8-5 分别用原理图形式和用 Verilog 代码设计一个具有 4 通道的三态总线控制电路，可分别对四个 8 位 LPM_RAM 进行数据读取和写入操作。仿真后证明设计的正确性

解：

（1）解题思路

8-5：

Verilog 代码设计：

quartus II 新建工程，顶层文件 tribus4.v。

之后新建文件 RAM78.v，add file to project。（参考 P173 例 7-6）

初始化采用 mif 文件。

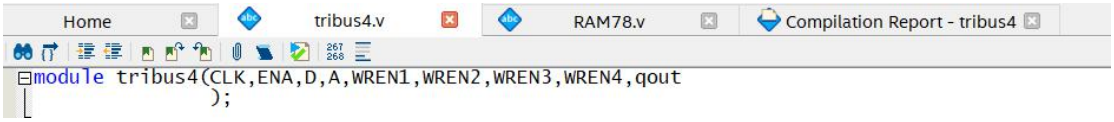
mif 文件的设计方法为 file new Memory Initialization File，地址线宽为 7 位，故 Number128. Word size 8 位。保存到工程文件夹。

在 tribus4 中调用 RAM78。

编译。

编写 testbench。

modelsim 仿真。

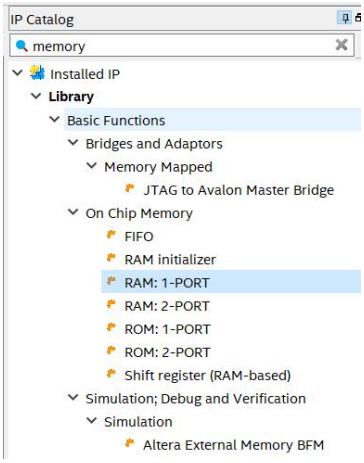


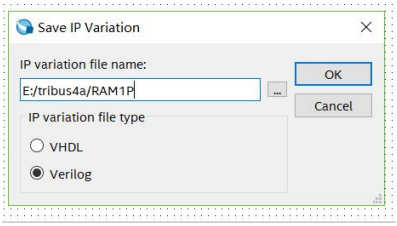
原理图形式：

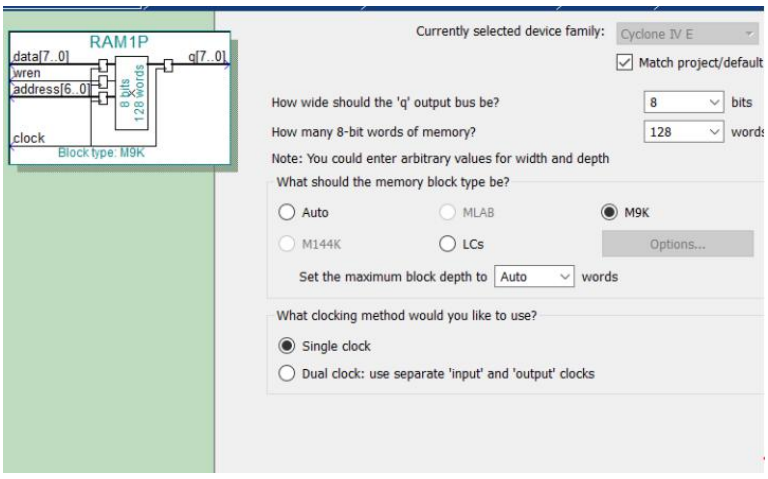
以原理图方式对 LPM_RAM 进行设置和调用：

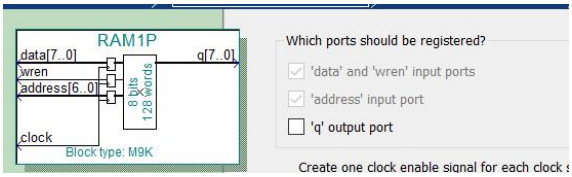
在 ipcatalog 搜索 momery

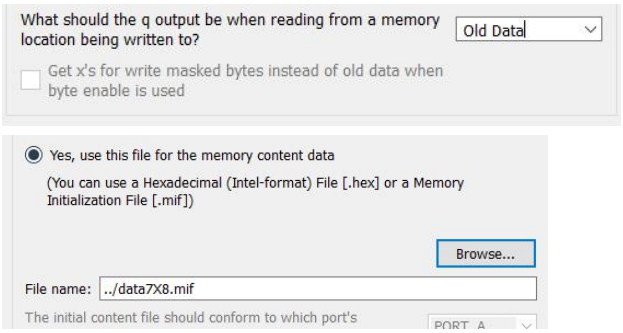
选择 RAM 1 PORT


IP Catalog
memory
Installed IP
Library
Basic Functions
Bridges and Adaptors
Memory Mapped
JTAG to Avalon Master Bridge
On Chip Memory
FIFO
RAM initializer
RAM: 1-PORT
RAM: 2-PORT
ROM: 1-PORT
ROM: 2-PORT
Shift register (RAM-based)
Simulation; Debug and Verification
Simulation
Altera External Memory BFM

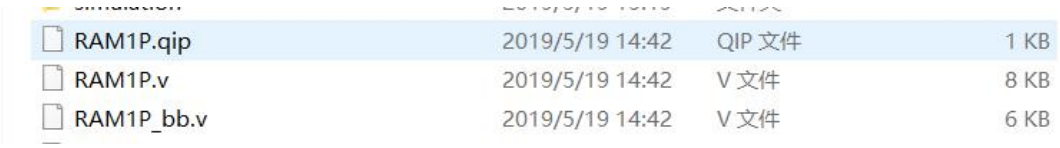

Save IP Variation
IP variation file name:
E:/tribus4a/RAM1P
OK
Cancel
IP variation file type
VHDL
Verilog


RAM1P
data[7..0]
wren
address[6..0]
clock
q[7..0]
8 bits
128 words
Block type: M9K
Currently selected device family:
Cyclone IV E
Match project/default
How wide should the 'q' output bus be?
8
bits
How many 8-bit words of memory?
128
words
Note: You could enter arbitrary values for width and depth
What should the memory block type be?
Auto
MLAB
M9K
M144K
LCs
Options...
Set the maximum block depth to
Auto
words
What clocking method would you like to use?
Single clock
Dual clock: use separate 'input' and 'output' clocks

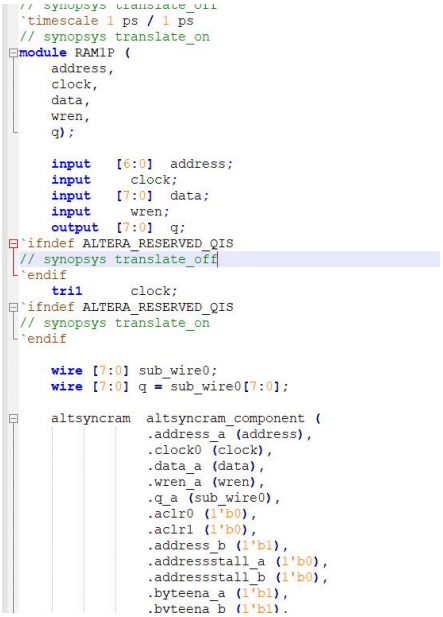

RAM1P
data[7..0]
wren
address[6..0]
clock
q[7..0]
8 bits
128 words
Block type: M9K
Which ports should be registered?
'data' and 'wren' input ports
'address' input port
'q' output port
Create one clock enable signal for each clock



这时设计成功后会在工程文件夹生成 RAM1P.v 文件。



用编辑器打开可以查看，和之前用 verilog 代码设计的 RAM78 功能相同。



```
// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module RAM1P (
    address,
    clock,
    data,
    wren,
    q);

    input   [6:0]  address;
    input     clock;
    input   [7:0]  data;
    input     wren;
    output  [7:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
    tri1      clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

    wire [7:0] sub_wire0;
    wire [7:0] q = sub_wire0[7:0];

    altsyncram  altsyncram_component (
                .address_a (address),
                .clock0 (clock),
                .data_a (data),
                .wren_a (wren),
                .q_a (sub_wire0),
                .aclr0 (1'b0),
                .aclr1 (1'b0),
                .address_b (1'b1),
                .addressstall_a (1'b0),
                .addressstall_b (1'b0),
                .byteena_a (1'b1),
                .byteena_b (1'b1),
```

之后按照 Verilog 代码形式的 RTL 图设计原理图。同时对 Verilog 代码形式的 testbench 稍作修改。

仿真即可。

（2）核心模块代码

8-5

Verilog 实现：

tribus4.v

```
module tribus4(CLK,ENA,D,A,WREN1,WREN2,WREN3,WREN4,qout
               );

input CLK,WREN1,WREN2,WREN3,WREN4;
input [7:0] D;
input[6:0] A; //address
input [1:0] ENA;
output [7:0] qout;
reg[7:0] qout;

wire [7:0] Q1,Q2,Q3,Q4;


RAM78 U1 (.Q(Q1),.D(D),.A(A),.CLK(CLK),.WREN(WREN1));   //ENA 选择 ram，选中后读取数据 Q。当 wren，读取的同时在相同位置写入数据 D

RAM78 U2 (.Q(Q2),.D(D),.A(A),.CLK(CLK),.WREN(WREN2));    //每个 ram 有独立的 wren。不用 ENA 选择写入哪个 RAM，只选择读哪个。

RAM78 U3 (.Q(Q3),.D(D),.A(A),.CLK(CLK),.WREN(WREN3));

RAM78 U4 (.Q(Q4),.D(D),.A(A),.CLK(CLK),.WREN(WREN4));

always@(ENA,Q1)
if(ENA==2'b00)
qout=Q1;
else  qout=8'hz;
always@(ENA,Q2)
if(ENA==2'b01)
qout=Q2;
else  qout=8'hz;
always@(ENA,Q3)
if(ENA==2'b10)
qout=Q3;
else  qout=8'hz;
always@(ENA,Q4)
if(ENA==2'b11)
qout=Q4;
else  qout=8'hz;

endmodule
```

RAM78.v

```
module RAM78(output wire[7:0] Q,input wire[7:0] D,input wire [6:0] A,input wire
CLK,WREN);
reg[7:0] mem[127:0] /* synthesis ram_init_file="data7X8.mif"*/;
always@(posedge CLK)
if(WREN) mem[A] <= D;
assign Q=mem[A];
endmodule
```

Data7X8.mif

```
-- Copyright (C) 2017  Intel Corporation. All rights reserved.
-- Your use of Intel Corporation's design tools, logic functions
-- and other software and tools, and its AMPP partner logic
-- functions, and any output files from any of the foregoing
-- (including device programming or simulation files), and any
-- associated documentation or information are expressly subject
-- to the terms and conditions of the Intel Program License
-- Subscription Agreement, the Intel Quartus Prime License Agreement,
-- the Intel FPGA IP License Agreement, or other applicable license
-- agreement, including, without limitation, that your use is for
-- the sole purpose of programming logic devices manufactured by
-- Intel and sold by Intel or its authorized distributors.  Please
-- refer to the applicable agreement for further details.

-- Quartus Prime generated Memory Initialization File (.mif)

WIDTH=8;
DEPTH=128;

ADDRESS_RADIX=UNS;
DATA_RADIX=HEX;

CONTENT BEGIN
    0    :   3D;
    1    :   1B;
    2    :   14;
    3    :   12;
    4    :   13;
    5    :   15;
    6    :   16;
    7    :   17;
```

```
    8    :   4A;
    9    :   18;
    10   :   20;
    11   :   19;
    12   :   21;
    13   :   22;
    14   :   23;
    15   :   24;
    16   :   25;
    17   :   26;
    18   :   27;
    19   :   28;
    20   :   29;
    21   :   30;
    22   :   31;
    23   :   32;
    24   :   33;
    25   :   34;
    26   :   35;
    27   :   36;
    28   :   37;
    29   :   38;
    30   :   39;
    31   :   40;
    32   :   41;
    33   :   42;
    34   :   43;
    35   :   44;
    36   :   45;
    37   :   46;
    38   :   47;
    39   :   48;
    [40..127]  :   00;
END;
```

原理图形式：

RAM1P.v

```
// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module RAM1P (
```

```
    address,
    clock,
    data,
    wren,
    q);

    input   [6:0]  address;
    input     clock;
    input   [7:0]  data;
    input     wren;
    output  [7:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
    tri1      clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

    wire [7:0] sub_wire0;
    wire [7:0] q = sub_wire0[7:0];

    altsyncram  altsyncram_component (
                .address_a (address),
                .clock0 (clock),
                .data_a (data),
                .wren_a (wren),
                .q_a (sub_wire0),
                .aclr0 (1'b0),
                .aclr1 (1'b0),
                .address_b (1'b1),
                .addressstall_a (1'b0),
                .addressstall_b (1'b0),
                .byteena_a (1'b1),
                .byteena_b (1'b1),
                .clock1 (1'b1),
                .clocken0 (1'b1),
                .clocken1 (1'b1),
                .clocken2 (1'b1),
                .clocken3 (1'b1),
                .data_b (1'b1),
                .eccstatus (),
                .q_b (),
                .rden_a (1'b1),
```

```
                .rden_b (1'b1),
                .wren_b (1'b0));
    defparam
        altsyncram_component.clock_enable_input_a = "BYPASS",
        altsyncram_component.clock_enable_output_a = "BYPASS",
        altsyncram_component.init_file = "../data7X8.mif",
        altsyncram_component.intended_device_family = "Cyclone IV E",
        altsyncram_component.lpm_hint                                          =
"ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=MYRM",
        altsyncram_component.lpm_type = "altsyncram",
        altsyncram_component.numwords_a = 128,
        altsyncram_component.operation_mode = "SINGLE_PORT",
        altsyncram_component.outdata_aclr_a = "NONE",
        altsyncram_component.outdata_reg_a = "UNREGISTERED",
        altsyncram_component.power_up_uninitialized = "FALSE",
        altsyncram_component.ram_block_type = "M9K",
        altsyncram_component.read_during_write_mode_port_a = "OLD_DATA",
        altsyncram_component.widthad_a = 7,
        altsyncram_component.width_a = 8,
        altsyncram_component.width_byteena_a = 1;


endmodule
```

（3）测试模块代码

8-5

针对 Verilog 代码设计的测试代码：

```
`timescale 1ns/1ps
module tribus4_test_tb;



    reg CLK,WREN1,WREN2,WREN3,WREN4;
    reg [7:0] D;
    reg [6:0] A;
    reg[1:0] ENA;
    wire [7:0]qout;

    tribus4 DUT (
    .CLK(CLK),.ENA(ENA),.D(D),.A(A),.WREN1(WREN1),.WREN2(WREN2),.WREN3(WREN3),.
WREN4(WREN4),.qout(qout)
    );

    initial begin
```

```
        WREN1=2'b0; WREN2=2'b0; WREN3=2'b0; WREN4=2'b0;
        A=7'b0000000;
        D = 8'b00000000;
        CLK=2'b0;
        ENA=2'b00;

    end
        always #80 ENA=(ENA+1)%4;
        always #10 CLK=~CLK;
        always #100 D=D+1;
        always #200 A=A+1;
        always #30 WREN1=~WREN1;
        always #40 WREN2=~WREN2;
        always #50 WREN3=~WREN3;
        always #60 WREN4=~WREN4;
endmodule
```

针对原理图设计的测试代码：

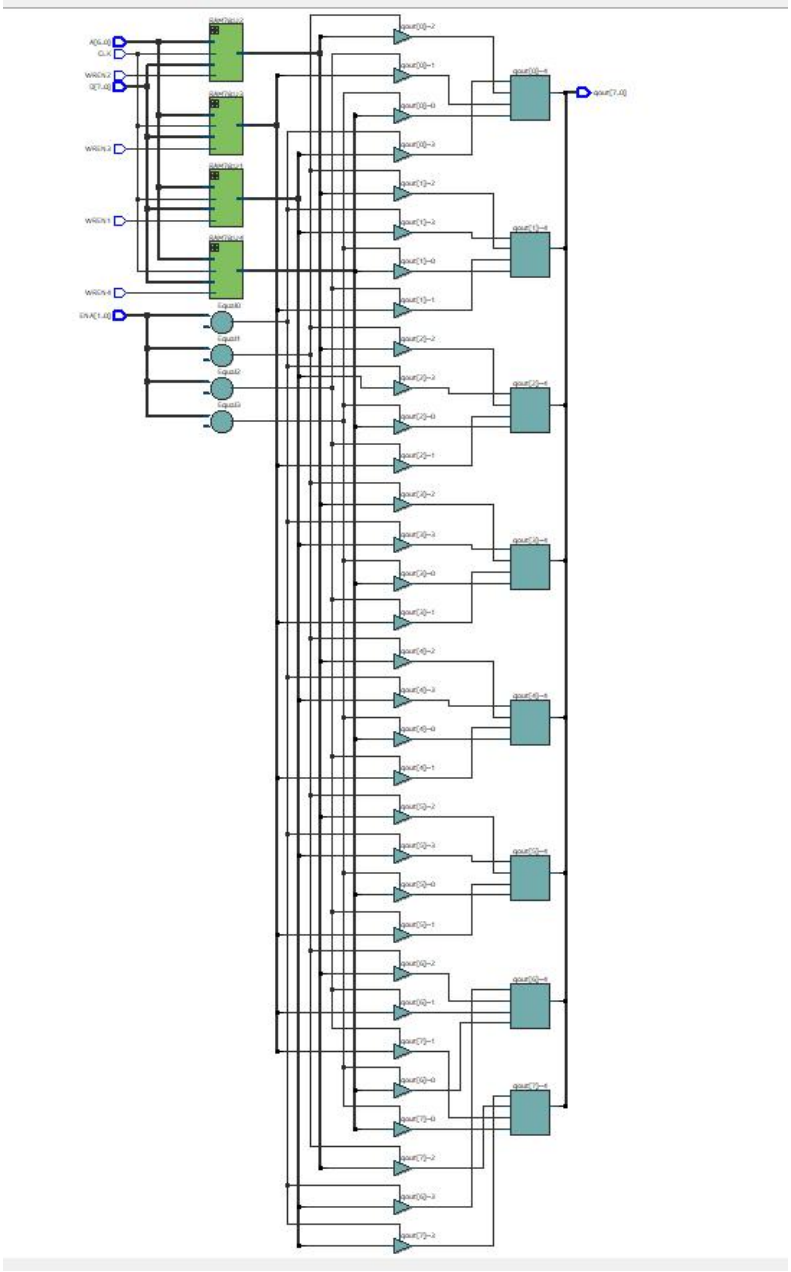
```
`timescale 1ns/1ps
module tribus4a_test_tb;



    reg CLK,WREN1,WREN2,WREN3,WREN4;
    reg [7:0] D;
    reg [6:0] A;
    reg[1:0] ENA;
    wire [7:0]qout;

    tribus4a DUT (
    .CLK(CLK),.ENA(ENA),.D(D),.A(A),.WREN1(WREN1),.WREN2(WREN2),.WREN3(WREN3),.
WREN4(WREN4),.qout(qout)
    );

    initial begin
        WREN1=2'b0; WREN2=2'b0; WREN3=2'b0; WREN4=2'b0;
        A=7'b0000000;
        D = 8'b00000000;
        CLK=2'b0;
        ENA=2'b00;

        /*#100 ENA=2'b01;
            #100 ENA=2'b10;
                #100 ENA=2'b11;
                    #100 ENA=2'b00;
```

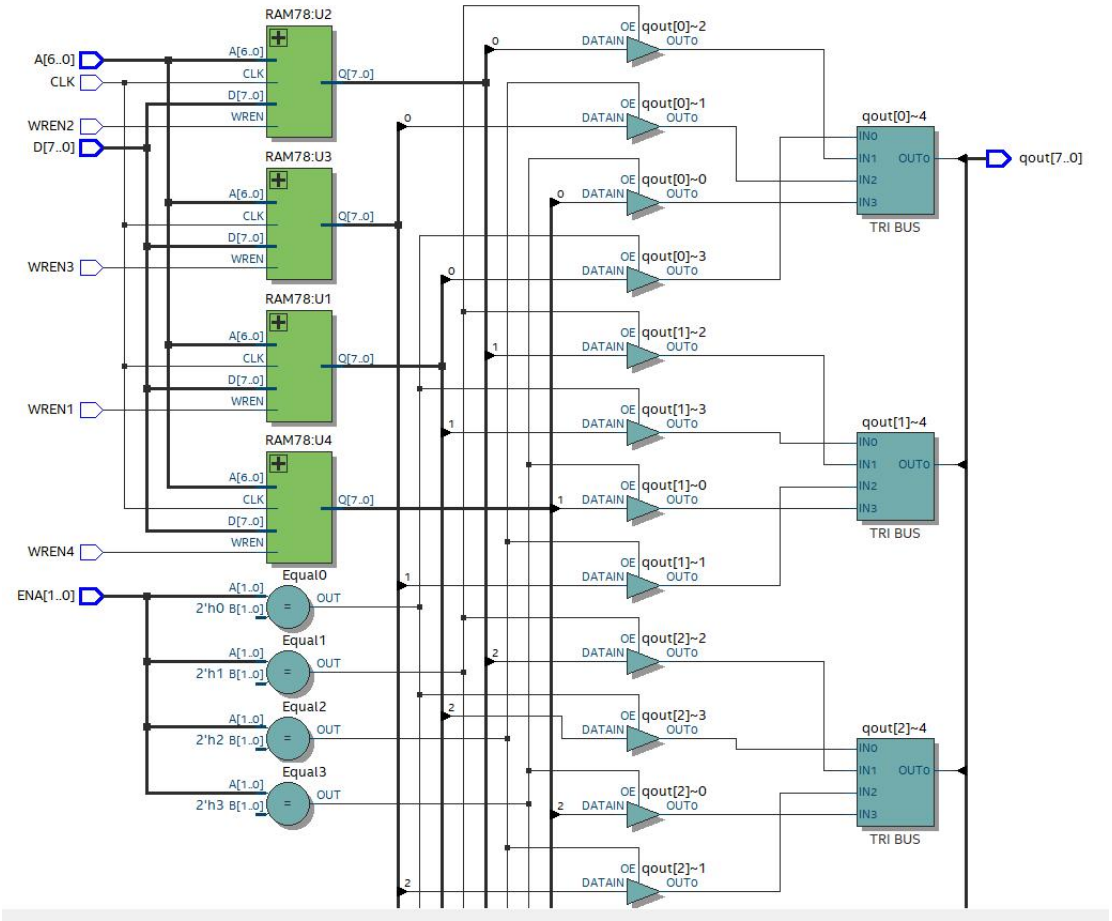
```
        #100 ENA=2'b01;
            #100 ENA=2'b10;
                #100 ENA=2'b11;
                    #100 ENA=2'b00;

        #100 ENA=2'b01;
            #100 ENA=2'b10;
                #100 ENA=2'b11;
                    #100 ENA=2'b00;

        #100 ENA=2'b01;
            #100 ENA=2'b10;
                #100 ENA=2'b11;
                    #100 ENA=2'b00;  */



    end
        always #80 ENA=(ENA+1)%4;
        always #10 CLK=~CLK;
        always #100 D=D+1;
        always #200 A=A+1;
        always #30 WREN1=~WREN1;
        always #40 WREN2=~WREN2;
        always #50 WREN3=~WREN3;
        always #60 WREN4=~WREN4;




endmodule
```

（4）RTL View 的网表图

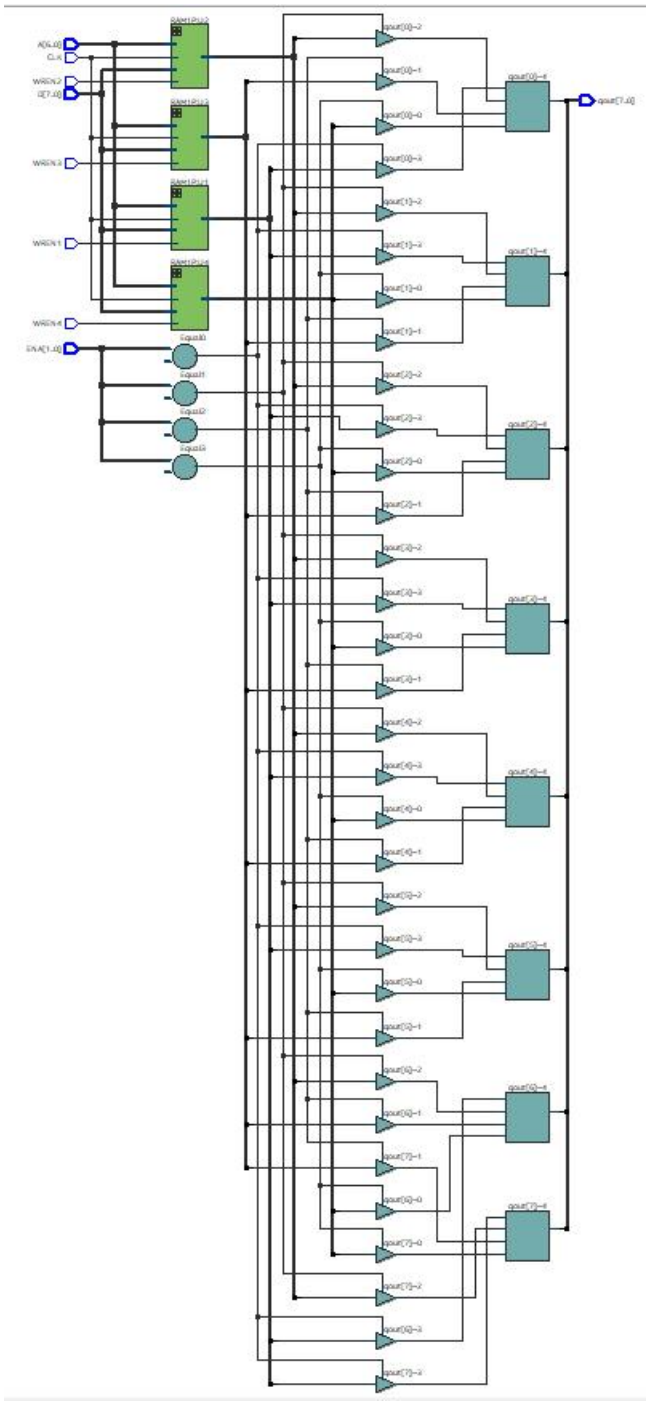
8-5

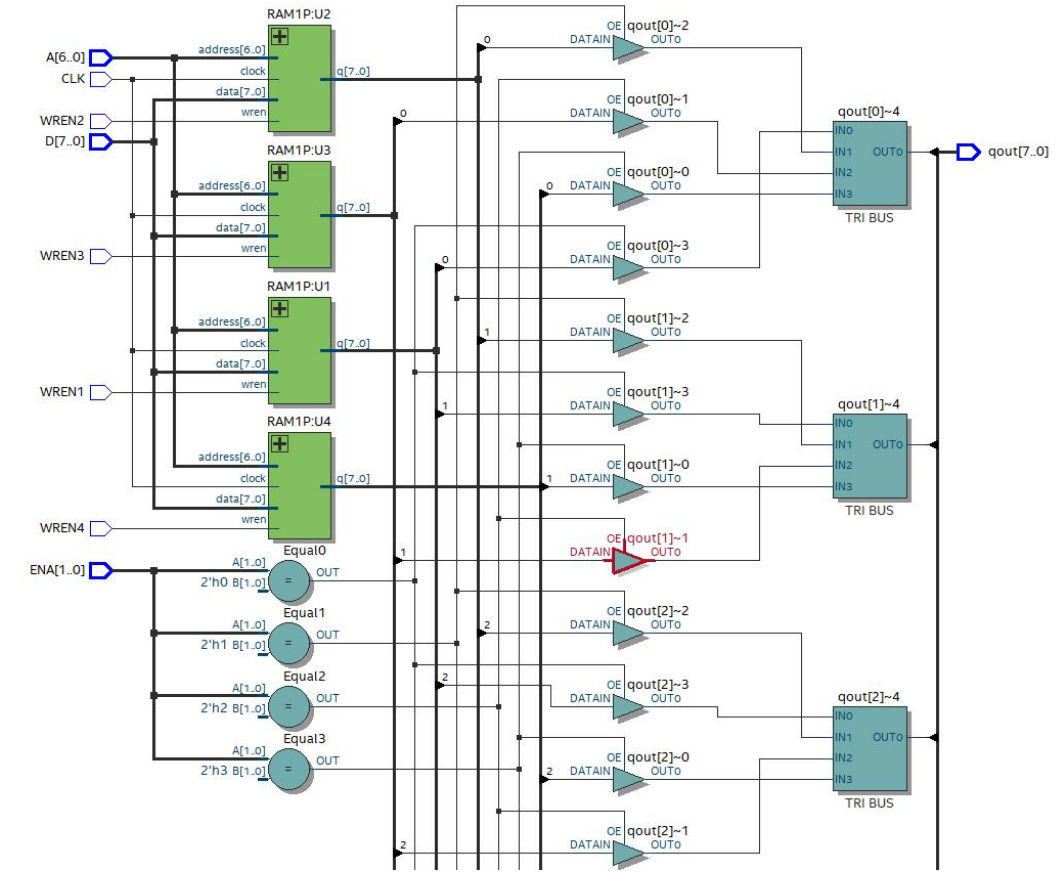
Verilog 代码设计：




RAM78:U2
RAM78:U3
RAM78:U1
RAM78:U4
A[6..0]
CLK
D[7..0]
WREN
Q[7..0]
WREN2
WREN3
WREN1
WREN4
ENA[1..0]
Equal0
Equal1
Equal2
Equal3
A[1..0]
B[1..0]
2'h0
2'h1
2'h2
2'h3
OUT
OE
DATAIN
OUTO
qout[0]~2
qout[0]~1
qout[0]~0
qout[0]~3
qout[1]~2
qout[1]~3
qout[1]~0
qout[1]~1
qout[2]~2
qout[2]~3
qout[2]~0
qout[2]~1
qout[0]~4
qout[1]~4
qout[2]~4
IN0
IN1
IN2
IN3
TRI BUS
qout[7..0]

原理图设计：

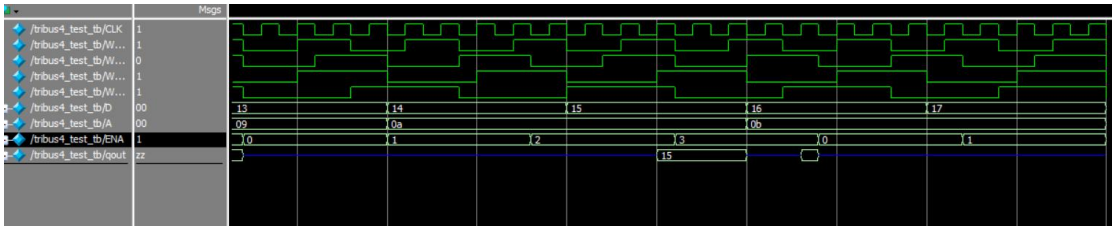




（5）ModelSim 仿真的结果图

8-5

Verilog 代码设计的仿真：



原理图设计的仿真：