Лабораторный практикум

«Проектирование цифровых устройств с помощью Verilog HDL»

Лабораторная работа №6
FLASH память

1.1 Виды энергонезависимой памяти

Ни один из блоков цифровых устройств, которые мы рассмотрели ранее не способен хранить информацию при отсутствии питания.

Чтобы решить эту проблему, на заре вычислительной техники, данные в цифровое устройство после подачи питания загружали с таких носителей, как перфокарты и, позже, магнитные ленты. Ещё позже для этих целей были разработаны накопители на гибких магнитных дисках — дискетах и жёстких магнитных дисках — «HDD». На данный момент для хранения данных при отсутствии питания наиболее широко применяется FLASH-память.

Энергонезависимые накопители информации обладают как преимуществами, так и недостатками по сравнению с энергозависимой RAM-памятью.

Как правило, энергонезависимая память существенно уступает по скорости работы RAM-памяти. Это ограничение удалось преодолеть только недавно: в 2016 году была представлена постоянная память, где информация хранится в виде спина электрона. Такая память по скорости работы не уступает современной RAM-памяти, такой как DDR5. Но подобная память ещё долгое время будет недоступна для рядового пользователя из-за высокой стоимости.

1.2 Принципы работы FLASH-памяти

В качестве элемента хранения информации FLASH-память

использует транзистор с плавающим затвором. Состояние затвора определяет бит хранимой информации.

На Рис.**??** схематично изображена структура такого транзистора.

[Picture goes here]

Как вы видите, он содержит два затвора: управляющий и плавающий. Плавающий затвор — это полупроводника, который полностью окружён диэлектриком. При этом плавающий затвор способен накапливать электроны. От величины накопленного заряда меняется «лёгкость» с которой транзистор открывается — т.е. величина напряжения «управляющий затвор—исток», при котором через транзистор начнёт течь ток. Чем больше электронов находятся в плавающем затворе, тем «легче» открывается транзистор — ток начинает протекать через него при меньшем напряжении «управляющий затвор—исток».

Для хранения информации используют следующий принцип (см. Рис.**??**): чтобы считать информацию, на управляющий затвор подаётся напряжение чтения — среднее между самым сильным и самым слабым напряжением, способным открыть затвор. Если транзистор открывается, значит в плавающем затворе были электроны и мы считаем, что в нём записано значение «0», если не открывается, значит электронов в плавающем затворе нет и записано значение «1».

[Picture goes here]

Осталось понять как можно «заставить» электроны попадать в плавающий затвор, ведь он изолирован диэлектриком. Не вдаваясь в подробности скажем, что если подать достаточно высокое напряжение «управляющий затвор—сток», то у электронов хватит энергии, чтобы «перескочить» диэлектрик и попасть в плавающий затвор. А если изменить полярность этого напряжения, то можно «выгнать» электроны наружу (см. Рис.**??**).

[Picture goes here]

Самое важное в этой идее то, что если электроны попали в плавающий затвор они не могут самостоятельно покинуть его через диэлектрик и будут оставаться там в течении многих лет. Таким образом и достигается сохранение записанной

информации при отсутствии питания.

Теперь мы знаем, что для того чтобы записать или считать информацию из FLASH-памяти надо использовать большую разность потенциалов. Но на самом деле транзистор устроен таким образом, что энергия, необходимая чтобы «загнать» электроны в плавающий затвор меньше энергии, необходимой, чтобы их «выгнать». Это делается чтобы при чтении значения электроны не покидали плавающий затвор.

При такой организации становится сложно обеспечить очистку каждого транзистора в отдельности, поэтому обычно стирается целая группа ячеек.

1.3 Особенности FLASH-памяти

Из-за особенностей транзистора с плавающим затвором, которые мы рассмотрели можно выделить следующие характерные черты FLASH-памяти:

- Запись значения возможна только из логической «1» в логический «0»;
- Удаление информации возможно только из группы ячеек одновременно (сектора);
- Удаление и запись информации приводят к деградации ячеек памяти;
- Чтение также приводит к деградации ячеек памяти, но в меньшей степени.

1.4 Структура FLASH-памяти

На Рисунке**??** изображена общая структура FLASH-памяти. Как видно она практически не отличается от RAM-памяти: из ячеек строится матрица, контролируемая управляющим блоком. А сам управляющий блок обеспечивает коммуникацию с внешними устройствами, дешифрацию адреса и управление записью и чтением массива элементов памяти. Подключение FLASH-памяти и управление ей со стороны цифрового устройства полностью зависит от того, как реализован блок управле-

ния — доступ к содержимому FLASH-памяти может быть синхронный или асинхронный, по последовательной или параллельной шине, с разделением шин адреса и данных или без него.

[Picture goes here]

1.5 Микросхема FLASH-памяти S29AL032D

Для практического знакомства с FLASH-памятью мы спроектируем контроллер микросхемы S29AL032D. Именно эта микросхема установлена на отладочной плате Altera DE1.

Основным источником информации о любой микросхеме служат технические условия (англ. datasheet). В этом документе содержатся все необходимые сведения для использования микросхемы: электрические параметры, размеры и тип корпуса, информация о выводах, и многие другие сведения. В том числе datasheet содержит данные о протоколах информационного обмена.

В нашем случае микросхема уже подключена, поэтому из всего datasheet нас, в первую очередь, интересует каким образом необходимо взаимодействовать с этой микросхемой, чтобы записать или считать данные.

Для разработки контроллера следует ознакомиться со следующими разделами документа:

11. Commands Definitions;
12. Write Operation Status;
17. AC Characteristics.

Далее будут приведены необходимые выдержки из документа, однако настоятельно рекомендуем ознакомиться с ним.

1.5.1 Проектирование контроллера S29AL032D

Как мы уже знаем, контроллер предназначен для обмена информацией с внешними цифровыми устройствами. Он дол-

жен предоставлять удобный, простой интерфейс и обеспечивать все необходимые взаимодействия с устройством. В таком случае другие блоки могут использовать один и тот же контроллер.

Чтобы начать разработку контроллера, нужно ответить важные вопросы: как должен работать наш контроллер и как он должен управляться?

Если мы хотим работать с памятью, то для нас наиболее важными являются операции записи и чтения данных. Тогда наиболее удобным для нас был бы уже знакомый интерфейс, похожий на RAM-память: данные для записи, данные для чтения, адрес и управляющие сигналы.

Теперь, когда мы определились с тем, как мы будем управлять контроллером, нам нужно понять как он должен взаимодействовать с самой микросхемой flash-памяти. Для этого изучим операции записи и чтения, описанные в datasheet S29AL032D.

1.5.2 Операция чтения

Чтение данных из микросхемы S29AL032D не требует никакой дополнительной подготовки. Временная диаграмма чтения приведена в пункте 17.2 datasheet и представлена на Рис.**??**

[Picture goes here]

Времена, указанные на диаграмме, приведены в Табл. 1.2

Обозн.	Описание	Min	Max
t_{RC}	Продолжительность цикла чтения	70нс	90нс
t_{ACC}	Задержка Адрес — Данные	70нс	90нс
t_{CE}	Задержка Выбор Чипа — Данные	70нс	90нс

Таблица 1.1: Временные характеристики операции чтения S29AL032D

Чтобы выполнить операцию чтения, нам нужно повторить эту временную диаграмму и соблюсти все временные интервалы. Но как это сделать?

Каким образом можно выдержать указанные временные интервалы?

Мы уже знаем, что единственным источником информации о времени для цифрового устройства может являться только сигнал синхронизации, частота которого заранее известна.

Привяжем времена, упомянутые в Таблице 1.2 к тактовому сигналу частоты 50 МГц, которым тактируется устройство. При этом учтём, что некоторые задержки могут быть равны нулю.

Полученная временная диаграмма показана на Рис. **??**

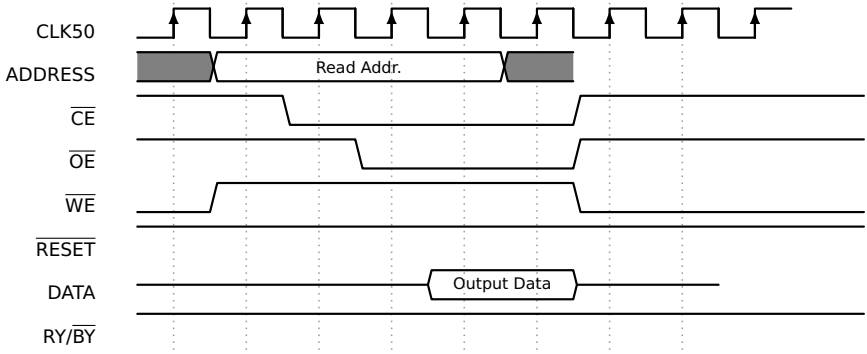

Рис. 1.1: Временная диаграмма операции чтения шины PCI

1.5.3 Операция записи

Обычно запись во flash-память — более сложная операция, чем чтение. Многие производители используют для записи специальные последовательности команд, защищая таким образом память от случайной записи.

Согласно datasheet S29AL032D (разделы 7 и 11) для того, чтобы осуществить запись нужного значения во flash-память, необходимо выполнить следующую последовательность из 4-х операций записи:

Записать данные AA по адресу AAA;
Записать данные 55 по адресу 555;
Записать данные A0 по адресу AAA;
Записать нужные данные по нужному адресу.

Временная диаграмма одной операции записи приведена в пункте 17.2??? datasheet и представлена на Рис.**??**, а её вре-

менные характеристики приведены в Табл. **??**.

[Picture goes here]

Обозн.	Описание	Min	Max
t_{RC}	Продолжительность цикла чтения	70нс	90нс
t_{ACC}	Задержка Адрес — Данные	70нс	90нс
t_{CE}	Задержка Выбор Чипа — Данные	70нс	90нс

Таблица 1.2: Временные характеристики операции чтения S29AL032D

Аналогично операции чтения, привяжем форму и времена временной диаграммы записи к тактовому сигналу, частотой 50 МГц. Полученная диаграмма, представлена на Рис. **??**

Также согласно datasheet, данные записываются не мгновенно. На то, чтобы провести операцию записи одного слова требуется порядка 11 мкс. Что приблизительно соответствует 550 тактам на частоте 50 МГц.

Также крайне важно, что при записи данных микросхема S29AL032D может менять значение с «1» на «0», но не наоборот!

Чтобы поменять значение с «0» на «1» требуется очистка целого фрагмента памяти, называемого сектором, либо полная очистка всей микросхемы!

Значит, для того, чтобы мы могли полноценно пользоваться микросхемой S29AL032D нам потребуется реализовать в контроллере функции очистки.

1.5.4 Операция очистки

Для очистки выбранного сектора необходимо выполнить следующую последовательность операций:

Записать данные AA по адресу AAA;
Записать данные 55 по адресу 555;
Записать данные 80 по адресу AAA;
Записать данные AA по адресу AAA;
Записать данные 55 по адресу 555;
Записать данные 30 по адресу сектора, который необходимо очистить.

Операция очистки сектора занимает существенное время,

и пока она не закончится, невозможно произвести запись или чтение из flash-памяти.

Операция полной очистки отличается только последним значением: для полной очистки данные 30 записываются по адресу AAA.

В datasheet на S29AL032D приведены следующие значения:

Очистка сектора - до NN мкс.

Полная очистка микросхемы - до NNN мкс.

1.5.5 Статус операции

Для того, чтобы контролировать завершение операций записи и очистки, а также отслеживать ошибки, которые могут возникнуть в процессе их выполнения необходимо получить информацию о статусе операции. Способы получения этой информации и её содержание приведены в разделе 12 datasheet. Далее мы отметим наиболее важные для нас моменты.

Так как на отладочном стенде Altera DE1, которым мы пользуемся для проведения лабораторных работ, не разведён сигнал BUSY микросхемы S29AL032D, то единственным способом получения статуса является чтение информации из адреса, по которому производилась запись.

Если операция удачно завершена — то будет получено значение, из указанного адреса. Для операции записи оно должно совпадать с тем, которое мы хотели записать, а для операции очистки должно содержать только единицы (8'hFF).

Если операция ещё не завершена, то биты [7:2] считанного значения будут содержать информацию о статусе операции. Расшифровка значений этих битов приведена в Табл. **??**

До окончания операции записи DQ[7] будет иметь значение противоположное записываемому («0» при очистке) - это основной признак того, что полученные данные отражают статус операции. Информация о битах статусного пакета приведена в Таблице **??**

Обратите внимание, что при повторном чтении некоторые биты статусного пакета меняют своё значение на противоположное. Это сделано, чтобы убедиться, что операция чтения выполняется корректно и микросхема не «зависла».

В информации о статусе операции есть важный признак: бит DQ[4] является признаком того, что время операции превысило максимально допустимое. Если этот бит принимает значение «1», то во время операции произошла какая-то ошибка. В подавляющем большинстве случаев это происходит при попытке записи в ячейку памяти, уже содержавшую какое-то значение.

1.5.6 Проектирование контроллера Flash (продолжение)

Теперь, когда мы познакомились с операциями, которые предстоит выполнять контроллеру, мы можем продолжить его проектирование.

Контроллер должен обеспечивать операции чтения, записи и очистки микросхемы. Для этого он должен последовательно обмениваться данными и производить проверку статуса операций. Значит в качестве его основы следует применить конечный автомат. Ведь именно конечный автомат позволяет нам разделить режимы работы и реализовать алгоритмы работы в цифровых устройствах.

Начнём проектировать конечный автомат с начального состояния - состояния бездействия. Будем постепенно наращивать его сложность и степень детализации, уточняя некоторые особенности.

[Picture goes here]

Из состояния бездействия возможны три различных перехода: операция чтения, операция записи и операция очистки.

[Picture goes here]

Теперь выделим основные этапы, которые присутствуют в этих операциях. Прежде всего нас интересуют сложные операции «запись» и «очистка».

Как уже говорилось, чтобы записать данные во flash-память требуется провести четыре обмена с flash-памятью. Но на этом нельзя заканчивать операцию, ведь необходимо дождаться окончания записи. Также надо учесть, что во время записи могут возникнуть ошибки.

Как мы уже говорили, для контроля статуса операции нам

нужно считать данные из адреса, по которому производится запись и проанализировать их. Отразим это в состояниях конечного автомата.

[Picture goes here]

Раньше мы не разделяли эти состояния и всё вместе называли «запись». Но, постепенно уточняя детали, мы разбили сложную операцию на более простые этапы.

Аналогично поступим с операцией очистки.

[Picture goes here]

Первое, что бросается в глаза - многократное повторение операций записи и чтения (которая используется при проверке статуса).

Также можно постараться выделить чтение и анализ статуса в отдельные состояния, общие для операций записи и очистки.

Тогда структура конечного автомата приобретает следующий вид:

[Picture goes here]

В состояниях W_n и E_n происходит запись значения во flash-память. В состояниях ST и R происходит чтение.

Можем ли мы выделить операции чтения и записи и реализовать их отдельно, чтобы затем использовать их как показано на графе переходов?

Чтобы понять это, сначала ответим на вопрос как вообще возможно реализовать эти операции в контроллере.

Для того, чтобы провести чтение, необходимо развернуть временную диаграмму, показанную на Рис.**??**. Тоже относится к записи: временную диаграмму записи, привязанную к тактовому сигналу, мы получили на Рис. **??**

Как мы уже обсуждали, схема которую можно использовать для разделения событий во времени — это конечный автомат. Например, чтобы реализовать операцию чтения, разобьём временную диаграмму чтения на этапы, и поставим каждому этапу в соответствие уникальное состояние, как представлено на Рис. **??**

Мы могли бы добавить эти состояния в конечный автомат, который мы уже начали проектировать, но тогда нам

пришлось бы каждое состояние чтения и записи разбить на несколько состояний. Это привело бы к ненужному усложнению структуры конечного автомата.

Вместо этого мы можем сделать отдельные небольшие модули, которые будут выполнять эти операции, и поместить автоматы в них. Например для операции чтения такой модуль будет управляться автоматом, представленным на Рис.**??**

Сигналом запуска для таких мини-автоматов будет признак того, что основной автомат находится в состоянии «чтение» или «запись».

Состояние «завершено» нужно для того, чтобы выработать сигнал окончания работы. Иначе «большой» автомат не будет иметь возможности «узнать» о том, что операция завершена и можно переходить в следующее состояние.

Теперь, когда мы оформили все основные идеи и общую структуру контроллера, можно преступить к его реализации на Verilog HDL.

Как всегда, начнём проектировать с интерфейса будущего контроллера - его входов и выходов. Так как контроллер будет обеспечивать доступ к памяти, мы хотели сделать его интерфейс похожим на интерфейс RAM-памяти. Но нам придётся ввести дополнительные сигналы для того, чтобы реализовать операцию очистки и индикацию ошибок.

Нам будет достаточно одного входа для адреса, так как мы не можем одновременно производить чтение и запись во flash-память.

Теперь опишем основной управляющий конечный автомат и мини-автомат чтения. Мини-автомат записи опишите самостоятельно.

Наладим связь между автоматами. Для этого определим управляющие сигналы (воздействия):

Начнём описывать исполняющую логику, которая будет задействована в различных состояниях: