实验名称：实验七 存储器

姓名：张涵之

学号：191220154

班级：周一 5-6

邮箱：191220154@smail.nju.edu.cn

实验时间：2020/10/18

7.2 存储器的实现

思考题：
如果将表 7-2 中存储器实现部分 assign dout = ram[outaddr];
改为 else dout <= ram[outaddr]; 该存储器的行为是否会发生变化？

该存储器的行为会发生变化。修改前在 if-else 语句外使用 assign 语句赋值，当 outaddr 发生变化时，无论使能端如何，输出的 dout 立即得到新地址对应的值；修改后在 if-else 语句内使用非阻塞赋值，则若 outaddr 发生变化，当且仅当使能端无效时输出的 dout 改变，且在下一个时钟周期到来才发生变化，读入新地址存储的值。

实验原理：利用开发板上的频率为 50MHz 的时钟，参考课件中 1 秒时钟生成代码设计一个分频器，其输入为 50MHz 的时钟，输出为一个频率为 1Hz，周期为 1 秒的时钟信号。用这个新的频率为 1Hz 的时钟信号作为设计的时钟信号，进行计数。
计时器的开始和暂停用 SW0 控制，清零功能用 SW1 控制，两个时钟信号分别用两个四位二进制数表示，其中个位逢九清零，十位进一，个位和十位均为九时双双清零，LEDR0 亮，判断清零实现之后，LEDR0 灭。个位与十位在七段数码管上分别显示。

7.2.1 存储器实例分析

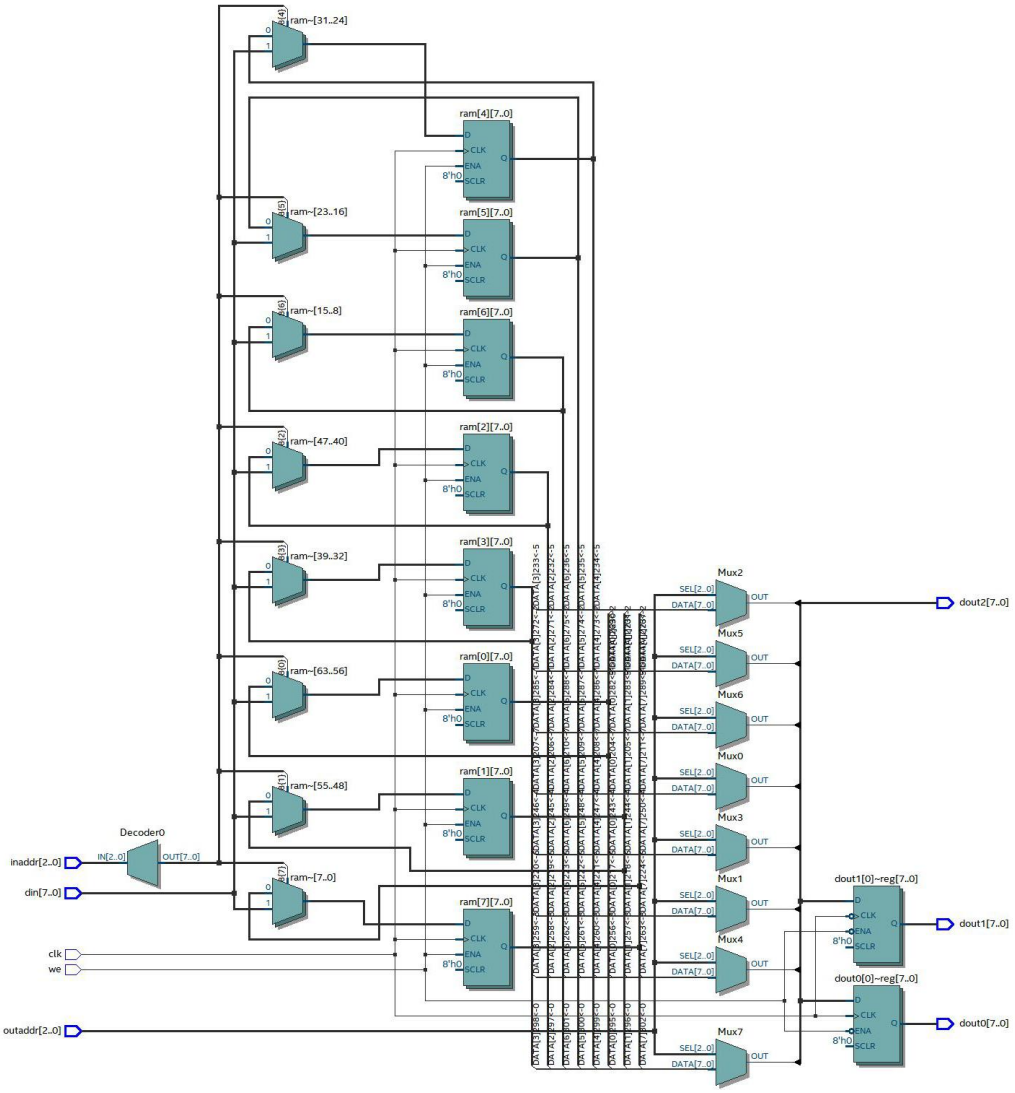
表 7-3 是一个存储器实例，请分析存储器结构和工作过程，查看此存储器的 RTL 图，检查存储器的输入输出和存储体的结构，并分析其三个输出端的结构的不同。

存储器实例如图：

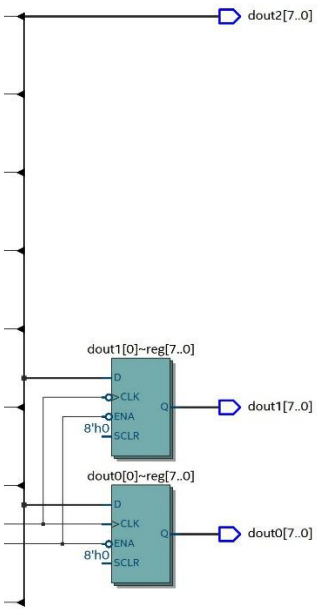
```
module v_rams_8(clk,we,inaddr,outaddr,din,dout0,dout1,dout2);
   input clk;
   input we;
   input [2:0] inaddr;
   input [2:0] outaddr;
   input [7:0] din;
   output [7:0] dout0,dout1,dout2;

   reg [7:0] ram [7:0];
   reg [7:0] dout0,dout1;

   initial
   begin
   ram[7] = 8'hf0; ram[6] = 8'h23; ram[5] = 8'h20; ram[4] = 8'h50;
   ram[3] = 8'h03; ram[2] = 8'h21; ram[1] = 8'h82; ram[0] = 8'h0D;
   end

   always @(posedge clk)
   begin
      if (we)
         ram[inaddr] <= din;
      else
         dout0 <= ram[outaddr];
   end
   always @(negedge clk)
   begin
      if (!we)
         dout1 <= ram[outaddr];
   end
   assign dout2 = ram[outaddr];
endmodule
```

分析结构和工作过程：
输入地址 inaddr 通过解码器，输出用于 ram 的选择，每个 m 接 1 寄存器，选择器用于判断是录入 din 还是维持原数据，outaddr 与寄存器的输出接入多路选择器，对 dout 需要录入的数据进行判断，另外两个寄存器决定了 dout0、dout1 和 dout2 的读入方式。

存储器的 RTL 图如图：



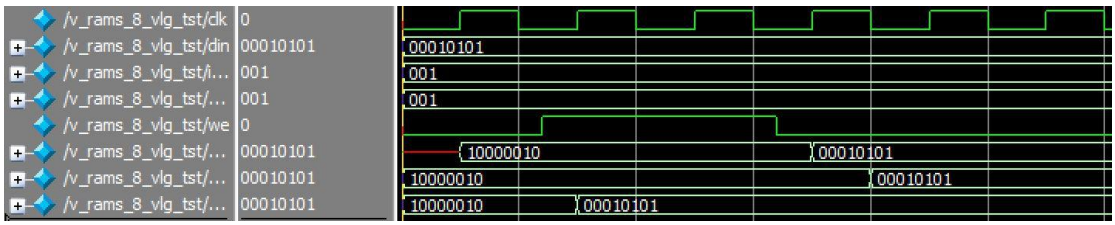
观察三个输出端的结构：



可见 dout2 不受使能端和时钟信号控制，输入 outaddr 立即读入相应下标的 ram。
另两者受使能端控制，dout0 的寄存器接入时钟信号，dout1 接入取反后的时钟信号。

为此实例设计一个测试代码，研究此三个端口输出数据时在时序上的差别，
结合 RTL 图，给出其工作时序的解释。

```
initial
begin
// code that executes only once
// insert code here --> begin
   clk = 0; din = 8'h15; inaddr = 3'b1; outaddr = 3'b1; we = 0; #12;
                                                        we = 1; #20;
                                                        we = 0; #50;
   $stop;
// --> end
//$display("Running testbench");
end
always
// optional sensitivity list
// @(event1 or event2 or .... eventn)
begin
// code executes for every event on sensitivity list
// insert code here --> begin
   #5; clk = ~clk;
//@eachvec;
// --> end
end
endmodule
```



结合 RTL 图可再次证实三个输出端：
dout0：有使能端，在时钟上升沿且使能为 0 时沿 D 触发器的 Q 端输出。
dout1：有使能端，在时钟下降沿且使能为 0 时沿 D 触发器的 Q 端输出。
dout2：无使能端，不受时钟控制，随着 outaddr 的变化而变化，直接输出。

7.4 实验内容

请在一个工程中完成如下两个存储器。两个存储器的大小均为 16×8，即每个存储器共有 16 个存储单元，每个存储单元都是 8 位的，均可以进行读写。

RAM1：初始化，输出端有缓存，输出地址有效后，时钟信号的上升沿到来才输出数据。

RAM2：利用 IP 核设计一个双口存储器，利用.mif 文件进行初始化，十六个单元的初始化值分别为：0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6,0xf7,0xf8,0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff。

此两个物理上完全不同的存储器共用时钟、读写地址和写使能信号，当写使能有效时，在时钟信号的有效沿写入数据；当写使能信号无效时，在时钟信号的有效沿输出数据。适当选择时钟信号和写使能信号，以能够分别对此两个存储器进行读写。

使用 FPGA 开发板的输入/输出资源完成此存储器的设计。写入时可以只写入 2 位数据。

实验目的：完成两个存储器。

实验原理：仿照课件设计，用七段数码管显示输出。

实验环境/器材：实验箱一个，笔记本电脑一台。

程序代码或流程图：

```
module ram_1(clk,we,inaddr,outaddr,din,dout);
   input clk;
   input we;
   input [2:0] inaddr;
   input [2:0] outaddr;
   input [2:0] din;
   output reg [7:0] dout;

   reg [7:0] ram [15:0];
   reg [7:0] temp;
   reg flag = 0;

   initial begin
      $readmemh("D:/workspace/exp7_2/mem1.txt",ram,0,15);
   end

   always @(posedge clk) begin
      if (we) begin
         ram[inaddr][2:0] <= din;
         ram[inaddr][7:3] <= 6'b000000;
      end
      if (flag) begin
         dout <= temp;
         flag = 0;
      end
      else begin
         temp <= ram[outaddr];
         flag = 1;
      end
   end
endmodule
```

```
module hex(in,out);
   input [3:0] in;
   output reg [6:0] out;

always @ (*)
   case(in)
      0: out = 7'b1000000;
      1: out = 7'b1111001;
      2: out = 7'b0100100;
      3: out = 7'b0110000;
      4: out = 7'b0011001;
      5: out = 7'b0010010;
      6: out = 7'b0000010;
      7: out = 7'b1111000;
      8: out = 7'b0000000;
      9: out = 7'b0010000;
      10: out = 7'b0001000;
      11: out = 7'b0000011;
      12: out = 7'b1000110;
      13: out = 7'b0100001;
      14: out = 7'b0000110;
      15: out = 7'b0001110;
      default: out = 7'b1111111;
   endcase
endmodule
```

```
module exp7_2(clk,we,inaddr,outaddr,din,hex1_1,hex1_2,hex2_1,hex2_2);
   input clk;
   input we;
   input [2:0] inaddr;
   input [2:0] outaddr;
   input [2:0] din;
   output [6:0] hex1_1;
   output [6:0] hex1_2;
   output [6:0] hex2_1;
   output [6:0] hex2_2;

   wire [7:0] dout1;
   wire [7:0] dout2;

   ram_1 r1(clk,we,inaddr,outaddr,din,dout1);
   ram_2 r2(clk,din,outaddr,inaddr,we,dout2);
   hex h1_1(dout1[3:0],hex1_1);
   hex h1_2(dout1[7:4],hex1_2);
   hex h2_1(dout2[3:0],hex2_1);
   hex h2_2(dout2[7:4],hex2_2);

endmodule
```

实验步骤/过程：
仿照课件中的代码进行代码编写和初始化，其中发现输出端有缓存，输出地址有效后，时钟信号的上升沿到来才输出数据。因此额外设置标志位，获得输出地址后，延迟一个时钟周期输出，以模拟利用 IP 核设计，.mif 初始化的双口寄存器的功能。
仿照课件中的操作流程做出用 IP 核设计，.mif 初始化的双口寄存器。
两个寄存器共用使能端、时钟信号、输入输出地址、输入数据，输出信号分别接入 hex 进行转换，两个八位二进制数分别接入两个七段数码管，以十六进制格式显示。

测试方法：在开发板上观察数码管输出。

实验结果：通过观察，开发板上读写模式下的输出显示均符合预期。

实验中遇到的问题及解决办法：实验非常顺利，没有遇到任何问题。

实验得到的启示：无。

意见和建议：无。