1 Игрушечный модуль памяти

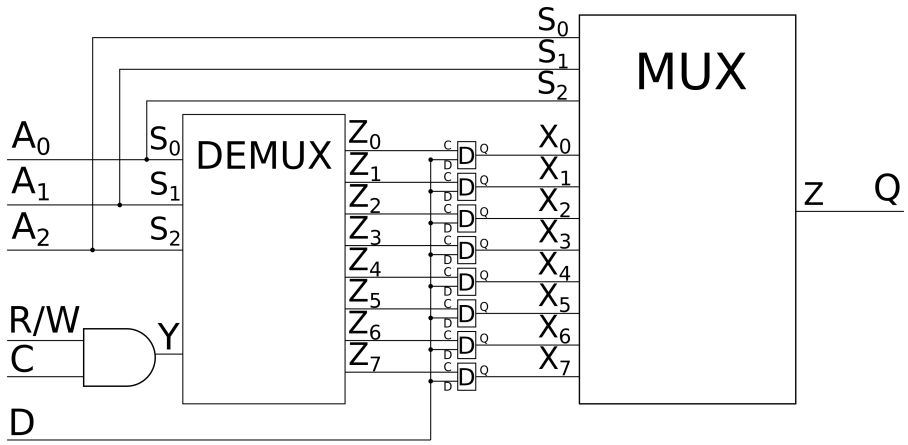

Рис. 1: Модуль памяти из D триггеров

Это конечно здорово, но никто так не делает. Потому что есть проблемы с масштабируемостью: на один гигабит понадобится слишком много проводов, кроме того будут проблемы с синхронизацией кучи триггеров.

2 Шина

Шина - набор проводов и протокол, по которому она используется.
PCI-E - внутренняя шина, внешний аналог - Thunderbolt.

3 Реальный модуль памяти

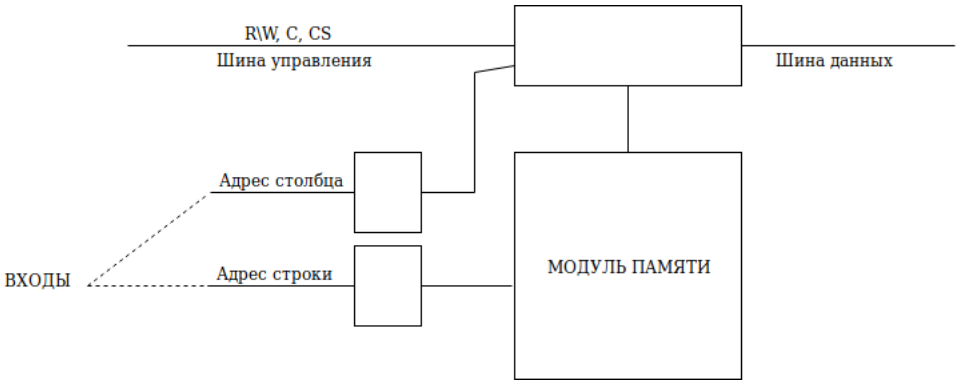

Рис. 2: Модуль памяти + контроллер

Заметим, что можем сократить число проводов вдвое, т.к. входы D и выходы Q никогда не используются одновременно. Сигнал $R\backslash W$ определяет, как мы используем провода. У нас есть провода и какой-никакой протокол обращения с ними, теперь это шина памяти.
У шины памяти есть две "подшины":

- Шина адреса. (Строго от контроллера к модулю)
- Шина данных. (От контроллера к модулю при записи, наоборот при чтении)

Идейно реальный модуль памяти организован в виде двумерной матрицы. Шина адреса тоже можем уменьшить вдвое: на первом такте будем передавать адрес строки, на втором - адрес столбца.

4 Ячейка памяти

4.1 Статическая

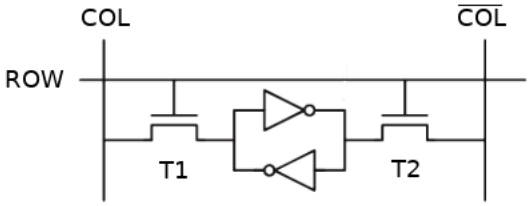

Рис. 3: Шеститранзисторная ячейка статической памяти

Является энергозависимой (т.е. без питания данные теряются).
Содержит 5 проводов: ROW, COL и $\overline{COL}$, а так же питание и земля.

4.1.1 Чтение

В теории, нужно только подать напряжение на ROW и считать значения с COL и $\overline{COL}$.
На практике*, COL и $\overline{COL}$ - достаточно длинные провода, и между ними может образоваться ненужное нам электрическое поле. Поэтому, чтобы ускорить чтение, используется более хитрый процесс: сначала на COL и $\overline{COL}$ подается высокое напряжение (логическая **1**). Затем напряжение подается на ROW, транзисторы $T1$ и $T2$ открываются, из-за чего напряжение на одной из линий COL и $\overline{COL}$ чуть-чуть падает. Определяя, на каком проводе напряжение выше, узнаем, что хранилось в ячейке: **0** или **1**.

4.1.2 Запись

Подаем на COL и $\overline{COL}$ то, что хотим записать. Затем подается напряжение на ROW.

4.1.3 Преимущесва

- Не требует постоянной перезарядки(поэтому и называется статической)
- Транзисторы переключаются быстрее, чем заряжается/разряжается конденсатор, поэтому работает быстрее DRAM.

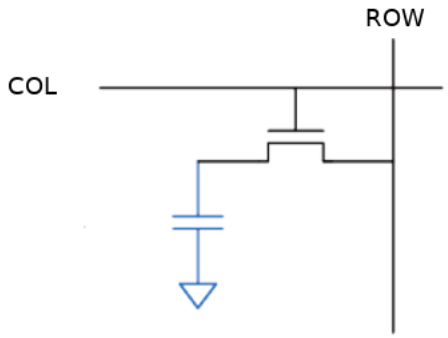

Рис. 4: Ячейка динамической памяти

4.1.4 Недостатки

- Дорого
- Занимает много места (целых 4/6/8/10 транзисторов на ячейку!)

4.1.5 В итоге

Используется там, где нужно мало быстрой памяти. Т.е. в кэшах, регистрах.

4.2 Динамическая

Тоже является энергозависимой.
Содержит всего три провода: COL, ROW и земля.
Состоит из одного транзистора и одного конденсатора.

4.2.1 Чтение

Читаем сразу всю строку. В теории: подаем напряжение на ROW и замеряем значения. Не забываем перезарядить, т.к. у конденсаторов есть свойство разряжаться при чтении. На практике*: подаём половинку напряжения логической **1** на столбцы. Затем подаем напряжение на ROW. Замеряем изменения: если стало меньше, чем подали - там **0**, стало больше - **1**.

4.2.2 Запись

Очень просто: подаем напряжение на ROW, затем подаем нужное напряжение на COL.

4.3 Преимущества

- Дешево
- Каждая ячейка занимает мало места -> можно сделать больше ячеек.

4.3.1 Недостатки

- После чтения нужно перезаряжать конденсаторы.

- Конденсаторы очень маленькие, имеют очень маленькую ёмкость, поэтому они разряжаются сами по себе за очень быстро. Нужно постоянно перезаряжать (считывать строку и записывать обратно).
  Этим может заниматься программист или контроллер памяти. В наше время этим занимается контроллер памяти.

- Все эти фокусы с конденсаторами достаточно долгие.

4.3.2 В итоге

Используется там где нужно много дешевой памяти. RAM, например.

5 Лирическое отступление

5.1 DMA и PIO

DMA (*Direct Memory Access*) - фича, которая позволяет некоторым аппаратным устройствам обращаться к памяти минуя процессор. Возникла потому что если каждое медленное устройство, которому нужно что-то от памяти, будет дергать быстрый процессор - всем станет очень грустно по скорости. Типичный представитель DMA устройства - HDD.
Как работает: CPU инициирует/разрешает передачу данных от одного устройства другому, а потом занимается своими делами, пока не получит прерывание (когда DMA контроллер закончит передачу данных).
PIO (*Programmed Input/Output*) - подход, когда процессор во время операции чтения/записи не может ничего делать. Если включить такой режим для HDD (вроде как можно было в BIOS), можно получить хороший такой проигрыш в скорости.

5.2 SATA

SATA (*Serial ATA*) - популярный интерфейс для подключения долговременной памяти. Можно подключать и HDD, и SSD, и SSHD.

5.3 Чипсет

Процессор очень быстрый, а все остальные не очень. Поэтому давайте высокоскоростные устройства подключим поближе к процессору (северный мост), а медленные - подальше (южный мост). Можно заметить, что в современности наблюдается тенденция засунуть весь северный мост на один кристалл к процессору, чтобы быстрее. Принцип SoC - "система на кристалле".

5.4 Поток сознания. Что-то из этого полезно, но это не точно

- **PS\2 не хотспот, а еще для него не нужен драйвер, в отличие от USB**

- Во времена DOS внутри каждой програмки (например игрушки) была программа конфигурации. Сначала нужно было запустить её, потом саму программу. Если программа не умела в имеющееся железо - ОЖВП.

- Одна из задач операционной системы - построение абстракции для работы на

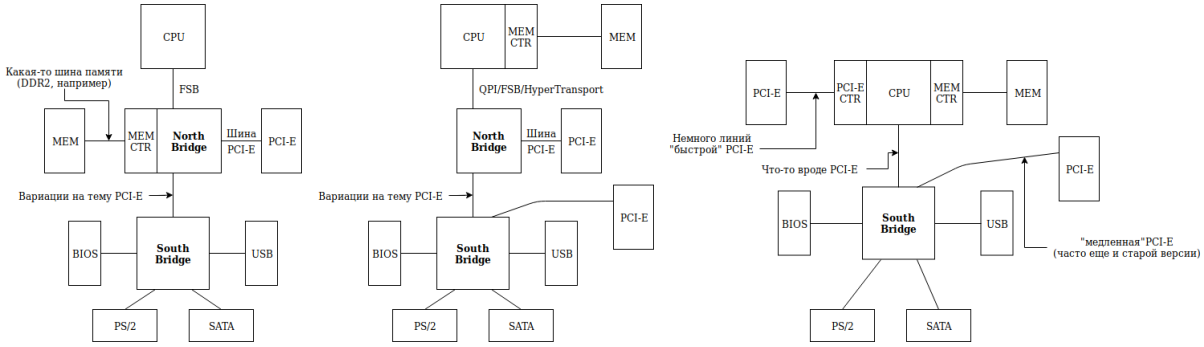

Рис. 5: Вариации чипсета. Вправо новее

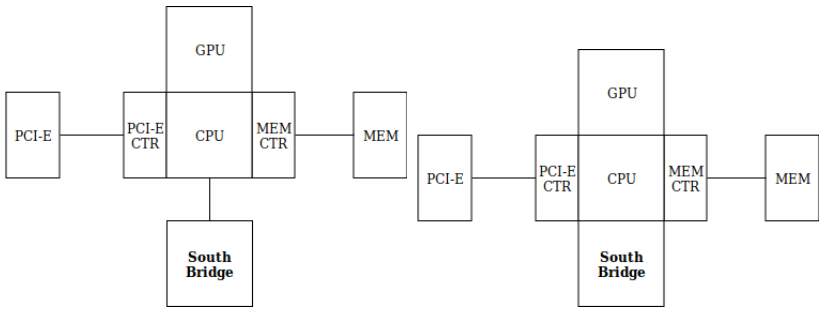

Рис. 6: Слева еще более новая вариация. Справа мобильный чипсет

любом железе. Следовательно можно использовать API, а ОС будет передавать управление драйверу. Драйвер транслирует команды API в команды для железки. Драйверы системно-зависимые.

- Вставьте сюда вашу любимую байку про заговор NVIDIA.

6 Источники информации

- Википедия
- Конспекты @ntwwwnt (есть в гуглопапке)