**ВВЕДЕНИЕ**

Целью курсового проекта является разработка микро-ЭВМ на ПЛИС согласно заданному варианту.

Разрабатываемая ЭВМ должна быть реализована на гарвардской архитектуре, в которой данные и команды хранятся в разных адресных пространствах. Для хранения команд должно использоваться синхронное ПЗУ, для хранения данных – синхронное ОЗУ. Разрядность шины адреса равна 8 битам, разрядность шины данных – 8 бит, что позволяет адресовать объем памяти 256 байт. Для команд АЛУ будет использоваться прямая и прямая регистровая адресация, для команд условного и безусловного перехода – прямая, для команды MOV – прямая, относительная со смещением.

В дополнении к ОЗУ и ПЗУ проектируемое устройство также должно содержать блок регистров общего назначения, состоящий из 4 регистров, и кэш с множественно-ассоциативным отображением (k=4), алгоритм замещения строк – наиболее давнего хранения. Кэш использует простую отложенную синхронизацию с памятью, то есть данные в памяти обновляются при замещении строки в кэше. Также в системе предусмотрен стек размерностью 5 слов и направлением роста вверх. Стек размещается в ОЗУ.

Арифметико-логическое устройство данной ЭВМ должно содержать команду сравнения операнд (CMP), команды логических операций «И» (AND), «ИЛИ» (OR), команду циклического сдвига влево (ROL). В систему команд должны входить команды записи в стек (PUSH) и считывания из него (POP), команды пересылки данных между регистрами и памятью (MOV), команда безусловного перехода (JMP), условного перехода (JMZ), а также команда остановки работы ЭВМ (HLT). При реализации АЛУ обязательно нужно предусмотреть наличие регистра флагов.

Для обмена данными без участия процессора схема должна содержать контроллер прямого доступа к памяти. Для предотвращения конфликтов доступа нескольких устройств к одной шине будет использоваться арбитраж. Для работы конвейера в разрабатываемой микро-ЭВМ будет реализован предсказатель условных переходов.

При разработке схемы и симуляции ее работы будет использоваться САПР Altera Quartus II 9.1.

1. РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ МИКРО-ЭВМ

В данном разделе описывается функциональный состав микро-ЭВМ, разработка и описание системы команд, описание взаимодействия блоков при выполнении команд программы.

* 1. Функциональный состав микро-ЭВМ

В разрабатываемой ЭВМ используются следующие функциональные блоки:

* блок ПЗУ
* блок ОЗУ
* блок центрального процессора, который включает в себя:
* устройство управления
* блок кэша памяти
* блок регистров общего назначения
* блок стека
* блок арифметическо-логического устройства
* блок предсказателя переходов
* блок КПДП

Блок ПЗУ предназначен для хранения команд микропрограммы. Шина адреса 8 бит, поэтому размер ПЗУ может хранить 28 = 256 8-битных слов. Шина данных 8 бит, поэтому объем ПЗУ равен 256 байт. Режим работы – синхронный.

Блок ОЗУ предназначен для хранения данных. Блок ОЗУ и периферийное устройство будут иметь одно адресное пространство. Ширина шины адреса позволяет адресовать 256 слов. ОЗУ и периферийное устройство будут адресовать по 128 слов и иметь объем 128 байт. Начало адресного пространство будет принадлежать ОЗУ. Режим работы – синхронный.

Адресное пространство выглядит следующим образом:

|  |  |
| --- | --- |
| Диапазон адресов: | Предназначение: |
| 0x00 – 0x7F | RAM |
| 0x80 – 0xFF | Периферийное устройство |

Устройство управления содержит в себе логику, позволяющую выполнять считывание команд, их дешифрацию, загрузку операнд, а также подавать управляющие сигналы на другие блоки. Данный блок имеет регистры Instruction Pointer (IP) – начальный адрес текущей команды, Instruction Register (IR) – текущий адрес для считывания части инструкции для выполнения, Operand1 Register (Op1R) – первый операнд, Operand2 Register (Op2R) – второй операнд. Так же данный блок содержит флаги для сохранения результатов команды CMP: ZF - устанавливается, если операнды не равны, иначе сбрасывается, AgB – первый операнд больше второго, AlB - первый операнд меньше второго. Устройство управление подключено к блокам РОН, КЭШ, АЛУ, предсказателя переходов, стека. Устройство управления имеет доступ ко всем системным шинам.

Обмен данных между центральным процессором и памятью происходит через блок кэша памяти. В нем содержится 4 блока, предназначенные для хранения недавно использовавшихся данных, а также логика управления ими. Кэш с множественно-ассоциативным отображением (k=4): имеет 4 банка памяти по 4 строки, в каждой строке по 4 слова. Таким образом, кэш имеет объем 32 слова. Кэш используется при обращении к памяти ОЗУ и периферийного устройства. Взаимодействие кэша с процессором происходит через сигналы чтения или записи данных, вход для адреса данных, вход для записи данных в память, выход для чтения данных, выходной сигнал о наличии данных. С памятью кэш взаимодействует, используя выходы, сигнализирующие о необходимость записать или считать данные из памяти, шину адреса, двунаправленную шину данных.

Блок регистров содержит 4 8-битных (по ширине шины данных) регистров для хранения данных. Данный блок предоставляет 2 операции: запись и чтение. Данные операции выполняются за 1 такт. Блок регистров имеет следующие входы: для тактирующего сигнала, для команд записи и чтения, для указания регистра, для входных данных. Также имеется выход для выходных данных.

Блок КПДП предназначен для обмена данными между ОЗУ и периферийным устройством без использования ЦП. КПДП имеет 3 служебных 8-битных регистров:

1. адрес начала данных в источнике;
2. адрес начала данных в приемнике;
3. количество слов для обмена.

Блок стека предназначен для хранения, изменения, выдачи адреса вершины стека. Сами данные хранятся в ОЗУ.При включении микро-ЭВМ происходит инициализация регистра адреса вершины стека. Стек растет вверх, поэтому начальное значение вершины стека равно последнему адресу ОЗУ – 0x7F. При помещении данных в стек значение регистра уменьшается на 1, при чтении – увеличивается на 1.

Для начала обмена данными через КПДП необходимо, чтобы служебные регистры были проинициализированы, количество слов для обмена было больше 0, а также системные шины были не заняты ЦП. Данный блок имеет доступ к шинам адреса, данных, управления, а также вход для тактирующего сигнала.

Децентрализованный арбитраж выполняет разделение времени использования общих шин блоком центрального процессора и КПДП для избегания конфликтных ситуаций. Приоритет ЦП выше, чем у КПДП.

* 1. Разработка системы команд

Размерности шины данных и адресной шины равны 8 битам, поэтому размер операндов равен 8 битам. Для кода инструкции также выделяется 8 бит. Итого размер команды равен 24 бита, поэтому ее считывание будет происходить за 3 такта.

Таким образом, система команд имеет следующий вид (Таблица 2):

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| Мнемоническая запись | Биты [7..0]  (код команды) | Биты [15..8]  (операнд 1) | Биты [23..16]  (операнд 2) |
| MOV1 reg, addr1 | 00000001 | reg | addr1 |
| MOV2 addr1, reg | 00000010 | addr1 | reg |
| MOV3 reg, addr2 | 00000011 | reg | addr2 |
| MOV4 addr2, reg | 00000100 | addr2 | reg |
| JMP addr1 | 00000101 | addr1 |  |
| JMZ addr1 | 00000110 | addr1 |  |
| CMP1 reg, addr1 | 00000111 | reg | addr1 |
| CMP2 reg, addr3 | 00001000 | reg | addr3 |
| AND1 reg, addr1 | 00001001 | reg | addr1 |
| AND2 reg, addr3 | 00001010 | reg | addr3 |
| OR1 reg, addr1 | 00001011 | reg | addr1 |
| OR2 reg, addr3 | 00001100 | reg | addr3 |
| ROL1 reg, addr1 | 00001101 | reg | addr1 |
| ROL2 reg, addr3 | 00001110 | reg | addr3 |
| PUSH reg | 00001111 | reg |  |
| POP reg | 00010000 | reg |  |
| HLT | 00010100 |  |  |

В данной таблице используются следующие обозначения: reg – номер регистра (AX – 0001, BX – 0010, CX – 0100, DX - 1000); addr1 – адрес операнда (прямая адресация), addr2 – адрес операнда (относительная со смещением), addr3 – адрес операнда (прямая регистровая).

При использовании относительной адресации со смещением за базовый адрес принимается значение, хранящееся в регистре BX.

**1.3** Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы

ЭВМ начинает работу при подаче тактирующего сигнала. На первом такте Устройство управления (УУ) выдает сигнал блоку стека для инициализации регистра Stack Pointer (SP).

УУ посылает сигнал на чтение команды из ПЗУ, выставляя значения из PC (начальное значение 0x00) на шину адреса команд. Считывает и сохраняет в регистры IR, Op1R, Op2R информацию о команде, декодирует инструкцию. Далее УУ подготавливает адреса, если используется относительная адресация со смещением, то происходит обращение к РОН для получения базового адреса, который хранится в регистре BX. По полученным адресам УУ считывает значения операндов, используя РОН или кэш. При отсутствии данных в кэше, тот обращается к ОЗУ или периферийному устройству, чтобы получить значения операндов: выставляет значения на шины адреса и управления, получает данные по шине данных.

Если выполняется команда условного перехода, УУ подает тактирующий сигнал, значение PC, значение флага Z на блок предсказателя переходов, который выдает результат предсказания.

Если выполняется арифметическая операция, то УУ подает значения операндов (после их получения) на вход АЛУ, который выдает результат и выставляет флаги сразу, либо, если выполняется команда ROL, выдает результат вместе с сигналом readyRol. Далее УУ записывает результат работы АЛУ в РОН.

Если выполняется команда MOV, УУ записывает данные из кэша в регистр общего назначения, либо в обратном направлении, в зависимости от типа команды MOV.

При выполнении команды POP, УУ подает данные из указанного регистра на шину данных, а также сигнал о записи блоку стека. Стек подает на шину адреса значение регистра SP, а также устанавливает управляющий сигнал о необходимости записать данные в ОЗУ.

При выполнении команды PUSH, УУ выдает сигнал блоку стека о необходимости считать данные. Блок стека уменьшает значение регистра SP, подает его на шину адреса, выдает сигнал ОЗУ о необходимости чтении данных. УУ записывает значение с шины данных в указанный регистр.

КПДП при наличии разрешения доступа к системным шинам и необходимости осуществить передачу данных, выставляет адрес и устанавливает управляющий сигнал о необходимости чтения для устройства-источника, сохраняет значения с шины данных себе в буфер. Далее выставляет адрес, значение буфера на шину данных и устанавливает управляющий сигнал о необходимости записи для устройства-приемника.

1. РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ ЭВМ
   1. Запоминающие устройства. Функциональный состав и временные диаграммы работы

В разрабатываемой микро-ЭВМ используются синхронные запоминающие устройства: ПЗУ (рис. 1) и ОЗУ (рис. 2). Они реализованы на основе стандартных компонентов Quartus II: ПЗУ – LPM\_ROM, ОЗУ – LPM\_RAM\_IO.

Для осуществления операции чтения из ПЗУ необходим 1 такт, выдача данных производится по фронту тактирующего сигнала.

Для ОЗУ при чтении или записи данных необходимо 2 такта. На первом такте указывается адрес, на втором завершается выполнение операции.



Рис.1 Условно-графическое обозначение блока ПЗУ

Блок ПЗУ (рис. 1) имеет три входных сигнала:

* *ADDRESS*  – адрес ячейки памяти для считывания;
* *clk* – тактирующий сигнал;
* *read* – запрос на чтение.

Выходной сигнал:

* *DATA* – прочитанные из ПЗУ данные



Рис.2 Условно-графическое обозначение блока ОЗУ

Блок ОЗУ (рис. 2) имеет четыре входных сигнала:

* *ADDRESS*  – адрес ячейки памяти;
* *clk* – тактирующий сигнал;
* *write* – запрос на запись;
* *read* – запрос на чтение.

Двунаправленный выход:

* *DATA* – данные, для чтения или записи.

**2.2** Блок регистров общего назначения

Регистры общего назначения (рис. 4) предназначены для временного хранения данных. Также регистр BX используется как базовый адрес при использовании относительной адресации со смещением.

Чтение данных осуществляется при подаче тактирующего сигнала и указания нужного регистра.

Запись данных с входа осуществляется по фронту тактирующего сигнала при поступлении сигнала о записи и указания номера регистра.



Рис.4 Условно-графическое обозначение блока регистров.

Блок регистров имеет следующие входы:

* *WRITE* – входные данные для записи;
* *REG\_NUM* – значение на запись в регистр;
* *CLK* – тактирующий сигнал.

Выходы:

* *AX* – данные в регистре AX;
* *BX* – данные в регистре BX.

Двунаправленный выход:

* *DATA* – данные, для чтения или записи.

**2.3** Блок стека

Стек (рис. 5) используется для временного хранения данных. Данный элемент работает по принципу LIFO (Last In, First Out). В разрабатываемой микро-ЭВМ данные, помещаемые в стек, располагаются в последних адресах памяти ОЗУ.

В блоке стека размещен регистр Stack Pointer (SP), который указывает на вершину стека – на адрес ячейки в ОЗУ, по которому будет производиться запись. В начале работы микро-ЭВМ происходит инициализация SP значением 0x7F – адрес последний ячейки памяти ОЗУ.

Блок стека должен выдавать адрес на шину адреса, по которому будет производиться чтение/запись. Также при данных операциях значение SP должно изменяться, указывая на первую пустую ячейку памяти.

Направление роста стека «вверх». Поэтому при записи в стек, адрес, хранящийся в SP, должен быть декрементирован, а при чтении – инкрементирован.

При выполнении команды POP производится запись данных по адресу SP: на шину адреса выставляется значение вершины стека, затем оно уменьшается на 1 и сохраняется.

При выполнении команды PUSH cначала производится уменьшение значения SP на 1, сохранение его, затем оно выставляется для чтения на шину адреса.



Рис. 5 Условно-графическое обозначение стека.

Блок стека имеет следующие входы:

* *INIT* – команда для инициализации регистра SP;
* *PUSH* – команда для записи в стек;
* *POP* –команда для чтения из стека;
* *CLK* – тактирующий сигнал.

Блок стека имеет один выход:

* ADDRESS – адрес для обращения в ОЗУ.

**2.4** КПДП

Контроллер прямого доступа к памяти (рис. 6) предназначен для выполнения обмена данными между некоторыми внешними устройствами без участия центрального процессора. В разрабатываемой микро-ЭВМ обмен может производиться между ОЗУ и периферийным устройством.

Блок КПДП имеет 3 8-битных служебных регистра. Первый предназначен для хранения начального адреса данных для обмена на устройстве-источнике, второй – начального адреса на устройстве-приемнике, куда будут сохраняться передаваемые данные. Третий предназначен для хранения количества передаваемых слов. Также имеется буферный регистр. Он используется для сохранения прочитанного слова, так как для осуществления обмена необходимо выставлять на шину адреса источника и приемника, что одновременно сделать нельзя.

КПДП начинает работать, если в регистре количества слов для передачи данных находится отличное от нуля значение, а также, если ЦП не использует системные шины в текущий момент.

В процессе пересылки данных, для изменения адресов и оставшегося количества слов для пересылки используются счетчики.



Рис.6 Условно-графическое обозначение КПДП.

Блок КПДП содержит один вход от арбитра, по которому идёт сигнал разрешения на занятие системных шин. Так же есть один выход, по которому блок КПДП сигнализирует о намерении занят системные шины. Ещё в данном блоке есть 3 двунаправленных входа:

* *data* – подключение к шине данных;
* *address* – подключение к шине адреса;
* *control* – подключение к шине управления;
* *Grant* – сигнал разрешения на работу с внешними шинами.

**2.5** Система арбитража шин

Арбитраж необходим для предотвращения конфликтов при одновременном обращении разных устройств к системным шинам (адреса, данных, управления). В разрабатываемой микро-ЭВМ данными устройствами являются ЦП и КПДП. ЦП имеет высший приоритет, поэтому когда ЦП необходимо использовать системные шины, КПДП приостанавливает свою работу.

**2.6** Организация кэш-памяти процессора

Кэш-память (рис. 7) используется в качестве буфера при обращении ЦП к ОЗУ. Применяется для ускорения процесса работы с памятью. Данное ускорение достигается за счет того, что кэш-память является более быстрой, по сравнению с ОЗУ. Также сокращается время, которое ЦП будет занимать системные шины.

В разрабатываемой микро-ЭВМ используется множественно-ассоциативный кэш (k=4). Это значит, что он будет иметь 4 банка памяти. Банк памяти адресуют 2 старшие бита адреса. Каждый банк имеет 4 строки, которые состоят из 4 слов. Таким образом, объем кэша равен 64 байтам, то есть составляет 25% от памяти ОЗУ и периферийного устройства (по 128 байт). Строка в банке определяется по тэгу, который состоит из 5-2 бит адреса. Слово в строке определяется по двум младшим битам адреса.

Алгоритм замещения строк определяет, каким образом будет происходить замещение строки в кэше, при отсутствии в нем данных по указанному адресу. В данном курсовом проекте используется алгоритм «наиболее давнего хранения». Для его реализации используются счетчики размерностью 2, для того, чтобы бы возможно было выбирать одну из четырех строк. При замещении строки происходит увеличение значения счетчика. Строка по порядковому номеру равному значению счетчика будет замещать при следующем промахе при обращении к кэшу.

Метод синхронизации с памятью определяет, каким образом значения из кэша будут обновлять значения в памяти. В разрабатываемой микро-ЭВМ используется простая отложенная память. При использовании данного типа синхронизации, обновление информации в памяти происходит лишь только при замещении строки кэша. Это позволяет снизить обращение к памяти, так как нет необходимости обновлять данные в памяти ОЗУ или периферийного устройства каждый раз, как они изменяются в кэше.

При попадании (обращении к данным, которые содержатся в кэше) чтение или запись данных совершается на один такт.

В случае промаха, кэш определяет строку для замещения, последовательно выдает адреса данных в строке, которые будут записаны в память и их значения, далее последовательно выдает адреса новых данных, которыми будет замещена старая строка, и сохраняет их.

При первом обращении к кэшу всегда будет происходить ситуация с промахом, так как кэш будет пуст. Однако нельзя допустить замещения строки, необходимо только сохранить данные из памяти и не записывать в нее старые значения строки. Для этого используются триггеры, которые хранят в себе информацию о том, были ли уже записаны данные, или же строка не была инициализирована.

Для удобства работы блок имеет выход ready, который символизирует о готовности данных для ЦП.



Рис. 7 Условно-графическое обозначение блока кэш.

Блок кэша содержит следующие входы:

* *CLK* – тактирующий импульс;
* *ADDRESS* – адрес требуемых данных для ЦП;
* *dataFromCPU* – данные поступающие от ЦП;
* *write* – сигнал для записи данных;
* *read* – сигнал для чтения данных.

Блок кэша содержит следующие выходы:

* *adrToRAM* – адрес для обращения к памяти;
* *dataToRAM* – для обмена данными между памятью и кэшем;
* *writeRAM* – запрос на запись данных в память;
* *readRAM* – запрос на чтение их памяти;
* *dataToCPU –* данные для процессора;
* *READY* – сигнал, говорящий о готовности данных для процессора.

**2.7** Устройство управления

Устройство управления (рис.8) является "сердцем" нашей ЭВМ. Устройство управления можно разделить условно на следующий под-блоки (в данной работе они все находятся в одном блоке, но их можно бы было разнести по разным блокам):

1. Блок работы с кэш-памятью
2. Блок работы с РОНами
3. Блок работы с АЛУ
4. Блок хранения и декодирования комманды
5. Блок формирования управляющих импульсов
6. Блок исполнения комманд

Опишем данные блоки поподробнее:

*Блок работы с кэш-памятью* ответственнен за выставление в нужный момент времени правильных сигналов на шину данных/шину управления/шину адреса, а так же за сохранение результата из кэша при его котовности в нужном регистре.

*Блок работы с РОНами* по характеристикам аналогичен предыдущему блоку.

*Блок работы с АЛУ* выставляет правильную коммнаду на АЛУ, подаёт правильные данные, а так же сохраняет результат выполнения комманды в нужном регистре при работе устройства управления с блоком АЛУ.

*Блок хранения и декодирования комманды* содержит 3 регистра для хранения всей комманды, а так же устройство для дешифрации формата команды. Так же данный блок в случае необюходимости подаёт комманду для загрузки дополнительных данных, если они нужны для текущией комманды (например, загрузка 3-го слова комманды, когда у нас загружено только 2).

*Блок формирования управляющих испульсов* превращает входной сигнал тактирующих импульсов в необходимую последовательность сигналов для управления всем блоком. Так как чтение/запись в память у нас является асинхронной операцией, то данный блок так же формирует задержку на время, пока не будут готовы все необходимые данные.

*Блок исполнения* комманд коммутирует все остальные блоки, позволяя правильно функционировать всему блоку УУ.



Рис. 8 Условно-графическое обозначение устройства управления.

Устройство управления содержит следующие входы:

* *clock* – входной тактирующий сигнал;
* *ready\_from\_cache* – сигнал о готовности данных из стека;
* *data\_from\_cache* – данные из стека;
* *flag\_above\_zero* – сигнал о положительном результате от АЛУ;
* *flag\_cf\_from\_alu* – сигнал о наличии переноса от АЛУ;
* *data\_result\_from\_alu –* результат операции из блока АЛУ;
* *stack\_overflow\_from\_alu –* сигнал о переполнении стека;
* *data\_from\_ron* – данные из блока РОНов;
* *wrong\_address –* сигнал о неправильном номере ячейки.

Устройство управления содержит следующие выходы:

* *control\_for\_cache* – управляющие сигналы для кэша;
* *data\_for\_cache* – данные для кэша;
* *address\_for\_cache* – адрес для кэша;

**2.8** Блок АЛУ

Блок АЛУ (рис. 9) используется для выполнения арифметических и логических операций. Разрабатываемая микро-ЭВМ позволяет выполнить следующие операции: CMP, AND, OR, ROL.

Данный блок принимает на свои входы значения двух операндов, выполняет над ними нужную операцию. В зависимости от команды, УУ смотрит нужный выход и использует этот результат. Все команды кроме ROL выполняются за 1 такт. Время выполнения операции циклического сдвига выполняется в зависимости от второго операнда. Поэтому для определения готовности результат используется выход *readyRol*.



Рис. 9 Условно-графическое обозначение блока АЛУ.

Блок АЛУ содержит следующие входы:

* *OP1* – данные первого операнда;
* *OP2* – данные второго операнда;
* *clk* – тактирующий сигнал.

Блок АЛУ содержит следующие выходы:

* *AeB*– сигнал о равенстве первого и второго операндов;
* *AgB* – сигнал о том, что первый операнд больше второго;
* *AlB* – сигнал о том, что первый операнд меньше второго;
* *AandB*  – результат выполнения логической операции AND;
* *AorB* – результат выполнения логической операции OR;
* *readyROL* – сигнал о выполнении циклического сдвига влево;
* *Shifted* – результат выполнения циклического сдвига влево.

**2.9** Блок предсказания переходов

Предсказание переходов применяется в системах с наличием конвейера. От исхода команды условного перехода зависит подготовленность последующих операций для выполнения. Система предсказания переходов может быть двух видов: статической и динамической. Для статической системы всегда предсказывается только один исход. Данная система облегчает реализацию, однако в большинстве систем она не применима.

Перехода

не было

Перехода

не было

11

10

Переход

был

Переход

произошёл

00

01

Переход

был

Рис.10 Состояния автомата А3.

Динамическая система предсказаний основана на таблице шаблонов истории (Pattern History Table – PHT). Она представляет собой набор из конечных автоматов. В разрабатываемой системе используется автомат А3 (рис.10). По условию шаблон имеет дину 4 бита, поэтому таблица PHT состоит из 16 автоматов. Состояния автоматов кодируется двумя битами, и изменяются на основе того, произошел ли переход на самом деле, или нет. Выбор элемента из таблицы PHT осуществляется по значению конкатенации двух бит из значения регистра PC и двух бит из глобальной истории переходов (GHT).



Рис.11 Условно-графическое обозначение блока

предсказания переходов

Блок предсказания переходов имеет следующие входы:

* *WasJmp* – сигнал, указывающий был ли переход на самом деле;
* *CLK* – тактирующий сигнал;
* *PC*– текущее значение регистра PC.

Блок предсказания переходов имеет один выход:

* *WillBeJmp* – результат предсказания перехода.

1. ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ

**3.1** Запоминающие устройства

Результаты моделирования работы ПЗУ в САПР Quartus II представлены на рис. 13. Мы производим чтение, указав адрес и установив сигнал для чтения. Сначала по адресу 0x00 – результат 1, затем по адресу 0x01 – 2. Как видно по рис.12 результаты совпадают. Чтение происходит по фронту тактирующего сигнала.

C:\Users\Dmitry\Downloads\co\rom_d.JPG

Рис.12 Дамп памяти ПЗУ

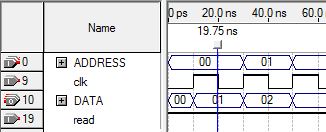


Рис.13 Функциональное моделирование блока ПЗУ

Результаты моделирования работы ОЗУ представлены на рис. 15. Дамп памяти ОЗУ до симуляции изображен на рис. 14, после – рис. 16. Считываем значения по адресам 0x00, 0x01, 0x02. Далее производим запись по адресу 0x02 значение AA. Убеждаемся в правильности результата, используя дампы памяти.

C:\Users\Dmitry\Downloads\co\rom_d0.JPG

Рис.14 Дамп памяти ОЗУ

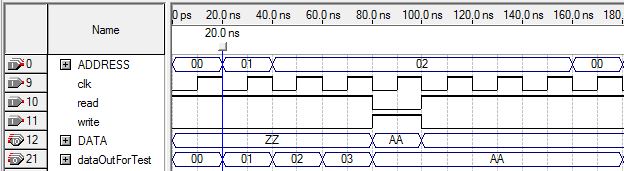


Рис.15 Функциональное моделирование блока ОЗУ

C:\Users\Dmitry\Downloads\co\rom_d1.JPG

Рис.16 Дамп памяти ОЗУ после симуляции

* 1. Блок регистров общего назначения

Результат моделирования работы блока регистров общего назначения представлен на рис. 17.

Сначала в регистре AX и BX 0. Далее мы записываем AX значение 0A, в BX – 03.

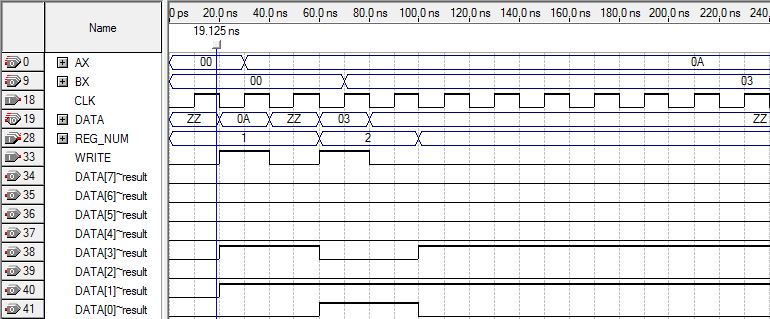


Рис.17 Функциональное моделирование блока РОН

* 1. Блок стека

Результат моделирования работы блока стека представлен на рис. 18. Начальный адрес (значение SP) после инициализации FF. После первой записи SP уменьшается и становится FE, после второй – FD. После считывания увеличивается – FE.

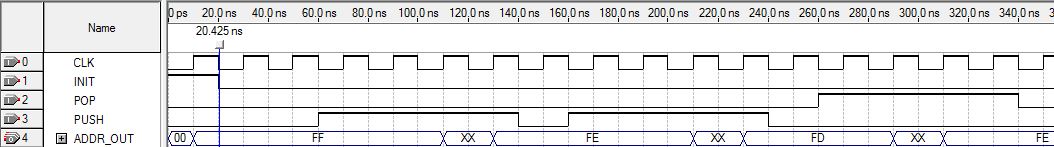


Рис. 18 Функциональное моделирование блока стека.

* 1. Блок КПДП

Результат моделирования работы блока КПДП представлен на рис. 21. На первом такте происходит инициализация служебных регистров: начальный адрес источника 0x01, начальный адрес приемника 0xAA, количество слов для передачи 5. Источник – ОЗУ, дамп памяти на рис. 19. Приемник – периферийное устройство, дамп памяти до передачи данных на рис. 20. В успешной передачи данных можно убедиться, взглянув дамп памяти устройства после окончания передачи на рис. 22.

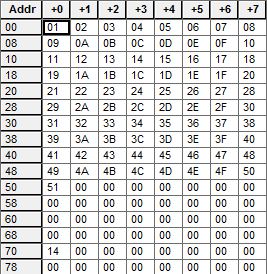


Рис. 19 Дамп памяти ОЗУ.

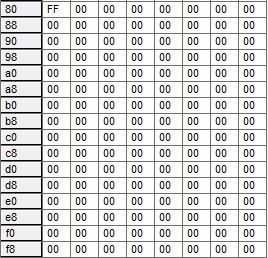


Рис. 20 Дамп памяти периферийного устройства до передачи данных.

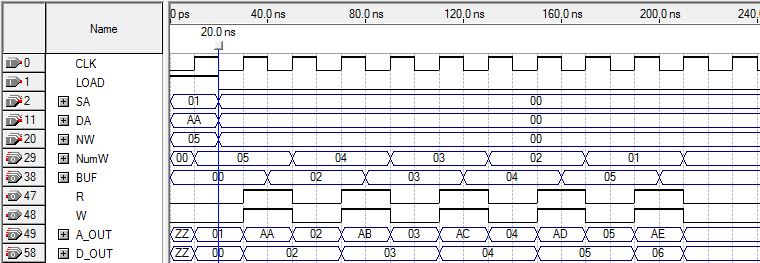


Рис. 21 Функциональное моделирование блока КПДП.

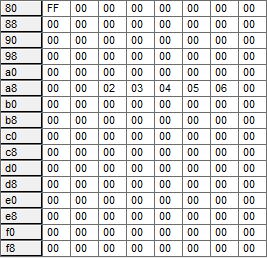


Рис. 22 Дамп памяти периферийного устройства после передачи данных.

* 1. Блок кэш-памяти

Результат моделирования работы блока кэша памяти представлены на рис. 23 и рис. 24.

На рис. 23 сначала мы считываем данные по адресу 0x00, однако кэш не инициализирован, поэтому строка, находящаяся в кэше не записывается в память. Выходы CELL00, CELL01, CELL02, CELL03 отображают состояние первой строк. На 4-7 тактах происходит запись в регистры кэша значений с шины данных от ОЗУ (выход dataFromRAM), считанных по адресам, которые мы можем наблюдать на выходе adrToRAM (miss). Далее на временном интервале от 170 нс до 210 нс происходит считывание значений, которые уже находятся в кэше (hit). Они попадают на выход dataToCPU. На верменном интервале от 210 нс до 230, происходит запись по адресу, данные по которому находятся в кэше, поэтому обращений к памяти нет (hit).

На рис. 24 можно увидеть, как происходит замещение строки кэша (miss). От 850 нс до 930 нс происходит запись слов из первой строки в память. На шину адреса выставляются адреса этих слов (adrToRam), на шину dataToRam сами значения. Далее на интервале от 950 происходит запись новой строки в кэш.

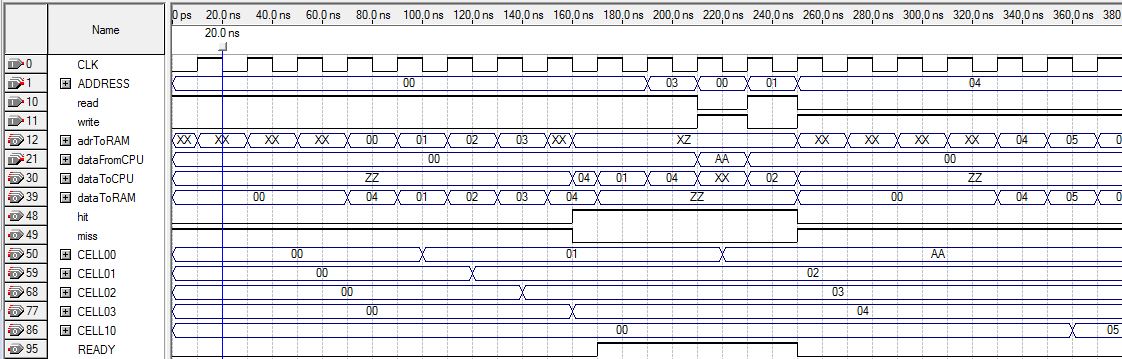


Рис. 23 Функциональное моделирование блока стека.

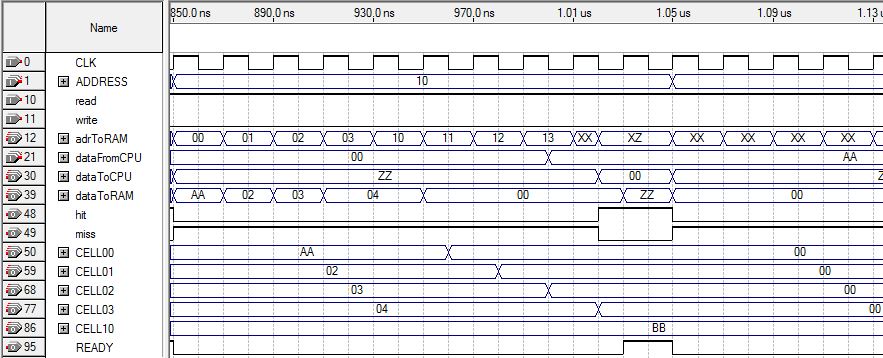


Рис. 24 Функциональное моделирование блока стека.

* 1. Устройство управления

Результат моделирования работы блока устройства управления в среде Quartus представлен в приложениях Е1 и Е2 (первый промежуток вермени), Е3 и Е4 (второй промежуток времени).

Как видно из схемы, устройство управления сначала начинает считывать команду по адресу $0000, затем первый операнд по адресу $0001. Т.к. первая команда у нас получилась addc, то больше операндов нам не надо, поэтому сразу после последней загрузки из кэша у нас начинается загрузка операндов из блока РОНов. Затем непосредственно происходит операция сложения, после этого результат сохраняется в блок РОНов. Затем следует чтение и исполнение команд push и pop. Затем идёт команда mov reg, reg. После этого идёт команда mov с базовой индексной адресацией. В качестве базового регистра используется пара регистров R10 и R11. Затем идёт mov addr, reg. Затем идёт команда mov reg, addr. Затем идёт команда безусловного перехода jmp, после чего УУ блокируется в ожидании байта данных.

* 1. Блок АЛУ

Результат моделирования работы АЛУ в среде Quartus представлен на рисунке 25.

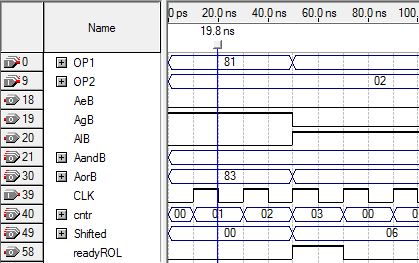


Рис. 25 Функциональное моделирование блока АЛУ.

На входы АЛУ подаются операнды. Результаты команд CMP (выходы AeB, AgB, AlB), AND (AandB), OR (AorB) готовы сразу. Результат команды ROL виден на выходе Shifted, он является валидным после установки сигнала readyROL.

* 1. Описание временной диаграммы работы всего устройства

Для демонстрации работы программы для демонстрации работы ЭВМ использовался следующий набор команд.

Символьный вид.

; тестирование команд MOV, заполнение кэша

MOV1 BX, $0x00

MOV3 AX, $0x01

MOV1 CX, $0x05

MOV2 $0x03, AX

MOV4 $0x00, AX

MOV1 CX, $0x08

MOV1 CX, $0x0C

MOV1 CX, $0x10

; проверка команды CMP

CMP AX, $0x00

CMP AX, $0x02

CMP AX, $0x10

JMZ $0x24

; команда расположена по адресу $0x25

JMP $0x4F

; команда расположена по адресу $0x50

; обмен регистров значениями

PUSH AX

PUSH BX

POP BX

POP AX

OR1 AX, $0x02

OR1 AX, $0x05

ROL1 AX, $0x01

HLT

Представление команд в памяти показано на рисунке.

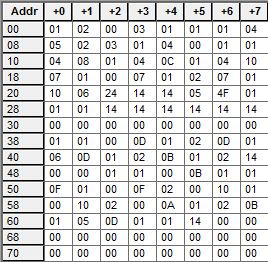


Рис. 26 Дамп памяти ПЗУ.

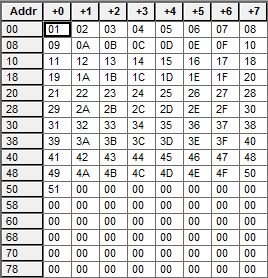


Рис. 27 Дамп памяти ОЗУ до выполнения программы.

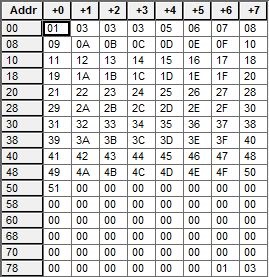


Рис. 28 Дамп памяти ОЗУ после выполнения программы.

В начале работы программы производится демонстрация работы команд MOV с различным типом адресации. Далее идет заполнение кэша данными, вызывая команду MOV.

Затем демонстрируется выполнение команды CMP (от 840 нс до 1100 нс) для случаев: первый операнд больше второго, равны, первый меньше второго. В результате флаг Z устанавливается, поэтому сработает команда условного перехода (от 1,1 мкс до 1,2 мкс). После этого выполняется команда безусловного перехода (от 1,2 мкс до 1,36 мкс).

Демонстрируется работа стека – производим обмен значениями регистров AX и BX (от 1,36 мкс до 1,66 мкс).

Демонстрация логических команд. Первый операнд и результат в регистре AX (от 1,66 мкс до 1,945 мкс).

В конце выполняется команда HLT, после которой приостанавливается работа процессора до его перезагрузки.

Результат функционального моделирования приведён в приложение Б.

**4** АНАЛИЗ И ОПТИМИЗАЦИЯ РАЗРАБОТАННОЙ МИКРО-ЭВМ

Для увеличения производительности микро-ЭВМ необходимо провести ряд улучшений.

Скорость работы разработанной системы увеличится, если уменьшить длину кода операции. Это позволило бы сократить длину некоторых инструкций до 2 слов.

В настоящее время микро-ЭВМ необходимо 3 такта для считывания любой команды. Однако некоторые из них имею лишь одну операнду, что позволяет выполнять считывание операции 2 такта. Требуется ввести динамическое определение количества тактов, необходимое для считывания всей команды. Это выполнение команд микро-ЭВМ.

Также безусловно, внедрение конвейера в разработанную систему позволит уменьшить временные затраты на начальных этапах выполнения команд (выборки, дешифрации команд, подготовки операндов).

ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта мною были изучены принципы построения простейших микро-ЭВМ, был спроектирован один из таких представителей.

Разработанная микро-ЭВМ имеет гарвардскую архитектуру, имеет в составе ПЗУ (256 байт), ОЗУ (128 байт), периферийное устройство с памятью 128 байт, регистры общего назначения, стек, АЛУ, кэш, КПДП. Ширина шины адреса – 8 бит, данных – 8 бит. Максимальное количество тактов необходимое для выполнения любой операции 14.

Как было отмечено в пункте 4, не все конструкторские решения являются оптимальными, возможны улучшения.

СПИСОК ЛИТЕРАТУРЫ

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

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

3. 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.