**module** tb\_instr\_mem;

// Подключение тестируемого модуля

instr\_mem dut (

.addr\_i(addr\_i),

.read\_data\_o(read\_data\_o)

);

// Определение параметров

**parameter** **MEM\_SIZE** = **1024**;

**parameter** **MEM\_WIDTH** = **32**;

// Определение входных сигналов

**logic** [**9**:**0**] addr\_i;

**logic** [**31**:**0**] read\_data\_o;

// Создание счетчика для адресов

**int** **unsigned** count = **0**;

// Тестовый сценарий

**initial** **begin**

// Чтение памяти инструкций с помощью заданного файла

$readmemh("program.mem", dut.memory);

// Генерация адресов и чтение данных

**repeat** (**MEM\_SIZE**) **begin**

// Задаем адрес в модуль

addr\_i = count;

// Делаем паузу для обновления чтения на выходе

#**1**;

// Проверяем считанные данные

**if** (dut.memory[count] === dut.read\_data\_o) **begin**

$display("Test passed for address %0d", count);

**end** **else** **begin**

$display("Test failed for address %0d. Expected: %h, Actual: %h", count, dut.memory[count], dut.read\_data\_o);

**end**

// Увеличиваем счётчик

count = count + **1**;

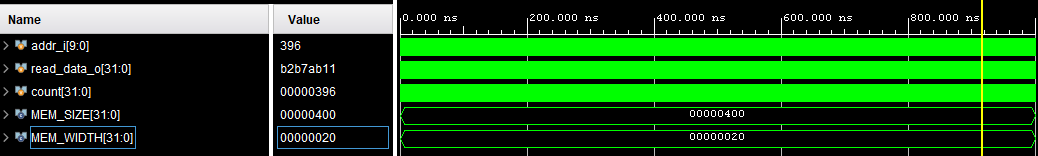
**end**

// Завершение теста

$finish;

**end**

**endmodule**



**module** tb\_data\_mem;

// Параметры

**parameter** **MEM\_SIZE** = **4096**;

**parameter** **MEM\_WIDTH** = **32**;

// Определение сигналов

**logic** clk;

**logic** [**31**:**0**] addr\_i;

**logic** [**31**:**0**] write\_data\_i;

**logic** mem\_req\_i;

**logic** [**31**:**0**] read\_data\_o;

// Создание экземпляра модуля

data\_mem dut (

.clk(clk),

.addr\_i(addr\_i),

.write\_data\_i(write\_data\_i),

.mem\_req\_i(mem\_req\_i),

.read\_data\_o(read\_data\_o)

);

// Инициализация тестового окружения

**initial** **begin**

// Входные значения

clk = **0**;

addr\_i = **32'h0000**\_0000;

write\_data\_i = **32'h1122**\_3344;

mem\_req\_i = **0**;

// Задержка перед началом теста

#**10**;

// Запрос на запись данных в память

addr\_i = **32'h0000**\_0004;

write\_data\_i = **32'h5566**\_7788;

mem\_req\_i = **1**;

#**10**;

mem\_req\_i = **0**;

// Запрос на чтение данных из памяти

addr\_i = **32'h0000**\_0004;

mem\_req\_i = **1**;

#**10**;

// Задержка перед завершением

#**10**;

$finish;

**end**

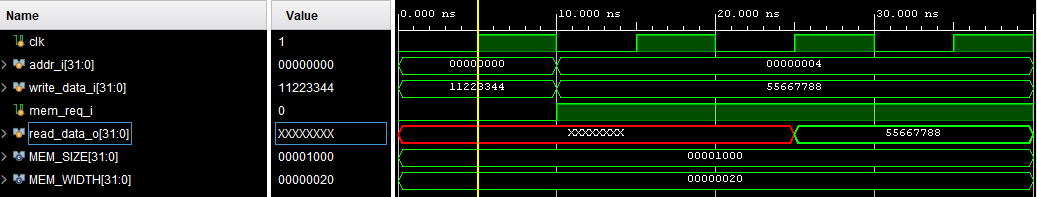
// Генерация тактового сигнала

**always** **begin**

#**5** clk = ~clk;

**end**

**endmodule**



**module** tb\_rf\_riscv;

// Параметры

**parameter** **REG\_FILE\_SIZE** = **1024**;

**parameter** **REG\_WIDTH** = **32**;

// Определение сигналов

**logic** clk;

**logic** [**9**:**0**] addr\_1;

**logic** [**9**:**0**] addr\_2;

**logic** [**9**:**0**] write\_addr;

**logic** write\_enable;

**logic** [**31**:**0**] write\_data;

**logic** [**31**:**0**] read\_data\_1;

**logic** [**31**:**0**] read\_data\_2;

// Создание экземпляра модуля

rf\_riscv dut (

.addr\_1(addr\_1),

.addr\_2(addr\_2),

.write\_addr(write\_addr),

.write\_enable(write\_enable),

.write\_data(write\_data),

.read\_data\_1(read\_data\_1),

.read\_data\_2(read\_data\_2)

);

// Инициализация тестового окружения

**initial** **begin**

// Входные значения

clk = **0**;

addr\_1 = **10'h000**;

addr\_2 = **10'h001**;

write\_addr = **10'h002**;

write\_enable = **1**;

write\_data = **32'h11223344**;

// Задержка перед началом теста

#**10**;

// Запись данных в регистровый файл

write\_enable = **1**;

#**10**;

write\_enable = **0**;

// Запрос на чтение данных из регистрового файла

#**10**;

addr\_1 = **10'h002**;

addr\_2 = **10'h003**;

// Задержка перед завершением

#**10**;

$finish;

**end**

// Генерация тактового сигнала

**always** **begin**

#**5** clk = ~clk;

**end**

**endmodule**

