FPGA系统设计原则和技巧之：FPGA系统设计的3种常用IP模块

9.3*FPGA系统设计*的3种常用*IP模块*

FPGA的开发工具软件，如QuartusII、ISE等，一般都会提供一些经过验证的*IP模块*。这些*IP模块*是芯片厂家提供的，所以只能用于该厂家的FPGA芯片设计中。这些IP主要包括以下几类。

·算术类，如乘法器、加法器、除法器等。

·逻辑门类，如与门、或门、非门等。

·*存储器*类，如FIFO、RAM、ROM、移位寄存器等。

·I/O类，如双向IO、PLL、LVDS等。

·接口类，如以太网MAC、PCI接口控制器、*高速串行收发器*（SERDES）等。

·商业IP核，需要付费购买的。

以上的IP中，最常用的3个IP分别是片上*存储器*（RAM/ROM/FIFO），*锁相环*（PLL/DLL）和*高速串行收发器*（SERDES）。灵活使用这些IP，可以提升设计的性能，同时降低设计的复杂度。

IP模块的实现方式主要有3种：使用HDL代码描述、使用综合约束属性例化或类推、使用器件商的IPcore生成器。

典型功能的IPcore都可以通过这3种途径实现。其中前两种方法需要学习综合RAM、ROM、CAM等存储单元的CodingStyle或约束属性，后一种方法非常方便、直接，建议初学者首先要掌握使用器件商IPcore生成器设计RAM、ROM、CAM等存储单元的方法。

下面通过QuartusII工具分别介绍Altera的3种IP模块的使用方法。

9.3.1片上*存储器*的使用方法

片上存储器由内嵌于FPGA内部的逻辑资源（分布RAM和块RAM）搭建而成。搭建出来的这些片上存储器的调用方法与分立器件基本一致，但却为开发者省去了大量的PCB布线资源。在小容量的存储器设计中有着非常广泛的应用，同时这些片上存储器还常常用作数据缓冲及时钟域转换等工作。

首先来介绍片上FIFO的使用方法。

1．FIFO

（1）打开宏模块向导管理器。

在QuartusII中，IP模块的生成都是通过“MegaWizardPlug-InManager”（宏模块向导管理器）实现的，它可以通过如图9.9的“Tools”菜单打开。

（2）选择新建宏模块。

在宏模块向导管理器的第1页，选择新建一个自定义的宏模块，如图9.10所示。

图9.9宏模块向导管理器调用 图9.10新建宏模块

（3）选择宏模块。

在宏模块向导管理器的第2页，管理器为我们提供了支持的宏模块树形目录。通过在该目录中选择相应的宏模块实现调用。同时在这一页中还可以选择应用的FPGA器件系列和宏模块的描述语言，并使用用户自定义的模块名，如图9.11所示。

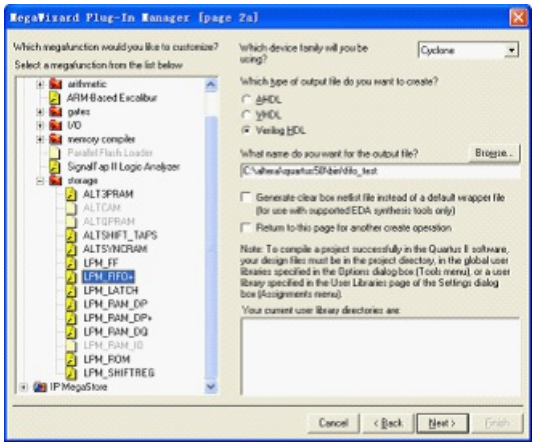

图9.11选择宏模块

在本例中，我们选择“LPM_FIFO+模块”进行实现。

（4）设置FIFO宽度和深度。

在宏模块向导管理器（FIFO）的第3页，可以设置FIFO的深度和宽度，同时在本页的左下角会计算出实现这样一个深度和宽度的FIFO所消耗的FPGA资源，如图9.12所示。

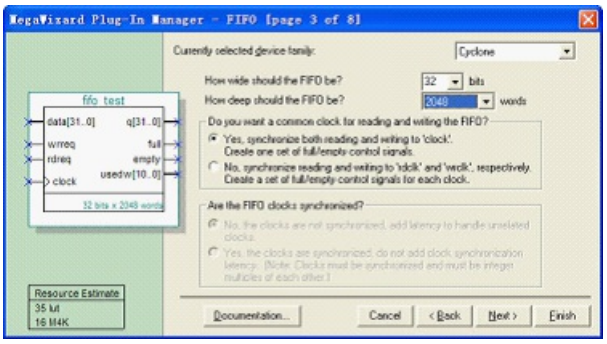

图9.12设置FIFO深度和宽度

（5）设置FIFO的控制信号。

在宏模块向导管理器（FIFO）的第4页，可以设置FIFO的控制信号，包括满信号full、空信号empty、使用字节信号组usedw[]、几乎满信号almostfull（可编程）、几乎空信号almostempty（可编程）、异步清零信号和同步清零信号。

通过选择是否打开这些信号来构造一个用户自定义的FIFO，如图9.13所示。

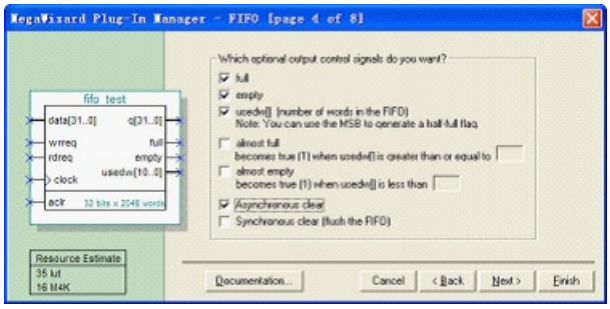

图9.13设置FIFO的控制信号

（6）设置FIFO的模式。

在宏模块向导管理器（FIFO）的第6页，可以设置FIFO的模式。分为Lagacy同步模式和Show-ahead同步模式。区别在于输出数据是在FIFO的读请求信号rdreq发出之前还是之后有效，用户可以根据需要进行选择，如图9.14所示。

（7）设置FIFO的外部属性。

在宏模块向导管理器（FIFO）的第7页，可以设置FIFO的外部属性，包括输出寄存器使用最佳速度策略还是最小面积策略，数据溢出及读空状态下的保护机制，还可以强制只利用逻辑单元来构造FIFO，如图9.15所示。

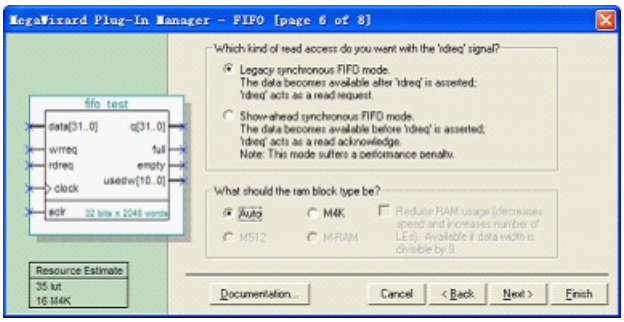

图9.14设置FIFO的模式

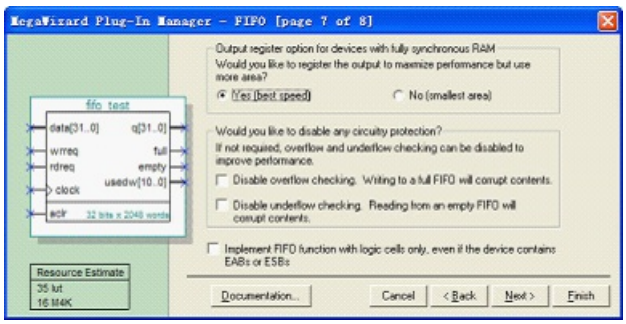

图9.15设置FIFO的外部属性

（8）选择生成的FIFO模块文件。

在宏模块向导管理器（FIFO）的第8页，也就是最后一页，可以选择生成的FIFO模块文件。在QuartusII软件中，宏模块向导管理器可为FIFO生成7个文件，如图9.16所示。

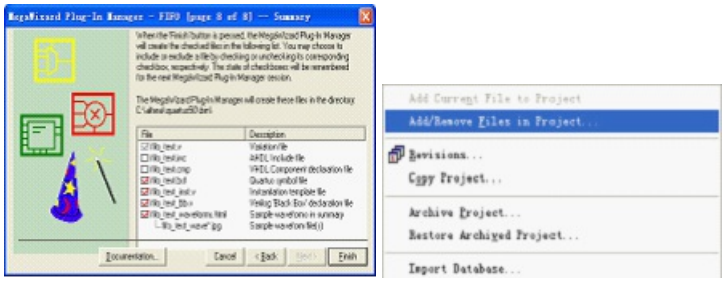

图9.16选择生成的FIFO模块文件图9.17打开工程添加/删除文件对话框

（9）向工程添加FIFO模块文件。

生成FIFO模块的文件后，在工程的目录下生成了选择的文件。要在工程中调用这些模块，首先要将这些文件添加到工程中来。

打开“Project”菜单，选择其中的“Add/RemoveFilesinProject…”选项，如图9.17所示，打开工程添加/删除文件对话框。

在打开的对话框中，选择向工程添加3个文件，如图9.18所示。

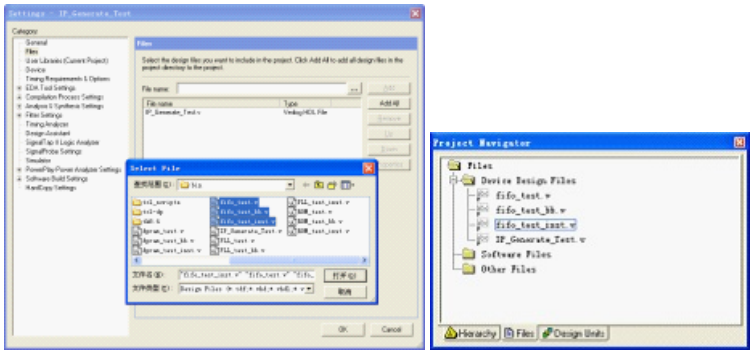

图9.18添加FIFO模块文件图 9.19工程浏览器

添加成功后，在工程浏览器中，可以看到器件设计文件中已经包含了这3个文件，如图9.19所示。

（10）FIFO模块实例化。

将模块文件加入工程后，设计者就可以调用这个片上FIFO，并实例化了。

首先我们看一下QuartusII是如何构建这个FIFO的。打开fifo_test.v文件，这是FIFO模块的构造设计文件。该构造设计文件主要可以分为下面4个

部分。

第一部分是端口声明和说明，这部分是根据用户自定义选择的控制信号及深度、宽度等参数决定的，代码如下：

```
modulefifo_test(
data, //数据输入
wrreq, //写请求
rdreq, //读请求
clock, //时钟
aclr, //异步清零
q, //数据输出
full, //满信号
empty, //空信号
usedw); //字节使用信号
input [31:0]data; //输入数据宽度
input wrreq;
input rdreq;
input clock;
input aclr;
output [31:0]q; //输出数据宽度
output full;
output empty;
output [10:0]usedw; //自动计算出的字节使用控制信号宽度
```

第二部分是FIFO端口信号与Altera宏模块之间的连线声明，其中只有输出信号需要声明，而输入信号可以直接调用，代码如下：

```
wire[10:0]sub_wire0; //usedw输出类型声明
wiresub_wire1; //空信号输出类型声明
wire[31:0]sub_wire2; //数据输出类型声明
wiresub_wire3; //满信号输出类型声明
wire[10:0]usedw=sub_wire0[10:0]; //usedw连线
wireempty=sub_wire1; //空信号连线
wire[31:0]q=sub_wire2[31:0]; //数据输出连线
wirefull=sub_wire3; //满信号连线
```

第三部分是实例化FIFO所调用的宏模块，这里调用的是scfifo模块，代码如下：

```
scfifoscfifo_component( //scfifo模块实例化
.rdreq(rdreq), //读请求
.aclr(aclr), //异步清零
.clock(clock), //时钟
.wrreq(wrreq), //写请求
.data(data), //输入数据
.usedw(sub_wire0), //字节使用信号
.empty(sub_wire1), //空信号
.q(sub_wire2), //数据输出
.full(sub_wire3) //满信号
```

.almost_empty(), //几乎空信号

.almost_full(), //几乎满信号

.sclr() //清零信号

.);

第四部分是参数设置，这里设置的是scfifo模块的参数，包括宽度、深度以及之前生成步骤中设置的那些参数，代码如下：

defparam

scfifo_component.lpm_width=32, //宽度为32位

scfifo_component.lpm_numwords=2048, //深度为2048字节

scfifo_component.intended_device_family=Cyclone,//器件族为Cyclone

scfifo_component.lpm_type=scfifo, //调用lpm为scfifo

scfifo_component.lpm_showahead=OFF, //关闭showahead模式

scfifo_component.overflow_checking=ON, //打开溢出校验

scfifo_component.underflow_checking=ON, //打开读空校验

scfifo_component.add_ram_output_register=ON; //使用输出寄存器

在生成的3个FIFO模块文件中，还包含一个fifo_test_bb.v文件，这个文件包含的是fifo_test.v文件的第一部分内容，也就是说这个文件是FIFO模块的端口声明模块。

剩下的一个文件是fifo_test_inst.v，inst是实例化的简称。在这个文件里面提供了FIFO模块实例化的模板，代码如下：

fifo_test fifo_test_inst(//fifo_test模块实例化

.data(data_sig), //输入数据

.wrreq(wrreq_sig), //写请求

.rdreq(rdreq_sig), //读请求

.clock(clock_sig), //时钟

.aclr(aclr_sig), //异步清零

.q(q_sig), //输出数据

.full(full_sig), //满信号

.empty(empty_sig), //空信号

.usedw(usedw_sig) //字节使用信号

);

通过复制这个模板至工程设计文件中，即可实现FIFO模块的实例化。在进行实例化时，可以修改实例化的名称以及FIFO模块端口信号线的连线资源名称。

下面是将FIFO实例化在IP_Generate_Test工程中的代码。

moduleIP_Generate_Test(...); //调用FIFO模块的设计文件模块

... //连线资源声明

fifo_test fifo_test_inst1(//实例化为fifo_test_inst1

.data(fifo_data), //FIFO数据输入为fifo_data

.wrreq(fifo_wrreq),

.rdreq(fifo_rdreq),

.clock(fifo_clock),

.aclr(fifo_aclr),

.q(fifo_q),

.full(fifo_full),

.empty(fifo_empty),

.usedw() //未实例化的端口保留空白表示不使用该端口

);

...

endmodule

至此，已经在一个工程中实现了片上FIFO的调用。

2．RAM

这里主要介绍双口RAM模块的生成方法，模块的构造方法及调用方法与FIFO模块类似，不详细介绍。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见FIFO的方法。

（2）选择宏模块。

在本例中，我们选择LPM_RAM_DP模块进行实现，如图9.20所示。

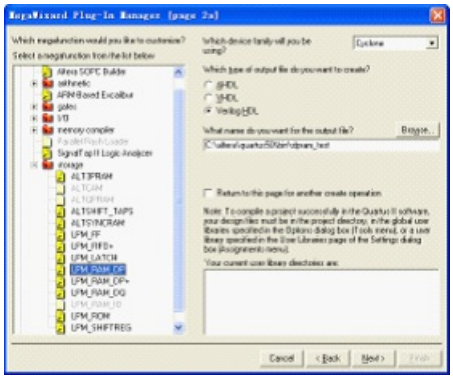

图9.20选择宏模块

（3）设置DPRAM端口数及容量单位。

在宏模块向导管理器（DPRAM）的第3页，可以设置DPRAM的端口数及容量单位。支持两种DPRAM：1个读端口和1个写端口的DPRAM，2个读端口和2个写端口的DPRAM。可以设置DPRAM的容量单位为位或者是字节。同时在本页的左下角会计算出实现这样一个DPRAM所消耗的FPGA资源，如图9.21所示。

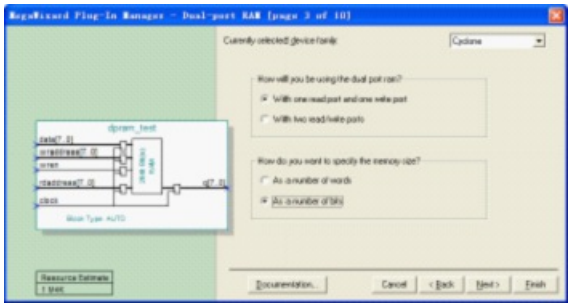

图9.21选择宏模块

（4）设置DPRAM数据宽度及容量。

在宏模块向导管理器（DPRAM）的第4页，可以设置DPRAM的数据宽度及容量，同时可以设置输入输出端口为不同的宽度实现串并或并串转换，如图9.22所示。

（5）设置DPRAM时钟及使能。

在宏模块向导管理器（DPRAM）的第5页，可以设置DPRAM的时钟及使能信号，可以将DPRAM设置为单时钟（输入输出使用同一个时钟），也可以使用独立的时钟。另外还可以为DPRAM增加读使能信号rden，如图9.23所示。

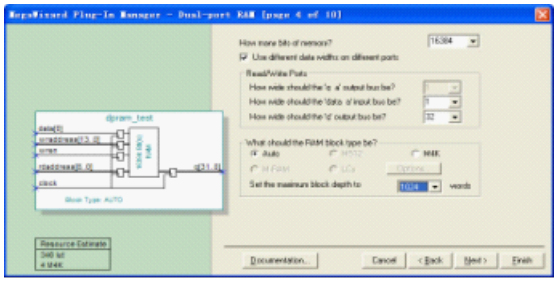

图9.22设置DPRAM数据宽度及容量

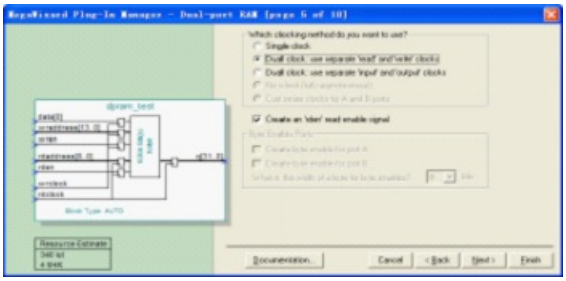

图9.23设置DPRAM时钟及使能

（6）设置DPRAM端口寄存器及清零信号。

在宏模块向导管理器（DPRAM）的第7页，可以设置DPRAM的端口寄存器及清零信号，可以选择是否增加端口的寄存器，如图9.24所示。

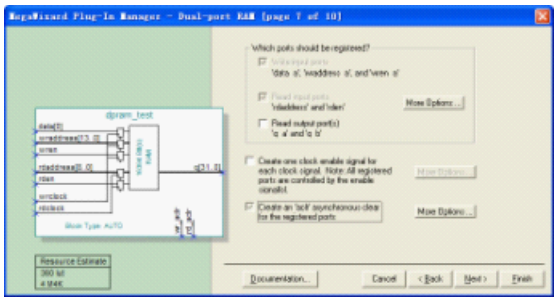

图9.24设置DPRAM端口寄存器及清零信号

（7）设置DPRAM初始化值。

在宏模块向导管理器（DPRAM）的第9页，可以设置DPRAM的初始化值。通过选择MIF文件或者HEX文件可以使用文件中的值对DPRAM进行初始化，如图9.25所示。

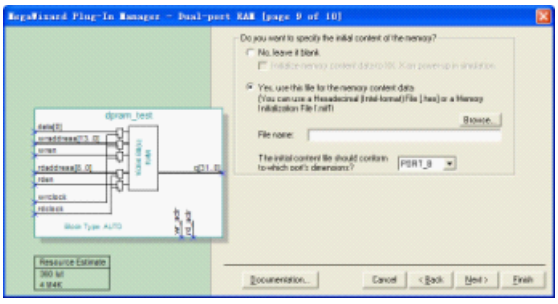

图9.25设置DPRAM初始化值

（8）生成DPRAM模块文件。

设置完成所有的参数，来到宏模块向导管理器（DPRAM）的最后一页。可以选择生成的DPRAM模块文件，共有6个文件可以生成，如图9.26所示。

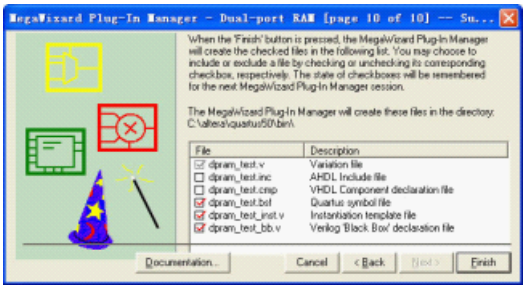

图9.26生成DPRAM模块文件

DPRAM的调用方式与FIFO的类似，其构建方式及模块的声明结构可以参看DPRAM模块的dpram_test.v和dpram_test_bb.v文件。

3. ROM

这里主要介绍ROM模块的生成方法，模块的构造方法及调用方法与FIFO模块类似，就不详细介绍了。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见FIFO的方法。

（2）选择宏模块。

在本例中，我们选择LPM_ROM模块进行实现，如图9.27所示。

（3）设置ROM宽度、深度及时钟控制方式。

在宏模块向导管理器（DPRAM）的第3页，可以设置DPRAM的宽度、深度及时钟控制方式。ROM宽度指的是存储器数据位宽，深度指的是具有该宽度的存储单元个数。宏模块向导管理器会根据设计者的选择自动生成ROM的地址总线，同时也会在左下角显示消耗的FPGA逻辑资源情况。

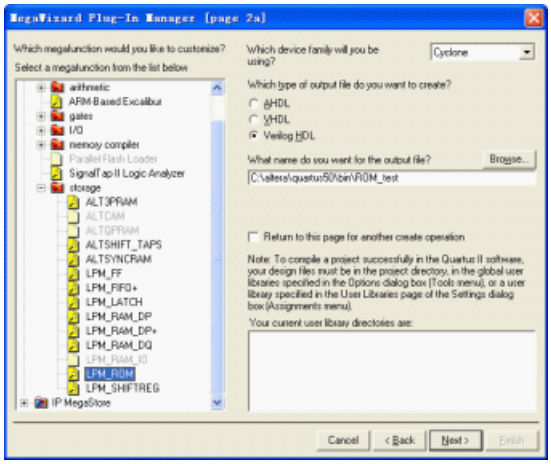

图9.27选择宏模块

ROM的时钟控制方式分为两种，一种为单时钟，另一种将输入时钟和输出时钟分离。ROM的输入输出使用的是同一个数据端口，时钟的分离实际上也是对同一个数据端口作控制，如图9.28所示。

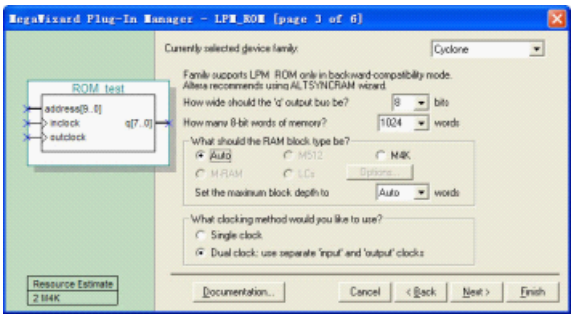

图9.28设置ROM宽度、深度及时钟控制方式

（4）设置端口寄存器及清零信号。

这个步骤与RAM中的第（6）步类似，不再详述，如图9.29所示。

（5）设置ROM初始化值。

同样可以参看RAM使用方法的第（7）步，如图9.30所示。

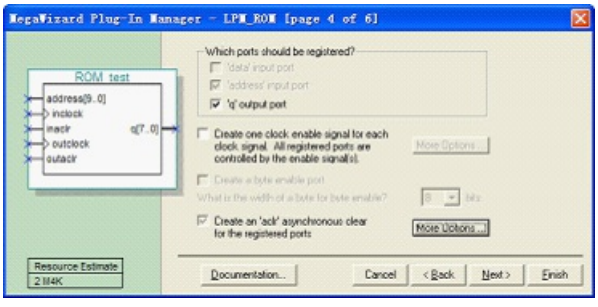

图9.29设置端口寄存器及清零信号

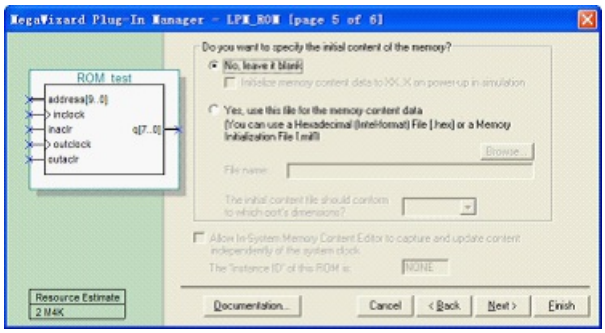

图9.30设置ROM初始化值

（6）选择生成ROM模块文件。

设置完成所有的参数，来到宏模块向导管理器（ROM）的最后一页。可以选择生成的ROM模块文件，同样有6个文件可以生成，如图9.31所示。

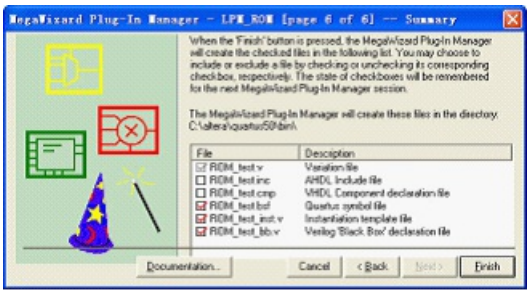

图9.31生成ROM模块文件

ROM的调用方式与FIFO的类似，其构建方式及模块的声明结构可以参看ROM模块的ROM_test.v和ROM_test_bb.v文件。

9.3.2*锁相环*产品的使用方法

*锁相环*产品在数字系统中的应用非常广泛，在Altera的FPGA产品中集成的*锁相环*产品是PLL模块，在Xilinx的FPGA产品中集成的*锁相环*产品是DLL模块。两者采用的技术不同，但是实现的基本功能大致相同，但也各有特点。

*锁相环*产品的原理及在FPGA内部的构造这里将不做介绍，只以Altera的CycloneFPGA的PLL为例讲解其使用方法。

CyclonePLL具有时钟倍频和分频、相位偏移、可编程占空比和外部时钟输出，进行系统级的时钟管理和偏移控制。PLL常用于同步内部器件时钟和外部时钟，使内部工作的时钟频率比外部时钟更高，时钟延迟和时钟偏移最小，减小或调整时钟到输出（TCO）和建立（TSU）时间。

（1）打开宏模块向导管理器并新建宏模块。

具体方法参见FIFO的方法。

（2）选择宏模块。

在本例中，我们选择ALTPLL模块进行实现，如图9.32所示。

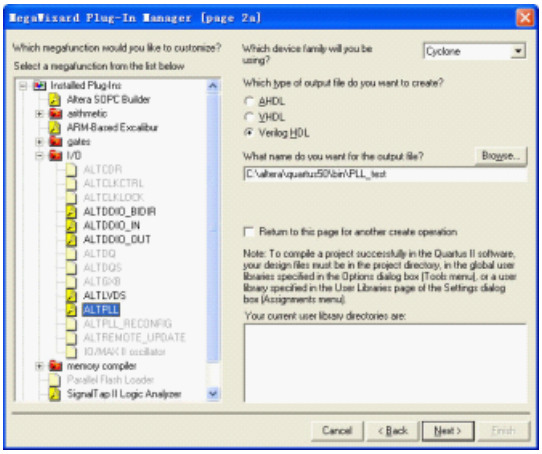

图9.32选择宏模块

（3）设置时钟源，PLL类型及工作模式。

在宏模块向导管理器（PLL）的第3页，设计者可以设置一些PLL基本属性，如器件族、速度级别及基准频率，还可以设置PLL的类型及工作模式。

值得注意的是，这里的基准频率并不一定是最终设计者向PLL输入的时钟。最终的PLL是通过比例值来对输入频率进行综合，所以PLL的关键参数是比例值，即图9.33中PLL_test框图中的Ratio值。

例如Ratio值为5/2（通过后面几个步骤实现），而基准频率为40MHz，则PLL的c0输出为100MHz。在调用这个PLL时，若设计者使用20MHz的频率输入，则输出的频率为50MHz，而不是100MHz。

另外，在这页中，设计者还可以设置PLL的工作模式，如图9.33所示。

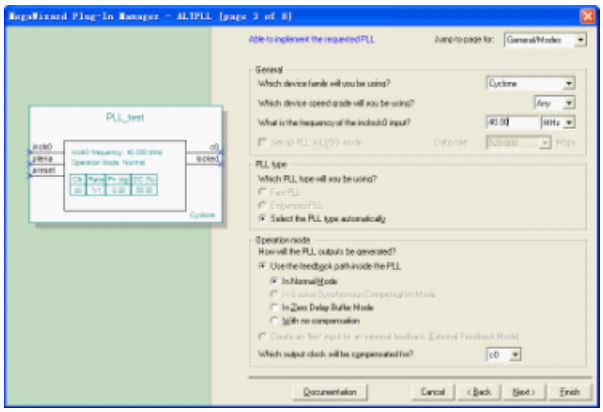

图9.33设置时钟源、PLL类型及工作模式

CyclonePLL支持3种（反馈）模式：标准、零延迟缓冲和无补偿。和其他Altera器件系列不同，CyclonePLL不支持外部反馈模式。所有支持的3种时钟反馈模式允许倍频/分频、相位偏移和可编程占空比。下面对每种模式进行讲解。

·标准模式。

在标准模式下，PLL把参考时钟和逻辑阵列或I/O单元的端口缓存器处的时钟信号相位对齐，补偿内部全局时钟网络延迟。在ALTPLL宏模块向导管理器中，可以定义PLL的哪个内部时钟输出（c0或c1）应该补偿。

如果在该模式中使用外部时钟输出（PLL[2..1]_OUT），则相对于时钟输入管脚有相位偏移。如果用内部PLL时钟输出驱动通用I/O管脚，则相对应的时钟输入管脚也有相位偏移。

·零延迟缓冲模式。

PLL外部时钟输出管脚（PLL[2..1]_OUT）的时钟信号和PLL输入时钟是相位对齐的，没有延迟。如果用c[1..0]端口驱动内部时钟管脚，那么相对于输入时钟管脚有相位偏移。

·无补偿模式。

在该模式下，PLL不补偿任何时钟网络。这样会有更佳的抖动性能，因为反馈到PFD的时钟不经过某些电路。相对PLL时钟输入，PLL内部和外部时钟输出都有相位偏移。

（4）设置可选输入输出。

在宏模块向导管理器（PLL）的第4页，可以选择PLL模块的可选输出信号，如图9.34所示。

CyclonePLL有4个控制信号pllena、areset、prdena和locked，进行PLL管理。

·pllena。

PLL启动信号pllena用于启动PLL。当pllena为低时，PLL时钟输出端口为低，PLL失锁。当pllena再次变高时，PLL重新锁定并重新同步输入时钟。因此，pllena是高有效信号。

因为在CycloneFPGA中没有专用的pllena管脚，内部逻辑或任意通用I/O管脚都可以驱动pllena端口。因为每个PLL都有自己的pllena控制电路或共享通用的pllena电路，这样就很灵活。pllena信号是可选的，如果软件中没有启动它，端口内部就连接到VCC。

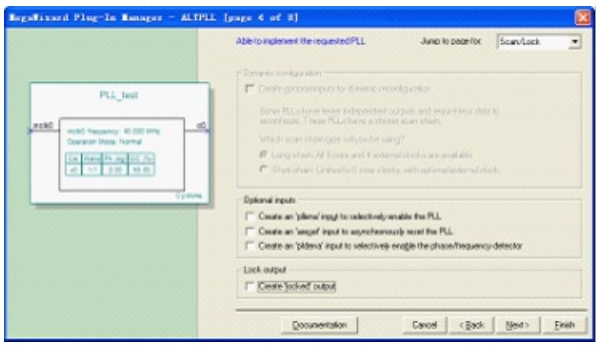

图9.34设置可选输入

·areset。

PLLareset信号是每个PLL的复位或重新同步输入。但驱动为高时，PLL计数器重置，清除PLL输出，造成PLL失锁，VCO复位后回到初始设

置。当areset再次变低，PLL重新开始锁定，PLL重新和输入时钟同步。如果目标VCO的频率低于标准频率，在锁定过程中PLL时钟输出起始频率值比所需值要高。areset是高有效信号。

CycloneFPGA可以从内部逻辑或任意通用I/O管脚驱动这个PLL输入信号。areset信号是可选的，如果在软件中没有使用它，该端口内部连接到GND。

·pfdena。

pfdena信号用可编程开关控制着PLL中PFD输出。如果把areset置低禁止PFD，那么VCO将以最后设置的控制电压和频率值工作，长时间会漂移到更低的频率。即使每个输入时钟PLL时钟输出也会继续触发，但是PLL可能会失锁。

当PLL失锁或输入时钟禁止时，系统会继续运行。因为在一段时间内最后锁定输出频率不会改变，所以可以用pfdena端口作为关机或清除功能。为了维持这一频率，系统在关机之前有时间储存当前的设置。如果pfdena信号再次变高，PLL重新锁定和输入时钟重新同步。因此pfdena管脚是高有效信号。

可以用任意通用I/O管脚或内部逻辑驱动pfdena输入信号。该信号是可选的，如果在软件没有使用它，该端口内部连接到VCC。

·locked。

当locked输出是逻辑高电平，表明PLL时钟输出和PLL参考输入时钟稳定同相。当PLL开始跟踪参考时钟时，locked端口可能会触发，无需额外电路。

PLL的locked端口可以使用任意通用I/O管脚和内部逻辑。这个locked信号是可选的，在监视PLL锁定过程中是非常有用的。

（5）设置c0输出参数。

在宏模块向导管理器（PLL）的第5页，可以选择PLL模块c0输出参数，包括Ratio比值、相移量、占空比等，如图9.35所示。

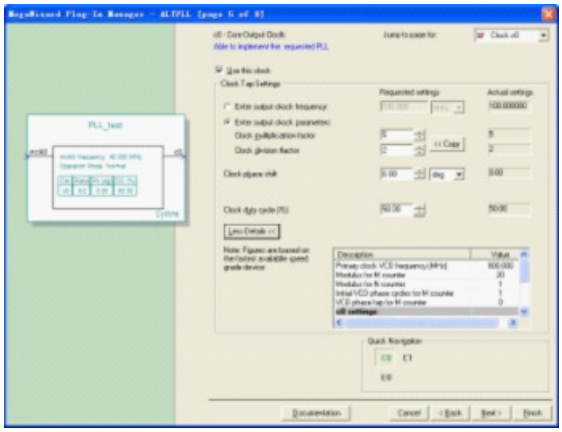

图9.35设置c0输出参数

（6）设置c1输出参数。

在宏模块向导管理器（PLL）的第6页，同样可以选择PLL模块c1输出参数，包括Ratio比值、相移量、占空比等，如图9.36所示。

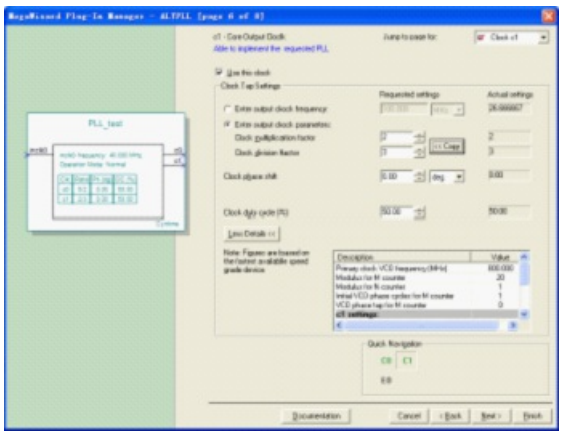

图9.36设置c1输出参数

（7）设置e0输出参数。

在宏模块向导管理器（PLL）的第7页，可以选择PLL模块e0输出参数，包括Ratio比值、相移量、占空比等，如图9.37所示。

图9.37设置e0输出参数

（8）选择生成PLL模块文件。

在宏模块向导管理器（PLL）的第8页，完成PLL参数设置后，可以选择生成的PLL模块文件，共包含6个文件可以生成，如图9.38所示。

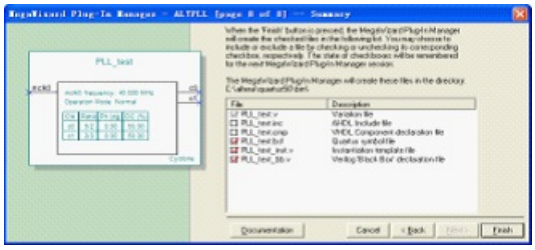

图9.38选择生成PLL模块文件

PLL的调用方式与FIFO的类似，其构建方式及模块的声明结构可以参看PLL模块的PLL_test.v和PLL_test_bb.v文件。

9.3.3*高速串行收发器*的使用方法

1．SERDES简介

随着对信息流量需求的不断增长，传统并行接口技术成为进一步提高数据传输速率的瓶颈。过去主要用于光纤通信的串行通信技术——SERDES正在取代传统并行总线而成为高速接口技术的主流。

SERDES是英文SERializer（串行器）/DESerializer（解串器）的简称。它是一种时分多路复用（TDM）、点对点的通信技术，即在发送端，多路低速并行信号被转换成高速串行信号，经过传输媒体（光缆或铜线），最后在接收端高速串行信号重新转换成低速并行信号。这种点对点的串行通信技术充分利用传输媒体的信道容量，减少所需的传输信道和器件引脚数目，从而大大降低通信成本。

随着半导体技术的迅速发展，计算机的性能和应用取得了长足进步。可是，传统并行总线技术PCI却跟不上处理器和存储器的进步，而成为提高数据传输速率的瓶颈。新一代PCI标准PCIExpress正是为解决计算机I/O瓶颈而提出的。

PCIExpress是一种基于SERDES的串行双向通信技术，每个通道的数据传输速率为2.5Gbit/s，可多达32通道，支持芯片与芯片和背板与背板之间的通信。国际互联网络和信息技术的兴起促成了计算机和通信技术的交汇，而SERDES串行通信技术逐步取代传统并行总线正是这一交汇的具体体现。

2．SERDES系统结构

基于SERDES的高速串行接口采用以下措施突破了传统并行I/O接口的数据传输瓶颈：一是采用差分信号传输代替单端信号传输，从而增强了抗噪声、抗干扰能力；二是采用时钟和数据恢复技术代替同时传输数据和时钟，从而解决了限制数据传输速率的信号时钟偏移问题。

一个典型SERDES收发机由发送通道和接收通道组成，如图9.39所示。编码器、串行器、发送器以及时钟产生电路组成发送通道，解码器、解串器、接收器以及时钟恢复电路组成接收通道。顾名思义，编码器和解码器完成编码和解码功能，其中8B/10B、64B/66B和不规则编码（scrambling）是最常用的编码方案。

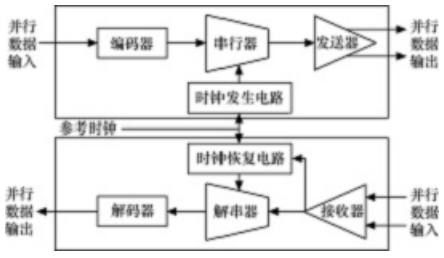

图9.39典型SERDES收发机结构图

串行器和解串器负责从并行到串行和从串行到并行的转换。串行器需要时钟产生电路，时钟发生电路通常由*锁相环*产品（PLL）来实现。解串器需要时钟和数据恢复电路（CDR），时钟恢复电路通常也由*锁相环*产品来实现，但有多种实现形式如相位插值、过剩抽样等。

发送器和接收器完成差分信号的发送和接收，其中LVDS和CML是最常用的两种差分信号标准。另外还有一些辅助电路也是必不可少的，例如环路（loopback）测试、内置误码率测试等。

3．SERDES实现方式

SERDES在系统中的实现虽然概念上比较简单，但是硬件实现要求很多细节正确无误，如信号端接、参考时钟的生成、*锁相环*产品（PLL）的使用、背板信号完整性和位错误率的评估等。

下面以Altera的StratixGXFPGA为例介绍其中的内嵌的数千兆位收发器功能块。

StratixGX器件将高速3.125Gbit/s收发器串行/解串行（SERDES）技术和业界最先进的FPGA架构相结合。在FPGA中嵌入的数千兆位收发器功能块，能够在需要灵活性、高性能和最先进功能的许多新系统中使用收发器。

下面是这个数千兆位收发器功能块的主要特点。

·支持从622MHz～3.125GHz的所有频率。

·每个块有4个独立的3.125Gbit/s全双工通道，每个器件多达20个通道（5个块）。

·支持3.1875Gbit/s的10Gbit光纤通道。

·集成时钟数据恢复（CDR）、模式检测、字对齐、8b/10b编解码器和同步功能。

·功耗很低，每4通道只有450mW（包括数千兆收发器功能块的功耗）。

·支持动态可编程：预加重、均衡和I/O缓冲上的差分输出电压。

·支持SerialLite协议，这是一个精简的点对点协议。

·对中等性能信号的差分片内匹配。

·完全实现10Gbit以太网物理介质接入层（PMA）和物理编码子层（PCS）功能。

·支持灵活的时钟拓扑，每个收发器模块中有一个发送器PLL和四个接收器PLL。

·采用1.5V、0.13mm全铜CMOS工艺技术制造，支持1.5VPCMLI/O标准。

·包括独立的发送器和接收器节电功能，在不工作过程中减小功耗。

·内嵌自检（BIST）功能，包括嵌入伪随机二进制序列（PRBS）图案生成和验证。

·有4个独立的环回路径用于系统验证。

可以看到，StratixGX的这个收发器功能非常强大，远远超出了图9.39所示的SERDES基本结构图的功能。图9.40是它的原理框图，需要说明的是，设计者在使用这个模块时并不一定完全实现其中的每一个模块，可以根据需要只实现其中的部分内容。

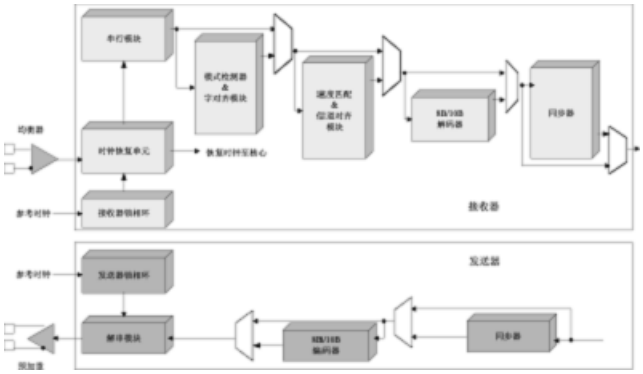

图9.40StratixGX收发机

下面介绍其中的一些模块。

（1）差分缓冲。

数千兆位收发器功能块差分I/O缓冲支持1.5VPCMLI/O标准，有许多改善系统信号完整性的特性。例如，动态可重配置预加重和均衡功能，调整信号以补偿信号经过传送介质的衰减。不同的可编程VOD设置确保了驱动强度匹配传输线电阻和线长。另外，差分片内匹配为中等性能信号提供了合适的接收器和发送器缓冲匹配。

（2）可编程发送预加重模块。

发送预加重块使收发器（SERDES）可以驱动更长的背板或超过1GHz频率下的电缆。在这些频率下，通道损耗是很高的，衰减是很明显的，因为眼图的关闭无法让发送的信号继续传送。预加重信号提升信号的高频部分，补偿传输线的衰减。使用可编程的预加重设置，能够为给定的传输线选择最优的水平（或者在软件设定，或者通过内部或外部信号动态选择），让信号眼图在远端张得最开。

（3）可编程接收均衡器模块。

接收均衡器块使得收发器（SERDES）驱动更长的背板或超过1GHz的电缆。当信号经过接收器均衡器块时，同样可以提升信号的高频部分能量，补偿传输线的高频衰减指标。可编程均衡水平可以根据传输线进行优化（或者在软件设定或者通过内部或外部信号动态选择），让信号眼图在CDR单元输入端张得最开。

（4）发送器和接收器PLL。

每个数千兆位收发器功能块有一个专用发送器PLL和四个专用接收器PLL，提供灵活的时钟拓扑，支持一系列的输入数据流。对于输出传送和接收，这些PLL根据更低速的输入参考时钟生成所需的时钟频率。每个PLL支持4、8、10、16或20的倍增因子。每个外部参考时钟或StratixGX内的各种时钟源都可以驱动PLL。

（5）时钟恢复单元。

CDR从输入串行数据流中提取时钟。恢复的时钟用于采样串行数据流、同步控制解串行器。

（6）串行/解串模块。

SERDES模块将输入的高速串行数据转换为更低速度的并行接口，反之亦然。SERDES模块可以配置为8、10、16或20位并行接口。

（7）模式检测器模块。

模式检测器模块识别输入数据流中的特殊模式。模式检测器包括一个内建的8b/10b的K28间隔符号检测和SONET的A1A2模式检测。在定制模式下，设计者可以创建专用模式。

（8）字对齐模块。

字对齐模块和模式检测器共同识别和调整正确的字节边界。此外，字对齐有一个定制模式，能够从FPGA内核逻辑人工地控制字节对齐。

（9）速率匹配器模块。

在串行数据传输中，发送和接收器件的时钟频率通常是不匹配的。这种不匹配会引起数据以略快或慢于接收器件能解释的速度传送。StratixGX速率匹配器从数据流中插入或删除如传送协议中定义的可移除的符号，解决了恢复时钟和PLD逻辑阵列时钟的频差，而不会丢失传送的数据。StratixGX数据匹配器为使用8b/10b编码数据的系统进行了优化。

（10）信道对齐器。

信道对齐器消除了实现4个收发器XAUI协议相关的信道至信道的偏移。信道对齐器消除4信道的偏移，为内核逻辑建立了可靠的以太网XGMII接口。

（11）8b/10b编解码器。

8b/10b编解码器模块将8比特的数据转换为10比特的数据，反之亦然。该算法平衡了串行数据流中“0”和“1”的数量，增加了变换密度，因此更易于接收器恢复串行数据。

（12）同步器。

同步器补偿了并行收发器接口和FPGA内核逻辑之间的相位差。

（13）内建自测。

BIST为收发器提供了一组强大的诊断能力。它包括伪随机二进制序列（PRBS）和其他图案的生成器和检查器。BIST也提供了4个环回配置用于系统诊断。