**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| Введение | 4 |
| 1. Разработка общей структуры микро-ЭВМ | 8 |
| 2. Разработка основных устройств микро-ЭВМ | 13 |
| 3. Принципиальная схема устройства управления. | 22 |
| Заключение | 25 |
| Литература | 26 |
| Приложения. | 27 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ВВЕДЕНИЕ**

В наше время компьютерные технологии используются во всех сферах жизни. Электронную начинку можно встретить как в космической отрасли, так и в детских игрушках и даже в тостере. Конечно, разные устройства используются для разных нужд, в них используются различные микропроцессорные устройства.

Наиболее простые устройства оснащаются микроконтроллерами, вычислительное ядро которых обычно выполнено из одной микросхемы.

Более сложные системы основаны на контроллерах, вычислительные части которых зачастую делаются в виде отдельных модулей.

Многим устройствам необходимо обмениваться данными с удаленными (внешними) устройствами. Такие системы основывают на микрокомпьютерах — более мощных микропроцессорных системах со средствами сопряжения с внешними устройствами.

Самой мощной и универсальной микропроцессорной системой является компьютер.

Существует несколько типов архитектур микропроцессорных систем. В данном курсом проекте реализована архитектура с общей шиной для данных и команд (принстонская, фон-неймановская архитектура). Соответственно, в составе системы в этом случае присутствует одна общая память, как для данных, так и для команд (рис. 1).

  
**Рис. 1.** Архитектура с общей шиной данных и команд.

Архитектура с общей шиной проще, она не требует от процессора одновременного обслуживания двух или более шин. Наличие единой памяти данных и команд позволяет использовать системы для широкого ряда задач. Например, в некоторых случаях нужна большая и сложная программа, а данных в памяти надо хранить не слишком много. В других случаях, наоборот, программа требуется простая, но необходимы большие объемы хранимых данных.

С точки зрения архитектуры, система не различает данные и команды, а значит команды могут подгружаться из внешних источников как обычные данные, а затем выполняться на процессорных элементах.

**1. РАЗРАБОТКА ОБШЕЙ СТРУКТУРЫ МИКРО-ЭВМ.**

**1.1 ФУНКЦИОНАЛЬНАЯ СХЕМА МИКРО-ЭВМ.**

Сердцем любой микропроцессорной системы является микропроцессор или просто процессор. Микропроцессор — это блок, который производит всю обработку информации внутри микропроцессорной системы. Остальные узлы выполняют вспомогательные функции: хранение информации (данных, текущей команды, текущего состояния), приема и передачи данных по опредленным протоколам, синхронизация работы процессора. Процессор выполняет арифметические функции (сложение, умножение и т.д.), логические функции (сдвиг, сравнение, маскирование кодов и т.д.), временное хранение кодов (во внутренних регистрах), пересылку кодов между узлами микропроцессорной системы, контроль за состоянием системы и предотвращением коллизий.

Все команды, выполняемые процессором, образуют архитектуру системы команд процессора. Структура и объем системы команд процессора определяют его быстродействие, гибкость, удобство использования. Система команд может быть рассчитана на узкий круг решаемых задач (у специализированных процессоров) или на максимально широкий круг задач (у универсальных процессоров). В системах, рассчитанных на узкий круг исполняемых программ, на производительность сильно влияет расположение отдельных блоков системы. Так, в некоторых системах с интенсивной работой с устройствами памяти скорость доступа к данным будет играть важнейшую роль, нежели скорость выполнения арифметических действий с данными.

Архитектура данного курсового проекта построена для интенсивных вычислений с использованием АЛУ и стека, а также для взаимодействия с внешними устройствами. Время доступа к данных регистров и стека мало (2 такта), в сравнении со временем доступа к памяти (при «cache hit» 2 такта, при «cache miss» 5 тактов, без учета задержек на ожидания доступа к памяти) или к памяти внешнего устройства.

**1.2. РАЗРАБОТКА СИСТЕМЫ КОМАНД**

В ходе проектирования курсового проекта, основываясь на архитектурных особенностях заданного варианта, а именно: ширина шины данных, ширина шина адреса, количество регистров и количество команд, обязательных к исполнению, заданная архитектура, была разработана система команд из 14 команд, включающая в себя команды работы с устройствами памяти, арифметико-логическим модулем и управления потоком команд.

В данной архитектуре команд роль устройства управления передана трехэтапному конвейеру, на каждом этапе выполняется отдельная часть, присущая работе каждого этапа. В ходе проектирования системы команд для каждого этапа конвейера выяснил, что на одном этапе для нескольких команд алгоритм действий совпадает.

Все команды имеют одинаковый размер, равный разрядности шины данных, а именно 16 бит.

Команда 1. Синтаксис «Pop R». Является командой выдачи данных из стека, единственным входным параметром команды адрес регистра R, в который будет записан результат выполнения команды. На первом этапе конвейера команда и адрес регистра пересылается во второй блок, где выполняется извлечение данных из стека. На третьем этапе данные сохраняются в регистр.

Команда 2. Синтаксис «HLT». Команда остановки микропроцессорной системы. Обнуляет все блоки обработки, промежуточные данные во временных регистрах конвейера и останавливает счетчик тактов. На первых двух тактах конвейера команда пересылается в следующий блок, на третьем этапе подается сигнал об остановке всех операций на всех блоках микропроцессорной системы.

Команда 3. Синтаксис «Push P». Команда записи данных в стек, единственным входным параметром команды является адрес регистра R, из которого будут записаны данные. На первом этапе конвейера данные из регистра R поступают во второй блок, где ведется запись данных в стек. Преимущество этой команды в том, что она выполняется на двух блоках конвейера без обращений к блоку памяти, что позволяет третьему блоку работать с этим блоком без коллизий.

Команды 4-7. Синтаксис «Команда R1,R2,R3». Команды с использованием арифметико-логического блока. Имеют три входных параметра – адреса двух регистров – операндов(R1 и R2) и адрес регистра сохранения результата(R3). На первом этапе конвейера операнды достаются из регистров и передаются во второй блок, где происходит выполнение операции на блоке АЛУ. В третий блок передается адрес регистра сохранения результата, результат выполнения команды, а также флаг переполнения для команд сдвига и вычитания. Команда 4 выполняет операцию вычитания (R3 = R1-R2), команда 5 выполняет операцию побитного «ИЛИ» (R3 = R1|R2), команда 6 выполняет операцию «НЕ исключающее ИЛИ» (R3 = NOT(R1 NOR R2)), команда 7 выполняет операцию кругового смещения влево регистра R1 на количество разрядов, хранящихся в R2.

Команда 8. Синтаксис «Mov R,Short\_Addr». Команда копирования 16 бит из регистра в память. На первом этапе конвейера данные достаются из регистра R, далее команда, адрес в памяти и данные передаются третьему блоку через второй для записи данных в память.

Команда 9. Синтаксис «Mov Short\_Addr, R». Команда копирования 16 бит из памяти в регистр. На первом этапе конвейера данные достаются из памяти с адресом High\_addr + Short\_Addr, далее команда, адрес регистра и данные передаются третьему блоку через второй для записи данных в регистр. Старшая часть адреса High\_Addr хранится в служебном регистре в первом блоке конвейера и передается всем остальным блокам конвейера.

Команда 10. Синтаксис «JMP Short\_Addr». Команда безусловного перехода. Изменяет указатель следующей команды в памяти. Команда выполняется только на первом этапе конвейера.

Команда 11. Синтаксис «Set Short\_Addr». Команда установки старшей части адреса следующей команды. Команда выполняется только на первом этапе конвейера.

Команда 12. Синтаксис «JMS Short\_Addr». Команда условного перехода. Изменяет указатель следующей команды в памяти. Команда и адрес передаются в третий блок, также из второго блока передается значение регистра флагов. Если флаг переполнения поднят, то осуществляется переход, иначе действий не производится. При переходе подается сигнал на изменение указателя следующей команды, а также устанавливаются первых два этапа конвейера в исходное состояние.

Команды 13-14. Синтаксис «SetDMA Direction,DST\_SHORT\_Addr| SRC\_SHORT\_ADR». Установка конфигурации адресов контроллера прямого доступа к памяти. Входные параметры направление (Direction) и адреса передаются в третий блок. В третьем блоке подаются сигналы на конфигурирование контроллера прямого доступа к памяти.

Данные о всех командах для всех этапов конвейера и общая система команд сведены в таблицы(см. Приложение). Как видно из таблиц, не все команды проходят все стадии конвейера. Данный факт позволяет на уровне написания программ оптимизировать работу микропроцессора.

**1.3. ОПИСАНИЕ ВЗАИМОДЕЙСТВИЯ ВСЕХ БЛОКОВ МИКРО-ЭВМ В ХОДЕ ВЫПОЛНЕНИЯ КОМАНД ПРОГРАММЫ.**

Классическая ЭВМ состоит из трех основных устройств: арифметико-логического устройства, устройства управления и запоминающего устройства. В данном курсовом проекте роль устройства управления передана трехэтапному конвейеру, а так же реализовано несколько видов запоминающих устройств, отличающихся различными алгоритмами взаимодействия. Рассмотрим особенности взаимодействия этих устройств(см. структурную схему в приложении 1).

Основной идеей данной архитектуры – это перенос функций устройства управления на блоки конвейера. Все три блока конвейера не управляются извне. Блоки работают автономно и общаются друг с другом только двумя сигналами: сигнал готовности передать очередную порцию данных следующему блоку и сигнал готовности принять данные следующим блоком. Это означает что первый блок работает тогда, когда выходные данные отсутствуют. Второй блок работает тогда, когда первый блок готов отдать данные и если третий блок забрал готовые данные. Третий блок работает только тогда, когда второй блок сформировал для него выходные данные. Данный подход позволяет реализовывать команды, которые выполняются не на всех блоках без внесения аппаратных изменений.

Первоначально весь код программы записан в ПЗУ блока памяти. Размер ПЗУ составляет 8кб и имеет адресацию 000h-1FFFh, что позволяет записать 4096 команд. Возможно также использование данных из ОЗУ в качестве команд, предварительно загрузив команды из внешнего устройства посредством КПДП. Объем ОЗУ так же равен 8кб, адреса 2000h-3FFFh, что позволяет удвоить максимальный размер программы.

После старта микропроцессора, первый блок конвейера считывает первую команду с адресом 000h. Все данные, которые приходят из блока памяти попадают в кэш, для более быстрого повторного использования. На основе этой команды принимается решение о дальнейшей работе остальных блоков конвейера. Исходя из системы команд, не все команды требуют выполнения на всех трех этапах конвейера.

Команды могут использовать в качестве хранилища данных четыре типа устройств – ОЗУ, ОЗУ внешнего устройства, регистры общего назначения и стек. Многие команды используют либо стек, либо регистры общего назначения, так как задержка доступа к этим устройствам много меньше чем доступ к данным из ОЗУ при «промахе» на кэше. Для доступа к данным из внешнего устройства необходимо загрузить данные в ОЗУ в блоке памяти, из которого данные достаются по алгоритму доступа к блоку памяти. В конвейере два блока имеют доступ к шине блока памяти. Для разрешения конфликта доступа используется статический централизованный параллельный арбитр. Блок записи (третий) имеет высший приоритет, что позволяет разрешить конфликт чтение перед записью для команд, следующих через одну после команды записи результата в память.

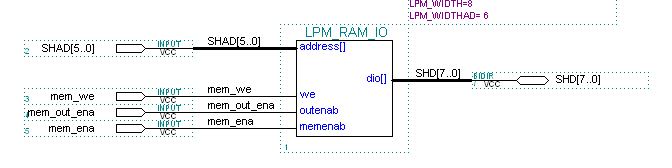
Блоки стека и АЛУ имеют выходы только к блоку выполнения команд конвейера. При такой организации нет конфликтов доступа к блокам, что ускоряет работу всего этапа конвейера.

**2. РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ.**

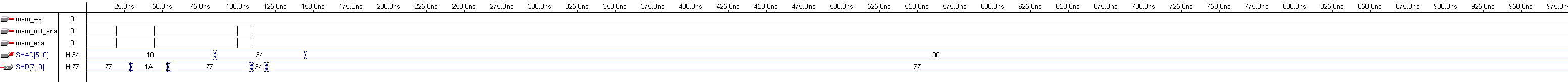
**2.1. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА. ФУНКЦИОНАЛЬНЫЙ СОСТАВ И ВРЕМЕННЫЕ ДИАГРАММЫ РАБОТЫ ОЗУ.**

а Разработчики фирмы Altera предоставляют несколько разновидностей запоминающих устройств для своих ПЛИС: ПЗУ - lpm\_rom\_io, ОЗУ- lpm\_ram\_dp, lpm\_ram\_dq и lpm\_ram\_io. lpm\_ram\_dp – это ОЗУ двойного доступа, позволяющее одновременно проводить операции чтения и записи в память. lpm\_ram\_dq – это ОЗУ с разделением шин данных для чтения и записи. lpm\_ram\_io – обычное ОЗУ с общей двунаправленной шиной для записи и чтения данных из памяти.

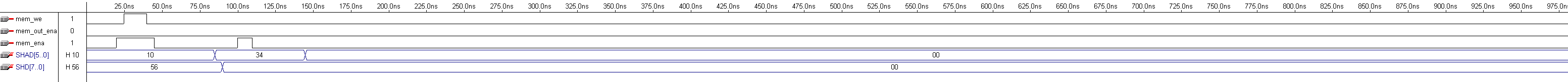
Наиболее просты в функционировании ЗУ - lpm\_rom\_io и lpm\_ram\_io. Рассмотрим работу асинхронных ПЗУ и ОЗУ этих типов. На рис. 2.1 изображена принципиальная схема ОЗУ lpm\_ram\_io, а на рис. 2.2 и 2.3 представлены временные диаграммы чтения и записи ОЗУ.



**Рис. 2.1.** Принципиальная схема ОЗУ.



**Рис. 2.2.** Временная диаграмма цикла чтения из ОЗУ



**Рис. 2.3.** Временная диаграмма цикла записи в ОЗУ.

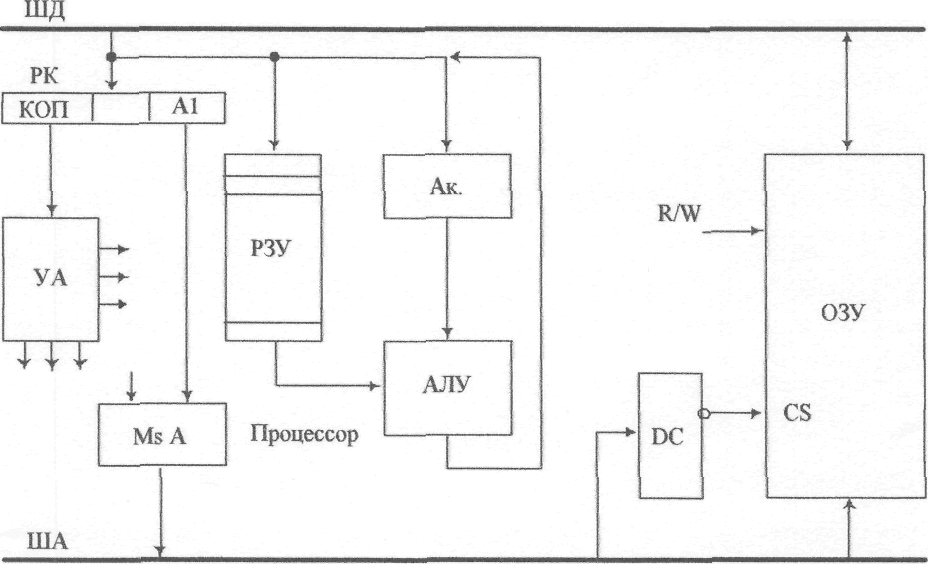
Как видно из рисунков 2.2 и 2.3 для ОЗУ требуется 3 управляющих сигнала: сигнал включения mem\_ena, сигнал выдачи данных на выход ОЗУ mem\_out\_ena и строб записи mem\_we.

**2.2. УСТРОЙСТВО УПРАВЛЕНИЯ.**

Устройство управления непрерывно выполняет циклическую последовательность действий: поочередное извлечение команд из памяти и их исполнение. В УУ наибольший интерес представляет регистр микрокоманды. Программа для микропроцессорной системы управления создается в виде последовательности команд. Каждая команда в общем случае содержит указание того, что должно быть сделано (код операции), и определение объекта действия, т.е. операнд. Способ задания операнда называется режимом адресации. Для микропроцессоров разработано около двух десятков режимов адресации, направленных на сокращение длины программы и времени ее выполнения. Наиболее простой, но и наиболее неэкономичной, является прямая адресация, при этом в поле операнда содержится полный адрес операнда в памяти. При использовании данного способа адресации второй операнд целесообразно адресовать косвенно или неявно. В противном случае хранение в структуре команды двух прямых

значений адресов приведёт к существенному увеличению длины формата, что в свою очередь ведёт к увеличению общего объёма оперативной памяти, используемой для хранения программ. Структурная схема, реализующая прямой способ адресации представлена на рис. 2.4.

При исполнении команд с данным способом адресации на первом этапе выполняется расшифровка поля КОП управляющим блоком. После расшифровки адрес А1 через мультиплексор адреса поступает на шину адреса. Дешифратор модуля ОЗУ активизирует на своём выходе соответствующую линию выборки. Далее операнд считывается в аккумулятор для последующей обработки в АЛУ.



**Рис. 2.4.** Структурная схема прямой адресации

Система команд.

Как было уже сказано, микропроцессор имеет индивидуальный набор команд. Но определенные закономерности присущи всем вычислительным системам, В зависимости от архитектуры конкретного микропроцессор построение слова, образующего команду, т.е. структура, которую принято называть форматом команды, может изменяться в широких пределах. Команды могут иметь фиксированную длину (например, одно машинное слово) или переменную длину (два, три и более слов); это определяется способом адресации к объекту действия, т.е. операнду. По формату набор команд можно разделить на две категории:

1. безадресные команды (без ссылки на данные). Команда содержит  
   только код операции и состоит из одного машинного слова.  
   Примерами являются команды ОСТАНОВ, ВОЗВРАТ и т.д.

2) команды с обращением к памяти (со ссылкой на данные).  
Команда должна содержать достаточно длинное слово, чтобы  
определить либо адрес данных в памяти, а в некоторых случаях  
промежуточный адрес, либо непосредственно сам операнд.

Диапазон адресации к ячейкам памяти и фактическая техника адресации, заложенная в архитектуре микропроцессора, определяют длину зоны операнда в команде. Для того чтобы можно было адресоваться к памяти емкостью 4К, требуется иметь зону адреса в 12бит, а для памяти емкостью 64К - 16бит. Соответственно приходится использовать команды, состоящие из двух, трех и более слов.

Классификация команд.

Все команды микропроцессора можно разделить на следующие основные типы или функциональные группы:

1. перемещения данных. Организуют передачу информации в  
   аккумулятор из памяти или, наоборот, из аккумулятора в память,  
   а также в аккумулятор из устройства ввода и из аккумулятора в  
   устройства вывода.
2. преобразования данных (арифметические и логические).  
   Используются для преобразования информации, т.е. выполнения  
   арифметических действий, например, СЛОЖИТЬ, ВЫЧЕСТЬ, и  
   логических операций, например, И, ИЛИ и т. д.
3. управления программой. Изменяют содержание программного  
   счетчика, являющегося адресом следующей команды.
4. ввода-вывода. Команды, организующие обмен информацией с  
   внешними устройствами системы.
5. специальные. Команды, которые непосредственно не выполняют  
   рассмотренные выше функции. Например, возможность  
   прерывания программы, запрет прерывания и т.д.

Рассмотрим формат команд для проектируемого ЭВМ:

Двухбайтная команда

1-ый байт:

|  |  |
| --- | --- |
| 7 4 | 3 0 |
| Код команды | № регистра |

2-ой байт

|  |
| --- |
| 7 0 |
| Адрес операнда |

В таблице 1 приведены коды команд.

Таблица 1

|  |  |
| --- | --- |
| Код команды | Команда |
| 0001 | MOV reg,adr |
| 0110 | MOV adr,reg |
| 1010 | JMP adr |
| 0101 | ADD reg,adr |
| 0100 | ADD adr,reg |
| 0010 | NOR reg,adr |
| 0011 | NOR adr,reg |
| 1100 | STI |
| 0101 | STI k |
| 1001 | CLI k |
| 1011 | HLT |

**2.3. АРИФМЕТИКО-ЛОГИЧЕСКОЕ УСТРОЙСТВО.**

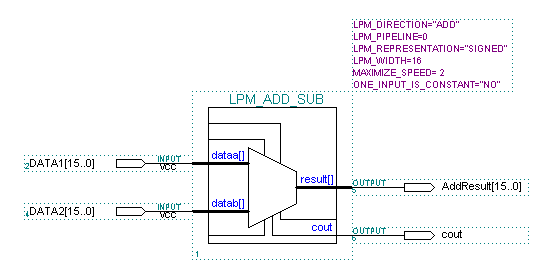
2.3.1Логическое построение АЛУ.

Этот блок микропроцессора представляет собой сложную двухвходовую комбинационную схему, которая сигналами от УУ настраивается на выполнение определенной операции. Свое название схема получила потому, что в ней комбинация сигналов на входе однозначно определяет сигнал на выходе.

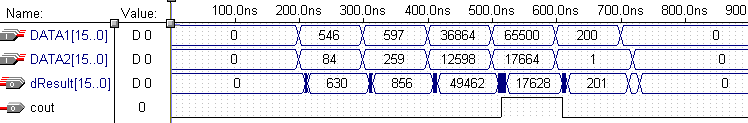
АЛУ представляет собой совокупность простейших электронных схем, способных хранить двоичные числа и выполнять над ними элементарные операции булевой алгебры. Данные, над которыми выполняются операции, поступают из регистров процессора, а результат также направляется в регистры. Эти регистры представляют собой элементы внутренней памяти процессора и связаны с АЛУ линиями передачи сигналов. В качестве вторичного результата АЛУ формирует значения набора двоичных флагов (признаков). Значения флагов также фиксируются в регистрах процессора. Блок управления передает в АЛУ сигналы, управляющие процессом выполнения операций, передачей операндов в АЛУ и считыванием результатов.

Среди стандартных элементов MAX+plus присутствует сумматор-вычитатель, который мы и будем использовать для реализации функции сложения.

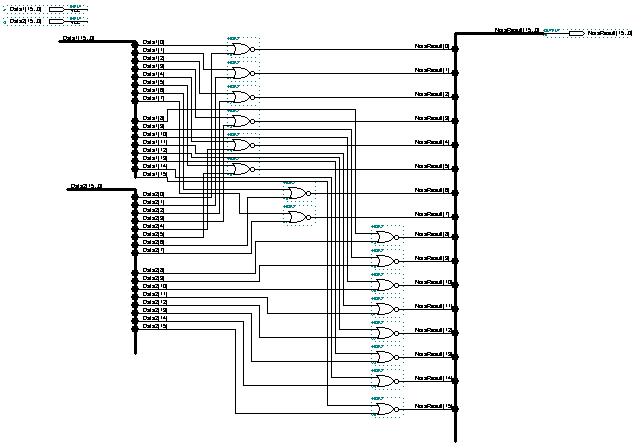
2.3.2.Реализация операции сложения.



Сигнал cout для формирования флага переполнения.



2.3.3. Реализация операции NOR



Блок РОНов вынесен за пределы АЛУ. Таким образом, АЛУ одним входом подключается к блоку РОНов, а другим к шине данных и через нее коммутируется с ОЗУ. Результат работы АЛУ может записываться как в блок РОНов, так и в ОЗУ.

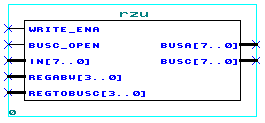
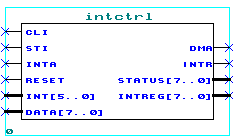


Схема блока РОНов представлена на рис. 2.6.

**Рис.2.6.** Схема блок РОНов.

**2.4. АППАРАТУРА И ФУНКЦИОНИРОВАНИЕ СИСТЕМЫ ПРЕРЫВАИЙ.**

Схема контроллера прерываний представлена на рис 2.9



**Рис.2.7.** Схема контроллера прерываний

Контроллер прерываний отвечает за взаимодействие процессора с устройствами ввода/вывода. Он включает в себя регистр масок, регистр статуса, регистр запроса прерываний. Если на вход контроллера пришел сигнал запроса на прерывание от устройства, контроллер прерываний проверяет, замаскировано ли оно. Если нет, то устанавливает сигнал INTR и ждет ответа от процессора.

Процессор может запретить некоторые или все прерывания для этого надо послать контроллеру соответствующую команду.

По стробу Reset во внутренний восьмиразрядный регистр маски IMR записывается число 0xFF (все прерывания разрешены). Шина INT[5..0] обеспечивает приём до 6 источников прерываний одновременно (приём рассчитан на импульсные сигналы). Запросы, пришедшие с этого входа, маскируются регистром IMR и сохраняются в регистре запросов IRR. Номер старшего прерывания дополнительно сохраняется в регистре статуса ISR. Тем самым реализуется приоритетность прерываний – прерывания со старшими номерами будут обслуживаться первыми. Прерывание INT4 интерпретируется как запрос от устройства, желающего использовать режим ПДП. Маскированный сигнал INT4 преобразуется в сигнал DMARequest. Прерывания INT0-INT5 интерпретируются как аппаратные. Если одно и более из этих прерываний демаскировано, то контроллер прерываний формирует на выходе INTR логическую единицу.

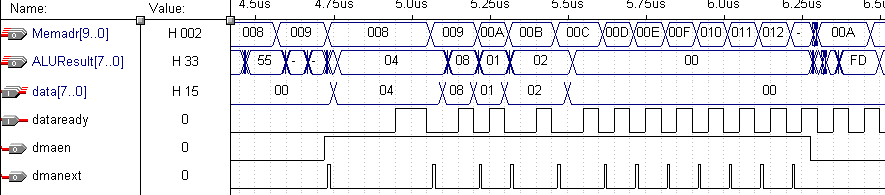
По единичному уровню сигнала INTA в регистре запросов IRR сбрасывается бит наиболее приоритетного прерывания, так как это прерывание считается взятым на обслуживание. Согласно принципу радиальной системы прерываний адреса обработчиков фиксированы и не могут быть изменены. Адрес вычисляется по следующему алгоритму: значение регистра ISR умножается на 2 (схемотехнически сдвигается на 1 бит).

Если в очереди больше нет запросов на обслуживание (IRR=0), то на выводе INTR формируется логический ноль.

По стробу на одном из входов cli или sti, контроллер прерываний выбирает байт с шины data[7..0] и интерпретирует его как номер прерывания, которое надо маскировать/демаскировать. Через внутреннюю схему управления irmodify производится сброс или установка соответствующего бита в регистре маски IMR.

**2.5. ПРЯМОЙ ДОСТУП К ПАМЯТИ. АППАРАТУРА И ФУНКЦИОНИРОВАНИЕ КОНТРОЛЛЕРА ПДП (КПДП).**

Контроллер ПДП служит для того, чтобы обеспечить взаимодействие внешнего устройства с ОЗУ без участия процессора. Для этого КПДП получив от контроллера прерываний сигнал, КПДП подключается к шинам и начинает вырабатывать управляющие сигналы и данные для ОЗУ.



**Рис.2.8.** Временная диаграмма работы КПДП.

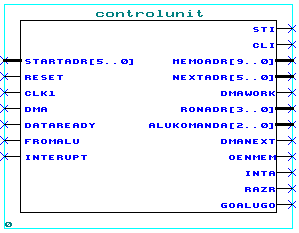
**2.6. МИКРО-ЭВМ. АППАРАТУРА И ФУНКЦИОНИРОВАНИЕ МИКРО-ЭВМ.**

Принципиальная схема микро-ЭВМ представлена в приложении. В данном устройстве была реализована шинная топология. Все блоки ЭВМ обращаются друг с другом посредством шин. Если одно устройство занимает шину, то другие отключены от нее и ждут необходимых управляющих сигналов. Рассмотрим работу микро-ЭВМ в целом. После прихода строба RESET, который поступает на устройство управления (УУ) процессор запускается. Внутри УУ блок выборки команды (УВК) инициализирует системный счетчик значением с шины StartAdr – с этого адреса начинается любая программа. УВК 2 байта из ОЗУ (значение программного счетчика каждый раз наращивается). Затем УВК дает сигнал о том, что команда считана и запускается счетчик выполнения команды. В течение не более 2-х тактов при подаче соответствующего адреса на ПЗУ выдается группа управляющих сигналов. После выполнения команды подается сигнал УВК о том, что надо выбирать следующую команду. И все повторяется заново. Если выполняемая команда арифметическо-логическая или пересылка данных, то сигналы подаются на АЛУ и ОЗУ. При выполнении операций маскирования/демаскирования УУ подает сигналы на контроллер прерываний. Это сигнал о приходе новой команды, код команды для контроллера КПДП, номер прерывания, которое надо замаскировать/демаскировать. Также на контроллер прерывания приходит сигнал разрешения прерывания. КПДП запускается по сигналу DMA. Он сразу выставляет запрос на прерывание и ждет разрешения записи в ОЗУ. После прихода разрешения прерывания КПДП выставляет строб записи в ОЗУ и каждый такт выдает на шины адреса и данных соответственно адрес, куда надо записать, и записываемое значение.

Временная диаграмма работы микро-ЭВМ представлена в приложении Г.

**3. ПРИНЦИПИАЛЬНАЯ СХЕМА УСТРОЙСТВА УПРАВЛЕНИЯ.**

Устройство управления предназначено для считывания очередной команды из ОЗУ. В соответствии с этой командой устройство управления вырабатывает управляющие сигналы для остальных блоков.



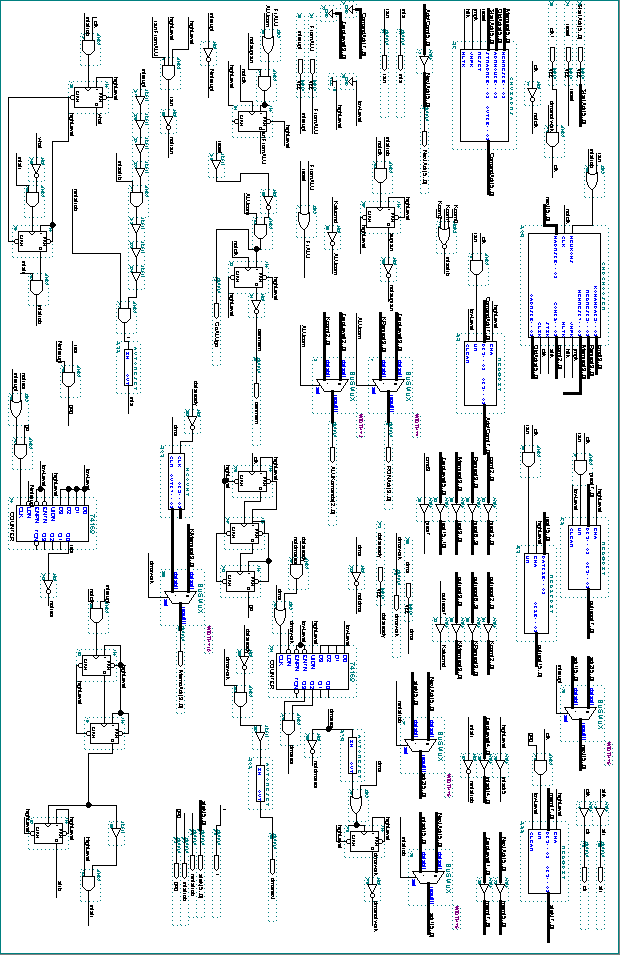
**Рис.3.1.** Схема УУ.

В таблице 2 представлены сигналы устройства управления.

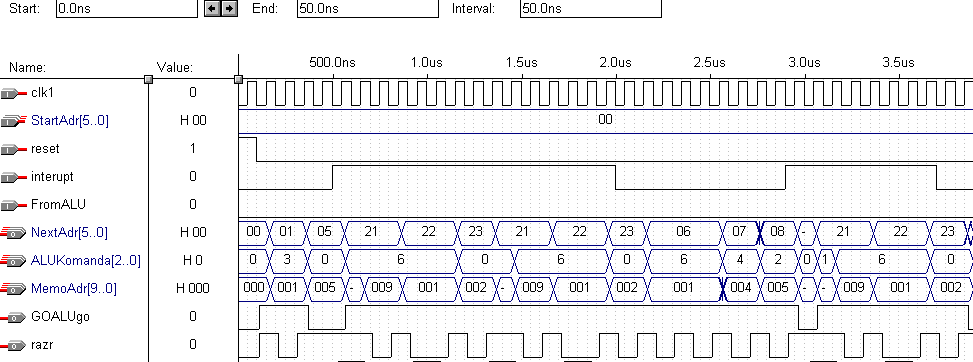
Таблица 2

|  |  |
| --- | --- |
| Входные сигналы | |
| CLK | Сигнал тактирования |
| STARTADR[5..0] | Данные с ШД |
| RESET | Запуск ЦП |
| FROMALU | Подтверждение от АЛУ |
| DMA | Запрос ПДП |
| DATAREADY | Готовность данных |
| INTRUPT | Запрос на прерывание |
| Выходные сигналы | |
| MEMOADR[9..0] | Шина адреса |
| OENMEM | Сигнал в ОЗУ |
| GOALUGO | Сигнал в АЛУ |
| RONADR[3..0] | Номер РОНа участвующего в арифметич. операции |
| ALUKOMANDA[2..0] | Команда АЛУ |
| NEXTADR[5..0] | Адрес след. команды |
| CLI | Команда CLI |
| STI | Команда STI |
| INTA | Завершение прерывания |
| DMANEXT | Запрос данных |

# Принципиальная схема УУ



Временная диаграмма работы АЛУ представлена на рисунке:



**Рис.3.2.** Временная диаграмма работы АЛУ.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта была разработана схема микро-ЭВМ с заданными характеристиками на элементной базе ПЛИС FLEX 10KE. Моделирование микро-ЭВМ происходило в среде автоматизированного проектирования Max Plus II фирмы Altera. Спроектированы основные узлы микро-ЭВМ и проработано моделирование результатов их работы на временном симуляторе.

Полученное устройство построено на принципах принстонской архитектуры (раздельная память команд и данных), устройство управления ЭВМ воспринимает команды нефиксированной длины (на выполнение одной команды уходит от 3 до 4 тактов), устройство обладает ориентировочной тактовой частотой 50MHz.

**Литература**

1. Применение микропроцессоров для автоматизации технологических  
   процессов. Вершин О.Е. -Л: Энергоиздат-Линенград, 1986.
2. ПЛИС фирмы "ALTERA": элементная база, система проектирования и  
   языки описания аппаратуры. Стешенко В.Б. -М.: Издательский дом  
   "Додэка-ХХГ, 2002.
3. Структурная организация и архитектура компьютерных систем.  
   Проектирование и производство. У.Столлингс. - М.: Издательский дом  
   "Вильяме", 2002
4. Кобяк И П. Организация ввода-вывода в компьютерных системах:

Методическое пособие по курсам "Теория и проектирование ЭВМ"' и "Структурная и функциональная организация ЭВМ" для студентов специальности 22.01 "Вычислительные машины, системы и сети" факультета здочного обучения. -Мн.гБГУИР., 1996.

5. КобякИ П. Архитектура компьютерных систем ч1и ч2:

Методическое пособие по курсам "Теория и проектирование ЭВМ1" и "Структурная и функциональная организация ЭВМ" для студентов специальности 22.01 "Вычислительные машины, системы и сети" факультета здочного обучения. -Мн.:БГУИР, 2001.

6. КобякИ П. Организация памяти компьютерных систем:

Методическое пособие по курсам "Теория и проектирование ЭВМ1" и "Структурная и функциональная организация ЭВМ" для студентов специальности 22.01 "Вычислительные машины, системы и сети" факультета здочного обучения. -Мн.:БГУИР, 2000.