[ВВЕДЕНИЕ 3](#_Toc356548044)

[2 РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ МИКРО-ЭВМ 5](#_Toc356548045)

[2.1 Функциональный состав микро-ЭВМ. 5](#_Toc356548046)

[2.2 Разработка системы команд. 7](#_Toc356548047)

[2.3 Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы. 9](#_Toc356548048)

[3 РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ 12](#_Toc356548049)

[3.1 Запоминающие устройства. Функциональный состав ПЗУ и ОЗУ. 12](#_Toc356548050)

[3.2 Устройство управления 15](#_Toc356548051)

[3.3 Стек. 16](#_Toc356548052)

# ВВЕДЕНИЕ

В соответствии с вариантом необходимо разработать микро-ЭВМ, основанную на гарвардской архитектуре (хранилища и каналы инструкций и данных физически разделены). Шина адреса имеет ширину 16 бит, а шина данных — 8 бит.

Постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ) должны иметь синхронное и асинхронное управление соответственно.

Устройство должно иметь возможность работать с прямой и прямой регистровой адресацией.

Микро-ЭВМ должна включать в себя 12 регистров общего назначения (РОН).

Сквозная кэш память с отображением должна иметь уровень ассоциативности равный 2. Алгоритм замещения строк — наименьшего использования.

Поддерживаемые операции должны включать в себя: **HLT, NOP, MOV, JMP, JNZ, PUSH, POP, SUB, AND, NOTZ, SRL.**

Стек должен расти вниз и вмещать 9 слов.

Микро-ЭВМ должна включать в себя децентрализованный кольцевой арбитр.

Предсказатель переходов должен работать по схеме А5 и использовать 3-битный Local HR.

Разработку описанного выше устройства будет проводиться с использованием специализированного приложения Altera Quartus II 9.1. Опыт работы в данном приложении был получен мной в ходе выполнения лабораторных работ по курсу в течении предыдущих семестров обучения. Данная среда разработки позволяет быстро создавать отдельные модули, тестировать их, а также инкапсулировать их в отдельные символы для дальнейшего использования.

# РАЗРАБОТКА ОБЩЕЙ СТРУКТУРЫ МИКРО-ЭВМ

В данном разделе описаны особенности функционального строения микро-ЭВМ, описание взаимодействия всех блоков и архитектура системы команд.

## Функциональный состав микро-ЭВМ.

В соответствии с заданием необходимо включить в реализацию следующие функциональные блоки:

* Устройство управления (УУ);
* Арифметико-логическое устройство (АЛУ);
* Стек;
* Регистры общего назначения (РОН);
* Оперативное запоминающее устройство (ОЗУ);
* Постоянное запоминающее устройство (ПЗУ);
* Арбитр

Рассмотрим подробнее каждый из блоков.

Устройство управления определяет набор возможностей управления состоянием центрального процессора микро-ЭВМ. Основное его назначение — контроль за работой микро-ЭВМ, отображение текущего состояния системы в целом, а также изменение этого состояния.

Арифметико-логическое устройство выполняет необходимые операции над предоставленными ей операндами. В данном устройстве определен набор команд, которые позволяют определенным образом изменять предоставленные данные. Как правило такими командами являются арифметические и логические операции.

Стек представляет из себя набор регистров, из которых в один момент времени виден только «верхний» — тот регистр, данные в котором наиболее новые. Для стека возможны три операции: помещение данных в стек, извлечение данных из стека, просмотр данных на вершине стека.

Блок регистров представляет из себя набор энергозависимой памяти небольшого объема, предназначенный для кратковременного хранения оперативной информации. Фактически, один регистр является набором триггеров для хранения данных и схемы для удобного управления работой этих триггеров. Блок регистров включает в себя набор регистров, каждый из которых имеет собственный адрес (имя/номер) в этом блоке. Регистр предоставляет следующие операции: чтение данных из регистра, запись данных в регистр.

Постоянное запоминающее устройство предназначено для долговременного хранения неизменяемых данных. Как правило такими данными является программа работы микро-ЭВМ.

Оперативное запоминающее устройство используется для энергозависимого хранения данных. По сравнению с регистрами, ОЗУ обладает большим временем доступа и большей вместимостью, что, как правило, объясняется расположением ОЗУ отдельно от ядра микро-ЭВМ, тогда как регистры обычно находятся на том же чипе. Асинхронный протокол работы ОЗУ обеспечивает стабильность работы при разных тактовых частотах памяти и процессора. Обратной стороной является уменьшение производительности вследствие усложнения протокола взаимодействия.

Арбитраж необходим в тех случаях, когда ведущим устройством на шине может быть не одно устройство, а несколько. В таких случаях специальным образом выбирается приоритетное устройство, которое и становится ведущим на шине. Децентрализованный кольцевой арбитр является вариантом реализации схемы арбитража, при котором каждое потенциально ведущее устройство запрашивает доступ к шине у одного из узлов арбитра (закрепленного за данным ведущим устройству), и при этом узлы объединены кольцевой топологией, которая используется для определения приоритетов.

## Разработка системы команд.

Согласно заданию необходимо реализовать 11 основных команд. Для разных видов адресации требуется от 4 до 16 битов на один аргумент операции (4 бита для регистровой адресации, 16 бит для прямой адресации).

Для упрощения реализации операций по обработке команд решено использовать команды фиксированной длины.

Также решено использовать формат команды, включающий в себя до 2 аргументов операции, при котором один аргумент будет фиксированного типа (регистр/стек), а тип второго может изменяться (регистр/стек/адрес памяти). Некоторые команды не требуют обоих аргументов. Для упрощения формата команды неиспользуемые аргументы становятся зарезервированными областями, что как правило также приводит к увеличению длины команды.

С учетом поставленных требований был разработан следующий формат команд:

Таблица 1. Битовое представление команды

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Биты | 31-28 | 27-26 | 25-24 | 23-20 | 19-16 | 15-0 |
| Поле | Код операции | Зарезерв. | Тип аргумента 2 | Зарезерв. | Аргумент 1 | Аргумент 2 |

Аргумент 1 представляет из себя номер регистра либо специальное значение (0xF) для вершины стека. Аргумент 2 может иметь следующие типы (которые также являются кодами поля типа аргумента 2):

* Тип 00 — прямая регистровая адресация, поле аргумента содержит номер регистра либо код вершины стека;
* Тип 01 — прямая адресация — поле аргумента содержит адрес ОЗУ, по которому располагаются данные;
* Тип 10 — непосредственная адресация. Решено добавит данный вид адресации, так как он позволяет упростить и ускорить процесс выборки аргументов в случае, если они не изменяются в ходе работы. В частности, при использовании данного способа можно увеличить скорость работы таких инструкций, как JMP или MOV, а так же сэкономить небольшой объем ОЗУ за счет ПЗУ (располагая аргументы прямо в команде).

Архитектура системы команд описана в таблице ниже.

Таблица 3. Архитектура системы команд

| Номер | Бинарный код | Символ | Описание | Наличие аргументов |
| --- | --- | --- | --- | --- |
| 1 | 0000 | HLT | Остановка выполнения | Нет |
| 2 | 0001 | NOP | Пустая операция | Нет |
| 3 | 0010 | MOV | Перемещение данных в регистр | 1, 2 |
| 4 | 0011 | MOVM | Перемещение данных в память | 1, 2 |
| 5 | 0100 | PUSH | Помещение данных в стек | 2 |
| 6 | 0101 | POP | Извлечение данных из стека | 1 |
| 7 | 0110 | JMP | Безусловный переход на инструкцию | 2 |
| 8 | 0111 | JNZ | Переход на инструкцию по отсутствию флага Z | 2 |
| 9 | 1000 | SRL | Логический сдвиг вправо | 1, 2 |
| 10 | 1001 | SUB | Вычитание беззнаковых целых чисел | 1, 2 |
| 11 | 1010 | AND | Бинарное И | 1, 2 |
| 12 | 1011 | NOTZ | Инверсия по флагу Z | 1 |

Преимуществами данной архитектуры являются простота формата команды, отсутствие необходимости в сложном блоке декодирования команд, простота загрузки команд из-за их фиксированного размера, возможность расширения.

Недостатком является большая избыточность (6 бит из 32 — зарезервированные), а также ограниченный набор используемых аргументов для команд.

## Описание взаимодействия всех блоков микро-ЭВМ при выполнении команд программы.

Для того чтобы устройство начало работать необходимо два условия:

1. Подача тактирующего сигнала на вход CLK;
2. П-образный сигнал на входе START.

При подаче П-образного сигнала устройство инициализирует необходимые узлы, а затем переходит к основному режиму работы. Данный режим состоит в попеременном чередовании двух этапов выполнения операций.

На первом этапе команда загружается из ПЗУ в регистр команд процессора IR. Адрес команды для загрузки указан в регистре адреса команды IP. На данном этапе блок загрузки инструкции выставляет нужные адреса на шину адреса ПЗУ, считывает байты команды в буфер, а затем записывает полную команду в регистр IR. Считывание одной команды требует 4 обращений к ПЗУ. Для минимизации повторных считываний команд можно использовать кэш.

На втором этапе три блока выполняют разные подэтапы одной команды — загрузка аргументов, выполнение, запись результатов. Примечательным является использование pull-based модели взаимодействия между блоками. В начале этапа устройство управления обращается к *последнему* блоку (записи) и ждет, пока он не выставит статус своей готовности в «1». Данный блок обращается к предыдущему блоку, ждет его готовности, считывает нужные данные, а затем выполняет свой этап команды. При этом предыдущий блок (исполнения) в свою очередь обращается к своему предыдущему блоку (чтения данных) и ждет загрузки аргументов.

На первом подэтапе блок загрузки данных загружает необходимые аргументы в соответствии с их типами, а затем передает их на второй этап. Для функционирования блок должен быть связан с блоком регистров IP/IR, блоком РОН, стеком и кэшем.

Второй подэтап — выполнение команды. На данном этапе над данными выполняется одна из операций заранее предопределенного набора. Как одна из операций, блок может просто пропускать данные на выход их блока.

Третьим подэтапом является запись данных в регистры/стек/память. По окончанию этапа в регистр адреса команды записывается новый адрес, который будет использован при загрузке команды в следующем цикле работы.

Каждый вышеописанный этап представляет собой отдельный логический блок, режим работы которого задается соседними блоками и выполняемой командой. В первую очередь, у блоков есть сигналы запроса и готовности, необходимые для реализации взаимодействия между блоками. Кроме того, блок выполняет действия согласно заранее заданному набору команд, определенная команда из которого выбирается после загрузки с помощью блока выборки команды из ПЗУ.

Блоки чтения и записи данных обращаются к ОЗУ через кэш с помощью асинхронного протокола. Кэш также имеет в своей основе автомат Мура. Строка кэша содержит в себе два 16-битных слова. Слова выравниваются по 1 биту (если считать от 0 справа).

# РАЗРАБОТКА ОСНОВНЫХ УСТРОЙСТВ МИКРО-ЭВМ

В данном разделе описываются и обосновываются решения принятые мною при проектировании всех имеющихся устройств микро-ЭВМ.

## Запоминающие устройства. Функциональный состав ПЗУ и ОЗУ.

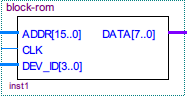


Рисунок 1. Условно-графическое обозначение блока ПЗУ.

Входы и выходы блока:

1. DEV\_ID — номер устройства. Данный вход устанавливает, на какие 4 старших бита адреса должна реагировать память. Это позволяет на одной шине адреса варьировать число блоков ПЗУ, что позволяет изменять объем постоянной памяти устройства.
2. ADDR — адрес, по которому происходит считывание данных
3. CLK — тактовый сигнал
4. DATA — вывод данных, которые располагаются по адресу ADDR

Блок ОЗУ представляет из себя синхронное ОЗУ со схемой управления, основанной на автомате Мура, с определенными состояниями простоя, принятия запроса, объявления готовности и завершения запроса.

Для совмещения адресных пространств ОЗУ и периферии старшие 4 бита адреса отдаются под номер устройства. Таким образом, можно, реконфигурируя аппаратное подключение ОЗУ/периферии к шине, задавать разные диапазоны адресов для разных блоков, а также расширять доступную память. Если к шине подключается устройство, которое может быть ведущим, необходимо подключать его через арбитр.

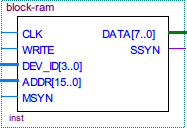


Рисунок 2. Условное графическое обозначение блока ОЗУ

Входы и выходы блока:

1. Сигнал CLK — сигнал тактовой частоты.
2. WRITE — если 1, то текущий запрос — на запись, иначе — на чтение.
3. ADDR, DATA и DEV\_ID аналогичны тем же выводам у ПЗУ.
4. MSYN/SSYN — сигналы синхронизации ведущего и ведомого устройства соответственно.

Для осуществления передачи данных ведущее устройство ждет, пока уровень SSYN не станет равным 0, затем выводит данные на вход ведомого устройства (на шину) и устанавливает MSYN=1. Далее ведомое устройство выполняет действия с данными, а ведущее ждет. Затеи ведомое устройство выведет данные на выходы и установит SSYN=1, это будет означать окончание обработки запроса. После этого ведущее устройство может снять MSYN, когда оно закончит запрос.

Здесь и далее основной принцип работы в асинхронном режиме соответствует вышеуказанному, названия выводов могут отличаться (например, REQ/RDY).

Блок регистров общего назначения состоит из 12 16-разрядных регистров. Обычно данные регистры используются для хранения временных результатов.

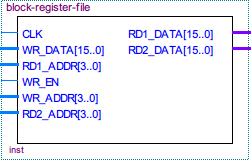


Рисунок 3. Условно-графическое обозначение блока РОН.

Блок регистров содержит два выхода для чтения данных и один вход для записи данных, что позволяет одновременно считывать два аргумента. Каждый регистр реализован на триггерах.

Входы и выходы блока:

1. CLK — тактовый сигнал
2. RD1\_DATA, RD2\_DATA — первый и второй выходы чтения
3. RD1\_ADDR, RD2\_ADDR — номера регистров первого и второго выходов чтения соответственно
4. WR\_DATA — вход данных для записи
5. WR\_ADDR — номер регистра для записи
6. WR\_EN — разрешение на запись.

Запись в регистр происходит по переднему фронту CLK при WR\_EN=1.

## Устройство управления

В данном проекте нельзя четко выделить в один блок логику управления микро-ЭВМ. Процедура выполнения команды разделена на функционально независимые блоки, каждый из которых управляется автоматом внутри него. Основным блоком управления микропроцессором является управляющий конечный автомат.

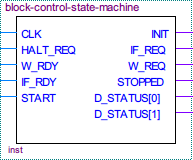


Рисунок 4. Условно-графическое обозначение управляющего конечного автомата.

Данный автомат координирует работу этапов выполнения команд, а также обеспечивает запуск и остановку выполнения.

Входы и выходы блока:

1. CLK — тактовый сигнал
2. START — сигнал запуска
3. HALT\_REQ — сигнал запроса остановки
4. INIT — выход инициализации, на данный выход при запуске микро-ЭВМ подается единичный сигнал, который сбрасывается в 0 при начале работы, что позволяет инициализировать различные блоки перед работой.
5. STOPPED — сигнал того, что процессор остановлен
6. IF\_REQ — выход запроса для блока загрузки инструкций
7. IF\_RDY — сигнал готовности блока загрузки инструкций
8. W\_REQ — выход запроса для блока записи — с данного запроса начинается выполнение загруженной команды.
9. W\_RDY — вход готовности блока записи, означает конец выполнения текущей инструкции.

Изначально блок находится в состоянии простоя. Когда на вход подается сигнал START, блок начинает последовательно вызывать сначала блок чтения инструкций из ПЗУ, а затем блок записи, что приведет к чтению аргументов, выполнению операции АЛУ, записи данных в память и обновлению регистра IP. В случае, если на вход поступает команда остановки, после выполнения инструкции блок переходит в состояние останова, из которого процесс выполнения может быть продолжен повторной отправкой сигнала START.

## Стек.

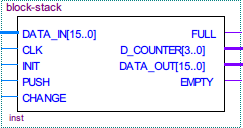


Рисунок 9. Условно-графическое обозначение стека.

Стек реализован как набор регистров. Для выбора нужного регистра используется счетчик, который необходимо инициализировать перед запуском выполнения.

Входы и выходы блока:

1. CLK — тактовый сигнал
2. INIT — вход сигнала инициализации, при установке в «1» внутренний счетчик стека сбрасывается на значение, соответствующее пустому стеку.
3. DATA\_IN — вход данных для записи в стек
4. DATA\_OUT — выход значения текущей вершины стека
5. CHANGE — сигнал проведения операции помещения в стек/изъятия из стека
6. PUSH — сигнал помещения в стек, если «1» — данные будут помещаться в стек, иначе — извлекаться из него.

При CHANGE установленном в «1» по тактовому сигналу стек будет увеличиваться/уменьшаться на 1 ячейку (в зависимости от PUSH).

## Арифметико-логическое устройство.

В состав АЛУ входят 2 регистра операндов, а также множество логических блоков, соответствующих определенной операции.

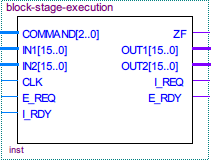


Рисунок 10. Условно-графическое обозначение АЛУ.

Входные и выходные сигналы:

1. CLK — тактовый сигнал
2. COMMAND — номер операции, которую должно выполнять АЛУ
3. IN1, IN2 — аргументы
4. E\_REQ — сигнал запроса операции АЛУ
5. E\_RDY — сигнал готовности операции АЛУ
6. I\_REQ — выход сигнала запроса загрузки данных от блока загрузки
7. I\_RDY — вход сигнала готовности от блока записи
8. ZF — значение флага Z
9. OUT1, OUT2 — выходные значения операции

Ниже представлен список номер операций:

Таблица 4. Cписок операций АЛУ.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер | АЛУ | Операнды | Код |
| 0 | Pass through | 2 | 000 |
| 1 | Shift Right Logical | 2 | 001 |
| 2 | Subtract | 2 | 010 |
| 3 | AND | 2 | 011 |
| 4 | Swap if Z | 1 | 100 |

АЛУ работает в качестве второй стадии выполнения команды в асинхронном режиме. Следует отметить операции Pass through (0) и Shift Right Logical (1). Pass through — отправка аргументов на выход без изменений, используется в таких командах, как MOV, JMP и прочие, не требующие вычислений, но использующие аргументы для записи данных. Операция Shift Right Logical реализована методом barrel shifter, что позволяет ей выполняться за 1 такт за счет увеличения логической сложности. Код операции определяется командным блоком после загрузки команды.

## Кэш данных

Используется кэш со сквозной записью с отображением с коэффициентом ассоциативности 2. Одна строка кэша содержит 2 16-битных слова.

Кэш данных управляется набором конечных автоматов. Основной принцип работы заключается в том, что главный автомат кэша определяет ситуацию, в которой находится кэш (простаивание, чтение/запись, попадание/промах) и при необходимости вызывает автомат, реализующий алгоритм для работы с нужным состоянием.

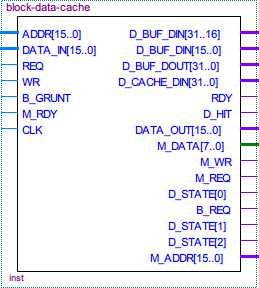


Рисунок 11. Условное графическое обозначение кэша

Входы и выходы блока:

1. CLK — тактовый сигнал
2. ADDR — запрашиваемый адрес памяти
3. DATA\_IN — данные для записи в память
4. DATA\_OUT — вывод данных кэша
5. WR — флаг выполнения записи в память
6. REQ — запрос данных из кэша/записи в кэш
7. RDY — сигнал готовности кэша
8. B\_REQ, B\_GRUNT — сигналы запроса и предоставления шины
9. MEM\_REQ, MEM\_RDY — сигналы запроса памяти и ее готовности
10. M\_WR — флаг записи в память
11. M\_DATA — вход/выход для данных с шины, к которой подключена ОЗУ
12. M\_ADDR — запрашиваемый адрес данных, выставляемый на шину
13. Выводы с префиксом «D\_» — отладочные выводы

Для чтения из памяти или записи в нее используются два соответствующих блока внутри кэша, которые отвечают за обращения к арбитру для доступа к шине и памяти.

# ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ

В данном разделе приведены временные диаграммы функционального моделирования отдельных модулей и всего устройства в целом.

## Блок ПЗУ

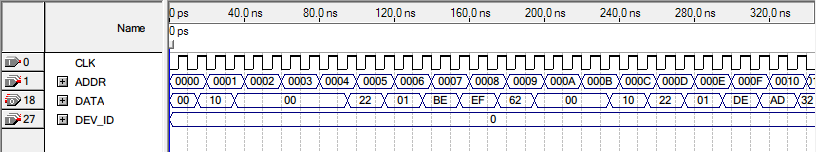


Рисунок 12. Функциональное моделирование ПЗУ

В ПЗУ по адресу 0x0004 располагается инструкция 0x2201BEEF. Из диаграммы функционального моделирования видно, что между выставлением адреса и получением данных присутствует задержка в 1,5 такта. Это связано с тем, что блок lpm\_ram\_dq защелкивает в буфер адрес по фронту тактового сигнала, а затем через 1 такт выставляет данные. Таким образом, между выставлением адреса на шину и считыванием данных должно проходить 2 такта (при условии, что и выставление адреса, и запись данных в буфер происходят по спаду тактового сигнала).

## Блок ОЗУ

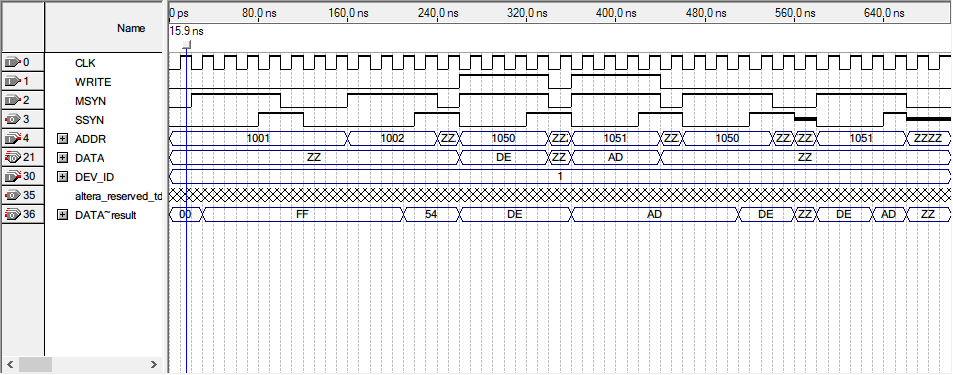


Рисунок 13. Функциональное моделирование ОЗУ

Из диаграммы функционального моделирования видно, как работает асинхронный протокол обмена. Сигналы ADDR и WRITE выставляются ведущим устройством одновременно с выставлением MSYN в «1», так как фактическое обновление состояния конченого автомата, управляющего работой ОЗУ, происходит по фронту тактового сигнала, а все обновления выходных сигналов происходят по спаду тактового сигнала.

Так как старшие 4 бита шины адреса определяют подключенное устройство, на диаграмме видны моменты, где отсутствие старших 4 бит адреса отключает ОЗУ от шины, что приводит к неопределенному уровню сигнала.

Средняя задержка между запросом данных из памяти и готовностью памяти составляет 4 такта.

## Регистры общего назначения

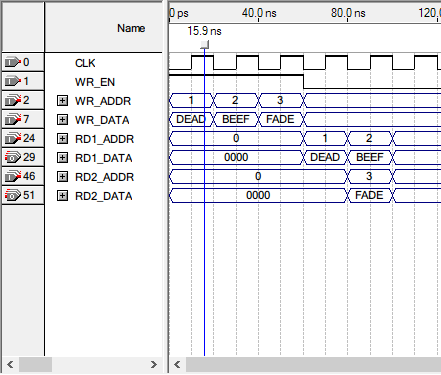


Рисунок 14. Функциональное моделирование блока РОН.

В течение первых трех тактов в регистры записываются 2-байтовые значение, далее они считываются. Так как у блока регистров есть два порта чтения данных, демонстрируется одновременное чтение сразу из обоих портов.

* 1. Стек

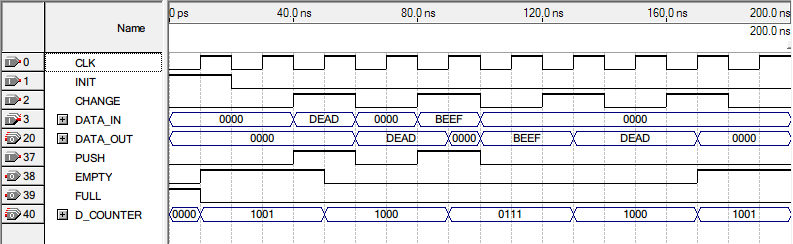


Рисунок 13. Функциональное моделирование стека.

Из диаграммы функционального моделирования можно видеть, как после

* 1. Арифметико-логическое устройство.

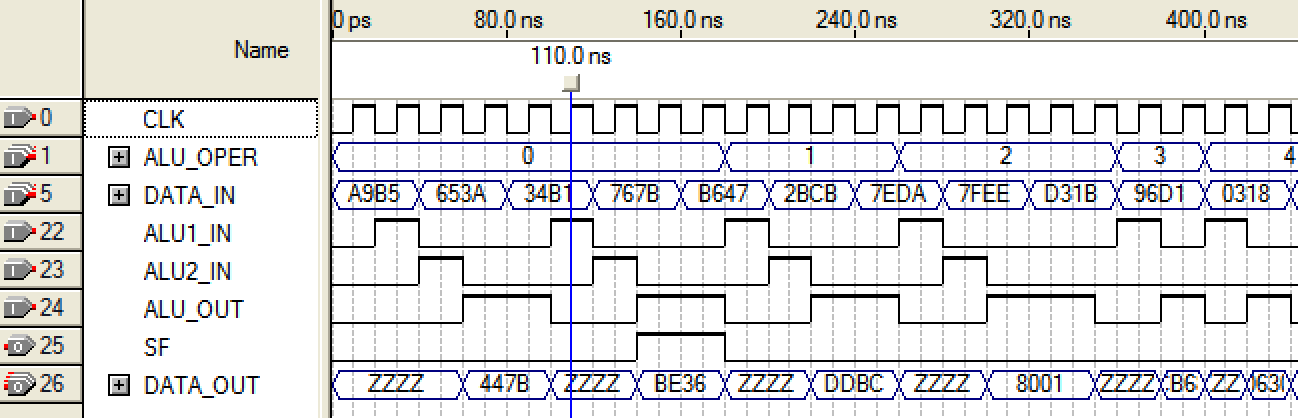


Рисунок 14. Функциональное моделирование АЛУ.

На интервале 0-180 дважды проверяется команда SUB. В первом случае очевидно первый операнд больше второго и флаг SF не выставляется, а во втором ситуация противоположная и мы можем заметить выставленный флаг. Далее по очереди проверяется каждая из 5 команд АЛУ.

* 1. Тестовая программа.

Для нормального функционирования устройства нам необходимо заполнить наши ПЗУ и ОЗУ соответствующими данными. Ниже представлены данные, хранящиеся в устройстве.

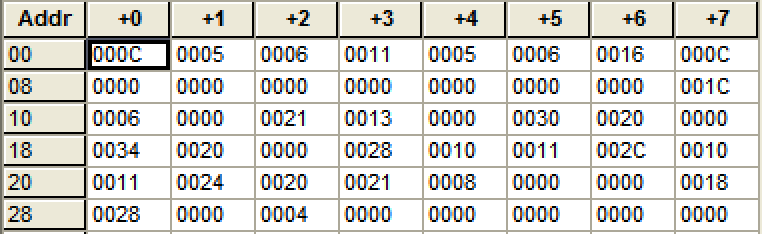


Рисунок 15. Содержимое ПЗУ команд.

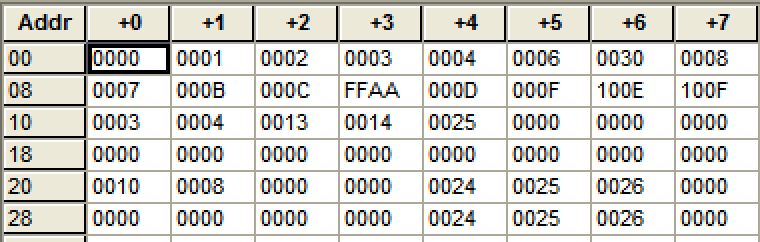


Рисунок 16. Содержимое ПЗУ данных.

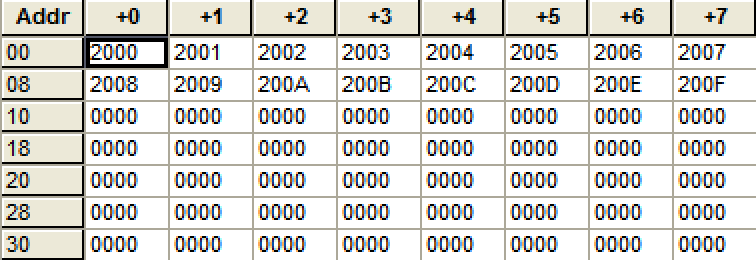


Рисунок 17. Содержимое ОЗУ (до моделирования).

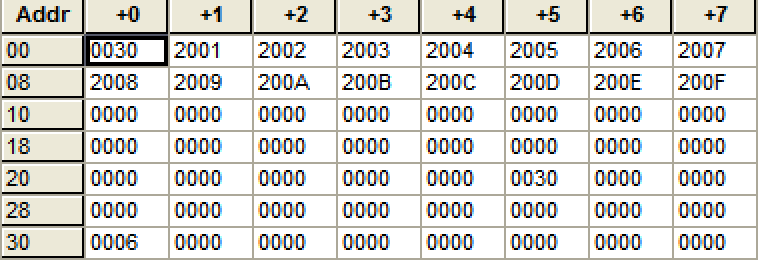


Рисунок 18. Содержимое ОЗУ (после моделирования).

Таблица 5. Тестовая программа.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Комманда | Адресация | Оп.1 | Оп.2 | Hex | Hex | Hex |
| 1 | MOV adr, reg | Прямая | 5 | 6 | 000C | 0005 | 0006 |
| 2 | MOV reg, adr | Косвенная | 5 | 6 | 0011 | 0005 | 0006 |
| 3 | JMP adr | Косвенная x2 | 12 | 0 | 0016 | 000C | 0000 |
| 4 | PUSH adr | Прямая | 6 | 0 | 001C | 0006 | 0000 |
| 5 | POP adr | Косвенная | 19 | 0 | 0021 | 0013 | 0000 |
| 6 | ROL reg | Прямая | 32 | 0 | 0030 | 0020 | 0000 |
| 7 | ROR reg | Прямая | 32 | 0 | 0034 | 0020 | 0000 |
| 8 | NAND reg, reg | Прямая | 16 | 17 | 0028 | 0010 | 0011 |
| 9 | NOR reg, reg | Прямая | 16 | 17 | 002C | 0010 | 0011 |
| 10 | SUB reg, reg | Прямая | 32 | 33 | 0024 | 0020 | 0021 |
| 11 | NOP | Прямая | 0 | 0 | 0008 | 0000 | 0000 |
| 12 | JMS adr | Прямая | 40 | 0 | 0018 | 0028 | 0000 |
| 13 | HLT | Прямая | 0 | 0 | 0004 | 0000 | 0000 |

Проанализируем работу программы. Первые 2 команды последовательно копирую сначала значения из памяти в регистра, потом из регистра в памяти. Результаты записи в память можно видеть на рисунке 18. Потом осуществляется безусловный переход. Далее проверяется работа стека, путем помещения туда значения и считывания его оттуда. Далее проверяется работа каждой из команд АЛУ. Далее выполняется команда ожидания и условного перехода. Если условие условного перехода будет выполнено ( флаг S=1 ) тогда программа начнет свое выполнение заново, если нет – то устройство остановит свою работу.

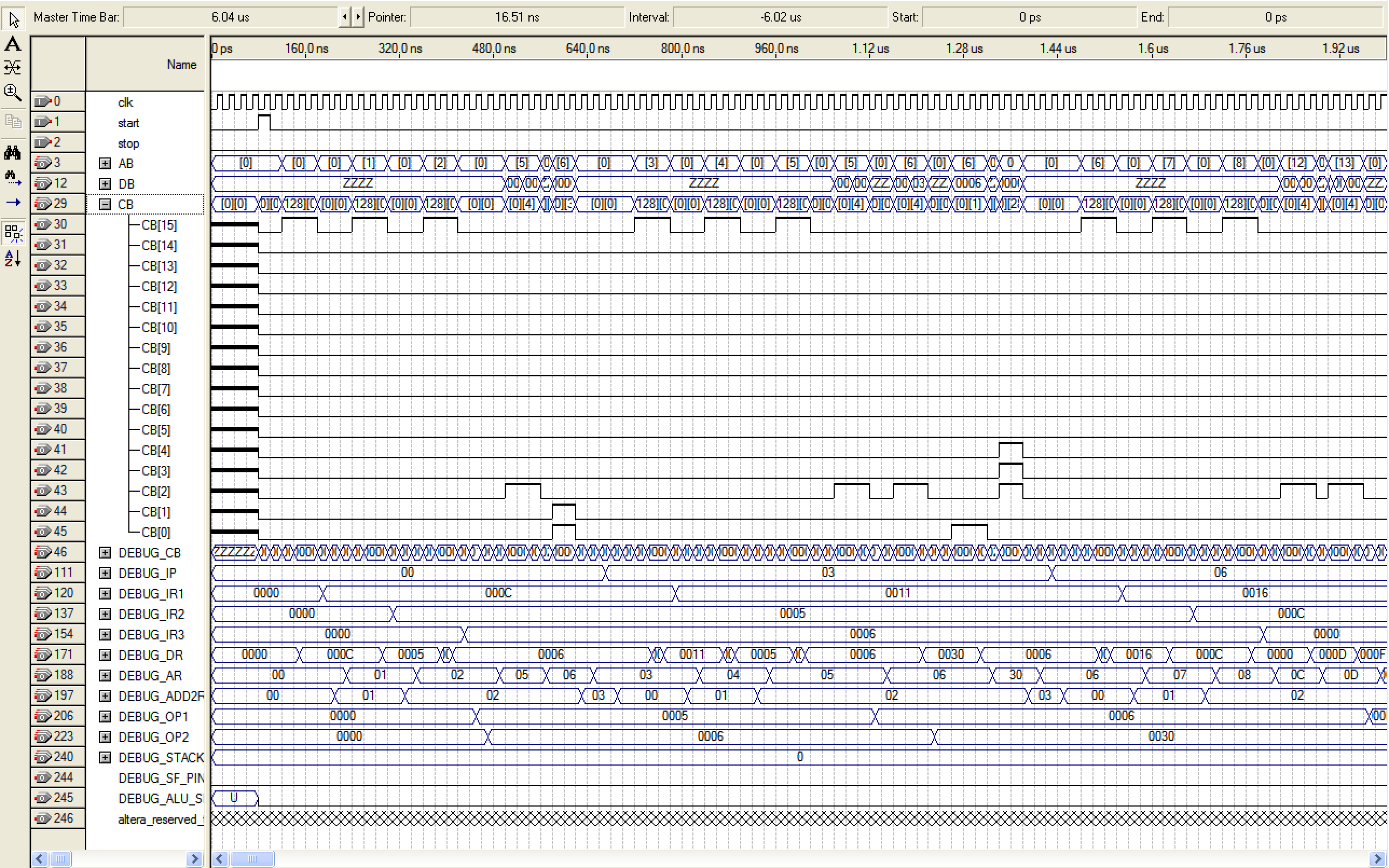


Рисунок 19. Функциональное моделирование работы микро-ЭВМ.

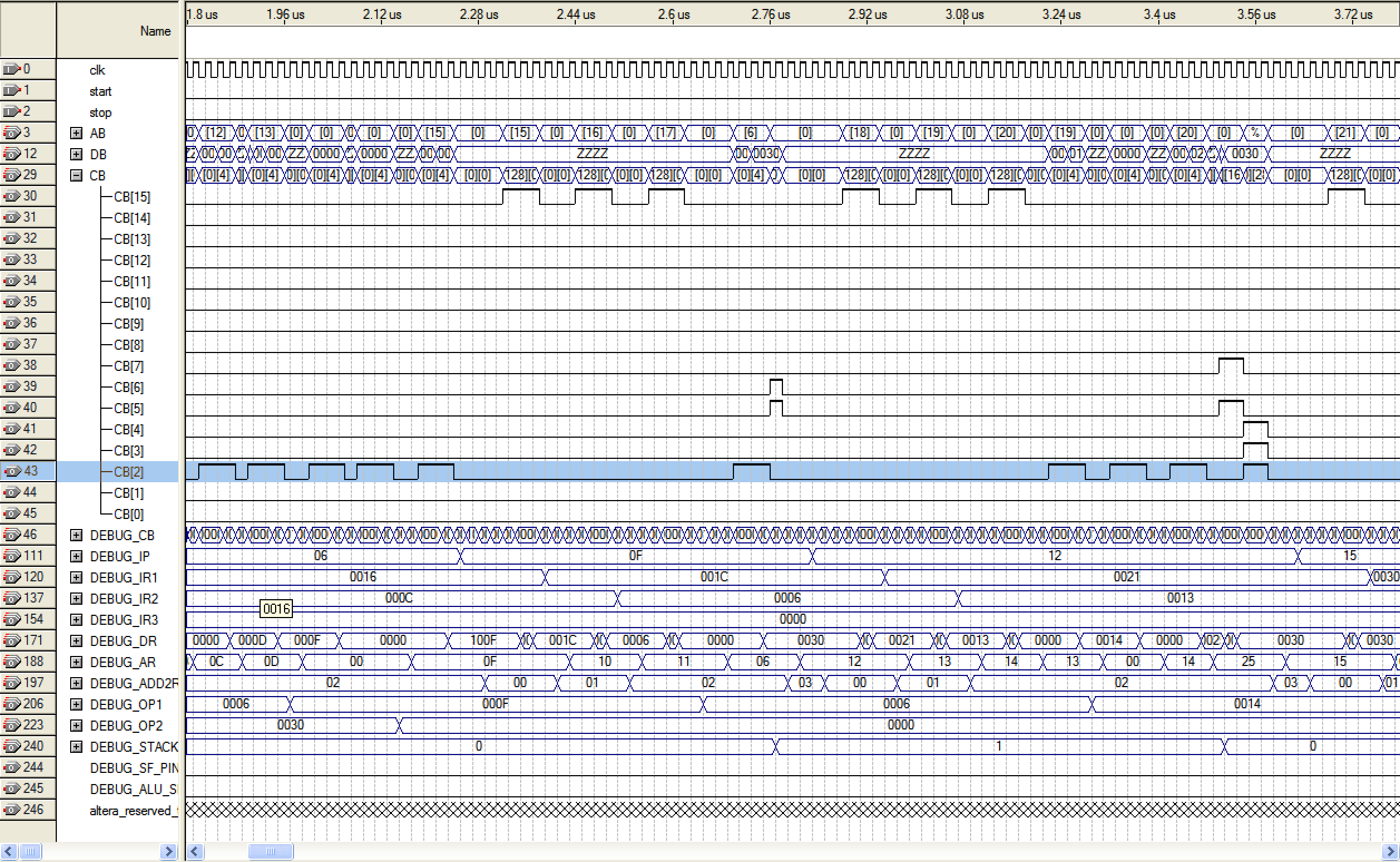


Рисунок 20. Функциональное моделирование работы микро-ЭВМ(продолжение).

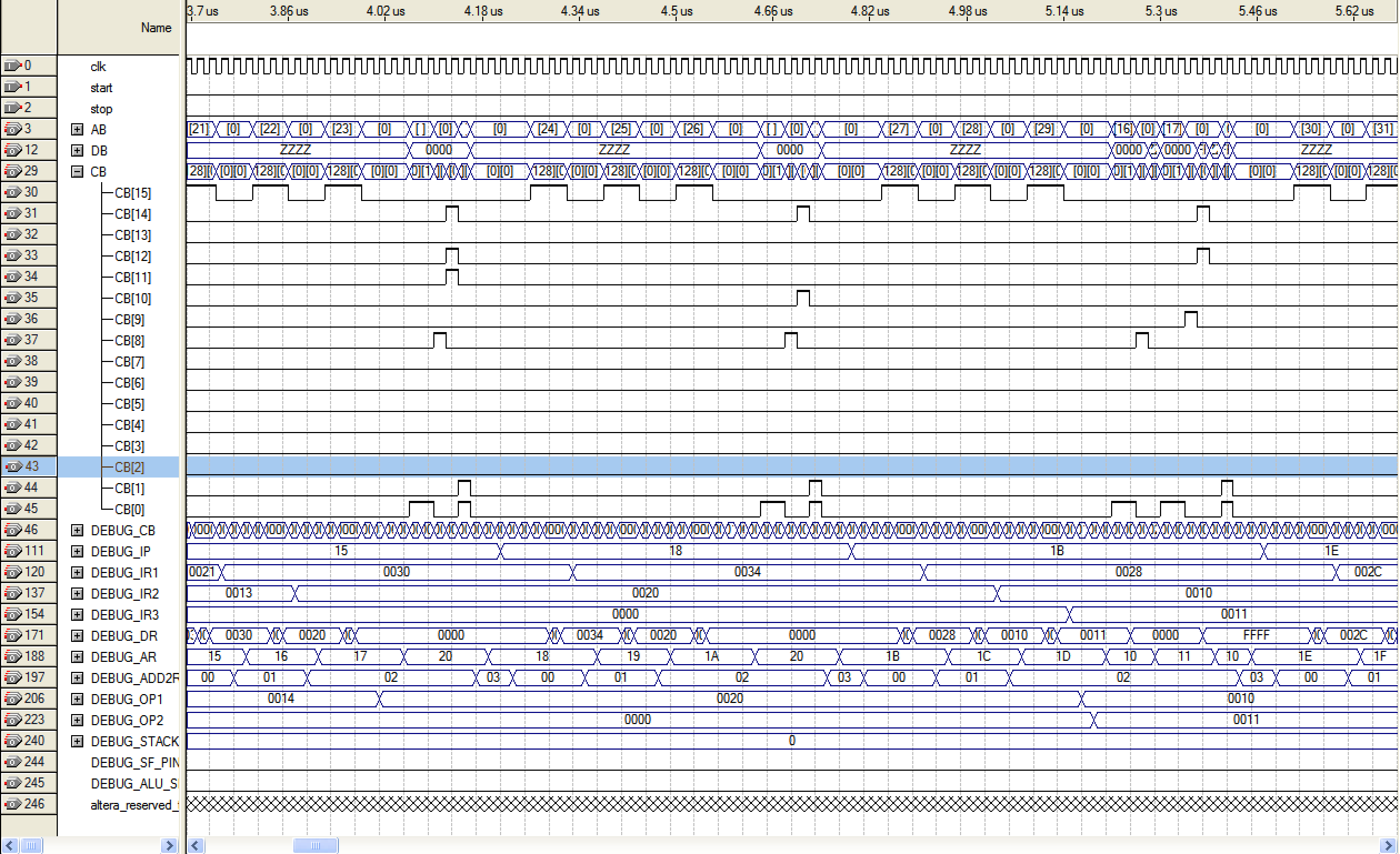


Рисунок 21. Функциональное моделирование работы микро-ЭВМ(продолжение).

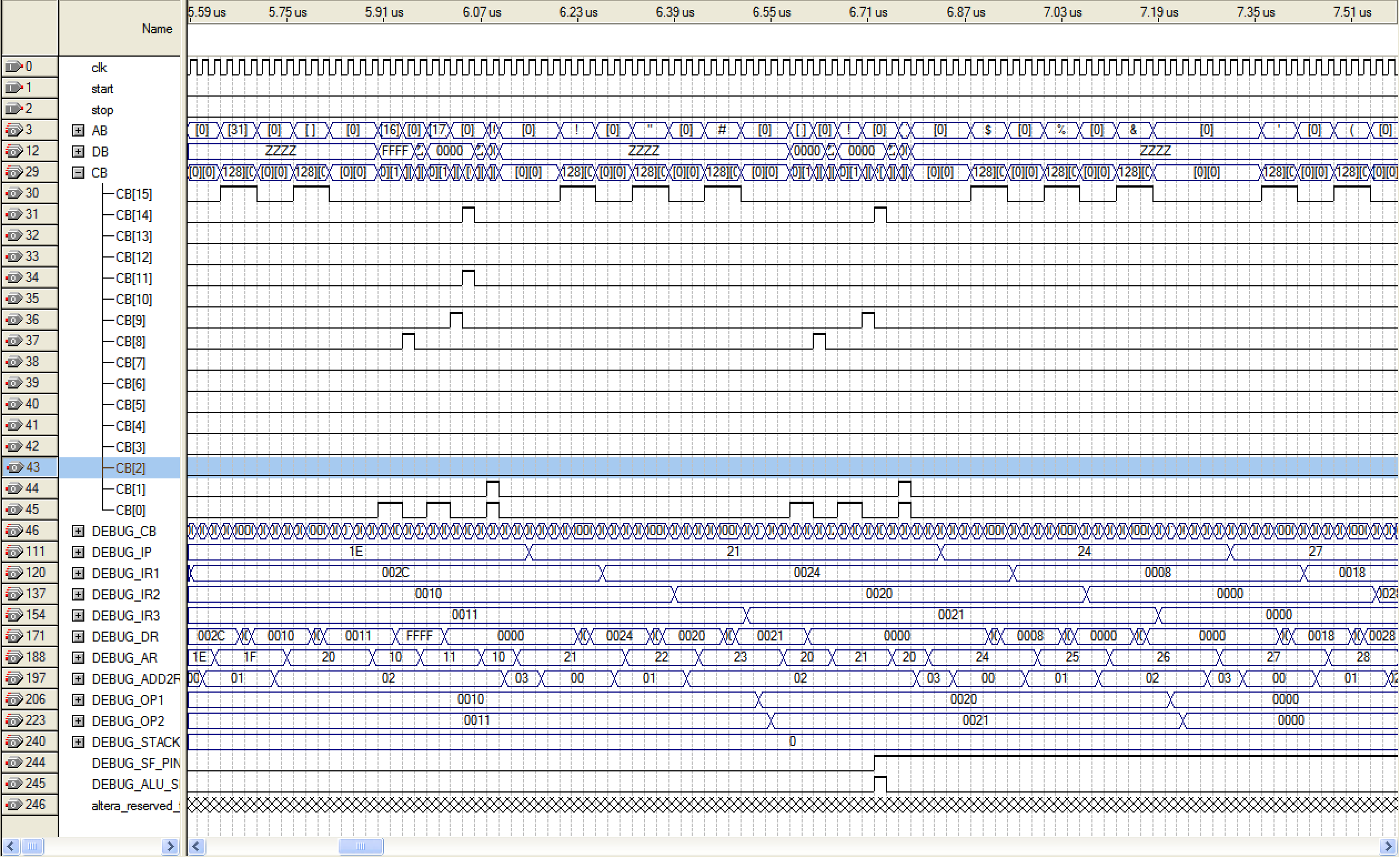


Рисунок 22. Функциональное моделирование работы микро-ЭВМ(продолжение).

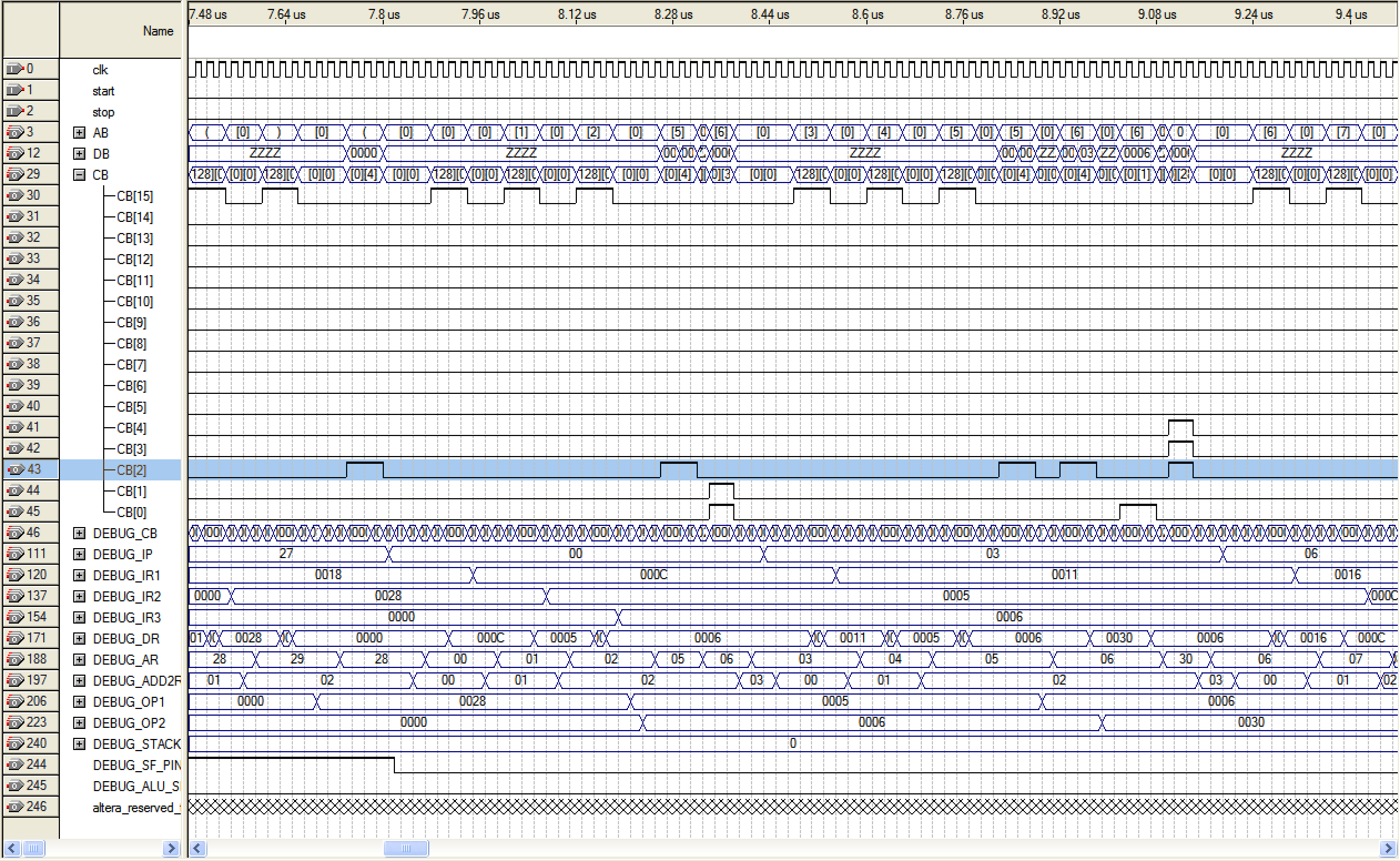


Рисунок 23. Функциональное моделирование работы микро-ЭВМ(продолжение).

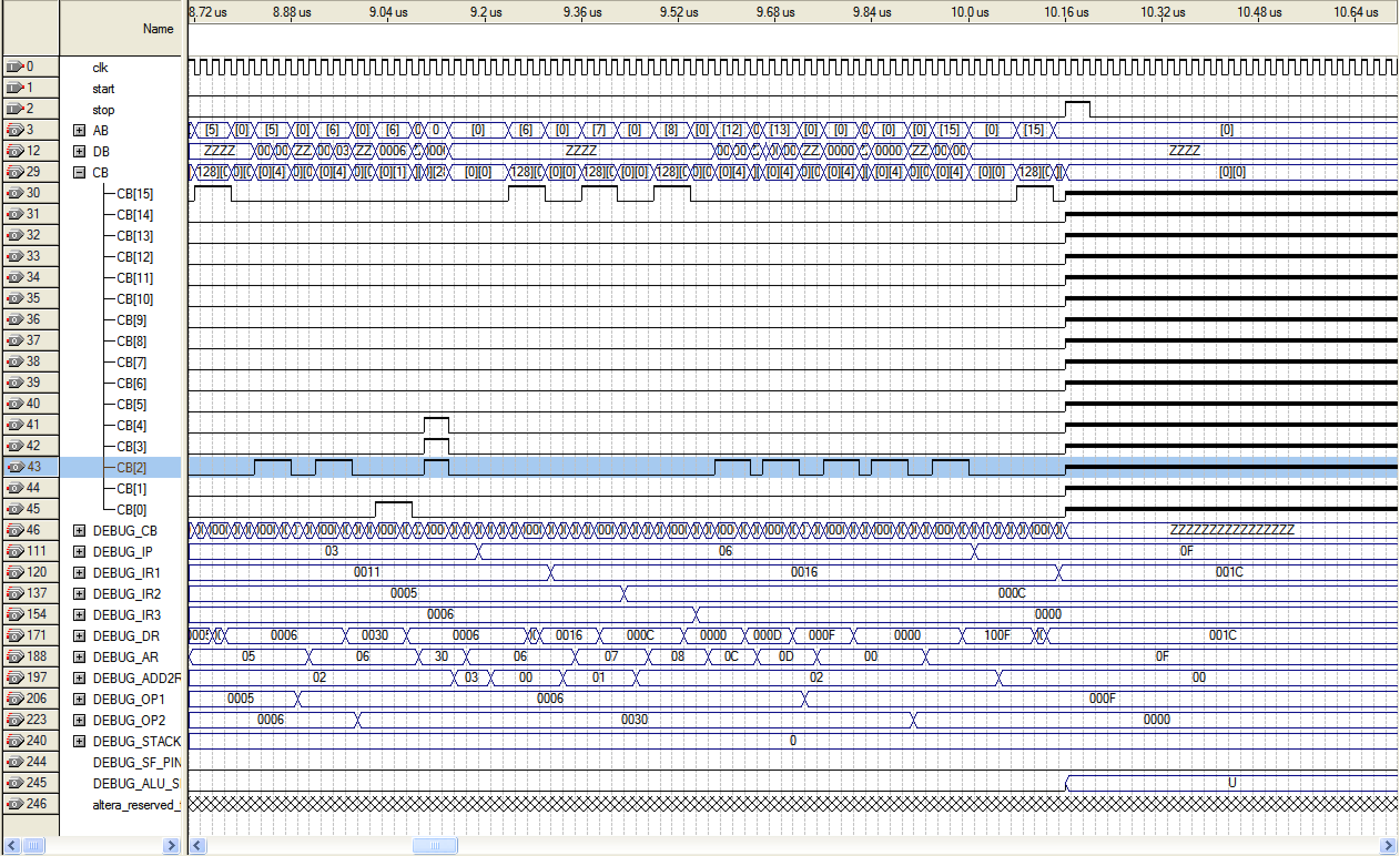


Рисунок 24. Функциональное моделирование работы микро-ЭВМ(продолжение).

1. АНАЛИЗ И ОПТИМИЗАЦИЯ РАЗРАБОТАННОЙ МИКРО-ЭВМ

В ходе работы было создано микро-ЭВМ, способное выполнять 13 различных команд, с возможностью выбора одного из 3 предложенных способов адресации.

Как видно из диаграмм выше самые затратные стадии выполнение каждой из команд – это стадия загрузки команды и стадия загрузки операндов. Стадия загрузки команды занимает в среднем 20 тактов, а стадия загрузки операндов занимает от 2 до 19 тактов в зависимости.

В целом самая длинная команда MOV adr, reg при выборе косвенной адресации длится 49 тактов, а самая короткая JMP adr при выборе прямой адресации 28 тактов.

Как видно из вышеперечисленного пространство для оптимизации в данном устройстве имеется с лихвой. Давайте рассмотрим некоторые шаги, которые могу ускорить работу данного микро-ЭВМ:

* **Использование переменной длины команд**. Как видно самым длительными является этап выборки команды из памяти. Все дело в том что при проектировании нами была выбрана фиксированная длина команд, в связи чем мы вынуждены трижды грузить из памяти блок в 16 бит. Что является не оптимальным решением, так как многие команды не требует 2 операндов или вовсе не требуют операндов. Использование переменной длины команд могло бы сократить количество необходимых тактов до 12 и даже до 6 тактов. Однако стоит иметь ввиду что это значительно усложнит внутреннюю структуру устройства управления.
* **Использование кэш-памяти.** Как известно обращение к памяти является самой медленной операцией. Ввод в микро-ЭВМ кэша помог бы значительно ускорить выполнение операций чтения и записи. Стоит отметить что из-за гарвардской архитектуры в нашем микро-ЭВМ присутствует два ПЗУ, однако использование кэш-памяти для ПЗУ команд не имеет смысла, так как обычно считывание команд происходит последовательно, исключая разве что случаи при выполнении команд условного и безусловного перехода.
* **Использование конвейера команд.** Сейчас при работе устройства все инструкции выполняются последовательно, таким образом многие устройства простаивают большую часть времени. Ввод конвейера частично решил бы данную проблему, однако потребовал бы значительного усложнения структуры микро-ЭВМ.
* **Совмещение сигнала конца команды с предыдущим сигналом.** Дело в том что сейчас каждый этап оканчивается специальной командой STEP\_FINISHED на который тратится дополнительный такт. Так как устройство содержит 4 этап, то целых 4 такта процессорного времени тратятся на простой переход между фазами.
* **Использование контроллера прямого доступа в памяти.** В данной архитектуре системы команда нету команды которая копирует значение из ячейки памяти в другую ячейку памяти. Для этого приходится использовать 2 команды: MOV adr, reg и MOV reg, adr. Это в свою очередь займет порядка 64 тактов. Данную проблему можно решить путем ввода дополнительной команды MOV adr, adr или же вводом контроллера прямого доступа в память. Он позволит разгрузить устройство управления.

ЗАКЛЮЧЕНИЕ

В ходе работы удалось реализовать полноценную микро-ЭВМ. При разработке возникало множество конфликтов и задач, к которым приходилось находить творческий подход. Конечно реализованная микро-ЭВМ не может конкурировать на рынку существующих, однако она с достоинством может занять место в качестве обучающего пособия для абитуриентов и студентов, увлекающимися проектированием ЭВМ. Еще раз перечислим основные характеристики:

1. Гарвардская архитектура;
2. 2 ПЗУ объемом 256 слов;
3. ОЗУ объемом 256 слов;
4. 16 регистров общего назначения;
5. Стек, размером 6 слов;
6. 13 различных команд.

Также в архитектуру ЭВМ были заложены возможности расширения, что позволяет продолжить работу над устройством в дальнейшем.