1 Обычная DRAM

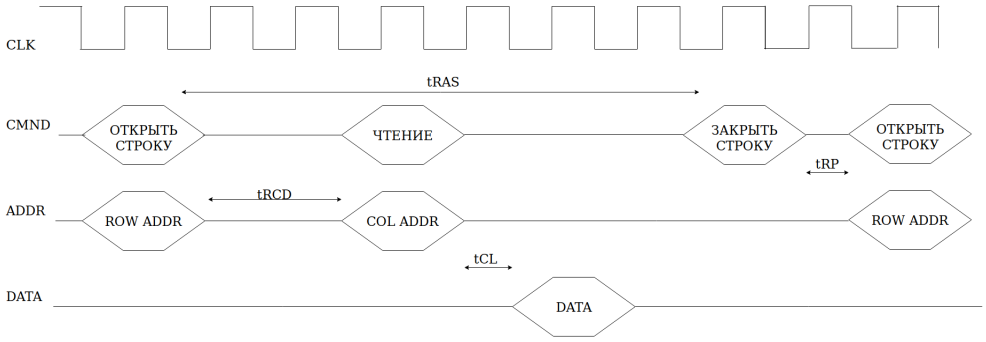

Рис. 1: Схема работы DRAM

Вернемся к устройству модуля памяти. За два такта передаём адрес. После выбора строки все ячейки этой строки считываются в буфер (SRAM). На этот момент в самом модуле в строке одни 0, т.к. конденсаторы разрядились при чтении. Уже из статической памяти считываем столбец и всю строку записываем обратно в динамическую.
Внимание_ 0: эта штука **асинхронная**.
Внимание_ 1: адрес столбца должен быть зажат пока не получены данные.

2 Тайминги

- **CL** - *CAS (Column Address Strobe) Latency* - число тактов между отправкой адреса столбца и началом получения данных.
- **tRCD** - *RAS (Row Address Strobe) to CAS Delay* - минимальное число тактов, которое нужно подождать при открытой строке, прежде чем открывать столбец.
- **tRP** - *Row Precharge* - минимальное число тактов, которое нужно подождать между закрытием строки и открытием новой.
- **tRAS** - *Row Active Strobe* - минимальное число тактов между откртием и закрытием строки.

3 Как сравнивать два модуля памяти?

- Если частота одинаковая, то чем меньше CL, тем лучше для последовательных запросов. Чем меньше tRP + tRAS, тем лучше для случайных запросов.
- Если частота разная, то такты в памяти с большей частотой будут означать пропорционально меньшие промежутки времени.

4 Характеристики памяти

Скорость бывает двух видов: **скорость доступа** *latency* и **скорость передачи данных** или **пропускная способность** *throughput*.
Скорость доступа характеризует время от получения запроса до получения первой

порции данных. В RAM скорость доступа = tCL + tRCD
Скорость передачи данных характеризует время от n-ной до $n+1$-ой порции данных. В RAM скорость передачи данных = tRAS + tRP
Как видно из графика, со скоростью доступа всё очень грустно.

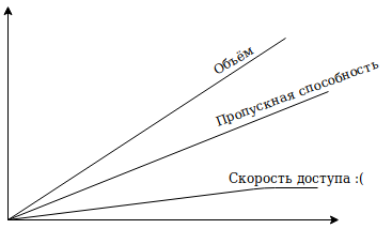

Рис. 2: График, показывающий как изменялись характеристики памяти с течением времени

5 Важная информация

Память - *пассивное* устройство. Процессор *запрашивает*, память *отвечает на запрос*.

6 FPM DRAM

FPM DRAM - textitFast Page Mode DRAM - а давайте закрывать строку не каждый раз, а только когда следующий запрос идет в другую строку. Т.е. читать столбцы одной строки без закрытия этой строки.
Улучшилась скорость передачи
Внимание_0: эта штука **асинхронная**.
Внимание_1: адрес столбца должен быть зажат пока не получены данные.

Figure 1: Simplified Fast Page Mode Read

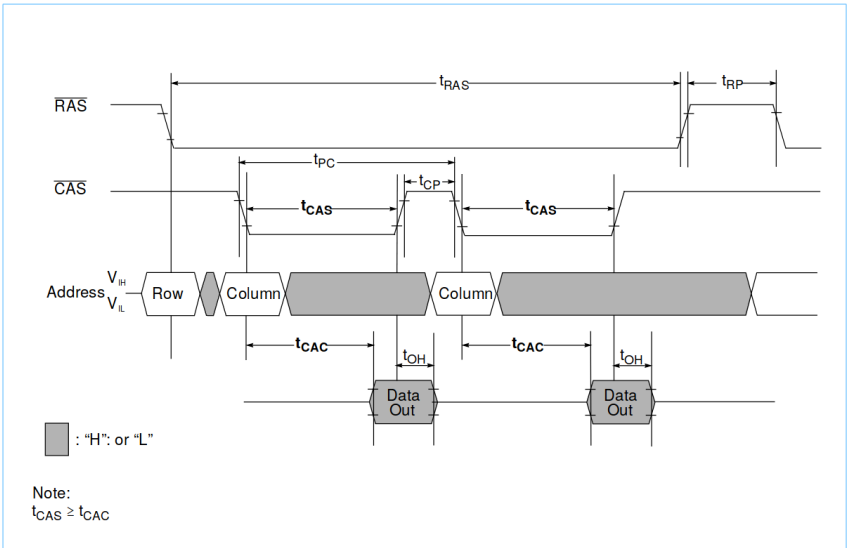

Рис. 3: Упрощенная схема работы FPM

7 EDO DRAM

EDO DRAM - *Enabled Data Out DRAM* - а давайте добавим буфер для адреса столбца. Теперь пока читается один столбец, мы можем уже подать адрес следующего, а после получения данных с первого столбца, сигнал на чтение второго.
Улучшилась скорость передачи.
Внимание: эта штука всё ещё **асинхронная**.

Figure 2. Simplified EDO Mode Read

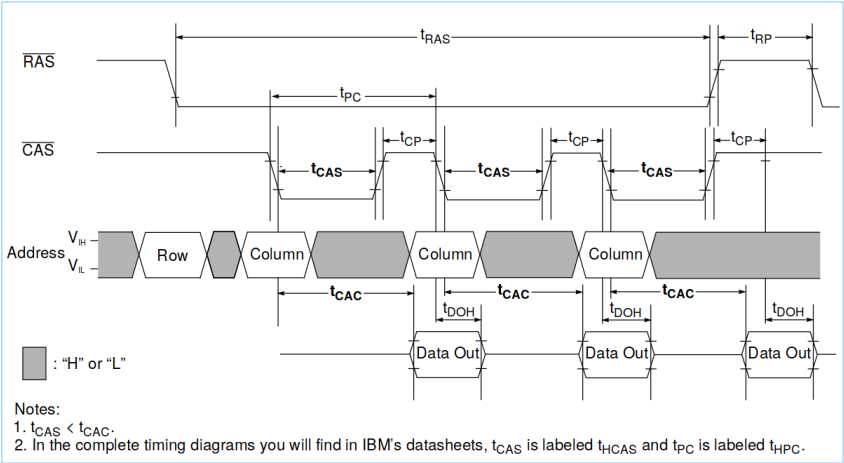

Рис. 4: Упрощенная схема работы EDO

8 BEDO DRAM

BEDO DRAM - *Burst EDO DRAM* - теперь на запрос чтения столбца память отдаёт не только нужный столбец, но и три следующих. Ограничения: адрес должен быть кратен четырем.
Улучшилась скорость передачи.
Внимание: эта штука всё ещё **асинхронная**.

9 SDRAM

До появления SDRAM контроллер память и модуль памяти *не были синхронизированы*. У них даже могли быть разные частоты.
Пусть у нас tRCD - 3 такта. Но контроллеру приходилось ждать "с запасом".
SDRAM - *Synchronious DRAM* - теперь контроллер и память на одной синхронизации. Следовательно контроллер может ждать ровно tRCD тактов.
Кроме того, в SDRAM увеличили ширину шины до 64 бит и появились **банки памяти** (от 2 до 4).
За счет поджатия таймингов и увеличения ширины шины увеличилась скорость передачи.
Внутренняя и внешняя шина SDR SDRAM работают на одной частоте. Если хочется, чтобы память работала быстрее, нужно поднимать частоты. А поднимать частоты достаточно грустно, потому что растет энергопотребление и тепловыделение.

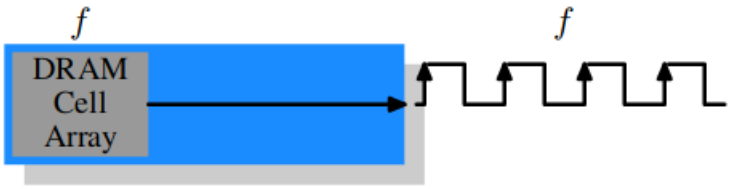

Рис. 5: **SDR SDRAM** - Single Data Rate SDRAM

10 Еще немного про модули памяти

Процессор явно работает быстрее, чем память. Следовательно у памяти рано или поздно появится очередь запросов. Хотелось бы сделать такую память, которая может обрабатывать запросы параллельно.

10.1 Многопортовая память

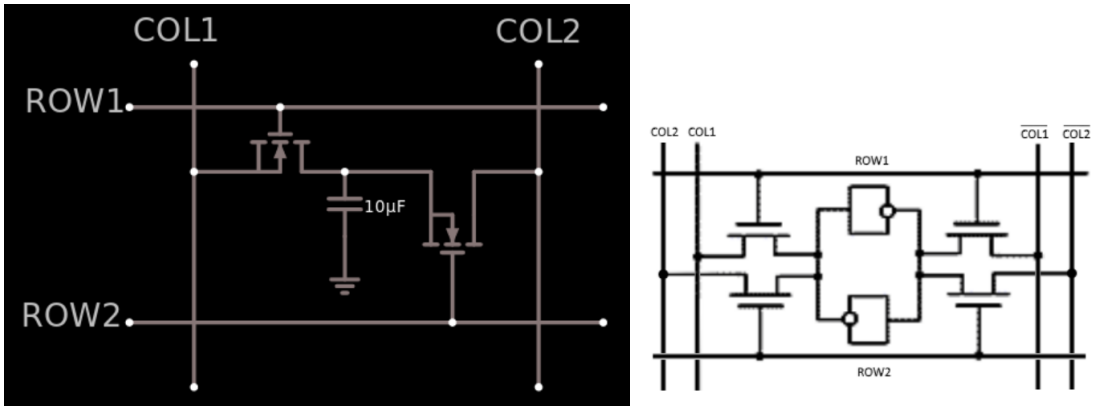

Рис. 6: Двухпортовые ячейки DRAM(слева) и SRAM(справа)

При таком подходе к каждой ячейке подходит два набора линий строки/столбца.
Идейно мы как будто бы обращаемся к двум разным матрицам, заполненным одинаковыми данными. **НО!** Мы не можем обратиться одновременно к одной и той же строке.
Улучшается время доступа к разным строкам.
На практике используется в SRAM, а в DRAM нет.

10.2 Многобанковая память

Идейно среднее между онопортовой и двухпортовой памятью)000))))
При таком подходе используем действительно две разных матрицы. Усложняется только интерфейсная схема, а сами ячейки остаются простыми.
В итоге получаем улучшения при работе со строками, лежащими в разных банках памяти.

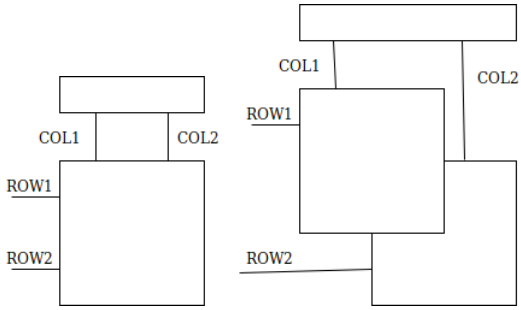

Рис. 7: Модуль многопортовой и многобанковой памяти

10.3 Многоканальный режим работы

Пусть есть sdram с шиной 64 бита. Втыкаем в оба канала модуль памяти. Теперь шина 128 бит. Желательно чтобы модули памяти были максимально похожи. Лучше чтобы они были одинакового объема, иначе контроллер может сказать "не, я так не умею и работать в одноканальном режиме.

11 DDR

DDR SDRAM - *Double Data Rate SDRAM* - увеличили внутреннюю шину в два раза, а ширину внешней оставили прежней. Можем передавать данные два раза за такт: по фронту и по спаду импульса синхронизации. (На пересечении прямого и инвертированного импульса синхронизации)
Маркетинговая уловка: а давайте введем понятие эффективной частоты (на какой частоте нужно было бы работать старому модулю памяти, чтобы передавать столько же информации). Нормальная частота не поменялась, но теперь можно говорить, что эффективная частота в два раза выше, чем раньше. Такие дела.

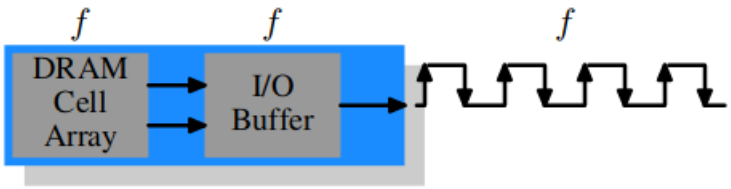

Рис. 8: DDR SDRAM

12 DDR2

Еще раз увеличили ширину внутренней шины в два раза, и подняли частоту внешней в два раза. Снизили напряжение питания. Увеличили число банков памяти до 8.

13 DDR3

Еще раз увеличили ширину внутренней шины в два раза, и подняли частоту внешней в два раза. Снизили напряжение питания.

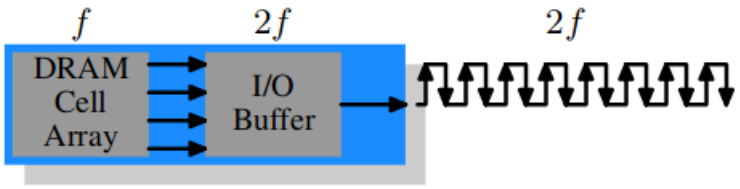

Рис. 9: DDR2 SDRAM

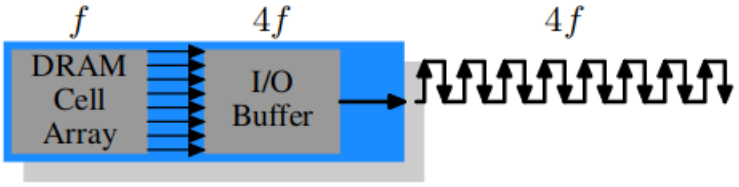

Рис. 10: DDR3 SDRAM

14 DDR4

~~Еще раз увеличили ширину внутренней шины...~~ А вот и нет. Дальше увеличивать ширину внутренней шины уже слишком жирно (там и так $8n$ проводов, если в SDRAM n). Поэтому только уменьшили напряжение питания. Увеличили число банков памяти до 16.

15 GDDR

Если память используется видео карточкой, то очень хочется поднять скорость передачи данных, т.е. увеличить тактовую частоту GDDR для графических процессоров. ИХ интересует почти только скорость передачи и пофиг на скорость доступа. Свою память могут позволить себе крутые дорогие видеокарточки. Из этого следует что мы можем позволить себе большее по цене и энергии. Можем позволить себе заплатить побольше и отвести побольше тепла, ради увеличения скорости передачи данных.

DDR2 -> GDDR2
говно получилось

DDR2 -> GDDR3
Оптимизация под увеличение скорости передачи

DDR3 -> GDDR4
тоже говно вышло

DDR3 -> GDDR5
Широко используется
Передача 4 порции данных за такт. Два канала данных. Эффективная частота завышена в 4 раза.

GDDR5x
Имеет режим в котором еще в два раза увеличили скорость передачи

16 HBM память*

Особая технология подключения памяти к процессору. Используется в топовых видеокарточках. Ширина шины сильно больше. Но технология сильно дороже и сложнее, чем обычные печатные платы.

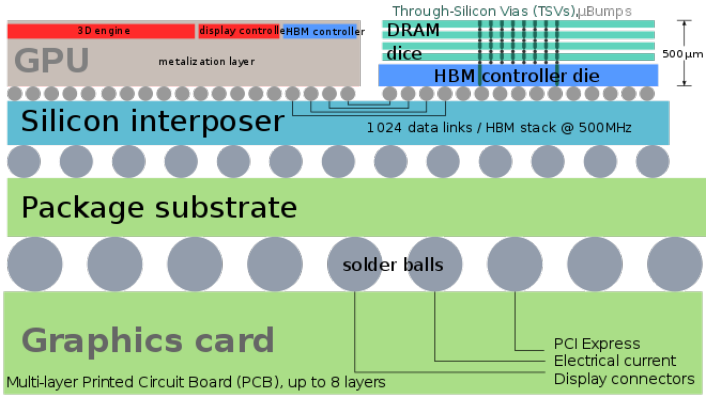

Рис. 11: Технология HBM в разрезе

17 Доступ к памяти

17.1 NUMA

NUMA ((Not-Uniform Memory Access)) - архитектура с неоднородным доступом к памяти. Т.е. не ко всем блокам памяти обращаемся за одинаковое время. Преимущества:

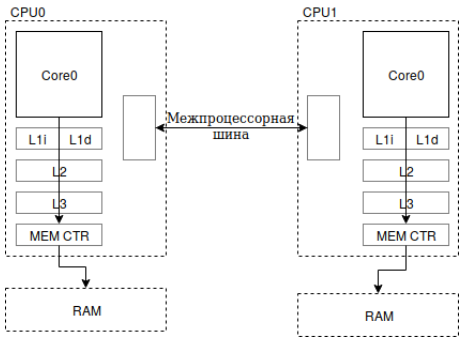

Рис. 12: Двухпроцессорная NUMA

лучше масштабируется, можем разместить больше блоков памяти. Больше каналов к памяти, следовательно выше параллельность запросов.
Недостатки: программы будут замедляться, если треды будут обращаться к "неблизкой"памяти.

17.2 UMA

UMA ((Uniform Memory Access)) - архитектура с однородным доступом к памяти. Т.е. ко всем блокам памяти обращаемся за одинаковое время. Преимущества: в теории не

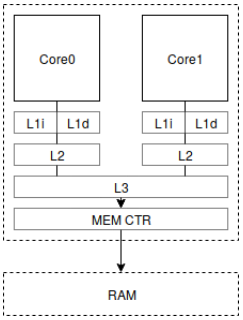

Рис. 13: Однопроцессорная двухъядерная UMA

можем проиграть с тем, что нужные данные лежат не в ближней памяти.
Недостатки: на практике сложно масштабировать, канал к памяти - ботлнек.

18 Источники

- Википедия
- конспекты лекций by @ntwwwnt
- "What Every Programmer Should Know About Memory"by Ulrich Drepper