程序存储器与数据存储器

程序存储器是用来存放用户程序的，通常采用只读存储器 ROM 来存储程序。数据存储器存放运算数据及中间结果，一般采用随机存储器 RAM 来实现其功能。这里主要介绍利用 LPM 模块构建程序存储器和数据存储器的方法。

CPU 中的一些重要部件，如 RAM，ROM 等，可直接调用 LPM 模块构成。因此在 FPGA 中利用嵌入式阵列块可以构成各种结构的存储器。下面详细介绍利用 MegaWizard Plug-In Manager 进行 LPM 模块调用和测试的一般方法。

1．定制初始化数据文件

首先确定 ROM 内的数据文件。QuartusII 能接受的 LPM 模块 LPM_ROM 中的初始化数据文件的格式有 2 种：Memory Initialization File（.mif）格式和 Hexadecimal（Intel-Format）File（.hex）格式。

实际应用中只要使用其中一种格式的文件即可。

（1）建立.mif 格式文件

首先在 Quartus II 中选择 ROM 数据文件编辑窗，即在 File 菜单中选择 New，并在 New 窗中选择 Other files 页，再选择 Memory Initialization File 项，单击 OK 按钮后产生 ROM 数据文件大小选择窗。根据实际情况，可选 ROM 的存储单元数 Number 为 64，数据宽 Word size 取 8 位。单击 OK 按钮，将出现如图 1 所示的空的 mif 数据表格。表格中的数据格式可通过鼠标右键点击窗口边缘的地址数据弹出的窗口选择。此表中任一数据（如第三行的 99）对应的地址为左列与顶行数之和（如 16+2=18。十六进制为 12H，即 00010010）。然后将数据填入此表中。完成后，在 File 菜单中单击 Save as 按钮，保存此数据文件，在这里不妨取名为 romd.mif。

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	254	252	249	245	239	233	225
8	217	207	197	186	174	162	150	137
16	124	112	99	87	75	64	53	43
24	34	26	19	13	8	4	1	0
32	0	1	4	8	13	19	26	34
40	43	53	64	75	87	99	112	124
48	137	150	162	174	186	197	207	217
56	225	233	239	245	249	252	254	255

图 1 将波形数据填入 mif 文件表中

（2）建立. hex 格式文件

建立.hex 格式文件与以上介绍的方法相同，只是在 New 窗中选择 Other files 项后，选择 Hexadecimal（Intel-Format）File 项，最后存盘.hex 文件。

2．定制 ROM 元件

首先完成存放数据 ROM 的设计。利用 MegaWizard Plug-In Manager 定制程序存储器 ROM 宏功能块，并将以上的数据文件加载于此 ROM 中。设计步骤如下：

首先建立工程文件，然后建立新文件，在加载器件前，首先定制 ROM 符号。

（1）打开 MegaWizard Plug-In Manager 初始对话框

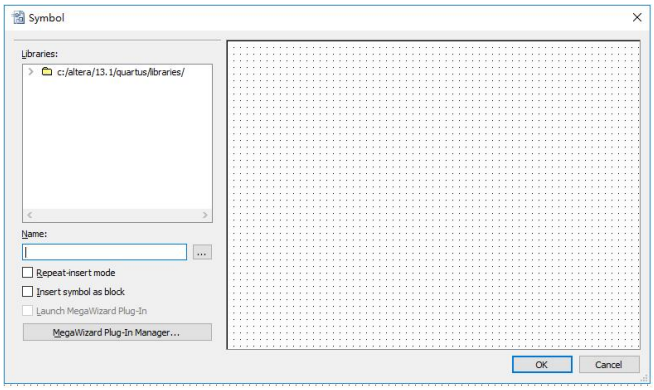

图 2 加载器件的界面

点击图 2 左栏下方的 MegaWizard Plug-In Manager，产生如图 3 所示的界面。

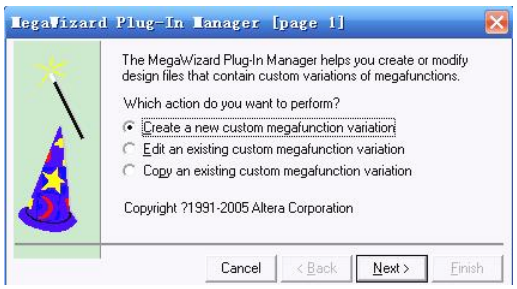

图 3 定制新的宏功能块

选择 Create a new custom…项（如果要修改一个已编辑好的 LPM 模块，则选择 Edit an existing custom…项），即定制一个新的模块。单击 Next 按钮后，产生如图 4 所示的对话框。

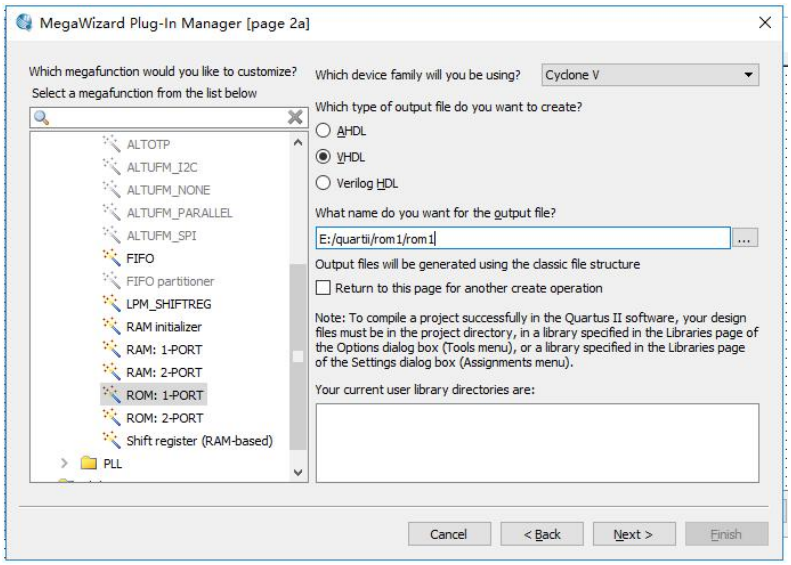

图 4　LPM 宏功能块设定

在左栏选择 Storage 项下的 ROM：1-PORT，再选择 Cyclone 器件和 VHDL 语言方式；最后输入 ROM 文件存放的路径和文件名：

E:/quartii/rom1/rom1（定制的 ROM 元件文件名），单击 Next 按钮。

（2）选择 ROM 控制线、地址线和数据线

在如图 5 所示的对话框中选择数据线位宽和 ROM 中存储单元数分别为 24 和 64；在“What should the memory...”栏选择默认的“Auto”。在适配中，Quartus II 将根据选中的目标器件系列，自动确定嵌入 RAM 模块的类型（Cyclone 系列为 M4K 等）。在“What clocking method would you..”栏选择 single clock。单击 Next 按钮后，产生如图 6 所示的对话框。

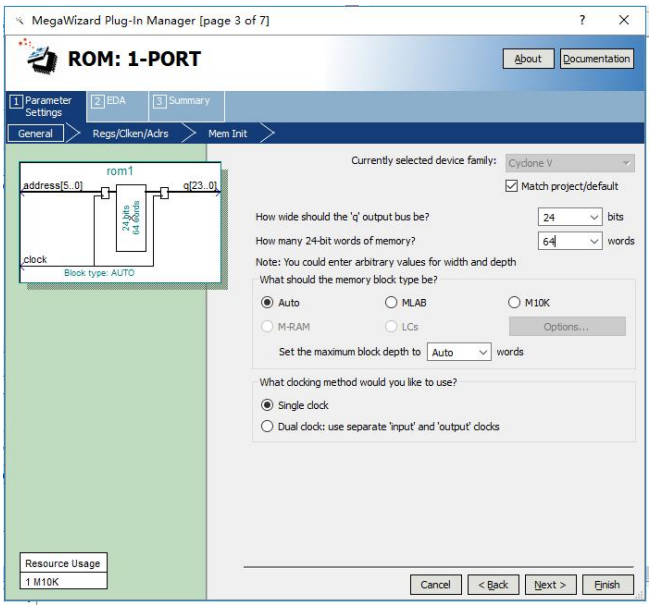

图 5 选择 data_rom 模块数据线和地址线宽度

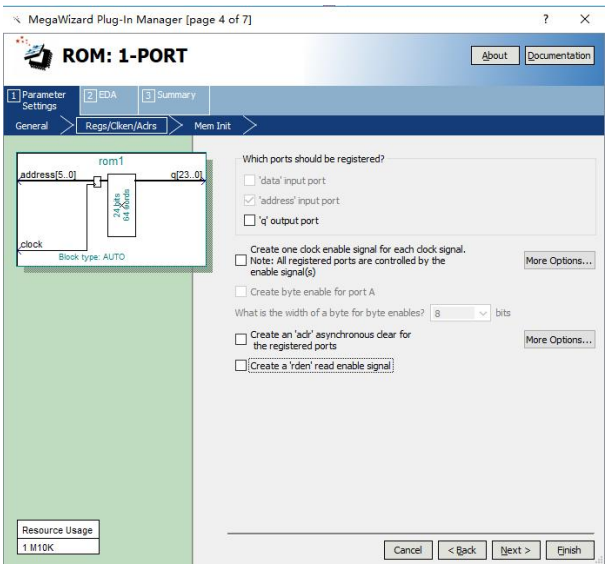

图 6 选择地址锁存信号

（3）单击 Next 按钮后出现图 7 的界面

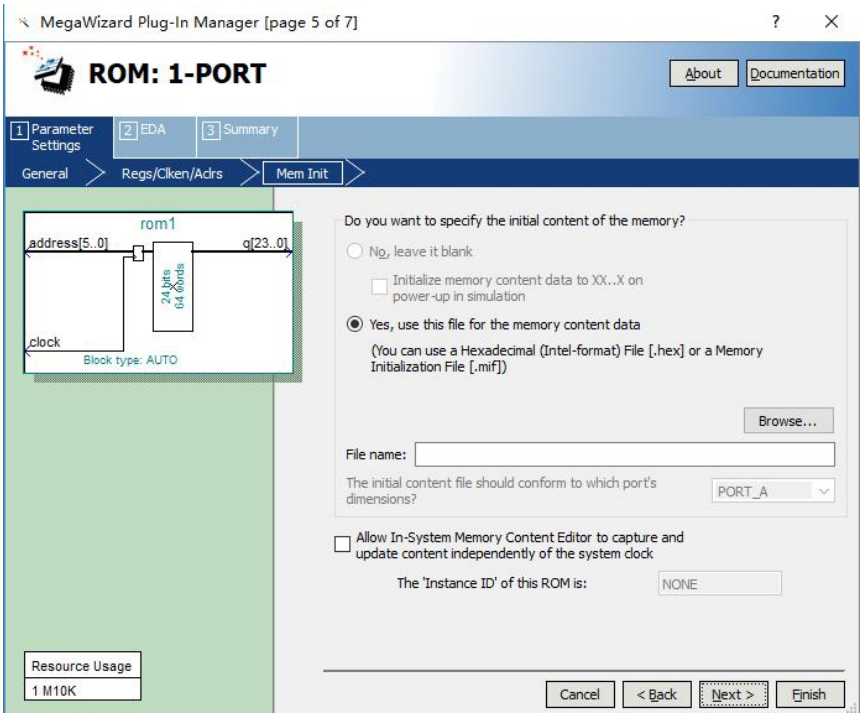

图 7 调入 ROM 初始化数据文件并选择在系统读写功能

在图7的“Do you want to...”栏选择打勾“yes，use this file for the memory content data”项，并按“Browse”钮，选择指定路径上的文件 sdata.hex。

在“Alow In-System Memory...”栏选择打勾，并在“The Instance ID of this rom”栏键入“rom1”，作为此 ROM 的 ID 名称。通过这个设置，可以允许 Quartus II 能通过 JTAG 口对下载于 FPGA

中的此 ROM 进行“在系统”测试和读写（如果需要读写多个嵌入的 ROM 或 RAM，ID 号选择“rom1”就作为此 ROM 的识别名称），这种在系统读写不影响 FPGA 中电子系统的正常工作。

最后点击图 7 的 Next 按钮，再点击 Finish 钮后完成 ROM 定制。

注意，对于 CycloneII 系列，凡是涉及 RAM，ROM 的 LPM 模块使用，都必须作如下设置（CycloneIV 器件无此必要）：

在菜单 Assignments 中选择 Setting 项，在弹出的对话框中选中 Analysis & Synthesis Settings 下的 Default Parameters 项。在此，在 Name 栏键入 CYCLONEII_SAFE_WRTIE；在 Default Setting 栏键入“VERIFIED_SAFE”,并分别按 Add 和 OK 按纽后关闭 Settings 窗。

3．微程序存储器 LPM_ROM 的设置

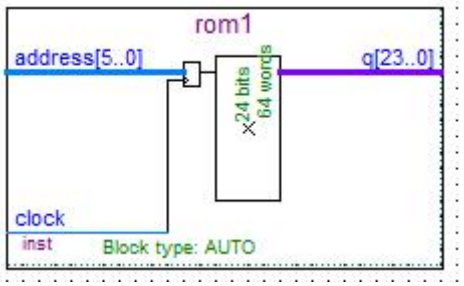

图8 ROM1的结构图

图 8 是一个用于存储微程序的 ROM，有 3 组信号，即 clock 是输入时钟脉冲；q[23..0]是此 ROM 的 24 位数据输出端；a[5..0]是 6 位读出地址。此图中的 ROM 是原理图方式表达的模块，参数设置方式同上。在设置中，应该设置其能够在系统 ROM/RAM 读写允许，以便能对 FPGA 中的 ROM 在系统读写。

用初始化存储器编辑窗口编辑 ROM 配置文件（文件名.mif）。这里预先给出后面将要用到的微程序文件：rom_a.mif 。rom_a.mif 中的数据是微指令码（图 2-16）。

ROM 中数据的写入可以有两种方法：

（1）通过初始化文件来编写

用初始化存储器编辑窗口编辑 ROM 配置文件（文件名.mif），与图形文件一起进行编译，生成下载文件（.SOF）。在对 FPGA 进行配置后，数据就写入了 ROM 中。

（2）利用 QuartusII 的在系统存储模块读写工具

可以了解 FPGA 中 ROM 中的数据，并对其进行在系统读写操作。

对于 Cyclone，CycloneII 等系列的 FPGA，只要对已调用的 ROM 或 RAM 模块适当设

置，就能利用 QuartusII 的 EAB（嵌入式阵列块）在系统（In-System）读写编辑器（In-System Memory Content Editor），直接通过 JTAG 口读取，并改写 FPGA 内处于工作状态的存储器中的数据，读取过程不影响 FPGA 的正常工作。

此编辑器的功能有许多用处，如在系统了解 ROM 中加载的数据、读取基于 EAB 的 RAM 中采样获得的数据，以及对嵌入在由 FPGA 资源构成的 CPU 中的数据 RAM 和程序 ROM 中的信息等。

（1）打开在系统存储单元编辑窗

在菜单 Tool 中选择 In-System Memory Content Editor 项，弹出的编辑窗如图 9 所示。点击右上的 Setup 钮，在弹出的 Hardware Setup 对话框中选择 Hardware settings 页，双击此页中的选项 USB-laster 之后，单击 Close 按钮，关闭对话框。这时将出现如图 10 所示的窗口信息（这里假设已将实验系统与 PC 机的并口相连，并打开了电源）。

图 9 In-System Memory Content Editor 编辑窗

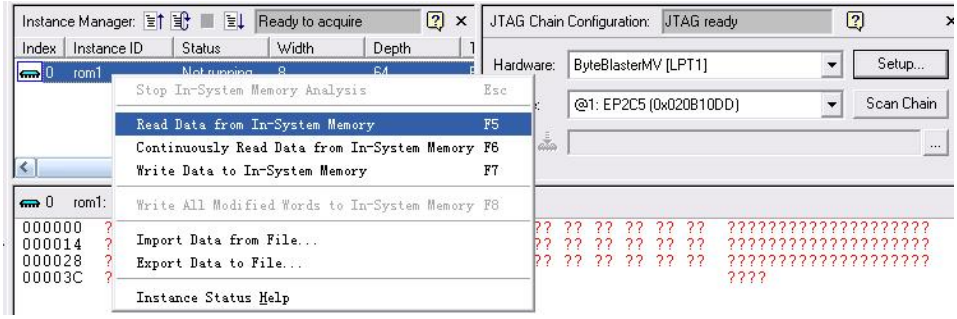

图 10 与实验系统上的 FPGA 通信正常情况下的编辑窗界面

（2）读取 ROM 中的波形数据

右键点击左上窗的数据文件名“ROM1”，将弹出如图 10 的下拉菜单，选择菜单中的 Read from In-System Memory 项，即出现图 11 所示的数据。这些数据是在系统正常工作的情况下通过 FPGA 的 JTAG 口从其内部 EAB ROM 中读上来的波形数据，它们应该与加载进去的文件 SDATA.hex 中的数据完全相同。

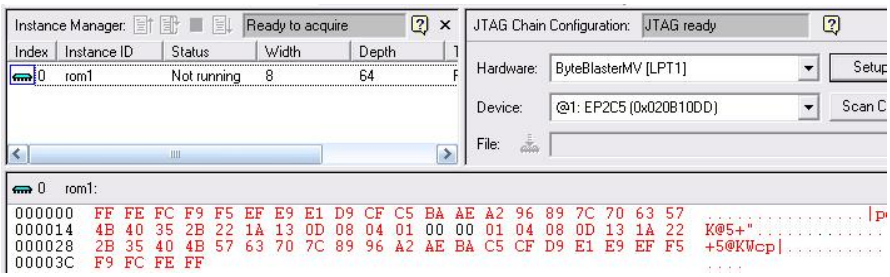

图 11 从 FPGA 中的 ROM 读取波形数据

（3）写数据

方法同读数据。首先如图 12 所示，编辑波形数据，如将最前面的 4 个 8 位数据都改为 11H，再右键点击左上窗的数据文件名“rom1”，选择弹出如图 10 所示的下拉菜单中的 Write Data to In-System Memory 项（也可点击上方含有下指箭头的按纽），即可将编辑后所有的数据通过 JTAG 口下载于 FPGA 中的 LPM_ROM 中，这时可以从示波器和 SignalTapII 上同时观察到波形的变化。

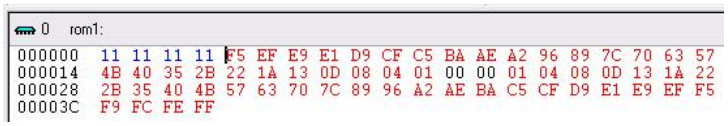

图 4-44　编辑波形数据

（4）输入输出数据文件

用以上相同的方法通过选择图 10 所示的下拉菜单中的“Export Data to File”或“Import Data from File”项即可将在系统读出的数据以 MIF 或 HEX 的格式文件存入计算机中，或将此类格式的文件“在系统”地下载到 FPGA 中去。

4. RAM 的调用和结构

RAM 定制与 ROM 基本相同，同样使用工具 MegaWizard Plug-In Manager。

在进入图 3 所示窗口后，选择 RAM:1-PORT 项，选择 Cyclone 器件，文件名可取为 sram.vhd。由图 9 显示的窗口可见，此 RAM 的数据宽度选择为 8；地址宽度为 6；有一个地址锁存时钟和一个写使能控制线。图 10 显示，RAM 中也能加载初始化数据文件（本例未加载），也能在系统读写，ID 名为 ram2。

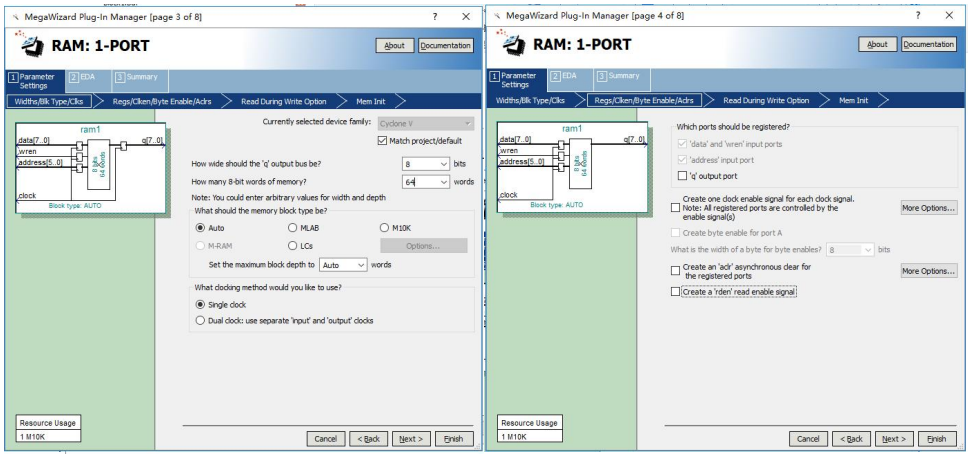

图 9 编辑定制 RAM

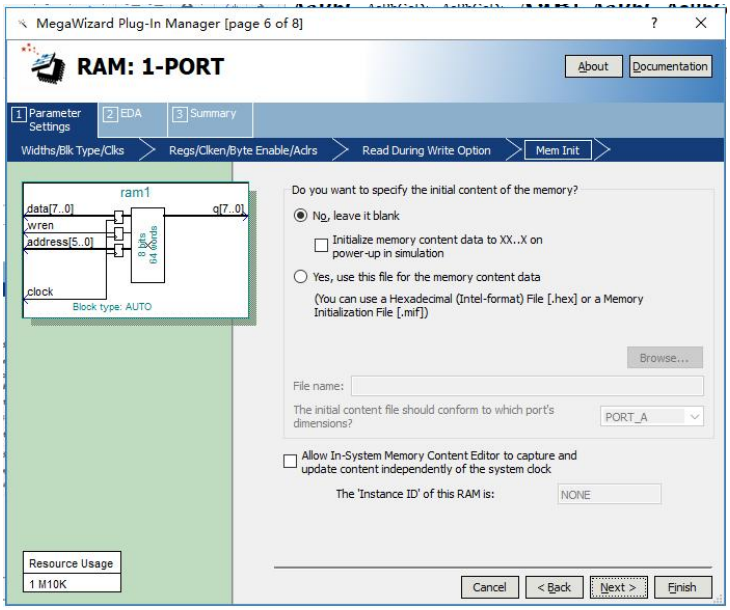

图 9（c）

图 9（c）中，RAM 可以加载初始化文件，也可是空。选 yes，加载初始化文件；选 no，内容为空。

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018108	00ED82	00C050	00E004	00B005	01A206	959A01	00E00F
08	00ED8A	00ED8C	00A008	008001	062009	062009	070A08	038201
10	001001	00ED83	00ED87	00ED99	00ED9C	31821D	31821F	318221
18	318223	00E01A	00A01B	070A01	00D181	21881E	019801	298820
20	019801	118822	019801	198824	019801	018110	000002	000003
28	000004	000005	000006	000007	000008	000009	00000A	00000B
30	00000C	00000D	00000E	00000F	000010	000011	000012	000013
38	000014	000015	000016	000017	000018	000019	00001A	00001C

图 10 rom_a.mif 中的数据

注意：地址 address 在写允许 wren 低电平后（wren 低电平为读允许）又重复了它在高电平时的地址值，这样便于验证已写进去的数据。数据从 ram_dp0 的左边 D[7..0]输入，从右边 Q[7..0]输出。