Федеральное государственное автономное образовательное учреждение высшего образования "Национальный Исследовательский Университет ИТМО"
Мегафакультет Компьютерных Технологий и Управления
Факультет Программной Инженерии и Компьютерной Техники

Вариант №4
Лабораторная работа 4
по дисциплине
'Функциональная схемотехника'

Выполнил Студент группы P33102
Лапин Алексей Александрович
Преподаватель:
Васильев С.Е.

г. Санкт-Петербург
2024г.

Содержание

Задание

В лабораторной работе вам предлагается разобраться во внутреннем устройстве простейшего процессорного ядра архитектуры RISC-V. Результатом изучения микроархитектуры процессорного ядра и системы команд RISC-V станут ваши функциональные и нефункциональные модификации ядра.

Основное задание:

1. Модифицировать процессорное ядро, в соответствии с вашим вариантом;
2. Подготовить тестовое окружение системного уровня и убедиться в корректности вашей реализации путём запуска симуляционных тестов.

Примечание: При непосредственном описании ваших модификаций в коде проекта, запрещено использовать симуляционные конструкции и арифметические операции, отличные от сложения и вычитания (то есть, умножение, деление и возведение в степень реализуйте сами посредством описания любого, понравившегося вам, алгоритма). В тестовом окружении использовать симуляционные конструкции и всевозможные арифметические операции можно (и даже нужно).

Варианты с буффером/очередью должны реализовать две команды.

- Первая команда - push xN - должна загружать данные из младшей части регистра xN в буффер/очередь.
- Вторая команда - pop xN - должна выгружать данные в младшую часть регистра xN.

Микроархитектурная схема модифицированного процессорного ядра

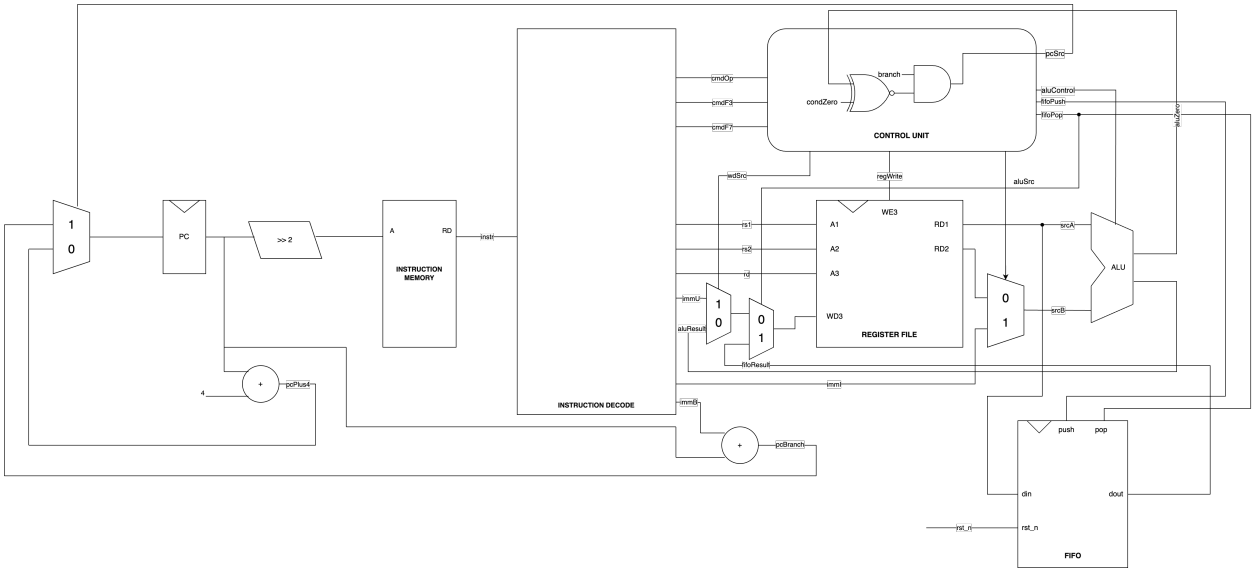


Рис. 1: Микроархитектурная схема процессорного ядра

FIFO - добавленный модуль

Описание работы модуля FIFO

1. Входные порты

- `clk` - тактовый сигнал.
- `rst_n` - асинхронный сигнал сброса (активный низкий).
- `din` - входные данные шириной `DATA_WIDTH` бит.
- `push` - сигнал записи данных в FIFO.
- `pop` - сигнал чтения данных из FIFO.

2. Описание работы модуля

Модуль FIFO (First-In-First-Out) реализует очередь с параметризуемой шириной данных (`DATA_WIDTH`) и глубиной (`PTR_WIDTH`). Основные компоненты модуля включают указатели чтения и записи (`rd_ptr` и `wr_ptr`), память FIFO (`mem`) и счетчик элементов (`count`).

- При активном сигнале сброса (`rst_n` = 0), указатели чтения и записи, а также счетчик элементов устанавливаются в ноль.
- При активном сигнале `push` и отсутствии переполнения (`full` = 0), данные из `din` записываются в память FIFO по адресу `wr_ptr`, указатель записи увеличивается на 1, и счетчик элементов увеличивается на 1.
- При активном сигнале `pop` и отсутствии сигнала empty (`empty` = 0), указатель чтения увеличивается на 1, и счетчик элементов уменьшается на 1.
- Выходные данные `dout` обновляются при активном сигнале `pop` и отсутствии сигнала empty(`empty` = 0), принимая значение из памяти FIFO по адресу `rd_ptr`.

3. Выходные порты

- `dout` - выходные данные шириной `DATA_WIDTH` бит.

Добавленные управляющие сигналы и их роли

1. fifoPush

- **Роль**: Этот сигнал управляет операцией записи (push) в FIFO. Если `fifoPush` установлен, данные из регистра `rd1` записываются в FIFO.
- **Использование**: В модуле `sr_control` сигнал `fifoPush` активируется при выполнении команды `RVOP_PUSH`.

2. fifoPop

- **Роль**:
  - Этот сигнал управляет операцией чтения (pop) из FIFO. Если `fifoPop` установлен, данные из FIFO читаются и передаются на выход.
  - Этот сигнал определяет источник данных для записи в регистр. Если `fifoPop` установлен, данные для записи берутся из FIFO, иначе используются данные из ALU или непосредственное значение.
- **Использование**:
  - В модуле `sr_control` сигнал `fifoPop` активируется при выполнении команды `RVOP_POP`.
  - В модуле `sr_cpu` сигнал `fifoPop` используется для выбора между результатом FIFO (`fifoResult`) и другими источниками данных (`immU` или `aluResult`).

Изменения в модулях

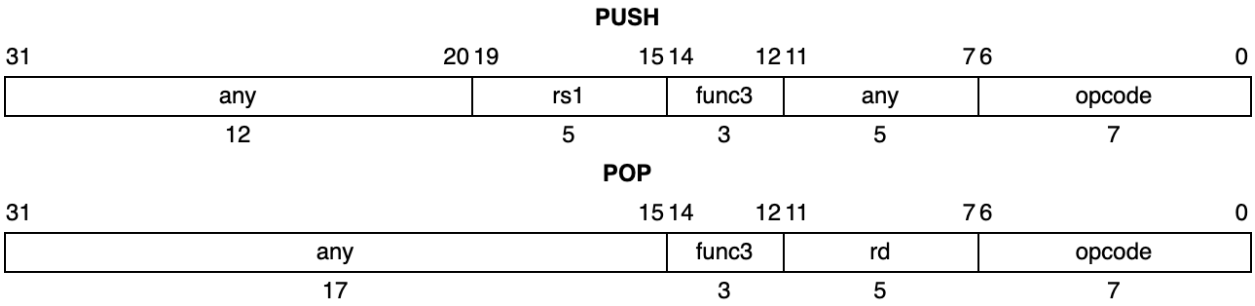
Модуль sr_cpu

- Добавлены новые управляющие сигналы `fifoPush`, `fifoPop`.
- Добавлен новый модуль `fifo` для работы с FIFO.
- Изменен сигнал `wd3` для учета нового источника данных `fifoResult`.

Модуль sr_control

- Добавлены новые управляющие сигналы `fifoPush`, `fifoPop`.
- Добавлены новые команды `RVOP_PUSH` и `RVOP_POP` для управления FIFO.

Описание добавленных форматов и инструкций



Формат инструкции push

Инструкция ‘push‘ используется для сохранения значения регистра в стек.

- **any** (12 бит): Поле, которое может содержать любое значение.

- **rs1** (5 бит): Исходный регистр, значение которого сохраняется в стек.
- **func3** (3 бит): Функциональный код, определяющий тип операции. В данном случае это ‘010‘. (sw)
- **any** (5 бит): Любые символы, так как результат не сохраняется в регистр.
- **opcode** (7 бит): Код операции. В данном случае это ‘0100011‘, что соответствует инструкции загрузки (load) (мы переиспользуем эту инструкцию).

Формат инструкции pop

Инструкция ‘pop‘ используется для извлечения значения из стека и сохранения его в регистр.

- **any** (17 бит): Поле, которое может содержать любое значение.
- **func3** (3 бит): Функциональный код, определяющий тип операции. В данном случае это ‘010‘. (lw)
- **rd** (5 бит): Целевой регистр, в который сохраняется извлеченное значение.
- **opcode** (7 бит): Код операции. В данном случае это ‘0000011‘, что соответствует инструкции загрузки (load) (мы переиспользуем эту инструкцию).

Результаты из отчёта по временным параметрам

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7,323 ns	Worst Hold Slack (WHS):	0,185 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	92	Total Number of Endpoints:	92	Total Number of Endpoints:	71

All user specified timing constraints are met.

Рис. 2: Отчет по временным параметрам

- **Worst Negative Slack (WNS)**:
  - **Setup**: 7,323 ns
  - **Hold**: 0,185 ns
  - **Pulse Width**: 4,500 ns

  Описание:
  - **Worst Negative Slack (WNS)** показывает наибольшее отрицательное отклонение времени задержки (slack) для каждого типа временных проверок (setup, hold, pulse width). Отрицательное значение указывает на то, что временные ограничения не выполнены.

  - **Setup**: Время задержки между тактовыми сигналами, необходимое для корректной работы схемы.
  - **Hold**: Минимальное время, в течение которого сигнал должен оставаться стабильным после тактового сигнала.
  - **Pulse Width**: Минимальная ширина импульса тактового сигнала.

- **Total Negative Slack (TNS)**:
  - **Setup**: 0,000 ns
  - **Hold**: 0,000 ns
  - **Pulse Width**: 0,000 ns

  Описание:

  - **Total Negative Slack (TNS)** показывает суммарное отрицательное отклонение времени задержки для всех конечных точек (endpoints). Значение 0 указывает на то, что все временные ограничения выполнены.

  OFFTOP: WNS vs TNS

  Worst Negative Slack (WNS)

  - WNS измеряет самый большой отрицательный запас времени среди всех таймингових путей в схеме. Это значение показывает, насколько самый плохой путь превышает заданное временное ограничение.
  - Если WNS равен 2 наносекунды, это означает, что самый критический путь в схеме опаздывает на 2 наносекунды относительно его временного ограничения.

  Total Negative Slack (TNS)

  - TNS суммирует отрицательные запасы времени всех таймингових путей в схеме, которые не соответствуют временным требованиям. Это дает общее представление о том, насколько велика общая задержка по всем путям, которые не укладываются в расписание.
  - Если TNS равен 10 наносекунд, это означает, что сумма всех отрицательных запасов времени по всем путям составляет 10 наносекунд.

- **Number of Failing Endpoints**:
  - **Setup**: 0
  - **Hold**: 0
  - **Pulse Width**: 0

  Описание:

  - **Number of Failing Endpoints** показывает количество конечных точек, для которых временные ограничения не выполнены. Значение 0 указывает на то, что все конечные точки соответствуют временным ограничениям.

- **Total Number of Endpoints**:

- **Setup**: 92
- **Hold**: 92
- **Pulse Width**: 71

Описание:

- **Total Number of Endpoints** показывает общее количество конечных точек, для которых проводились временные проверки.

Выводы из отчета

1. **Worst Negative Slack (WNS)**:
   - Значение WNS для setup времени составляет 7,323 ns, что указывает на значительное отрицательное отклонение. Это может быть проблемой, так как указывает на то, что временные ограничения для setup времени не выполнены.
   - Значение WNS для hold времени составляет 0,185 ns, что также является отрицательным, но менее критичным.
   - Значение WNS для pulse width составляет 4,500 ns, что также указывает на нарушение временных ограничений.
2. **Total Negative Slack (TNS)**:
   - Все значения TNS равны 0, что указывает на то, что в сумме все временные ограничения выполнены.
3. **Number of Failing Endpoints**:
   - Количество конечных точек, для которых временные ограничения не выполнены, равно 0 для всех типов проверок. Это хороший знак, указывающий на то, что все конечные точки соответствуют временным ограничениям.
4. **Total Number of Endpoints**:
   - Общее количество конечных точек для setup и hold проверок составляет 92, а для pulse width — 71. Это количество конечных точек, для которых проводились временные проверки.

Итог

Несмотря на то, что Worst Negative Slack (WNS) показывает отрицательные значения для setup, hold и pulse width, общее количество конечных точек, для которых временные ограничения не выполнены, равно 0. Это указывает на то, что все конечные точки соответствуют временным ограничениям, и проект в целом соответствует временным требованиям. Однако, стоит обратить внимание на значительное отрицательное отклонение для setup времени и рассмотреть возможность оптимизации схемы для улучшения временных характеристик.

Выводы по работе

Анализ архитектурных решений

2xMux 2-to-1 vs Mux 4-to-1

При реализации выбора источника данных для записи в регистр можно было пойти двумя путями:

1. Сделать четырех входной мультиплексор и расширить сигнал wdSrc до двух бит, что обеспечивает возможность выбора из четырех источников данных.

2. Сделать два мультиплексора с двумя входами и два сигнала оправления ими. Это обеспечивает возможность выбора из трех источников данных.

В данной работе был выбран второй вариант.
Достоинства:

- **Избежание лишнего состояния:**

  В 4-разрядном мультиплексоре одно состояние будет лишним, что может привести к неопределенному поведению или необходимости дополнительной логики для обработки этого состояния. Два 2-разрядных мультиплексора позволяют избежать этой проблемы.

- **Более гибкая конфигурация** При необходимости у нас есть отдельные сигналы, которые показывают, "работает"или нет алу и также для fifo.

Недостатки:

- **Усложнение схемы:**

  Требуется два разных сигнала выбора, заместо одного. Это усложняет проектирование схемы.

- **Потенциальное увеличение стоимости**

  Так как требуется больше мультиплексоров и проводов, это может привести к увеличению стоимости.

- **Уменьшение скорости работы**

  Использование двух мультиплексоров может увеличить задержку сигнала, так как сигнал должен пройти через два устройства вместо одного.

- **Потенциальное увеличение потребления энергии** Два устройства могут потреблять больше энергии по сравнению с одним

При проектировании для меняя важнее было избежать лишнего состояния, поэтому был выбран второй вариант.

fifoSign vs fifoPush и fifoPop

Мы могли бы обединить сигналы fifoPush и fifoPop в один сигнал двух битный fifoSign, который бы определял направление данных в FIFO. Однако это привило бы к значительному усложнению логики управления.

Использование fifoSrc

Заместо использования fifoPop для двух ролей:

- Выбора источника данных для записи в регистр.
- Определения направления данных в FIFO.

Мы могли бы использовать отдельный сигнал fifoSrc для выбора источника данных для записи в регистр и fifoPop для определения направления данных в FIFO.

Достоинства fifoSrc:

- **Улучшенная читаемость и уменьшение риска ошибок в коде:**

  Разделение сигналов на ‘fifoSrc‘ и ‘fifoPop‘ делает код более понятным и легким для чтения. Это может упростить процесс отладки. Также разделение функций на разные сигналы уменьшает вероятность ошибок, связанных с неправильной интерпретацией одного сигнала, выполняющего несколько ролей.

- **Улучшенная гибкость:**

  Использование отдельного сигнала ‘fifoSrc‘ позволяет более гибко управлять путями записи данных. Например если реализовать dout в виде последовательной логики, то мы получим выход только к концу сигнала, когда fifoPop будет уже закрыт. Если использовать fifoSrc, то можно считывать данные из fifo в любой момент времени, к примеру на следующем такте после fifoPop.

Недостатки fifoSrc:

- **Усложнение схемы:**

  Добавление нового сигнала увеличивает общее количество сигналов в системе, что усложняет схему и потребует больше ресурсов для управления этими сигналами.

- **Увеличение стоимости:**

  Увеличение количества сигналов приводит к увеличению стоимости из-за дополнительных ресурсов.

- **Увеличение потребления энергии:**

  Увеличение количества сигналов может привести к увеличению потребления энергии.

Был выбран вариант с использование fifoPop для определения направления данных в FIFO и fifoSrc для выбора источника данных для записи в регистр, так как это упрощает схему и экономит ресурсы.

Использовать выход ALU как вход FIFO

Достоинства:

- **Возможность использовать результат ALU перед записью в FIFO:**

  Использование выхода ALU как входа FIFO позволяет делать вычисления перед записью. Но это потребует изменение формата команды и некоторых архитектурных решений.

- **Потенциальная возможность сокращения управляющих сигнал:** Если бы мы использовали fifoSrc и wdScr, то в таком решении можно было бы убрать один из сигналов.

Недостаки:

- **Уменьшение скорости работы**

  Использование FIFO в качестве входа ALU может увеличить задержку сигнала, так как сигнал должен пройти через ALU даже если не требуется вычисления.

Был выбран вариант использовать FIFO параллельно ALU.

Использовать многотактовую реализацию процессора RISC-V

Достоинства:

- **Потенциально можно увеличить тактовую частоту процессора:**

  За счет разбиения на стадии можно выполнять некоторые операции параллельно, чтобы дать более медленным операциям время. Но так как в данной работе нет модулей, которые работают более одного такта, то это не даст значительных преимуществ.

- **Общая память для данных и комманд:**

  В многотактовой реализации можно использовать одну память для комманд и данных, что может уменьшить количество требуемых ресурсов.

Недостатоки:

- **Усложнение схемы:**

  Многотактовая реализация усложняет схему.

- **Большее количество ресурсов:**

  В многотактовой реализации требуется больше регистров, управляющих сигналов и тп.

Заключение

В результате выполнения лабораторной работы были успешно реализованы и протестированы модификации процессорного ядра архитектуры RISC-V. Добавление модуля FIFO и соответствующих команд ‘push‘ и ‘pop‘ позволило расширить функциональность процессора. Проведенный анализ временных параметров подтвердил соответствие временным требованиям, несмотря на некоторые отрицательные значения WNS. Выбор архитектурных решений был обоснован необходимостью упрощения схемы, избижанием вероятных ошибок при проектировании и улучшением гибкости управления.