# ВВЕДЕНИЕ

В настоящее время электронные вычислительные машины получили широкое. В тех случаях, когда размеры устройства становятся критически важными, приходится отказываться от универсальности ЭВМ в пользу миниатюризации и увеличения быстродействия. В связи с этим огромную актуальность приобрели компактные, специализированные, легко встраиваемые микропроцессорные устройства.

Такие устройства проектируются в расчёте на выполнение ограниченного набора функций. При этом основными критериями для проектировщика служат: компактность, быстродействие, минимальные аппаратные затраты, надёжность и т.д.

Таким образом задачей курсового проектирования является разработка схемы микро-ЭВМ на ПЛИС согласно варианту.

На разработку схемы устройства оказывает влияние тип архитектуры, к которому относится микро-ЭВМ. Разрабатываемое устройство соответствует Гарвардскому типу архитектуры. В Гарвардской архитектуре принципиально различаются два вида памяти: память программ (ПЗУ) и память данных (ОЗУ). В Гарвардской архитектуре невозможно производить операцию записи в память программ, что исключает возможность случайного разрушения управляющей программы в случае неправильных действий над данными.

# 1 РАЗРАБОТКА СТРУКТУРЫ МИКРО-ЭВМ

**Вариант:**

Тип архитектуры — Гарвардская

Разрядность Шин :

* + - Память – 10
    - Адреса – 14

Память:

* ПЗУ – асинхр.
* ОЗУ – синхр.
* Тип адресации – косвенная регистровая

Команда условного перехода – JBZ

РОН – 14

АЛУ:

* Ариф. Команды – ADDC
* Логич. Команды – NOT , OR
* Сдвиговые команды – SRA

Стек:

* Объем – 10
* Направление роста – вверх

**1.1 Функциональный состав микро-ЭВМ**

Структура микро-ЭВМ является магистрально-модульной. В такой структуре имеется группа магистралей (шин), к которым подключаются различные модули (блоки), обменивающиеся между собой информацией по одним и тем же шинам поочередно.

Набор функциональных блоков микро-ЭВМ может отличаться в зависимости от задач, выполняемых устройством. Однако можно выделить несколько модулей, характерных для каждого устройства данного типа: блок памяти и блок исполнения команд.

Блок памяти разрабатываемой микро-ЭВМ состоит из ПЗУ и ОЗУ, что позволяет разделить шины команд и данных. Разрядность ячеек в данных блоках соответствует ширине соответствующих шин.

Блок выборки команд отвечает за считывание команд из памяти, наращивании значения счетчика команд.

Блок исполнения команд включает в себя блок декодирования команд, устройство управления, арифметико-логическое устройство, стек и блок регистров общего назначения. Далее приводится описание каждого структурного блока.

Устройство управления микро-ЭВМ обеспечивает выполнение последовательности микроопераций в соответствии с кодом текущей команды и организует выборку команд в соответствии с выполняемой программой. Помимо вышеперечисленного на этот блок отводится задача тактирования команд и управления их различными стадиями.

Одним из основных узлов вычислительной системы является арифметико-логическое устройство. Назначением блока АЛУ является формирование результата арифметических и логических команд над операндами, попадающими на вход. На вход АЛУ, помимо операндов, подается некоторый набор управляющих сигналов, а на выходе формируется требуемое значение в зависимости от выполняемой операции.

Для хранения данных при работе микро-ЭВМ может использоваться стек. Стек – это память с линейно упорядоченными ячейками и специальным механизмом доступа, исключающим необходимость указания адреса при обращении, организованного по принципу FIFO. Стековая память в разрабатываемом устройстве представляет собой набор из 10 регистров, снабженный указателем стека SP. Для доступа к данным используются команд PUSH и POP.

Важной составляющей современных процессоров является блок внутренней памяти, реализованный в виде набора программно доступных регистров, называемых регистрами общего назначения. Число РОН в процессоре разрабатываемого устройства равно 14. При наличии данного блока операнды команд могут размещаться в одной из двух запоминающих сред – в ОЗУ или в РОН. Применение данного блока позволяет сократить время выполнения операций.

**1.2 Разработка системы команд**

В устройстве применяется фиксированная длина команд. Это увеличивает количество памяти, необходимое для хранения программы, но повышает быстродействие системы, т.к. не требуется вычисление адреса, по которому хранится команда.

Общая структура команды представлена в таблице 1.1.1.

Таблица 1.2.1. Структура команды

|  |  |  |
| --- | --- | --- |
| КОП | Адрес операнда 1 | Адрес операнда 2 |
| 4 бита | 6 бит | 5 бит |

В таблице 1.2.2. отображен список команд микро-ЭВМ с соответствующими типами адресации.

Таблица 1.2. 2. Система команд микро-ЭВМ

|  |  |  |
| --- | --- | --- |
| Команда | Адресация операнда | КОП |
| mov reg, $mem | Прямая | 0001 |
| mov $mem, reg | Прямая | 0010 |
| push reg | Прямая | 0011 |
| pop reg | Прямая | 0100 |

Продолжение таблицы 1.2.2.

|  |  |  |
| --- | --- | --- |
| Команда | Адресация операнда | КОП |
| jmp $mem | Прямая | 0101 |
| jbz $mem | Прямая | 0110 |
| hlt |  | 0000 |
| addc reg, reg | Прямая | 0111 |
| not reg | Прямая | 1000 |
| or reg, reg | Прямая | 1001 |
| sra reg, reg | Прямая | 1010 |
| addc reg, [reg] | Косвенная регистровая | 1011 |
| not [reg] | Косвенная регистровая | 1100 |
| or reg, [reg] | Косвенная регистровая | 1101 |
| sra [reg], reg | Косвенная регистровая | 1110 |
| mov reg, [reg] | Косвенная регистровая | 1111 |

**1.3 Взаимодействие блоков микро-ЭВМ при выполнении программ**

Выполнение команды начинается с ее считывания из ПЗУ и заполнения специальных регистров процессора. Команда записывается в регистр IR, сохраняются адреса операндов. На следующем такте увеличивается значение счетчика команд IP.

Значение регистра IR поступает на декодер для определения выполняемых инструкций. Если считанная команда не является командой HLT, то управление передается на один из блоков, отвечающих за выполнение заданной последовательности действий. При определении команды HLT работа микро-ЭВМ приостанавливается.

Под передачей управления подразумевается начало тактирования определенного блока команд, а также открытие элементов lpm\_bustri для синхронизации управляющих сигналов. Так же формируются управляющие команды для работы с блоками RAM и ROM, стеком и регистрам общего назначения.

Когда выполнение операции завершено блок команды устанавливает в единичное значение флаг успешного выполнения команды, после чего обнуляется счетчик в блоке управление и начинается выполнение новой операции.

# 2 РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ

С функциональными схемами блоков можно ознакомиться в Приложении А.

**2.1 Блоки памяти**

Асинхронный блок ПЗУ используется для хранения команд. Данный блок реализован на основе стандартного модуля lpm\_rom. Разрядность ячеек ПЗУ согласно варианту составляет 15 бит. Файл инициализации ROM.hex содержит код тестовой программы.



Рис. 2.1.1 – УГО ROM

Рис. 2.1.2 – Внутренняя структура блока ROM

Входные сигналы:

* address – адрес данных для чтения
* readRom – сигнал, разрешающий считывание данных

Выходной сигнал:

* out – выход данных ROM

Блок ОЗУ представляет собой синхронную память, которая используется для хранения операндов и реализована при помощи стандартного модуля lpm\_ram\_dq. Разрядность каждой ячейки памяти в данном блоке была выбрана исходя из варианта задания, и равна 15 битам.



Рис. 2.1.3 – УГО RAM

Входные сигналы:

* inData – входная шина данных RAM
* command – шина, передающая управляющие сигналы wren и clk
* address – адрес данных для чтения

Выходной сигнал:

* out – выход данных RAM



Рис. 2.1.4 – Внутренняя структура блока ROM

**2.2. Стек и регистры общего назначения**

Блок стека состоит из 10 элементов lpm\_dff, декодера для выбора нужной ячейки и указателя на вершину стека.

Текущий указатель стека указывает на последний добавленный элемент. При выполнении команды push происходит проверка регистра SP. Если SP указывает на последний элемент стека, то формируется сигнал о переполнении, а сам указатель на вершину своего значения не изменяет. При выполнении команды pop так же осуществляется проверка значения SP. Если указатель на вершину отображает отсутствие элементов в стеке, то формируется сигнал об ошибке, а на шину данных выдается значение 0.



Рис. 2.2.1. – УГО Стека

Входные сигналы:

* + clk – тактирующий синхросигнал
  + IN\_DATA – входные данные
  + command – сигналы команд push и pop

Выходные сигналы:

* OUT\_DATA – выходные данные
* stack\_full – флаг переполнения стека
* stack\_empty – флаг сигнализации о попытке извлечения из пустого стека

Блок регистров общего назначения с точки зрения реализации схож с реализацией стека. Он состоит из 14 запоминающих элементов lpm\_dff, декодера, который определяет номер регистра согласно полученному адресу.



Рис. 2.221. – УГО РОН

Входные сигналы:

* + clk – тактирующий синхросигнал
  + IN\_DATA – входные данные
  + command – сигналы команд read и write
  + Address – адрес регистра к которому происходит обращение

Выходной сигнал:

* OUT\_DATA – выходные данные

**2.3. Арифметико-логическое устройство**

Блок АЛУ должен выполнять заданные в условии команды. АЛУ состоит из декодера, необходимого для выбора исполняемой команды, и блоков команд.

Задача блока ADDC состояла в выполнении арифметическую команду ADDC, суть которой заключается в сложении двух входящих операндов со значением флага CF. При этом данная команда также может устанавливать данный бит в регистре флагов, следовательно, с учетом того, что для регистра флагов запись происходит асинхронно, необходимо было обеспечить корректную установку данного флага. Для этого использовался сумматор с входом clock, для обеспечения задержки при записи в один такт.

Команды NOT и OR реализованы с использованием стандартных блоков lpm\_inv и lpm\_or соответственно.

Команда арифметического сдвига SRA была реализована с помощью элемента lpm\_clshift. Однако для использования данного элемента с разрядностью шин, заданных вариантом, потребовало внесения изменений в VHDL код блока. Были изменены размерности входа и выхода элемента.



Рис. 2.3.1 УГО АЛУ

Входные сигналы:

* + Ax – операнд 1
  + Bx – операнд 2
  + comCode – код выполняемой команды
  + inCF – значение флага CF

Выходной сигнал:

* outALU – результат работы АЛУ
* outCF – вывод флага CF
* outPF – вывод флага PF



Рис. 2.3.2 Блок команды ADDC



Рис. 2.3.2 Блок команды SRA



Рис. 2.3.4 Схема блока АЛУ

**2.4 Управляющий блок. CPU. Общая схема**

На рисунке 2.4.1 представлено УГО блока CPU.

****

Рис. 2.4.1. УГО CPU

Входные сигналы:

* + clk – тактирующий синхросигнал
  + romDATA – входные данные из памяти команд
  + ramDATA – входные данные из памяти данных

Выходной сигнал:

* ramAddress – шина адреса для блока памяти данных
* romAddress – шина адреса для блока памяти команд
* command – шин управляющих команд
* stackFull – флаг переполнения стека (используется для тестирования)
* stackEmpty – флаг извлечения из пустого стека (используется для тестирования)
* address – шина адреса внутри блока (используется для тестирования)
* CPUdata – шина данных внутри блока (используется для тестирования)

Наличие разделенных шин адреса и данных обусловлено типом архитектуры.

Реализация блока представлена на рисунке 2.4.2.

В состав общей схемы устройства входят блок CPU и модули памяти RAM и ROM. Для последних выделены отдельные шины адреса и данных, так как разрабатываемая архитектура базируется на Гарвардском типе архитектуры, а значит необходимо представить отдельное адресное пространство для данных и команд.

Так же выделены шины команд command[5..0] и control[14..0]. Шина command используется для управления блоками устройства:

* command[0] – команда push для стека
* command[1] – команда pop
* command[2] – команда read для РОН
* command[3] – команда write для РОН
* command[4] – команда read для RAM
* command[5] – команда write для RAM

Шина control применяется для передачи сигналов, обеспечивающих функционирование устройства.

Общая схема устройства представлена на рисунке 2.4.3.

Управляющий блок включает в себя блок регистров специального назначения и блоки команд. Задачей блока CU является формирование управляющих сигналов и изменения состояния служебных регистров. На рисунках 2.4.4 и 2.4.5 представлен блок регистров специального назначения и общий вид блока команд соответственно. Общая схема управляющего блока приводится в Приложении А.



Рис. 2.4.2 Внутренняя структура блока CPU



Рис. 2.4.3 Общая схема устройства



Рис. 2.4.4 Блок регистров специального назначения



Рис. 2.4.5 Общий вид блока команды

Блоки команд имеют общую структуру. Различие заключается в способах формирования управляющих сигналов.



Рис. 2.4.6 Команда mov $mem, reg

# 3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ

**3.1 Функциональное моделирование блоков стека и РОН**

Блоки стека и регистров общего назначения имеют общую структуру и принцип работы.

На рисунке 3.1.1. представлен результат моделирование стека. Можно заметить, что при попытке извлечения данных из пустого стека выставляется флаг stack\_empty. При попытке записи в заполненный стек выставляется флаг переполнения stack\_full.

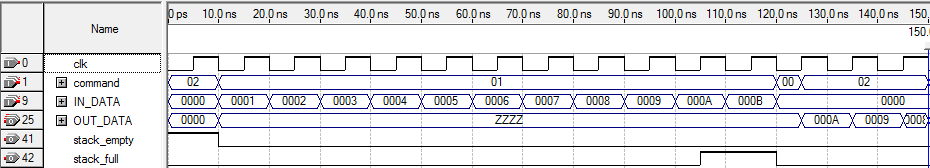


Рис. 3.1.1. Функциональное моделирование стека

На рисунке 3.1.2. отображен результат моделирования блоков регистров.

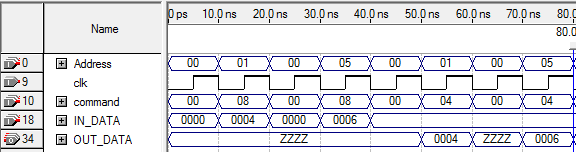


Рис. 3.1.2. Функциональное моделирование блока РОН

**3.2 Функциональное моделирование блоков памяти**

Рисунки 3.2.1 и 3.2.2 отображают результат моделирования асинхронного блока ПЗУ и синхронного блока ОЗУ.

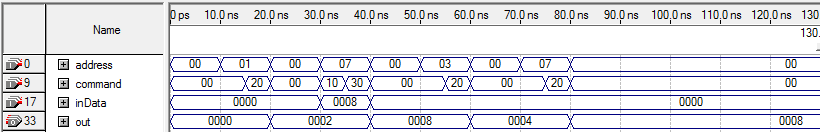
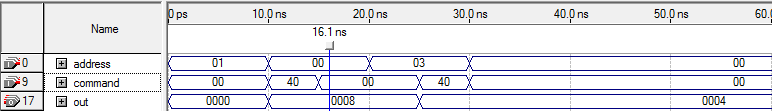


Рис. 3.2.1. Функциональное моделирование блока RAM



**3.3 Функциональное моделирование АЛУ**

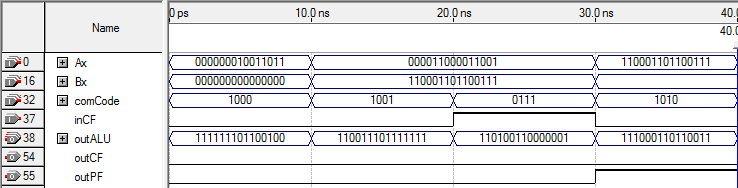


Рис. 3.3.1. Функциональное моделирование АЛУ

**3.4 Моделирование общей схемы**

Для моделирования системы использовалась программа, список команд которой приведен в таблице.

Таблица 3.4.1 Список команд программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Запись на языке ассемблера | Запись в шестнадцатеричной системе | Запись в двоичной системе |
| 0 | MOV $5, 1 | 10А1 | 0010 00010100001 |
| 1 | MOV $2, 3 | 1043 | 0010 00001000011 |
| 2 | PUSH Reg3 | 1860 | 0011 00001100000 |
| 3 | POP 1 | 2080 | 0100 00010000000 |
| 4 | MOV 4, $1 | 881 | 0001 00010000001 |
| 5 | ADDC 3, 1 | 3861 | 0111 00001100001 |
| 6 | NOT 1 | 4020 | 1000 00000100000 |
| 7 | OR 3, 1 | 4861 | 1001 00001100001 |
| 8 | SRA 3, 1 | 5061 | 1010 00001100001 |
| 9 | JBZ $7 | 12512 | 0110 00011100000 |
| 10 | JMP $10 | 2940 | 0101 00101000000 |
| 11 | MOV $7, 6 | 10E6 | 0010 00011100110 |
| 12 | MOV $8, 5 | 1105 | 0010 00100000101 |
| 13 | OR 3, [6] | 6866 | 1101 00001100110 |
| 14 | NOT [5] | 70A1 | 1110 00010100001 |
| 15 | MOV 1, [5] | 25 | 0000 00000100101 |
| 16 | hlt | 7800 | 1111 00000000000 |

## ЗАКЛЮЧЕНИЕ

Разработанное в рамках курсового проектирования устройство имеет ряд достоинств и недостатков. К достоинствам системы можно отнести возможность расширения функционала устройства за счет возможности добавления новых функциональных блоков.

Спроектированная система может быть расширена путем добавления

конвейера и предсказателя переходов, контроллера прямого доступа к памяти

(КПДП), КЭШ памяти и арбитража шин. Так же возможна оптимизация работы путем пересмотра блоков команд в управляющем блоке и уменьшения количества тактов, затрачиваемых на выполнение операций.

Разработка проекта выполнялась в программе Altera Quartus 9.1, которая позволяет разрабатывать схемы различной сложности, проверять их работоспособность путем подачи различных входных значений, а также следить за тем, что происходит с разработанной схемой во время исполнения той или иной команды посредством временных диаграмм.