

# 第 10 章 DSP Builder 设计深入

应用 Matlab/DSP Builder 可以对多种类型的电子线路模块或系统进行建模、分析和硬件实现，且更擅长于一些较复杂的功能系统，及偏向于高速算法方面的模块的设计和实现，还能利用 HDL Import 模块将 HDL 文本设计转变成为 DSP Builder 元件。

本章将给出一些 DSP 及数字通信领域中实用模块的设计实例，以及基于 Matlab/DSP Builder 平台的 IP 核的应用。

## 10.1 FIR 数字滤波器设计

FIR (Finite Impulse Response: 有限冲激响应) 滤波器在数字通信系统中，被大量用于以实现各种功能，如低通滤波、通带选择、抗混叠、抽取和内插等。

在 DSP Builder 的实际应用中，FIR 滤波器是最为常用的模块之一。DSP Builder 的 FIR 滤波器设计方式有多种，作为示例，本节介绍基于模块的 FIR 与基于 IP 的 FIR 设计方法。

### 10.1.1 FIR 滤波器原理

对于一个 FIR 滤波器系统，它的冲激响应总是有限长的，其系统函数可以记为：

$$H(z) = \sum_{k=0}^M b_k z^{-k} \quad 10-1$$

最基本的 FIR 滤波器可用下式表示：

$$y(n) = \sum_{i=0}^{L-1} x(n-i)h(i) \quad 10-2$$

其中  $x(n)$  是输入采样序列， $h(n)$  是滤波器系数， $L$  是滤波器的阶数， $y(n)$  表示滤波器的输出序列。也可以用卷积来表示输出序列  $y(n)$  与  $x(n)$ 、 $h(n)$  的关系。

$$y(n) = x(n) * h(n) \quad 10-3$$

图 10-1 中显示了一个典型的直接 I 型 3 阶 FIR 滤波器，其输出序列  $y(n)$  满足下列等式：

$$h(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3) \quad 10-4$$

在这个 FIR 滤波器中，总共存在 3 个延时结，4 个乘法单元，一个 4 输入的加法器。如果采用普通的数字信号处理器（DSP Processor）来实现，只能用串行的方式顺序地执行延时、乘加操作，不可能在一个 DSP 处理器指令周期内完成，必须用多个指令周期来完成。

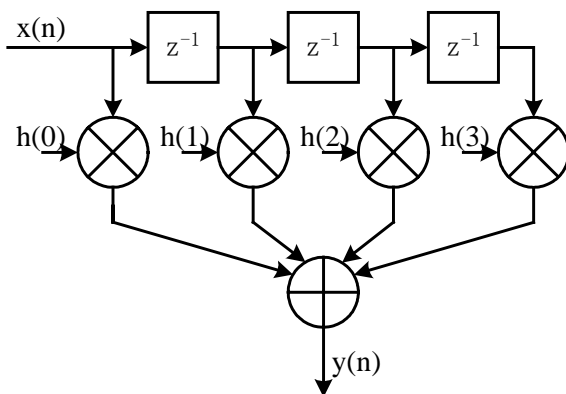


图 10-1 3 阶 FIR 滤波器结构

但是，如果采用 FPGA 来实现，就可以采用并行结构，在一个时钟周期内得到一个 FIR 滤波器的输出，不难发现图 10-1 的电路结构是一种流水线结构，这种结构在硬件系统中有利于并行高速运行。

### 10.1.2 使用 DSP Builder 设计 FIR 滤波器

使用 DSP Builder 可以方便地在图形化环境中设计 FIR 数字滤波器，而且滤波器系数的计算可以借助 Matlab 强大的计算能力和现成的滤波器设计工具来完成。

#### 1. 3 阶常数系数 FIR 滤波器设计

一个 3 阶 FIR 滤波器的  $h(n)$  可以表示为下式：

$$h(n) = C_q (h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3)) \quad 10-5$$

$$\begin{aligned} \text{其中：} \quad & h(0) = 63 \\ & h(1) = 127 \\ & h(2) = 127 \\ & h(3) = 63 \end{aligned}$$

$C_q$  是量化时附加的因子。这里采用直接 I 型来实现该 FIR 滤波器。利用 Matlab 设计好的 3 阶直接 I 型 FIR 滤波器模型图可以参见图 10-2。

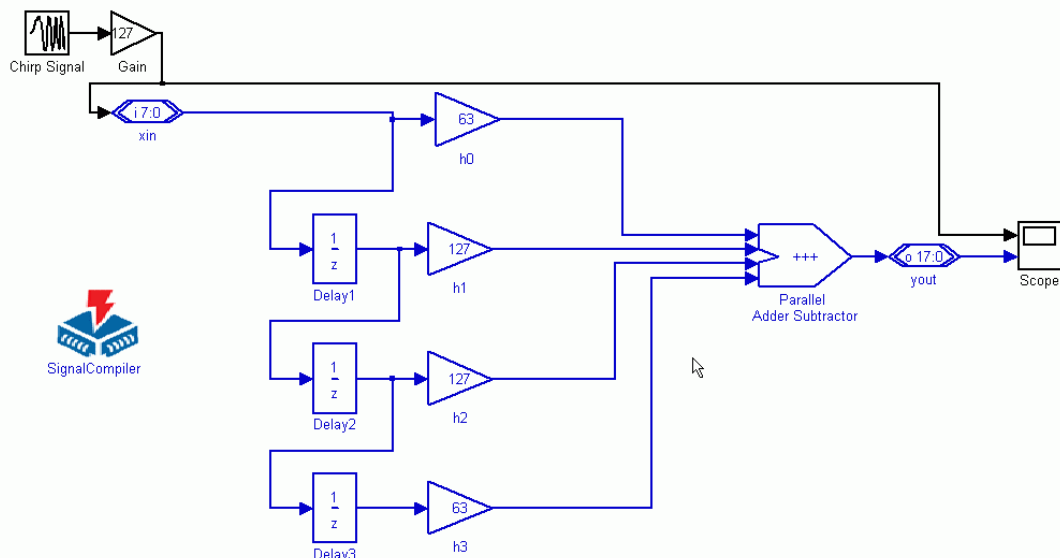


图 6-2 带有仿真信号模块的 3 阶滤波器模型

由于 FIR 滤波器的系数  $h(n)$  已经给定，是一个常数，由图中看到，在 DSP Builder 库中可以用 Gain（增益）模块来实现  $h(k) \times x(n-k)$  的运算，用延时 Delay 模块来实现输入信号序列  $x(n)$  的延时。设计完 3 阶 FIR 滤波器模型后，就可以添加 simulink 模块进行仿真了，如图 10-2 所示。新增的仿真模块参数作如下设置：

Chirp Signal 模块：(Chirp Signal)

库：Simulink 中 Sources 库

参数“Initial Frequency (Hz)”设为“0.1”

参数“Target time”设为“10”

参数“Frequency at target time (Hz)”设为“1”

参数“Interpret vectors parameters as 1-D”选中

Gain 模块：(Gain)

库：Simulink 中 Math Operations 库

参数“Gain”设为“127”

参数“Multiplication”设为“Element wise(K.\*u)”

Scope 模块：(Scope)

库：simulink 中 sinks 库

参数“Number of Axes”为“2”

其中 Chirp Signal 模块为线性调频信号发生模块，生成一个线性调频信号 0.1Hz~1Hz。

在该模型仿真中，使用默认的仿真参数。仿真结果如图 10-3 所示。一个线性调频信号通过 3 阶 FIR 滤波器后，幅度发生了变化，频率越高，幅度被衰减得越多。

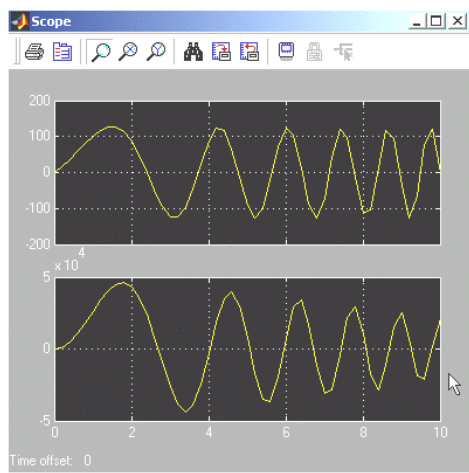


图 10-3 FIR 滤波器仿真结果

## 2、4 阶 FIR 滤波器节设计

以下将设计一个系数可变的 FIR 滤波器节。对于直接 I 型的 FIR 滤波器(结构见图 10-4)是可以级联的。也就是说,在滤波器系数可变的情况下,可以预先设计好一个 FIR 滤波器节,在实际应用中通过不断地调用 FIR 滤波器节,级联起来,用来完成多阶 FIR 滤波器的设计。

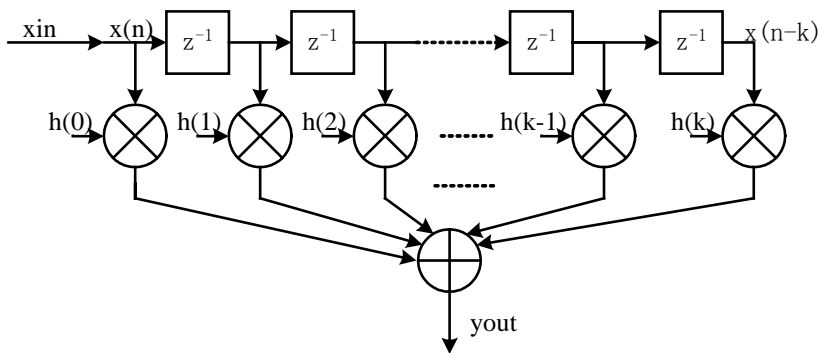


图 10-4 直接 I 型 FIR 滤波器结构

图 10-5 是一个直接 I 型的 4 阶 FIR 滤波器节的结构。为了使该滤波器节的调用更为方便,在  $x_{in}$  输入后插入了一个延时单元,由 3 阶滤波器演变成 4 阶的,不过常数系数项(  $z^0$  系数项)  $h(0)$  恒为 0。由于在通信应用中, FIR 滤波器处理的往往是信号流,增加一个延时单元不会影响 FIR 滤波器处理的结果,只是系统延时增加了一个时钟周期。

对于该 FIR 滤波器节,其系统函数可以用下式来表示:

$$H(z) = h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + h(4)z^{-4} \quad 10-6$$

由于浮点小数在 FPGA 中实现比较困难，实现的资源代价太大。在 DSP Builder 中不妨使用整数运算来实现，最后用位数舍取的方式得到结果。

为了使参数可变，FIR 滤波器系数  $h(1)$ 、 $h(2)$ 、 $h(3)$ 、 $h(4)$  也作为输入端口。在本设计中输入序列  $x(n)$  的位宽设为 9 位。图 10-6 显示的就是一个设计好的 4 阶 FIR 滤波器节，与图 10-2 常数 FIR 滤波器相比，用 Product（乘法）模块代替了 Gain（增益）模块。

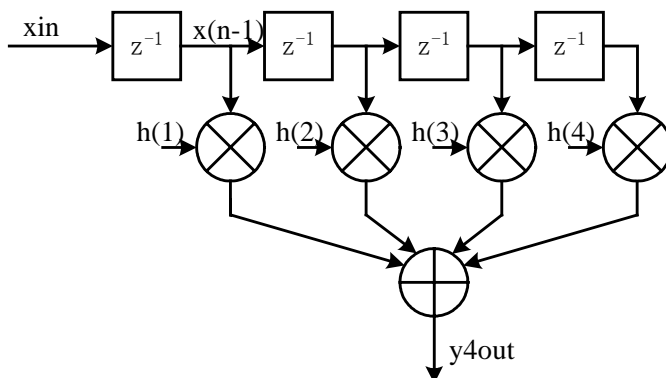


图 10-5 直接 I 型 4 阶 FIR 滤波器节

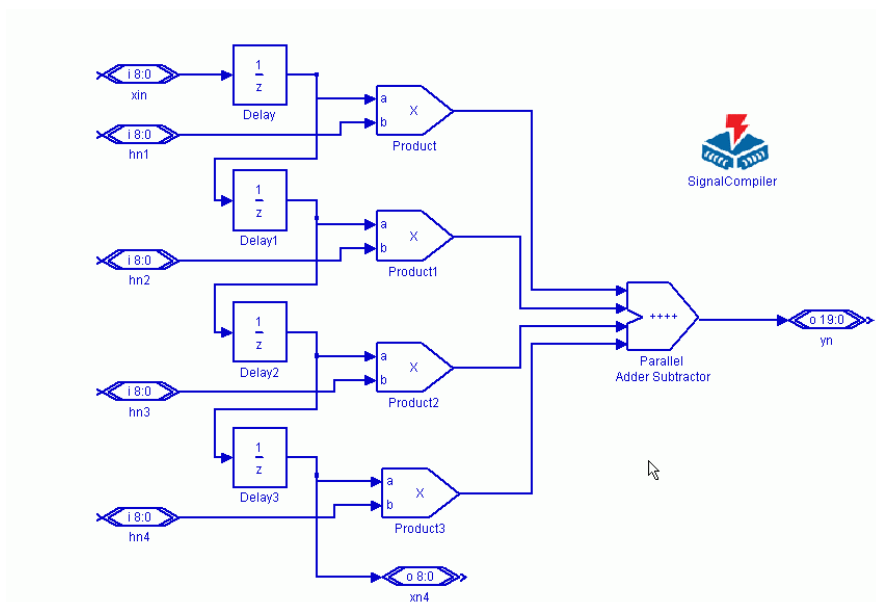


图 10-6 直接 I 型 4 阶 FIR 滤波器节

图 10-6 中相关模块的参数设置如下：

$x_{in}$ 、 $hn_0$ 、 $hn_1$ 、 $hn_2$ 、 $hn_3$  模块：(Altbus)

库：Altera DSP Builder 中 IO & Bus 库

参数“Bus Type”设为“signed Integer”

参数“Node Type”设为“Input port”

参数“number of bits”设为“9”

yn模块: (Altbus)

库: Altera DSP Builder中IO & Bus库

参数“Bus Type”设为“signed Integer”

参数“Node Type”设为“Output port”

参数“number of bits”设为“20”

xn4模块: (Altbus)

库: Altera DSP Builder中IO & Bus库

参数“Bus Type”设为“signed Integer”

参数“Node Type”设为“Output port”

参数“number of bits”设为“9”

Parallel Adder Subtractor模块: (Parallel Adder Subtractor)

库: Altera DSP Builder中Arithmetic库

参数“Add(+)Sub(-)”设为“++++”

使用“Pipeline”

参数“Clock Phase Selection”设为“1”

Delay、Delay1、Delay2、Delay3模块: (Delay)

库: Altera DSP Builder中Storage库

参数“Depth”设为“1”

参数“Clock Phase Selection”设为“1”

Product模块: (Product)

库: Altera DSP Builder中Arithmetic库

参数“Pipeline”设为“2”

参数“Clock Phase Selection”设为“1”

不选择“Use LPM”

### 3、16 阶 FIR 滤波器模型设计

利用以上设计的 4 阶 FIR 滤波器节可以方便地搭成  $4 \times n$  阶直接 I 型 FIR 滤波器(注意:  $h(0) = 0$ )。比如要实现一个 16 阶的低通滤波器,可以调用 4 个 4 阶 FIR 滤波器节来实现。

为了设计 4 阶 FIR 滤波器节子系统,首先需要建立一个新的 DSP Builder 模型,复制上节的 FIR4tap 模型到新模型。由 FIR4tap 模型建立子系统,并对端口信号进行修改,把子系统更名为 fir4tap,如图 10-7 所示。fir4tap 的内部结构示于图 10-8。

然后组成 16 阶 FIR 滤波器模型。为此复制 4 个 fir4tap,并将它们衔接起来,前一级的输出端口 x4 接后一级的 x 输入端口。并附加上 16 个常数端口,作为 FIR 滤波器系数的输入。把 4 个子系统 fir4tap 的输出端口 y 连接起来,接入一个 4 输入端口的加法器,得到 FIR 滤波器的输出 yout。注意,在完成子系统设计后,要按照第 9 章中所述的方式,修改其 Mask 参数 Mask Type 为“SubSystem AlteraBlockSet”。

设计好的 16 阶 FIR 滤波器见图 10-9 所示。

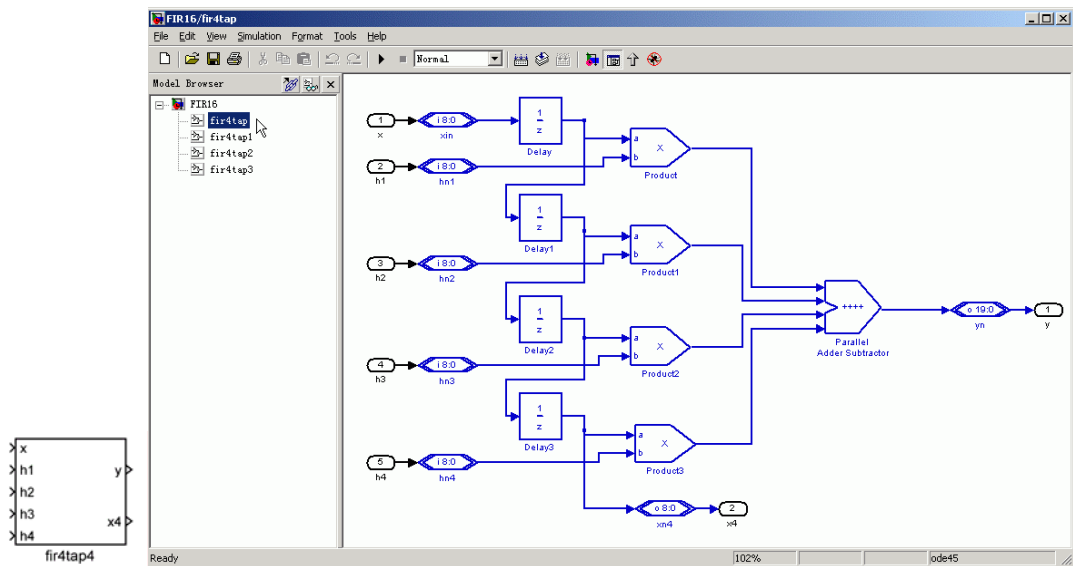


图 10-7 fir4tap 子系统

图 10-8 fir4tap 子系统内部原理图

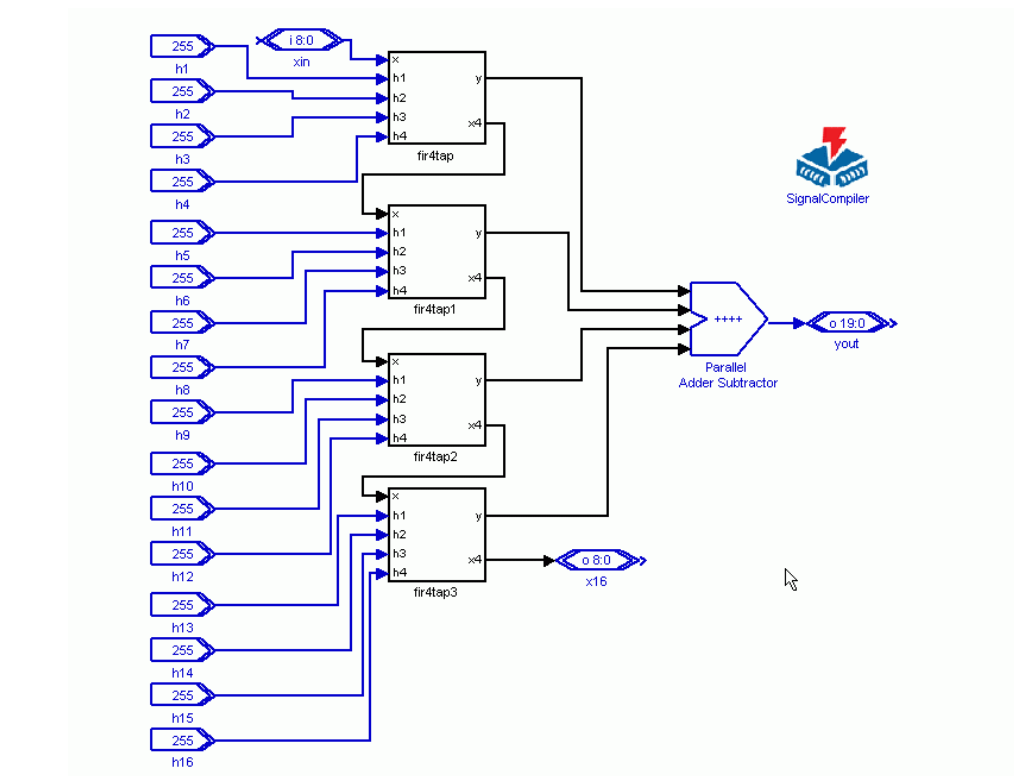


图 10-9 16 阶直接 I 型 FIR 滤波器模型

16 阶直接 I 型 FIR 滤波器模型中，新增加的模块作如下设置：

xin 模块：(Altbus)

库：Altera DSP Builder 中 IO & Bus 库

参数“Bus Type”设为“signed Integer”  
 参数“Node Type”设为“Input port”  
 参数“number of bits”设为“9”

yout模块: (Altbuss)  
 库: Altera DSP Builder中IO & Bus库  
 参数“Bus Type”设为“signed Integer”  
 参数“Node Type”设为“Output port”  
 参数“number of bits”设为“20”

x16模块: (Altbuss)  
 库: Altera DSP Builder中IO & Bus库  
 参数“Bus Type”设为“signed Integer”  
 参数“Node Type”设为“Output port”  
 参数“number of bits”设为“9”

Parallel Adder Subtractor模块: (Parallel Adder Subtractor)  
 库: Altera DSP Builder中Arithmetic库  
 参数“Add(+)Sub(-)”设为“++++”  
 使用“Pipeline”  
 参数“Clock Phase Selection”设为“1”

h0、h1、h2、h3、h4、h5、h6、h7、h8、  
 h9、h10、h11、h12、h13、h14、h15模块: (Delay)  
 库: Altera DSP Builder中IO & Bus库  
 参数“Bus Type”设为“Signed Integer”  
 参数“number of bits”设为“9”

注意, 在图 10-9 中, 对 h1~h16 统一设置了一个值: 255, 而实际上滤波器的系数要根据具体要求进行计算的。在系数计算后, FIR 滤波器才能真正应用。

### 10.1.3 使用 Matlab 的滤波器设计工具

可以十分方便地利用 Matlab 提供的滤波器设计工具获得各种滤波器的设计参数。这里以一个 16 阶的 FIR 滤波器 ( $h(0) = 0$ ) 为例, 滤波器指标参数如下:

- 低通滤波器
- 采样频率  $F_s$  为 48kHz, 滤波器  $F_c$  为 10.8kHz,
- 输入序列位宽为 9 位 (最高位为符号位)

在此利用 Matlab 来完成 FIR 滤波器系数的确定, 详细步骤如下:

#### 1、打开 Matlab 的 FDATool

Matlab 集成了一套功能强大的滤波器设计工具 FDATool (Filter Design & Analysis Tool), 可以完成多种滤波器的设计、分析和性能评估。

点击 Matlab 主窗口下方的“Start”开始按钮, 按图 10-10 选择“ToolBox”→“Filter Design”



→ “Filter Design & Analysis Tool (FDATool)”，打开 FDATool (如图 10-11 所示)。

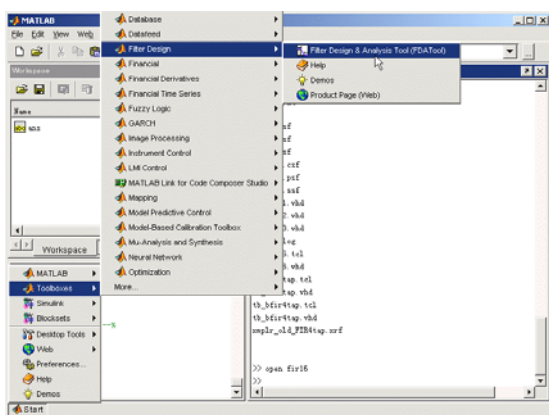


图 10-10 打开 FDATool

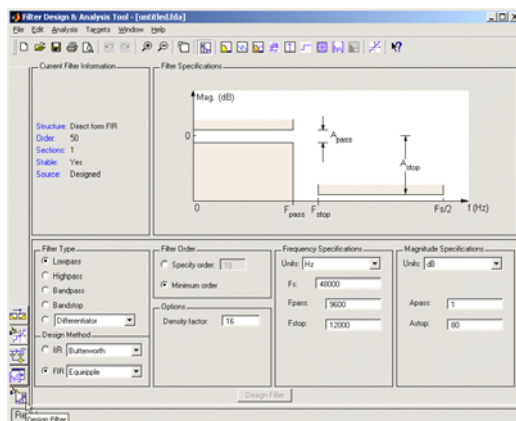


图 10-11 FDATool 界面

## 2、选择 Design Filter

FDATool 左下侧排列了一组工具按钮，功能分别是：



滤波器转换 (Transform Filter)



设置量化参数 (Set Quantization Parameters)



实现模型 (Realize Model)



导入滤波器 (Import Filter)



设计滤波器 (Design Filter)

选择其中的  按钮，进入设计滤波器界面，再选择：

- 滤波器类型 (Filter Type) 为低通 (Lowpass)
- 设计方法 (Design Method) 为 FIR，采用窗口法 (Window)
- 滤波器阶数 (Filter Order) 定制为 15
- 窗口类型为 Kaiser，Beta 为 0.5
- $F_s$  为 48kHz， $F_c$  10.8kHz

注意，在滤波器阶数选择时，在此设置的是 15 阶，而不是 16 阶！这是由于在前面设计的 16 阶 FIR 滤波器的常数系数项  $h(0) = 0$ 。

其系统函数  $H(z)$  可以用下式来表示：

$$H(z) = \sum_{k=1}^{16} b_k z^{-k} \quad 10-7$$

或可写成:

$$H(z) = z^{-1} \sum_{k=0}^{15} b_k z^{-k} \quad 10-8$$

即可以看成是一个 15 阶的 FIR 滤波器的输出结果经过了一个单位延时单元  $z^{-1}$ 。所以在 FDATool 中把它当成 15 阶 FIR 滤波器来计算参数。

点击 **Design Filter**，让 Matlab 计算 FIR 滤波器系数并作相关分析。

### 3、滤波器分析

计算完 FIR 滤波器系数后，往往需要对设计好的 FIR 滤波器进行相关的性能分析，以便了解是否满足设计要求。分析操作步骤如下：

选择 FDATool 的菜单“Analysis”→“Magnitude Response”，启动幅频响应分析。图 10-12 显示了滤波器的幅频响应图，x 轴为频率，y 轴为幅度值（单位为 dB）。

在图的左侧列出了当前滤波器的相关信息：

- 滤波器类型为：Direct Form FIR（直接 I 型 FIR 滤波器）
- 滤波器阶数为：15

注意，不是每一种 FIR 滤波器设计方法计算的滤波器都是直接 I 型结构的。如果在 DSP Builder 中设计的 FIR 滤波器为直接 I 型结构，那就必须保证在这里显示的 FIR 滤波器器结构为“Direct Form FIR”。

选择菜单“Analysis”→“Phase Response”，启动相频响应分析。图 10-13 显示了滤波器的相频响应，可以看到设计的 FIR 滤波器在通带内相位响应为线性的，即该滤波器是一个线性相位的滤波器。

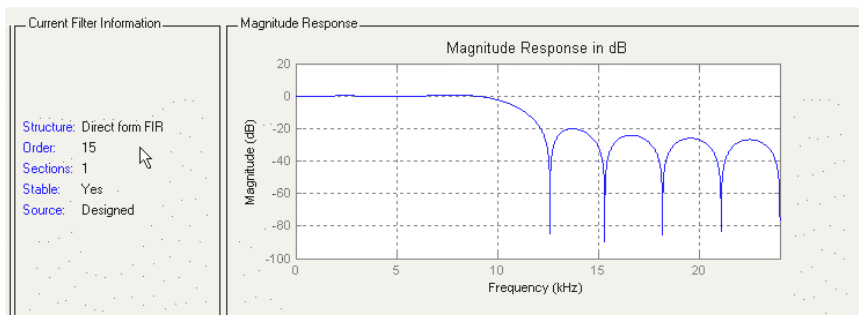


图 10-12 FIR 滤波器的幅频响应

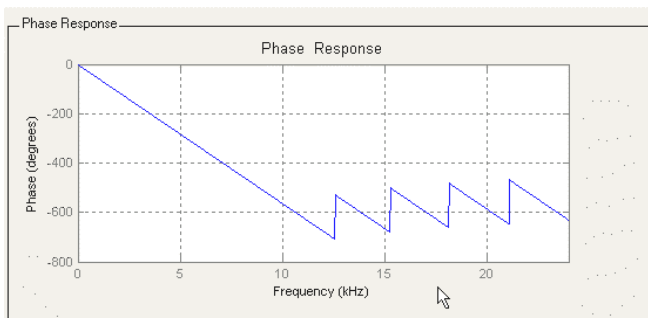


图 10-13 FIR 滤波器的相频响应

图 10-14 显示了滤波器幅频特性与相频特性的比较。这可以通过选择菜单“Analysis”→“Magnitude & Phase Response”，来启动分析。选择菜单“Analysis”→“Group Delay Response”，启动群延时分析，波形如图 10-15 所示。

在菜单“Analysis”下还有一些分析：

- “Impulse Response”：冲激响应，见图 10-16
- “Step Response”：阶跃响应，见图 10-17。
- “Pole/Zero Plot”，零极点图，见图 10-18。

由于直接 I 型 FIR 滤波器只有零点，所以在图 10-18 中没有极点的存在。

求出的 FIR 滤波器的系数可以选择菜单“Analysis”→“Filter Coefficients”来观察，图 10-19 列出了 FDATool 计算的 15 阶直接 I 型 FIR 滤波器部分系数。

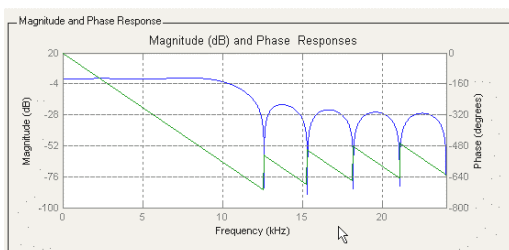


图 10-14 幅频响应与相频响应比较

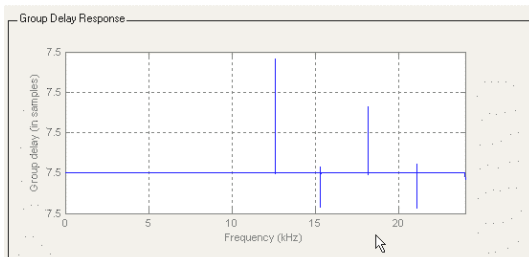


图 10-15 FIR 滤波器的群延时

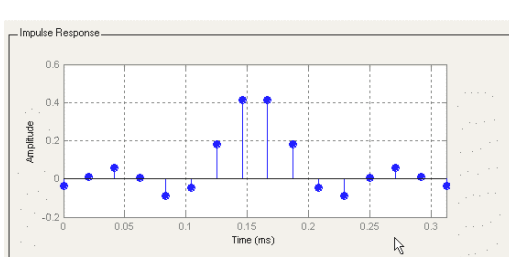


图 10-16 FIR 滤波器的冲激响应

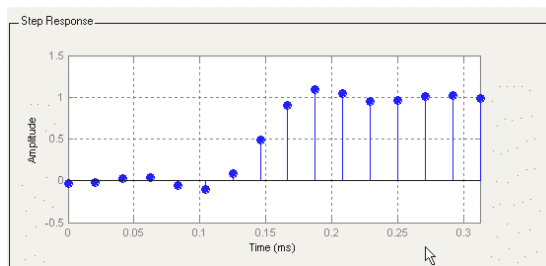


图 10-17 FIR 滤波器的阶跃响应

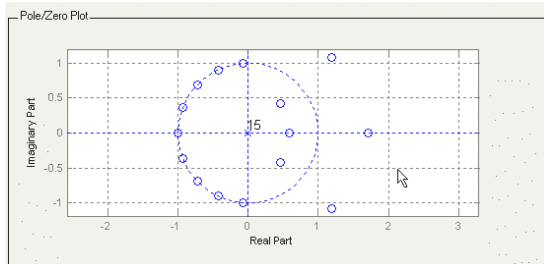


图 10-18 FIR 滤波器的零极点

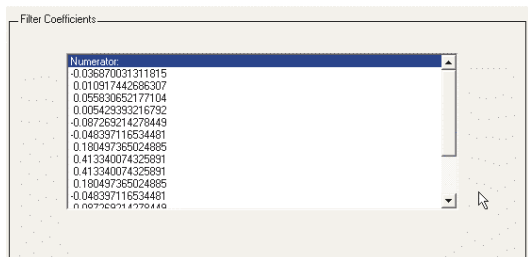



图 10-19 FIR 滤波器系数

#### 4、量化

从图 10-19 可以看到，FDATool 计算出的值是一个有符号小数，而在 DSP Builder 下建立的 FIR 滤波器模型需要一个整数（有符号整数类型）作为滤波器系数。所以必须进行量

化，并对得到的系数进行归一化。为此，点击 FDATool 左下侧工具按钮  进行量化参数设置。在设置“Turn quantization”前选择“√”，如图 10-20 所示。

在滤波器的设计指标中，已经提到 FIR 滤波器的输入位宽是 9 位的，表示为有符号数。在图 10-20 中设置前 4 项的量化格式（Format）为“[9 8]”，表示量化后位宽为 9 位，绝对值为 8 位；设置后 2 项（乘积、乘积和）的量化格式为“[32 30]”。

点击 Optimization 按钮，打开图 10-21 所示的对话框。在此量化优化设置对话框中，选择相关的优化选项。

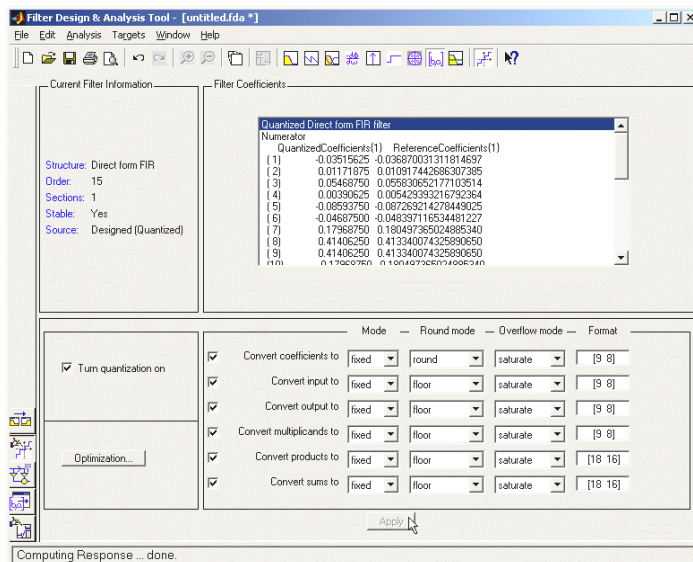


图 10-20 量化参数设置

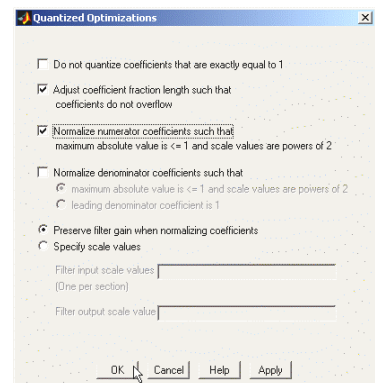


图 10-21 量化优化设置

在图 10-22 显示了量化后的部分系数值。注意在这里系数仍是用小数表示的，不同于量化前的系数，现在其二进制表示的位数已满足量化要求了。

设计的 FIR 滤波器在量化后，滤波器的性能会有所改变，其幅频响应、相频响应也有

所变化，见图 10-23 的量化后幅频、相频响应波形图。

量化在带来实现方便的同时也带来了量化噪声，图 10-24 显示了量化带来的噪声分析。

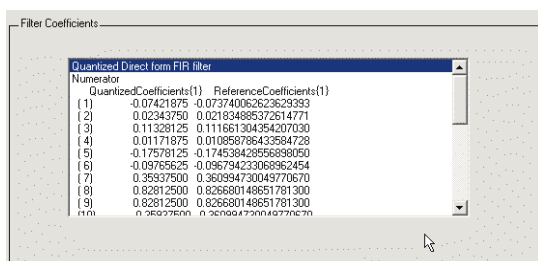


图 10-22 量化后系数

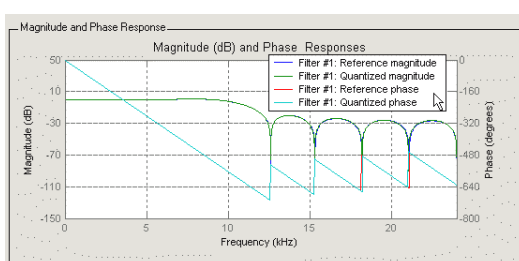


图 10-23 量化后幅频、相频响应

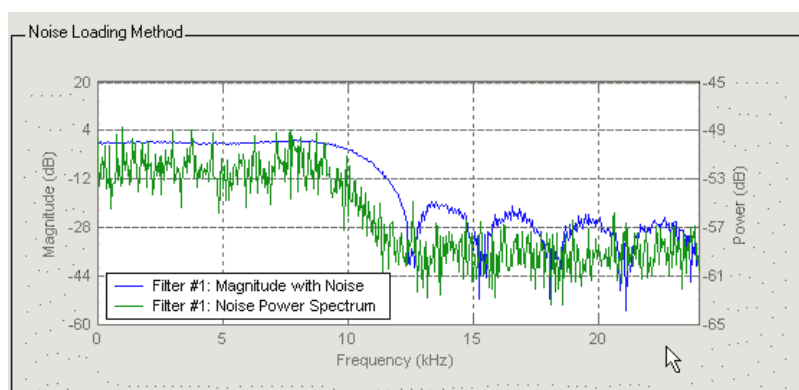


图 10-24 量化后噪声分析

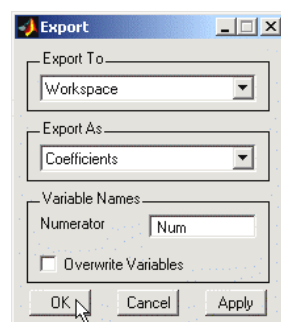


图 10-25 导出系数对话框

### 5、导出滤波器系数

为导出设计好的滤波器系数，选择 FDATool 菜单“File”→“Export...”，打开导出(Export)对话框，如图 10-25 所示的窗中，选择导出到工作区 (Workspace)。这时滤波器系数就存入了一个一维变量 Num。不过这时 Num 中的元素是以小数形式出现的：

```
Num =
    -0.0742    0.0234    0.1133    0.0117   -0.1758   -0.0977
     0.3594    0.8281    0.8281    0.3594   -0.0977   -0.1758
     0.0117    0.1133    0.0234   -0.0742
```

现在若要在 FIR 滤波器模型中使用这些数据，还需要将它们转化为整数。

在 Matlab 主窗口的命令窗口中键入：

```
Num * (2^8)
```

得到：

```
>> Num*(2^8)
ans =
Columns 1 through 10
   -19     6    29     3   -45   -25    92   212   212    92
Columns 11 through 16
   -25   -45     3    29     6   -19
```

## 6、修改 FIR 滤波器模型添加参数

修改图 10-9 的电路，把计算出的系数逐个填入到 FIR 滤波器模型中，见图 10-26。这样一个 16 阶直接 I 型 FIR 低通滤波器就设计完成了。

## 7、导出滤波器系数的另一种方法

在 FIR 滤波器阶数较大时，若按照以上的导出滤波器系数的方法，会不太方便，而且系数在设计要求有所变化时，修改极为不利。对此，可以按照以下方法来导出：

把 FIR 滤波器模型中的 h1~h16 模块的参数“Constant Value（常数值）”设置为：

$$\text{Num}(n) \cdot (2^8)$$

其中 Num 同上文所述，是 FDATool 的系数导出，n 用具体的数字来代替，如 h1 模块用 Num(1)\*(2^8)，h2 模块用 Num(2)\*(2^8)。

最后利用 SignalCompiler，选定器件系列，把模型转成 VHDL 文件，用 QuartusII 进行综合/适配，锁定管脚和下载至 FPGA 中，就可以完成硬件实现了。

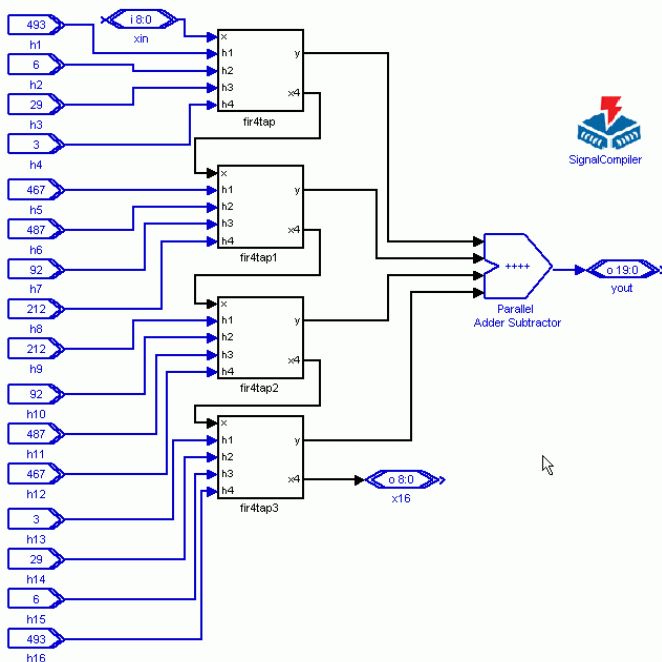


图 10-26 16 阶低通 FIR 滤波器

### 10.1.4 使用 FIR IP Core 设计 FIR 滤波器

对于一个面向市场和实际工程应用的系统设计，开发效率十分重要。然而对于一般的设计者，在短期内不可能全面了解 FIR 滤波器（指在 FPGA 上实现）相关的优化技术，也没有必要了解过多的细节。此外，FIR 滤波器滤波系数的确定，即 FIR 滤波器的设计方法，

也是比较烦琐的，需要花费大量的精力和时间才能设计出在速度、资源利用、性能上都满足要求的 FIR 滤波器。此外，虽然 DSP Builder 提供了大量的基本 DSP 模块，但是要了解用哪些模块构建一个高效的 FIR 滤波器仍然不是一件简单的事情。

但是，如果采用设计好的 FIR 滤波器的 IP 核，几乎可以很容易地解决以上的问题。对于 IP 核，在速度、资源利用、性能上往往进行过专门的优化，还提供了相关的 IP 应用开发工具。

Altera 提供的 FIR Compiler 是一个结合 Altera FPGA 器件的 FIR Filter Core, DSP Builder 与 FIR Compiler 可以紧密结合起来, DSP Builder 提供了一个 FIR Core 的应用环境和仿真验证环境。以下通过介绍采用 DSP Builder 和 FIR Compiler 设计一个高速低通 FIR 滤波器，来介绍 IP 在 Matlab 工具上的基本使用方法。

使用 FIR Core 之前，首先必须保证 Matlab、DSP Builder、QuartusII 以及 IP 核的本身，即 FIR Compiler 等工具安装正确。如果一切正常，就可以在 Simulink 库管理器的 Altera DSP Builder 库中看到 MegaCore Functions 子库中含有 FIR Compiler 等 IP 模块了（图 10-27）。

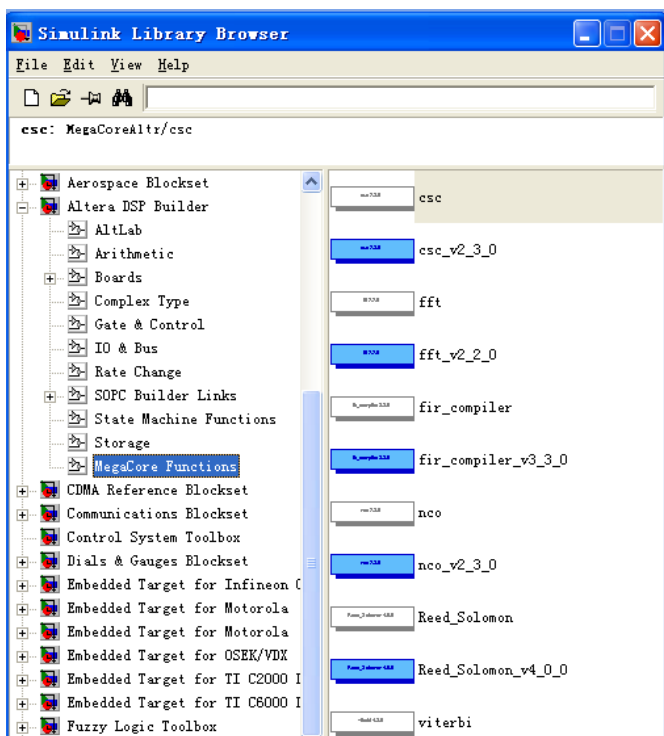


图 10-27 IP Core 模块库

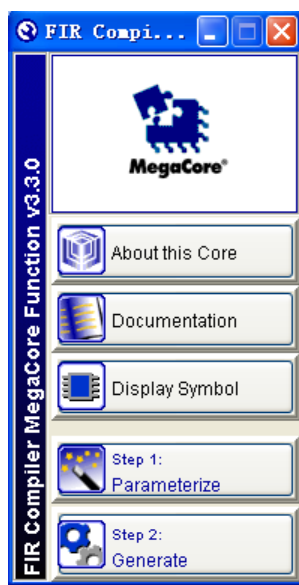


图 10-28 设置 FIR Core 参数

MegaCore 是 Altera 的 IP Core 计划中的一个组成部分，FIR Compiler 作为一个 MegaCore，不附带在 DSP Builder 和 QuartusII 中，需要单独向 Altera 公司购买或申请试用版。现在最新的 FIR Compiler 的版本可以支持 QuartusII 和 DSP Builder。

从图 10-27 可以看到，FIR Filter Core 在 DSP Builder 中是以模块的方式出现。可以使用第 9 章介绍的调用 DSP Builder 模块的方式来使用。步骤如下：

## 1、FIR 滤波器核的使用

为了调用 FIR Core, 在 Simulink 环境中新建一个模型, 放置 SignalCompiler 模块和 FIR 模块。注意, 在 DSP Builder 中使用 FIR Compiler 时, 需要有 SignalCompiler 的支持, 所以在使用配置 FIR 模块时, 必须放置 SignalCompiler 模块。

## 2、配置 FIR 滤波器器核

首先确定好 FIR 滤波器的设计指标。

双击新模型中的 FIR 模块, 弹出功能选择窗图 10-28, 点击 Parameterize 按钮, 打开 FIR 滤波器核的参数设置窗口, 进行 FIR 滤波器参数的配置 (图 10-29)。在这里, 设计者只要把设计要求直接输入到对应的设置框, 并点击“Apply (应用)”按钮, FIR 滤波器的系数就自动计算完成, 并在窗口中显示了设计的 FIR 滤波器的幅频特性。

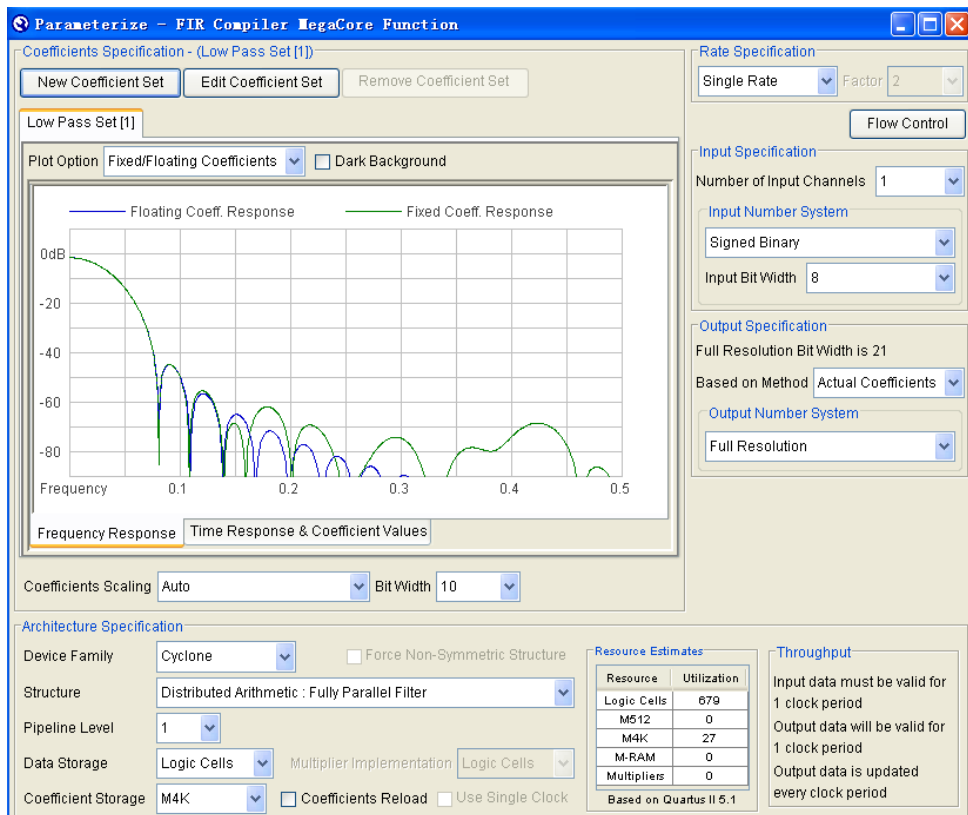


图 10-29 确定 FIR 滤波器系数

除了将设计指标的输入外, 还有一些选项可以设置。点击上方的“Edit Coefficient Set”按钮, 出现如图 10-30。在此, 可以作一些设置, 比如 FIR 滤波器设计时采用的窗口类型, 在这里设为 Hanning 窗。设计者不用关心这些窗口类型在 FIR 滤波器设计的具体实现何种算法, 只要观察不同的窗 (Window) 对 FIR 滤波器性能的影响的仿真情况即可, 然后选定其中一种最符合设计性能要求的窗口类型。



点击“Apply”按钮后，在图中的左侧就显示了计算出的滤波器系数。

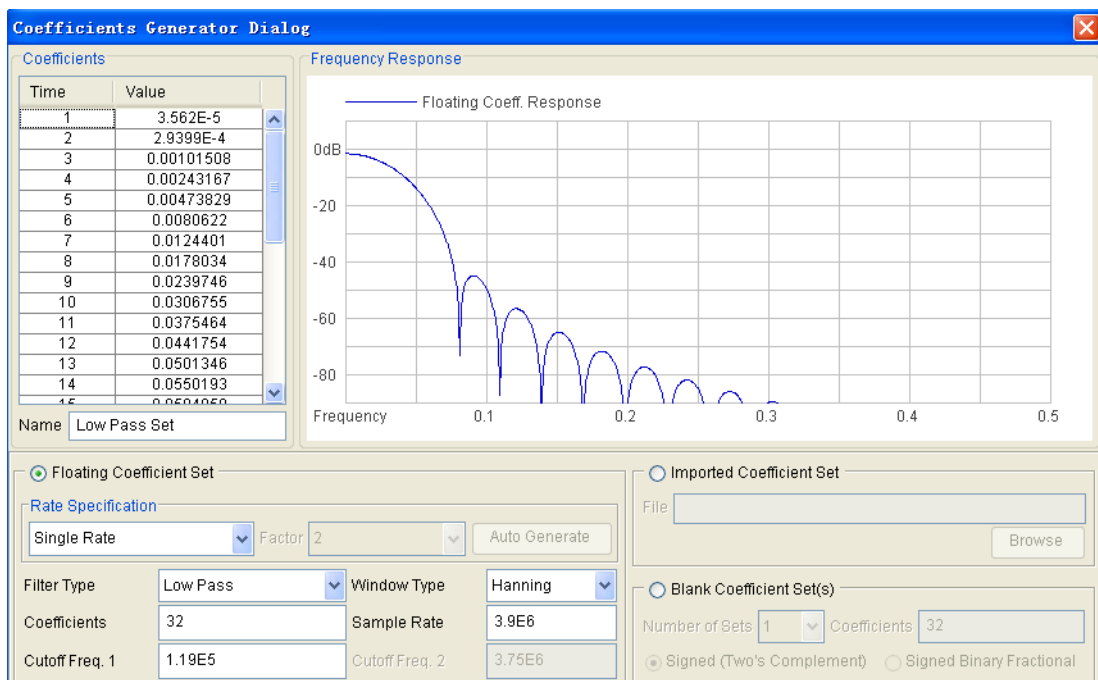


图 10-30 确定 FIR 工作方式

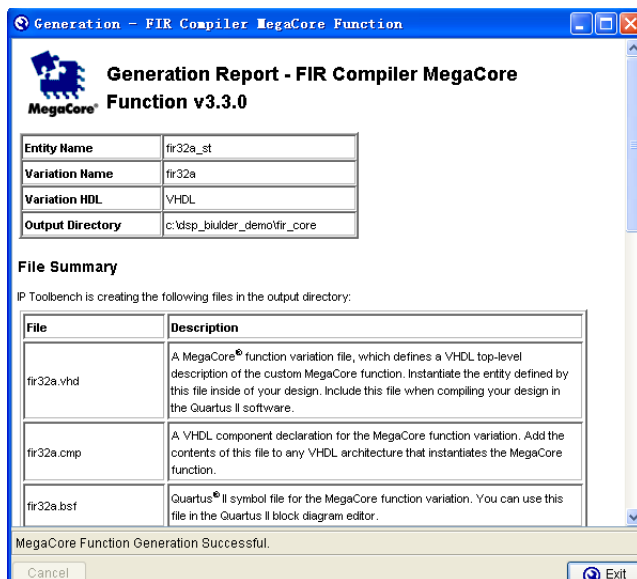


图 10-31 FIR 设定信息窗

关闭窗口后，即完成了 FIR 核的参数设置。这时会弹出一个信息窗（图 10-31），从中可以了解有关此 FIR 核设定后的相关情况。

由图 10-29 可见，滤波器系数精度为 10 位；器件为 Cyclone；结构为并行滤波器结构；

选择了 1 级流水线；滤波器由 LC 逻辑宏单元构成；系数数据存于 FPGA 的 M4K 模块中；1 个输入通道；8 位有符号并行数据输入；全精度数据输出。下方列表列出资源占用情况。

由图 10-30 可见，选择了 Low Pass Set 低通滤波器工作模式；Hanning 窗；采样率 3.9E6；截止频率 1.19E5 等。

为了测试此设定好的 FIR 模块，特建立如图 10-32 所示的电路模型。此模型的左边是以下不同性能特性的信号发生器，以供 FIR 核的测试。右边电路是将输出截取高了 10 位，并转化成了无符号输出，以便能在实验系统上的 Cyclone FPGA 上进行实测。

注意，此模型也能利用 HIL 模块来进行硬件仿真测试。此工作留给读者完成。

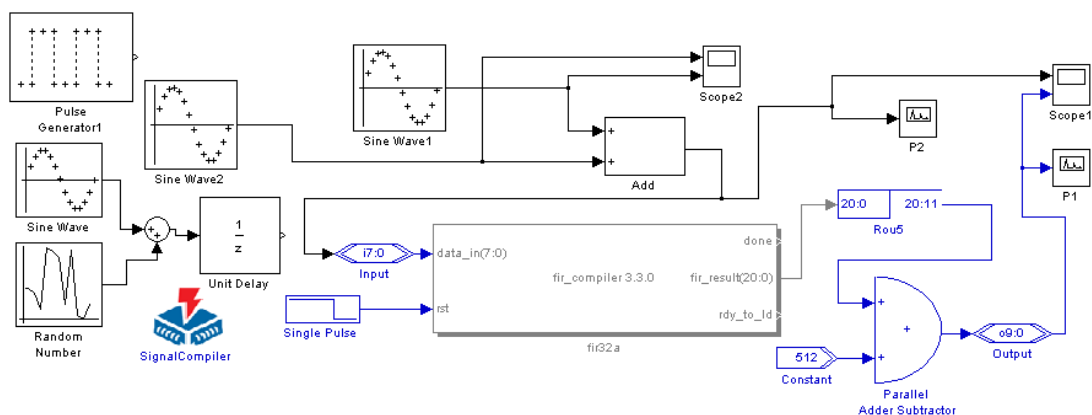


图 10-32 FIR 滤波器核的测试电路模型

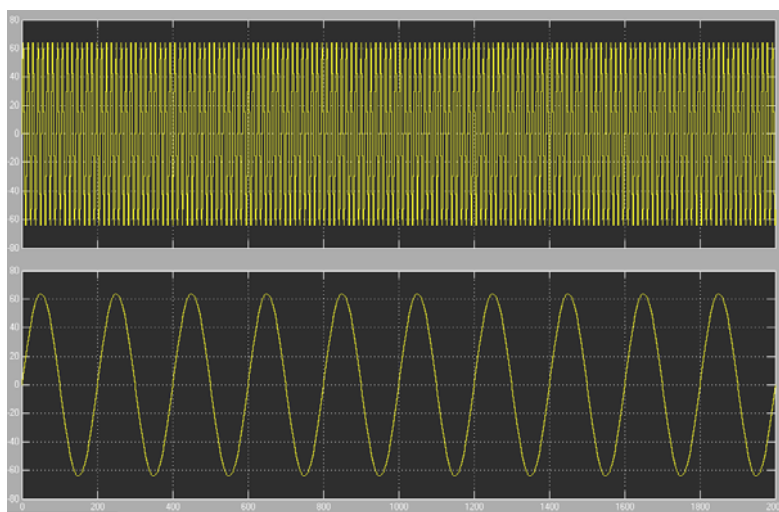


图 10-33 Scope2 显示波形

图 10-32 对 FIR 核选择的输入信号来自两个正弦信号的叠加信号，Scope2 的显示波形如图 10-33 所示。图上方的信号可以认为是干扰波，下方是传输信号波。他们叠加以后的波形如图 10-34 所示。图 10-32 中的 Scope1 显示的波形如图 10-34 所示。上方是输入 FIR 模块的信号，下方是 FIR 模块输出的信号。显然具有良好的滤波作用。

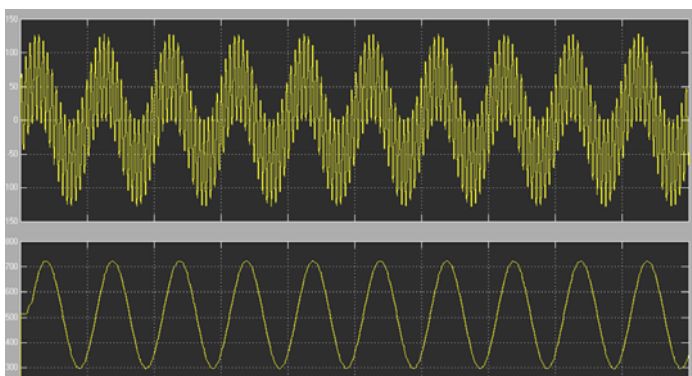


图 10-34 Scope1 显示波形

图 10-32 中的 P2 频谱仪显示的波形如图 10-35 所示。上方的波形是 FIR 核的输入信号；下方的波形是输入信号的频谱。图 10-32 中的 P1 频谱仪显示的波形如图 10-36 所示。上方波形是 FIR 核的输出信号；下方波形是输入信号的频谱。显然，高频率的信号已被滤除。

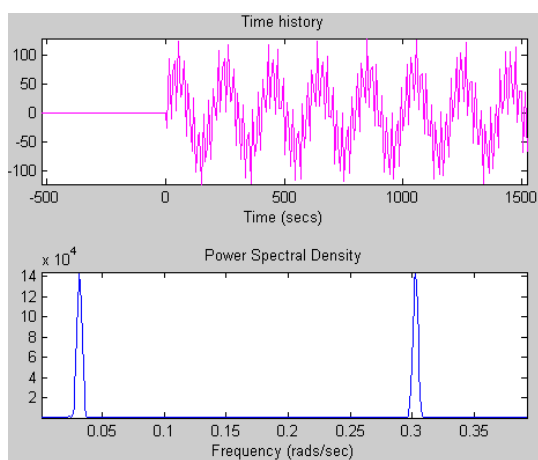


图 10-35 P2 频谱仪显示波形

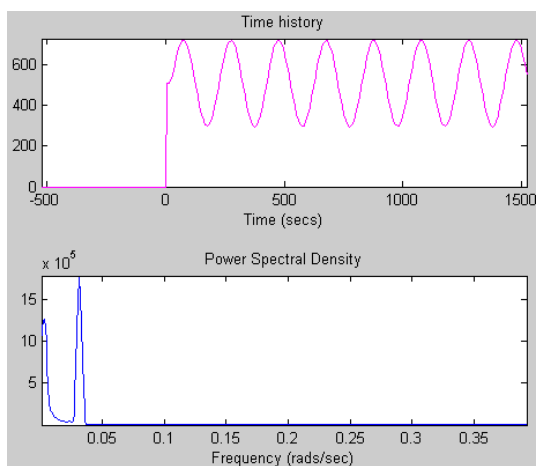


图 10-36 P1 频谱仪显示波形

## 10.2 VHDL 模块插入仿真与设计

在 Simulink 平台上可以利用 Altera DSP Builder 库的 HDL Import 模块将 HDL 文本设计变成 DSPBuilder 设计模块，参与这个模型的软件仿真、硬件仿真、VHDL 转换和硬件实现。本节通过一个实例说明 HDL 插入模块设计方法：

### 1. 完成 VHDL 设计

此例是一个 FIR 滤波器。其 VHDL 描述如例 10-1、10-2 和例 10-3 所示，其中例 10-1

是顶层设计，实体名：`fir_vhdl`。注意，这里的 3 个程序仅给出了实体。完整的设计文件可通过以下路径找到相关示例：

Altera\DSPBuilder\DesignExamples\Tutorials\BlackBox\HDLImport

**【例 10-1】**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
Entity fir_vhdl is
  Port(clock: in std_logic;
        sclr : in std_logic:='0';
        data_in : in std_logic_vector(15 downto 0);
        data_out : out std_logic_vector(32 downto 0));
end fir_vhdl;
```

**【例 10-2】**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;
USE lpm.lpm_components.all;
ENTITY final_add IS
  PORT ( data, datab : IN STD_LOGIC_VECTOR ( 32 DOWNT0 0 );
        Clock, aclr : IN STD_LOGIC ;
        Result : OUT STD_LOGIC_VECTOR ( 32 DOWNT0 0 ) );
END final_add;
```

**【例 10-3】**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
ENTITY four_mult_add IS
  PORT( clock0 : IN STD_LOGIC := '1';
        dataa_0 : IN STD_LOGIC_VECTOR ( 15 DOWNT0 0 ) := ( OTHERS => '0' );
        aclr3 : IN STD_LOGIC := '0';
        datab_0 : IN STD_LOGIC_VECTOR ( 13 DOWNT0 0 ) := ( OTHERS => '0' );
        datab_1 : IN STD_LOGIC_VECTOR ( 13 DOWNT0 0 ) := ( OTHERS => '0' );
        datab_2 : IN STD_LOGIC_VECTOR ( 13 DOWNT0 0 ) := ( OTHERS => '0' );
        datab_3 : IN STD_LOGIC_VECTOR ( 13 DOWNT0 0 ) := ( OTHERS => '0' );
        shiftouta : OUT STD_LOGIC_VECTOR ( 15 DOWNT0 0 );
        result : OUT STD_LOGIC_VECTOR ( 31 DOWNT0 0 ) );
END four_mult_add;
```

## 2. 调入 HDL Import 模块

选择如图 9-81 所示的窗口,将 HDL Import 模块拖入 Simulink 模型编辑窗,例如图 10-37 所示的电路中。

## 3. 加入 VHDL 设计文件

双击图 10-37 所示的 HDL 模块,即弹出图 10-39 所示的对话框。

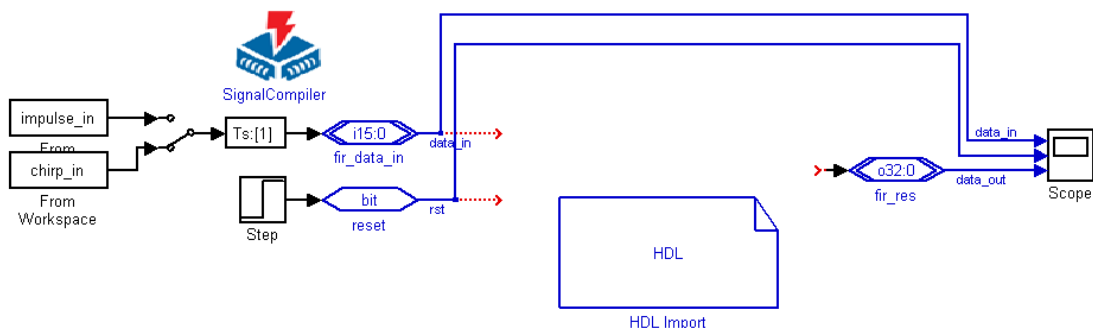


图 10-37 在一个 Simulink 空模型中调入一个 HDLImport 模块

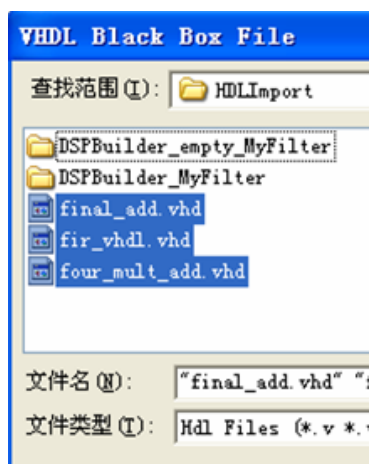


图 10-38 浏览到 3 个 VHDL 文件

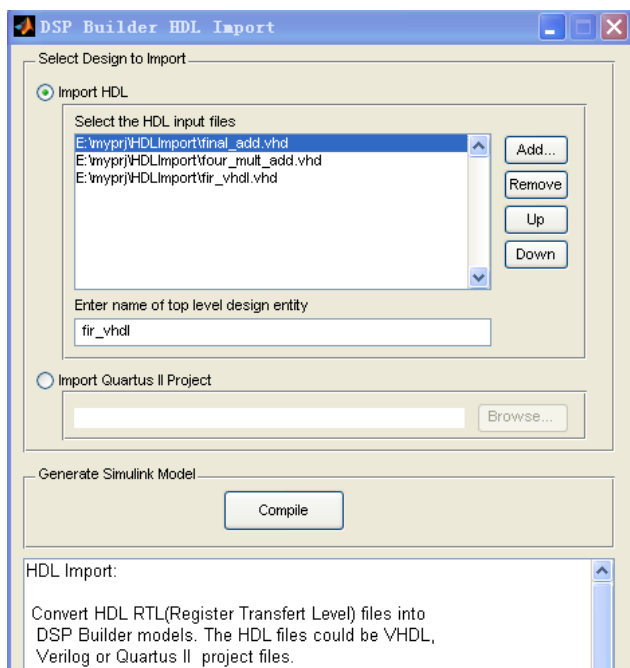


图 10-39 加入 3 个 FIR 设计文件

在图 10-39 所示的窗口中点击 ADD 钮,在弹出的窗(图 10-38)中选中已设计好的 3 个 VHDL 文件,将他们加入到图 10-39 所示的栏中。然后在下一栏中键入顶层设计的实体名: fir\_vhdl。

最后按下 Compile 按钮，进行编译。完成后关闭窗口。

#### 4. 仿真

HDL Import 模块生成后，按图 10-40 连接好进行仿真。图 10-40 中的 fir\_vhdl 的输入信号是扫频信号源。仿真波形如图 10-41 所示。上面的波形是输入的扫频率信号，下面的波形是滤波后的输出信号，波形幅度随输入信号的频率的提高而降低，滤波性能明显。

此后就可以利用 SignalCompiler 将其进行转换、综合和适配。

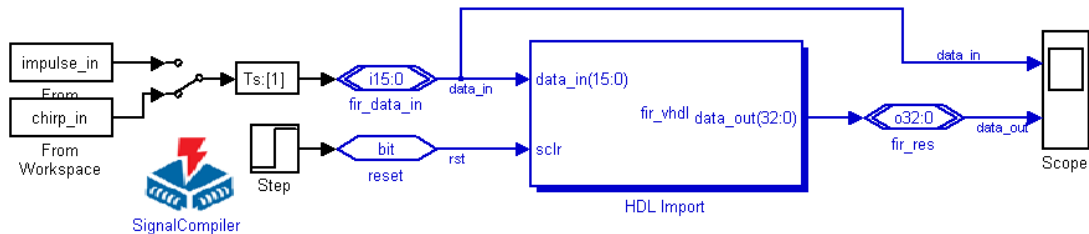


图 10-40 构成一个完整设计

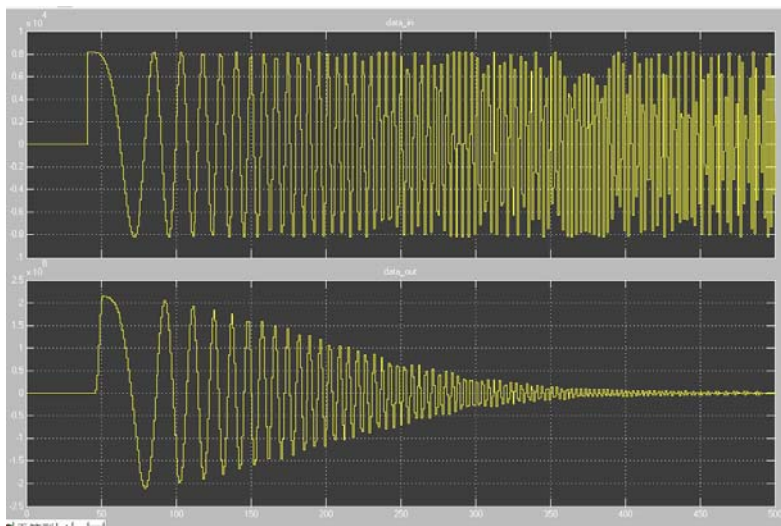


图 10-41 图 10-40 模型的仿真波形

## 10.3 正交幅度调制与解调模型设计

正交幅度调制电路框图如图 10-42 所示，图中， $a(t)$ 和  $b(t)$ 为两路相互独立的待传送的调制信号（通常为基带信号）。载频信号源由基于 DDS 的正交信号发生器产生，它输出两路互为正交的正弦信号。经过两个乘法器可以获得互为正交的平衡调幅波，即不带载频的双边带调幅波；其中一路为同相信号  $I(t)$ ，另一路为正交信号  $Q(t)$ ，二者的表达式分别为：

$$I(t) = a(t) \cos \omega_0 t \quad ; \quad Q(t) = b(t) \sin \omega_0 t$$

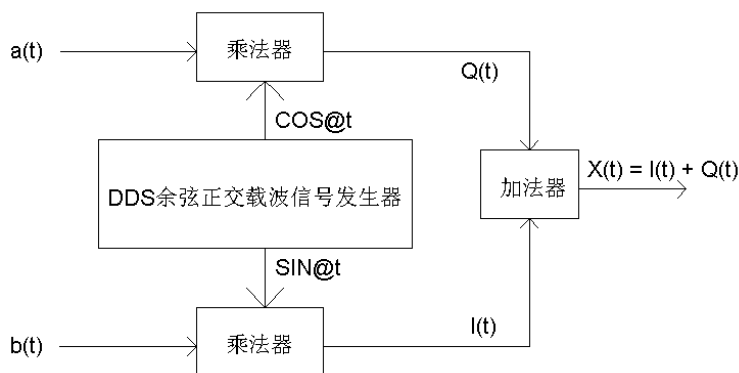


图 10-42 正交幅度调制原理图

这里假设乘法器的乘法系数为 1，此两路信号经过加法器后产生了互为正交的调幅信号，表达式为（设加法器的系数为 1）：

$$X(t) = I(t) + Q(t) = a(t) \cos \omega_0 t + b(t) \sin \omega_0 t$$

式中， $X(t)$  是  $I(t)$ 、 $Q(t)$  两信号相加而得，所以， $X(t)$  的频带宽度等于  $I(t)$  或  $Q(t)$  信号中带宽最宽者（或等于  $a(t)$  或  $b(t)$  二者最宽带宽的 2 倍），而不是二者之和。如此可压缩已调信号的带宽，增加信道容量。如果要对调制信号  $X(t)$  进行解调，不失真地解出原调制信号  $a(t)$  和  $b(t)$ ，就需要用正交同步解调的方法，这一方法的原理框图如图 10-43 所示。

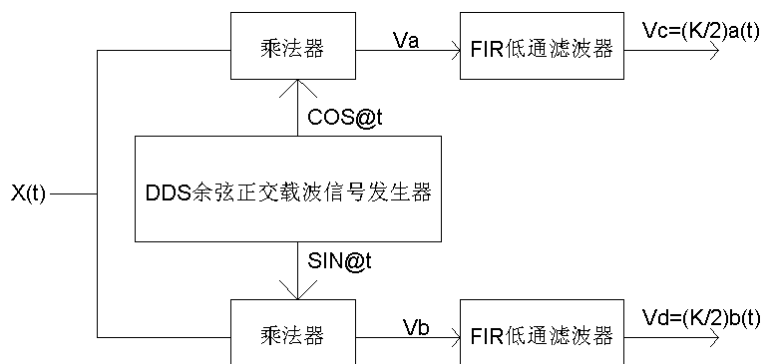


图 10-43 正交幅度信号解调原理图

已调正交调幅信号  $X(t)$  分别与正交信号发生器产生的互为正交的余弦信号和正弦信号相乘后产生两路输出信号，它们分别为：

$$v_A = X(t) \cos \omega_0 t = a(t) \cos^2 \omega_0 t + b(t) \sin \omega_0 t \cos \omega_0 t$$

$$= \frac{1}{2}a(t) + \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}b(t)\sin 2\omega_0 t$$

$$v_B = X(t)\sin \omega_0 t = b(t)\sin^2 \omega_0 t + a(t)\sin \omega_0 t \cos \omega_0 t$$

$$= \frac{1}{2}b(t) - \frac{1}{2}a(t)\cos 2\omega_0 t + \frac{1}{2}a(t)\sin 2\omega_0 t$$

经过 FIR 低通滤波器滤波后，上式的低频信号 $(1/2)a(t)$ 和 $(1/2)b(t)$ 即被取出，故得解调输出为（设低通得传输系数为  $K$ ）：

$$v_C = \frac{K}{2}a(t)$$

$$v_D = \frac{K}{2}b(t)$$

由于基于 DDS 的正交信号发生器的两个正交信号  $\cos \omega_0 t$  和  $\sin \omega_0 t$  是严格正交的，且与  $I(t)$ 、 $Q(t)$  信号的载波保持了良好的同步，因此能不失真地解调出  $a(t)$  和  $b(t)$  信号来。

图 10-44 所示的是正交幅度调制和解调 Simulink 电路模型。

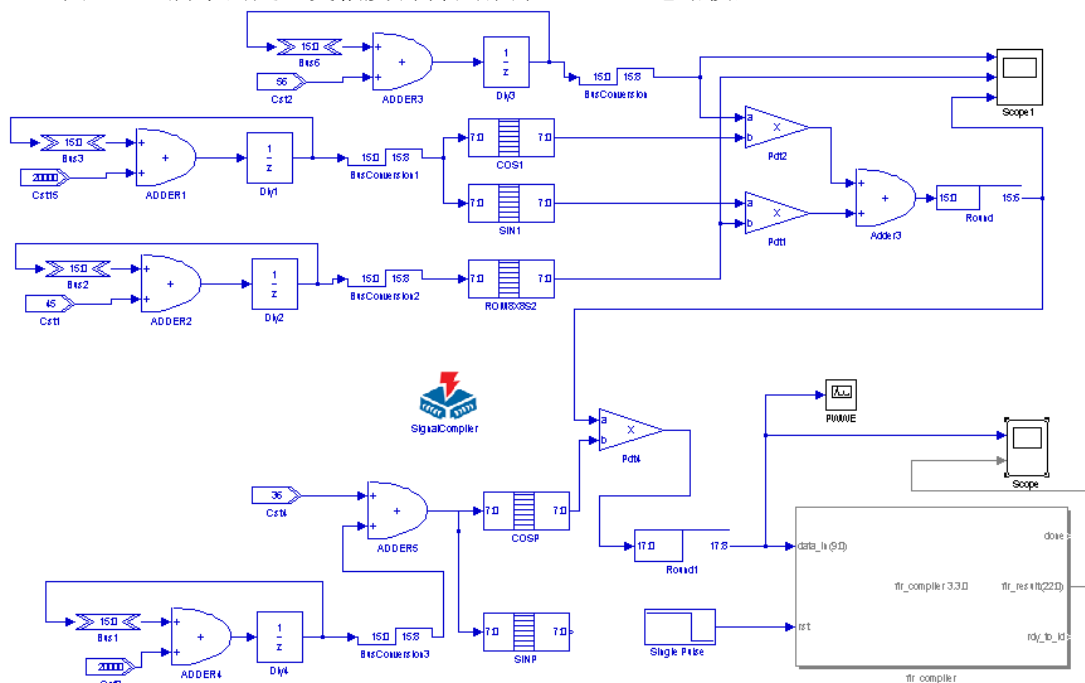


图 10-44 FIR 滤波器核的测试电路模型

图 10-44 的上部电路是调制电路，下部是解调电路，使用的滤波器是 FIR 核，不过只用了单路滤波。图 10-45 是 Scope1 显示的波形。其中上栏和中栏是两种不同波形的调制信



号，锯齿波和正弦波。下栏是混合了载波的输出波形。

图 10-46 是图 10-44 电路中频谱仪的输出波形。其上部波形与图 10-47 的上部波形相同，是正交相乘后进入 FIR 模块前的波形。图 10-46 下部是此波的频谱；图 10-47 的下部波形是 FIR 滤波后的输出波形，即解调出来的信号。如果对另一路滤波，则能解调出锯齿波。

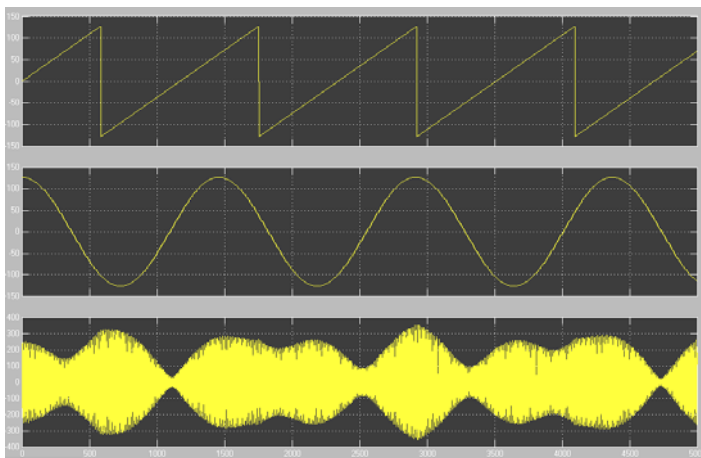


图 10-45 FIR 滤波器核的测试电路模型

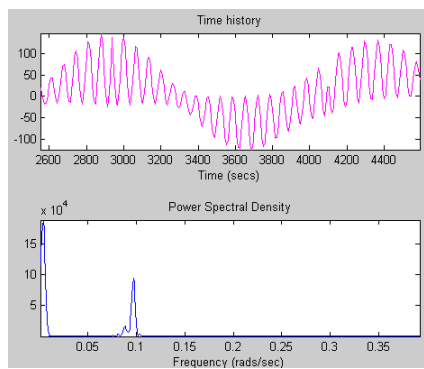


图 10-46 FIR 滤波器核的测试电路模型

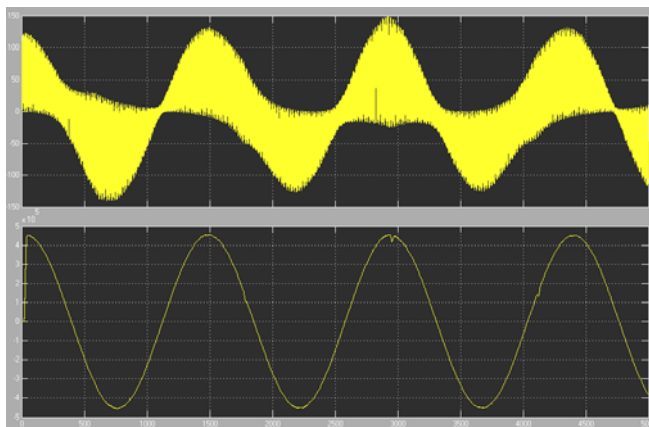


图 10-47 FIR 滤波器核的测试电路模型

## 10.4 NCO IP 核应用

与以上使用 FIR 核的方法相同，也能在 Simulink 平台上使用数控振荡器 NCO 核。

图 10-48 是一个 NCO 核的使用示例。NCO 的参数设置方法可以参考第 4 章。

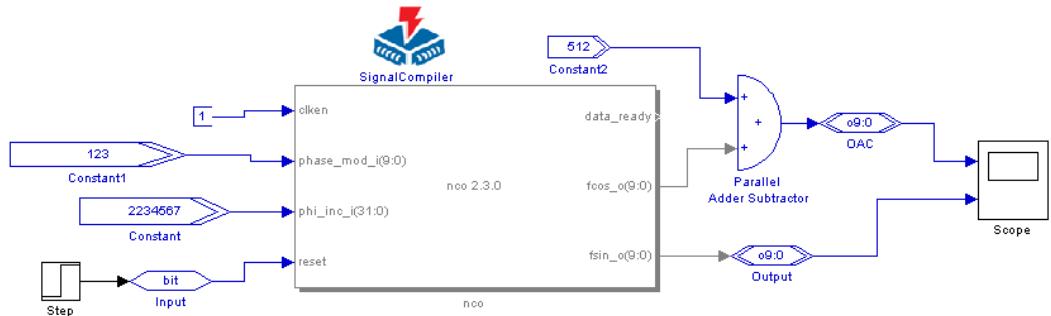


图 10-48 FIR 滤波器核的测试电路模型

## 10.5 基于 IP 的数字编译码器设计

在第 9 章中已经讨论了数字编译码器的设计。如果将编解码模块与前面介绍的滤波器、FFT 变换器、数控振荡器 NCO 等 IP 核相结合，可以在同一块 FPGA 上完成一个数字信号处理系统的几乎所有功能模块，且规模大，性能优越，在许多方面将大大优于 DSP 处理器构成的系统。本节将介绍两例基于 IP 核的编解码模型的设计。

### 10.5.1 RS 码

在实际的数字通信传输信道上，信号发生错误是不可避免的。可以采用信道编码来尽可能地降低误码率。在信道编码中除了需要传送的信息外，还加入了一些冗余信息，以便在接收端检出错误。对于检出错误的处理方式常用的有三种：检错重发（ARQ）、前向纠错（FEC）、混合纠错（HEC）。对于前向纠错，是不需要反馈信道的，在信道编码中含有纠错信息，实时性较强。RS 编码在前向纠错中使用比较常见。

RS 码是 Reed Solomon 码的简称，是属于循环码 BCH 码的一种，对于突发错误，RS 码具有很好的纠错能力。

一个  $(n, k)$  RS 码，输入信号分成  $k \cdot m$  比特一组，每组包括  $k$  个符号，每个符号由  $m$  个比特构成。对于一个可以纠正  $t$  个符号错误的 RS 码参数如下：

参数名称	参数值	单位
码长	$n = 2^m - 1$	符号
信息段	$k$	符号
监督段	$n - k = 2t$	符号
最小码距	$2t - 1$	符号
符号	$m$	比特

对于 RS 码的编码器可以用带反馈的移位寄存器来实现，不过实现比较复杂。Altera 为

RS 码提供了 IP Core，即 RS Compiler，可以大大简化 RS 编码译码器的设计。RS Compiler 除了可在 QuartusII 中使用外，还可与 DSP Builder 配合使用（图 10-49）。

按照图 10-49，新建一个模型，放置一个 Reed Solomon 模块，双击该模块后将出现 RS Compiler 对话框，如图 10-50 所示，选择“Encode”编码器。然后点击“Next”按钮，进行 RS 编码器的参数设置（图 10-51）。设置完成后，设计者就可以在 Simulink 中如同调用其它 DSP Builder 模块一样调用 RS 编码器来完成更大的设计了。

同样方法，使用 RS Compiler 核也可以设计 RS 译码器。

同设计 RS 编码器时一样调用 RS Compiler，选择类型为“DeCode”译码器（图 10-52）。其参数设置与 RS 编码器相同，不再赘述。

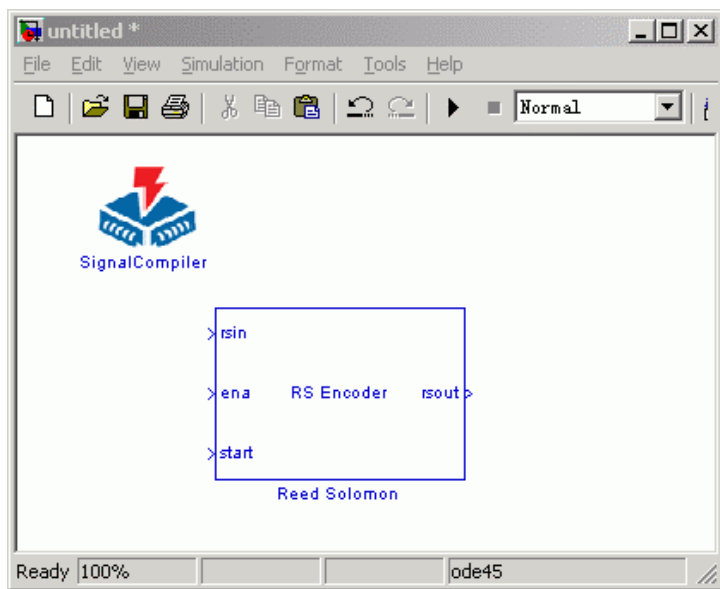


图 10-49 RS Compiler 与 DSP Builder 集成

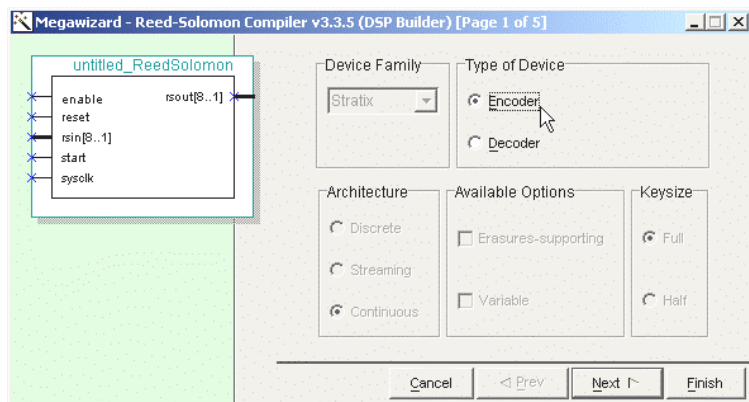


图 10-50 选择类型为 RS 编码器

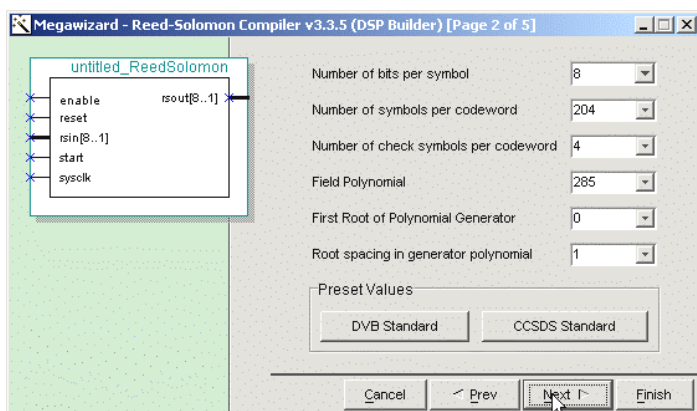


图 10-51 确定参数

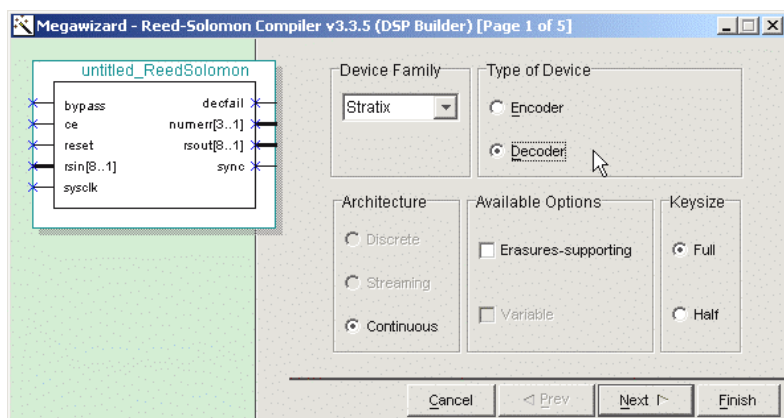


图 10-52 选择类型为 RS 译码器

## 10.5.2 Viterbi 译码

除了上面提到的 RS 编码，卷积码在数字通信上的应用也很多。Viterbi 译码是卷积码的一种较好的译码方式。卷积码与 RS 码不同，卷积码编码后的  $n$  个码元不但与当前段的  $k$  个信息相关，而且与前面  $(N-1)$  段的信息相关，即编码后相互关联的码元为  $Nn$  个。因而，在相同码元个数下，卷积码的纠错能力更强，但译码的复杂性也随之提高。在卷积码的三种译码方式：门限译码、Viterbi 译码、序列译码中，Viterbi 译码的性能最好。Viterbi 译码基于最大似然译码原理，而且在译码时无须反馈操作。

同样可以使用 IP Core 设计 Viterbi 译码器。

为了简化 Viterbi 译码器在 FPGA 上的实现，Altera 提供了 Viterbi 译码器的 IP Core: Viterbi Compiler。可在 DSP Builder 上集成使用。

具体的 Viterbi Compiler 使用方法可参考 Altera 的使用手册。Viterbi 译码器的其他实现

方法比较复杂，具体的 Viterbi 译码原理可参见相关书籍。

## 习 题

10-1 简述使用 Fdatool 设计 FIR 滤波器的主要步骤。

10-2 讨论在 FPGA 上实现 FIR 滤波器时的系数量化问题及结果的截断问题。

10-5 FFT 涉及复数运算，在实际使用过程中输入、输出往往为实数序列。对于实数序列，可以使用与 FFT 类似的变换，即离散余弦变换 (DCT)。DCT 被广泛用于图像、音频压缩等领域。在音频处理中，经常用到 8 点一维 DCT 变换。下式是 MP3 解码算法中的合成子带滤波器组使用的修正的  $DCT_{32 \rightarrow 64}$  (MDCT) 公式：

$$x[i] = \sum_{k=0}^{31} X[k] * \cos \left[ \frac{i+16}{64} \pi (2k+1) \right], i=0..63$$

因为 DCT 变换是周期性的，计算  $DCT_{32 \rightarrow 32}$  就可以得到  $DCT_{32 \rightarrow 64}$  的所有值。

$DCT_{32 \rightarrow 32}$  的公式是：

$$x[i] = \sum_{k=0}^{31} X[k] * \cos \left[ \frac{i * \pi}{64} \pi (2k+1) \right], i=0..31$$

根据快速 DCT 的 Lee 氏算法，将 32 点 DCT 进行分解，以 8 点 DCT 为基础。8 点 DCT 的 Lee 氏快速 DCT 算法结构如图 10-53 所示。

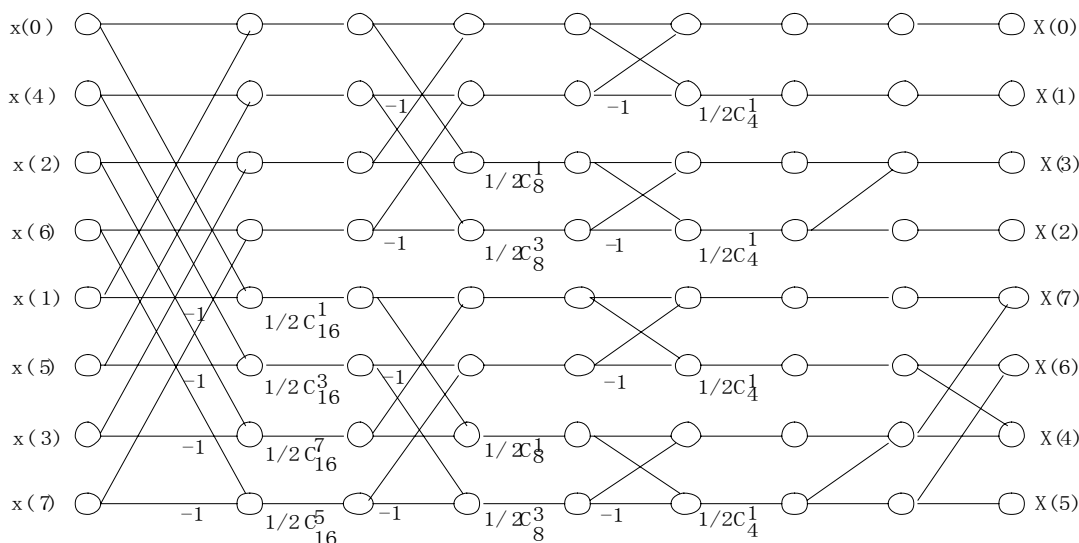


图 10-53 Lee's 快速 DCT 算法

$$\text{其中 } C_k^i = \cos\left(\pi \frac{i}{k}\right)$$

试用 Simulink/DSP Builder 建立该 DCT 模型。

## 实验与设计

### 实验 10-1 FIR 数字滤波器设计实验

(1) 实验目的: 学习并掌握利用 Matlab/Simulink、DSP Builder 和 QuartusII 设计不同类型的 FIR 滤波器, 包括建模、参数计算、系统仿真、综合、时序仿真、硬件实现与测试。

(2) 实验任务 1: 根据 10.1.2, 完成一个 3 阶常数系数 FIR 滤波器设计。

(3) 实验任务 2: 设计一个 5 阶常数系数 FIR 滤波器。已知其系统函数为:

$$h(n) = C_q (h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3) + h(4)x(n-4) + h(5)x(n-5))$$

$$\text{其中: } h(0) = 25, h(1) = 93, h(2) = 212, h(3) = 212, h(4) = 93, h(5) = 25$$

$$C_q = 0.04$$

试参照 10.1.2 建立一个模型, 并给出 Simulink 仿真结果。

(4) 实验任务 3: 按照 10.1 节, 完成 16 阶直接 I 型滤波器模型设计。

(5) 实验任务 4: 设计一个 64 阶的直接 I 型滤波器模型, 设计参数如下:

1、高通滤波器; 2、采样频率  $F_s$  为 48kHz, 滤波器  $F_c$  为 10.8kHz; 3、输入序列位宽为 9 位 (最高位为符号位)

(6) 实验任务 5: 在一般应用中, 需要设计的 FIR 滤波器往往是线性相位的, 其滤波器系数是对称的, 可以通过优化滤波器结构来减少 FIR 滤波器实现的运算量。比如对于实验任务 2, 就是一个线性相位的 FIR 滤波器, 其中:

$$h(0) = h(5) = 25, h(1) = h(4) = 93, h(2) = h(3) = 212$$

那么就有:

$$h(n) = C_q [h(0)(x(n) + x(n-5)) + h(1)(x(n-1) + x(n-4)) + h(2)(x(n-2) + x(n-3))]$$

试按照上式重新构建线性相位 FIR 滤波器模型。

(7) 实验任务 6: 在 DSP Builder 中也可以使用 Shift Taps 模块和 Multiply Add 模块来设计 FIR 滤波器, 例如图 6-67 中构成的滤波器。

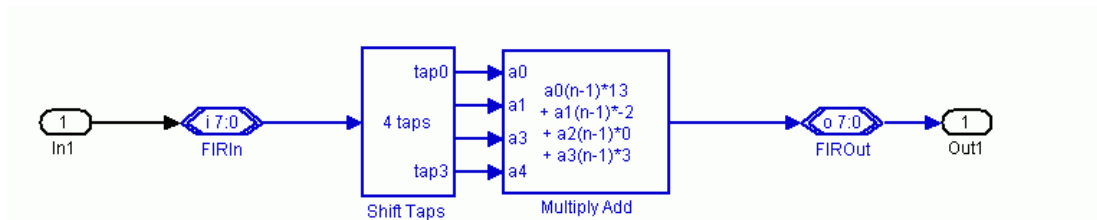


图 10-54 Shift Taps 模块和 Multiply Add 模块应用

试用这两种模块从新设计实验任务 2 的 5 阶常数系数 FIR 滤波器。

(8) 实验任务 7: 参照 10.1.4 节, 使用 FIR Compiler 和 IP Core 设计一个 32 阶 FIR 低通滤波器。

(9) 实验任务 8: 利用 DSP Builder 安装路径: \DSPBuilder\designexamples\Fir32\中给出的 32 阶固定系数 FIR 滤波器文件 AltrFir32.mdl, 利用各种信号源进行仿真测试, 并对测试结果进行分析。之后在对输入输出口的数据类型转变后, 完成硬件实现。信号采样可以使用 20MHz 的 5510, 信号输出使用 100MHz 速率的 5651 完成。最后进行硬件测试, 并将实测结果与 Simulink 的仿真结果进行比较。

(10) 实验报告: 根据以上要求和实验内容, 记录并分析所有实验结果, 完成实验报告。

## 实验 10-2 编译码器与调制解调模块设计实验

(1) 实验任务 1: 参照相关章节, 练习 RS 码的 IP Core 的使用。

(2) 实验任务 2: 练习 Viterbi 译码的 IP Core 的使用。

(3) 实验任务 3: 设计一个 QPSK 的调制模型, 输入信号经过 RS 编码, 再设计一个 QPSK 模型的解调模型。

(4) 实验任务 4: 设计正交幅度调制解调电路, FIR 滤波器可以是一个 16 阶的滤波器, 也可以用 FIR IP 核来担任。最后用 Cyclone FPGA 实现。

## 实验 10-3 HDL Import 模块应用实验

实验任务: 用 VHDL 设计一个 32 位乘 32 位复数乘法器, 再于 Simulink 平台上用 HDL Import 模块实现仿真。