

## 原创性声明

本人郑重声明：所呈交的论文，是本人在导师的指导下，读研期间独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名： 李永杰

日期： 2008-05-04

## 数字信号处理的FPGA实现

学 校：           华侨大学          

学 院：           信息科学与工程学院          

专 业：           信号与信息处理专业          

年 级：           2007级          

姓 名：           李永杰          

地 址：           福建省泉州市华侨大学信息学院2007级          

          信号与信息处理专业          

邮 编：           362021          

指导教师：           凌朝东 副教授          

2008年5月4日

# 数字信号处理的FPGA实现

**摘要:** 文章简要介绍了两种常见的数字信号处理的FPGA实现方式，频率合成器、AM和FM调制，并通过仿真和配置验证。

**关键词:** DSP    FPGA IMPLEMENTATION

## DSP With FPGA

Abstract: This paper briefly illustrates the implementation of two familiar digit signal process with FPGA, frequency synthesis and AM/FM modulated radio signal transmitter, also completes emulation and download verification.

Key words: DSP    FPGA Implementation

# 目 录

引言	.....
第1章 相关概念概述	.....
1.1 数字频率合成器	.....
1.2 AM、FM发射机	.....
第2章 DSP的FPGA实现与验证	.....
2.1 数字频率合	.....
2.2 AM/FM调制信号	.....
第3章 结论	.....

# 引言

数字信号处理技术已经成功运用于信号地滤波、语音、图像、音频、信息系统、控制和仪表设备。可编程数字信号处理器在 20 世纪 70 年代地引入更是使 DSP 技术突飞猛进，取得巨大成功，这些 PDSP 都是基于精简指令集（RISC）计算机范例的架构。它的优势源于大多说信号处理算法的乘—累加运算（MAC）都是非常密集的。通过多级流水线架构，PDSP 可以获得仅受阵列乘法器的速度限制的 MAC 速度。由此可以认为 FPGA 也能够用来实现 MAC 单元，且具有速度优势，但是，如果 PDSP 能够满足所需要的 MAC 速度，那么 PDSP 在成本问题上更具有成本优势，但随着 FPGA 成本降低，这个优势正在缩小。另一方面，现在我们还发现了许多高带宽的信号处理应用领域，例如：无线电、多媒体或卫星通信，FPGA 技术可以通过一个芯片上的多级 MAC 单元来提供更多的带宽。此外，在诸如 CORDIC、NTT 和差错校正算法等算法中，FPGA 较 PDSP 更有效率优势。

FPGA 技术的关键就是利用强有力的设计工具以：

- ▲ 缩短开发周期
- ▲ 提高器件资源利用率
- ▲ 提供综合器的选择，例如：在最佳速度和设计规模之间做出选择

## 第一章 相关概念概述

### 1. 频率合成器

直接数字频率合成(DDS—Digital Direct Frequency Synthesis)技术是一种新的频率合成方法，是频率合成技术的一次革命，JOSEPH TIERNEY 等 3 人于 1971 年提出了直接数字频率合成的思想，但由于受当时微电子技术和数字信号处理技术的限制，DDS 技术没有受到足够重视，随着电子工程领域的实际需要以及数字集成电路和微电子技术的发展，DDS 技术日益显露出它的优越性。

DDS 是一种全数字化的频率合成器，由相位累加器、波形 ROM、D/A 转换器和低通滤波器构成。时钟频率给定后，输出信号的频率取决于频率控制字，频率分辨率取决于累加器位数，相位分辨率取决于 ROM 的地址线位数，幅度量化噪声取决于 ROM 的数据位字长和 D/A 转换器位数。

DDS 有如下优点：(1) 频率分辨率高，输出频点多，可达 N 个频点(N 为相位累加器位数)；(2) 频率切换速度快，可达 us 量级；(3) 频率切换时相位连续；(4) 可以输出宽带正交信号；(5) 输出相位噪声低，对参考频率源的相位噪声有改善作用；(6) 可以产生任意波形；(7) 全数字化实现，便于集成，体积小，重量轻，因此八十年代以来各国都在研制和发展各自的 DDS 产品，如美国 QUALCOMM 公司的 Q2334, Q2220；STANFORD 公司的 STEL-1175, STEL-1180；AD 公司的 AD7008, AD9850, AD9854 等。这些 DDS 芯片的时钟频率从几十兆赫兹到几百兆赫兹不等，芯片从一般功能到集成有 D/A 转换器和正交调制器。

### 2. AM/FM

使载波振幅按照调制信号改变的调制方式叫调幅。经过调幅的电波叫调幅波。它保持着高频载波的频率特性，但包络线的形状则和信号波形相似。调幅波的振幅大小，由调制信号的强度决定。调幅波用英文字母 AM 表示。调幅波输出  $y(t) = (C + a(t)) * \text{sine}(2 * \pi * F * t)$ ；设  $F = 1\text{MHz}$ ，C 为常数，a(t) 为调制信号， $\text{sine}(2 * \pi * F * t)$  为载波，且两者都用直接频率合成技术实现。

使载波频率按照调制信号改变的调制方式叫调频。已调波频率变化的大小由调制信号的大小决定，变化的周期由调制信号的频率决定。已调波的振幅保持不变。调频波的波形，就像是个被压缩得不均匀的弹簧，调频波用英文字母 FM 表示。 $y(t) = C * \sin((2 * \pi * F + \delta f * a(t)) * t)$ ，其中 C 为常数， $F = 4.5\text{MHz}$ ，a(t) 为调制信号， $\delta f$  比例常数。

## 第二章 DSP 的 FPGA 实现与验证

1. 本节的设计的是一个直接数字频率合成器，包括一个 32 位的累加器 add1 和 lpm\_ff0，有 8 个最高有效位 acc[31..24]，连在一个 rom1 的查阅表上，从而产生所需要的输出波形。利用 ALTERA 的 Quartus II 软件的图形化解决方案如图 2-1 所示：

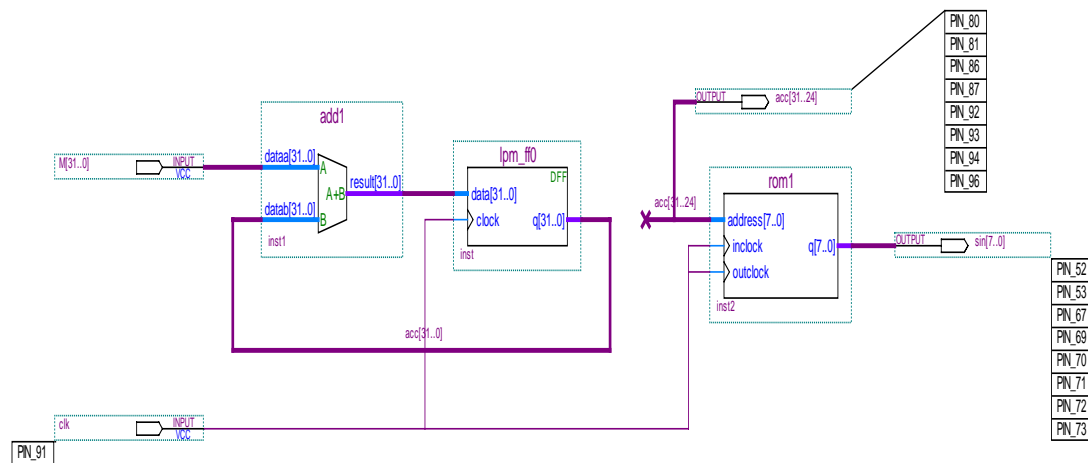


图 2-1 频率合成器的图形化设计

各个模块都是采用定制宏模块的方法，使用 MegaWizard Plug-In Manager 工具完成只需按需要设置各个模块参数即可。顶层文件对应的 HDL 描述为：(VHDL)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY work;

ENTITY fun_text IS
    port
    (
        clk : IN STD_LOGIC;
        M : IN STD_LOGIC_VECTOR(31 downto 0);
        acc : OUT STD_LOGIC_VECTOR(31 downto 24);
        sin : OUT STD_LOGIC_VECTOR(7 downto 0)
    );
END fun_text;

ARCHITECTURE bdf_type OF fun_text IS

    component lpm_ff0
        PORT(clock : IN STD_LOGIC;
            data : IN STD_LOGIC_VECTOR(31 downto 0);
            q : OUT STD_LOGIC_VECTOR(31 downto 0)
        );
    end component;


```

```

component add1
    PORT(dataaa : IN STD_LOGIC_VECTOR(31 downto 0);
          datab : IN STD_LOGIC_VECTOR(31 downto 0);
          result : OUT STD_LOGIC_VECTOR(31 downto 0)
    );
end component;

component rom1
    PORT(inclock : IN STD_LOGIC;
          outclock : IN STD_LOGIC;
          address : IN STD_LOGIC_VECTOR(7 downto 0);
          q : OUT STD_LOGIC_VECTOR(7 downto 0)
    );
end component;

signal acc_ALTERA_SYNTHESIZED : STD_LOGIC_VECTOR(31 downto 0);
signal SYNTHESIZED_WIRE_0 : STD_LOGIC_VECTOR(31 downto 0);

BEGIN

b2v_inst : lpm_ff0
PORT MAP(clock => clk,
          data => SYNTHESIZED_WIRE_0,
          q => acc_ALTERA_SYNTHESIZED);

b2v_inst1 : add1
PORT MAP(dataaa => M,
          datab => acc_ALTERA_SYNTHESIZED,
          result => SYNTHESIZED_WIRE_0);

b2v_inst2 : rom1
PORT MAP(inclock => clk,
          outclock => clk,
          address => acc_ALTERA_SYNTHESIZED(31 downto 24),
          q => sin);
acc(31 downto 24) <= acc_ALTERA_SYNTHESIZED(31 downto 24);

END;

```

工程设置：目标芯片选择开发板芯片 EP2C5T144C8, 编译进程设置见图 2-2、2-3

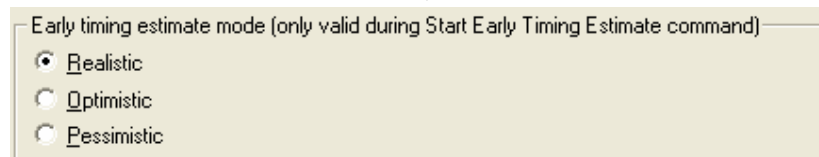


图 2-2 early timing estimate mode

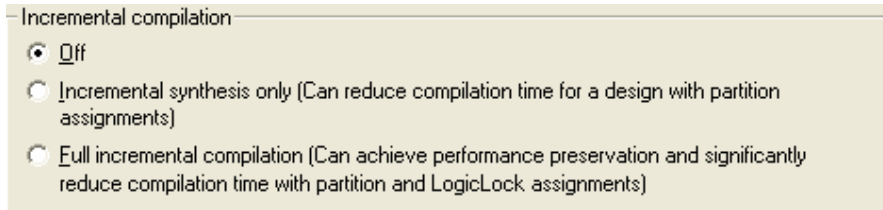


图 2-3 incremental compilation

分析与综合设置见图 2-4, 其余保持默认。

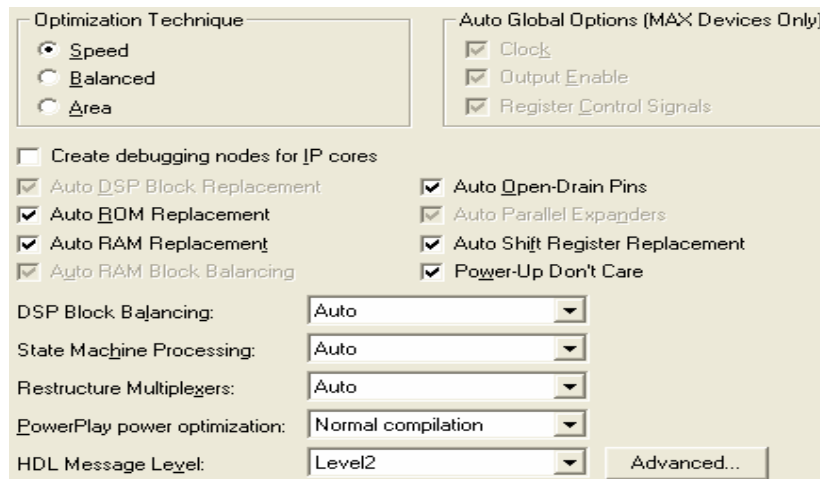


图 2-4 analysis & synthesis setting

编译资源使用情况见图 2-2:

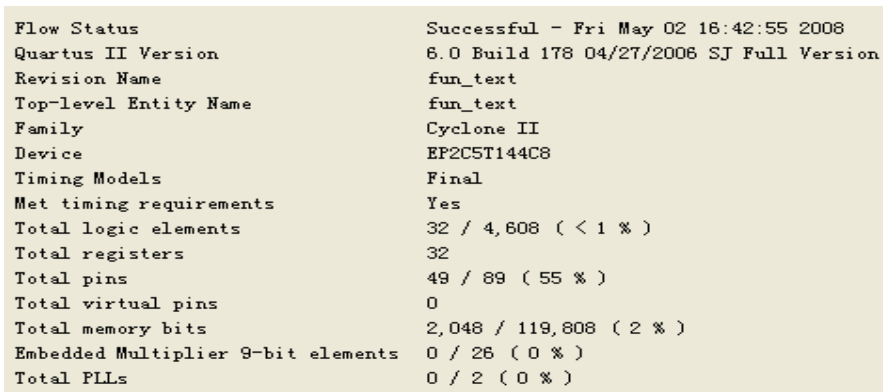


图 2-5 编译报告

时序仿真: 结束时间设为 1us, 累加器初值设为  $M=715827883 (2^{32}/6)$ , 这样合成器的周期就是 6 个时钟周期长, 结果如图 2-6 所示:

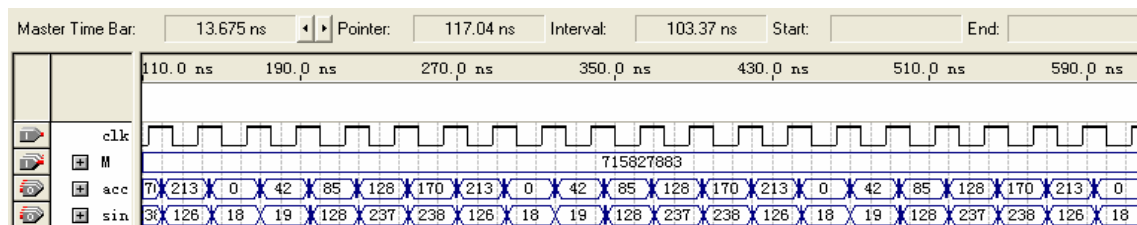


图 2-6 时序仿真图

性能分析如图 2-7 所示:

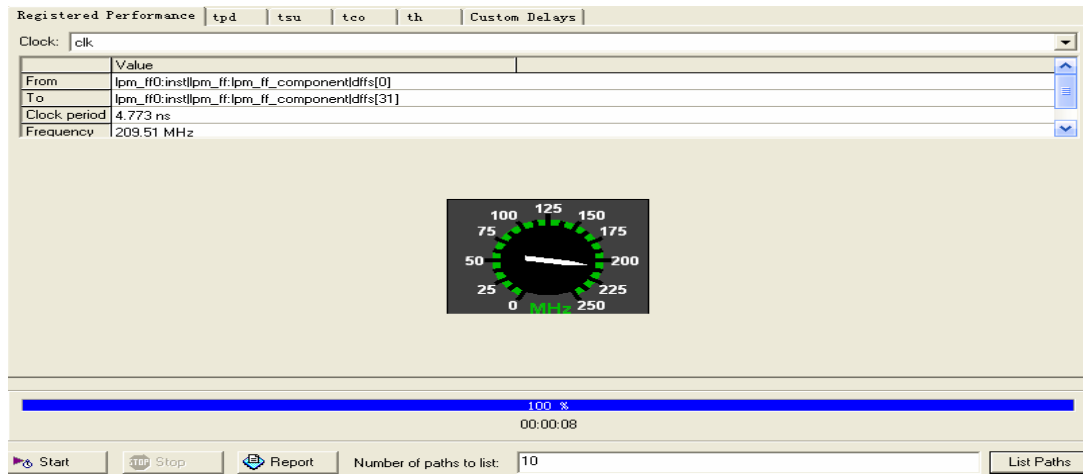


图 2-7 频率合成器设计的寄存器性能

下载并验证设置  $M=2^{32}/4$ ,  $CLK=60KHz$ , 经过 DA 转换后输出为  $60K/4=15 KHz$ 。如图 2-8 所示:

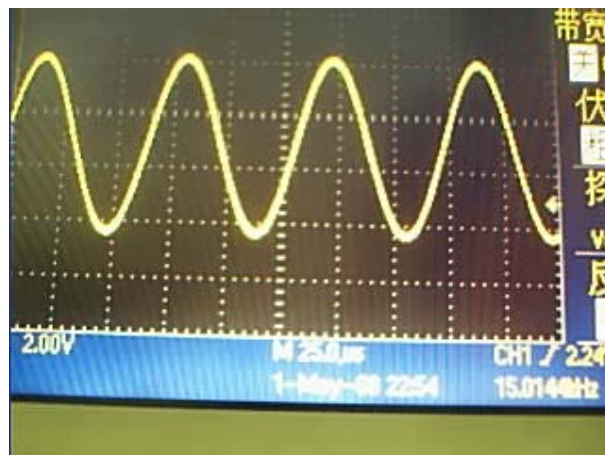


图 2-8 实际输出信号

## 2. AM/FM 调制信号发生器

AM 信号的产生是把调制信号, 和载波信号全部用频率合成技术产生, 然后做乘运算。

调制信号与载波信号的地址产生是用累加器, 且频率分别设置为 400Hz 和 1MHz, 对应的累加器初始值分别为 0xh85E7 和 0xh51EB84C。

```
always @ (posedge CLOCK_50) begin
    DDS_accum <= DDS_accum + 32'h51EB84C ;
    DDS_accum_mod = DDS_accum_mod + 32'h85E7 ;
end

//载波 LUT
sync_rom sineTable(CLOCK_50, DDS_accum[31:24], sine_out);

//调制信号 LUT
sync_rom modTable(CLOCK_50, DDS_accum_mod[31:24], mod_out);
```

//ROM 查找表

```
module sync_rom (clock, address, sine);
    input clock;
    input [7:0] address;
    output [9:0] sine;
    reg [9:0] sine;
    always @ (posedge clock)
    begin
        case (address)
            8'h00: sine = 10'h100 ;8'h01: sine = 10'h106 ;8'h02: sine = 10'h10c ;
            8'h03: sine = 10'h112 ;8'h04: sine = 10'h118 ;8'h05: sine = 10'h11f ;
            8'h06: sine = 10'h125 ;8'h07: sine = 10'h12b ;8'h08: sine = 10'h131 ;
            8'h09: sine = 10'h137 ;8'h0a: sine = 10'h13d ;8'h0b: sine = 10'h144 ;
            8'h0c: sine = 10'h14a ;8'h0d: sine = 10'h14f ;8'h0e: sine = 10'h155 ;
            8'h0f: sine = 10'h15b ;8'h10: sine = 10'h161 ;8'h11: sine = 10'h167 ;
            8'h12: sine = 10'h16d ;8'h13: sine = 10'h172 ;8'h14: sine = 10'h178 ;
            8'h15: sine = 10'h17d ;8'h16: sine = 10'h183 ;8'h17: sine = 10'h188 ;
            8'h18: sine = 10'h18d ;8'h19: sine = 10'h192 ;8'h1a: sine = 10'h197 ;
            8'h1b: sine = 10'h19c ;8'h1c: sine = 10'h1a1 ;8'h1d: sine = 10'h1a6 ;
            8'h1e: sine = 10'h1ab ;8'h1f: sine = 10'h1af ;8'h20: sine = 10'h1b4 ;
            8'h21: sine = 10'h1b8 ;8'h22: sine = 10'h1bc ;8'h23: sine = 10'h1c1 ;
            8'h24: sine = 10'h1c5 ;8'h25: sine = 10'h1c9 ;8'h26: sine = 10'h1cc ;
            8'h27: sine = 10'h1d0 ;8'h28: sine = 10'h1d4 ;8'h29: sine = 10'h1d7 ;
            8'h2a: sine = 10'h1da ;8'h2b: sine = 10'h1dd ;8'h2c: sine = 10'h1e0 ;
            8'h2d: sine = 10'h1e3 ;8'h2e: sine = 10'h1e6 ;8'h2f: sine = 10'h1e9 ;
            8'h30: sine = 10'h1eb ;8'h31: sine = 10'h1ed ;8'h32: sine = 10'h1f0 ;
            8'h33: sine = 10'h1f2 ;8'h34: sine = 10'h1f4 ;8'h35: sine = 10'h1f5 ;
            8'h36: sine = 10'h1f7 ;8'h37: sine = 10'h1f8 ;8'h38: sine = 10'h1fa ;
            8'h39: sine = 10'h1fb ;8'h3a: sine = 10'h1fc ;8'h3b: sine = 10'h1fd ;
            8'h3c: sine = 10'h1fd ;8'h3d: sine = 10'h1fe ;8'h3e: sine = 10'h1fe ;
            8'h3f: sine = 10'h1fe ;8'h40: sine = 10'h1ff ;8'h41: sine = 10'h1fe ;
            8'h42: sine = 10'h1fe ;8'h43: sine = 10'h1fe ;8'h44: sine = 10'h1fd ;
            8'h45: sine = 10'h1fd ;8'h46: sine = 10'h1fc ;8'h47: sine = 10'h1fb ;
            8'h48: sine = 10'h1fa ;8'h49: sine = 10'h1f8 ;8'h4a: sine = 10'h1f7 ;
            8'h4b: sine = 10'h1f5 ;8'h4c: sine = 10'h1f4 ;8'h4d: sine = 10'h1f2 ;
            8'h4e: sine = 10'h1f0 ;8'h4f: sine = 10'h1ed ;8'h50: sine = 10'h1eb ;
            8'h51: sine = 10'h1e9 ;8'h52: sine = 10'h1e6 ;8'h53: sine = 10'h1e3 ;
            8'h54: sine = 10'h1e0 ;8'h55: sine = 10'h1dd ;8'h56: sine = 10'h1da ;
            8'h57: sine = 10'h1d7 ;8'h58: sine = 10'h1d4 ;8'h59: sine = 10'h1d0 ;
            8'h5a: sine = 10'h1cc ;8'h5b: sine = 10'h1c9 ;8'h5c: sine = 10'h1c5 ;
            8'h5d: sine = 10'h1c1 ;8'h5e: sine = 10'h1bc ;8'h5f: sine = 10'h1b8 ;
            8'h60: sine = 10'h1b4 ;8'h61: sine = 10'h1af ;8'h62: sine = 10'h1ab ;
            8'h63: sine = 10'h1a6 ;8'h64: sine = 10'h1a1 ;8'h65: sine = 10'h19c ;
            8'h66: sine = 10'h197 ;8'h67: sine = 10'h192 ;8'h68: sine = 10'h18d ;
```

8' h69: sine = 10' h188 ;8' h6a: sine = 10' h183 ;8' h6b: sine = 10' h17d ;  
8' h6c: sine = 10' h178 ;8' h6d: sine = 10' h172 ;8' h6e: sine = 10' h16d ;  
8' h6f: sine = 10' h167 ;8' h70: sine = 10' h161 ;8' h71: sine = 10' h15b ;  
8' h72: sine = 10' h155 ;8' h73: sine = 10' h14f ;8' h74: sine = 10' h14a ;  
8' h75: sine = 10' h144 ;8' h76: sine = 10' h13d ;8' h77: sine = 10' h137 ;  
8' h78: sine = 10' h131 ;8' h79: sine = 10' h12b ;8' h7a: sine = 10' h125 ;  
8' h7b: sine = 10' h11f ;8' h7c: sine = 10' h118 ;8' h7d: sine = 10' h112 ;  
8' h7e: sine = 10' h10c ;8' h7f: sine = 10' h106 ;8' h80: sine = 10' h100 ;  
8' h81: sine = 10' h0f9 ;8' h82: sine = 10' h0f3 ;8' h83: sine = 10' h0ed ;  
8' h84: sine = 10' h0e7 ;8' h85: sine = 10' h0e0 ;8' h86: sine = 10' h0da ;  
8' h87: sine = 10' h0d4 ;8' h88: sine = 10' h0ce ;8' h89: sine = 10' h0c8 ;  
8' h8a: sine = 10' h0c2 ;8' h8b: sine = 10' h0bb ;8' h8c: sine = 10' h0b5 ;  
8' h8d: sine = 10' h0b0 ;8' h8e: sine = 10' h0aa ;8' h8f: sine = 10' h0a4 ;  
8' h90: sine = 10' h09e ;8' h91: sine = 10' h098 ;8' h92: sine = 10' h092 ;  
8' h93: sine = 10' h08d ;8' h94: sine = 10' h087 ;8' h95: sine = 10' h082 ;  
8' h96: sine = 10' h07c ;8' h97: sine = 10' h077 ;8' h98: sine = 10' h072 ;  
8' h99: sine = 10' h06d ;8' h9a: sine = 10' h068 ;8' h9b: sine = 10' h063 ;  
8' h9c: sine = 10' h05e ;8' h9d: sine = 10' h059 ;8' h9e: sine = 10' h054 ;  
8' h9f: sine = 10' h050 ;8' ha0: sine = 10' h04b ;8' ha1: sine = 10' h047 ;  
8' ha2: sine = 10' h043 ;8' ha3: sine = 10' h03e ;8' ha4: sine = 10' h03a ;  
8' ha5: sine = 10' h036 ;8' ha6: sine = 10' h033 ;8' ha7: sine = 10' h02f ;  
8' ha8: sine = 10' h02b ;8' ha9: sine = 10' h028 ;8' haa: sine = 10' h025 ;  
8' hab: sine = 10' h022 ;8' hac: sine = 10' h01f ;8' had: sine = 10' h01c ;  
8' hae: sine = 10' h019 ;8' haf: sine = 10' h016 ;8' hb0: sine = 10' h014 ;  
8' hb1: sine = 10' h012 ;8' hb2: sine = 10' h00f ;8' hb3: sine = 10' h00d ;  
8' hb4: sine = 10' h00b ;8' hb5: sine = 10' h00a ;8' hb6: sine = 10' h008 ;  
8' hb7: sine = 10' h007 ;8' hb8: sine = 10' h005 ;8' hb9: sine = 10' h004 ;  
8' hba: sine = 10' h003 ;8' hbb: sine = 10' h002 ;8' hbc: sine = 10' h002 ;  
8' hbd: sine = 10' h001 ;8' hbe: sine = 10' h001 ;8' hbf: sine = 10' h001 ;  
8' hc0: sine = 10' h001 ;8' hc1: sine = 10' h001 ;8' hc2: sine = 10' h001 ;  
8' hc3: sine = 10' h001 ;8' hc4: sine = 10' h002 ;8' hc5: sine = 10' h002 ;  
8' hc6: sine = 10' h003 ;8' hc7: sine = 10' h004 ;8' hc8: sine = 10' h005 ;  
8' hc9: sine = 10' h007 ;8' hca: sine = 10' h008 ;8' hcb: sine = 10' h00a ;  
8' hcc: sine = 10' h00b ;8' hcd: sine = 10' h00d ;8' hce: sine = 10' h00f ;  
8' hcf: sine = 10' h012 ;8' hd0: sine = 10' h014 ;8' hd1: sine = 10' h016 ;  
8' hd2: sine = 10' h019 ;8' hd3: sine = 10' h01c ;8' hd4: sine = 10' h01f ;  
8' hd5: sine = 10' h022 ;8' hd6: sine = 10' h025 ;8' hd7: sine = 10' h028 ;  
8' hd8: sine = 10' h02b ;8' hd9: sine = 10' h02f ;8' hda: sine = 10' h033 ;  
8' hdb: sine = 10' h036 ;8' hdc: sine = 10' h03a ;8' hdd: sine = 10' h03e ;  
8' hde: sine = 10' h043 ;8' hdf: sine = 10' h047 ;8' he0: sine = 10' h04b ;  
8' he1: sine = 10' h050 ;8' he2: sine = 10' h054 ;8' he3: sine = 10' h059 ;  
8' he4: sine = 10' h05e ;8' he5: sine = 10' h063 ;8' he6: sine = 10' h068 ;  
8' he7: sine = 10' h06d ;8' he8: sine = 10' h072 ;8' he9: sine = 10' h077 ;  
8' hea: sine = 10' h07c ;8' heb: sine = 10' h082 ;8' hec: sine = 10' h087 ;

```

8'hed: sine = 10'h08d ;8'hee: sine = 10'h092 ;8'hef: sine = 10'h098 ;
8'hf0: sine = 10'h09e ;8'hf1: sine = 10'h0a4 ;8'hf2: sine = 10'h0aa ;
8'hf3: sine = 10'h0b0 ;8'hf4: sine = 10'h0b5 ;8'hf5: sine = 10'h0bb ;
8'hf6: sine = 10'h0c2 ;8'hf7: sine = 10'h0c8 ;8'hf8: sine = 10'h0ce ;
8'hf9: sine = 10'h0d4 ;8'hfa: sine = 10'h0da ;8'hfb: sine = 10'h0e0 ;
8'hfc: sine = 10'h0e7 ;8'hfd: sine = 10'h0ed ;8'hfe: sine = 10'h0f3 ;
8'hff: sine = 10'h0f9 ;
endcase
end
endmodule

```

//AM 信号

```
assign VGA_R = (sine_out * ((mod_out+256)<<1) )>>10 ; //works well
```

```
assign VGA_SYNC = 1 ;
```

```
assign VGA_BLANK = 1 ;
```

```
assign VGA_CLK = CLOCK_50 ;
```

下载后测试输出，AM 信号为 999.99KHz 符合要求：如图 2-9 所示：

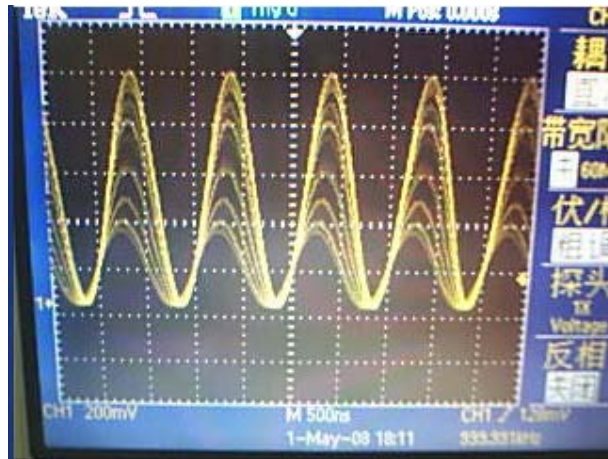


图 2-9 AM 信号的示波器显示

FM 信号产生原理同 AM 信号，此时载波信号为

```
DDS_accum <= DDS_accum + 32'h16FC7BBC + mod_out<<15
```

调制信号为  $DDS\_accum\_mod = DDS\_accum\_mod + ((SW[0])? 32'hF5E7 : 32'h85E7) ;$

载波使用的是方波，也是基于 LUT

// 载波方波查找表

```
module sqwave (clock, address, sq);
```

```
input clock;
```

```
input [7:0] address;
```

```
output [9:0] sq;
```

```
reg [9:0] sq;
```

```
always @ (posedge clock)
```

```
begin
```

```
sq <= (address<128)? 10'h1ff : 10'h001 ;
```

```
end
endmodule
```

载波对应的查找表语句为: `sqwave sqTable(CLOCK_50, DDS_accum[31:24], sq_out);`  
调制信号对应的查找表为: `sync_rom modTable(CLOCK_50, DDS_accum_mod[31:24], mod_out);`且  
ROM 中为正弦波表同 AM 信号。频率设为 800Hz。  
编译报告如图 2-10 所示:

Flow Status	Successful - Tue Oct 24 11:49:43 2006
Quartus II Version	6.0 Build 202 06/20/2006 SP 1 SJ Web Edition
Revision Name	DE2_Default
Top-level Entity Name	DE2_Default
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	No
Total logic elements	279 / 33,216 ( < 1 % )
Total registers	204
Total pins	425 / 475 ( 89 % )
Total virtual pins	0
Total memory bits	2,560 / 483,840 ( < 1 % )
Embedded Multiplier 9-bit elements	2 / 70 ( 3 % )
Total PLLs	1 / 4 ( 25 % )

图 2-10 编译报告

硬件在 DE2 平台实现, 选用 VGA 的 DA 转换实现输出, 频率为 4.58MHz 左右, 如图 2-11 所示:

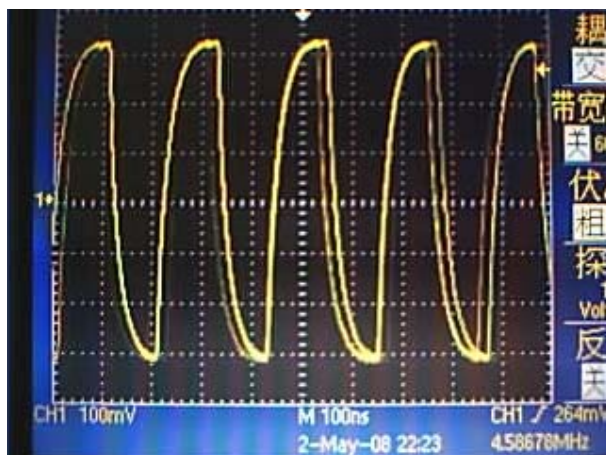


图 2-11 FM 示波器测量

### 第三章结论

本论文实现了数字频率合成器、AM、FM 等信号的 FPGA 实现, 并测试验证了结果, 很好的完成了预期设计, 当然实现不是最主要的, 主要是在想通过本论文传递一种信号, DSP 在 FPGA 上的实现将是完美的, FPGA 能提供传统 PDSP 无法在同种价格下实现的并行运算, 大大提高运算速度, 随着 FPGA 的成本进一步降低, 和高端 FPGA 的实现, DSP 必将与 FPGA 结合的更加紧密完美!

参考文献:

- [1] 刘凌. 数字信号处理FPGA实现. 北京:清华大学出版社, 2006
- [2] JO Hamblen, TS Hall and MD Furman, *Rapid prototyping of digital systems*, Springer 2005