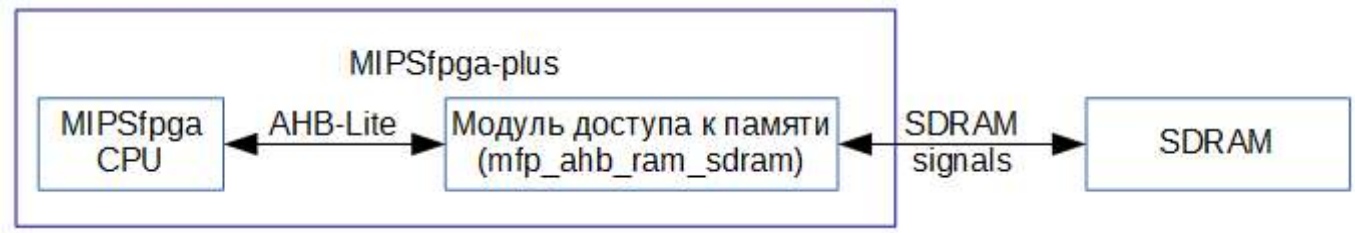
SparF 10 февраля 2017 в 00:44

MIPSfpga и SDRAM. Часть 2

Анализ и проектирование систем*, FPGA*, Программирование микроконтроллеров*

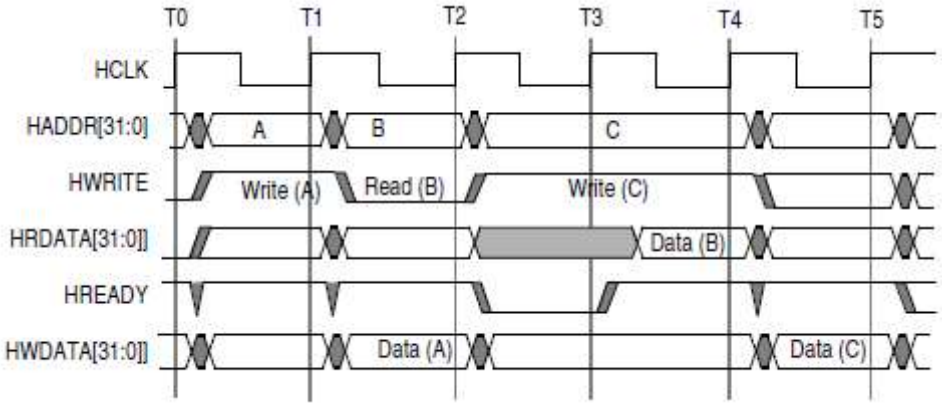
Ссылка на первую часть

Рассматриваемая нами конфигурация состоит из следующих элементов:



Шина AHB-Lite

Является основным инструментом для общения ядра MIPSfpga с внешним миром. Из нее в модуль доступа к SDRAM поступают команды на чтение и запись информации, по ней же передаются считываемые и записываемые данные. Основная особенность: фаза адреса последующей команды совпадает по времени с фазой данных текущей команды. Лучше всего это видно на следующей диаграмме:



Краткое описание изображенных сигналов: HCLK — тактовый сигнал; HADDR — адрес, данные по которому мы хотим записать или прочитать на следующей фазе, задается мастером; HWRITE — при высоком уровне на следующей фазе должна быть произведена операция записи, выставляется мастером; HRDATA — прочитанные данные; HREADY — флаг завершения текущей операции; HWDATA — записываемые данные, выставляются мастером

в состав пакета MIPSfpga.

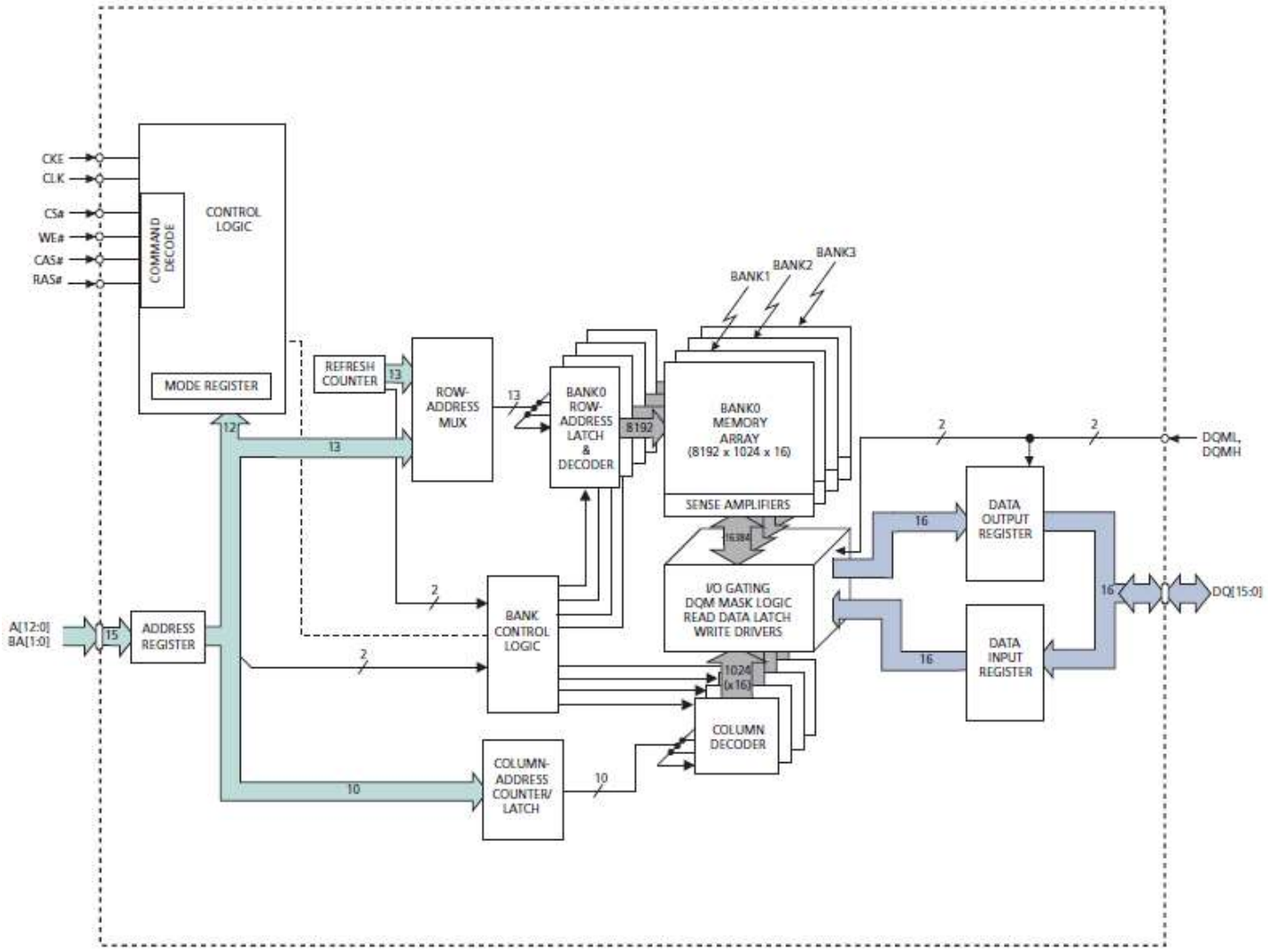
Память SDRAM

Основные принципы, на которых построена SDRAM, очень хорошо описаны в Главе 5 учебника Харрис-энд-Харрис [1]. Отметим главные моменты:

- для хранения одного бита используется информация о наличии или отсутствии заряда конденсатора;
- память организована в виде матриц из емкостей и управляющей логики: со столбцами и строками;
- во время операции чтения заряд ячейки (конденсатора) расходуется, после чтения ее приходится подзаряжать;
- во время бездействия величина сохраненного заряда также уменьшается (пусть и медленнее) — ячейки памяти требуют периодической подзарядки (т.н. регенерации).

Дальнейшее рассмотрение продолжим на примере микросхемы MT48LC64M8A2 компании Микрон. Помимо очень удобного и детального даташита компания предоставляет Verilog модель для симуляции работы с этим чипом памяти. Что, с одной стороны, значительно упрощает разработку, а с другой, позволяет, не имея отладочной платы, запустить MIPSfpga внутри симулятора и посмотреть, как ядро взаимодействует с SDRAM.

Структурная схема чипа памяти представлена на рисунке ниже.



Основные элементы:

- банк (матрица) памяти (4x bank memory array) — именно здесь хранятся интересующие нас данные. В рассматриваемой микросхеме 4 банка, в каждом из которых 8192 строки и 1024 столбца по 16 бит каждый. Итого, суммарная емкость чипа 4x8192x1024x16 = 512 Mb = 64 MB.
- устройство управления (control logic, bank control logic) — обеспечивают декодирование полученной команды и выдачу соответствующих управляющих сигналов на остальные элементы;
- мультиплексоры, защелки и декодеры адресной шины (row-address mux, 4x bank row-address latch & decoder, column-address counter/latch, column decoder) — обеспечивают хранение адресной информации строк, столбцов и банков памяти, поступающей в разных командах;

- регистры и логика шины данных (data output register, data input register, i/o gating, dqm mask logic) — обеспечивают ввод/вывод данных при операциях чтения и записи, позволяют работать с масками (когда из 16 бит нам нужен только старший или младший байты), обеспечивают перевод выводов шины данных в Z-состояние, шина является двунаправленной.

Условия функционирования

Для корректной работы ОЗУ нам необходимо выполнить ряд условий. Часть из них рассматривать не будем: обеспечение температурного режим, стабильность частоты и питания, уровни сигналов (статическая дисциплина), правильная разводка на плате. В поле нашего зрения остается:

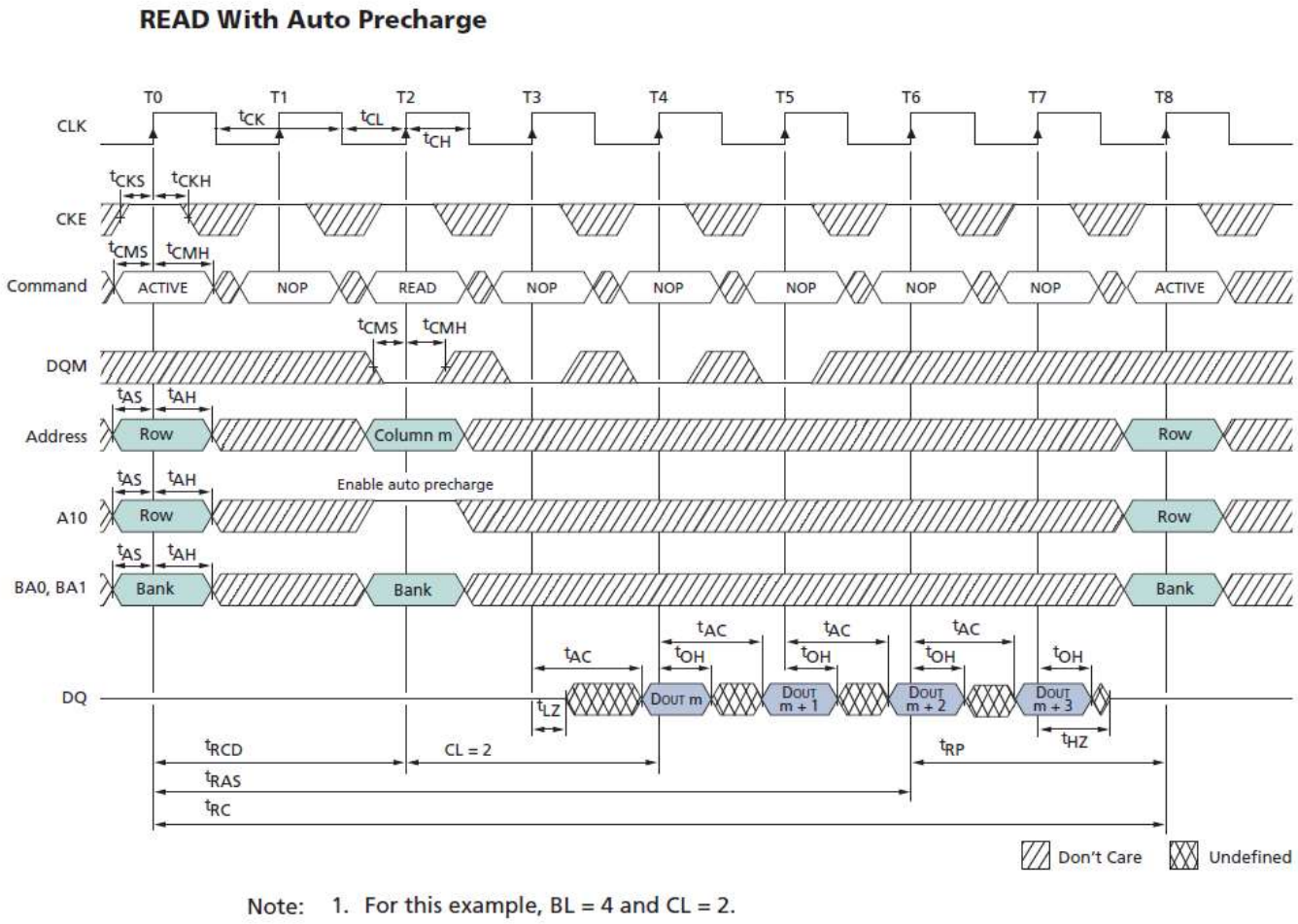
- подача корректных управляющих сигналов, соответствующих той или иной команде;
- удовлетворение требованиям динамической дисциплины (Глава 3 учебника Харрис-энд-Харрис [1])с учетом требований документации на микросхему [2].

Для того, чтобы предметно понимать о чем идет речь, рассмотрим, что должен сделать модуль доступа к памяти при чтении данных из ОЗУ. В качестве примера выступит случай т.н. READ With Auto Precharge — когда микросхема после операции чтения сама обеспечивает подзарядку ячеек, к которым мы обратились. Инициализация модуля (INIT), операции записи (WRITE) или автоматическая регенерация (AUTO_REFRESH) выполняются аналогичным образом, с разницей в выполняемых командах и накладываемых временных ограничениях.

Ниже приведены выкопировки из даташита: таблица истинности для команд и временная диаграмма, на которой показано как производится корректное чтение данных.

Name (Function)	CS#	RAS#	CAS#	WE#	DQM	ADDR	DQ	Notes
COMMAND INHIBIT (NOP)	H	X	X	X	X	X	X	
NO OPERATION (NOP)	L	H	H	H	X	X	X	
ACTIVE (select bank and activate row)	L	L	H	H	X	Bank/row	X	2
READ (select bank and column, and start READ burst)	L	H	L	H	L/H	Bank/col	X	3
WRITE (select bank and column, and start WRITE burst)	L	H	L	L	L/H	Bank/col	Valid	3
BURST TERMINATE	L	H	H	L	X	X	Active	4
PRECHARGE (Deactivate row in bank or banks)	L	L	H	L	X	Code	X	5
AUTO REFRESH or SELF REFRESH (enter self refresh mode)	L	L	L	H	X	X	X	6, 7
LOAD MODE REGISTER	L	L	L	L	X	Op-code	X	8
Write enable/output enable	X	X	X	X	L	X	Active	9
Write inhibit/output High-Z	X	X	X	X	H	X	High-Z	9

Примечание: L — низкий уровень, H — высокий уровень, X — не имеет значения, High-Z — высокий импеданс.



Note: 1. For this example, BL = 4 and CL = 2.

Примечание: tCMS — command setup time, tCMH — command hold time, tAS — address setup time, tAH — address hold time, tRCD — active command to read, tRAS — command period (ACT to PRE), tRC — command period (ACT to ACT), tLZ — output Low impedance time, tAC — access time from clock, tOH — output data hold time, tRP — commad period (PRE to ACT). Минимальные значения этих и других параметров для разных условий приведены в документации на чип памяти.

Последовательность действий при чтении данных (по тактам)

T0. Не позднее чем за tCMS до фронта CLK обеспечить наличие установивших сигналов на выводах CS#, RAS#, CAS#, WE#, DQM (далее — команда), соответствующих команде ACTIVE. Указанные сигналы не должны менять свое состояние в течении tCMH с момента фронта CLK. Не позднее чем за tAS до фронта tCLK установить на шине адреса (A[12:0]) адрес строки, на шине адреса банка памяти (BA[1:0]) — адрес банка памяти. Эти сигналы должны быть стабильны в течении tAH после фронта CLK.

T1. В течении (tRCD — 1 такт) подавать команду NOP. По истечении этого временного промежутка ранее переданный адрес строки будет гарантированно сохранен в row-address latch & decoder соответствующего банка памяти, произойдет выбор одной из 8192 строк (см.структурную схему чипа).

T2. Не позднее чем за tCMS до фронта CLK обеспечить ввод команды READ, не менять команду в течении tCMH с момента фронта CLK. Не позднее чем за tAS до фронта tCLK установить на шине адреса адрес столбца, на шине адреса банка памяти — адрес банка памяти. Десятый бит шины адреса устанавливается в 1 как признак того, что после чтения необходимо выполнить Auto Precharge.

T3-T7. Обеспечить подачу команды NOP на все время чтения данных и не менее чем на (tRC — 1 такт) с момента подачи команды ACTIVE.

T4. Спустя CL тактов (т.н. CAS Latency, CAS) считанные данные будут гарантированно присутствовать на шине данных DQ. Если более точно, то они появятся на шине спустя (1 такт + tAC) — для случая, когда CAS = 2. И будут стабильны в течении минимум tOH после фронта CLK. За это время данные с шины необходимо считать.

Если смотреть на взаимодействие внутри чипа, то за время (1 такт + tAC) адрес столбца будет сохранен в column-address counter/latch, на выходе соответствующего банку памяти column decoder будет установлены сигналы, выбирающие 16 бит необходимого нам столбца, эти данные поступят в data output register и, в итоге, окажутся на шине данных (DQ[15:0]).

T5-T7. Рассматриваемый нами пример предполагает, что чип памяти был настроен на выполнение пакетных операций (burst) с размером пакета BL = 4 (burst length, задается в числе других параметров командой LOAD MODE REGISTER, в текущей реализации модуля доступа к памяти он задан как BL = 2, чтобы получить 32 бита данных). По этой причине в течении последующих трех тактов column-address counter/latch будет автоматически увеличиваться на единицу, а на выход шины данных — поступит еще 3x16 бит.

Необходимо учесть, что количество тактов не обязательно будет равным 8, как изображено на диаграмме (T0-T7) — оно должно быть увеличено в большую сторону с целью удовлетворения требований всех временных ограничений: tRCD, tRC и т.д.

Требования временных ограничений выполняются с помощью

- смещения фазы тактового сигнала, на котором работает память (CLK) относительно тактового сигнала, на которой работает модуль доступа к памяти — для малых промежутков (tCMS, tCMH, tAS, tAH, tAC, tOH);
- подачей пустых команд (NOP) на больших промежутках (tRCD, tRC, tRP) — где размер задержки превышает ширину 1 такта тактового сигнала. Для этого в состав конечного автомата модуля введены соответствующие состояния.

Смещение фазы тактового сигнала

Есть несколько хороших источников ([3] и [4]), которые аргументировано противопоставляют "научный" подход определения смещения фазы тактового сигнала методу "проб и ошибок". В этих документах приведен ряд формул по вычислению границ "безопасных окон", в которые нужно подставить значения задержек. После чего предлагается сместить тактовые сигналы таким образом, чтобы их фронты оказались максимально близки к центрам этих "окон". Соглашаясь с тем, что описанная методика работает, хочу обратить внимание на несколько более "ленивый" вариант этого же подхода (мне кажется, что он изображен на 12 и 20 страницах презентации, но т.к. комментариев к ней нет, то я в этом не уверен):

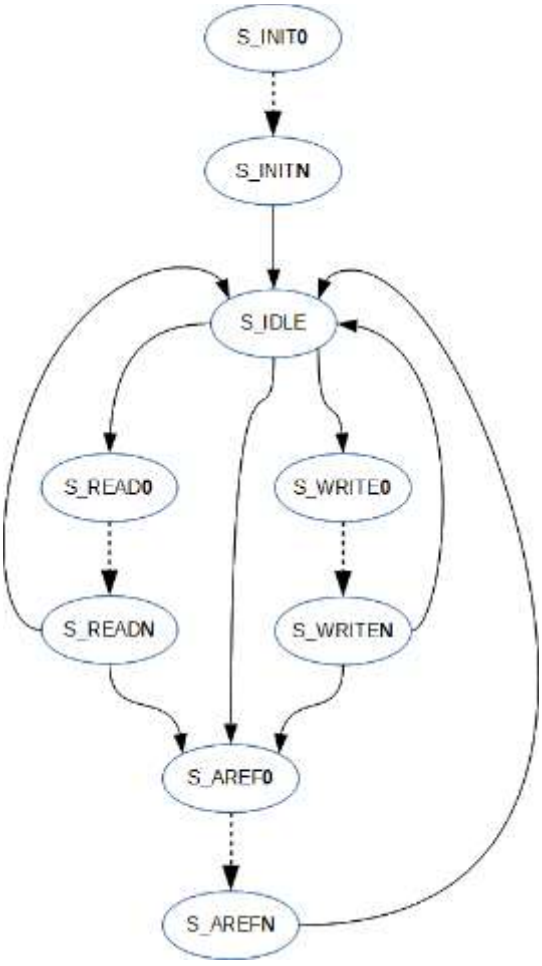
- берем два листка/полоски бумаги в клетку (можно миллиметровку);
- с соблюдением масштаба наносим на каждый из них несколько тактов тактового сигнал, один из них — для fpga, другой — для микросхемы памяти;
- с соблюдением масштаба отмечаем на каждом из них:
  запрещенные зоны, в которых считываемый входной сигнал не должен меняться (А);
  зоны, в которых значение выходного сигнала не определено;
  зоны, в которых выходной сигнал является валиндным (Б).
- располагаем полоски бумаги параллельно и смещаем их относительно друг друга (a-la логарифмическая линейка) так, чтобы зоны А находились как можно ближе к центрам зон Б и ни в коем случае не выходили за их границу.
- линейкой измеряем полученное смещение тактовых сигналов, переводим его в ns согласно масштаба.

Чтобы обеспечить точное и стабильное смещение фаз в состав системы необходимо включить PLL-модуль. Обычно я добавляю еще 3-й тактовый сигнал с частотой в 4 раза выше, чем другие и небольшим фазовым смещением — для того, чтобы использовать его в

качестве тактовой частоты для логического анализатора (SignalTap) при отладке взаимодействия с памятью в железе.

Модуль доступа к памяти

В этом разделе приведены диаграмма состояний конечного автомата модуля доступа к памяти, а также отдельные строчки кода модуля, описывающие процедуру чтения данных (с указанием номеров строк кода для упрощения навигации). Исходные коды модуля целиком: mfp_ahb_ram_sdram.v. В случае, если чтение скриншотов с кодом доставляет вам дискомфорт, фрагменты исходников из статьи (включая комментарии к ним) продублированы на github.



Состояния конечного автомата, описывающие процедуру чтения, полностью соответствуют тому, что было описано выше на примере диаграммы READ With Auto Precharge.

```
82
83                  S_READ0_ACT         = 20,   /* Doing ACTIVE */
84                  S_READ1_NOP         = 21,   /* Waiting for DELAY_tRCD after ACTIVE */
85                  S_READ2_READ        = 22,   /* Doing READ with Auto precharge */
86                  S_READ3_NOP         = 23,   /* Waiting for DELAY_tCAS after READ */
87                  S_READ4_RD0         = 24,   /* Reading 1st word */
88                  S_READ5_RD1         = 25,   /* Reading 2nd word */
89                  S_READ6_NOP         = 26,   /* Waiting for DELAY_afterREAD - it depends on tRC */
90
```

Правила перехода между этими состояниями:

```
133     always @ (*) begin
134
135         //State change decision
136         case(State)
137             S_IDLE              :   Next = NeedAction ? (HWRITE ? S_WRITE0_ACT : S_READ0_ACT)
138                                                       : (NeedRefresh ? S_AREF0_AUTOREF : S_IDLE);

155             S_READ0_ACT         :   Next = (DELAY_tRCD == 0) ? S_READ2_READ : S_READ1_NOP;
156             S_READ1_NOP         :   Next = DelayFinished ? S_READ2_READ : S_READ1_NOP;
157             S_READ2_READ        :   Next = (DELAY_tCAS == 0) ? S_READ4_RD0 : S_READ3_NOP;
158             S_READ3_NOP         :   Next = DelayFinished ? S_READ4_RD0 : S_READ3_NOP;
159             S_READ4_RD0         :   Next = S_READ5_RD1;
160             S_READ5_RD1         :   Next = (DELAY_afterREAD != 0) ? S_READ6_NOP : (
161                                            NeedRefresh  ? S_AREF0_AUTOREF : S_IDLE );
162             S_READ6_NOP         :   Next = ~DelayFinished ? S_READ6_NOP : (
163                                            NeedRefresh  ? S_AREF0_AUTOREF : S_IDLE );

175         endcase
176     end
```

Там, где необходима задержка, она заносится в регистр delay_n, нулевое значение регистра соответствует флагу DelayFinished. На статусах S_READ4_RD0 и S_READ4_RD1 производится считывание данных из шины DQ:

```
178     always @ (posedge HCLK) begin
179
180         //short delay and count operations
181         case(State)
182
186             S_READ0_ACT         :   delay_n <= DELAY_tRCD       - 1;
187             S_READ2_READ        :   delay_n <= DELAY_tCAS       - 1;
188             S_READ5_RD1         :   delay_n <= DELAY_afterREAD  - 1;
189
192             default             :   if (|delay_n) delay_n <= delay_n - 1;
193         endcase
194
203         //data and addr operations
204         case(State)
205
214             S_READ4_RD0         :   DATA [15:0] <= DQ;
215             S_READ5_RD1         :   HRDATA <= { DQ, DATA [15:0] };
216
218         endcase
219     end
```

Кодирование команд и их вывод в зависимости от текущего состояния:

```
221     // SADDR = { BANKS, ROWS, COLUMNS }
222     wire  [COL_BITS - 1 : 0]  AddrColumn  = SADDR_old [ COL_BITS - 1 : 0 ];
223     wire  [ROW_BITS - 1 : 0]  AddrRow     = SADDR_old [ ROW_BITS + COL_BITS - 1 : COL_BITS ];
224     wire  [BA_BITS  - 1 : 0]  AddrBank    = SADDR_old [ SADDR_BITS - 1 : ROW_BITS + COL_BITS ];
225
226     reg    [ 4 : 0 ]    cmd;
227     assign  { CKE, CSn, RASn, CASn, WEn } = cmd;
228
229     parameter   CMD_NOP_NCKE        = 5'b00111,
230                 CMD_NOP             = 5'b10111,
231                 CMD_PRECHARGEALL    = 5'b10010,
232                 CMD_AUTOREFRESH     = 5'b10001,
233                 CMD_LOADMODEREG     = 5'b10000,
234                 CMD_ACTIVE          = 5'b10011,
235                 CMD_READ            = 5'b10101,
236                 CMD_WRITE           = 5'b10100;
248   // set SDRAM i/o
249   assign DQM = { DM_BITS { 1'b0 }};
250
251   always @ (*) begin
252       case(State)
253           default             :   cmd = CMD_NOP;
254
259           S_READ0_ACT         :   begin cmd = CMD_ACTIVE;         ADDR = AddrRow;     BA = AddrBank; end
260           S_READ2_READ        :   begin cmd = CMD_READ;           ADDR = AddrColumn | SDRAM_AUTOPRCH_FLAG;
261                                         BA = AddrBank; end
262
267       endcase
268
274   end
```

Все задержки являются настраиваемыми и задаются в параметрах модуля, что должно

упростить портирование на другие платы, а также модификацию настроек в случае изменения частоты тактового сигнала.

```
module mfp_ahb_ram_sdram
#(
    parameter   ADDR_BITS           = 13,       /* SDRAM Address input size */
                ROW_BITS            = 13,       /* SDRAM Row address size */
                COL_BITS            = 10,       /* SDRAM Column address size */
                DQ_BITS             = 16,       /* SDRAM Data i/o size, only x16 supported */
                DM_BITS             = 2,        /* SDRAM Data i/o mask size */
                BA_BITS             = 2,        /* SDRAM Bank address size */
                SADDR_BITS          = (ROW_BITS + COL_BITS + BA_BITS),

    // delay params depends on Datasheet values, frequency and FSM states count
    // default values are calculated for simulation(!): Micron SDRAM Verilog model with fclk=50 MHz and CAS=2
    parameter   DELAY_nCKE          = 20,       /* Init delay before bringing CKE high
                                                   >= (T * fclk) where T    - CKE LOW init timeout
                                                                        fclk - clock frequency  */
                DELAY_tREF          = 390,      /* Refresh period
                                                   <= ((tREF - tRC) * fclk / RowsInBankCount)  */
                DELAY_tRP           = 0,        /* PRECHARGE command period
                                                   >= (tRP * fclk - 1)                         */
                DELAY_tRFC          = 2,        /* AUTO_REFRESH period
                                                   >= (tRFC * fclk - 2)                        */
                DELAY_tMRD          = 0,        /* LOAD_MODE_REGISTER to ACTIVE or REFRESH command
                                                   >= (tMRD * fclk - 2)                        */
                DELAY_tRCD          = 0,        /* ACTIVE-to-READ or WRITE delay
                                                   >= (tRCD * fclk - 1)                        */
                DELAY_tCAS          = 0,        /* CAS delay, also depends on clock phase shift
                                                   =  (CAS - 1)                                */
                DELAY_afterREAD     = 0,        /* depends on tRC for READ with auto precharge command
                                                   >= ((tRC - tRCD) * fclk - 1 - CAS)          */
                DELAY_afterWRITE    = 2,        /* depends on tRC for WRITE with auto precharge command
                                                   >= ((tRC - tRCD) * fclk - 1)                */
                COUNT_initAutoRef   = 2         /* count of AUTO_REFRESH during Init operation */
)
```

Список литературы

[1] Учебник Дэвида Харриса и Сары Харрис «Цифровая схемотехника и архитектура компьютера»
[2] Документация на микросхему памяти MT48LC64M8A2 компании Микрон;
[3] Документация Quartus. Ядро конроллера SDRAM (перевод)
[4] SDRAM PLL Tuning (презентация)
[5] Ryan Donohue. Synchronization in Digital Logic Circuits (презентация)
[6] Документация на микросхему памяти IS42S16320D

Все даташиты, статьи и презентации на которые есть ссылки в статье, доступны на github.

Теги: fpga, микроэлектроника, verilog, sdram, mipsfpga, mips

Хабы: Анализ и проектирование систем, FPGA, Программирование микроконтроллеров

27 Карма 0 Рейтинг

Stanislav Zhelnio @SparF

Пользователь

Facebook Github

Комментарии 3

iCpu
10.02.2017 в 06:47

Я, конечно, эстет, и тёмные темы студии мне визуально очень нравятся, но, может, оставите код в теге [CODE], хотя бы в спойлере под картинками?

+1 Ответить

SparF
10.02.2017 в 20:04

Честно пытался вставить код в тег, но местная подсветка синтаксиса + умирающие отступы превращают его во что-то страшное и нечитаемое. Продублировал часть статьи на github.

0 Ответить

Mirn
27.09.2018 в 17:47

подскажите пожалуйста какую скорость SDRAM Вам удалось получить на каких конфигурациях и частотах?
Я совсем недавно начал это дело изучать и получается всего 50-60% от макс теоретической (когда данные по DQ считаются при каждом такте всегда)

0 Ответить

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

6 декабря 2018 в 17:11

RAM with Simple direct-mapped cache simulation on FPGA in Verilog

+7 3.9K 14 14 +14

23 февраля 2017 в 10:07

MIPSfpga и внутрисхемная отладка

+23 7.6K 53 2 +2

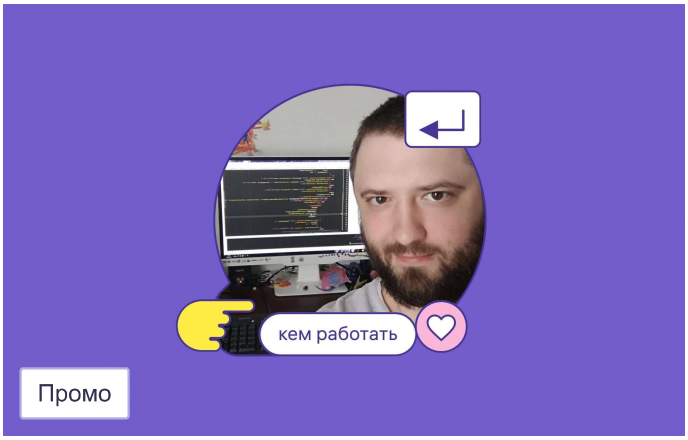
10 февраля 2017 в 00:44

MIPSfpga и SDRAM. Часть 1

+37 6.9K 49 0

МИНУТОЧКУ ВНИМАНИЯ

Разместить



Кем работать в ИТ в 2022: Frontend-разработчик

Анкета: попробуй себя в роли автора контент-студии Хабра

ЗАКАЗЫ

сегодня в 10:28

Как быстро реализовать поиск на корпоративном портале

+25 767 10 2 +2

Разбираем формулу КСП и объясняем разницу между χ^2-тестами

Турбо

ЧИТАЮТ СЕЙЧАС

Национальная система DNS-спуффинга

5.7K 15 +15

Зарплаты айтишников в первом полугодии 2022: впервые за пять лет средняя зарплата не изменилась

8.7K 15 +15

Очень странные дела на GitHub

23K 33 +33

Новые владельцы сервиса Resume.io уволили российскую команду и передали разработку в Индию

8.1K 35 +35

GitLab намерен удалять бесплатно размещённые проекты, если там не будет активности в течение года

4.3K 41 +41

Дарим ноутбук за лучшую статью о Java

Интересно

РАБОТА

Системный аналитик
369 вакансий

Все вакансии

Ваш аккаунт

Войти
Регистрация

Разделы

Публикации
Новости
Хабы
Компании
Авторы
Песочница

Информация

Устройство сайта
Для авторов
Для компаний
Документы
Соглашение
Конфиденциальность

Услуги

Корпоративный блог
Медийная реклама
Нативные проекты
Образовательные программы
Стартапам
Мегапроекты

Настройка языка

Техническая поддержка

Вернуться на старую версию