# ВВЕДЕНИЕ

На данный момент невозможно представить нашу жизнь без электронных вычислительных машин, они внедрены во все сферы жизни человека и упрощают его повседневные задачи. В современном мире размер устройства стал критически важен, в следствии чего 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, цена и другие факторы.

Собственно, задачей курсового проектирования является разработка схемы микро-ЭВМ на ПЛИС согласно выданному варианту.

На разработку схемы устройства оказывает влияние тип архитектуры, что является основным фактором при реализации устройства, и многие второстепенные факторы, к которым относится проектируемая микро-ЭВМ. Разрабатываемое устройство соответствует Гарвардскому типу архитектуры. В Гарвардской архитектуре принципиально различаются два вида памяти: память программ или же команд (ПЗУ) и память данных (ОЗУ). В Гарвардской архитектуре невозможно производить операцию записи в память команд, что исключает возможность случайного разрушения управляющей программы в случае неправильных действий над данными.

Разрабатываемое устройство в минимальной конфигурации будет обладать возможность реализации 16 различных команд. Заданием курсового проекта предусмотрена возможность доработки архитектуры системы команд, непосредственно, для расширения функциональных возможностей разрабатываемой микро-ЭВМ.

# 1 РАЗРАБОТКА СТРУКТУРЫ МИКРО-ЭВМ

Задание, согласно варианту, приведено в таблицах 1.1 – 1.4.

Таблица 1.1 – Задание часть первая

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| №  варианта | Тип архитектуры | Разрядность шин | | Память | | |
| Адреса | Данных | ПЗУ | ОЗУ | Тип адресации |
| 54 | Принстонская | 14 | 9 | Синхронная | Синхронная | Косвенная регистровая |

Таблица 1.2 – Задание часть вторая

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Команда условного перехода | РОН | КЭШ | | |
| k | Алгоритм замещения строк | Синхронизация с памятью |
| JAZ | 8 | 2 | LFU | Сквозная с отображением |

Таблица 1.3 – Задание часть третья

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| АЛУ | | | | Арбитраж шин |
| Арифметические команды | Логические  команды | | Сдвиговые  команды |
| SUB | NOT | NXOR | ROR | Децентрализованный параллельный |

Таблица 1.4 – Задание часть четвертая

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Стек | | Схема предсказания Переходов | | | КПДП | Конвейер |
| Объем | Направление Роста | Тип автомата | Бит | Тип шаблона |
| 10 | Вверх | А5 | 5 | PC |

* 1. **Функциональный состав микро-ЭВМ**

За основу проекта положена магистрально-модульная структура микро-ЭВМ, которая является одной из самых популярных архитектур в современных устройствах. Модульный принцип позволяет потребителю самому комплектовать нужную ему конфигурацию и производить при необходимости ее модернизацию. Модульная организация опирается на магистральный (шинный) принцип обмена информацией между подключенными блоками (модулями).

Набор функциональных модулей микро-ЭВМ может отличаться в зависимости от задач, непосредственно выполняемых устройством, и, собственно, назначения устройства. Однако можно выделить несколько модулей, характерных для каждого устройства данного типа: модуль памяти, модули выборки и выполнения команд, а также модуль кэш-памяти.

Блок памяти, разрабатываемой микро-ЭВМ, состоит из ПЗУ (ROM) и ОЗУ (RAM), что позволяет разделить шины команд и данных. Данное разделение приводит к увеличению производительности устройства в целом. Разрядность ячеек в данных модулях соответствует ширине соответствующих шин, что будет описано при разработке архитектуры системы команд.

Блок выборки команд отвечает за считывание команд из памяти и наращивание значения счетчика команд.

Блок исполнения команд включает в себя блок декодирования команд, устройство управления, арифметико-логическое устройство, стек и блок регистров общего назначения. Далее приведено 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, за счет чего на выходе формируется требуемое значение, которое зависит 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. Стековая память в разрабатываемом устройстве представляет собой набор из 12 регистров, снабженная указателем стека SP. Для доступа к регистрам стека используются команды PUSH и POP.

Важной составляющей современных процессоров является блок внутренней памяти, реализованный в виде набора программно-доступных регистров, называемых регистрами общего назначения (РОН). Число РОН в процессоре разрабатываемого устройства равно 10. При наличии данного блока операнды команд могут размещаться в одной из двух запоминающих сред – в ОЗУ (RAM) или в РОН. Применение данного блока позволяет сократить время выполнения операций, за счет быстрого доступа к операндам.

Блок кэш-памяти является неотъемлемой частью современных микро-ЭВМ. Данный модуль используется процессором для уменьшения среднего времени доступа к памяти. Достигается это посредством хранения часто используемых данных из основной памяти, т.е. когда процессору необходимо обратится в память для чтения или записи данных, он первоначально проверяет, доступна ли их копия в кэше. В случае успеха проверки процессор производит операцию, используя кэш, что значительно быстрее использования основной памяти устройства. Данные между кэшем и основной памятью передаются блоками фиксированного размера.

**1.2 Разработка системы команд**

В разрабатываемом устройстве применяется строго фиксированная длина команды, что увеличивает количество памяти, необходимое для хранения кода программы, но повышает быстродействие системы, т.к. не требуется вычисление адреса памяти, по которому хранится команда, и упрощает представление архитектуры системы команд.

Общая структура команды для прямой регистровой и косвенной регистровой адресаций, назовем ее первая команда, приведена в таблице 1.5.

Общая структура команды для непосредственной адресации, назовем ее вторая команда, приведена в таблице 1.6.

Общая структура команды для прямой адресации, назовем ее третья команда, приведена в таблице 1.7.

Структура команды HLT, назовем ее четвертая команда, приведена в таблице 1.8.

Таблица 1.5 – Структура первой команды

|  |  |  |  |
| --- | --- | --- | --- |
| Безразличные биты | Адрес регистра | Адрес регистра | КОП |
| 17 бит | 3 битa | 3 битa | 4 бита |

Таблица 1.6 – Структура второй команды

|  |  |  |
| --- | --- | --- |
| Безразличные биты | Адрес памяти | КОП |
| 9 бит | 14 бит | 4 бита |

Таблица 1.7 – Структура третьей команды

|  |  |  |  |
| --- | --- | --- | --- |
| Безразличные биты | Адрес памяти | Адрес регистра | КОП |
| 6 бит | 14 бит | 3 бита | 4 бита |

Таблица 1.8 – Структура четвертой команды

|  |  |
| --- | --- |
| Безразличные биты | КОП |
| 23 бита | 4 бита |

В таблице 1.9 представлен список команд микро-ЭВМ с соответствующими типами адресации и кодами операций.

Таблица 1.9 – Система команд микро-ЭВМ

|  |  |  |
| --- | --- | --- |
| Команда | Адресация операнда | КОП |
| MOV Reg, [Reg] | Косвенная регистровая | 0000 |
| SUB Reg, [Reg] | Косвенная регистровая | 0001 |
| NOT [Reg] | Косвенная регистровая | 0010 |
| NXOR Reg, [Reg] | Косвенная регистровая | 0011 |
| ROR Reg, [Reg] | Косвенная регистровая | 0100 |
| JMP Adr | Непосредственная | 0101 |
| JAZ Adr | Непосредственная | 0110 |
| MOV Adr, Reg | Прямая | 0111 |
| MOV Reg, Adr | Прямая | 1000 |
| SUB Reg, Reg | Прямая регистровая | 1001 |
| NOT Reg | Прямая регистровая | 1010 |
| NXOR Reg, Reg | Прямая регистровая | 1011 |
| ROR Reg, Reg | Прямая регистровая | 1100 |
| PUSH Reg | Прямая регистровая | 1101 |
| POP Reg | Прямая регистровая | 1110 |
| HLT | - | 1111 |

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

Выполнение команды начинается с ее считывания из ПЗУ (ROM) и заполнения специальных регистров процессора (Регистров особого назначения). Команда записывается в регистр IR (регистр команд), 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 и шин. Так же формируются управляющие команды для работы с блоками RAM и ROM, кэш-памятью, стеком и регистрами общего назначения (РОН).

Когда выполнение операции завершено блок команды устанавливает в единичное значение флаг успешного выполнения команды, после чего обнуляется счетчик в блоке управления и в случае поступления новой команды (операции) начинается ее выполнение.

# 2 РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ

С функциональными схемами блоков можно ознакомиться в конце пояснительной записки.

**2.1 Блок памяти**

Блок RAM 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. Схема внутренней структуры блока RAM приведена на рисунке 2.4.

Рисунок 2.3 – УГО RAM

Входные сигналы:

* Address – шина адреса данных для чтения;
* Data – входная шина данных RAM;
* Control – шина из трёх сигналов: сигнала, разрешающего считывание данных, сигнала, разрешающего запись данных и тактирующего сигнала clock.

Выходной сигнал:

* Data – шина вывода данных RAM.

Рисунок 2.4 – Внутренняя структура блока ROM

**2.3 Блок стек-памяти**

Блок стека состоит из 12 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. Схема внутренней структуры блока стек-памяти приведена в конце ПЗ (Э1).

Рисунок 2.5 – УГО Стека

Входные сигналы:

* + Clock – тактирующий сигнал;
  + Data – входная шина данных;
  + Control – шина из двух сигналов: сигнала push и сигнала pop.

Выходные сигналы:

* Data – шина вывода данных;
* Overflow – флаг переполнения стека;
* Empty – флаг пустого стека.

**2.4 Блок регистров общего назначения**

Блок регистров общего назначения с точки зрения реализации схож с реализацией стека. Он состоит из 10 запоминающих элементов lpm\_dff и декодера, который определяет номер регистра согласно полученному адресу. Схема внутренней структуры блока РОН приведена в конце ПЗ (Э2).

Рисунок 2.6 – УГО РОН

Входные сигналы:

* Address – шина адреса регистра;
* Data – входная шина данных;
* Control – шина из трёх сигналов: сигнала, разрешающего считывание данных, сигнала, разрешающего запись данных и тактирующего сигнала clock.

Выходной сигнал:

* Data – шина вывода данных.

**2.5 Арифметико-логическое устройство**

Блок АЛУ должен выполнять заданные в условии команды. АЛУ состоит из декодера, необходимого для выбора исполняемой команды, и блоков команд. Схема общей внутренней структуры блока АЛУ приведена в конце ПЗ (Э3).

Задача блока ADDC состояла в выполнении арифметической команды 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. Схема внутренней структуры команды ADDC приведена на рисунке 2.8.

Команда логического сдвига SLL была реализована с помощью элемента lpm\_clshift. Схема внутренней структуры команды SLL приведена на рисунке 2.9.

Команды NOT и NOTZ реализованы с использованием стандартного блока lpm\_inv, команда NOTZ, собственно, с учетом флага ZF. Схема внутренней структуры команд NOT и NOTZ приведена на рисунке 2.10.

Рисунок 2.7 – УГО АЛУ

Входные сигналы:

* + FirstOperand – первый операнд;
  + SecondOperand – второй операнд;
  + CommandCode – код выполняемой команды;
  + Implement – разрешающий сигнал;
  + PreviuosFlagValueZF – значение флага ZF;
  + PreviuosFlagValueCF – значение флага CF.

Выходные сигналы:

* PreviuosFlagValueCF – вывод флага СF;
* PreviuosFlagValueZF – вывод флага ZF;
* Result – шина вывода результата вычислений.

**2.6 Организация кэш-памяти процессора**

В связи с построением проекта на базе Гарвардской архитектуры, блок кэш-памяти физически разделен на блок кэш-памяти команд и блок кэш-памяти данных. Задача блока кэш-памяти команд состоит в чтении и хранение команд, что способствует ускорению работы, в связи с быстрым доступом к последующим выполняемым командам.

Задача блока кэш-памяти данных совпадает с блоком кэш-памяти команд, за исключением возможности записи данных в память, что обусловлено работой RAM. Схемы внутренней структуры блока кэш-памяти приведены в конце ПЗ (Э4 – Э8).

Рисунок 2.12 – УГО блока кэш-памяти

Входные сигналы:

* Address – шина адреса данных;
* InputData – входная шина считанных данных;
* ResultUpdate – входная шина данных для записи;
* Read – сигнал, разрешающий считывание данных;
* Write – сигнал, разрешающий запись данных;
* Clk – тактирующий сигнал.

Выходное сигналы:

* CPU – шина вывода данных;
* ramAddress – шина адреса команды в памяти RAM;
* control – шина из трёх сигналов для управления памятью RAM: сигнала, разрешающего чтение, сигнала, разрешающего запись, и тактирующего сигнала;
* hit – сигнал определения необходимых данных;
* display – сигнал записи изменённого значения;
* Change – шина для вывода изменяемых значений.

**2.7 Управляющий блок. CPU**

Входные сигналы:

* + Clock – тактирующий сигнал;
  + Command – шина входных данных из памяти команд;
  + Data – шина входных данных из памяти данных.

Выходные сигнал:

* CommandCPU – шина вывода команды;
* RomAddress – шина адреса для блока памяти команд;
* ControlRom – шина управляющих команд для памяти команд;
* HitRom – сигнал определения необходимой команды;
* CPU – шина вывода данных;
* CommandCacheAddress – шина адреса команды для кэш-памяти команд;
* ReadCommandCache – сигнал, разрешающий чтение команд из кэш-памяти команд;
* CommandCacheClock – тактирующий сигнал для начала работы кэш-памяти команд;
* DataCacheAddress – шина адреса данных для кэш-памяти данных;
* DataCacheRead – сигнал, разрешающий чтение данных из кэш-памяти данных;
* RegisterAddress – шина адреса регистра, содержащего операнд или адрес операнда в памяти;
* DataCacheClock – тактирующий сигнал для начала работы кэш-памяти данных;
* ControlRegister – шина управляющих команд для РОН;
* DataCacheWrite – сигнал, разрешающий запись данных в память;
* ALU – шина вывода результата вычислений операций блоком АЛУ;
* Overflow – флаг переполнения стека;
* Empty – флаг пустого стека;
* RamAddress – шина адреса для блока памяти данных;
* ControlRam – шина управляющих команд для памяти данных;
* Display – сигнал записи изменённого значения;
* Data – шина выходных данных.

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.

Управляющий блок включает в себя блок регистров специального назначения и блоки команд и интерфейсы для их реализации. Задачей блока CU является формирование управляющих сигналов и изменения состояния служебных регистров.

Формированием сигналов для управления блоками устройства занимаются исполнительные блоки команд, для каждой реализовываемой команды разработаны соответствующие блоки и управляющие сигналы для них, а также общий интерфейс управления чтением и реализацией команд.

С общей схемой устройства управления, а также внутренними схемами команд можно ознакомится в конце ПЗ (Э9 – Э12).

С общей схемой микро-ЭВМ можно ознакомится в конце ПЗ (Э13).

# 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. представлен результат моделирование стека. Можно заметить, что при попытке извлечения данных из пустого стека выставляется флаг Empty. При попытке записи в заполненный стек выставляется флаг переполнения Overflow. На первых четырех тактах происходит внесение данных в стек-память, на последующих четырёх, начиная с 45нс, происходит вынесение данных.

Рисунок 3.1 – Функциональное моделирование стека

На рисунке 3.2. представлен результат моделирование РОН. Блок работает следующим образом: по наличию тактирующего сигнала и сигнала запись в регистр, декодер выставляет сигнал разрешения записи на регистр, номер которого соответствует пришедшему адресу, а в случае чтения данных из регистра, декодер также анализирует пришедший адрес и выставляет сигнал разрешения чтения данных. Исполнение записи можно увидеть на первых трех тактах моделирования, исполнение чтения на последующих трех.

Рисунок 3.2 – Функциональное моделирование блока РОН

**3.2 Функциональное моделирование блока памяти**

Рисунки 3.3 и 3.4 отображают результат моделирования блока ROM и блока RAM. На рисунке 3.5 представлен дамп памяти RAM до моделирования, на рисунке 3.6, соответственно, дамп памяти RAM после моделирования.

Рисунок 3.3 – Функциональное моделирование блока RAM

Рисунок 3.4 Функциональное моделирование блока ROM

Рисунок 3.5 Дамп памяти RAM до моделирования

Рисунок 3.6 Дамп памяти RAM после моделирования

**3.3 Функциональное моделирование АЛУ**

Рисунок 3.7 отображает результат моделирования блока АЛУ. На первом и пятом тактах реализована команда ADDC, на пятом такте с флагом переноса. На втором и шестом тактах реализована команда NOT. На третьем и седьмом тактах реализована команда NOTZ, на седьмом такте с флагом нуля. На четвертом и восьмом тактах реализована команда SLL.

Рисунок 3.7 – Функциональное моделирование блока АЛУ

**3.4 Примеры моделирования команд**

На рисунке 3.8 приведено моделирование команд ADDC и SLL, на 365нс начинается исполнение команды ADDC, последовательно происходит чтение значений из регистров 5 и 9, это видно исходя из моделирования, после чего происходит суммирование полученных значений. На 435нс начинается исполнение команды SLL, также последовательно происходит чтение значений из регистров 5 и 9, после чего происходит сдвиг значения пятого регистра на количество бит, равное значению 9 регистра.

Рисунок 3.8 – Выполнение команд суммирования и сдвига

Рисунок 3.9 – Выполнение команды чтения данных в регистр

На рисунке 3.9 приведено моделирование команды MOV Register, Address, на 85нс начинается исполнение команды MOV, последовательно происходит чтение данных в кэш-память, это видно исходя из моделирования, после чего на 160нс происходит вывод необходимых данных на общую шину данных CPU и последующее формирование управляющих сигналов для РОН, а также запись необходимого значения.

**3.5 Моделирование общей схемы**

Как уже было сказано ранее в общую схему микро-ЭВМ входят блоки RAM и ROM памяти, а также блок CPU, состоящий в свою очередь из блоков АЛУ, стек-памяти, кэша-данных, кэша-памяти и устройства управления. Взаимодействие блоков начинается по подаче тактирующего сигнала clock на блок CPU, который в свою очередь идет на устройство управления, где и формируются управляющие сигналы. После начала тактирования, устройством управления формируется сигнал о чтение команд из кэша команд, который в свою очередь формирует сигнал чтения команд и ROM памяти. После выполнения чтения, регистр IR загружает первую команду, после чего происходит её декодирование и исполнение. После того как команда завершена, блок исполнения команды выдает сигнал о завершении функционирования, после чего указатель IP наращивается и происходит чтение последующей команды из кэш-памяти команд, в случае если команда уже содержится в кэш-памяти, простоя блока не происходит, а сразу же начинается исполнение считанной команды.

Для моделирования системы использовалась программа, список команд которой приведен в таблице 3.1.

Таблица 3.1 – Список команд программы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Запись в символьном представлении | Запись в шестнадцатеричной системе | Запись в двоичной системе |
| 1 | MOV Reg5, $100 | 135C5 | 00010001100100000101 |
| 2 | MOV $215, Reg5 | 235C5 | 00100011010111000101 |
| 3 | PUSH Reg5 | 32800 | 00110010100000000000 |
| 4 | MOV Reg1, $25 | 10641 | 00010000011001000001 |
| 5 | JMP $8 | 50200 | 01010000001000000000 |
| 6 | POP Reg 9 | 44800 | 01000100100000000000 |
| 7 | ADDC Reg5, Reg9 | 82A40 | 10000010101001000000 |
| 8 | NOT Reg9 | 94800 | 10010100100000000000 |
| 9 | SLL Reg2, Reg9 | B1240 | 10110001001001000000 |
| 10 | JZ $16 | 60400 | 01100000010000000000 |
| 11 | NOTZ Reg5 | A2800 | 10100010100000000000 |
| 12 | SLL Reg5, Reg9 | B2A40 | 10110010101001000000 |
| 13 | ADDC Reg5, [Reg9] | C4940 | 11000100100101000000 |
| 14 | JMP $24 | 50600 | 01010000011000000000 |
| 15 | NOT [Reg9] | D4800 | 11010100100000000000 |
| 16 | NOTZ [Reg5] | E2800 | 11100010100000000000 |
| 17 | SLL Reg5, [Reg9] | F4940 | 11110100100101000000 |
| 18 | MOV Reg7, [Reg9] | 749C0 | 01110100100111000000 |
| 19 | PUSH Reg7 | 33800 | 00110011100000000000 |
| 20 | HLT | 00000 | 00000000000000000000 |

## 

На рисунке 3.10 приведено моделирование чтения первых четырех команд и выполнение команды MOV Register, Address, до 75нс происходит последовательное чтение команд в кэш-память команд, после чего первая команда поступает на УУ. На 85нс начинается исполнение команды MOV, последовательно происходит чтение данных в кэш-память, это видно исходя из моделирования, после чего на 160нс происходит вывод необходимых данных на общую шину данных CPU и последующее формирование управляющих сигналов для РОН, а также запись необходимого значения в пятый регистр.

Рисунок 3.10 – Функциональное моделирование чтения команд и

выполнение команды MOV Register, Address

На рисунке 3.11 приведено моделирование команд MOV Address, Register, PUSH Register и MOV Register, Address, на 225нс начинается исполнение первой команды MOV, происходит запись данных из пятого регистра в ячейку памяти по адресу 215, после чего на 285нс происходит выполнение команды PUSH, как видно из моделирования, данные из пятого регистра записываются в стековую память. На 325нс происходит выполнение второй команды MOV, на 400нс происходит вывод данных на общую шину данных и запись полученных данных в первый регистр.

Рисунок 3.11 – Функциональное моделирование команд MOV Address, Register, PUSH Register и MOV Register, Address

На рисунке 3.12 приведено моделирование команд JMP Address, POP Register и ADDC Register, Register. На 525нс начинается исполнение команды JMP, происходит перемещение указателя команд на значение равное восьми, после чего на 565нс происходит чтение команд из памяти команд начиная с восьмого адреса. На 645нс происходит выполнение команды POP, как видно из моделирования, данные из стековой памяти записываются в девятый регистр. На 685нс происходит выполнение команды ADDC, данные из пятого и девятого регистров последовательно выводятся на шину данных, после чего происходит суммирование полученных данных.

Рисунок 3.12 – Функциональное моделирование команд JMP Address,

POP Register и ADDC Register, Register

На рисунке 3.13 приведено моделирование команд NOT Register, SLL Register, Register и JZ Address. На 755нс начинается исполнение команды NOT, происходит инвертирование данных из девятого регистра, после чего на 825нс происходит команда сдвига, которая получает на выходе нулевое значение, устанавливая флаг. На 945нс происходит выполнение команды JZ, после чего указатель команд смещается на значение равное шестнадцати.

Рисунок 3.13 – Функциональное моделирование команд NOT Register,

SLL Register, Register и JZ Address

На рисунке 3.14 приведено моделирование команд NOTZ Register, SLL Register, Register и ADDC Register, [Register]. На 1075нс начинается исполнение команды NOTZ, т.к. флаг ZF не сброшен, то инвертирование не происходит и на шину данных выводится первоначальное значение, флаг ZF остается в единичном состоянии, после чего на 1135нс происходит выполнение команды сдвига, которая сбрасывает флаг ZF. На 1215нс исполняется команда ADDC, происходит чтение данных по адресу, значение которого хранится в пятом регистре, после чего на 1313нс происходит суммирование полученных данных со значением из девятого регистра.

Рисунок 3.14 – Функциональное моделирование команд NOTZ Register,

SLL Register, Register и ADDC Register, [Register]

На рисунке 3.15 приведено моделирование команд JMP Address, NOT [Register], NOTZ [Register] и SLL Register, [Register]. На 1365нс начинается исполнение команды JMP, происходит смещение указателя команд на значение равное двадцати четырем. На 1475нс происходит выполнение команды NOT, а на 1545нс – команды NOTZ, обе команды инвертируют значение полученные из девятого и пятого регистров соответственно. На 1615нс происходит выполнение команды логического сдвига SLL.

Рисунок 3.15 – Функциональное моделирование команд JMP Address,

NOT [Register], NOTZ [Register] и SLL Register, [Register]

На рисунке 3.16 приведено моделирование команд MOV Register, [Register] и PUSH Register. На 1695нс начинается исполнение команды MOV, происходит чтение данных по адресу, значение которого хранится в девятом регистре, после чего происходит запись полученных данных в седьмой регистр. На 1835нс происходит выполнение команды PUSH, данные из седьмого регистра заносятся в стековую память.

Рисунок 3.16 – Функциональное моделирование MOV Register, [Register]

и PUSH Register

Рисунок 3.17 – Дамп RAM памяти после моделирования

## ЗАКЛЮЧЕНИЕ

Разработанная в рамках курсового проектирования микро-ЭВМ имеет ряд достоинств и недостатков. К достоинствам системы можно отнести возможность расширения функционала устройства за счет добавления новых функциональных блоков.

Спроектированная система может быть расширена путем добавления

конвейера и предсказателя переходов, контроллера прямого доступа к памяти

(КПДП) 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.