

**双通道 14 位 DA 输出模块**

**AN9767 用户手册**

**Rev. 1.00**

**ALINX**

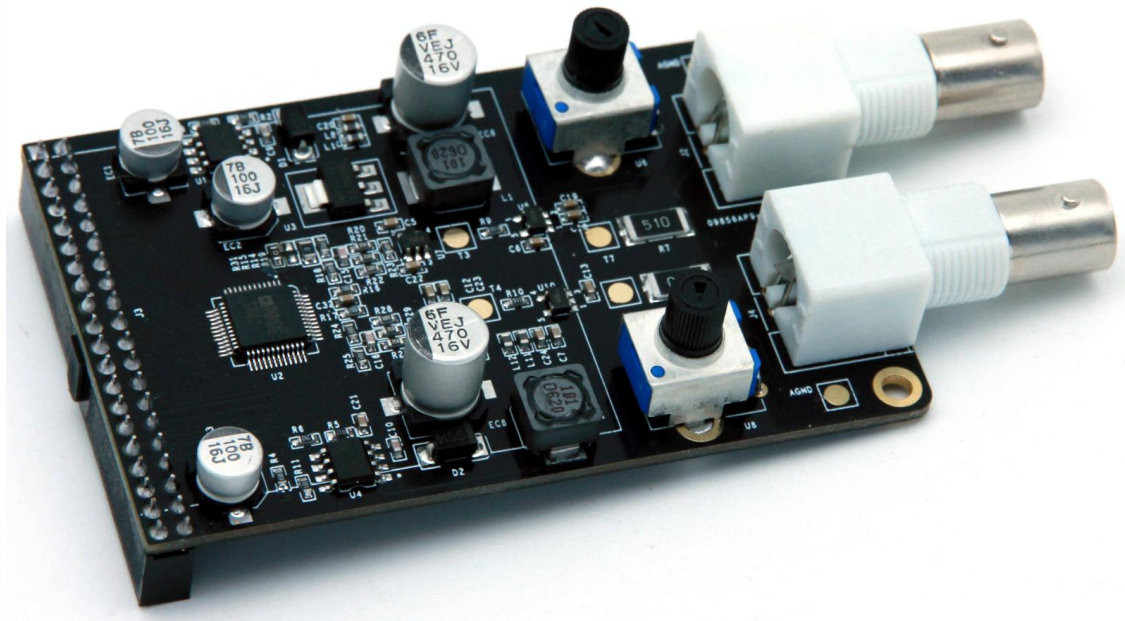
## 版本记录

版本	时间	作者	描述
Rev1.00	2017-4-28		First Release

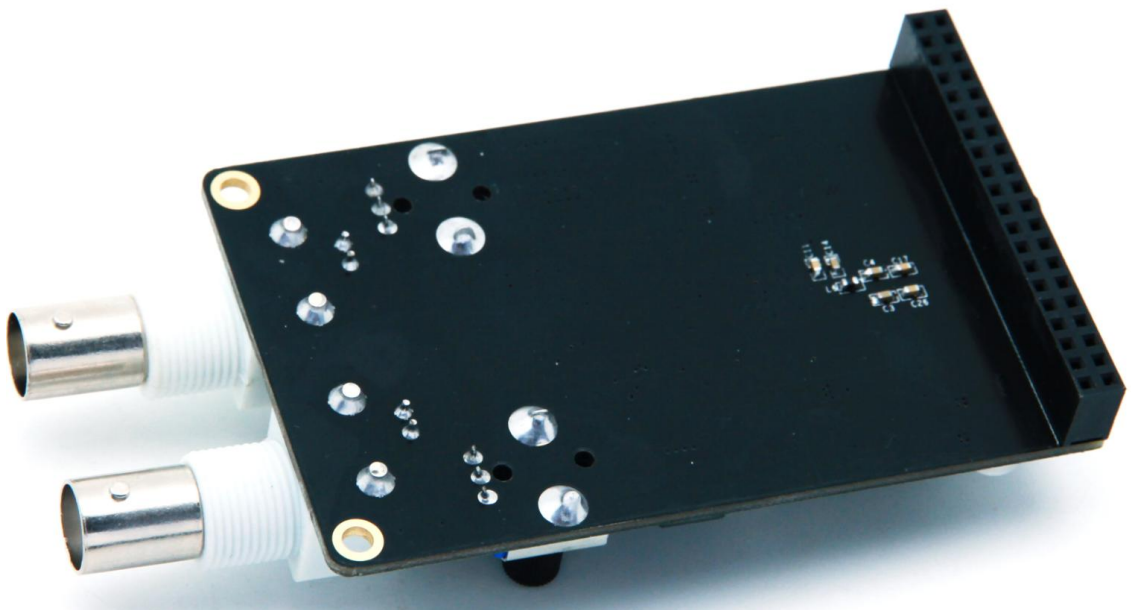
## 第一部分 双通道 14 位 DA 模块说明介绍

黑金双通道 14 位 DA 输出模块 AN9767 采用 ANALOG DEVICES 公司的 AD9767 芯片，支持独立双通道、14 位、125MSPS 的数模转换。模块留有一个 40 针的排母用于连接 FPGA 开发板，2 个 BNC 连接器用于模拟信号的输出。

AN9767 模块实物照片如下：



AN9767 模块正面图



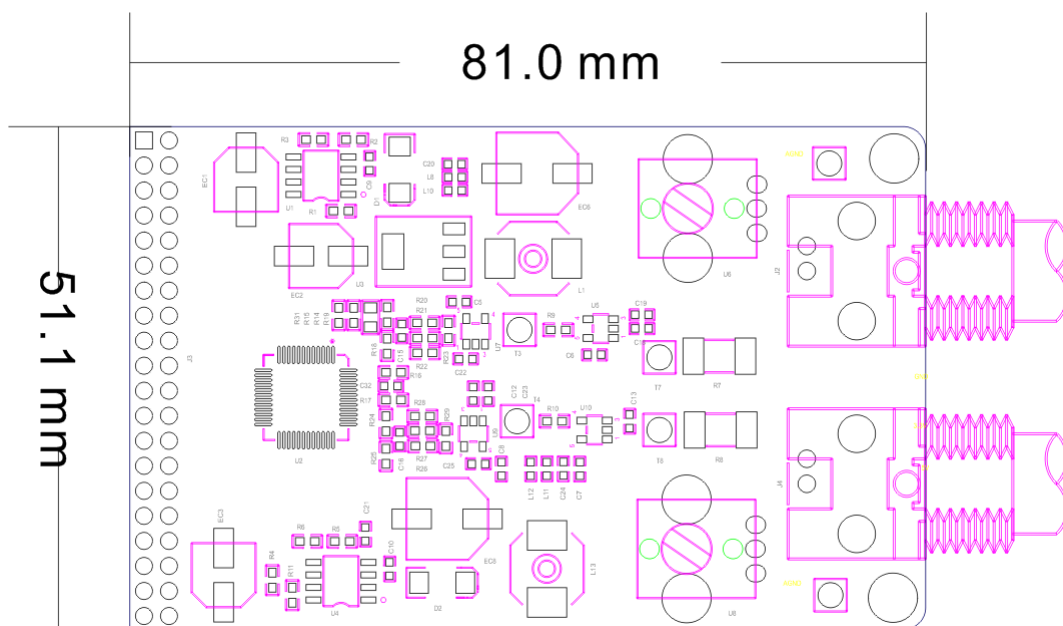
AN9767 模块背面图

## 1.1 AN9767 模块的参数说明

以下为 AN9767 双通道 DA 模块的详细参数:

- DA 转换芯片 : AD9767 ;
- 通道数 : 2 通道 ;
- DA 转换位数 : 14bit ;
- DA 更新速率 : 125 MSPS ;
- 输出电压范围 : -4V~+4V ;
- 模块 PCB 层数 : 4 层 , 独立的电源层和 GND 层 ;
- 模块接口 : 40 针 2.54mm 间距排座 , 方向向下 ;
- 工作温度 : -40°~85° 模块使用芯片均满足工业级温度范围
- 输出接口 : 2 路 BNC 模拟输出接口 ( 用 BNC 线可以直接连接到示波器 ) ;

## 1.2 AN9767 模块的参数说明

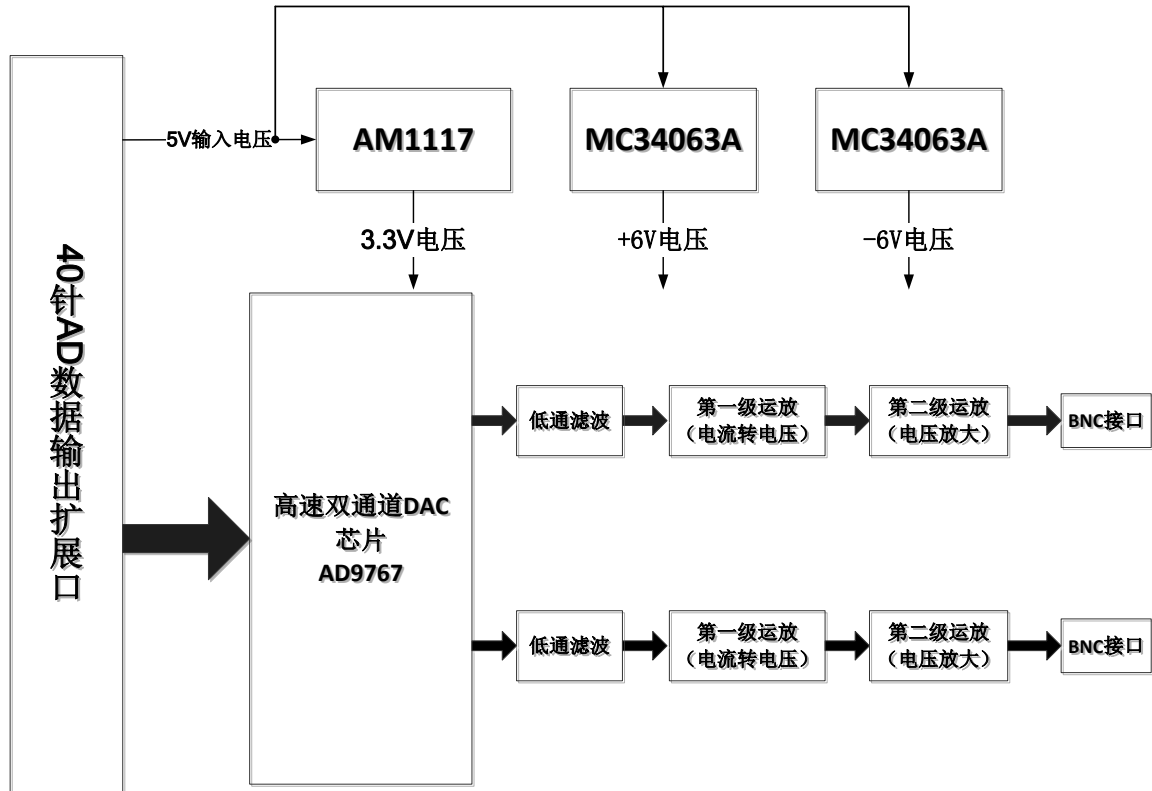


AN9767 双通道 DA 模块尺寸图

## 第二部分 模块功能说明

### 2.1 AN9767 模块原理框图

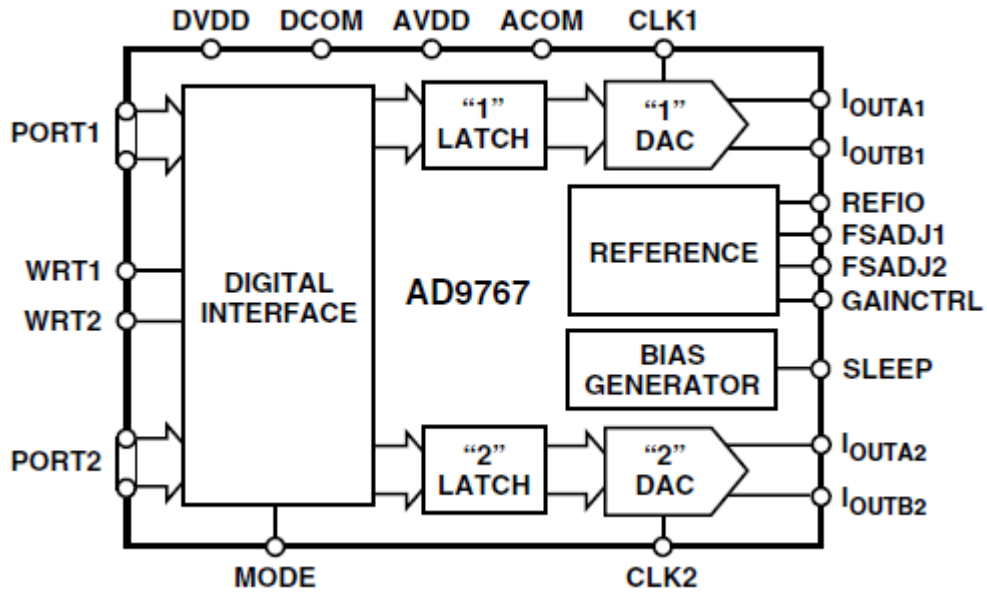
AN9767 模块的原理设计框图如下：



### 2.2 AD9767 芯片简介

AD9767 是双端口、高速、双通道、14 位 CMOS DAC, 芯片集成两个高品质 TxDAC+®内核、一个基准电压源和数字接口电路,采用 48 引脚小型 LQFP 封装。器件提供出色的交流和直流性能,同时支持最高 125 MSPS 的更新速率。

AD9767 的功能框图如下：



## 2.3 电流电压转换及放大

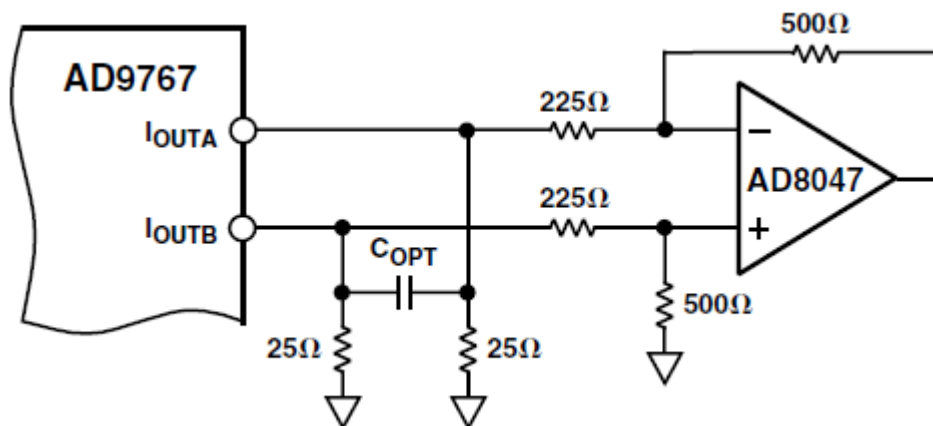
AD9767 的两路 DA 输出都为补码形式的电流输出  $I_{outA}$  和  $I_{outB}$ 。当 AD9767 数字输入为满量程时 (DAC 的输入的 14 位数据都为高),  $I_{outA}$  输出满量程的电流输出 20mA。 $I_{outB}$  输出的电流为 0mA。具体的电流和 DAC 的数据的关系如下公式所示：

$$I_{OUTA} = (DAC\ CODE / 16384) \times I_{OUTFS}$$

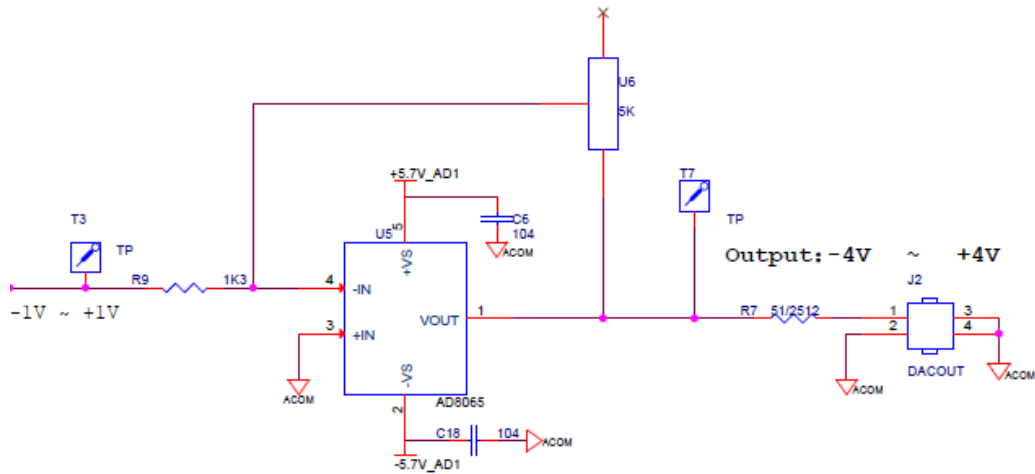
$$I_{OUTB} = (16383 - DAC\ CODE) / 16384 \times I_{OUTFS}$$

其中  $I_{outFS} = 32 \times I_{ref}$ ，在 AN9767 模块设计中,  $I_{ref}$  的值由电阻 R16 的值决定 如果  $R16 = 1.92K$  那  $I_{ref}$  的值就是 0.625mA。这样  $I_{outFS}$  的值就是 20mA。

AD9767 输出的电流通过第一级运放 AD6045 转换成 -1V~+1V 的电压。具体的转换电路如下图所示：



第一级运放转换后的 $-1V \sim +1V$  的电压通过第二级运放变换到更高幅度的电压信号，这个运放的幅度大小可以通过调整板上的可调电阻来改变。通过第二级运放，模拟信号的输出范围高达 $-4V \sim +4V$ 。

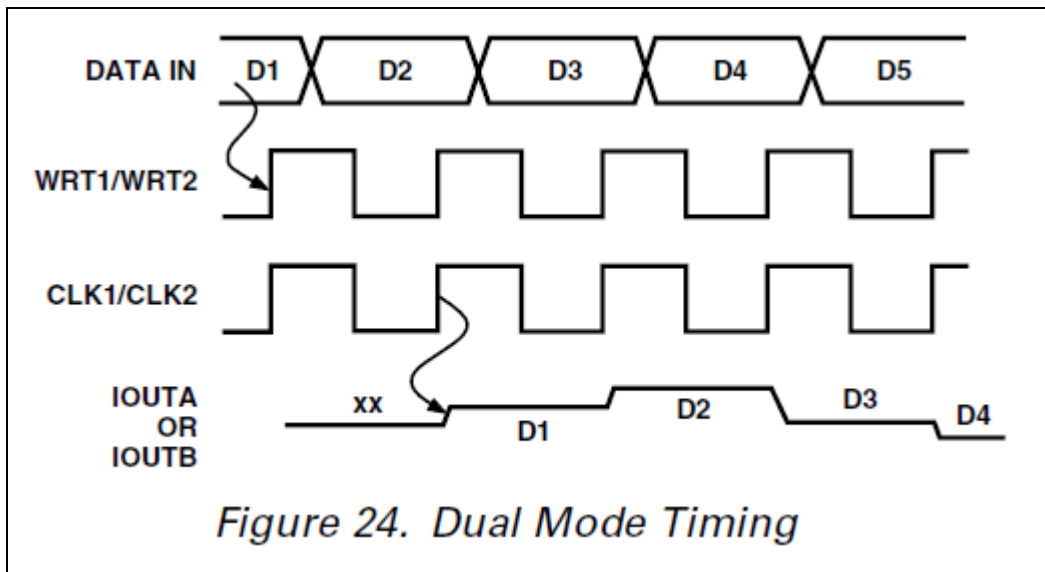


下表为数字输入信号和各级运放输出后的电压对照表：

DAC 数据输入值	AD9767 电流输出	第一级运放输出	第二级运放输出
3fff(14 位全高)	+20mA	-1V	+4V
0(14 位全低)	-20mA	+1V	-4V
2000 (中间值)	0mA	0V	0V

## 2.4 数字接口定义

AD9767 芯片的数字接口可以通过芯片的模式管脚(MODE)来配置成双端口模式(Dual)或者交叉(Interleaved)模式。在 AN9767 模块设计中，AD9767 芯片是工作在双端口模式，双通道的 DA 数字输入接口是独立分开的。双端口模式(Dual)的数据时序图如下图所示：



给 AD9767 芯片的 DA 数据通过时钟 CLK 和写信号 WRT 的上升沿输入到芯片进行 DA 转换。

## 2.5 模块 40 针排母的引脚分配：

引脚号	引脚名称	备注
1	GND	地
2	+5V	5V 电源输入
3	P1_DATA13	DAC 通道 1 数据 13 位
4	P1_DATA12	DAC 通道 1 数据 12 位
5	P1_DATA11	DAC 通道 1 数据 11 位
6	P1_DATA10	DAC 通道 1 数据 10 位
7	P1_DATA9	DAC 通道 1 数据 9 位
8	P1_DATA8	DAC 通道 1 数据 8 位
9	P1_DATA7	DAC 通道 1 数据 7 位
10	P1_DATA6	DAC 通道 1 数据 6 位
11	P1_DATA5	DAC 通道 1 数据 5 位
12	P1_DATA4	DAC 通道 1 数据 4 位
13	P1_DATA3	DAC 通道 1 数据 3 位
14	P1_DATA2	DAC 通道 1 数据 2 位
15	P1_DATA1	DAC 通道 1 数据 1 位
16	P1_DATA0	DAC 通道 1 数据 0 位
17	P1_WRT	DAC 通道 1 数据写信号
18	P1_CLK	DAC 通道 1 数据写时钟
19	P2_CLK	DAC 通道 2 数据写时钟
20	P2_WRT	DAC 通道 2 数据写信号
21	P2_DATA13	DAC 通道 2 数据 13 位
22	P2_DATA12	DAC 通道 2 数据 12 位

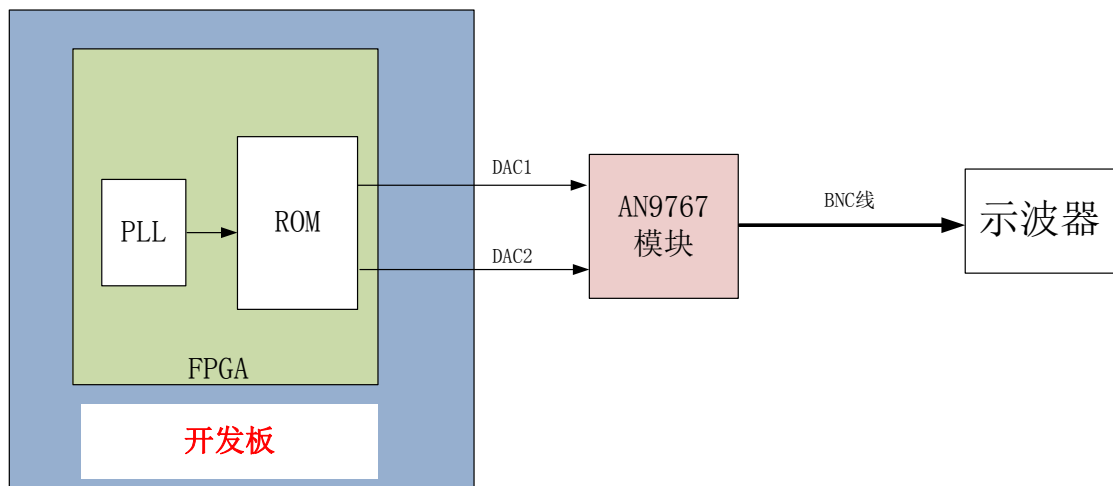


23	P2_DATA11	DAC 通道 2 数据 11 位
24	P2_DATA10	DAC 通道 2 数据 10 位
25	P2_DATA9	DAC 通道 2 数据 9 位
26	P2_DATA8	DAC 通道 2 数据 8 位
27	P2_DATA7	DAC 通道 2 数据 7 位
28	P2_DATA6	DAC 通道 2 数据 6 位
29	P2_DATA5	DAC 通道 2 数据 5 位
30	P2_DATA4	DAC 通道 2 数据 4 位
31	P2_DATA3	DAC 通道 2 数据 3 位
32	P2_DATA2	DAC 通道 2 数据 2 位
33	P2_DATA1	DAC 通道 2 数据 1 位
34	P2_DATA0	DAC 通道 2 数据 0 位
35	-	NC
36	-	NC
37	GND	地
38	GND	地
39	-	NC
40	-	NC

## 第三部分 正选波产生程序说明

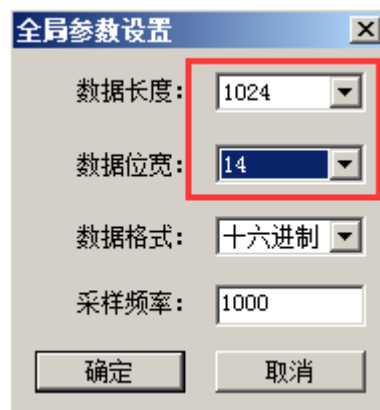
我们提供了 2 个 AN9767 模块的 DA 测试程序，一个是正选波发生程序，通过 AN9767 模块来实现正选波信号的输出。另一个三角波发生程序，通过 AN9767 模块来实现三角波信号的输出。因为三角波的例子比较简单，我们不再介绍，下面为大家介绍一下正选波输出程序：

正选波测试程序是通过读取 FPGA 内部的一个 ROM 中存储的正选波数据，然后把正选波的数据输出到 AN9767 模块进行数模的转换，从而得到正选波的模拟信号。正选波测试程序的示意图如下：

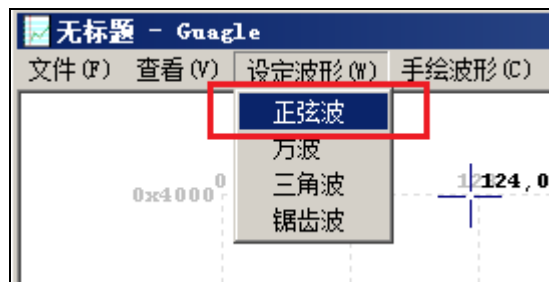


### 3.1 生成 ROM 初始化文件

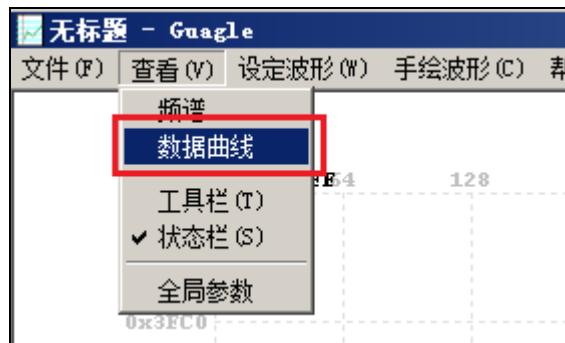
程序中我们会用到一个 ROM 用于存储 1024 个 14 位的正弦波数据，首先我们需要准备 ROM 的初始化文件(如果是 ALTERA 开发板的话是 mif 文件，如果是 Xilinx 开发板的话是 coe 文件)。以下为生成正弦波 ROM 数据文件的方法：首先打开“软件工具”目录下的 Guagle\_wave 工具，选择菜单“查看”->“全局参数设置”，设置参数如下：



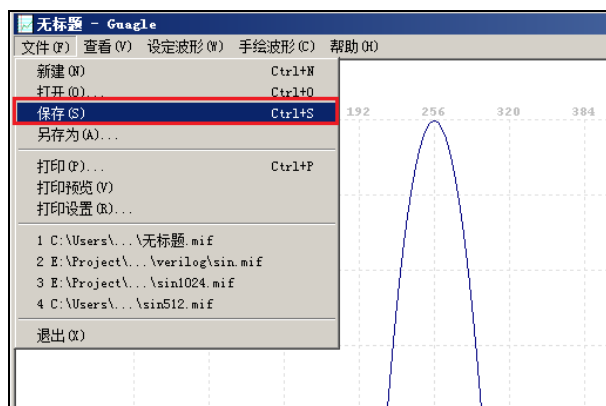
在菜单 " 设定波形 " 里选择正弦波。

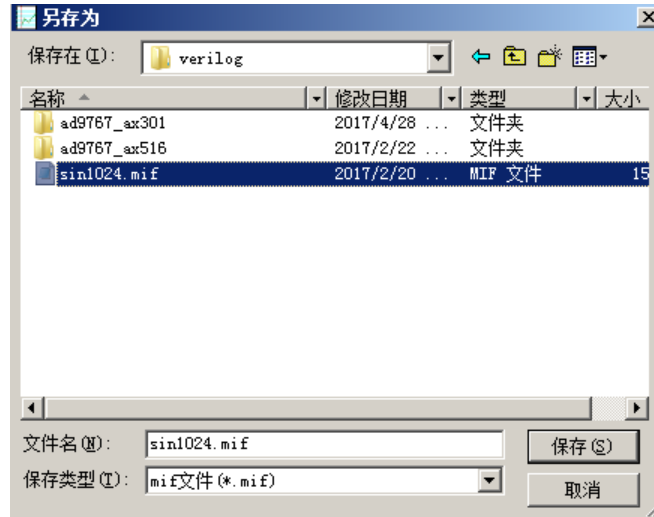


在菜单 " 查看 " 里选择数据曲线。



保存为 mif 格式的文件 sin1024.mif ( 注意不要选择另存为，不然存储的文件为空 )。

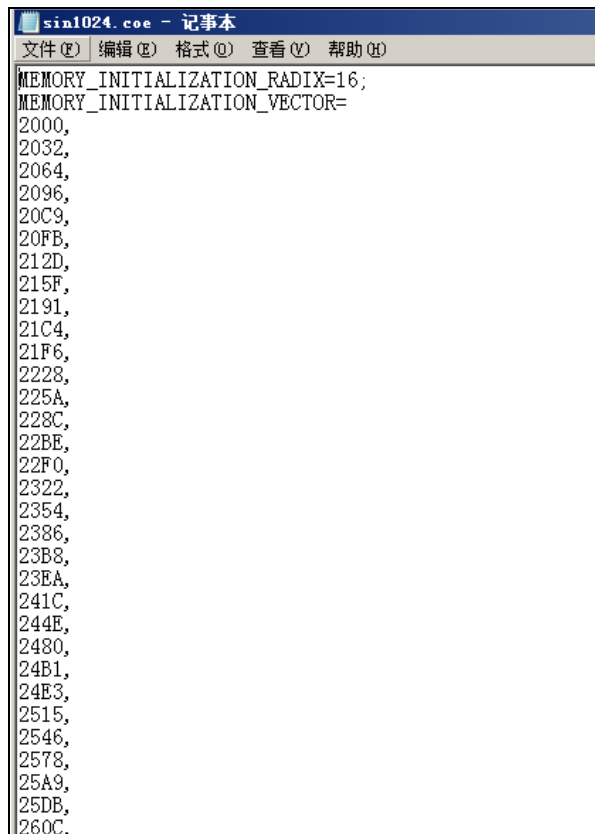




这样 mif 文件就生成好了，对于 Altera 的开发板 mif 文件就是 ROM 的初始化文件了。我们可以用记事本打开 sin1024.mif 文件查看里面的数据格式和内容。

```
DEPTH = 1024;
WIDTH = 14;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
    BEGIN
0000 : 2000;
0001 : 2032;
0002 : 2064;
0003 : 2096;
0004 : 20C9;
0005 : 20FB;
0006 : 212D;
0007 : 215F;
0008 : 2191;
0009 : 21C4;
000A : 21F6;
000B : 2228;
000C : 225A;
000D : 228C;
000E : 22BE;
000F : 22F0;
0010 : 2322;
0011 : 2354;
0012 : 2386;
0013 : 23B8;
0014 : 23EA;
0015 : 241C;
0016 : 244E;
```

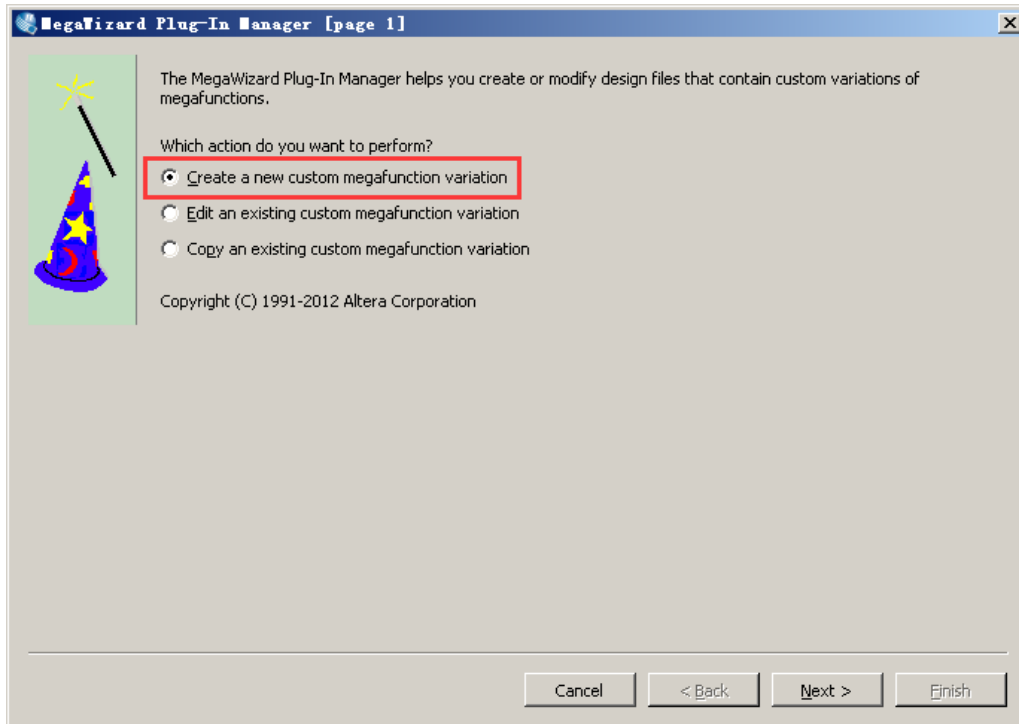
对于 Xilinx 的开发板，用户还需要把它修改为 Coe 的文件格式。因为 Xilinx 的 ROM 的初始化文件 coe 的格式和 mif 文件的格式有些不一样，所以我们需要把生成的 mif 用文本文件或者 excel 重新编辑成如下的格式(16 进制,1024 个数据长度)：



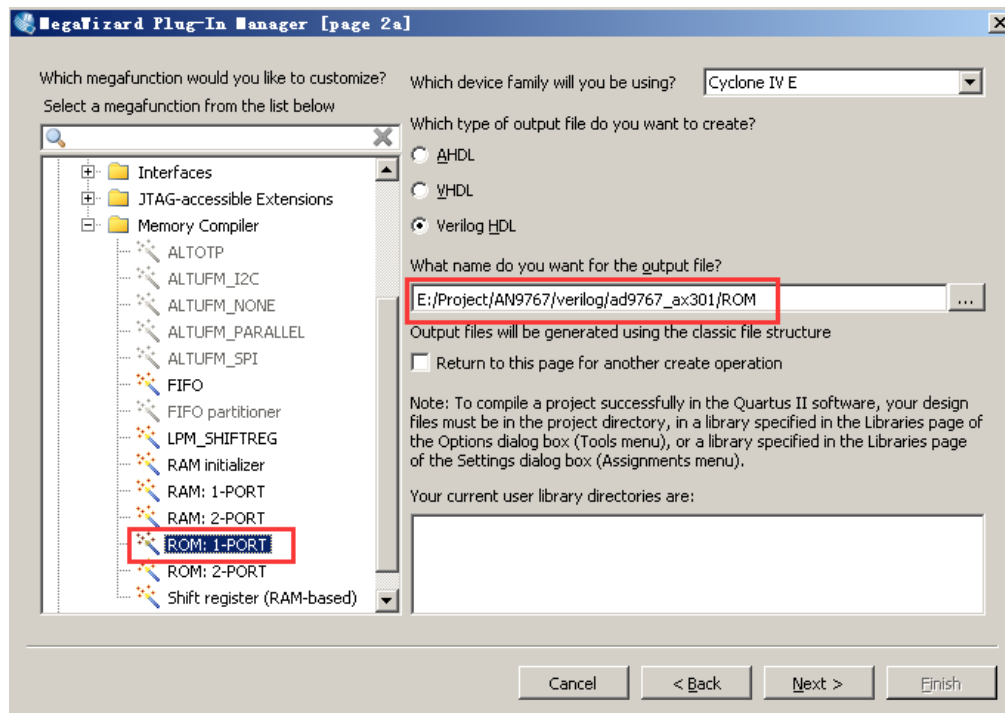
```
sin1024.coe - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
MEMORY_INITIALIZATION_RADIX=16;
MEMORY_INITIALIZATION_VECTOR=
2000,
2032,
2064,
2096,
20C9,
20FE,
212D,
215F,
2191,
21C4,
21F6,
2228,
225A,
228C,
22BE,
22F0,
2322,
2354,
2386,
23B8,
23EA,
241C,
244E,
2480,
24B1,
24E3,
2515,
2546,
2578,
25A9,
25DB,
260C.
```

### 3.2 添加 ROM IP 及配置初始化文件

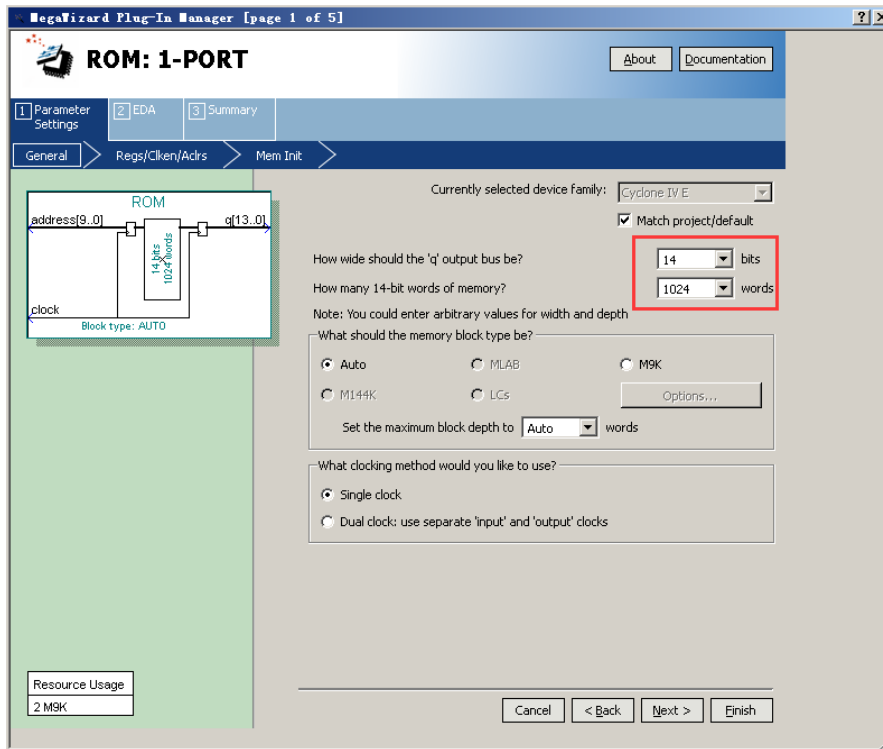
下面我们以 AX301 开发板为例介绍一下 ROM IP 的生成和配置, 我们先新建一个 Quartus 工程, 向工程中添加一个 ROM IP, 选择菜单 Tool->MegaWizard Plug-In Manager, 在弹出的 MegaWizard Plug-in Manager 窗口选择第一项 "Create a new custom megafunction variation".



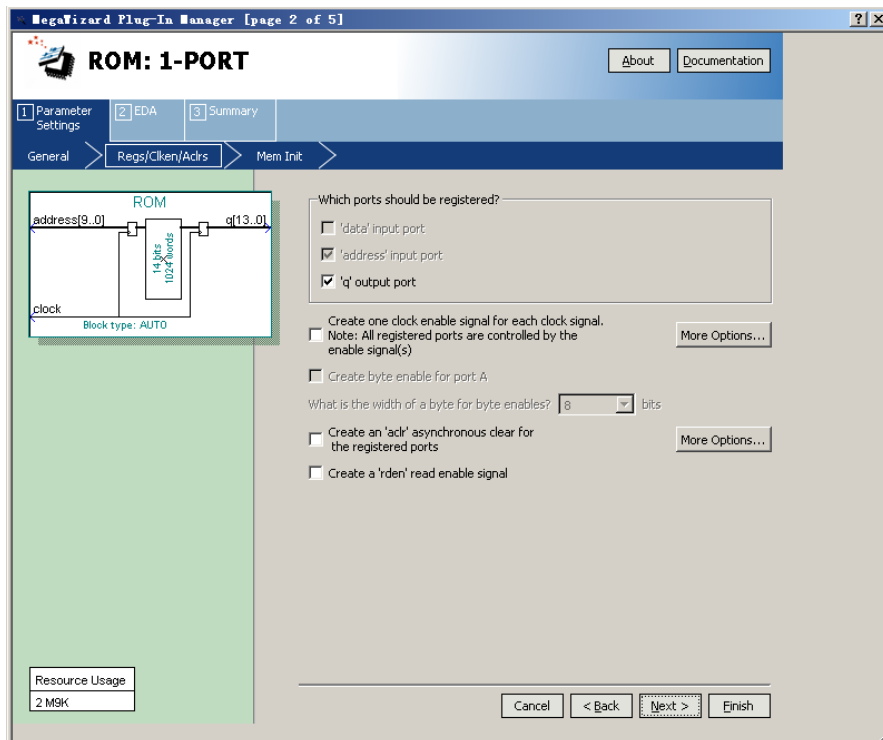
选择 ROM:1-PORT, 再在 output file 栏里输入存放 IP 的目录和名称, 这里我们取名为 rom。



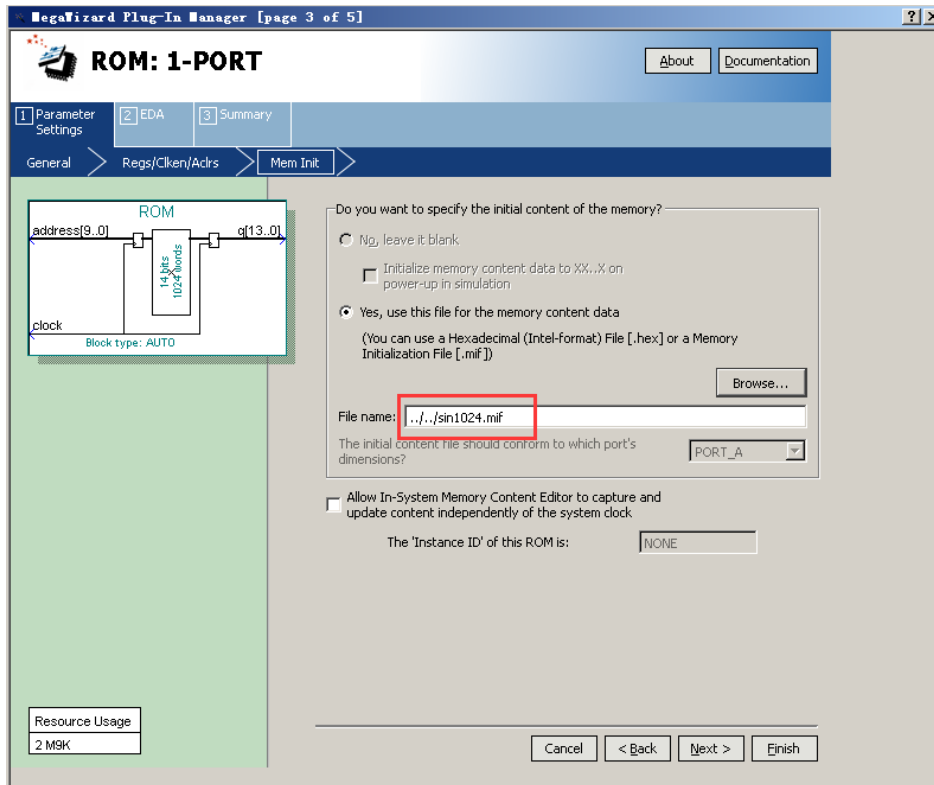
选择 ROM 的数据宽度为 14bits,数据的深度为 1024 个数据。这个设置刚好和前面生成的 mif 文件的大小一致。



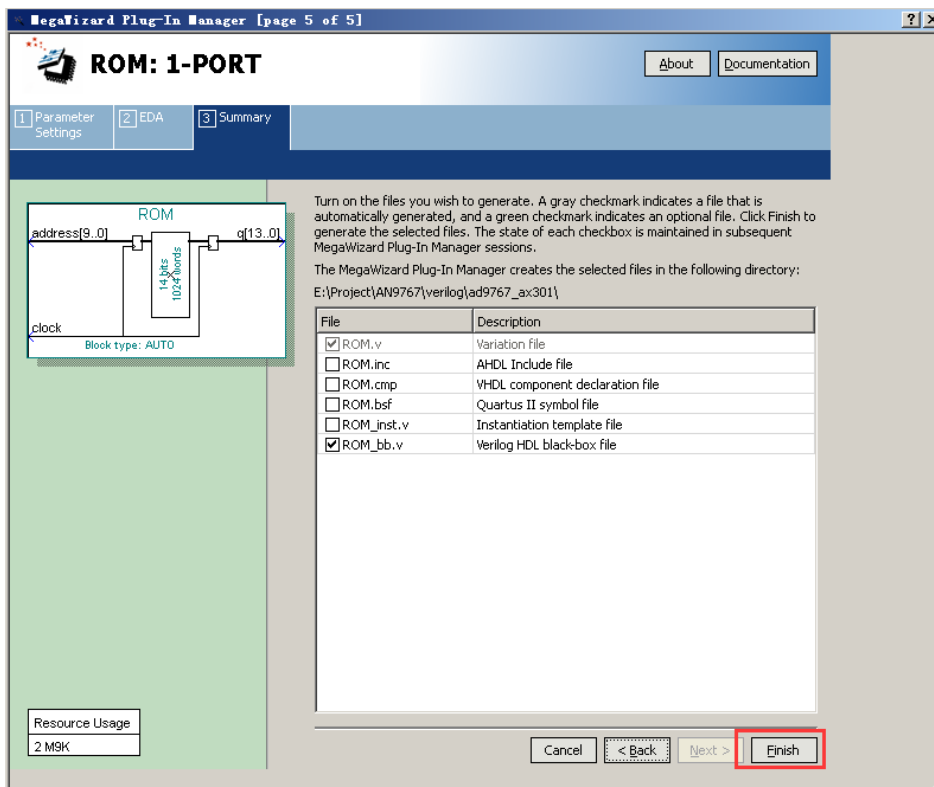
这一页不用修改，保留默认设置，点击 Next 按钮。



在这页面中,点击 Browse..按钮再选择刚才我们生成的 ROM 初始化文件 sin1024.mif。



点击 Finish 完成按钮，完成 ROM IP 的添加。



关于 Xilinx FPGA 开发板的 ROM IP 的添加和配置我们这里就不介绍了，配置的方法跟 ALTERA 的配置都差不多的。



### 3.3 编写正选波发生程序

```

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
//正选波发生器--2 路输出 -10V ~ +10V
//////////////////////////////////////////////////////////////////
module ad9767_test(
    input clk,          //fpga clock

    output da1_clk,     //DA1 时钟信号
    output da1_wrt,     //DA1 数据写信号
    output [13:0] da1_data, //DA1 data

    output da2_clk,     //DA2 时钟信号
    output da2_wrt,     //DA2 数据写信号
    output [13:0] da2_data //DA2 data

);

reg [9:0] rom_addr;

wire [13:0] rom_data;
wire clk_50;
wire clk_125;

assign da1_clk=clk_125;
assign da1_wrt=clk_125;
assign da1_data=rom_data;

assign da2_clk=clk_125;
assign da2_wrt=clk_125;
assign da2_data=rom_data;

//DA output sin waveform
always @(negedge clk_125)
begin
    rom_addr <= rom_addr + 1'b1 ; //一个正选波采样点为 1024,输出正选波频率 125/1024=122Khz
    // rom_addr <= rom_addr + 4 ; //一个正选波采样点为 256,输出正选波频率 125/256=488Khz
    // rom_addr <= rom_addr + 128 ; //一个正选波采样点为 8,输出正选波频率 125/1024=15.6Mhz
end

ROM ROM_inst (
    .clock (clk_125), // input clka

```

```
.address (rom_addr), // input [8 : 0] addra
.q      (rom_data) // output [7 : 0] douta
);

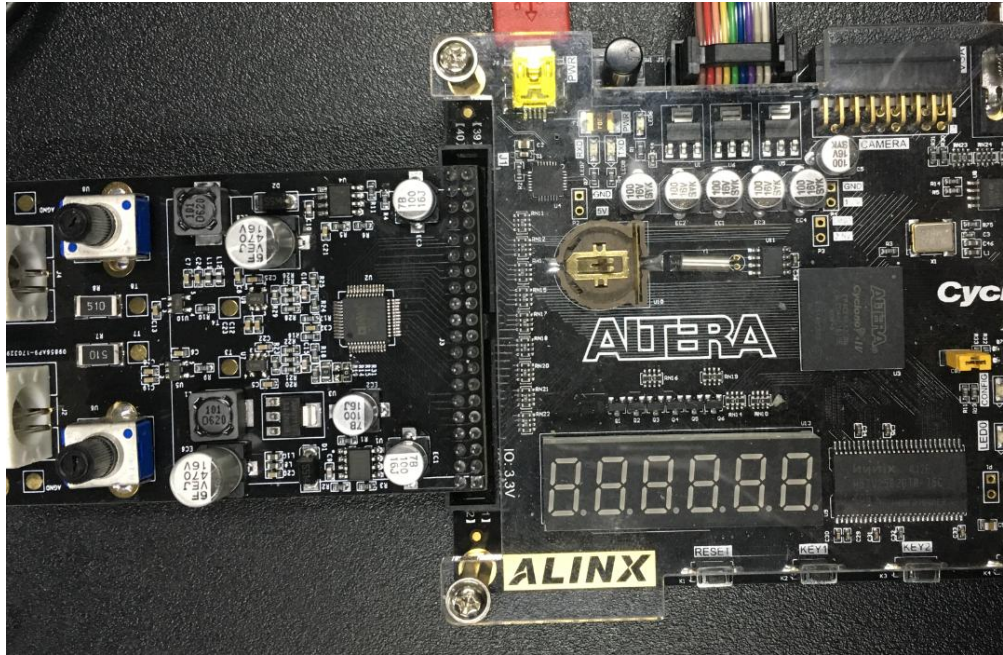
PLL PLL_inst(
    .areset (1'b0),
    .inclk0 (clk),
    .c0     (clk_50),
    .c1     (clk_125),
    .locked ()
);

endmodule
```

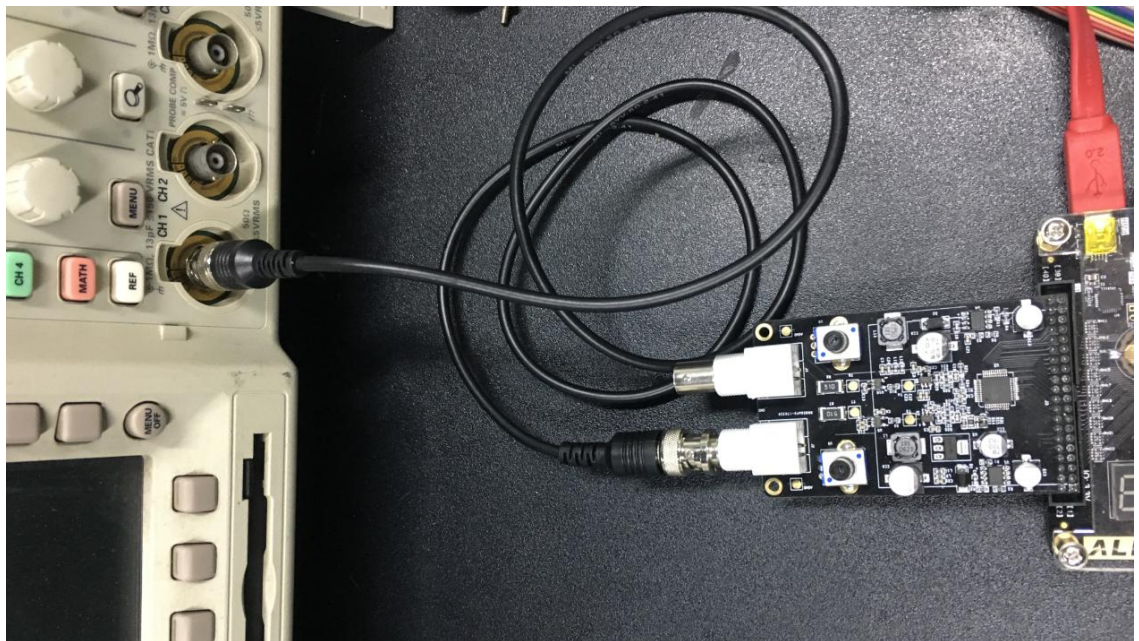
程序比较简单，通过一个 PLL IP 来产生 125M 的 DA 输出时钟，然后就是循环读取存放在 ROM 中的 1024 个数据，并输出到通道 1 和通道 2 的 DA 数据线上。程序中可以通过地址的加 1，加 4，或者加 128 来选择输出不同的频率的正选波。

## 第四部分 硬件连接

AN9767 模块和 FPGA 开发板的硬件连接很简单，只要把 DA 模块的 40 针的母座 J3 插到 FPGA 开发板的扩展口上，连接器的管脚 1 对齐就好了。以下为黑金 AX301B 开发板的 J1 扩展口和 AN9767 双通道 DA 模块的硬件连接图（如果需要连接 J2 扩展口，管脚需要重新分配）：



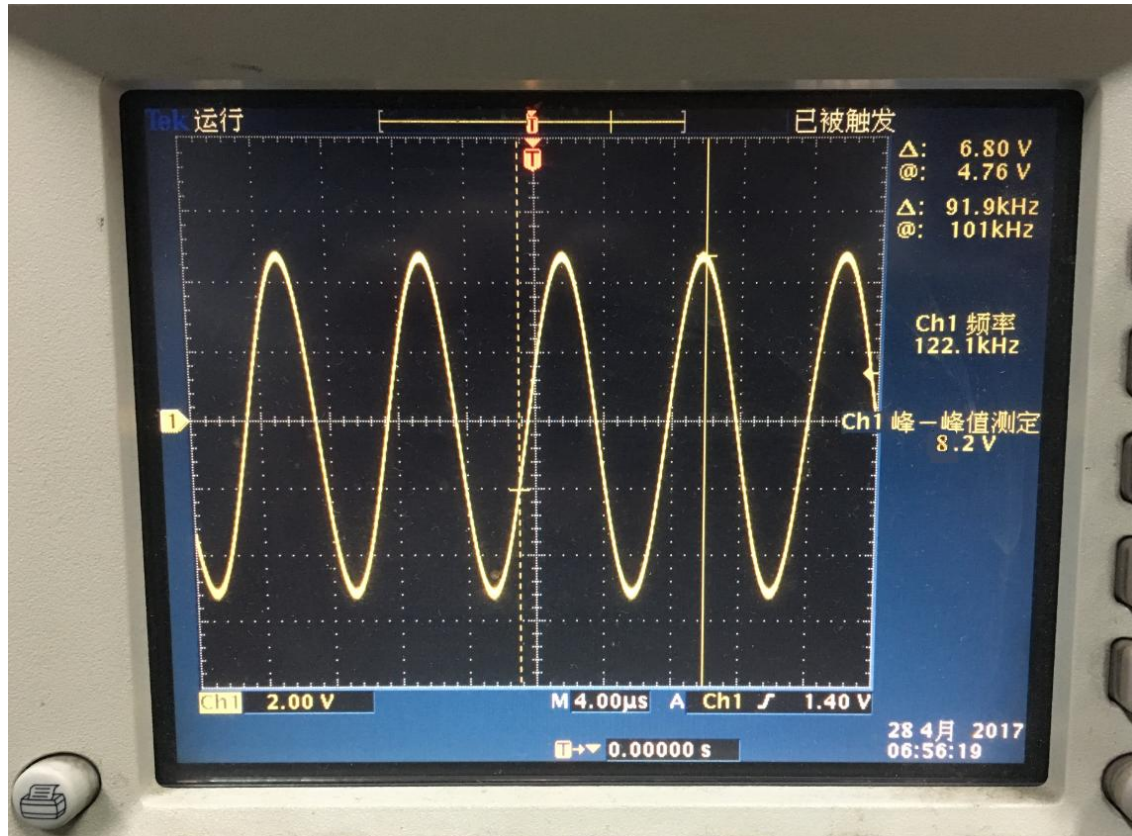
然后用我们提供的 BNC 线连接 AN9767 模块的模拟输出（通道 1 和通道 2 都可以）到示波器显示器上。



开发板上电，然后下载程序就可以从示波器上观察从 DA 模块输出的模拟信号的波形了。

## 第五部分 波形实验

下载 sof 文件 ( Xilinx 开发板是下载 bit 文件 ) 到 FPGA , 如果 AN9767 模块已经用 BNC 线连接到示波器的话, 我们就可以在示波器上看到正选波了。



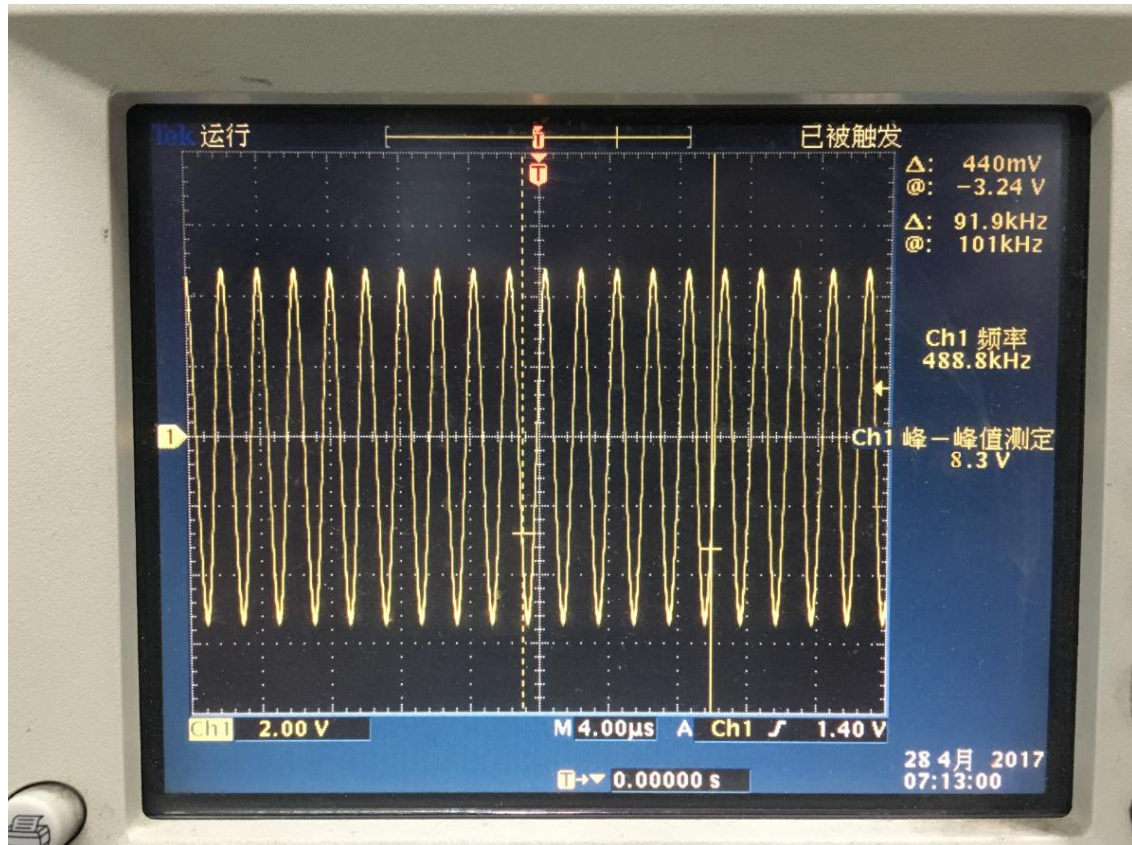
如果我们把程序中的地址修改成+4 的方式, 这样一个正选波的输出的点为 256 个, 输出的正选波的频率会提高 4 倍。

```

35  always @(negedge clk_125)
36  begin
37      // rom_addr <= rom_addr + 1'b1 ; //
38      rom_addr <= rom_addr + 4 ; //一个
39      // rom_addr <= rom_addr + 128 ; //一
40
41
42
43

```

程序修改后, 重新下载 FPGA 后, 正选波的频率变高, 示波器显示的波形如下:



用户也可以通过调节 AN9767 模块上的可调电阻来改变 2 个通道输出波形的幅度。

