**Лабораторна робота №5**

**Тема:** контролер динамічної оперативної пам'яті DRAM

**Мета:** Отримати знання про функціонування динамічної оперативної пам’яті; отримати навики проектування контролера динамічної оперативної пам’яті.

**Теоретичні відомості**

Динамічна оперативна пам’ять (Dynamic Random Access Memory, DRAM) є одним з видів пам'яті із довільним доступом. DRAM характеризується великою щільністю комірок (ячеек), що забезпечує її низьку вартість. Комірка (ячейка) DRAM являє собою конденсатор, заряд якого відповідає значенню біта інформації, що зберігається у комірці. Але цей заряд поступово зменшується та через деякий час інформація втрачається. Тому для DRAM характерним є періодичний процес регенерації, який оновлює заряди у комірках.

Для зменшення кількості виводів мікросхеми DRAM повна адреса комірки поділяється на **адресу рядка** та **адресу стовпця**, які передаються до мікросхеми DRAM послідовно. Для зазначення частини адреси, яка передається у поточний момент, використовуються сигнали активації рядка та стовпця (RAS та CAS).

На рис. 1 наведена схема підключення контролера DRAM, який з’єднує процесор та мікросхему DRAM.

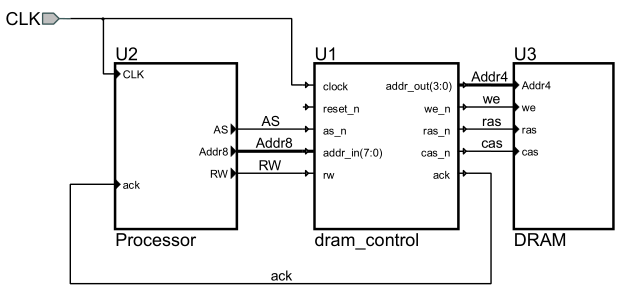


Рис. 1. Використання контролера динамічної пам'яті

Контролер DRAM приєднується до шини адреси і при виконанні операцій запису/зчитування здійснює перетворення 8-бітної адреси, що передається від процесора по шині Addr\_in(7:0), в дві 4-бітних адреси – номер рядка та номер стовпця в матриці мікросхеми DRAM, які потім у послідовному режимі передаються мікросхемі DRAM по шині Addr\_out(3:0). Крім того, завдання контролера DRAM полягає в періодичному генеруванні сигналів для проведення циклу регенерації. У разі несвоєчасного проведення циклу регенерації інформація у комірках мікросхеми DRAM буде втрачена.

Часова діаграма функціонування контролера DRAM при здійсненні операції зчитування з мікросхеми DRAM наведена на рис. 2. На діаграмі можна виділити 5 станів контролера DRAM:

1. Очікування.

2. Вибір рядка.

3. Вибір стовпця.

4. Зчитування даних.

5. Готовність даних.

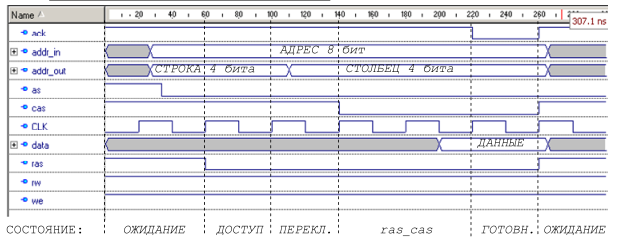


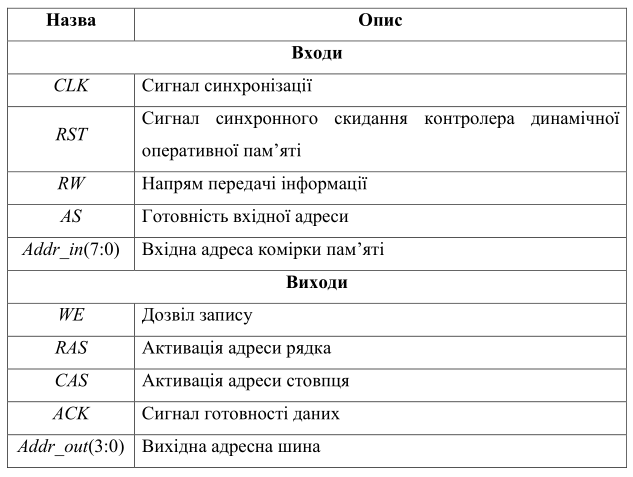
Рис. 2. Тимчасова діаграма функціонування контролера

при здійсненні операції зчитування з DRAM

В процесі операції читання або запису спочатку на шину адреси подається адреса рядка. При активізації сигналу RAS вона завантажується в регістр адреси рядка мікросхеми DRAM. Через деякий час на шину адреси подається адреса стовпця, яка завантажується в регістр засувку адреси стовпця при активізації сигналу CAS. Далі в залежності від сигналу напряму дані або виставляються на шину даних, або зчитуються з неї.

**Завдання**

Таблиця 1. Інтерфейс контролера динамічної оперативної пам’яті



1. Розробити програму контролера динамічної оперативної пам’яті з

інтерфейсом, який наведено у табл. 1.

2. Провести моделювання розробленої програми.

**Код програми**

**/\*Поведенческий код\*/**

**// ОПРЕДЕЛЕНИЯ КОНСТАНТ**

`define DEL 1 // Задержка распространения сигнала в

// устройстве. Нулевая задержка может

// привести к проблемам

`define RBC\_CYC 2 // Число тактов для удержания сигнала RAS

// перед установкой сигнала CAS

`define CBR\_CYC 1 // Число тактов для удержания сигнала CAS

// перед установкой сигнала RAS

`define RACW\_CYC 1 // Число тактов для удержания сигналов

// RAS и CAS вместе при записи

`define RACR\_CYC 2 // Число тактов для удержания сигналов

// RAS и CAS вместе при считывании

`define RACRF\_CYC 1 // Число тактов для удержания сигналов

// RAS и CAS вместе при регенерации

`define CNT\_BITS 2 // Число тактов, необходимое счетчику

// для подсчитывания представленных

// ниже сигналов

`define REF\_cnt 24 // Число тактов между регенерациями

`define REF\_BITS 5 // Число битов, необходимое счетчику

// для отсчитывания промежутка

// между регенерациями

`define AOUT 4 // Разрядность шины адреса

// со стороны DRAM

`define AIN 2\*`AOUT // Разрядность шины адреса

// со стороны процессора

**// ГЛАВНЫЙ МОДУЛЬ**

module Dram\_control(

clock,

reset\_n,

as\_n,

addr\_in,

addr\_out,

rw,

we\_n,

ras\_n,

cas\_n,

ack);

**// ВХОДЫ**

input clock; // Тактирующий сигнал для

// конечных автоматов

input reset\_n; // Срабатывающий по спаду

// синхронный сигнал сброса reset

input as\_n; // Срабатывающий по спаду

// строб адреса

input [`AIN-1:0] addr\_in; // Адрес со стороны процессора

input rw; // Сигнал выбора запись/считывания

// = 1 соответствует считыванию

// = 0 соответствует записи

**// ВЫХОДЫ**

output [`AOUT-1:0]addr\_out; // Адрес для DRAM

output we\_n; // Сигнал разрешения записи

output ras\_n; // Строб адреса строки для

// памяти DRAM

output cas\_n; // Строб адреса столбца для

// памяти DRAM

output ack; // Сигнал подтверждения

// для процессора

**// ОБЪЯВЛЕНИЯ СИГНАЛОВ**

wire clock;

wire reset\_n;

wire [`AIN-1:0] addr\_in;

wire as\_n;

wire rw;

wire we\_n;

wire ras\_n;

wire cas\_n;

wire ack;

wire [`AOUT-1:0] addr\_out;

reg [3:0] mem\_state; // Конечный автомат

wire col\_out; // Адрес колонки/строки

// = 1 – выбор колонки

// = 0 – выбор строки

reg [`CNT\_BITS-1:0] count; // Счетчик тактов

reg [`REF\_BITS-1:0] ref\_count; // Счетчик тактов при

// регенерации

reg refresh; // Запрос на регенерацию

**// ПАРАМЕТРЫ**

// Биты состояний представляют следующие сигналы:

// col\_out, ras, cas, ack

parameter[3:0] // Состояния автомата

IDLE = 4’b0000,

ACCESS = 4’b0100,

SWITCH = 4’b1100,

RAS\_CAS = 4’b1110,

ACK = 4’b1111,

REF1 = 4’b0010,

REF2 = 4’b0110;

**// ОПЕРАТОРЫ ПОТОКОВ ДАННЫХ**

// Пересылка сигналов на управляющие выходы из состояния автомата

assign col\_out = mem\_state[3];

assign ras\_n = ~mem\_state[2];

assign cas\_n = ~mem\_state[1];

assign ack = mem\_state[0];

// Сброс сигнала we\_n в течении регенерации

assign #`DEL we\_n = rw | (mem\_state == REF1) |

(mem\_state == REF2);

// Пересылка адресов столбца или строки на DRAM в зависимости от

// значения сигнала col\_out

assign #`DEL addr\_out = col\_out? addr\_in[`AOUT-1:0] :

addr\_in[`AIN-1:`AOUT];

**// ОСНОВНОЙ КОД**

// Отслеживание событий на сигнале reset

always @(reset\_n) begin

if (~reset\_n) begin

#`DEL assign mem\_state = IDLE;

assign count = `CNT\_BITS’h0;

assign ref\_count = `REF\_CNT;

assign refresh = 1’b0;

end

else begin

#`DEL;

deassign mem\_state;

deassign count;

deassign ref\_count;

deassign refresh;

end

end

// Ожидание возрастания тактирующего сигнала

always @(posedge clock) begin

// Наступило ли время запроса на регенерацию?

if (ref\_count == 0) begin

refresh <= #`DEL 1’b1;

ref\_count <= #`DEL `REF\_CNT;

end

else

ref\_count <= #`DEL ref\_count – 1;

// Обнуление счетчика тактов

if (count)

count <= #`DEL count – 1;

case (mem\_state)

IDLE: begin

// Запрос на регенерацию имеет наибольший приоритет

if (refresh) begin

// Загрузка счетчика для отсчитывания времени

// удержания сигнала CAS

count <= #`DEL `CBR\_CYC;

mem\_state <= #`DEL REF1;

end

else if (~as\_n) begin

// Загрузка счетчика для отсчитывания времени

// удержания сигнала RAS

count <= #`DEL `RBC\_CYC;

mem\_state <= #`DEL ACCESS;

end

end

ACCESS: begin

mem\_state <= #`DEL SWITCH;

end

SWITCH: begin

if (count == 0) begin

mem\_state <= #`DEL RAS\_CAS;

if (rw)

count <= #`DEL `RACR\_CYC;

else

count <= #`DEL `RACW\_CYC;

end

end

RAS\_CAS:begin

if (count == 0) begin

mem\_state <= #`DEL ACK;

end

end

ACK: begin

mem\_state <= #`DEL IDLE;

end

REF1: begin

if (count == 0) begin

mem\_state <= #`DEL REF2;

count <= #`DEL `RACRF\_CYC;

end

end

REF2: begin

if (count == 0) begin

mem\_state <= #`DEL IDLE;

refresh <= #`DEL 1’b0;

end

end

endcase

end

endmodule // Конец модуля Dram\_control

**/\*Код уровня регистровых передач\*/**

**// ОПРЕДЕЛЕНИЯ КОНСТАНТ**

`define DEL 1 // Задержка распространения сигнала в

// устройстве. Нулевая задержка может

// привести к проблемам

`define RBC\_CYC 2 // Число тактов для удержания сигнала RAS

// перед установкой сигнала CAS

`define CBR\_CYC 1 // Число тактов для удержания сигнала CAS

// перед установкой сигнала RAS

`define RACW\_CYC 1 // Число тактов для удержания сигналов

// RAS и CAS вместе при записи

`define RACR\_CYC 2 // Число тактов для удержания сигналов

// RAS и CAS вместе при считывании

`define RACRF\_CYC 1 // Число тактов для удержания сигналов

// RAS и CAS вместе при регенерации

`define CNT\_BITS 2 // Число тактов, необходимое счетчику

// для подсчитывания представленных

// ниже сигналов

`define REF\_cnt 24 // Число тактов между регенерациями

`define REF\_BITS 5 // Число битов, необходимое счетчику

// для отсчитывания промежутка

// между регенерациями

`define AOUT 4 // Разрядность шины адреса

// со стороны DRAM

`define AIN 2\*`AOUT // Разрядность шины адреса

// со стороны процессора

**// ГЛАВНЫЙ МОДУЛЬ**

module Dram\_control(

clock,

reset\_n,

as\_n,

addr\_in,

addr\_out,

rw,

we\_n,

ras\_n,

cas\_n,

ack);

**// ВХОДЫ**

input clock; // Тактирующий сигнал для

// конечных автоматов

input reset\_n; // Срабатывающий по спаду

// синхронный сигнал сброса reset

input as\_n; // Срабатывающий по спаду

// строб адреса

input [`AIN-1:0] addr\_in; // Адрес со стороны процессора

input rw; // Сигнал выбора запись/считывания

// = 1 соответствует считыванию

// = 0 соответствует записи

**// ВЫХОДЫ**

output [`AOUT-1:0]addr\_out; // Адрес для DRAM

output we\_n; // Сигнал разрешения записи

output ras\_n; // Строб адреса строки для

// памяти DRAM

output cas\_n; // Строб адреса столбца для

// памяти DRAM

output ack; // Сигнал подтверждения

// для процессора

**// ОБЪЯВЛЕНИЯ СИГНАЛОВ**

wire clock;

wire reset\_n;

wire [`AIN-1:0] addr\_in;

wire as\_n;

wire rw;

wire we\_n;

wire ras\_n;

wire cas\_n;

wire ack;

wire [`AOUT-1:0] addr\_out;

wire col\_out; // Адрес колонки/строки

// = 1 – выбор колонки

// = 0 – выбор строки

reg [`CNT\_BITS-1:0] count; // Счетчик тактов

reg [`REF\_BITS-1:0] ref\_count; // Счетчик тактов при

// регенерации

reg refresh; // Запрос на регенерацию

**// ПАРАМЕТРЫ**

// Биты состояний представляют следующие сигналы:

// col\_out, ras, cas, ack

parameter[3:0] // Состояния автомата

IDLE = 4’b0000,

ACCESS = 4’b0100,

SWITCH = 4’b1100,

RAS\_CAS = 4’b1110,

ACK = 4’b1111,

REF1 = 4’b0010,

REF2 = 4’b0110;

**// ОПЕРАТОРЫ ПОТОКОВ ДАННЫХ**

// Пересылка сигналов на управляющие выходы из состояния автомата

assign col\_out = mem\_state[3];

assign ras\_n = ~mem\_state[2];

assign cas\_n = ~mem\_state[1];

assign ack = mem\_state[0];

// Сброс сигнала we\_n в течении регенерации

assign #`DEL we\_n = rw | (mem\_state == REF1) |

(mem\_state == REF2);

// Пересылка адресов столбца или строки на DRAM в зависимости от

// значения сигнала col\_out

assign #`DEL addr\_out = col\_out? addr\_in[`AOUT-1:0] :

addr\_in[`AIN-1:`AOUT];

**// ОСНОВНОЙ КОД**

// Отслеживание событий, управляющих переходом между

// состояниями

always @(posedge clock or negedge reset\_n) begin

if (~reset\_n) begin

mem\_state <= #`DEL IDLE;

count <= #`DEL `CNT\_BITS’h0;

ref\_count <= #`DEL `REF\_CNT;

refresh <= #`DEL 1’b0;

end

else begin

// Наступило ли время запроса на регенерацию?

if (ref\_count == 0) begin

refresh <= #`DEL 1’b1;

ref\_count <= #`DEL `REF\_CNT;

end

else

ref\_count <= #`DEL ref\_count – 1;

// Обнуление счетчика тактов

if (count)

count <= #`DEL count – 1;

case (mem\_state)

IDLE: begin

// Запрос на регенерацию имеет наибольший приоритет

if (refresh) begin

// Загрузка счетчика для отсчитывания

// времени удержания сигнала CAS

count <= #`DEL `CBR\_CYC;

mem\_state <= #`DEL REF1;

end

else if (~as\_n) begin

// Загрузка счетчика для отсчитывания

// времени удержания сигнала RAS

count <= #`DEL `RBC\_CYC;

mem\_state <= #`DEL ACCESS;

end

end

ACCESS: begin

SWITCH: begin

if (count == 0) begin

mem\_state <= #`DEL RAS\_CAS;

if (rw)

count <= #`DEL `RACR\_CYC;

else

count <= #`DEL `RACW\_CYC;

end

end

RAS\_CAS:begin

if (count == 0) begin

mem\_state <= #`DEL ACK;

end

end

ACK: begin

mem\_state <= #`DEL IDLE;

end

REF1: begin

if (count == 0) begin

mem\_state <= #`DEL REF2;

count <= #`DEL `RACRF\_CYC;

end

end

REF2: begin

if (count == 0) begin

mem\_state <= #`DEL IDLE;

refresh <= #`DEL 1’b0;

end

end

endcase

end

end

endmodule // Конец модуля Dram\_control

**Висновок:** під час виконання лабораторної роботи ми отримали знання про функціонування динамічної оперативної пам’яті; отримали навики проектування контролера динамічної оперативної пам’яті. Розроблено програму контролера динамічної оперативної пам’яті з інтерфейсом, який наведено у табл. 1. Проведено моделювання розробленої програми.