Лабораторная работа № 4

Память

1. Цель работы

Изучение способов реализации различных типов памяти на языке SystemVerilog.

2. Краткие теоретические сведения

Обобщённо память можно рассматривать как двумерный массив запоминающих элементов, каждый из которых хранит один бит данных. Содержимое памяти записывается и считывается по строкам. Строка выбирается адресом (Address). Записанные или считанные значения называются данными (Data). Матрица с N-битным адресом и M-битными данными имеет 2^N строк и M столбцов. Каждая строка данных называется *словом*. Таким образом, матрица содержит 2^N M-битных слов. На рисунке 1 приведено условное обозначение такой матрицы памяти.

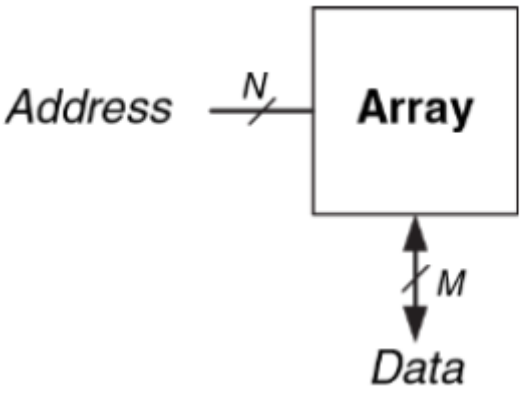

Рисунок 1 – Условное обозначение обобщенной матрицы памяти

Запоминающие устройства классифицируются по способу хранения битов. Запоминающие устройства делятся на два больших класса: оперативные запоминающие устройства (ОЗУ) (RAM, память с произвольным доступом) и постоянные запоминающие устройства (ПЗУ) (ROM, память только для чтения).

Современные ПЗУ не являются постоянными в строгом значении слова: они могут программироваться, т.е. информация в них может записываться. Различие между ОЗУ и ПЗУ состоит в том, что запись в ПЗУ требует больше времени, и они являются энергонезависимыми.

В листинге 1 приведён пример описания ОЗУ размерностью 64 слова по 32 бита. У этого ОЗУ есть синхронный вход разрешения записи. Другими словами, запись в память происходит по переднему фронту тактового импульса, если сигнал разрешения записи (write enable) *we* находится в активном состоянии. Чтение

происходит немедленно. Непосредственно после включения питания содержимое ОЗУ непредсказуемо. Схема такой памяти приведена на рисунке 2.

Листинг 1 Пример реализации ОЗУ на языке SystemVerilog

```
1   module dmem(
2       input logic clk, we,
3       input logic [5:0] a,
4       input logic [31:0] wd,
5       output logic [31:0] rd
6       );
7
8       logic [31:0] RAM[63:0];
9
10      assign rd = RAM[a[5:0]];
11
12      always_ff @(posedge clk)
13         if (we) RAM[a[5:0]] <= wd;
14
15  endmodule
```

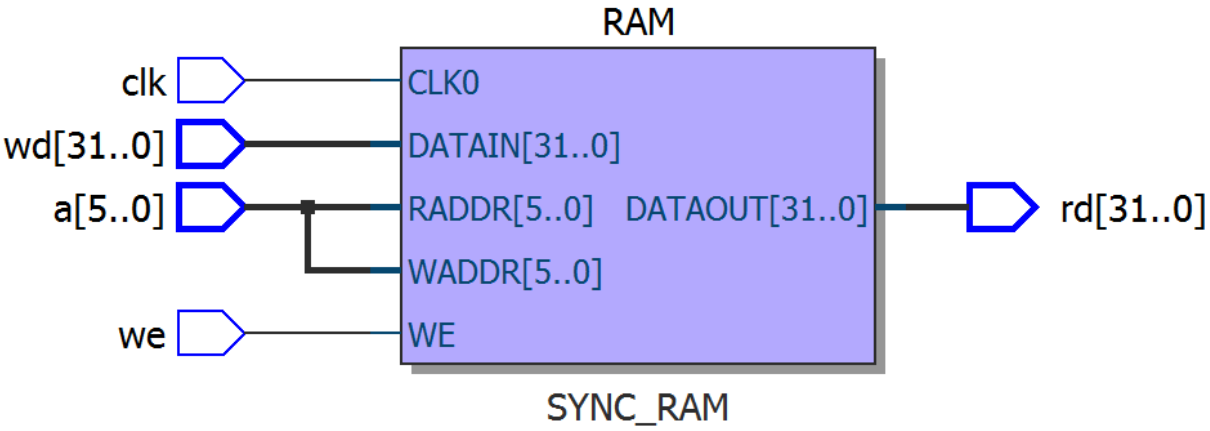

Рисунок 2 – Схема ОЗУ

Пример реализации ПЗУ приведён в листинге 2. Для записи заданных заранее значений нужно в модуль памяти добавить блок initial, в котором, используя оператор $readmemh, указать файл с данными. Схема такой памяти приведена на рисунке 3.

Таким же образом можно инициализировать не только ПЗУ, но и ОЗУ.

Листинг 2 – Пример реализации ПЗУ на языке SystemVerilog

```
module inited_read_only_mem(
    input logic [5:0] a,
    output logic [31:0] rd
    );

    logic [31:0] RAM[63:0];

    initial
       $readmemh("memfile.dat", RAM);

    assign rd = RAM[a]; // word aligned

endmodule
```

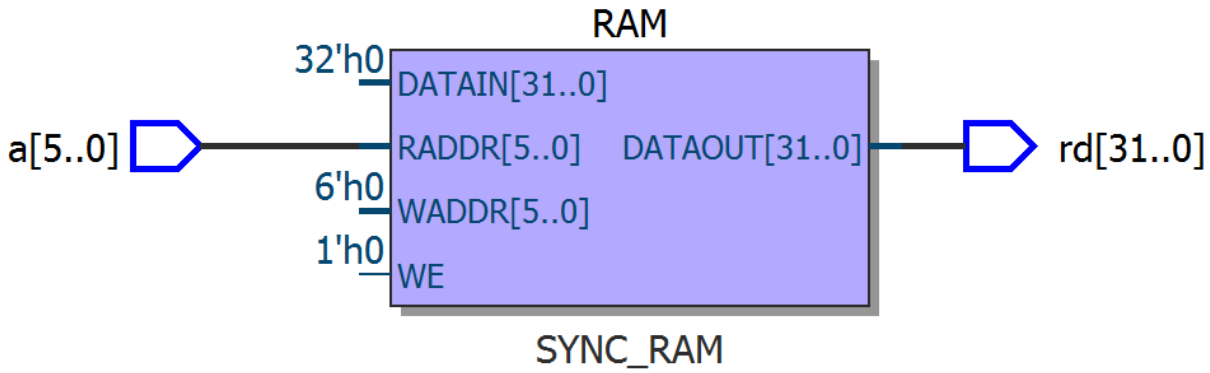

Рисунок 3 – Схема ПЗУ

3. Разработка тестирующего модуля

Для инициализации переменных в тестовый модуль можно после создания экземпляра тестируемого модуля добавить блок initial. Далее значения этих переменных можно менять или не менять в процедурном блоке always. В листинге 3 приведён шаблон тестирующего модуля, в котором задаётся значение входа writeenable равным нулю и нигде не меняется.

Листинг 3 – Шаблон тестирующего модуля с инициализацией начальных значений

```
module testbench();

    logic clk, writeenable;
    logic [31:0] writedata, readdata;
     logic [5:0] dataadr;

    // instantiate device to be tested
     my_memory mem(clk, writeenable, dataadr, writedata, readdata);

```

```
10      // initialize test
11      initial begin
12
13         writeenable = 0;
14         ...
15      end
16
17      // generate clock to sequence tests
18      always begin
19         clk <= 1; # 5; clk <= 0; # 5;
20      end
21
22      // check results
23      always @(negedge clk) begin
24           dataadr <= ...
25           $display(...);
26         if (...) begin
27            $stop;
28         end
29      end
30
31   endmodule
32
```

4. Задание

1. Спроектируйте на языке SystemVerilog модуль 32-разрядной памяти объёмом 64 слова.
2. Напишите для неё тестирующий модуль, в котором проверьте, что в памяти находятся неопределённые значения.
3. Запишите число, отличное от нуля, по адресу N, где N – номер по журналу.
4. Проверьте, что число записалось по заданному адресу, а в остальных ячейках по прежнему находятся неопределённые значения.
5. Спроектируйте на языке SystemVerilog второй модуль 32-разрядной ОЗУ объёмом 32 слова. Проинициализируйте этот модуль заданными значениями. Предварительно расширьте количество записываемых значений до 30 ячеек.

```
00000000
00000000
00000055
00000000
00020007
00000000
00000000
00000044
00000000
00001043
01000008
00000000
00003040
```

6. Скопируйте из проинициализированной памяти значение, записанное по адресу N, в первый блок памяти по адресу N*3.

7. Убедитесь, что значение скопировалось по заданному адресу, а в остальных ячейках находятся неопределённые значения.
8. Спроектируйте на языке SystemVerilog модуль 32-разрядной ПЗУ. Проинициализируйте ПЗУ заданными значениями.
9. Скопируйте значение из ПЗУ с адреса 0’b000010 в ОЗУ по адресу N.
10. Убедитесь, что значение скопировалось по заданному адресу, а в остальных ячейках находятся неопределённые значения.