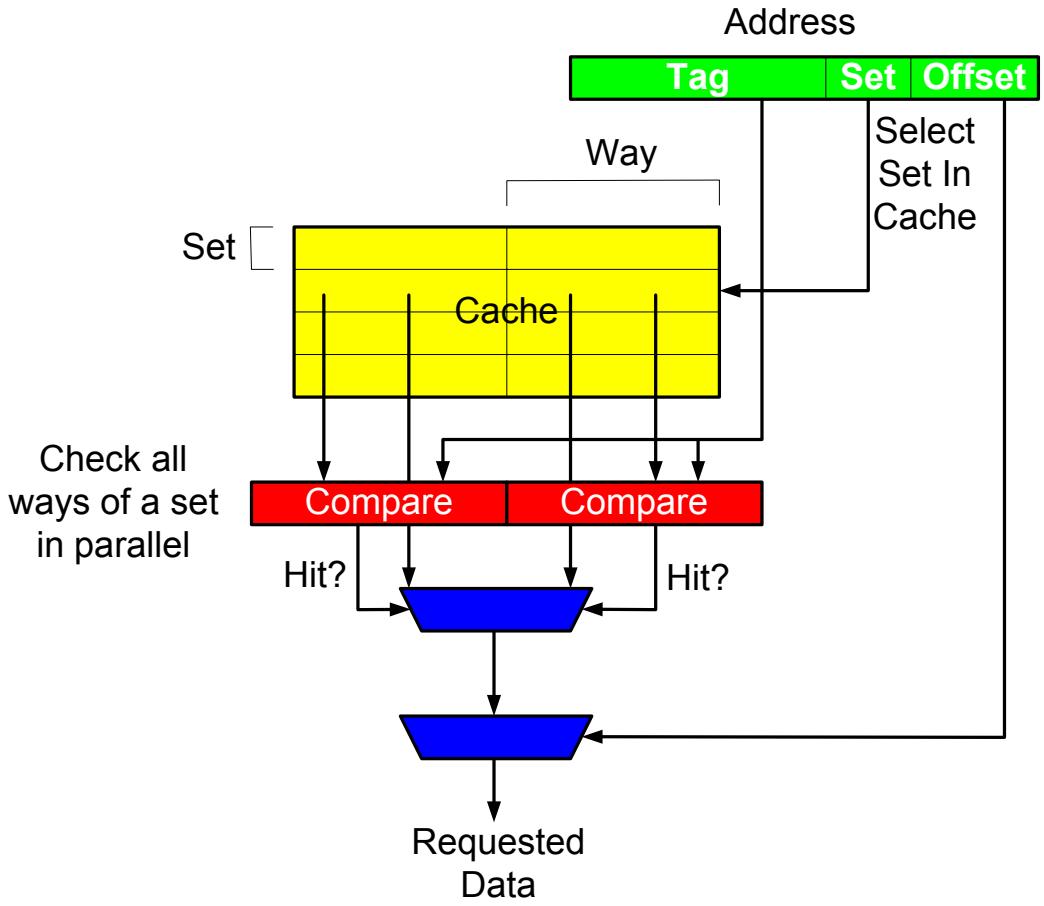


Рис. 2.3 Организация кэш-памяти с множественно-ассоциативным отображением (k=2)

В качестве ассоциативного признака (т.е. тэга) хранящегося в кэш-памяти блока данных используется старшая часть адреса, которая совпадает для всех слов, входящих в блок. Младшая часть адреса запрашиваемого центральным процессором слова используется в качестве смещения *(offset)* относительно начала блока данных для извлечения из него нужного слова. В зависимости от способа организации кэш-памяти (рис. 2.2. и 2.3), адрес запрашиваемого слова может делиться не на два, а на три поля – тэг, индекс *(set)* и смещение. В этом случае, индекс задаёт номер набора, в котором хранится блок данных, содержащий запрашиваемое слово.

Схемы замещения строк в кэш памяти могут быть следующими:

- без анализа предыдущих запросов (циклически либо случайно),
- по принципу наиболее давнего обращения,
- по принципу наиболее давнего хранения,
- по принципу наименьшего использования.

2.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

2.2.1. Задание и рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в новом проекте, не используя предыдущих наработок.

Задача состоит в создании кэш-памяти заданного типа, размера для обслуживания основной памяти с заданным типом замещения. Конкретные характеристики разрабатываемой кэш-памяти приведены в таблице вариантов 2. Размер шины адреса вычисляется исходя из варианта задания – по нему определяется размер кэш-памяти в байтах, затем определяется размер ОЗУ исходя из кратности его размера объёму кэш-памяти. Адресация ОЗУ по умолчанию – побайтная.

Разрешается использовать функциональное моделирование. В качестве элементов памяти для строк кэш-памяти удобно использовать регистры (например, параметризированный модуль LPM_DFF, в котором легко задать необходимую разрядность).

Содержимое кэш-памяти следует просматривать непосредственно на элементах памяти без вывода на выходные пины, поскольку использование большого количества пинов не позволит Quartus найти подходящую ПЛИС, на которой можно будет реализовать проект.

Таблица 2. Варианты заданий к лабораторной работе № 2

№	Тип кэш-памяти	Количество строк кэш	Количество слов в строке	Количество строк в наборе	Принцип замещения строк в кэш	Кратность размера ОЗУ объёму кэш-памяти	Разрядность шины данных, бит
1	Полностью ассоциативное отображение	4	2/4	1	Без анализа	16	8/16
2	---//---	4	2/4	1	LRU		8/16
3	---//---	4	2/4	1	Наиб. давнего хранения		16
4	---//---	4	4	1	Наименьшего использования	24	16
5	---//---	4	8	1	Случайный	12	16
6	Прямое отображение	8	8	1	Нет	16	8/16
7	---//---	10	4	1	---//---	12	16
8	---//---	12	4	1	---//---	16	16
9	---//---	16	2	1	---//---	10	8/16

Продолжение таблицы 2.

№	Тип кэш-памяти	Количество строк в кэш	Количество слов в блоке данных	Количество модулей, *k*	Принцип замещения строк в кэш	Кратность размера ОЗУ объёму кэш-памяти	Разрядность шины данных, бит
10	k-мерное частичное ассоциативное отображение	4	4/8	2	Без анализа	24	8/16
11	---//---	6	2/4	2	LRU	16	8/16
12	---//---	8	2/4	2	Наиболее давнего хранения	12	8/16
13	---//---	10	2	2	Наименьшего использования	16	8/16
14	---//---	12	2/4		Без анализа	8	8/16
15	---//---	4	2/4	4	LRU	10/12	8/16
16	---//---	8	2/4	4	Наиболее давнего хранения	12/16	8/16
17	---//---	16	2/4	4	Наименьшего использования	8/10	8/16
18	---//---	4	2	8	LRU	10	16
19	---//---	6	4	8	Наиболее давнего хранения	8	8

2.2.2. Содержание отчета по лабораторной работе №2

1. Задание.
1.1 Расчёт объёма кэш-памяти и ОЗУ.
1.2 Разбиение адреса на поля – тэга, индекса и смещения.
2. Схема кэш-памяти.
3. Результаты функционального моделирования

ЛАБОРАТОРНАЯ РАБОТА № 3
"Взаимодействие кэш-памяти и ОЗУ"

3.1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Построение многоуровневой памяти приводит к необходимости решения вопроса согласования хранимых на разных уровнях иерархии данных в случае их изменения центральным процессором или же путём прямого доступа в память (ПДП) устройства ввода-вывода. Вследствие максимального динамизма обмена данными названная задача особенно остро проявляется между уровнями кэш – ОЗУ.

Управление хранением данных отличается от управления загрузкой их в регистры ЦП в силу следующих причин:

- сохранение данных не требует простоя ЦП;
- сохранение меняет содержимое кэш;
- многие устройства ввода/вывода имеют возможность ПДП.

Существует два основных способа (рис. 3.1) организации процесса сохранения данных в уровнях памяти в случае их изменения центральным процессором:

- отложенная запись *(Write-back),*
- сквозная запись *(Write-Through)*.

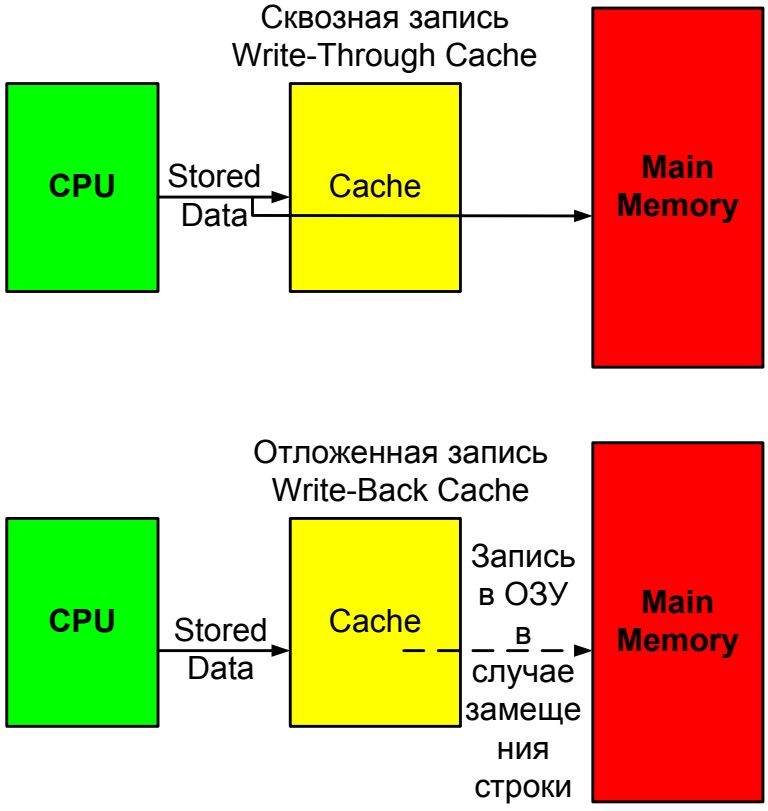


Рис. 3.1. Методы синхронизации данных между кэш-памятью и ОЗУ

В случае **сквозной записи** прежде всего обновляется слово, хранящееся в основной памяти. Если в кэш есть копия слова, то она также обновляется. В случае если копии нет, то возможны два варианта:

- из основной памяти в кэш пересылается слово с ожиданием, что скоро появится необходимость обращения к изменённому слову. Подобная стратегия обновления называется *сквозная запись с отображением*.

- в кэш ничего не добавляется. В таком случае при последующем обращении к изменённому слову будет сгенерирован сигнал кэш-памяти «промах» и слово будет загружено в кэш из ОЗУ. Данная стратегия носит название – *сквозная запись без отображения*.

Главным достоинством метода сквозной записи является отсутствие необходимости ожидания на «выгрузку» в ОЗУ сохранённого слова при замещении его в кэш-памяти другим словом (блоком данных). Изменённые данные уже синхронизированы с основной памятью и, таким образом, изменение логики работы кэш-памяти при загрузке в неё данных не требуется. Соответственно, указанный метод является простым с точки зрения аппаратной реализации.

Недостатком метода будет являться потеря времени при каждой операции записи данных центральным процессором. Данный способ синхронизации содержимого кэш-памяти и ОЗУ применялся в i80486.

Для улучшения производительности систем, реализующих данный подход, применяется *буферизированная сквозная запись (write behind)*. В систему добавляется буферная память, работающая по принципу FIFO и независящая от ядра процессора. В случае записи изменённых данных они сохраняются в кэш-память и буфер. Соответственно, центральному процессору нет необходимости ожидать пока буфер запишет слово в основную память. Таким образом, при данной реализации кэш-памяти центральный процессор не взаимодействует с ОЗУ вовсе (!) – чтение производится из кэш, запись – в буфер.

При повышении тактовой частоты системной шины и, тем более, шины ЦП – память возникает необходимость минимизации ненужных транзакций шины. С этой точки зрения существует ненулевая вероятность того, что данные могут быть перезаписаны несколько раз, прежде чем ЦП понадобится их считать и тем, более, прежде чем они будут замещены в кэш-памяти новым блоком данных.

При реализации способа **отложенной записи** *(write back)* изменённое слово заносится только в кэш-память. Если строки в кэш-памяти нет, то сначала производится загрузка строки в кэш-память, потом её изменение. При замещении строки сперва производится её обязательная запись в основную память, а уже затем загрузка нового блока данных.

Для данного метода характерно, что после заполнения кэш-памяти при каждом промахе при чтении будут осуществляться две пересылки между ОЗУ и кэш памятью – сохранение выгружаемого (изменённого) блока данных и загрузка в кэш нового.

С целью минимизации трафика на шине в кэш-памяти для каждой строки вводится дополнительный бит – флаг изменения. Если в строке было изменение хотя бы одного слова, то флаг устанавливается в 1, соответственно при замещении блока данных запись в ОЗУ производится только для строк, у которых флаг имеет значение 1. Данный подход носит название – *флаговая отложенная запись*.

- Отложенная запись *(write-back cache)* в среднем на 10% эффективнее:
  - может записывать сразу всю строку (объединяя несколько операций записи в одну),
  - использует свойство локальности ссылок.
- Сквозная запись *(write-through cache)* проще в реализации:
  - не требует флага – была строка записана или же нет,
  - не требует ожидания (пока появится место) на запись строки в ОЗУ при её замещении в кэш,
  - не требует специальных интерфейсов с I/O.

3.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

3.2.1. Задание и рекомендации по выполнению лабораторной работы

При выполнении лабораторной работы необходимо использовать наработки из предыдущей лабораторной работы.

К архитектуре кэш-памяти из предыдущей лабораторной работы добавить блок синхронизации содержимого кэш-памяти и ОЗУ.

Варианты способов обмена кэш-памяти и ОЗУ приведены в таблице 3. Разрешается использовать функциональное моделирование.

Таблица 3. Варианты заданий к лабораторной работе № 3

№	Способ синхронизации данных	Сложность варианта (0-5)	Уточнение варианта
1	Сквозная запись	2	сквозная запись с отображением
2	--//--	1	сквозная запись без отображения
3	--//--	5	буферизированная сквозная запись
4	Отложенная	3	простая отложенная запись
5	запись	4	флаговая отложенная запись

3.2.2. Содержание отчета по лабораторной работе №3

1. Задание.
2. Описание работы схемы при промахе/попадании операций чтение/запись.
3. Схемы подключения и схемы управления связкой «кэш-память – ОЗУ».
4. Результаты функционального моделирования

ЛАБОРАТОРНАЯ РАБОТА № 4

"Динамическое предсказание условных переходов"

4.1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

При построении конвейеризированных вычислительных систем разработчикам приходится решать задачи, не возникающие при проектировании обычных однотактовых или многотактовых процессорных устройств. Помимо задач устранения рисков по данным, а так же структурных рисков, возникает проблема оптимизации вычислительных затрат ЦП при обработке команд условных переходов (УП). В отличие от команды принудительного перехода – т.е. JMP, условный переход может состояться или нет. В случае обработки потока команд конвейером вычисление условия будет произведено уже после того, как первые стадии конвейера будут заполнены командами, идущими в естественной последовательности за ещё не проверенной командой УП. В случае если переход состоится, то первые стадии конвейера должны быть очищены и заполнены командами, начиная с адреса перехода. Данный процесс приводит к задержкам работы конвейера, что негативно сказывается на общей производительности вычислительной системы.

Для оптимизации загрузки конвейера проблема условного перехода решается с помощью различных способов, которые можно разделить на две группы – статические и динамические. Примерами статических способов являются две простейшие стратегии поведения при прогнозировании переходов – «Переход состоится всегда» и «Переход не состоится никогда».

В рамках лабораторной работы необходимо аппаратно реализовать один из вариантов динамических схем предсказания исходов команд условных переходов.

Все схемы, реализующие динамическое предсказание УП – предполагают накопление информации об исходе предшествующих команд УП. В зависимости от типа схемы предсказание делается либо на основе предшествовавших исходов именно этой команды (располагающейся в памяти по текущему адресу) либо на основе предыдущих исходов нескольких других команд УП, содержащихся в коде программы, либо их комбинации.

Для упорядоченного хранения информации об исходах различных команд УП (либо их последовательностей), как правило, используется таблица истории шаблонов (PHT – *Pattern History Table*). В каждой из строк, которой хранится так называемый шаблон вызова и соответствующее ему состояние автомата предсказания (рис. 4.1).

Автомат предсказания – один из нескольких вариантов (в рамках данной лабораторной работы) конечных автоматов Мура. На основании текущего состояния автомата делается прогноз об исходе поступившей команды УП. Состояния различных вариантов реализации автоматов и переходы между ними приведены на рис. 4.2 – 4.6. Наименования приведенных автоматов слегка различаются в различных источниках [3,4], но схемы их работы совпадают.

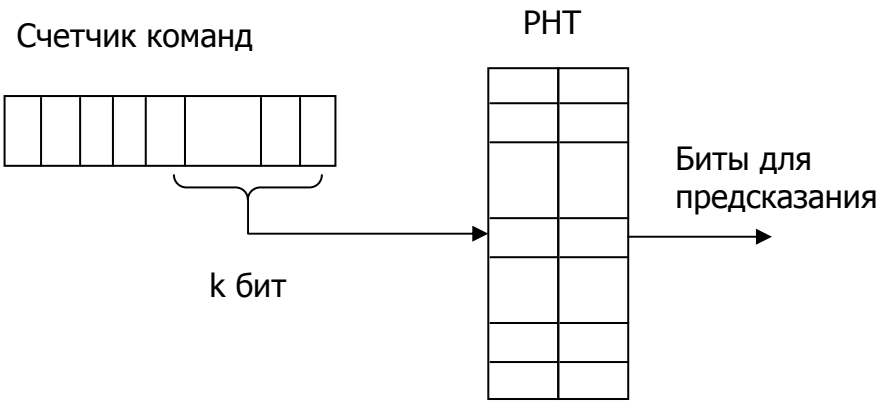


Рис. 4.1. Пример организации таблицы истории шаблонов, в качестве шаблона, в данном случае, используется часть адреса команды УП

Простейшим вариантом автомата является схема А1, которая представляет собой тривиальное хранение предыдущего исхода команды условного перехода (рис. 4.2). Предсказание перехода делается до момента вычисления условия в соответствии с состоянием автомата – переход будет, если предыдущее выполнение команды окончилось переходом либо переход не состоится, если в предыдущий раз перехода не было. В соответствии с полученным предсказанием производится заполнение начальных стадий конвейера команд – либо командами, следующими за командой УП, либо командами, находящимися по адресу перехода. По вычислении условия перехода автомат переводится в соответствующее состояние, которое будет в свою очередь использовано для предсказания перехода в следующий раз, когда данная команда УП поступит на конвейер.

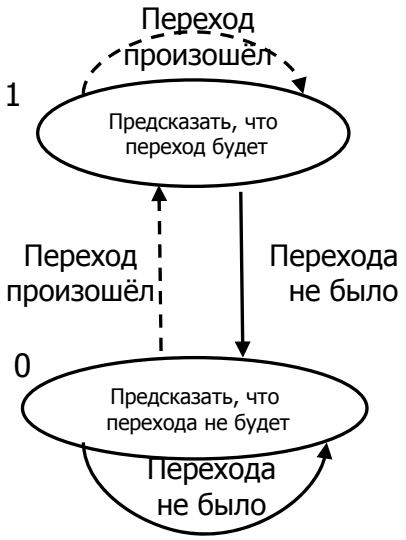


Рис. 4.2 Однобитовый автомат А1 – «повтор предыдущего исхода»

При условии использовании автомата А2 после обработки очередной команды УП содержимое ячейки РНТ, хранящей состояние автомата изменяется.

Так как автомат А2 функционирует в режиме регистра сдвига, то можно сказать, что содержимое соответствующей ячейки РНТ сдвигается влево на один разряд, а на освободившееся место заносится 1, если переход был, или 0 – если не было.

Если в ячейке РНТ есть хоть одна 1, то делается предсказание, что переход будет, в ином случае – перехода не будет.

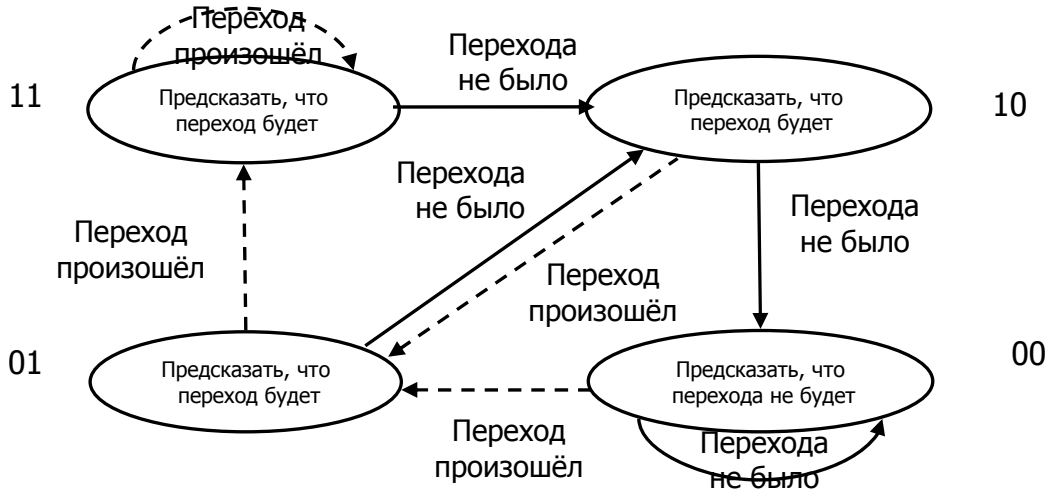


Рис. 4.3. Автомат А2: состояния, схема переходов из одного состояния в другое и предсказания, выдаваемые схемой в зависимости от состояния

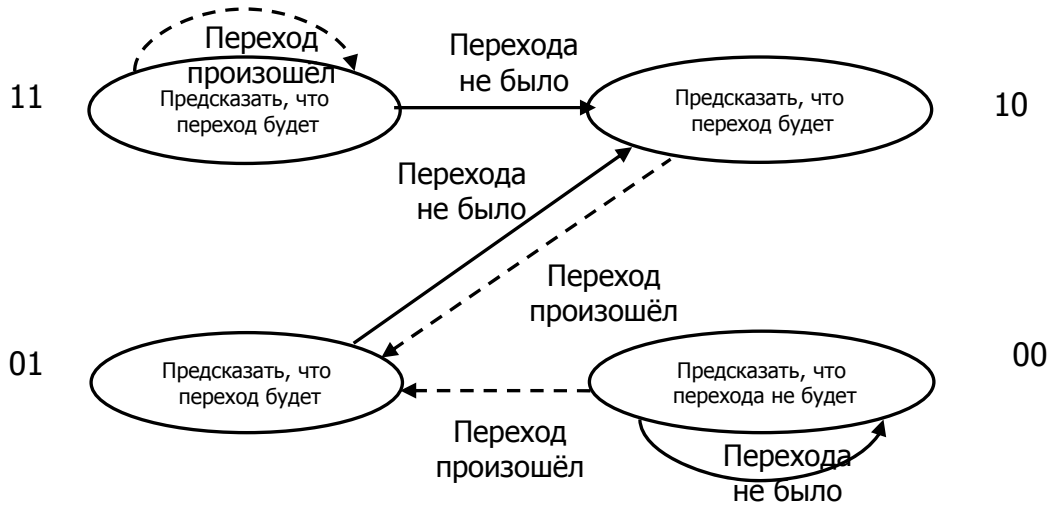


Рис. 4.4. Автомат А3: состояния, схема переходов из одного состояния в другое и предсказания, выдаваемые схемой в зависимости от состояния

В случае использования автомата А3, элементы РНТ представляют собой реверсивные счётчики, работающие в режиме с насыщением (минимум 00, мак-

симум 11). Логика предсказания для автомата А3 получила название «алгоритм Смита». Автомат может находиться в четырёх состояниях:

00 – перехода не будет (сильное предсказание);
01 – перехода не будет (слабое предсказание);
10 – переход будет (слабое предсказание);
11 – переход будет (сильное предсказание).

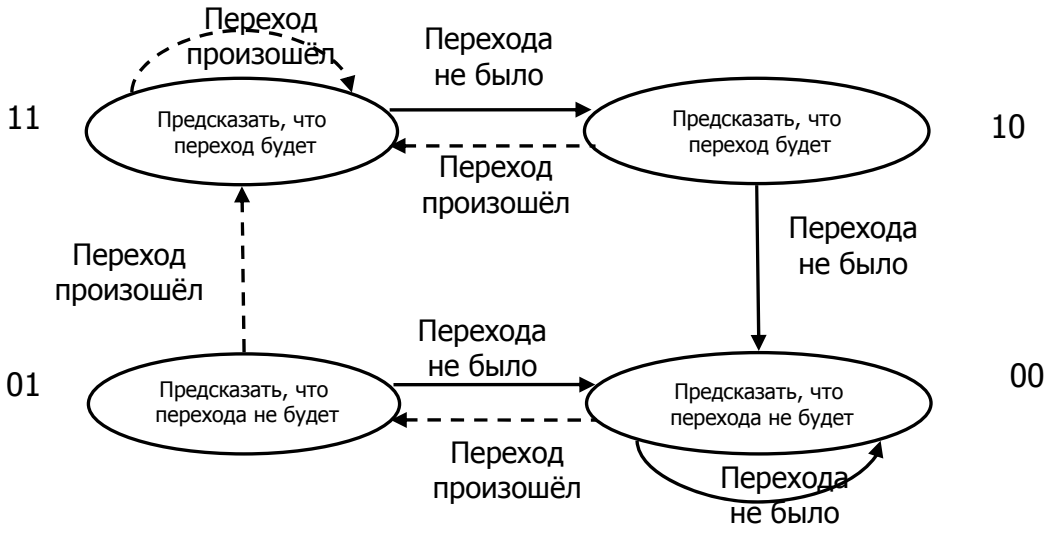


Рис. 4.5. Автомат А4: состояния, схема переходов из одного состояния в другое и предсказания, выдаваемые схемой в зависимости от состояния

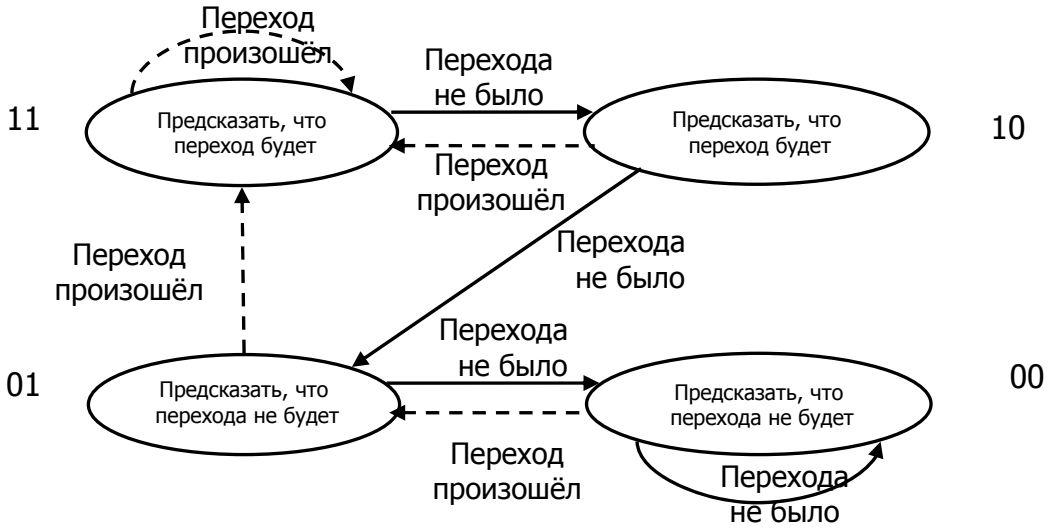


Рис. 4.6. Автомат А5: состояния, схема переходов из одного состояния в другое и предсказания, выдаваемые схемой в зависимости от состояния

Варианты автоматов А4 и А5 являются вариациями автомата А2. Их состояния приведены на рис. 4.5 и 4.6 соответственно.

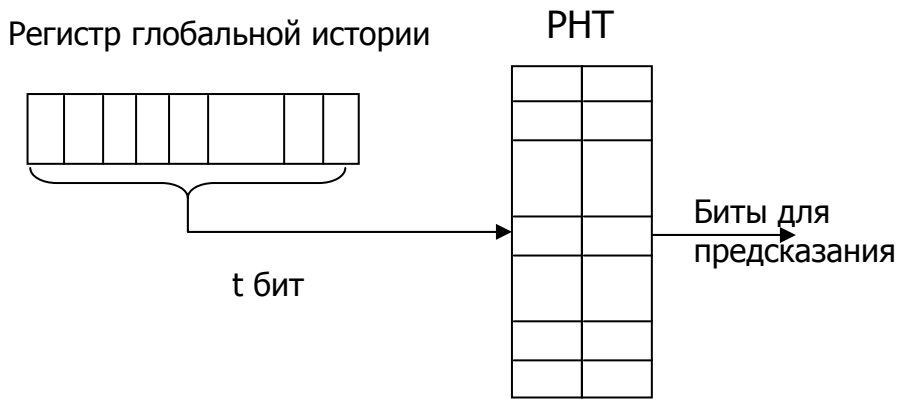


Рис. 4.7. Пример организации таблицы истории шаблонов, в качестве шаблона, в данном случае, используется регистр глобальной истории

Как отмечалось выше, таблица истории шаблонов представляет собой универсальный инструмент – в качестве шаблона могут использоваться различные по исходному смыслу битовые комбинации. Так на рис. 4.1 представлен случай, когда в качестве шаблона используются младшие биты адреса команды УП. В свою очередь на рис. 4.7. изображена схема, в которой в качестве шаблона используется так называемый регистр глобальной истории, который хранит общую информацию об исходах последних *t* команд условного перехода вне зависимости от их месторасположения в коде программы.

Как ни парадоксально с точки зрения логики, но лучшие результаты по точности предсказания показывают схемы, комбинирующие оба приведённых выше подхода. На рис. 4.8 приведён пример, демонстрирующий подобную комбинацию.

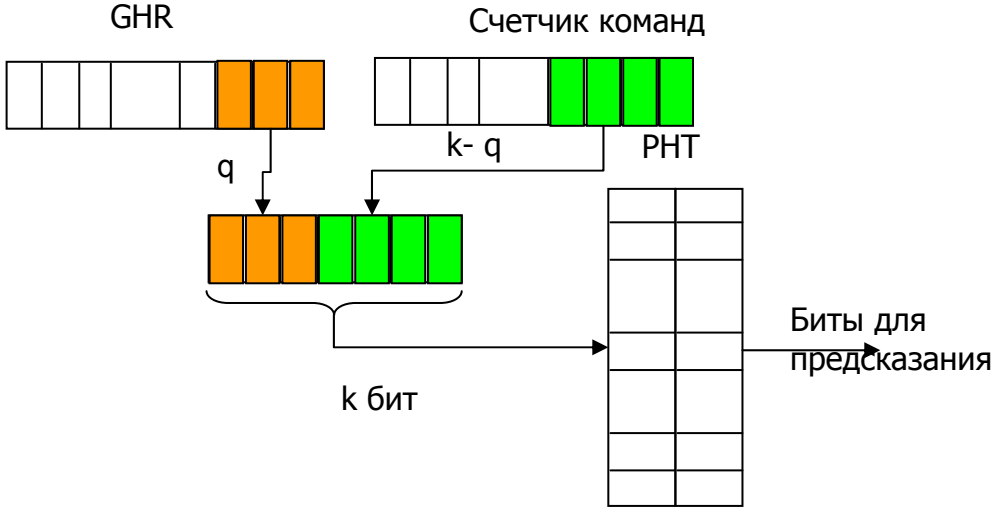


Рис. 4.8. Пример организации таблицы истории шаблонов, в качестве шаблона использована конкатенация битов регистра глобальной истории и счётчика команд

При разработке реальных схем предсказания исходов условных переходов инженера сталкиваются с так называемой проблемой «холодного старта». Она заключается в том, что до тех пор, пока таблица истории шаблонов не будет за-

полнена реальными данными, схема не будет предсказывать переходы с необходимой точностью. Для решения данной проблемы были разработаны двух уровневые схемы предсказания, суть которых сводится к использованию нескольких принципиально отличающихся схем предсказания переходов, которые дают различную точность работы на этапе насыщения схем реальными данными и в режиме штатной работы. Решение о выборе предсказания конкретной схемы принимается селектором, который учитывает положительные результаты предсказания каждой из схем и выбирает ту, которая в последние N раз ошибалась меньше. Пример подобной схемы приведён на рис. 4.9.

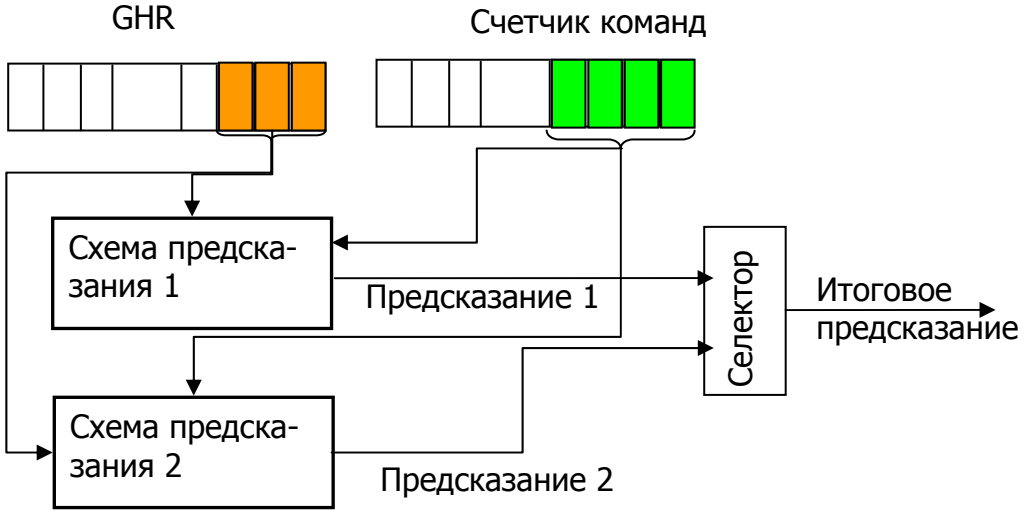


Рис. 4.9. Пример двухуровневой схемы предсказания

4.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

4.2.1. Задание и рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в новом проекте, не используя предыдущих наработок.

Задача состоит в создании блока предсказания переходов в соответствии с вариантом задания и моделирования его работы. Адреса команд условного перехода, так же как и их исход можно задавать в ручном режиме. Непосредственно выполнение самих команд перехода можно не моделировать. Основным требованием является задержка задаваемого в ручном режиме результата исхода команды УП по отношению к её адресу и предсказанию, выдаваемому блоком предсказания. Варианты задания лабораторной работы приведены в таблице 4. Разрешается использовать функциональное моделирование.

При выполнении задания необходимо обратить внимание на приведенные ниже моменты.

1. Моделирование потока команд УП можно задавать единичными битами выставляемыми на отдельных сигнальных линиях, каждая из которых за-

креплена за определённой командой УП. При этом в каждый отдельный момент времени (в рамках тактового сигнала) должна быть активирована лишь одна линия. Альтернативным вариантом задания команд УП является задание адреса конкретной команды на шине адреса ЦП.

2. Для моделирования исхода УП (который в реальности определяется состоянием флагов процессора) можно использовать отдельную сигнальную линию. Наличие единичного сигнала на ней в какой-либо момент времени обозначает положительный исход той команды УП, бит которой активирован в этот момент на сигнальных линиях команд УП, либо адрес которой выставлен на шине адреса. Нулевой сигнал соответствует отрицательному исходу (т.е. переход не состоялся) соответствующей команды УП.

3. В случае необходимости использования счётчика команд в схеме, адреса команд УП задаются произвольно – в ручном режиме.

4. На выходе схемы должен генерироваться сигнал о предсказании переходов поступающих команд УП.

Таблица 4. Комбинации параметров схем предсказания переходов для формирования вариантов заданий к лабораторной работе № 4

№	Тип автомата	Количество уровней схемы	Количество бит шаблона	Способ формирования шаблона
1	A2	1	8/16/32	GHR / PC/ GHR||PC / GHR⊕PC
2	A3	1	8/16/32	GHR / PC/ GHR||PC / GHR⊕PC
3	A4	1	8/16/32	GHR / PC/ GHR||PC / GHR⊕PC
4	A5	1	8/16/32	GHR / PC/ GHR||PC / GHR⊕PC
5	A1 & A2	2	8/16	GHR / PC/ GHR||PC / GHR⊕PC
6	A1 & A3	2	4/8/16	GHR / PC/ GHR||PC / GHR⊕PC

4.2.2. Содержание отчета по лабораторной работе №4

1. Задание.
2. Общая схема динамического предсказания исходов команд УП.
3. Результаты моделирования схемы.
4. Статистика результатов работы схемы – общее количество поступивших команд УП, количество совершённых и несовершённых переходов, соотношение удачных предсказаний к общему количеству переходов.

ЛАБОРАТОРНАЯ РАБОТА № 5
"Моделирование работы конвейера команд центрального процессора"

5.1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Конвейеризация любого процесса подразумевает его разделение на более-менее одинаковые по длительности этапы и позволяет выполнять эти этапы одновременно и параллельно для нескольких объектов, над которыми производится конвейеризируемый процесс. В вычислительной технике метод конвейеризации используется достаточно широко – допускается как конвейеризация транзакций шин, обращений к памяти, так и конвейеризация процесса выполнения самих команд центрального процессора.

Как правило, цикл выполнения центральным процессором любой команды состоит из следующих этапов:

1. Выборка команды (ВК).
2. Декодирование команды (ДК).
3. Вычисление адресов операндов (ВА).
4. Выборка операндов (ВО).
5. Исполнение команды (ИК).
6. Запись результата (ЗР).

Не смотря на гипотетическую возможность выравнивания большинства из приведенных этапов по времени на практике все этапы выполняются различное количество тактов. При этом основная проблема заключается не в разнице скоростей доступа к памяти и, например, АЛУ. Сложность состоит в том, что команды могут иметь различный формат, различное количество операндов, которые могут находиться как в регистрах процессора, так и в памяти, а также быть заданы с помощью достаточно большого количества вариантов адресации операндов.

В соответствии с непредсказуемостью поступающих на вход команд, а так же взаимосвязью этапов конвейера, в большинстве случаев должен быть предусмотрен механизм приостановки работы соответствующей стадии конвейера, если последующая стадия не успевает принять результаты её работы. При этом, если возможная нерегулярность по времени работы присуща всем стадиям конвейера, становится возможной определённая самокомпенсация по времени исполнения этапов конвейера, которая может быть задействована путём введения достаточных по ёмкости буферов для хранения промежуточных данных между стадиями конвейера.

Тем не менее, основная проблема конвейеризации любого процесса может быть сведена к тюркской пословице – скорость каравана соответствует скорости его самого медленного верблюда, или иными словами – если одна из стадий явно медленнее других, то общая производительность конвейера будет соответствовать скорости именно этой стадии.

В отличие от однотактовой и многотактовой схем построения центральныхпроцессоров, в конвейеризированном процессоре должны быть явно выде-

лены и синхронизированы между собой регистры хранения промежуточных результатов. Пример подобной схемы приведён на рис. 5.1. Так как схема является структурной на ней не показаны сигналы управления выполнением каждого из этапов, перевода его в случае необходимости в режим «холостого хода» и возобновления работы.

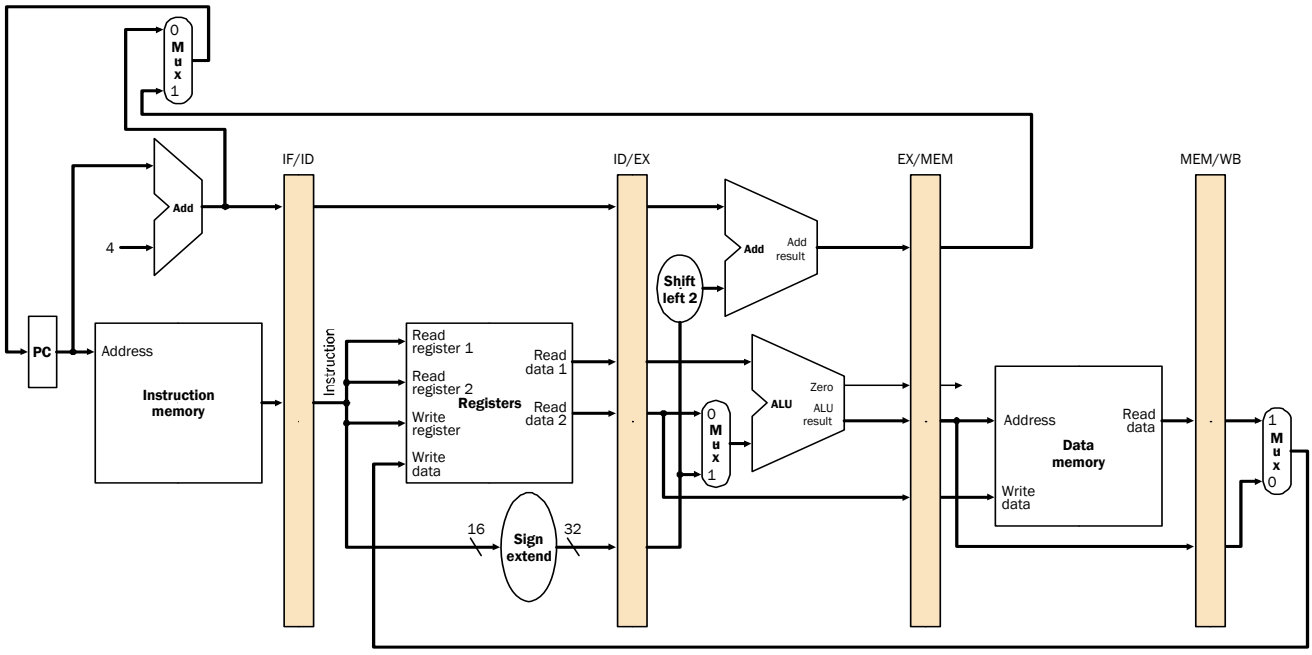


Рис. 5.1. Пример схемы конвейера команд процессора

5.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

5.2.1. Задание и рекомендации по выполнению лабораторной работы

Лабораторную работу рекомендуется выполнять в новом проекте, не используя предыдущих наработок.

Задача состоит в воссоздании трёх первых стадий конвейера команд: выборки команды, декодирования команды и выборки операндов из памяти. Выполнение самих команд арифметико-логическим устройством можно сделать по желанию с использованием проекта лабораторной работы из предыдущего семестра. Варианты задания лабораторной работы приведены в таблице 5. Разрешается использовать функциональное моделирование.

Минимальный вариант выполненной лабораторной работы должен содержать кэш-память команд, блок формирования адреса команды, блок декодирования команды, блок выборки операндов, кэш-память данных и соответствующие шины – адреса и данных. Пример подобной схемы (за исключением кэш-памяти данных) показан на рис. 5.2.

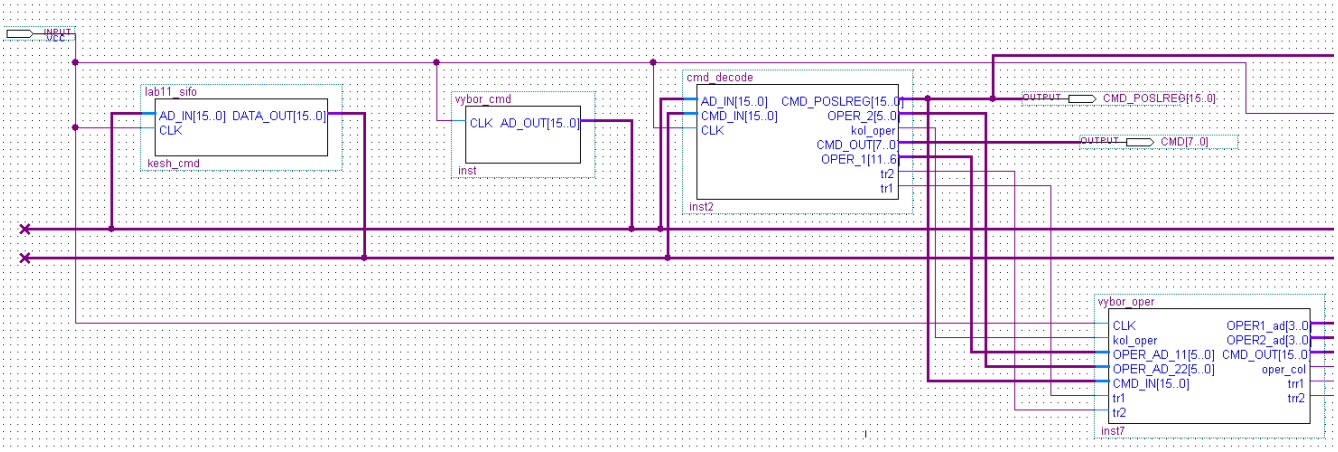


Рис. 5.2. Пример компоновки блоков конвейера команд

Основным требованием к выполненной лабораторной работе является демонстрация задержки выполнения команды в период её прохождения по стадиям конвейера. Для каждой команды в результатах моделирования должны быть отражены – её адрес, её код операции, адреса операндов, и значения самих операндов, извлечённых из кэш-памяти данных либо регистров. Пример, корректных результатов моделирования лабораторной работы приведён на рис. 5.3.

Для сдачи варианта лабораторной работы повышенной сложности требуется реализовать механизм задержки (перевода их в режим холостого хода) предыдущих стадий конвейера, если стадия выборки операндов не успевает обслужить текущую команду до поступления результатов обработки следующей команды с предшествующих стадий.

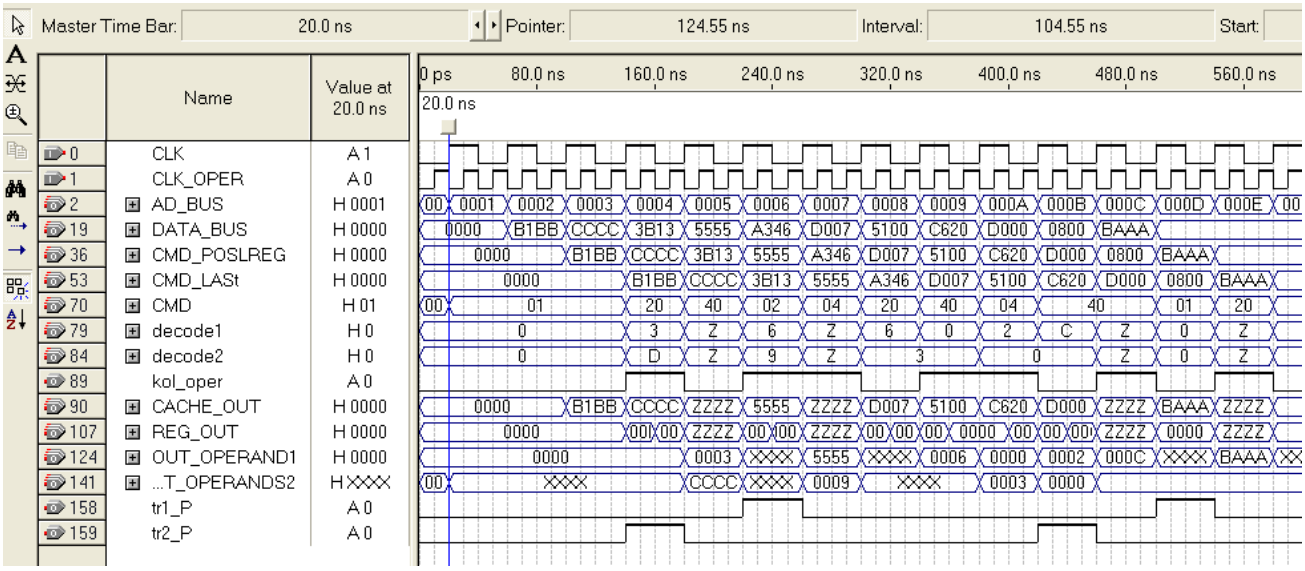


Рис. 5.3. Пример результатов моделирования конвейера команд процессора

Таблица 5. Варианты заданий к лабораторной работе № 4

№	Разряд-ность шины данных	Разряд-ность шины адреса	Количе-ство опе-рандов в команде	Количество реги-стров	Типы операндов: M –память, R - регистр
1	8	8	0/3	16	MRR, RRR
2	8	12	1/2	24	M, R, MR, RR
3	16	16	0/2	16	MR, RR
4	16	8	0/3	16	MRR, RRR
5	4	6	0/1	12	M, R
6	8	12	1/3	24	MRR, RRR
7	4	8	1/3	8	MRR, RRR
8	8	8	0/2	8	MR, RR
9	8	16	0/2	8	MR, RR
10	16	12	1/2	32	M, R, MR, RR
11	4	6	1/2	16	M, R, MR, RR
12	8	20	0/1	32	M, R
13	8	8	1/2	16	M, R, MM
14	8	4	1/3	16	M, R, MMR
15	16	8	2/3	16	MM, MR, RR, MRR
16	16	10	2/3	32	MM, MR, RR, MRR, RRR

5.2.2. Содержание отчета по лабораторной работе №5

1. Задание. Формат команд.
2. Общая схема конвейера команд.
3. Результаты моделирования схемы.

ТРЕБОВАНИЯ К КУРСОВОМУ ПРОЕКТУ ПО ДИСЦИПЛИНЕ «СТРУКТУРНАЯ И ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ ЭВМ»

6.1 СОДЕРЖАНИЕ КУРСОВОГО ПРОЕКТА СиФО ЭВМ

Курсовой проект по дисциплине СиФО ЭВМ является логическим завершением лекционного материала и обобщением обоих лабораторных практикумов по курсу. Тема курсового проекта – «Разработка микро-ЭВМ на ПЛИС».

При разработке курсового проекта настоятельно рекомендуется использовать наработанные в ходе выполнения лабораторных работ схемотехнические решения отдельных блоков.

Курсовой проект должен содержать:

- пояснительную записку объемом не менее 30 страниц, выполненных шрифтом Times New Roman 14, с одинарным междустрочным интервалом.
- графическую часть, содержащую не менее чем, четыре чертежа формата А3,
- компакт диск с исходными схемами проекта, выполненными в САПР Altera Quartus v. 7.11 – v. 9.0.

Оформление чертежей курсового проекта должно соответствовать требованиям, приведенным в методических указаниях к дипломному проектированию [15].

6.1.1. Содержание пояснительной записки

Пояснительная записка должна содержать следующие обязательные разделы:

Лист задания с вариантом разрабатываемой микро-ЭВМ (1 стр.)

Введение - реферативное описание архитектурных свойств заданного варианта микро-ЭВМ (1 стр.)

1. Разработка общей структуры микро-ЭВМ (6-9 стр.)

1.1. Функциональный состав микро-ЭВМ (2-3 стр.)

1.2. Разработка системы команд (2-3 стр.)

1.3. Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы (2-3 стр.)

2. Разработка основных устройств микро-ЭВМ (11-18 стр.)

2.1. Запоминающие устройства. Функциональный состав и временные диаграммы работы ОЗУ (1-2 стр.)

2.2. Устройство управления: а) Центральный узел распределения управляющих сигналов; б) Узел местного управления фазами выполнения команд; в) Блок указателя команд IP и блок регистра кода команд IR. г) Блок управления стековой памятью (3-4 стр.)

2.3. Арифметико-логическое устройство: а)Логическое построение АЛУ и блок РОНов; б) Реализация арифметических и логических операций (2-3 стр.)

2.4. Организация кэш-памяти процессора (2-3 стр.)

2.5. Описание системы предсказания переходов (1-2 стр.)
2.6. Аппаратура и функционирование КПДП (1-2 стр.)
2.7. Описание арбитра шины (1-2 стр.)

3. Функциональное моделирование (8-14 стр.)
3.1. Всех указанных выше блоков по отдельности (6-10 стр.)
3.2. Описание временной диаграммы работы всего устройства (2-4 стр.)

4. Анализ и оптимизация разработанной микро-ЭВМ (3-5 стр.)
4.1. Сокращение длительности выполнения отдельных команд (1-2 стр.)
4.2. Конвейерное выполнение фаз последовательностей команд (2-3 стр)
4.3. Реализация микропрограммного варианта устройства управления (2-3 стр.)

Заключение – общие технические характеристики разработанной микро-ЭВМ по архитектуре, быстродействию, топологическим свойствам, оптимальности конструкторских решений (1-2 стр).

Все функциональные блоки, которые не будут представлены на чертежах А3 должны быть отображены в пояснительной записке на отдельных листах формата А4.

6.1.2. Критерии оценки курсового проекта

Курсовой проект будет оцениваться по 10 бальной системе исходя из следующих критериев:

1. Наличие в тексте ПЗ более трёх предложений, являющихся плагиатом, – минус 1 балл итоговой оценки.
2. Отсутствие в ПЗ функционального блока или несоответствие блока, выданному варианту – минус 1 балл за каждый блок.
3. Отсутствие результатов общего моделирования всех блоков разрабатываемой ЭВМ в одном общем проекте – минус 1 балл.
4. Небрежное оформление ПЗ – т.е. наличие орфографических, грамматических, синтаксических, стилистических ошибок (более 5 шт. на ПЗ), небрежно отформатированный текст (отсутствие единого стиля), нарушение нумерации подзаголовков и рисунков, небрежно оформленный список литературы – минус 1 балл.
5. Небрежное оформление чертежей – рамки не по ГОСТам, отсутствие линий связи для шин и основных управляющих сигналов – минус 1 балл.
6. Логическая нестыковка разработанных модулей между собой (несоответствие по функционалу, сигналам) – минус 1 балл.
7. Отсутствие дампов памяти при моделировании работы ОЗУ, ПЗУ, кэш-памяти – минус 1 балл.
8. Плагиат блоков – минус 1 балл за блок.
9. Отсутствие в тексте ПЗ листинга программы включающей в себя все команды разработанной архитектуры микро-ЭВМ, приведённой в символьном и бинарном виде – **курсовой к защите не принимается**.

Дополнительные баллы будут начисляться за нижеследующее:

1. Курсовые проекты, сданные на проверку до 1 мая текущего года – плюс 1 балл.
2. Подробное описание устройства управления, формул логических сигналов – плюс 1 балл.
3. Грамотное обоснование выбора того или иного схемотехнического решения – плюс 1 балл.
4. Оптимизация работы всей системы в целом, с примерами моделирования до оптимизации и после неё – плюс 1 балл.
5. Красота схемотехнического решения любого сложного блока – плюс 1 балл.

Таблица 6. Варианты построения микро-ЭВМ на ПЛИС

№ п/п	Тип арх-ры	Разрядность шин		Память			РОН	КЭШ		
		Адр	данных	ПЗУ	ОЗУ	Тип Адресации*	шт.	k	Алгоритм Замещ-я строк	Синхр-я с памятью
1	Гарвард вард-ская	8	8	синхр	синхр	Косвенная	**4**	2	LRU	Сквозн. с ото-браж
2		12	16				**8**	4		
3		14	8			Прям рег	**10**	2	Наиб давн хранения	
4		16	16				12	4		
5	Прин-стон-ская	8	8	Асинхр	асинхр	косвенная	**14**	2	Без анализа	Сквозн. без ото-браж
6		12	16				**16**	4		
7		14	16			Прям рег	**4**	2	LRU	
8		16	8				8	4		
9	Гарвард вард-ская	8	8	Синхр	асинхр	Косв рег	**10**	2	Наиб давн хранения	про-стаяот-лож
10		12	8				**12**	4		
11		14	16			Базов смещ	**14**	2	Без анализа	
12		16	16				16	4		
13	Прин-стон-ская	8	16	Асинхр	синхр	Косв рег	**4**	2	LRU	Про-стая от-лож
14		12	16				**8**	4		
15		14	8			Базов смещ	**10**	2	Наиб давн хранения	
16		16	8				12	4		
17	Гарвард вард-ская	8	16	Асинхр	синхр	непосредств	**16**	4	Без анализа	Флагов отлож
18		12	8				**20**	2		
19		14	8			Прям Рег	**24**	4	LRU	
20		16	16				28	2		
21	Принстон-ская	8	16	Синх	асинхр	непосредтв	**16**	4	Наиб давн хранения	Сквозн. с отображ
22		12	8				**20**	2		
23		14	16			Прям рег	**24**	4	Без анализа	
24		16	8				28	2		
25	Гарвард вард-ская	8	8	Асинхр	Асинхр	Относ смещ	**10**	4	LRU	Флагов отлож
26		12	16				**12**	2		
27		14	8			Баз инд	**14**	4	Наиб давн хранения	
28		16	16				16	2		
29	Прин-стон-ская	8	8	Синхр	Синхр	Относ смещ	**4**	4	Без анализа	Простая отлож
30		12	16				**8**	2		
31		14	16			Баз инд	**10**	4	LRU	
32		16	8				12	2		

Таблица 7. Продолжение вариантов задания

№ п/п	**АЛУ***			Арбитраж шин	Стек Объем/ напр роста	Схема Предск. переходов		КПДП	
	Арифм. К-ды	Логич. К-ды	Сдвиговые К-ды				Бит, тип шаблона	**Нач.Адр./** Объем (Б)	
1	INCS	NOT, AND	SLL	Централ парал	4/вверх	A2	3, PC ⊕ GHR	**4**	**4**
2			SRL		5/вниз			6	8
3		OR, NOR	SLA		6/вверх	A4	4, PC(2) || GHR(2)	8	12
4			SRA		7/вниз			10	6
5	SUB	NAND,NOR	ROL	Централ послед	8/вверх	A3		4	10
6			ROR		9/вниз			6	8
7		NOTZ, AND	SLL		10/вверх	A5	4 GHR	8	12
8			SRL		11/вниз			10	6
9	CMP	XOR, NAND	SLA	Децентр парал	12/вверх	A2		4	6
10			SRA		4/вниз		5, PC	6	10
11		NXOR, OR	ROL		5/вверх	A4		8	4
12			ROR		6/вниз			10	8
13	ADDC	AND, OR	SLL	Децентр кольц	7/вверх	A3		4	6
14			SRL		8/вниз			6	10
15		NAND, NOTZ	SLA		9/вверх	A5	4, PC ⊕ GHR	8	4
16			SRA		10/вниз			10	8
17	ADDC	NOT, AND	ROL	Централ парал	11/вверх	A2		24	12
18			ROR		12/вниз			16	8
19		OR, NOR	SLL		4/вверх	A4	4, PC(2) || GHR(2)	8	20
20			SRL		5/вниз			10	24
21	CMP	NAND,NOR	SLA	Централ послед	6/вверх	A3		24	10
22			SRA		7/вниз			8	16
23		NOTZ, AND	ROL		8/вверх	A5	4, PC	12	8
24			ROR		9/вниз			8	10
25	SUB	XOR, NAND	SLL	Децентр парал	10/вверх	A2	3, PC ⊕ GHR	24	12
26			SRL		11/вниз			16	16
27		NXOR, OR	SLA		12/вверх	A4	4, PC(2) || GHR(2)	8	20
28			SRA		4/вниз			10	24
29	INCS	AND, OR	ROL	Децентр кольц	5/вверх	A3		4	6
30			ROR		6/вниз			6	10
31		NAND, NOTZ	SLL		7/вверх	A5	5 GHR	8	4
32			SRL		8/вниз			10	8

***Прим.** (для всех вариантов): Помимо указанных в таблице 7 команд для всех вариантов необходимо реализовать обязательные команды – MOV adr, reg;

MOV reg, adr; JMP adr; PUSH reg, POP reg, HLT, а так же в дополнение к указанным в варианте способам адресации **необходимо реализовать прямую адресацию памяти**.

При реализации АЛУ и УУ обязательно предусмотреть наличие регистра флагов.

6.1.3. Расшифровка команд в таблице

SLL (Shift Left Logical) – логический сдвиг влево,
SRL (Shift Right Logical) – логический сдвиг вправо,
SLA (Shift Left Arithm.) – арифметический сдвиг влево,
SRA(Shift Right Arithm.) – арифметический сдвиг вправо,
ROL(Rotate Left Log.) – циклический сдвиг влево,
ROL(Rotate Right Log.) – циклический сдвиг вправо,
INCS (инкремент по флагу S) – увеличение на значение флага S,
ADDC (сложение с учетом переноса) – сложение с учётом флага CF,
SUB - вычитание,
CMP (3 выхода: больше, меньше или равно) – сравнение двух операндов,
NOT – логическое отрицание,
AND – логическое умножение,
OR – логическое сложение,
NOR – логическая операция ИЛИ-НЕ,
NAND – логическая операция И-НЕ,
NOTZ – инверсия по флагу Z,
XOR – исключающее ИЛИ,
NXOR – исключающее ИЛИ-НЕ.

Список литературы

1. Столлингс, У. Структурная организация и архитектура компьютерных систем/ У. Столлингс. 5-е изд. – М.: "Вильямс", 2001. Пер. с англ. – 892 с.

2. Таненбаум, Э. Архитектура компьютерных систем/ Э. Таненбаум. 4-е изд. – М.: "ПИТЕР", 2002. Пер. с англ. – 698 с.

3. Цилькер, Б.Я. Организация ЭВМ и систем/ Б.Я. Цилькер, С.А. Орлов. – М.: "Питер", 200. – 668 с.

4. Yeh T.Y. and Patt Y.N. Alternative Implementations of Two-Level Adaptive Branch Prediction, The 19th Annual International Symposium on Computer Architecture Gold Coast, Australia, May 19-21, 1992, pp 124 -134.

5. Грушвицкий, Р. Проектирование систем на микросхемах программируемой логики/ Р. Грушвицкий. – СПб.: "Питер", 2002. – 608 с.

6. Угрюмов Е. Цифровая схемотехника. - М.: "С-Петербург", 2001, 518 стр.

7. Майоров С.А. Введение в микроЭВМ. - Л.: Машиностроение, 1988.

8. Соловьев В.В. Проектирование функциональных узлов цифровых систем на программируемых логических устройствах. - Мн.: "Бестпринт", 1996.

9. Солонина А. Алгоритмы и процессоры цифровой обработки сигналов. - СПб.: "Питер", 2001. – 464 с.

10. Гук М. Аппаратные интерфейсы. Энциклопедия. – СПб.: "Питер", 2002. – 528 с.

11. Шагурин И.И. Процессоры семейства Intel P6. Архитектура, программирование, интерфейс. – М.: "Телеком", 2000. – 248 с.

12. Рудометов Е. Материнские платы и чипсеты. – СПб.: "Питер", 2000. – 256 с.

13. Бибило П.Н. Синтез логических схем с использованием языка VHDL. М.: СОЛОН-Р, 2002.

14. Антонов А. П. Язык описания цифровых устройств AlteraHDL. - М. : РадиоСофт, 2001.

15. Глецевич И.И., Прытков В.А., Отвагин А.В. Методические указания по дипломному проектированию для студентов специальности 40 02 01 «Вычислительные машины, системы и сети». – Минск БГУИР, 2009, 99 с.

Св. план 57, поз.

Учебное издание

Самаль Дмитрий Иванович
Колонов Виталий Валентинович

“Структурная и функциональная организация ЭВМ”
Лабораторный практикум
для студентов специальности 1- 40 02 01

РедакторГ.С. Корбут
Корректор
Подписано в печать Формат 60х84 1/16.
Бумага офсетная. Печать ризографическая. Усл.печ.л.
Уч.-изд.л. 2,0 Тираж экз. Заказ
Издатель и полиграфическое исполнение:Учреждение образования «Белорусский государственный университетинформатики ирадиоэлектроники»
ЛИ № 02330.0494371 от 16.03.2009. ЛП № 02330.0494175 от 03.04.2009
220013, Минск, П. Бровки, 6.