第 8 章　宏功能模块与 IP 核应用

为了减轻设计开发者的工作量，设计厂商在 EDA 开发软件中提供了一些可配置使用的 IP（Intellectual Property）核，包括基本功能模块、运算单元、存储单元、时钟锁相环等，使用这些 IP 核不仅能够大大减轻开发者的工作量，提高工作效率，更能提高系统可靠性。Altera 公司 Quartus II 开发软件提供的 Mega Wizard Plug-In Manager 工具、Xilinx 公司 ISE 开发软件提供的 Core Generator 工具均可进行 IP 核的配置调用。

本章将介绍使用 Quartus II 开发软件进行 LPM（Library of Parameterized modules）模块的调用。LPM 为参数可设置模块库，相当于 IP 核，用户通过例化参数，设置模块的功能，满足设计需求。下面将详细介绍 LPM_RAM、LPM_ROM、LPM_PLL 模块的定制使用，然后介绍片内逻辑分析仪工具 SignalTap II 的应用。

8.1　LPM_RAM

本节将介绍在 Quartus II 中利用 Mega Wizard Plug-In Manager 工具设计单端口 RAM 模块。

8.1.1　LPM_RAM 宏模块定制

1. 打开 MegaWizard Plug-In Manager

在如图 3-30 所示 Quartus II 界面中，选择“Tools”→“MegaWizard Plug-In Manager”命令，弹出如图 8-1 所示的对话框，选中“Create a new custom megafunction variation”选项，单击“Next”按钮，弹出宏模块功能设置窗口，如图 8-2 所示。

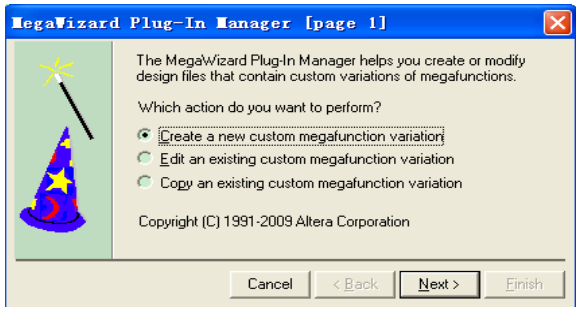

图 8-1　定制新的宏模块

单击图 8-2 左侧的“Memory Compiler”目录，在展开的器件选项中选择“RAM:1-PORT”，此模块为单端口 RAM 存储器，在目标芯片系列栏选择 Cyclone 系列，输出文件类型选择 VHDL，输入存放目录及文件名选择 E:\ram_test\ram_lpm，调用的 RAM 名称为 ram_lpm。

2. RAM 参数配置

在图 8-2 中单击“Next”按钮，弹出如图 8-3 所示 RAM 参数配置窗口。选择存储数据宽度为 8，存储容量为 32，其他为默认配置。窗口左侧显示了调用 RAM 模块的端口信号：8 位

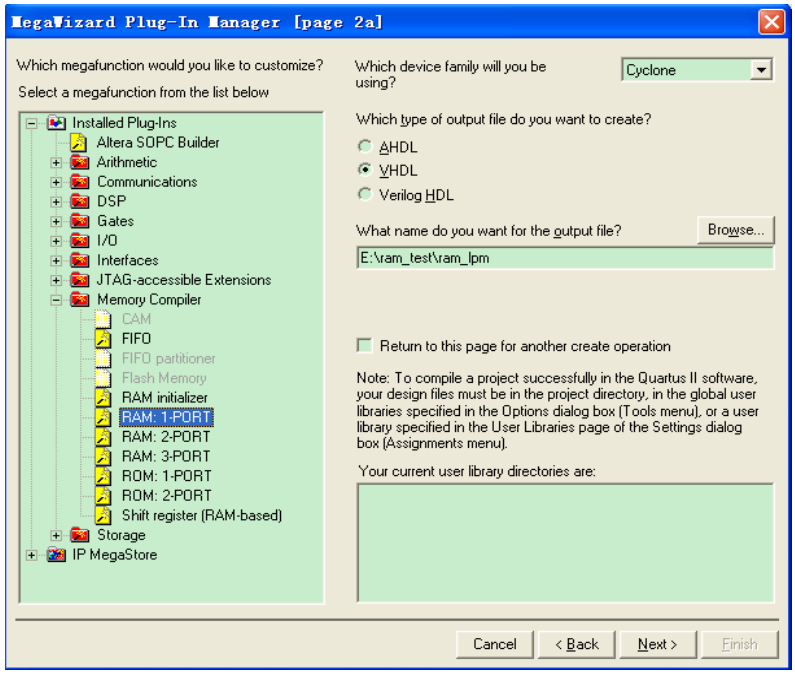

图 8-2　宏模块功能设置窗口

数据输入信号 data；读/写使能信号 wren，wren 为高电平表示写使能有效，wren 为低电平表示读使能有效；5 位地址输入信号 address；时钟信号 clock；8 位数据输出信号 q。

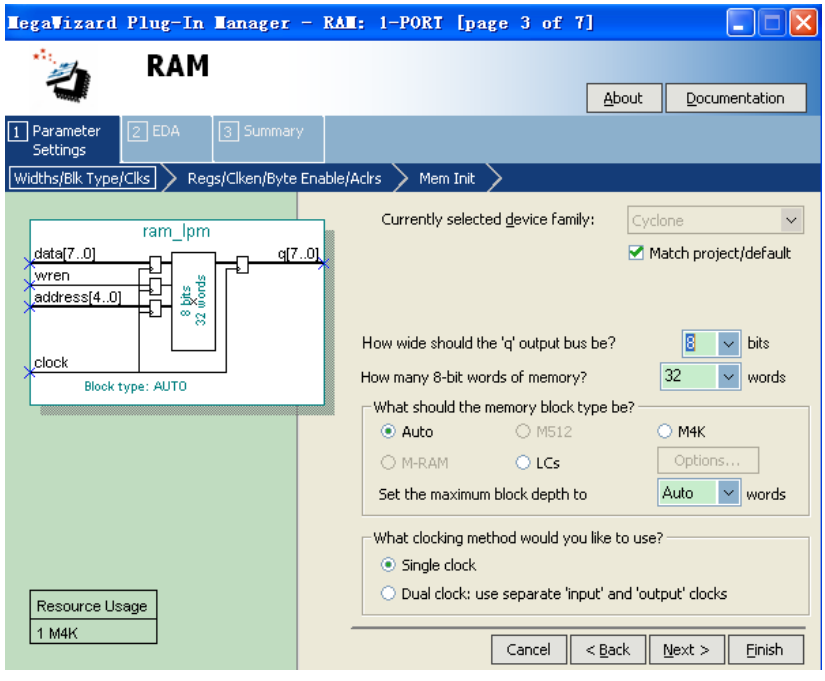

图 8-3　RAM 参数配置窗口

3. 初始化数据

在图 8-3 中，单击“Next”按钮，弹出输出锁存选择窗口，如图 8-4 所示，选择输出信号 q 锁存。

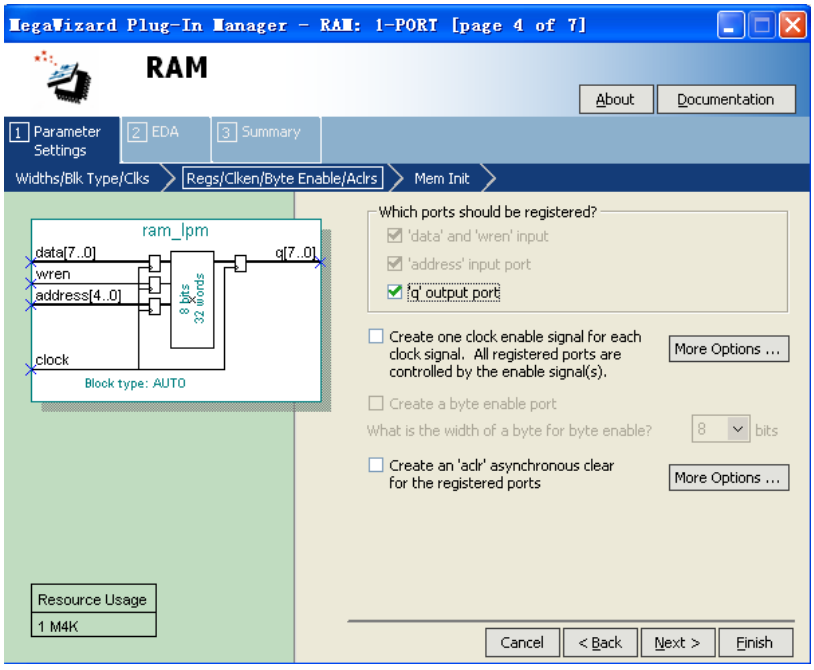

图 8-4　输出锁存选择窗口

在图 8-4 中，单击“Next”按钮，弹出 RAM 初始化文件选择窗口，如图 8-5 所示。如果选择初始化文件，可在对 RAM 存储器读/写控制前，往 RAM 存储器中写入初始数据。

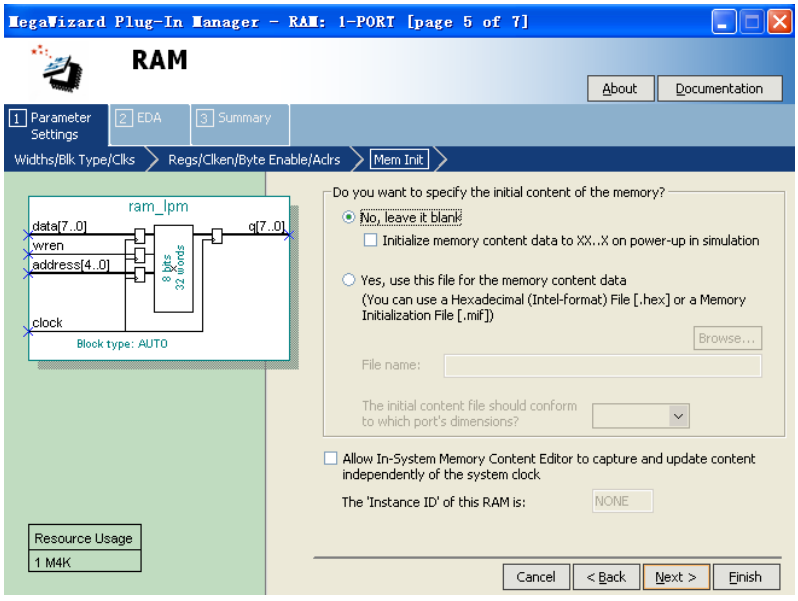

图 8-5　RAM 初始化文件选择窗口

4. 完成 RAM 定制

在图 8-5 中，单击“Next”按钮，弹出仿真库显示窗口，如图 8-6 所示，单击“Next”按钮，弹出 RAM 模块概要窗口，如图 8-7 所示。

在 RAM 模块概要窗口中，显示了配置该模块生成的文件，单击“Finish”按钮，完成对 LPM_RAM 模块的配置。

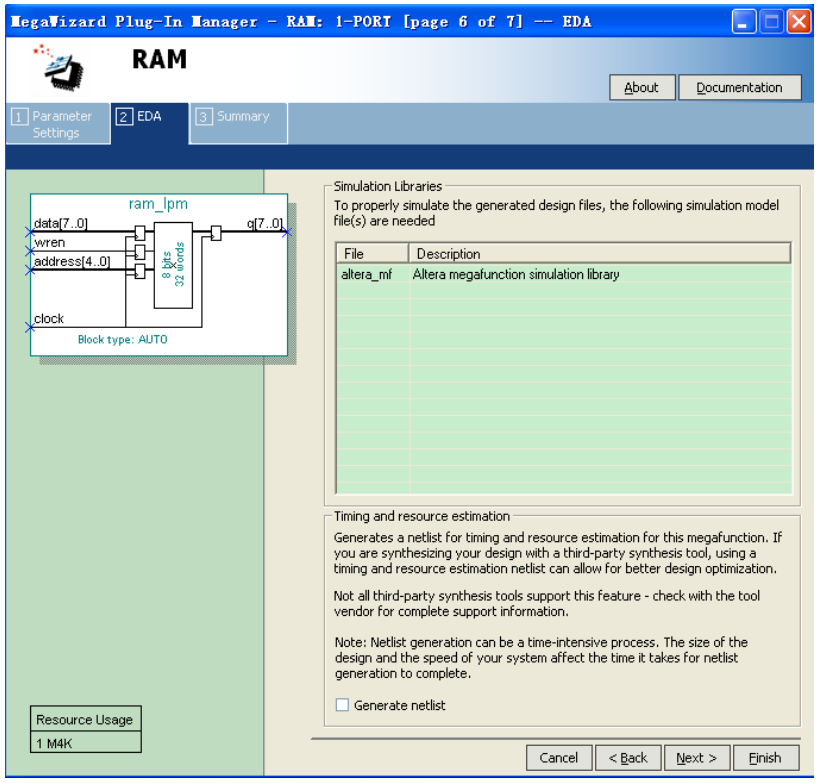

图 8-6　仿真库显示窗口

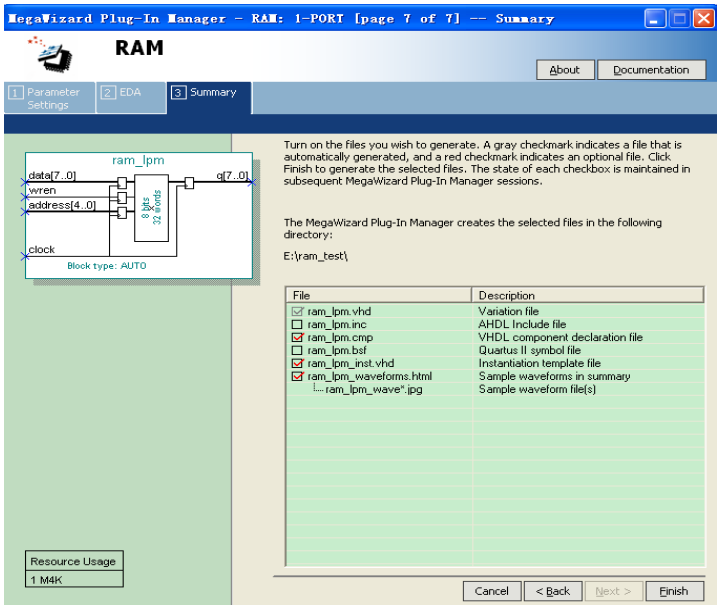

图 8-7　RAM 模块概要窗口

8.1.2　工程编译

接下来对工程进行编译，检验程序语法是否有误。选择“Processing”→“Start Compilation”命令进行工程编译。如果编译正确，信息显示栏会出现成功提示，同时，编译选项左边出现

绿色对钩，如图 8-8 所示。如果编译结果错误，则根据信息提示，对程序进行修改，直至编译通过。

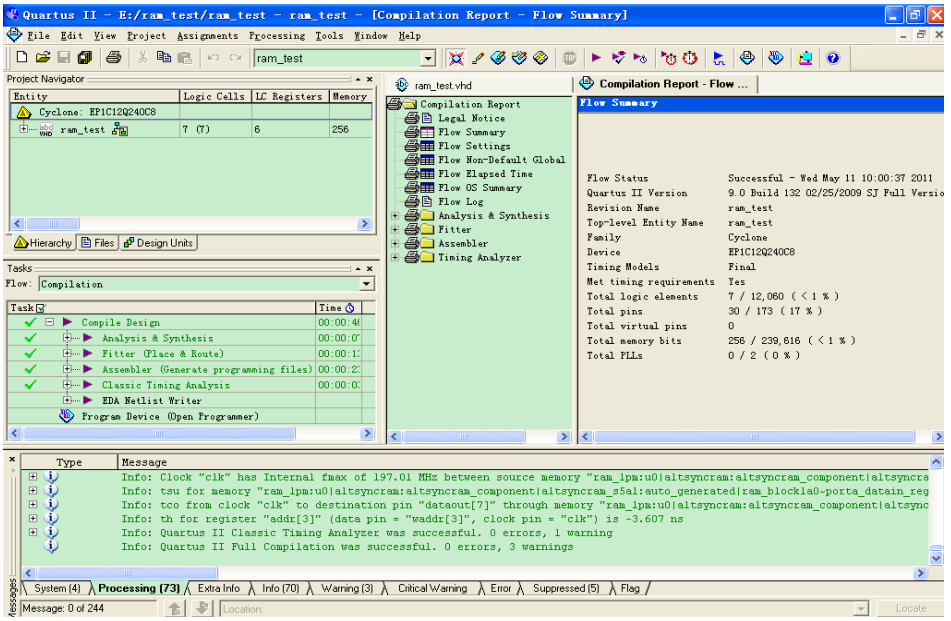

图 8-8　编译结果显示

8.1.3　仿真验证

程序编译通过后，只能表示程序没有语法错误，不能确定程序逻辑是否正确。通过仿真可以验证程序逻辑功能是否满足要求。下面将详细介绍本例的仿真验证过程。

1．建立仿真波形文件

选择“File”→“New”命令，弹出文件类型选择窗口，如图 8-9 所示，选择“Vector Waveform File”项，单击“OK”按钮。弹出输入波形文件设置窗口，如图 8-10 所示，右击“Waveform1.vwf”选项，在弹出的选项中选择“Detach Window”，则该波形设置窗口可全屏显示，便于编辑信号。

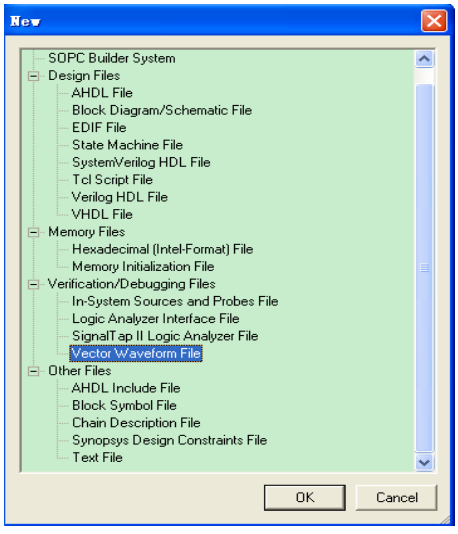

图 8-9　波形文件选择窗口

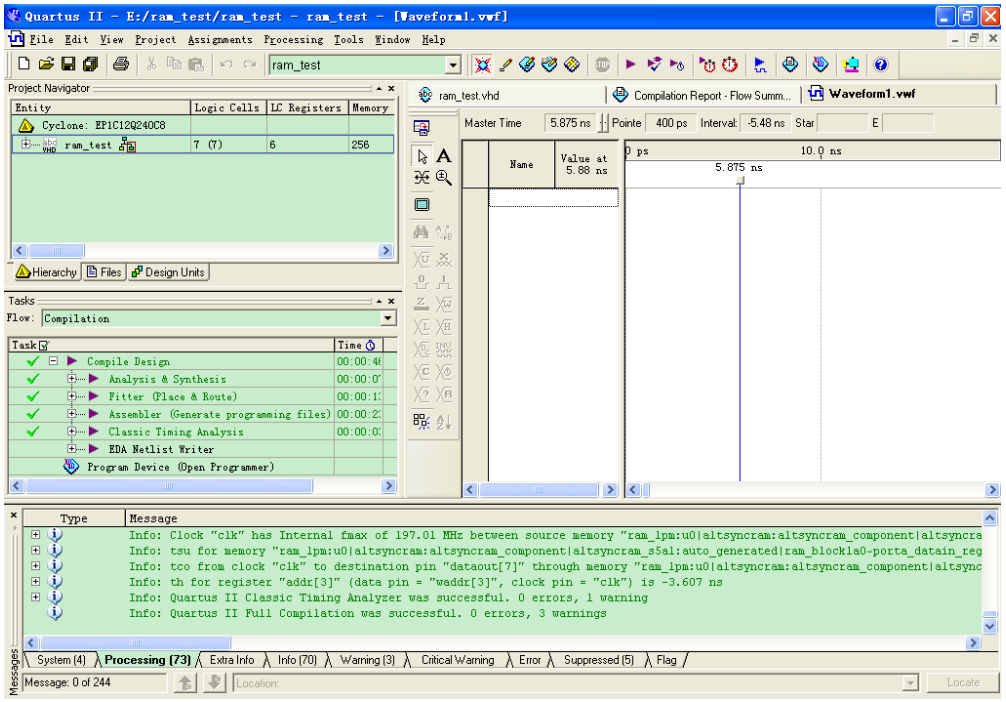

图 8-10　输入波形文件设置窗口

2. 调入节点

选择“View”→“Utility Windows”命令，选择“Node Finder”选项，弹出如图 8-11 所示节点查找窗口，在“Filter”栏中选择“Pins: all”选项，单击右侧的“List”按钮，则在下面窗口中显示出该工程顶层源文件实体中定义的端口节点，将节点信号逐个选中后，拖到图 8-10 所示的输入波形文件设置窗口中，全部端口信号移动结束后，关闭节点查找窗口，则输入波形设置窗口中显示出所有端口节点信号，如图 8-12 所示。

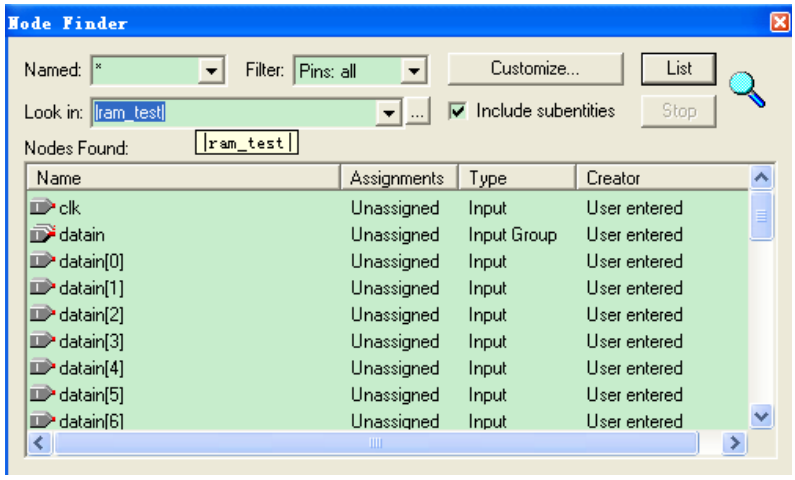

图 8-11　节点查找窗口

3. 设置仿真时间

在输入波形文件设置窗口中，选择“Edit”→“End Time”选项进行仿真时间设置，在弹出窗口中设置仿真时间为 50μs，如图 8-13 所示。

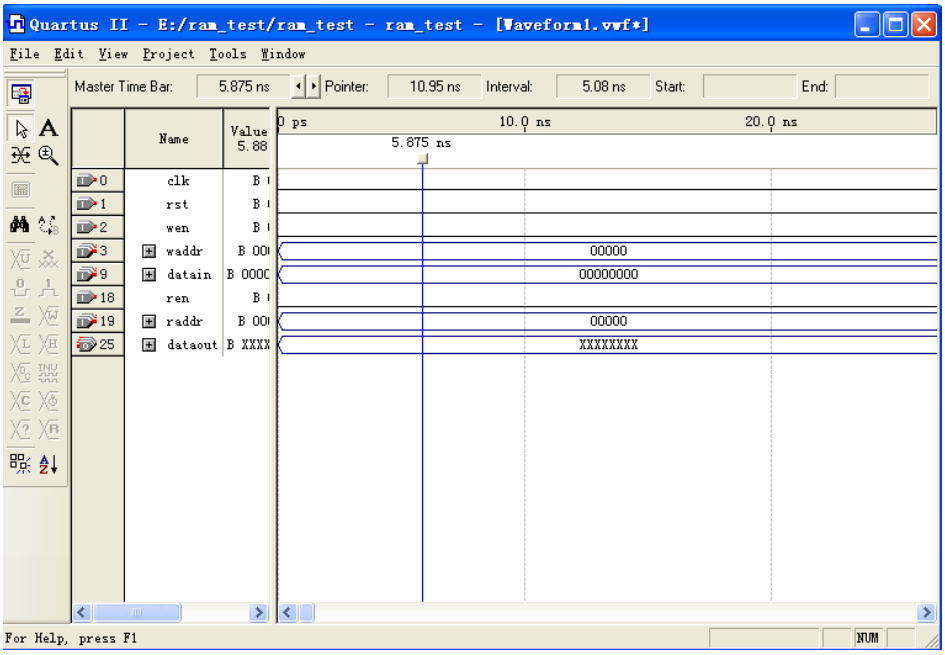

图 8-12　输入波形文件设置窗口

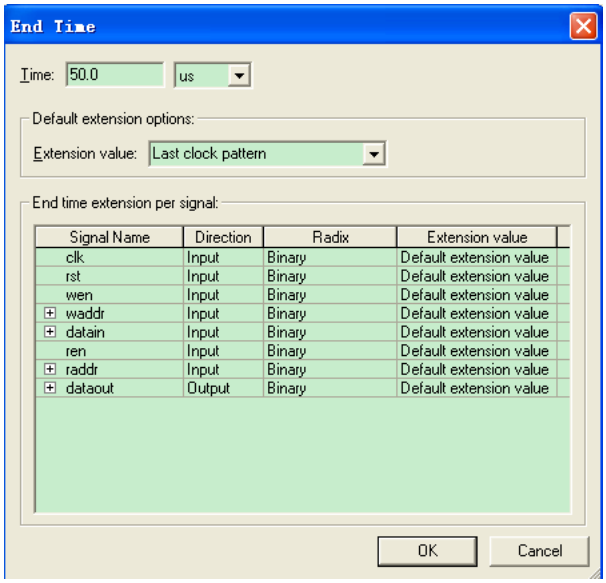

图 8-13　仿真时间设置窗口

4．编辑波形

在波形编辑窗口中逐次选中输入信号，并设置其波形，具体操作如下：

单击输入信号 clk，在左侧工具列表中，选择 Overwrite Clock 时钟设置选项，弹出如图 8-14 所示时钟设置窗口，设置时钟周期为 1μs，时钟偏移为 0，占空比为 50%。

单击输入信号 rst，rst 为复位信号，且低电平有效，在前 4 个时钟周期设置 rst 为低电平，后面时间为高电平。设置方法为：左键选中前 4 个周期内的 rst 信号，在左侧工具列表中选择 Forcing Low（0）选项，则被选中的 rst 信号设为低电平：左键选中第 4 个时钟周期后时间段的 rst 信号，在左侧工具列表中选择 Foring High 选项，把选中的信号设为高电平。

单击写使能信号 wen，在复位信号变为高电平后的一段时间内，将 wen 信号设为高电平。

单击写地址信号 waddr，在 waddr 左边“+”符号左侧空间双击，弹出总线数据格式设置对话框，如图 8-15 所示，在“Radix”下拉列表中选择十六进制 Hexadecimal。选择写使能信号 wen 高电平时对应的 waddr 信号，单击左侧工具列表中的 Count Value 工具，弹出如图 8-16 所示窗口，“Radix”下拉列表中选择十六进制 Hexadecimal，初始值选择 00，“Increment by”栏输入 1，即每时钟周期增加 1。

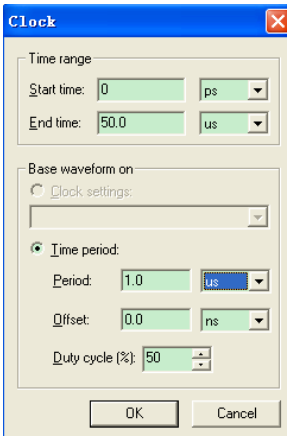

图 8-14　时钟设置窗口

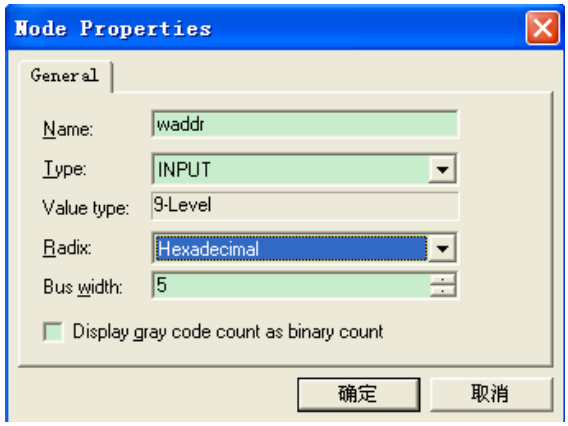

图 8-15　总线数据格式设置对话框

单击“Timing”选项卡，在“Count every”栏中选择 1.0μs，使数据增加与时钟信号同步，如图 8-17 所示。

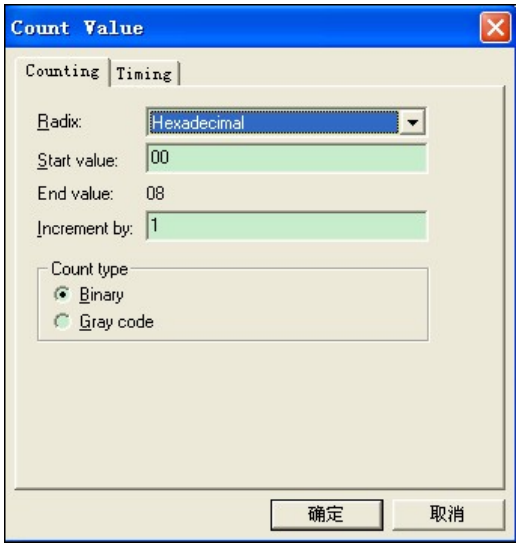

图 8-16　总线数据设置窗口

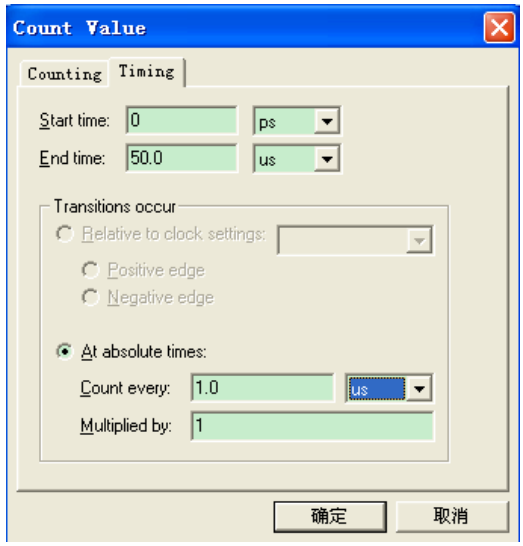

图 8-17　总线数据格式设置窗口

单击写数据信号 datain，单击左侧工具列表中的“Count Value”工具，设置输入数据初始值为 0x20，其他与 waddr 设置相同。

单击读使能信号 ren，当写使能信号无效后，选中 ren 信号，并设为高电平。

单击读地址信号 raddr，选择读使能 ren 信号为高电平时对应的 raddr 信号，单击左侧工具列表中的“Count Value”工具，设置 raddr 初始值为 00，其他选项与 waddr 设置相同。

以上操作完成后，输入波形设置窗口如图 8-18 所示。

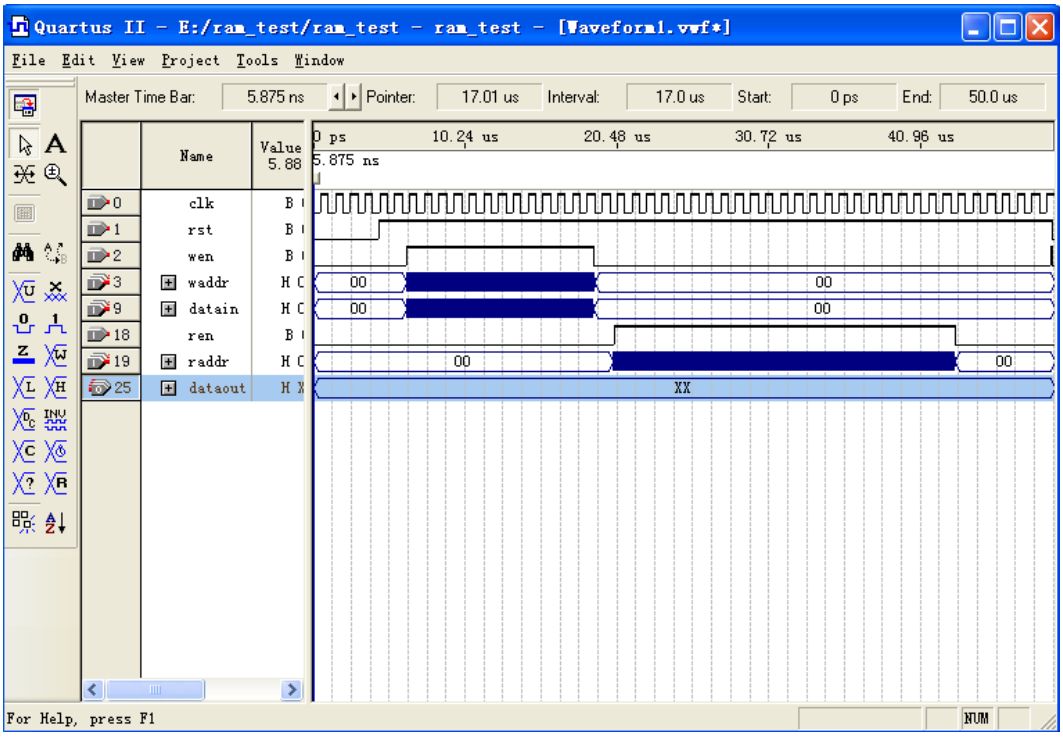

图 8-18　输入波形设置窗口

5．保存波形文件

选择“File”→“Save”命令，弹出如图 8-19 所示文件保存窗口，设置输入波形文件名 ram_test，保存在工程 ram_test 目录下。

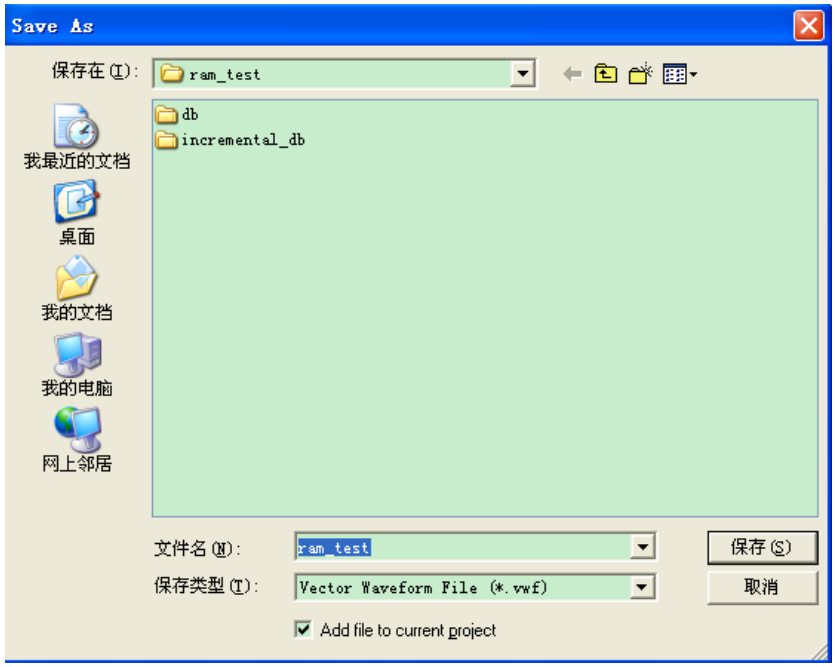

图 8-19　输入波形文件保存窗口

6．时序仿真

（1）时序仿真设置

选择“Assignments”→“Settings”命令，弹出如图 8-20 所示窗口。单击左侧的“Simulator Settings”，在右侧界面“Simulation mode”栏中选择“Timing”，以进行时序仿真，在“Simulation input”栏选择建立的输入波形文件 ram_test.vwf，单击“OK”按钮完成设置。

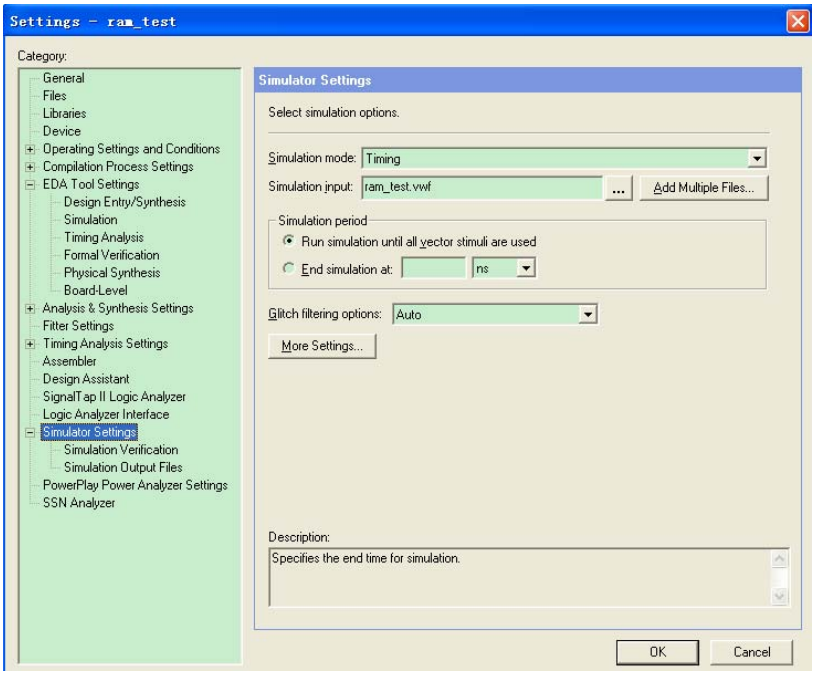

图 8-20　时序仿真设置窗口

（2）启动时序仿真

选择“Processing”→“Start Simulation”命令，则启动时序仿真。时序仿真波形如图 8-21 所示。由仿真波形图可以看出，当写使能信号有效时，在输入时钟同步下，输入数据被写入到写地址对应的存储单元；当写使能无效、读使能有效时，在时钟信号同步下，读地址对应的存储单元的数据从数据输出端口读出，写入数据被读出。由此可判断源程序逻辑正确。

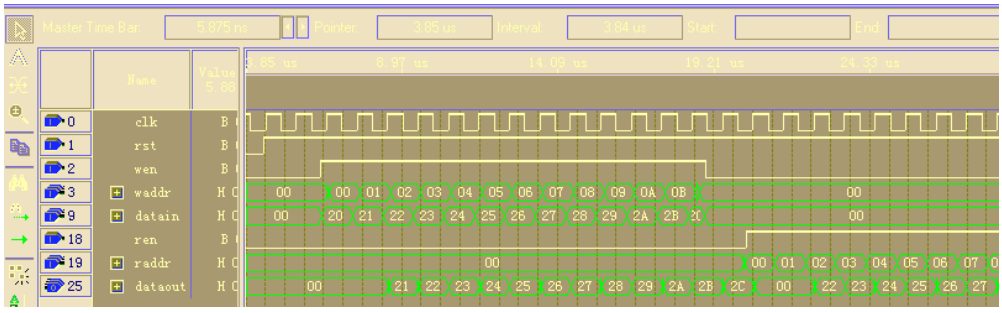

图 8-21　时序仿真波形图

8.1.4　查看 RTL 原理图

选择“Tools”→“Netlist viewers”命令，选择“RTL Viewer”，则显示该功能对应的 RTL（寄存器传输级）原理图，如图 8-22 所示。

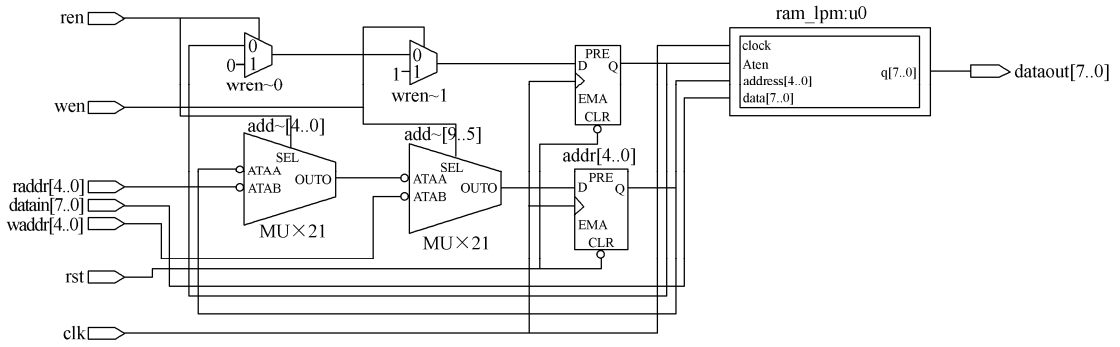

图 8-22　RTL 原理图

8.1.5　LPM_RAM 应用

【例 8-1】

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ram_test is
port (rst: in std_logic;                                    --复位信号，低电平有效
        clk: in std_logic;                                  --时钟信号
      wen: in std_logic;                                    --写使能信号，高电平有效
      ren: in std_logic;                                    --读使能信号，高电平有效
      waddr: in std_logic_vector(4 downto 0);               --写端口地址信息
      raddr: in std_logic_vector(4 downto 0);               --读端口地址信息
      datain: in  std_logic_vector(7 downto 0);             --数据写入端口
      dataout: out std_logic_vector(7 downto 0)             --数据输出端口
      );
end entity;

architecture behav of ram_test is
component ram_lpm is                                        --调用单端口 RAM 存储器
port(clock: in std_logic;
      wren: in std_logic;
      data: in std_logic_vector(7 downto 0);
      address: in std_logic_vector(4 downto 0);
      q: out std_logic_vector(7 downto 0)
      );
end component;
signal wren: std_logic;
signal addr: std_logic_vector(4 downto 0);
begin
process(rst,clk,wen,ren)
begin
     if  rst='0' then
        wren<='0';
```

```
            addr<="00000";
        elsif clk'event and clk='1' then
           if wen='1' then
             wren<='1';
             addr<=waddr;
           elsif ren='1' then
                wren<='0';
                addr<=raddr;
           end if;
        end if;
    end process;
    u0: ram_lpm port map(clock=>clk,wren=>wren,data=>datain,
                                address=>addr,q=>dataout);
    end behav;
```

上述 VHDL 程序设计了一个双端口 RAM 存储器，其中一个端口为数据写入端口，输入信号有写使能（wen）、数据写入地址（waddr）、数据输入端口（datain）；另一个端口为数据读出端口，输入信号有读使能（ren）、数据读地址（raddr），输出信号为数据输出端口（dataout）。程序内部引用了单端口 RAM 存储器 ram_lpm 模块。

8.2 LPM_ROM 宏模块

8.2.1 建立初始化数据文件

ROM 为只读存储器，要提前往内部存储单元写入数据，正常工作时，只能执行读操作，从 ROM 中读取数据。下面将介绍如何给 rom_lpm 模块配置初始化数据。

1．打开数据文件编辑窗口

选择“File”→“New”命令，在弹出的文件类型选择窗口中，选择“Memory Initialization File”选项，如图 8-23 所示。

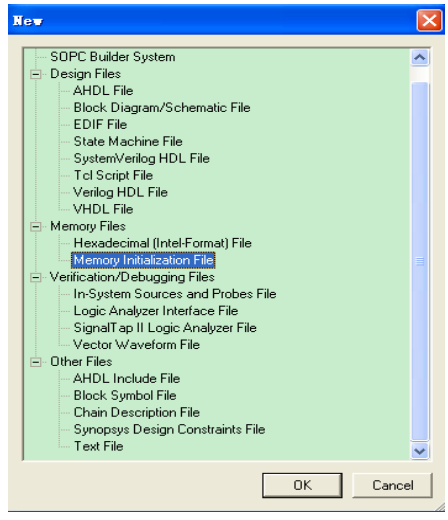

图 8-23　输入文件类型选择窗口

2．设定数据文件容量

在图 8-23 中，单击“OK”按钮，弹出存储器初始化数据设置窗口，如图 8-24 所示，设置数据个数为 32，数据宽度为 8。

3．输入数据

在图 8-24 中，单击“OK”按钮，打开 mif 数据表格，如图 8-25 所示。

在图 8-25 所示初始化数据输入界面，在 mif 数据表格中输入初始化数据，地址 0 存储单元写入数据 16，地址 1 存储单元写入数据 17，……，地址 31 存储单元写入数据 47。

4．保存 mif 文件

输入完成后，选择“File”→“Save”命令进行保存，保存文件类型可为 mif 格式或 Hex 格式。本例选择 mif 存储格式，将初始化数据文件

保存为 rom_data.mif。

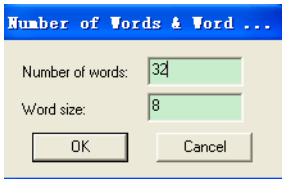

图 8-24　存储器初始化数据设置窗口

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	16	17	18	19	20	21	22	23
8	24	25	26	27	28	29	30	31
16	32	33	34	35	36	37	38	39
24	40	41	42	43	44	45	46	47

图 8-25　初始化数据输入界面

在该工程目录下打开 rom_data.mif 文件，里面内容如下：

```
WIDTH=8;
DEPTH=32;
ADDRESS_RADIX=UNS;
DATA_RADIX=UNS;
CONTENT BEGIN
    0   :   16;
    1   :   17;
    2   :   18;
    3   :   19;
    4   :   20;
    5   :   21;
    6   :   22;
    7   :   23;
    8   :   24;
    9   :   25;
    10  :   26;
    11  :   27;
    12  :   28;
    13  :   29;
    14  :   30;
    15  :   31;
    16  :   32;
    17  :   33;
    18  :   34;
    19  :   35;
    20  :   36;
    21  :   37;
    22  :   38;
    23  :   39;
    24  :   40;
    25  :   41;
    26  :   42;
    27  :   43;
    28  :   44;
    29  :   45;
    30  :   46;
    31  :   47;
END;
```

在上述文件中可进行初始化数据的修改，修改后保存即可。存储器初始化文件的生成也可以使用文本方式生成，在文本文件中输入上述格式文件内容，保存为 mif 类型文件。

8.2.2 LPM_ROM 宏模块配置

1．定制 LPM_ROM

选择“Tools”→“MegaWizard Plug-In Manager”命令，弹出如图 8-26 所示对话框，选择“Create a new custom megafunction variation”选项，单击“Next”按钮，弹出宏模块功能设置窗口，如图 8-27 所示。

图 8-26　宏模块选项

在宏模块功能设置窗口中，单击左侧的“Memory Compiler”目录，在展开的选项中选择“ROM:1-PORT”，此模块为单端口 ROM 存储器，在目标芯片系列栏选择 Cyclone 系列，输出文件类型选择 VHDL，输入存放目录及文件名 E:\rom_test\rom_lpm，调用的 ROM 名称为 rom_lpm。

2．参数配置

在图 8-27 中，单击“Next”按钮，弹出如图 8-28 所示的 ROM 参数配置窗口。设置输出数据总线宽度为 8，存储容量为 32，其他为默认配置。窗口左侧显示了调用 ROM 模块的端口信号：5 位地址输入信号 address；时钟信号 clock；8 位数据输出信号 q。

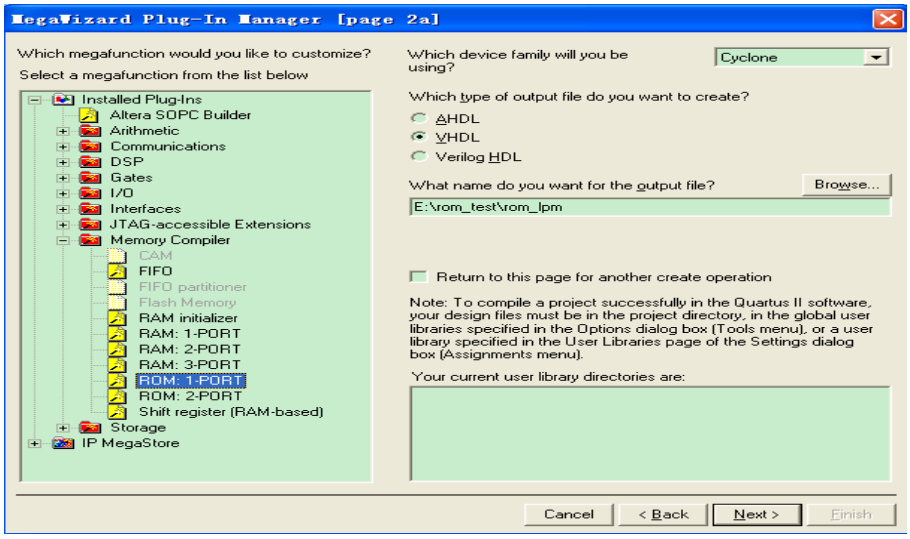

图 8-27　宏模块功能设置窗口

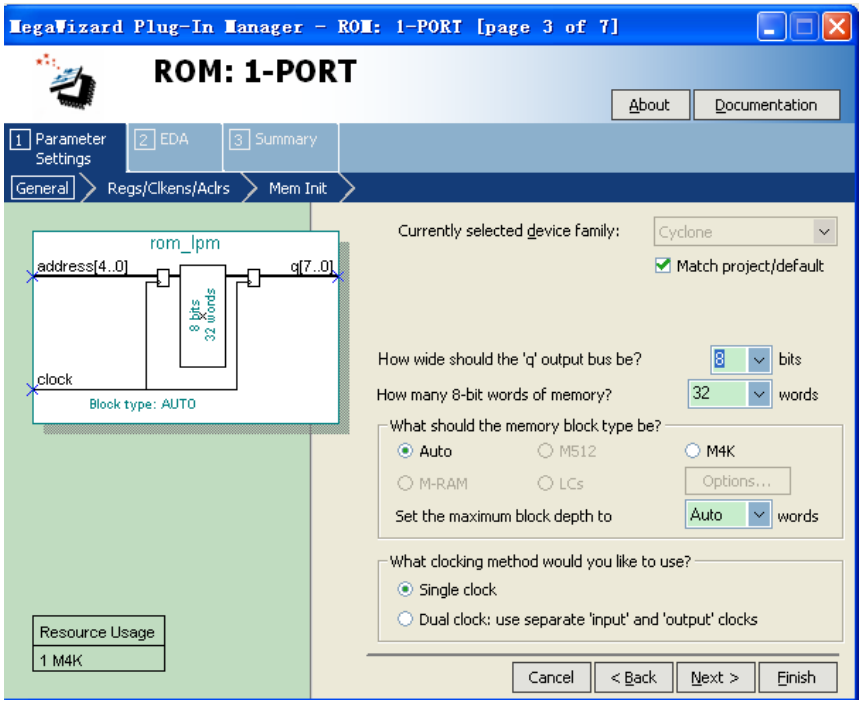

图 8-28　ROM 参数配置窗口

单击“Next”按钮，弹出输出锁存选择窗口，如图 8-29 所示，选择输出信号 q 锁存。

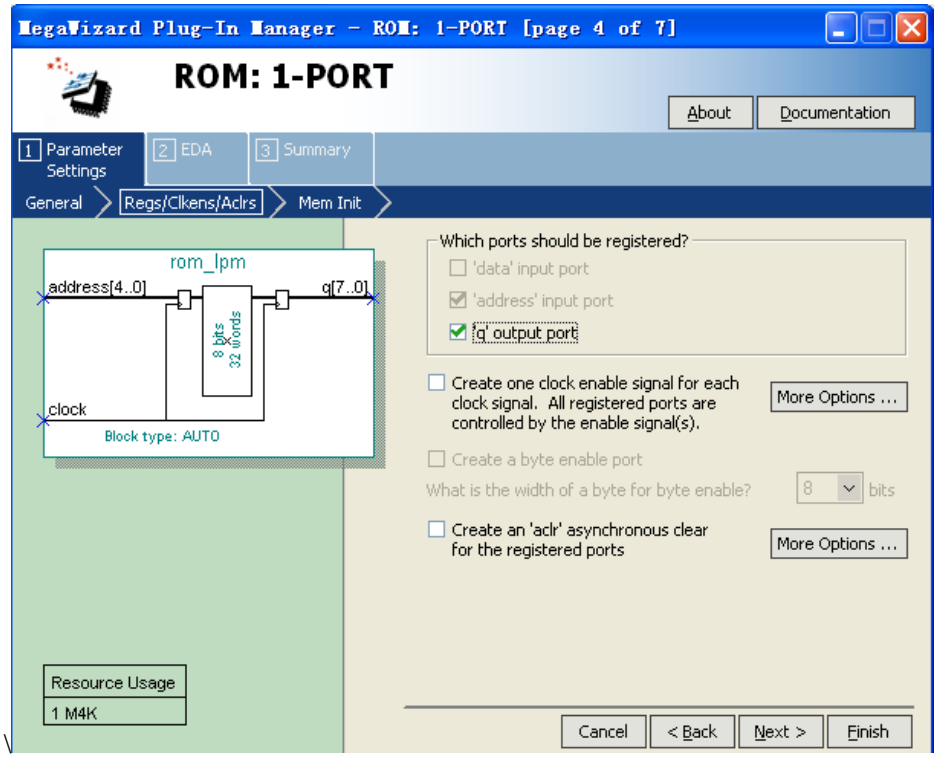

图 8-29　输出锁存选择窗口

3．设置初始化文件

单击“Next”按钮，弹出 ROM 初始化文件选择窗口，如图 8-30 所示。单击“Browse”按钮，选择前面建立的初始化数据文件，选择目录为：E:\rom_test\rom_test.mif。

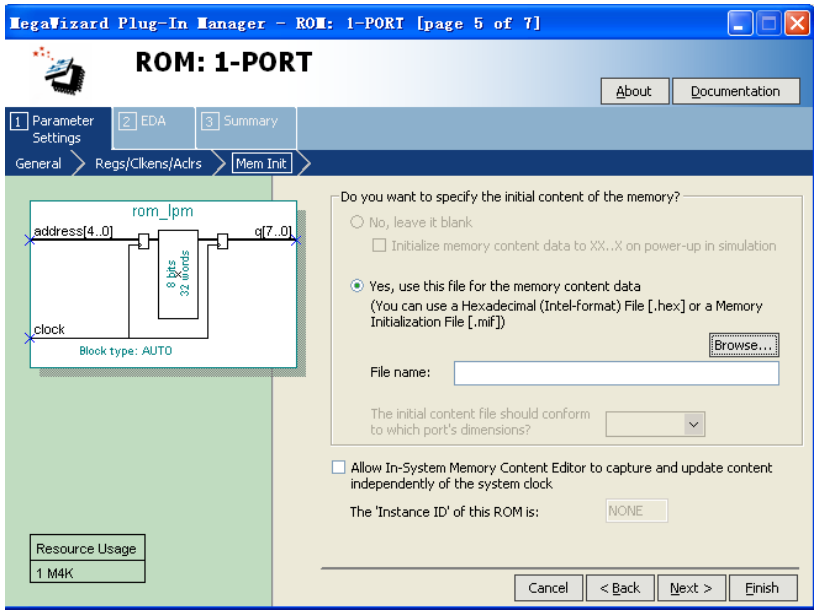

图 8-30　初始化文件选择窗口

4．完成 LPM_ROM 配置

在图 8-30 中，单击“Next”按钮，弹出仿真库显示窗口，如图 8-31 所示；单击“Next”按钮，弹出 ROM 模块的概要窗口，图 8-32 中单击“Finish”按钮，完成对 LPM_ROM 模块的配置。

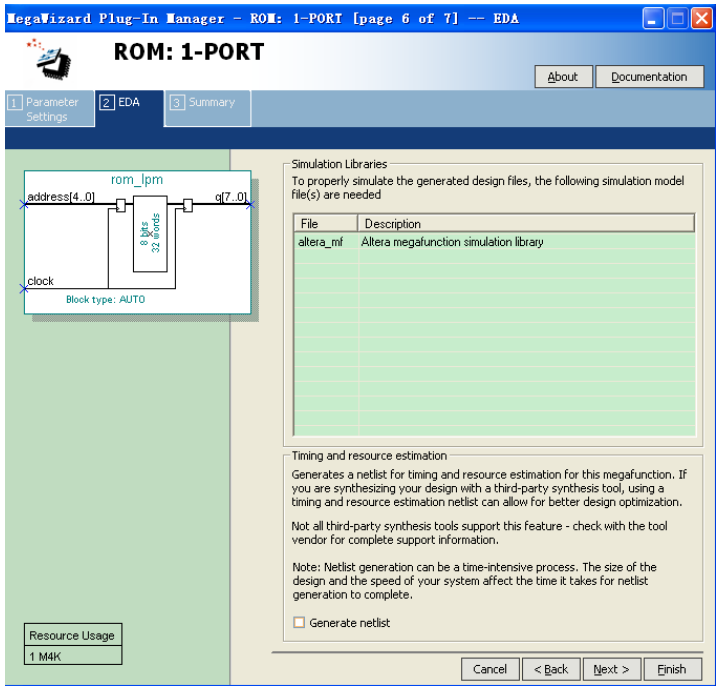

图 8-31　仿真库显示窗口

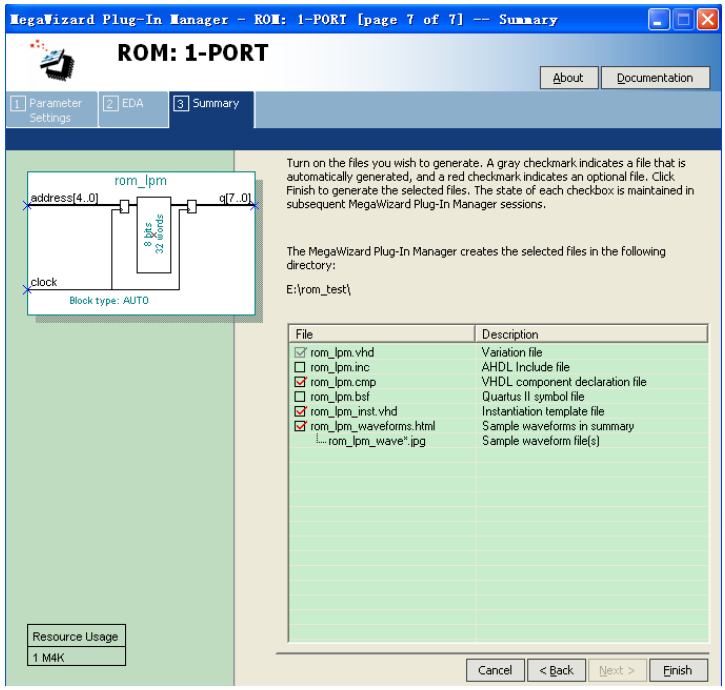

图 8-32　ROM 模块概要窗口

选择“Processing”→“Start Compilation”命令进行工程编译，编译正确后，可通过时序仿真进行波形验证。

8.2.3　仿真验证

1．建立仿真波形文件

参考 LPM_RAM 模块应用中输入仿真波形的建立过程，在此不再讲述具体流程，仿真波形文件中，输入端口信号波形设置如图 8-33 所示。

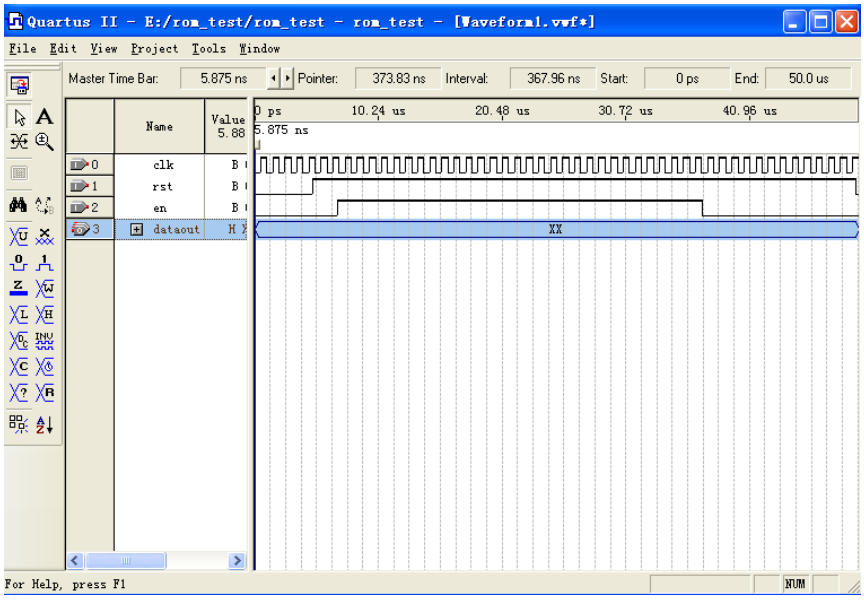

图 8-33　ROM 输入仿真波形设置

2．保存输入波形文件

保存文件名为 rom_test.vwf。

3．时序仿真

选择“Assignments”→“Settings”命令，弹出如图 8-34 所示窗口，单击左侧的“Simulator Setting”，在右侧界面“Simulation mode”栏选择“Timing”，以进行时序仿真，在“Simulation input”栏选择建立的输入波形文件 rom_test.vwf，单击“OK”按钮完成设置。

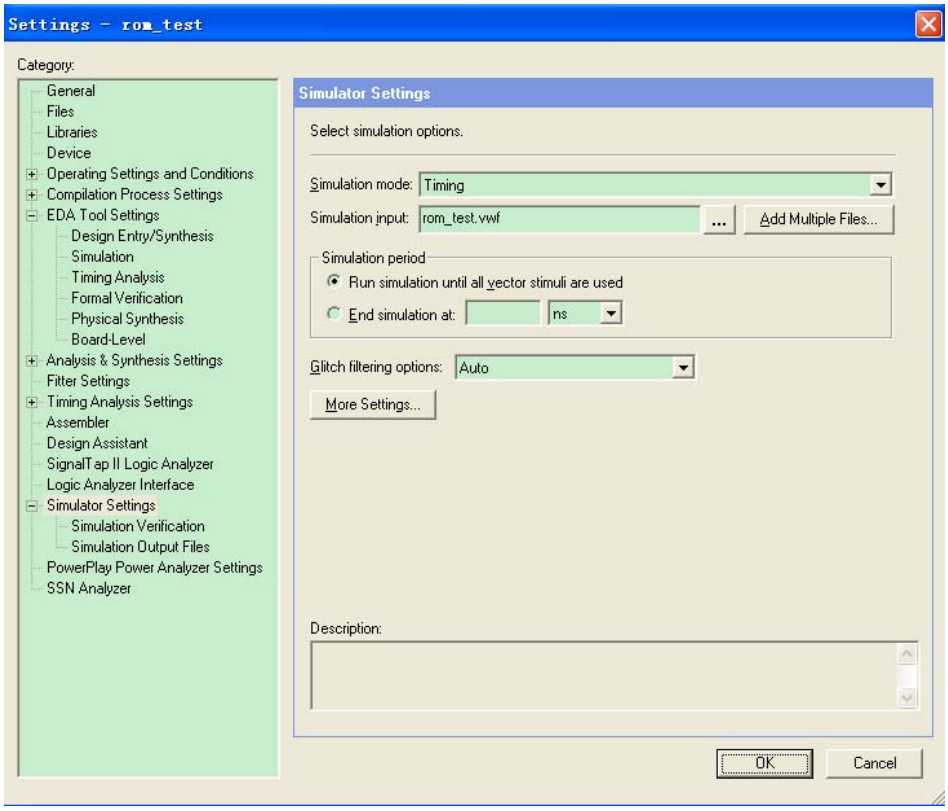

图 8-34　时序仿真设置窗口

选择“Processing”→“Start Simulation”命令，运行时序波形仿真，仿真结果如图 8-35 所示。根据仿真波形图可知，当使能信号 en 有效时，在时钟信号 clk 同步下，ROM 存储单元中的存储数据被读出，0 地址对应数据 10（十六进制数，对应十进制数 16），1 地址对应数据 11，读出数据内容与 ROM 初始化文件中写入的数据对应，可判读该工程逻辑正确。

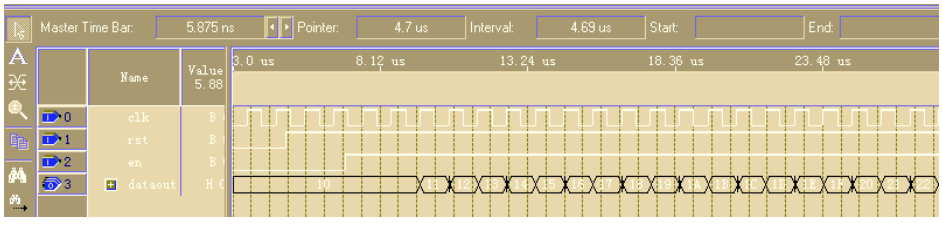

图 8-35　时序仿真结果图

4．RTL 结果

该工程对应的 RTL 原理图如图 8-36 所示。