|  |  |  |
| --- | --- | --- |
| 题目： | 我规划的一堂数字信号实验课  ——基于国产FPGA的IP核的简单FIR滤波器 |  |

# 第一章 绪 论

## 1.1 本课题研究意义

随着工艺和EDA技术的进步，在信号处理方面，将模拟信号转为数字信号，将信号去从数字信号的角度进行理解与处理已经成为了主流方式，但模拟转数字的方式所需要处理的数字信号是海量的，对速度、资源利用率、功耗等性能的要求也不断拔高，这些需求引导着各种高性能数字计算电路发展的飞速发展。 其中，数字滤波器是处理数字信号最重要的工具之一，其常被分为无限脉冲响应滤波器（Infinite Impulse Response Filter，IIR 滤波器）和有限长单位冲激响应滤波器（Finite Impulse Response Filter，FIR 滤波器）[1]。其中，和 IIR 滤波器相比，FIR 滤波器能保证较严格的线性相位特性[2]，具有稳定性强、硬件易实现等优势，因此在被更广泛的应用[3]。

现场可编程门阵列（Field Programmable Gate Array，FPGA）常于高速并行数据处理与数据传输，也能大大缩短数字产品开发周期，因此得到了广泛的应用。FPGA的IP核是对应FPGA芯片开发公司发布于其FPGA开发平台上的预先编写验证的硬件设计模块，具有可复用性、可配置性、标准化与可靠性，通过复用IP核能大大加快一个数字产品的开发周期。

基于FPGA 的 FIR 滤波器是 FPGA 数字信号处理领域中的较热门的研究方向之一[4]，然而现有的基于FPGA的FIR滤波器大多是基于赛灵思或者因特尔的FPGA芯片，使用的IP核也是这些外国公司的专利，本课题期望通过使用国产FPGA，并使用其开发的国产IP核快速设计一FIR 低通滤波器，为国产FPGA设计FIR滤波器提供一个的例子。

## 1.2 国内外研究现状分析

1985年，Xilinx生产出了第一块FPGA元器件，由于它有着集成度高、方便易用、开发和上市周期短的绝对优势，使得FPGA器件在数字设计和电子生产中得到迅速普及和应用，发展潜力十分巨大。现在FPGA已经发展到可以利用硬件乘加器、片内储存器、逻辑单元、流水处理技术等特有的硬件结构，高速完成FFT 、FIR 、复数乘加、卷积、三角函数以及矩阵运算等数字信号处理。这样可以完成信号处理的主要技术，如中频采样、参数估计、自适应滤波、脉冲压缩、自适应波束形成和旁瓣对消等。

近几年，国产FPGA也有了不错的发展，逐渐跟上国外厂商的步伐，有了自己的EDA开发平台，目前比较知名的国产FPGA有安陆、高云、紫光同创等公司。

## 1.3 研究思路

通过对目前FIR数字滤波器的几种实现方法的简单分析，选定认为基于三角形窗的FIR滤波器，将FIR滤波器的抽头系数通过matlab导出并进行一定脚本代码的处理后导入FPGA的FIR滤波器IP进行初始化。使用其他IP核生成信号，使用FIR滤波器进行滤波，使用modelsim平台进行仿真验证。

# 第二章 FIR数字滤波器的设计方法

## 2.1 理论部分

### 2.1.1  引言

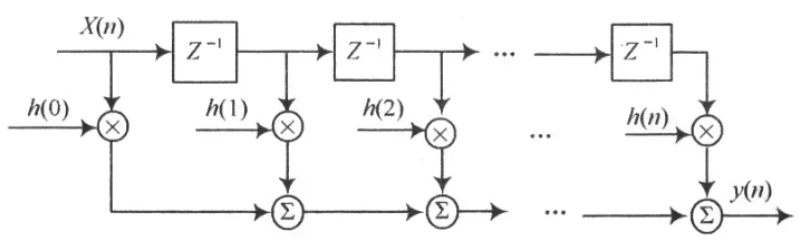
数字信号的滤波运算实际上是一种加权运算，输入信号与不同权重的系数进行乘 加运算后能够保留目标信号，衰减干扰信号。FIR 滤波器还具有线性相位的特点，在有效频率输入范围内，经过系统信号的绝对相位可能会产生变化，但是各个信号间的相对相位不变，此类滤波器的系数具有对称关系。若系数对称，采用一定的设计技巧能 够有效节约硬件资源。

### 2.1.2  FIR数字滤波器的基础

通常情况下一般数字滤波器的N阶FIR数字滤波器基于输入信号x(n)的表达式为：

(2-1)

这个公式给我们了一个非常明了的直接型网络结构，该结构表现出N个乘法器，每次采样y(n)的内容是n次乘法和n-1次加法，然后做乘累加之和。如图2-1所示：



**图2-1 FIR滤波器直接型网络结构**

若h(n)呈现对称特性，即此具有线性相位的滤波器是FIR数字滤波器。滤波器的基础网络结构可以相互进行转换。

(2-2)

(2-3)

(2-4)

公式2-2 ，2-3， 2-4是在窗函数设计法下计算截止频率，根据过度带宽计算滤波器阶数的计算公式。

### 2.1.3  数字滤波器的设计原理

在FIR数字滤波器设计中，一般常使用窗函数法进行设计，本文使用窗函数和Matlab工具箱—Filter Designer Tools共同进行设计。

FIR数字滤波器的一般性设计步骤如下：

1.给出滤波器指标：给出截止频率，理想幅频响应，相位响应，阶数

2.确定滤波器实际参数：实际阶数，实际响应

3.验证设计结果并迭代

### 2.1.4  FIR数字滤波器的理论计算与参数转换

用窗函数法设计FIR带通滤波器。指标如下：

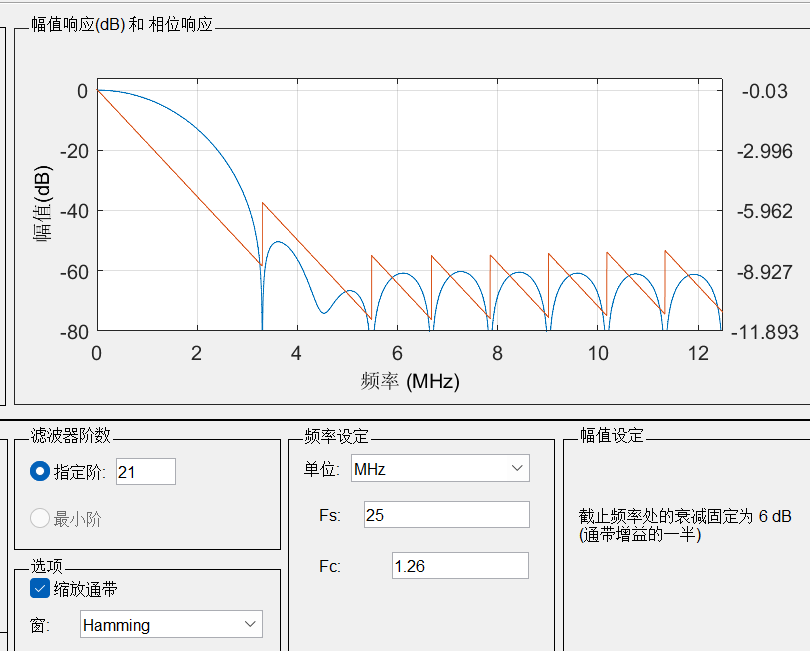
**表2.1：滤波器设计指标**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **抽样频率** | **通带截止频率** | **阻带起始频率** | **阻带衰减** | **通带衰减** | **选用窗** |
| **25MHz** | **3MHz** | **7Mz** | **>50dB** | **<3dB** | **海明窗（Hamming）** |

根据表2.1的设计指标，代入公式2-2 ，2-3 ，2-4 ，可计算得到 截止频率为1.26MHz，过度带宽为 0.16Π ，阶数N=21 。

### 2.1.5 滤波器系数实际设计结果：

1）使用Filter Designer Tools工具箱设计按照理想参数设计所得如图2-2



**图2-2海明窗FIR低通滤波器幅/相频响应**

2）导出滤波器系数后使用matlab脚本函数生成滤波器系数文件，用来初始化FPGA的FIR滤波器IP，为了符合FIR滤波器IP的初始化文件规范，对系数进行一定程度修改，得到22个滤波器系数如下表2.2：

**表2.2 滤波器抽头系数**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 系数位序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 系数值 | 5 | 5 | 30 | 86 | 185 | 332 | 517 | 721 | 914 | 1066 | 1149 | 1149 | 1066 | 914 | 721 | 517 | 332 | 185 | 86 | 30 | 5 | 5 |

## 2.2基于国产高云FPGA的FIR滤波器电路设计

### 2.2.1 国产高云FPGA 及其开发平台介绍：

广东高云半导体科技股份有限公司成立于2014年，是一家专业从事现场可编程逻辑器件（FPGA）研发与设计的国产FPGA高科技公司，致力于向客户提供从芯片、EDA开发软件、IP、开发板到整体系统解决方案的一站式服务。经过多年的积累，高云半导体在FPGA芯片架构、SOC芯片设计、FPGA集成EDA开发环境、FPGA通用解决方案等整个生态链均有核心自主知识，以及国内外发明专利。通过最新工艺的选择和设计优化，高云半导体已经取得与现有市场国际巨头同类产品媲美的高质量、高可靠性FPGA产品，并已经在汽车、工业控制、电力、通信、医疗、数据中心等应用领域实现规模量产。

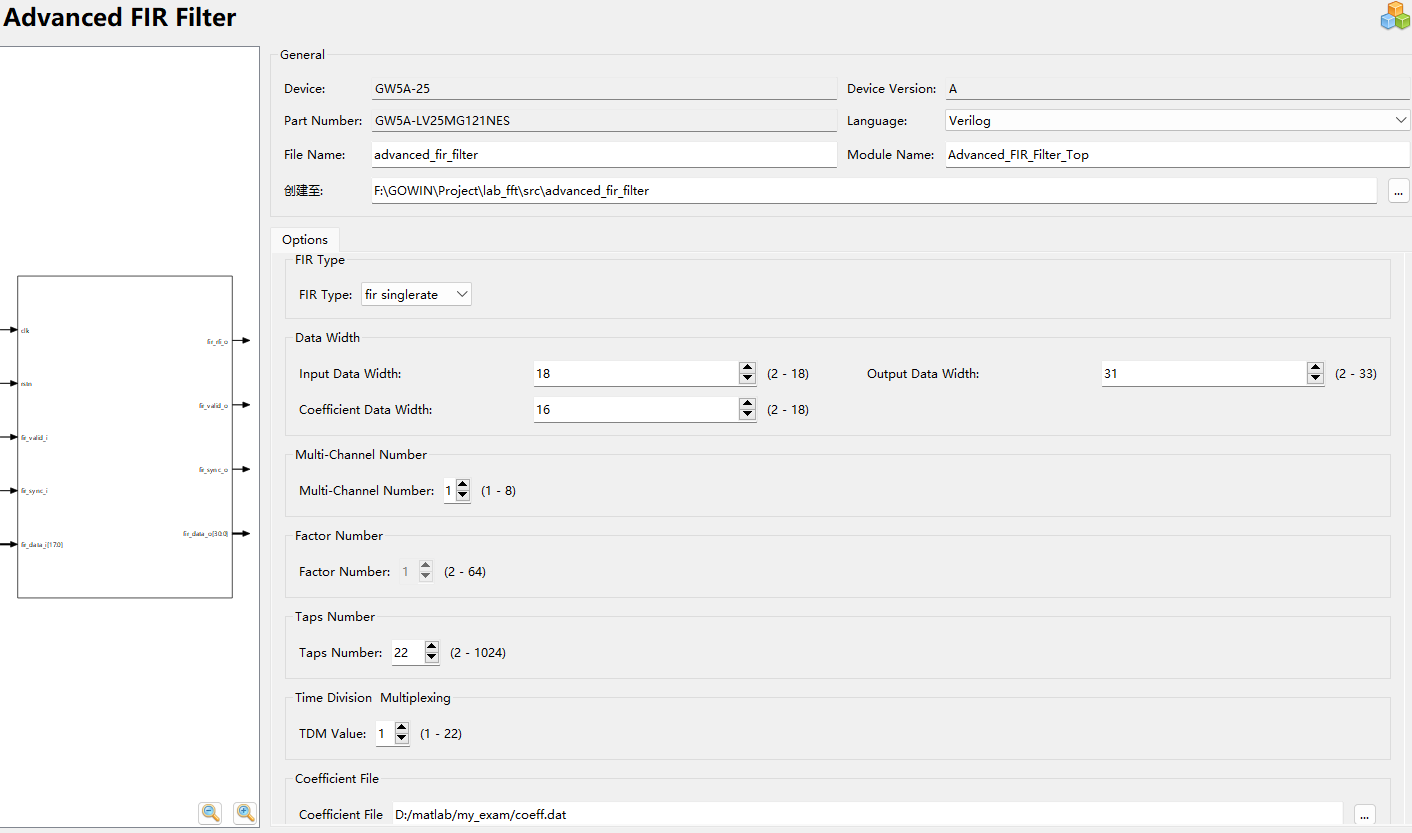
### 2.2.2 所使用的高云FPGA的IP核的介绍与配置：

高云FPGA的IP核体系较为完善，涵盖了通讯、图像处理、工业控制、汽车电子、消费电子、人工智能、数据中心，本次设计FIR滤波器使用到的高云FPGA IP核有5个，分别是PLL锁相环时钟IP，Multiplier 乘法器IP核 ，DDS 和DDS II 数字频率合成器IP核， Advanced FIR filter滤波器IP核。

1. 配置Advanced FIR filter滤波器IP核：

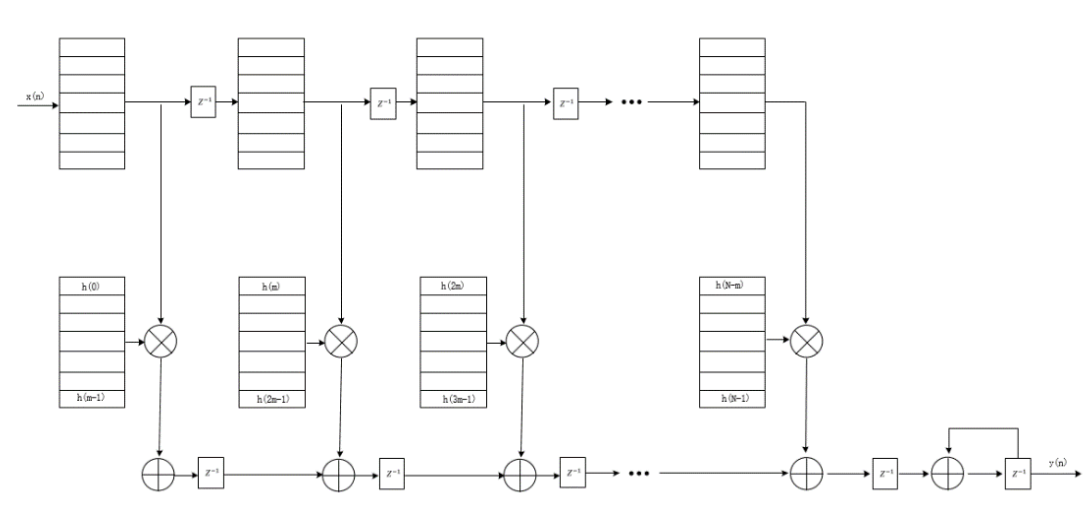
Gowin Advanced FIR Filter IP 实现有限长单位冲激响应（FIR，Finite Impulse Response）滤波器功能。根据 FIR 滤波器的不同，支持 4 种模式配置，同时支持多通道处理、时分复用处理等功能配置。

FIR配置如图2-3，配置为FIR Singlerate结构，输入18bit，输出31bit，抽头系数22个系数



**图2-3 FIR滤波器IP核配置**

FIR Singlerate 模式通过转置型结构实现 FIR 滤波器，输入到输出通过流水线处理，可以很快得到结果，适用于数据和系数位宽较大，但阶数比较小的设计。FIR Singlerate模式的基本结构如下图 2-4 所示。



**图2-4 FIR Singlerate结构基本框图**

通过比较图2-1 FIR直接网格型结构，可以知道FIR Singlerate结构就是其进行了一定优化后的结果。

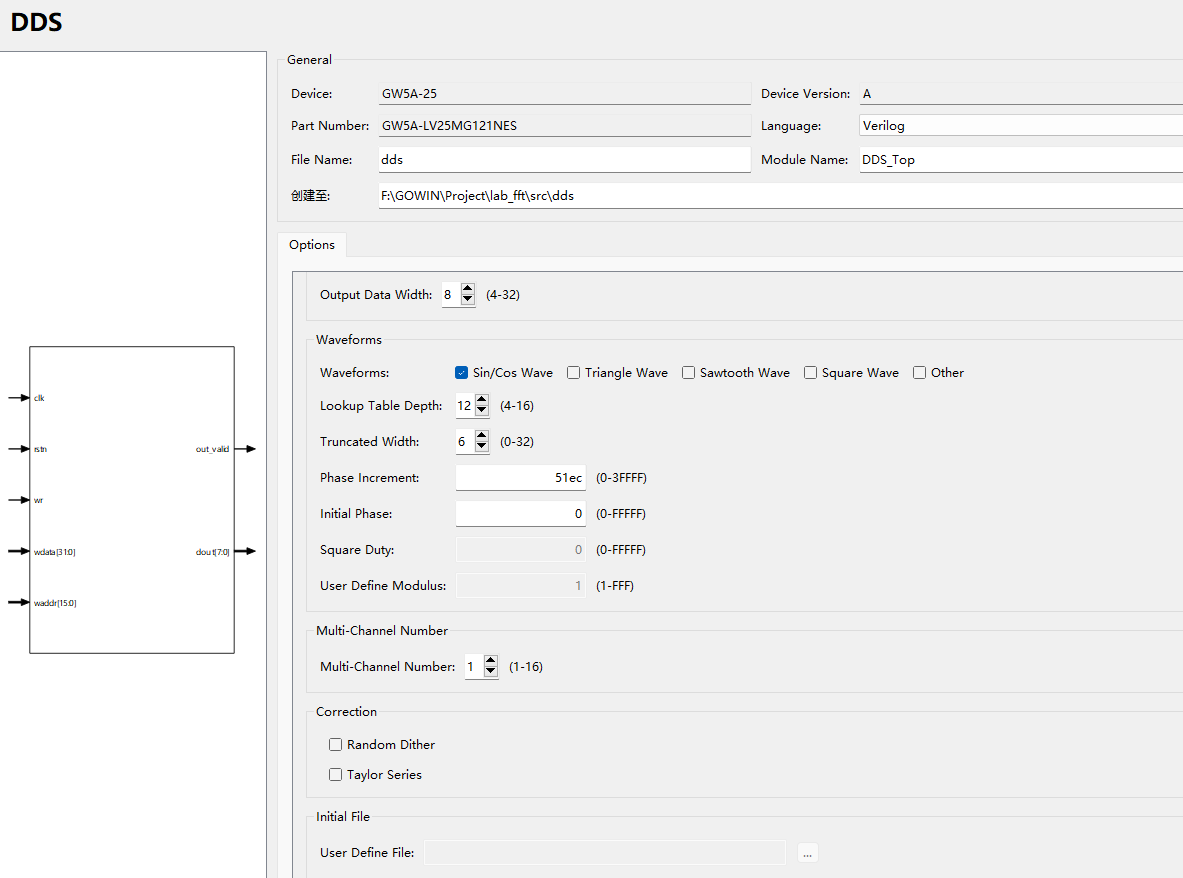
该FIR IP核还可以配置为FIR Symmetry （FIR 对称性）结构 和FIR Polyphase Decimator （FIR 多相抽取器）结构 与 FIR Polyphase Interpolator（FIR 多相插值器）结构，FIR Symmetry结构利用了FIR结构的奇偶对称性，能节省FPGA的资源，提高效率。FIR Polyphase Decimator结构 与 FIR Polyphase Interpolator结构是通过将原滤波器拆分为子滤波器的方法，节省硬件资源，提高数据并行率和吞吐量。

1. 配置DDS IP核（数字频率合成器/数字信号生成器）：

Gowin DDS IP是直接数字式频率合成器，可实现从相位概念出发直接合成所需正弦波形。该IP可配置多个独立的分时复用通道、输出数据位宽、频率及相位偏移量，使用灵活方便。

如图2-5，对IP核DDS进行配置，配置输出8bit，查找表深度12bit，截断位6bit，相位增量PINC为20972（十六进制为51ec），配置后DDS将会直接输出一个频率为4MHz的数字信号，不需要对DDS进行额外的操作。根据高云用户手册，DDS输出正弦波的频率计算公式为：

此处DDS的输入频率为100MHz。

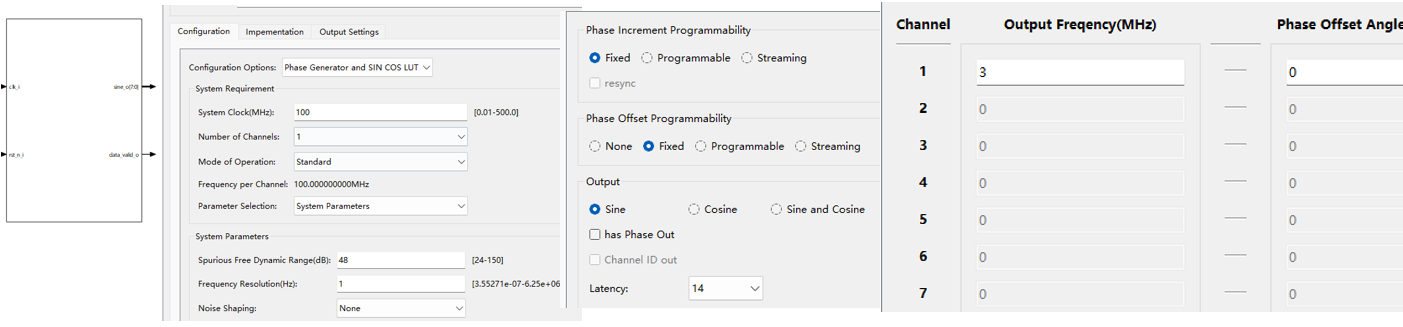


**图2-5 IP核DDS配置界面**

1. 配置IP核DDS II（直接式数字频率合成器/数字信号生成器）：

Gowin DDS II IP实现了直接数字式频率合成器(DDS)的功能。该内核支持正弦、余弦或正交输出功能，输出采样精度为 4 至 25位，也支持相位输出功能。该内核可以单独配置为相位生成器和正弦余弦查找表。该内核支持分时复用多达16个通道。该内核可以通过相位抖动和泰勒级数校正选项来实现使用最少的 FPGA 资源达到高动态范围信号。该内核还具有可选的相位偏移功能。

如图2-6 对IP核DDS II进行配置 ,根据高云用户手册进行配置，输入频率为100Mz，选取输出cosine波形，仅输出幅度不输出相位，输出频率3MHz，设置无杂散动态范围为48dB，令输出位宽为8bit 。

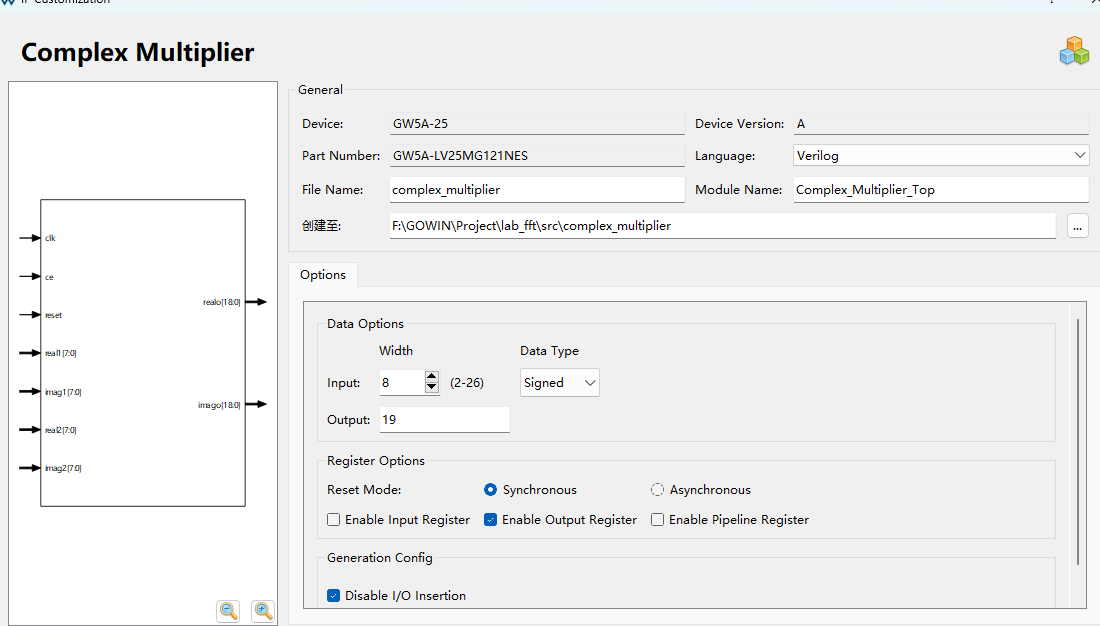


**图2-6 IP核DDS II的配置**

1. 配置IP核 Complex Multiplier（乘法器）：

Gowin Complex Multiplier是一种数字信号处理的IP，主要用于实现复数的乘法，也可实现实数的乘法（只采用实部即可）

配置如图2-7所示，配置为有符号整数，输入8bit，输出19bit，同步复位。

****

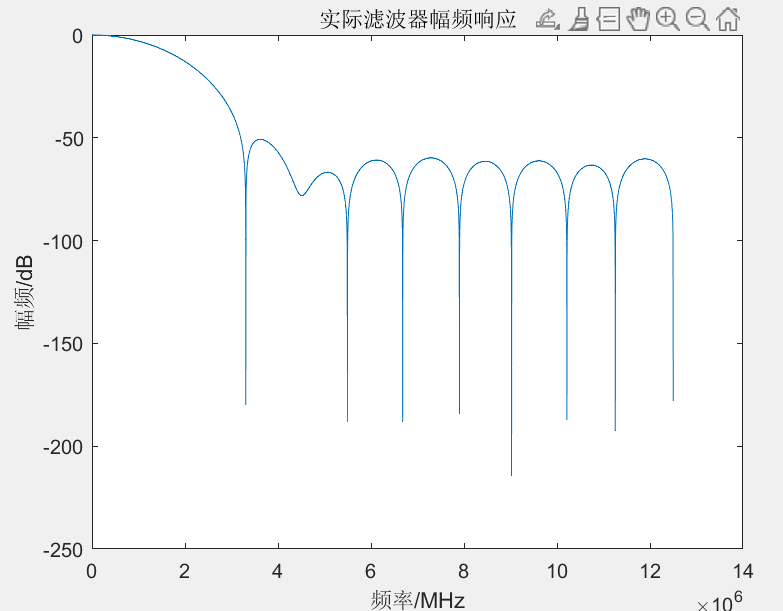
**图2-7 乘法器IP核配置**

1. 配置IP核 PLL（锁相环）：

用于为FIR滤波器系统的各个IP模块提供频率，系统频率50MHz，输出频率100MHz，32MHz，50MHz

### 2.2.3实际滤波器程序设计（使用海明窗的21阶FIR数字滤波器）

MATLAB软件仿真的滤波器的幅度频响特性如下图2-8 所示。

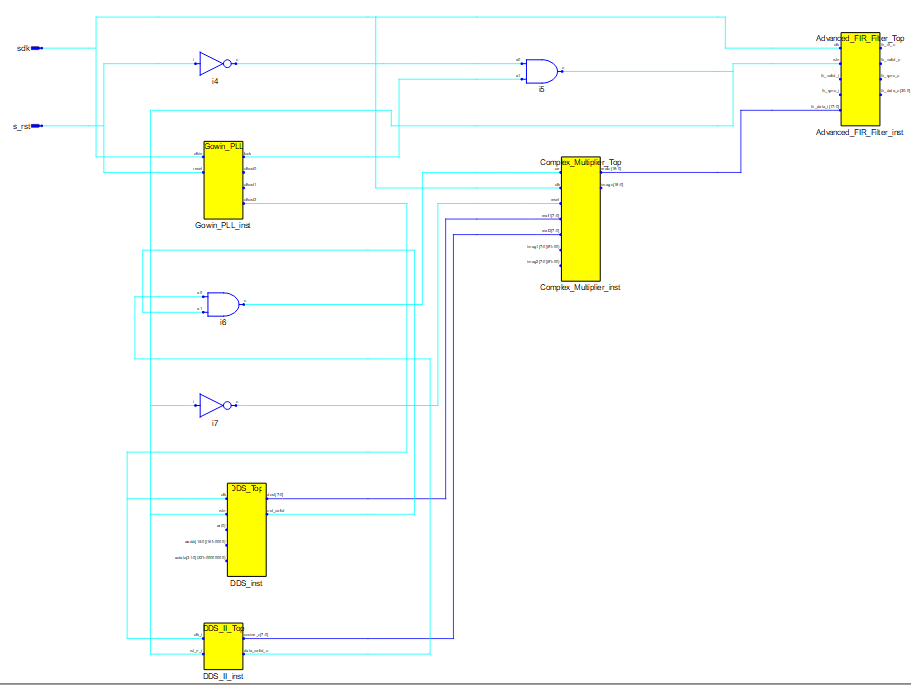


**图2-8 21阶FIR滤波器的幅频特性曲线**

### 2.2.4.VerilogHDL的实现

Verilog顶层文件与仿真文件放于附录，可供查看。

如图2-8所示，为编译后生成的RTL框图



**图2-8 RTL 系统级框图**

# 第三章 FIR滤波器仿真验证

## 3.1设置混合信号

通过乘法器IP核，将3MHz和4MHz正弦波相乘，根据三角函数变换公式，混合信号将包含1MHz信号，和7MHz信号，通过FIR滤波器后，7MHz信号将被滤除，留下1MHz正弦波信号信号。

(3-1)

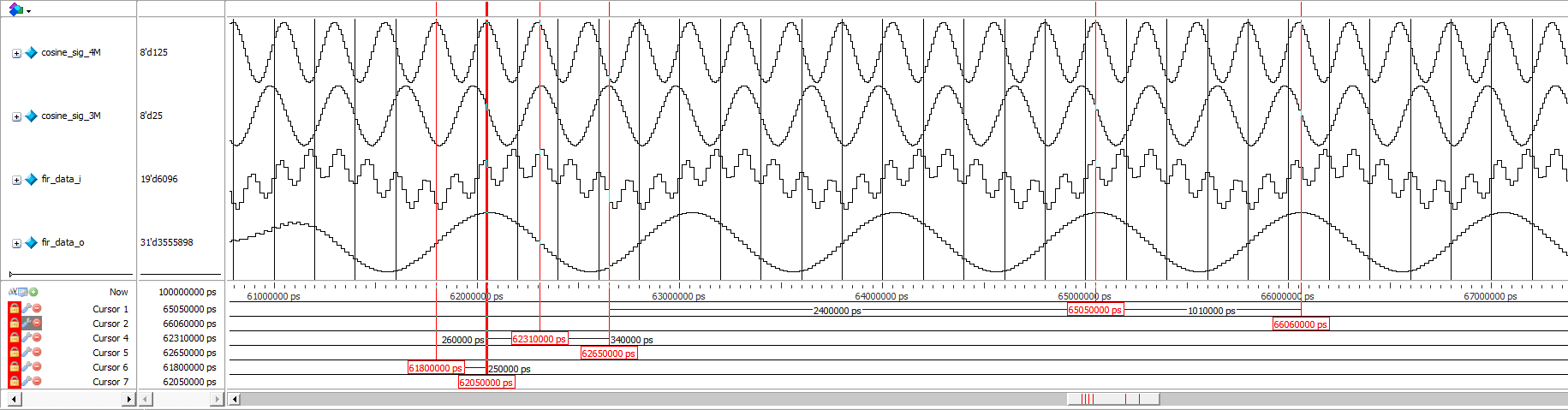
(3-2)

(3-3)

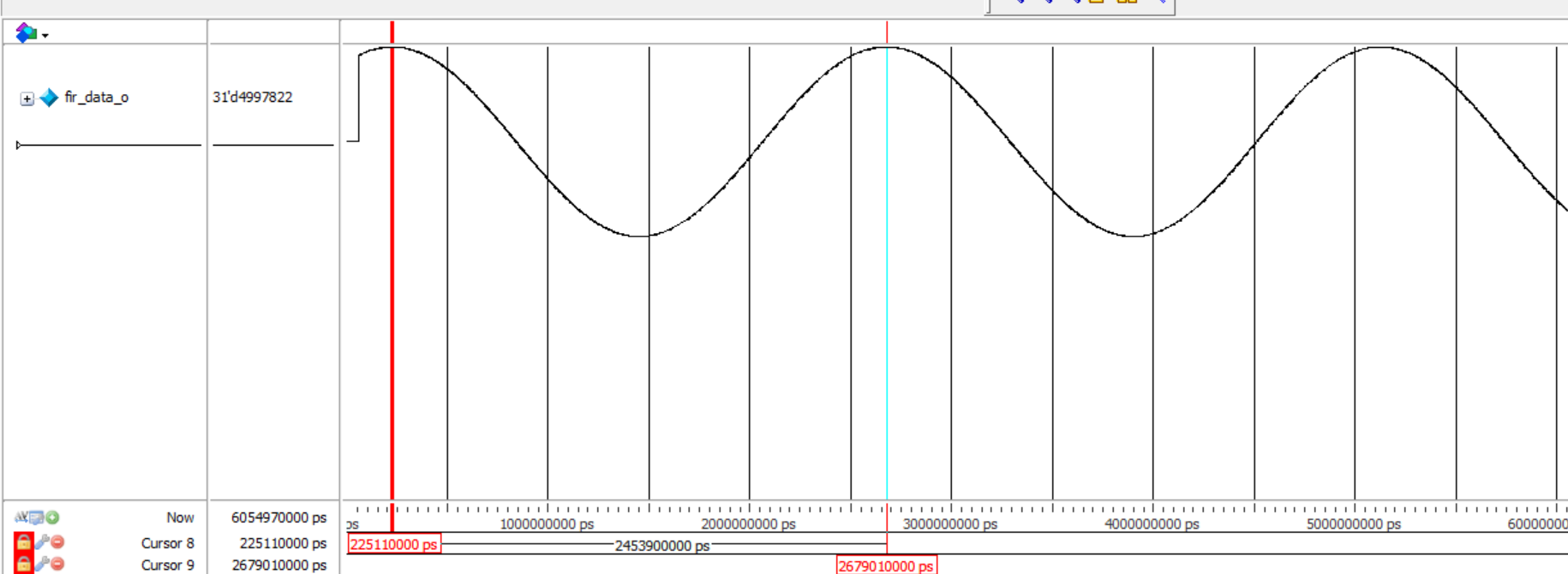
(3-4)

## 3.2 仿真结果：

如图3-1， cosine\_sig\_4M（4MHz）信号和cosine\_sig\_3M（3MHz）信号相乘后得到fir\_data\_i(包含了7MHz信号和1MHz)信号，输出fir\_data\_0(1MHz)信号。 图中对每个正弦信号进行了时间测量，测得cosin\_sig\_4M信号周期为250000ps，即250ns，即为4MHz；测得cosine\_sig\_3M信号周期为340000ps，即340ns，约为3MHz ； 测得fir\_data\_o信号周期为10 10000ps ，即1010ns，即1MHz信号。



**图3-1 仿真信号图**

****

**图3-2 FIR滤波后得到500Hz信号**

如图3-2，将一个DDS的输出频谱设置为4.0005MHz，即4M+500Hz , 再与另一个4MHz的DDS输出信号相乘后经过FIR低通滤波器，就能得到输出500Hz信号。（26790 10000ps =2679010ns =2.679ms ,约为 373.2Hz ，误差源于DDS IP核 和 手动测量）

# 第四章 总结与展望

本次课程设计设计了一个基于FPGA的IP核的FIR数字滤波器，使用matlab工具箱，采用海明窗设计了FIR低通滤波器，截止频率1.26MHz，阻带衰减>50dB ，使用国产高云FPGA的IP核：FIR 滤波器、两个DDS频率生成器、复杂乘法器、PLL锁相环等5个IP核，进行了滤波器快速设计，并编写了工程Top文件和testbench文件，使用modelsim软件进行仿真验证，将一个 3MHz信号和4MHz信号相乘后经过FIR滤波器滤波，3MHz信号和4MHz信号相乘后得到1MHz信号和7MHz信号的混杂信号，滤波后仅剩1MHz信号，验证得到结果正确。

该基于FPGA的FIR滤波器也能够应用于多种数字信号处理领域，比如果心电信号，心电信号通常频率较低，通过传感器采集需要进行滤波、放大等操作，可将采集到的原始的心电信号经过在一些前置操作后使用5MHz以上的信号进行载波，得到5M+几百Hz的信号，再与5MHz本频信号相乘后经过FIR滤波器，根据三角函数变换公式，就能够得到一个几百Hz信号。

本次课程设计还有改进的空间，即实际上板测试，但是目前手里有的国产FPGA开发板并没有搭载DAC的模块，只能遗憾止步于此。

最后，使用matlab工具箱构造基于窗函数FIR滤波器函数，设计基于FPGA的FIR数字滤波器已经是一个比较常见的课题和设计方式，有着广泛的应用，本次课程设计虽然也是基于FPGA简单设计了一个FIR滤波器，但是也拥有自己的特色：使用国产FPGA和IP核进行设计。在国产FPGA逐渐“崛起”的当下，想要国产FPGA发展得更好，得到真正长远的发展，非常需要更多的开发者主动尝试使用国产FPGA及器开发平台，为开发社区提供更多的开发例子，降低软件入门门槛，提高使用体验。

# 第五章 参考文献

[1] 陈媛媛, 刘有耀. FIR 滤波器的 FPGA 设计与实现[J]. 电子设计工程, 2017, 25(24): 65-69+73. DOI:10.14022/j.cnki.dzsjgc.2017.24.015.

[2] 蒋林, 葛中芹, 杨旭等. 基于 FPGA 的高阶组合结构 FIR 数字滤波器设计[J]. 实验室科学, 2021, 24(03): 72-77.

[3] 周亚凤. 基于 FPGA 的 FIR 滤波器的设计[D]. 南京: 南京工业大学, 2005.

[3] 孙建明, 赵刚, 张迎华. 基于 DA 算法的高速高阶 FIR 滤波器的 FPGA 实现[J]. 信息与电 子工程, 2007, 5(06): 432-436.

[5] 刘圆, 黄晨灵, 高佩君, 等. 基于分段查找表的高速 FIR 滤波器的设计实现[J]. 微 电子学, 2006(05):674-678.

［6］何存富，王森，吴强，等．基于 ＦＰＧＡ 的 ＦＩＲ 滤波器在超声 导波接收系统中信号降噪的应用［Ｊ］．北京工业大学学报， ２０１８，４４（０５）：６５８　６６４

［7］张亮，刘 宇 红．基 于 ＦＰＧＡ 的 ＦＩＲ 滤 波 器 设 计 方 法 研 究 ［Ｊ］．通信技术，２０１６，４９（０８）：１０８３　１０８７． ［３］陶国彬，张秀艳，任玉霞．ＦＩＲ 滤波器的等波纹最优化设计 ［Ｊ］．大庆石油学院学报，２００７（０６）：１０５　１０７，１１１，１３１

# 第五章 代码附录

## 5.1 工程top文件

1. module fir\_top (
2. input               sclk                 ,
3. input               s\_rst                ,

6. output wire         sign\_led
8. );

11. //=================================================\
12. //\*\*\*\*\*\*Define Parameter  and Internal Signal\*\*\*\*
13. //=================================================\
15. //DDS
16. wire                clk\_32M                                         ;
17. wire                clk\_100M                                        ;
18. wire                s\_rst\_n                                         ;
19. wire [ 7:0]         cosine\_sig\_4M                                   ;
20. wire                cosine\_sig\_valid\_4M                             ;
21. wire                clk\_50M                                         ;
22. wire[ 7:0]          cosine\_sig\_3M                                   ;
23. wire                cosine\_sig\_valid\_3M                             ;

26. //multi
27. wire[18:0]          mix\_cosine\_sig                                  ;
28. wire[18:0]          mix\_cosine\_sig\_mx                               ;
30. //fir filter
31. wire                fir\_rfi\_o                                     ;
32. reg                 fir\_valid\_i                                   ;
33. reg                 fir\_sync\_i                                    ;
34. wire[18:0]          fir\_data\_i                                    ;
35. wire                fir\_valid\_o                                   ;
36. wire                fir\_sync\_o                                    ;
37. wire[30:0]          fir\_data\_o                                    ;
39. //delay
40. reg [15:0]          delay\_cnt                                       ;
41. wire                flag\_delay\_end                                  ;
43. //PLL
44. wire                pll\_lock                                        ;
46. //===========================================\
47. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main Code \*\*\*\*\*\*\*\*\*\*
48. //===========================================\


52. always @(posedge sclk or negedge s\_rst\_n) begin
53. **if**(!s\_rst\_n)
54. delay\_cnt <= 'd0 ;
55. **else** **if**(flag\_delay\_end ==1'b0  )
56. delay\_cnt   <= delay\_cnt + 1'b1  ;
58. end
60. assign flag\_delay\_end   = (delay\_cnt >= 500)? 1'b1: 1'b0 ;//延时停止标志位

63. Gowin\_PLL Gowin\_PLL\_inst (
64. .lock       (pll\_lock   ), //output lock
65. .clkout0    (clk\_50M    ), //output clkout0
66. .clkout1    (clk\_32M    ), //output clkout1
67. .clkout2    (clk\_100M   ), //output clkout1
68. .clkin      (sclk       ), //input clkin
69. .reset      (s\_rst      ) //input reset
70. );
72. assign  s\_rst\_n =   (~s\_rst) & (pll\_lock)       ;//常态下处于高电平,按下去为低电平复位
73. //output    4M cos
74. DDS\_Top DDS\_inst(
75. .clk            (clk\_100M               ), //input clk
76. .rstn           (s\_rst\_n                ), //input rstn
77. .wr             (1'b0                   ), //input wr
78. .waddr          ('d0                    ), //input [15:0] waddr
79. .wdata          ('d0                    ), //input [31:0] wdata
80. .dout           (cosine\_sig\_4M          ), //output [7:0] dout
81. .out\_valid      (cosine\_sig\_valid\_4M    ) //output out\_valid
82. );
83. // input    3M cos
84. DDS\_II\_Top DDS\_II\_inst(
85. .clk\_i          (clk\_100M                ), //input clk\_i
86. .rst\_n\_i        (s\_rst\_n                ), //input rst\_n\_i
87. .cosine\_o       (cosine\_sig\_3M          ), //output [7:0] sine\_o
88. // .sine\_o          (cosine\_sig\_3M          ),
89. .data\_valid\_o   (cosine\_sig\_valid\_3M    ) //output data\_valid\_o
90. );
91. //3M\_COSINE \* 4M\_COSINE= (3+4)=7M & (4-3)=1M 延时一周期出数据
92. Complex\_Multiplier\_Top Complex\_Multiplier\_inst(
93. .ce             (cosine\_sig\_valid\_3M & cosine\_sig\_valid\_4M              ), //input ce
94. .clk            (sclk                                                   ), //input clk
95. .reset          (~s\_rst\_n                                               ), //input reset
96. .real1          (cosine\_sig\_4M                                          ), //input [7:0] real1
97. .real2          (cosine\_sig\_3M                                          ), //input [7:0] real2
98. .imag1          ('d0                                                    ), //input [7:0] imag1
99. .imag2          ('d0                                                    ), //input [7:0] imag2
100. .realo          (mix\_cosine\_sig                                         ), //output [18:0] realo
101. .imago          (mix\_cosine\_sig\_mx                                      ) //output [18:0] imago
102. );
103. //FIR Filter :FC=8M
104. always @(posedge sclk or negedge s\_rst\_n) begin
105. **if**(!s\_rst\_n)begin
106. fir\_valid\_i <= 1'b0  ;
107. fir\_sync\_i  <= 1'b0 ;
108. end
109. **else** **if**(fir\_rfi\_o == 1'b1 && flag\_delay\_end== 1'b1)begin
110. fir\_valid\_i <= 1'b1 ;
111. fir\_sync\_i  <= 1'b1 ;
112. end
113. **else** begin
114. fir\_valid\_i <= 1'b0  ;
115. fir\_sync\_i  <= 1'b0 ;
116. end
118. end
119. assign  fir\_data\_i  = (fir\_sync\_i)? mix\_cosine\_sig:     'd0         ;
120. Advanced\_FIR\_Filter\_Top Advanced\_FIR\_Filter\_inst(
121. .clk            (sclk                                                 ), //input clk
122. .rstn           (s\_rst\_n                                              ), //input rstn
123. .fir\_rfi\_o      (fir\_rfi\_o                                            ), //output fir\_rfi\_o
124. .fir\_valid\_i    (fir\_valid\_i                                          ), //input fir\_valid\_i
125. .fir\_sync\_i     (fir\_sync\_i                                           ), //input fir\_sync\_i
126. .fir\_data\_i     (fir\_data\_i                                           ), //input [18:0] fir\_data\_i
127. .fir\_valid\_o    (fir\_valid\_o                                          ), //output fir\_valid\_o
128. .fir\_sync\_o     (fir\_sync\_o                                           ), //output fir\_sync\_o
129. .fir\_data\_o     (fir\_data\_o                                           ) //output [30:0] fir\_data\_o
130. );


134. endmodule

## 5.2 工程testbench文件

1. //=========================================\
2. //\*TOP
3. //\*
4. //=========================================\
5. `timescale 1ns/1ns
6. module tb\_dds\_fir ();
8. reg clk ;
9. reg s\_rst;
11. initial begin
12. clk = 0; //50MHz
13. s\_rst = 1 ;
14. #10 s\_rst = 0;
15. //   #20 rstn = 1;
16. end
18. GSR GSR(.GSRI(1'b1));
20. always #10 clk <= ~ clk ;    //产生50MHz时钟
22. fir\_top fir\_top\_inst(
23. .sclk        (clk )      ,
24. .s\_rst        (s\_rst)       ,

27. .sign\_led    ()
29. );
31. endmodule

## 5.3 matlab生成FIR抽头系数文件

1. %%
2. clear;
3. %数组NUM来源于matlab filter设计工具导出的抽头系数，直接copy就行
4. NUM=[ 0  -0.0086  0.0725  0.4361  0.4361  0.0725  -0.0086 0];
5. Fs=4e6;%信号频率
6. N=length(NUM);
7. T=1/Fs;Tp=N\*T;%采样时间
8. t=(0:N-1)/Fs;%t=0-2s
10. N\_fft = 4e6; % 频率点的数量，通常选择2的幂次方以利用FFT的效率
11. % 计算频率向量（归一化）
12. f = (0:N\_fft-1)\*(Fs/N\_fft);
13. % f=0:Fs/N:Fs\*(N-1)/N;%步长f1/N
14. NUM1 = NUM\*10000
15. disp(NUM1);
16. % 使用 freqz 计算频率响应
17. [H3, w3] = freqz(NUM, 1, N\_fft, Fs);
19. % w=2\*pi\*f/Fs;%归一化频
20. plot(w3, 20\*log10(abs(H3)));
21. title('实际滤波器幅频响应');
22. xlabel('频率/MHz');
23. ylabel('幅频/dB');
25. load("Num.mat");
26. file\_name="./coeff.dat" ;
27. fid =fopen(file\_name,'w');
28. **for** i= 1: 8
29. fprintf(fid ,'%d\n',NUM1(i));
30. end
31. fclose(fid);