День 1.

Была начата разработка контроллера памяти:

1. Создан модуль контроллера memory\_controller;
2. Сделан расчёт параметров (модуль параметризирован):

- ширины адресного пространства для банок (BANK\_ADRESS),

- для строк (ROW\_ADRESS), колонн (COLUMN\_ADRESS),

- определена ширина шины данных (4 бита).

Определены:

- время одного такта (CAS\_LATENCY – 7 нс (для CAS = 3)),

- задержка для перезарядки линии слов (RP\_TIME = 15 нс (для CAS = 3, -7)),

- задано время рефреша ячеек памяти (REFR\_TIME = 64 мс);

**CAS** – количество тактов и соответственно время, которое необходимо для того что бы защёлкнулся адрес колонны и был выдан результат (CAS = 3 такта = 22,5 нс) (RAS аналогично для строк);

1. Объявлены все (или почти все, надо ещё разобраться входы/выходы);
2. Создан каркас конечного автомата: добавлены состояния INIT\_HOLD, PRECHARGE, IDLE, AUTO\_REFR, переходы и выходные состояния для них в рамках инициализации памяти;
3. Реализована инициализация памяти до шага заполнения внутренних регистров;
4. Добавлен флаг для инициализации init\_flag;

День 2.

1. Параметры для задержек (REFR\_TIME, RP\_TIME, RC\_TIME) были переведены из секунд в количество циклов для упрощения условий и для того, чтобы эти условия были более специфичными для данного по условию контроллеру;
2. Выходы пришлось сделать регистрами reg, иначе ModelSim ругается
3. Добавлено состояние MRS – для записи параметров burst во внутренний регистр при инициализации памяти, так как эти данные не определены при запуске платы. Параметры для burst заданные по умолчанию:

– Full page,

– Sequential,

– CAS = 3,

– Standard operation Mode,

– Programmed Burst Length;

Данное состояние описывает команду MRS памяти. Переход из этого состояния сразу в IDLE, чтобы контроллер подавал NOP и память ушла в IDLE тоже.

1. Переходы между состояниями при инициализации были обыграны через использование двух флагов init\_flag и MRS\_flag. Первый необходим для того, чтобы контроллер выполнял порядок действий по маршруту инициализации (по факту был в состоянии инициализации), а второй флаг нужен для того, чтобы было возможным двойной переход к refresh\_all, сначала при неактивном флаге MRS\_flag переход к рефрешу, потом когда случился переход флаг выставляется активным и переход уже будет к MRS. Все команды нужные для инициализации игнорируются без флага, которой снимается при переходе в последнее состояние MRS. Чтобы установка обоих флагов не была противоречивой, было принято решение вынести установку флага в последовательностную часть, где установка происходит синхронно.
2. Так же для отсчёта, прошедшего времени нужно было отсчитать количество тактов, для этого был создан второй счётчик counter\_db (дублирующий счётчик). В последовательностной части проверяется является ли следующие состояние таким, которое нуждается в подсчёте тактов и если да, то выставляется маркер времени от которого идёт отсчёт тактов. Но при этом, чтобы он не обновлялся постоянно (ведь состояние может не меняться много тактов, но при этом next\_state будет состоянием, которое нуждается в подсчёте), проверяется не является ли текущее состояние этим же самым состоянием.
3. Получилось запустить симуляцию и разобраться в параметрах симуляции. Создан модуль тестбенча.
4. План на следующие дни:

Сделать рефреш, чтение и запись

и

Сделать тестбенч для проверки работы инициализации, проверить правильность отправки команд при инициализации

(В любом порядке)

**↓**

Сделать тестбенч для рефреша, чтения и записи

**↓**

Оформить документацию

**↓**

Разработать доп.функции

1. Показана схема переходов контроллера памяти (стрелочкой показаны безусловные переходы по окончанию операции). И соответствующие этим состояниям команды для памяти (или их состояния, кажется что это одно и то же, но не совсем).

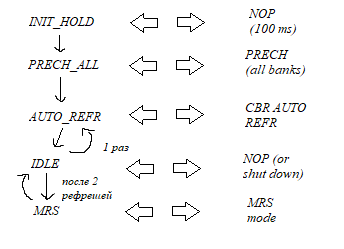


Рисунок 1 – Состояния контроллера и соответствующие иму команды для памяти

День 3.

По расчётам выходит, что цикл чтения в худшем случае постоянного выбора новой строки занимает 4 цикла. В данном случае при частоте 133 МГц имеем

Началась разработка чтения в одиночно режиме. Перед чтением (и перед записью) необходимо использовать команду ACTIVE, которая позволяет активировать строку для работы с ней.

Если чтение в этой же строке или в этой же банке, то для одиночного чтения будет использоваться READ, после которого будет использоваться PRECHARGE.

Если переход на следующую банку, то можно использовать READA.

Таким образом средняя длина записи равна 7 циклам (по длине исполнения для READ, как самой частой операции).

Можно оптимизировать схему и хранить в данном случае информацию не подряд, а по банкам. Четыре подряд идущих слова разбиваются на 4 банки. В данном случае получится, благодаря READA, сократить время чтения до 4 циклов. Но перезапись данных может перестать быть простой задачей. Хотя если использовать WITEA, то можно будет использовать тот же подход.