第6章

LPM宏模块应用

6.1 LPM计数器模块调用

6.1.1 计数器模块文本的调用与参数设置

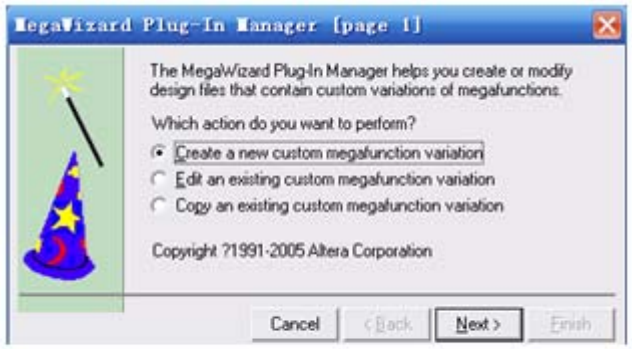

图 6-1　定制新的宏功能块

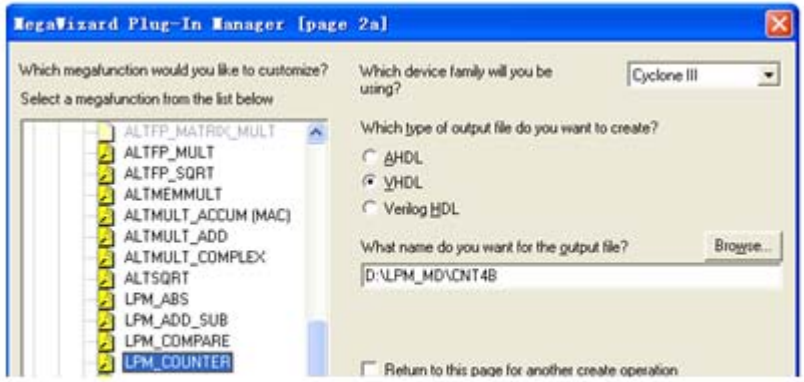

图 6-2　LPM 宏功能块设定

6.1 LPM计数器模块调用

6.1.1 计数器模块文本的调用与参数设置

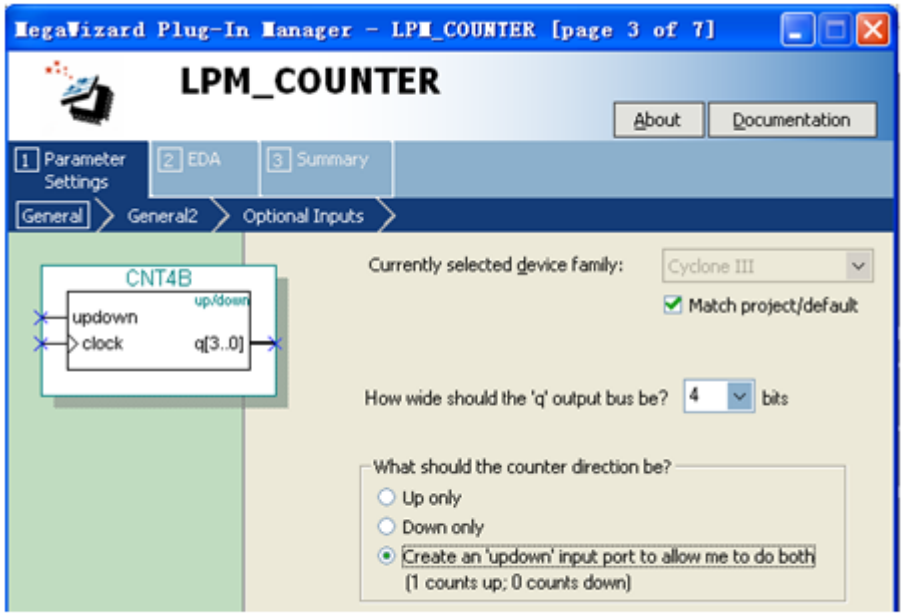

图 6-3 设 4 位可加减计数器

6.1 LPM计数器模块调用

6.1.1 计数器模块文本的调用与参数设置

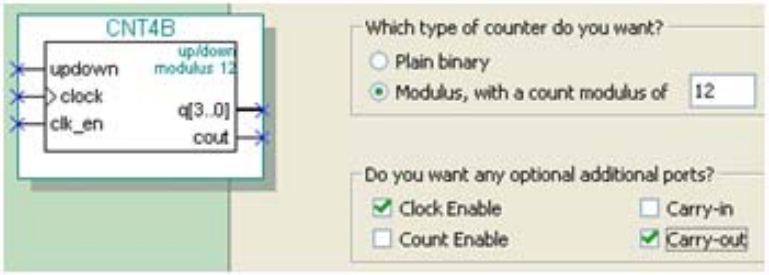

图 6-4 设定计数器，含时钟使能和进位输出

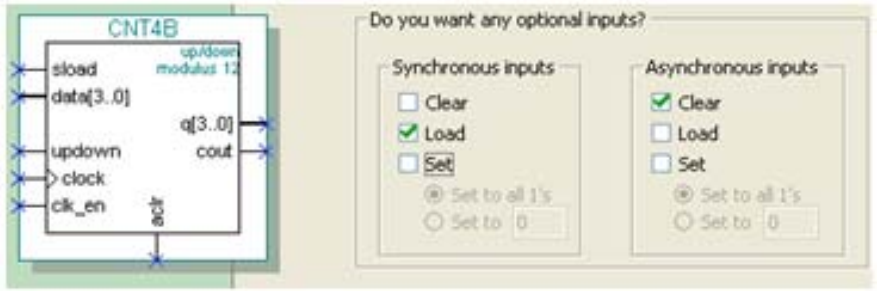

图 6-5 加入 4 位并行数据预置功能

【例 6-1】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY lpm;  --打开 LPM 库
USE lpm.all;  --打开 LPM 程序包
ENTITY CNT4B IS--异步清 0、时钟使能、时钟输入、同步预置数加载控制、加减控制
  PORT (aclr, clk_en, clock, sload, updown  : IN STD_LOGIC ;
        data : IN STD_LOGIC_VECTOR (3 DOWNTO 0);-- 4位预置数
       cout     : OUT STD_LOGIC ;                          --进位输出
       q        : OUT STD_LOGIC_VECTOR (3 DOWNTO 0) );--计数器输出
END CNT4B;
ARCHITECTURE SYN OF cnt4b IS
       SIGNAL sub_wire0 : STD_LOGIC ;
       SIGNAL sub_wire1 : STD_LOGIC_VECTOR (3 DOWNTO 0);
       COMPONENT lpm_counter                    --以下是参数传递说明语句
GENERIC(lpm_direction,lpm_port_updown ,lpm_type : STRING;--参数定义
                lpm_modulus, lpm_width : NATURAL );      --参数定义
PORT (sload, clk_en, aclr, clock, updown : IN STD_LOGIC ;
                                  cout     : OUT STD_LOGIC ;
                                  q : OUT STD_LOGIC_VECTOR (3 DOWNTO 0);
                           data     : IN STD_LOGIC_VECTOR (3 DOWNTO 0) );
       END COMPONENT;
BEGIN
cout <= sub_wire0;  q <= sub_wire1(3 DOWNTO 0);
lpm_counter_component : lpm_counter GENERIC MAP (  --参数传递例化语句
lpm_direction => "UNUSED",                         --单方向计数参数未用
                lpm_modulus => 12,                    --定义模 12 计数器
                lpm_port_updown => "PORT_USED",       --使用加减计数
                lpm_type => "LPM_COUNTER",            --计数器类型
                lpm_width => 4    )                   --计数位宽
PORT MAP (sload=>sload,clk_en=>clk_en,aclr=>aclr, clock => clock,
       data => data,updown => updown,cout=>sub_wire0,q => sub_wire1);
END SYN;
```

6.1 LPM计数器模块调用

6.1.1 计数器模块文本的调用与参数设置

【例 6-2】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY CNT4BIT IS
   PORT (CLK,RST,ENA ,SLD,UD : IN std_logic;
         DIN  : IN std_logic_vector(3 DOWNTO 0);  COUT : OUT std_logic;
         DOUT   : OUT std_logic_vector(3 DOWNTO 0));
END ENTITY CNT4BIT;
ARCHITECTURE translated OF CNT4BIT IS
    COMPONENT CNT4B
        PORT (aclr,clk_en,clock,sload,updown  : IN STD_LOGIC ;
                data     : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
                cout     : OUT STD_LOGIC ;
                q        : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END COMPONENT;
 BEGIN
    U1 : CNT4B PORT MAP (sload => SLD, clk_en => ENA,aclr => RST,
         cout=>COUT, clock=>CLK, data=>DIN, updown=>UD, q=>DOUT);
  END ARCHITECTURE translated;
```

6.1 LPM计数器模块调用

6.1.2 创建工程与仿真测试

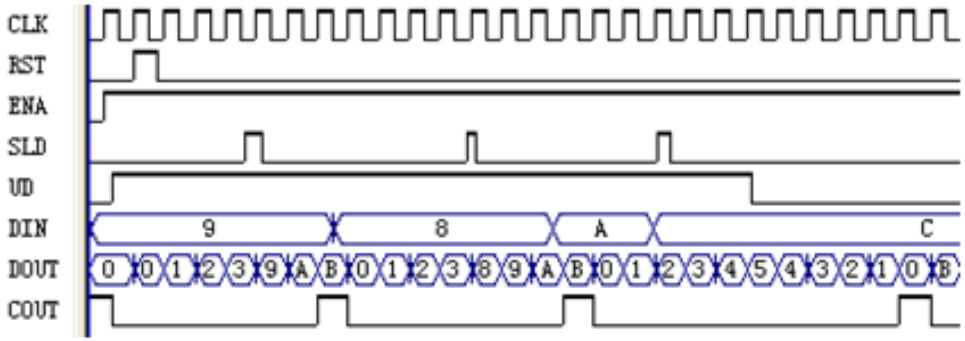

图 6-6 CNT4BIT 的仿真波形

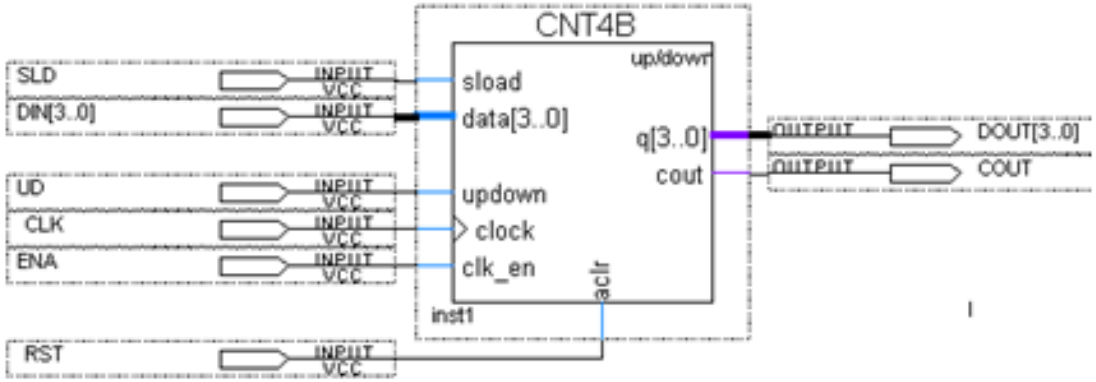

图 6-7 原理图输入设计

6.2 乘法器的VHDL代码表述和相关属性设置

【例 6-3】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
ENTITY MULT8 IS
PORT (A1,B1 : IN  SIGNED(7 DOWNTO 0) ;
      R1 : OUT SIGNED(15 DOWNTO 0));
END ;
ARCHITECTURE bhv OF MULT8 IS
 attribute multstyle : string;
 attribute multstyle of R1 : signal is "LOGIC";--使用逻辑资源构建乘法器
BEGIN
   R1 <= A1 * B1  ;
END bhv;
```

Family	Cyclone III
Device	EP3C55F484C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	0 / 55,856 (0 %)
Total combinational functions	0 / 55,856 (0 %)
Dedicated logic registers	0 / 55,856 (0 %)
Total registers	0
Total pins	32 / 328 (10 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	1 / 312 (< 1 %)
Total PLLs	0 / 4 (0 %)

图 6-9　例 6-3 的编译报告

图 6-8　例 6-3 的编译报告

【例 6-4】

```
attribute multstyle of R1 : signal is "DSP";--使用 DSP 模块构建乘法器
```

6.3 LPM 随机存储器的设置和调用

6.3.1 存储器初始化文件

1. .mif格式文件

（1）直接编辑法。

DATA7X8.mif

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	80	86	8C	92	98	9E	A5	AA
08	B0	B6	BC	C1	C6	CB	D0	D5
10	DA	DE	E2	E6	EA	ED	F0	F3
18	F5	F8	FA	FB	FD	FE	FE	FF
20	FF	FF	FE	FE	FD	FB	FA	F8
28	F5	F3	F0	ED	EA	E6	E2	DE
30	DA	D5	D0	CB	C6	C1	BC	B6
38	B0	AA	A5	9E	98	92	8C	86
40	7F	79	73	6D	67	61	5A	55
48	4F	49	43	3E	39	34	2F	2A
50	25	21	1D	19	15	12	0F	0C
58	0A	07	05	04	02	01	01	00
60	00	00	01	01	02	04	05	07
68	0A	0C	0F	12	15	19	1D	21
70	25	2A	2F	34	39	3E	43	49
78	4F	55	5A	61	67	6D	73	79

图 6-10　mif 文件编辑窗

6.3 LPM 随机存储器的设置和调用

6.3.1 存储器初始化文件

1. .mif格式文件

（2）文件直接编辑法。

【例 6-5】

```
DEPTH=128; 数据深度，即存储的数据个数
WIDTH=8;                  ：输出数据宽度
ADDRESS_RADIX = HEX;    ：地址数据类型，HEX 表示选择 16 进制数据类型
DATA_RADIX = HEX;         ：存储数据类型，HEX 表示选择 16 进制数据类型
CONTENT                   ：此为关键词
BEGIN                     ：此为关键词
0000      :      0080;
0001      :      0086;
0002      :      008C;
    …（数据略去）
007E      :     0073;
007F      :     0079;
END;
```

6.3 LPM 随机存储器的设置和调用

6.3.1 存储器初始化文件

（3）高级语言生成。

（4）专用mif文件生成器。

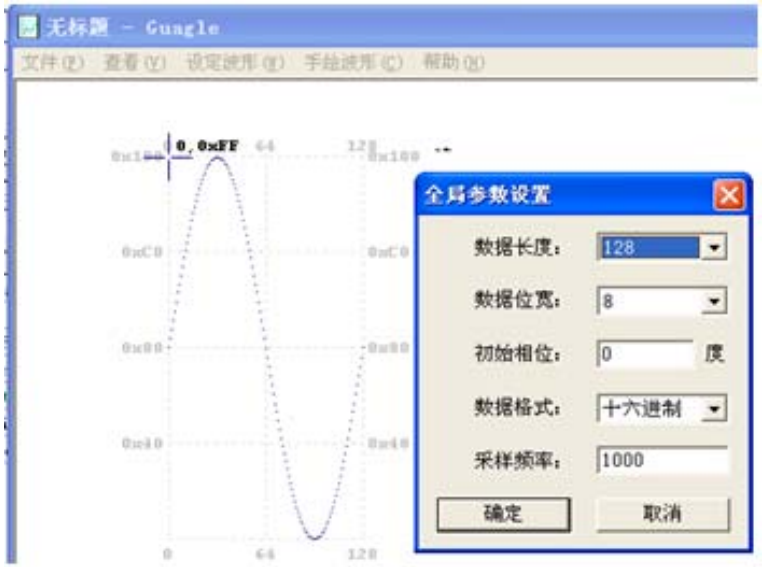

图 6-11 利用 mif 生成器生成 mif 正弦波文件

图 6-12 mif 文件

2. .hex格式文件

6.3 LPM 随机存储器的设置和调用

6.3.2 LPM_RAM的设置和调用

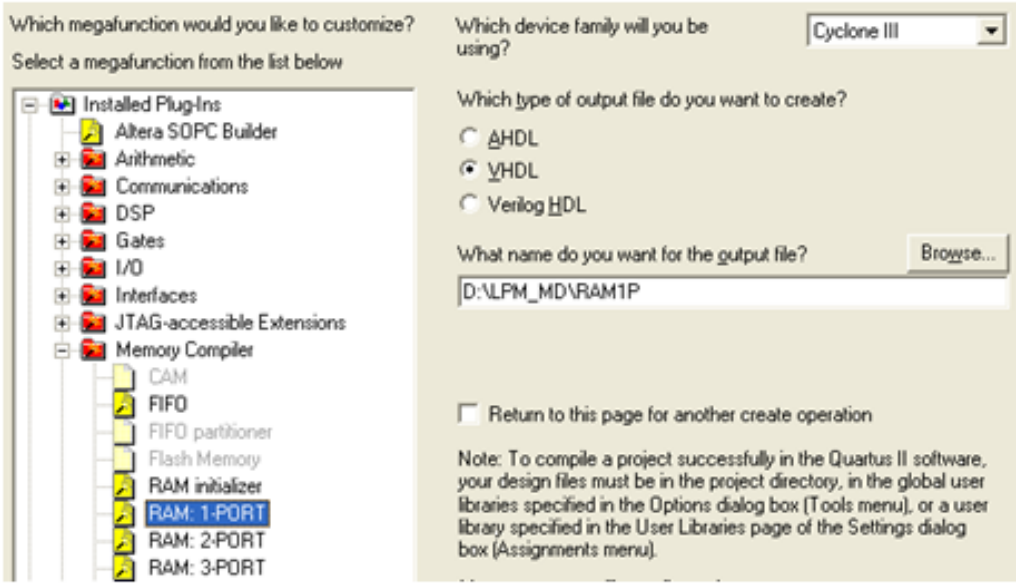

图 6-13　调用单口 LPM RAM

6.3 LPM 随机存储器的设置和调用

6.3.2 LPM_RAM的设置和调用

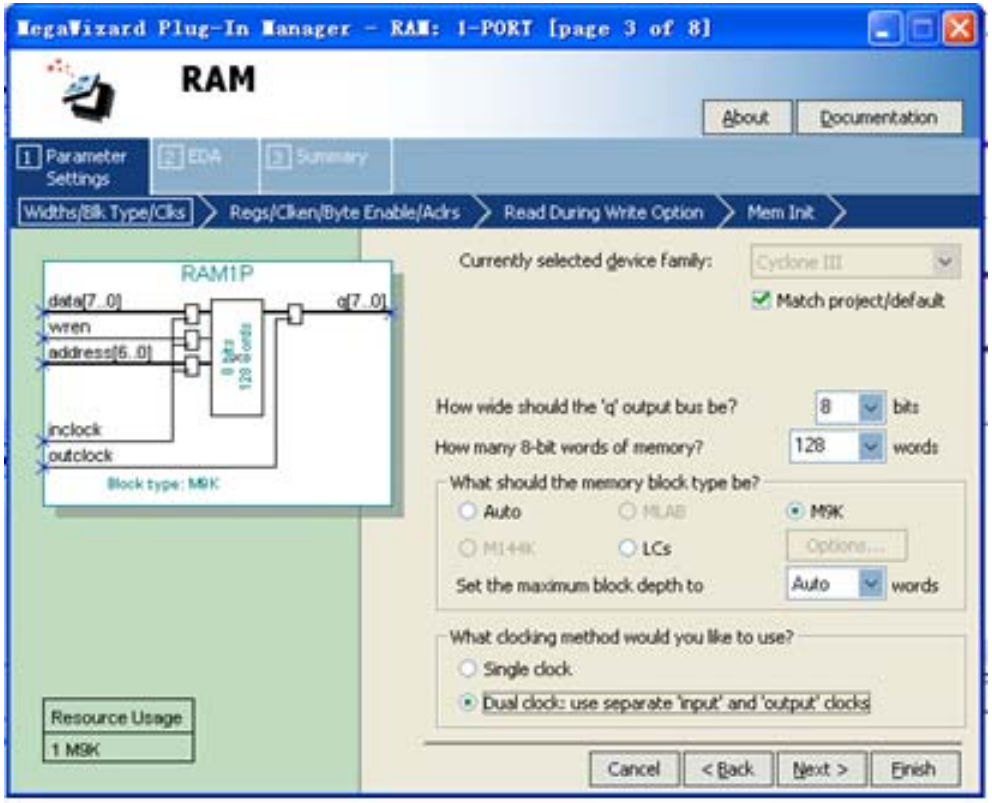

图 6-14 设定 RAM 参数

6.3 LPM 随机存储器的设置和调用

6.3.2 LPM_RAM的设置和调用

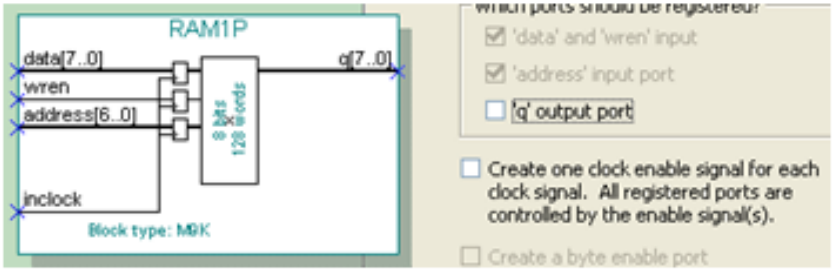

图 6-15 设定 RAM 仅输入时钟控制

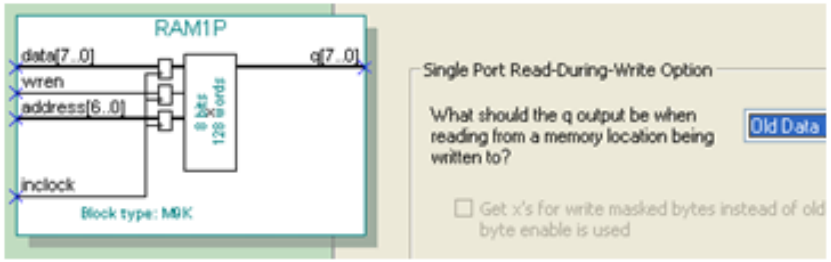

图 6-16 设定在写入同时读出原数据：Old Data

6.3 LPM 随机存储器的设置和调用

6.3.2 LPM_RAM的设置和调用

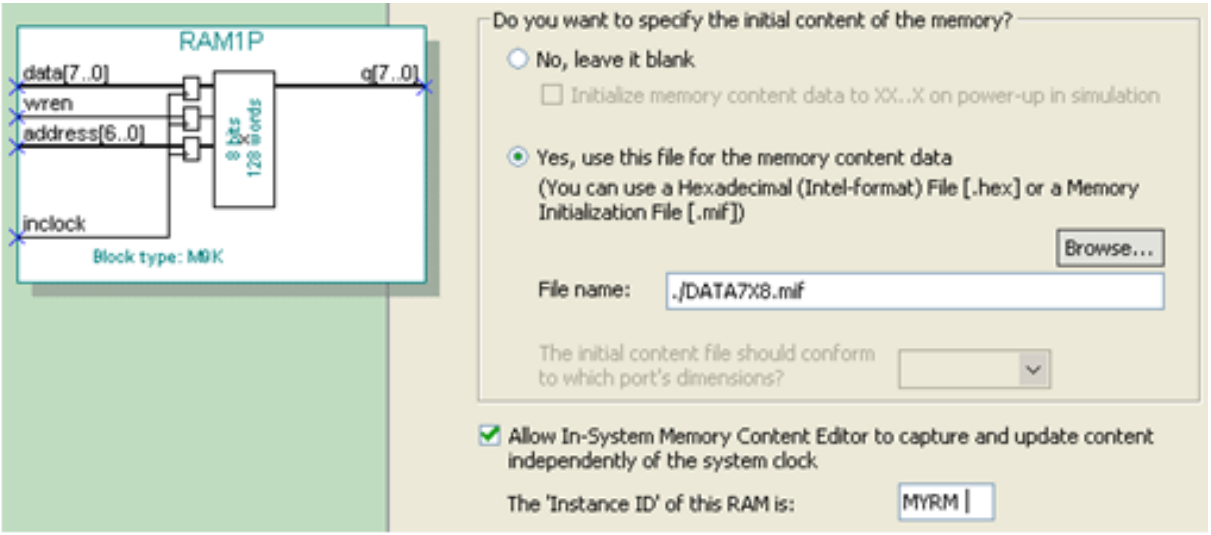

图 6-17 设定初始化文件和允许在系统编辑

6.3 LPM 随机存储器的设置和调用

6.3.2 LPM_RAM的设置和调用

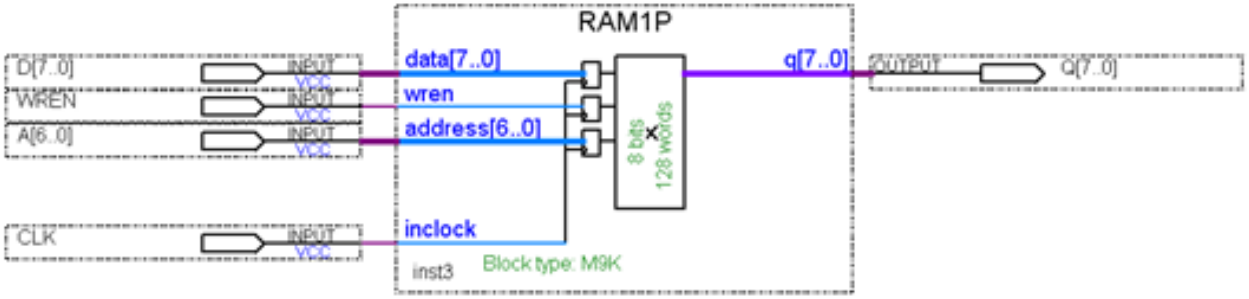

图 6-18 为了单独测试，在原理图上连接好 RAM 模块

6.3 LPM 随机存储器的设置和调用

6.3.3 仿真测试RAM宏模块

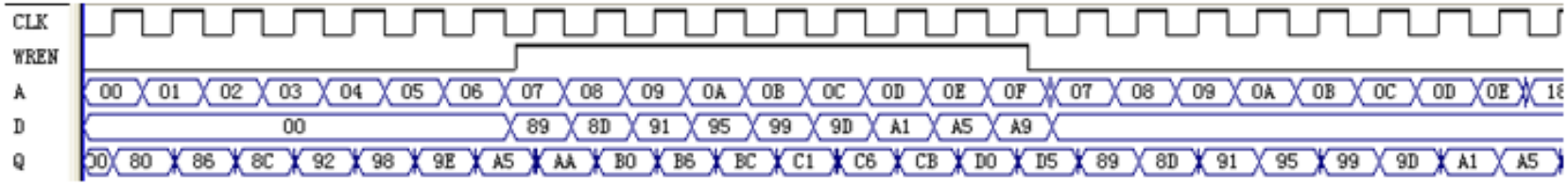

图 6-19 的 RAM 的仿真波形

6.3 LPM 随机存储器的设置和调用

6.3.4 存储器配置文件属性定义和结构设置

【例 6-6】

```
ARCHITECTURE bhv OF RAM78 IS
TYPE G_ARRAY IS ARRAY(0 TO 127) OF STD_LOGIC_VECTOR(7 DOWNTO 0) ;
SIGNAL MEM : G_ARRAY;
 attribute ram_init_file : string;
 attribute ram_init_file of MEM :
 SIGNAL IS  "data7x8.mif";
BEGIN
```

Family	Cyclone III
Device	EP3C5E144C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	1,344 / 5,136 (26 %)
Total combinational functions	832 / 5,136 (16 %)
Dedicated logic registers	1,032 / 5,136 (20 %)
Total registers	1032
Total pins	25 / 95 (26 %)
Total virtual pins	0
Total memory bits	0 / 423,936 (0 %)
Embedded Multiplier 9-bit elements	0 / 46 (0 %)
Total PLLs	0 / 2 (0 %)

图 6-20 例 5-10 的编译报告

Family	Cyclone III
Device	EP3C5E144C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	0 / 5,136 (0 %)
Total combinational functions	0 / 5,136 (0 %)
Dedicated logic registers	0 / 5,136 (0 %)
Total registers	0
Total pins	25 / 95 (26 %)
Total virtual pins	0
Total memory bits	1,024 / 423,936 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 46 (0 %)
Total PLLs	0 / 2 (0 %)

图 6-21 例 6-6 的编译报告

6.4 LPM_ROM的定制和使用示例

6.4.1 LPM_ROM的定制调用和测试

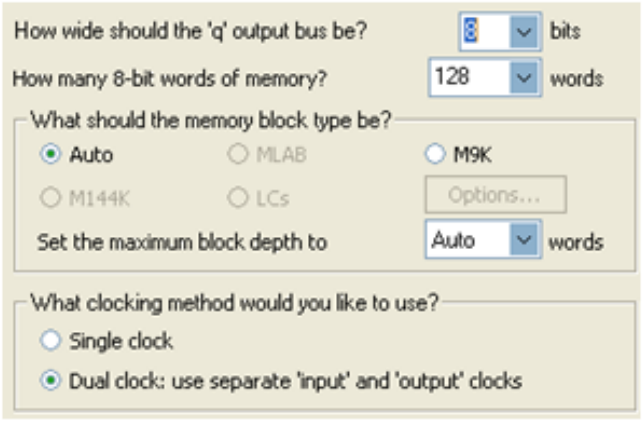

图 6-22 对 LPM_ROM 设置参数

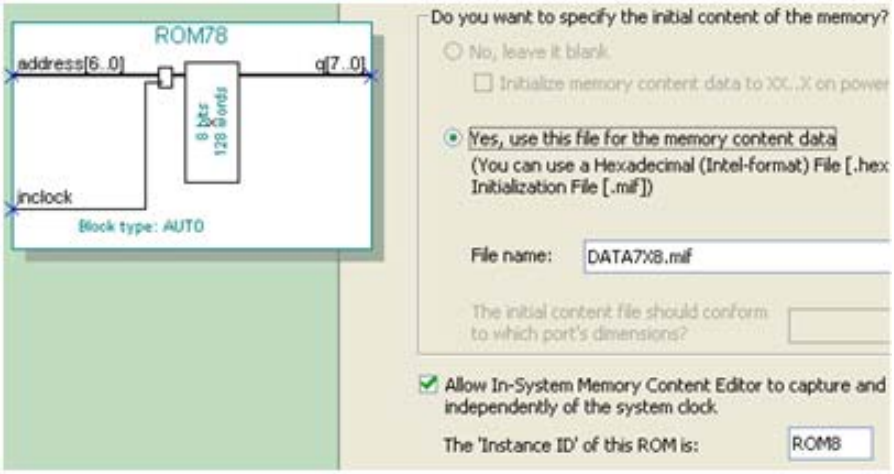

图 6-23 加入初始化配置文件并允许在系统访问

6.4 LPM_ROM的定制和使用示例

6.4.2 简易正弦信号发生器设计

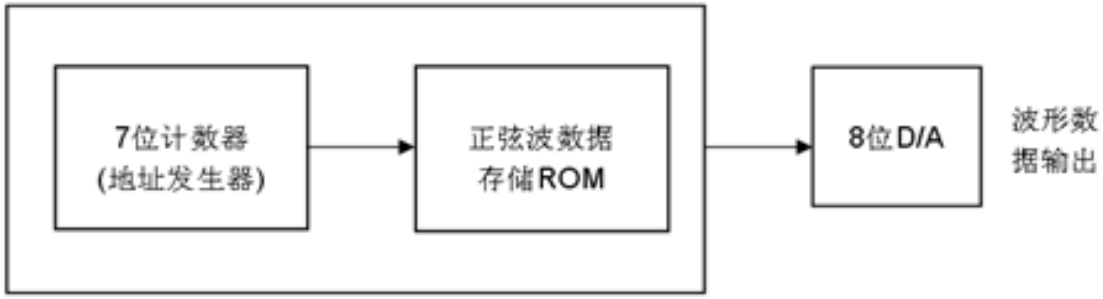

图 6-24 正弦信号发生器结构框图

6.4 LPM_ROM的定制和使用示例

6.4.2 简易正弦信号发生器设计

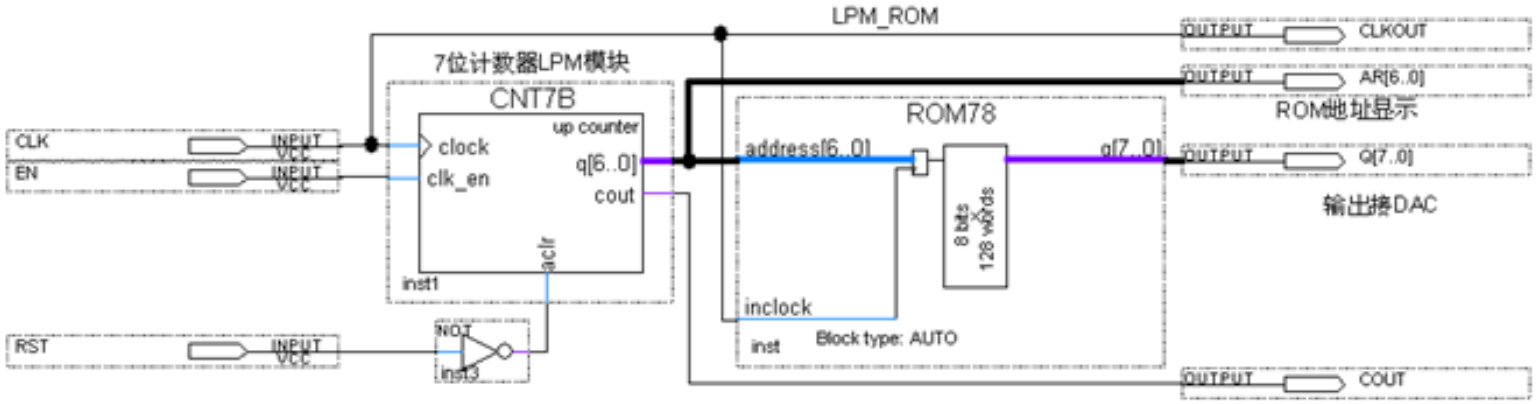

图 6-25 正弦信号发生器电路原理图

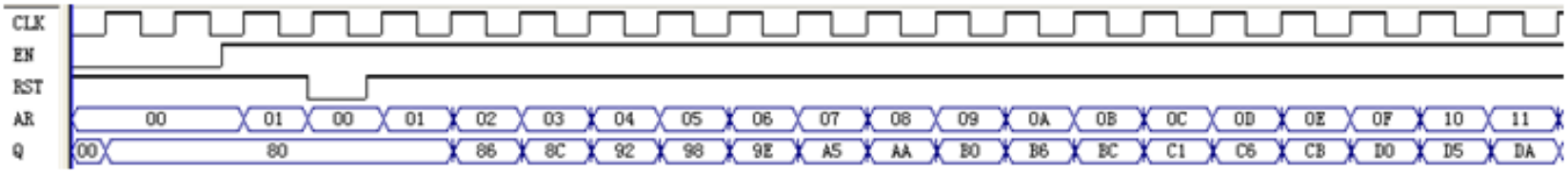

图 6-26 图 6-25 电路仿真波形

6.4 LPM_ROM的定制和使用示例

6.4.3 正弦信号发生器硬件实现和测试

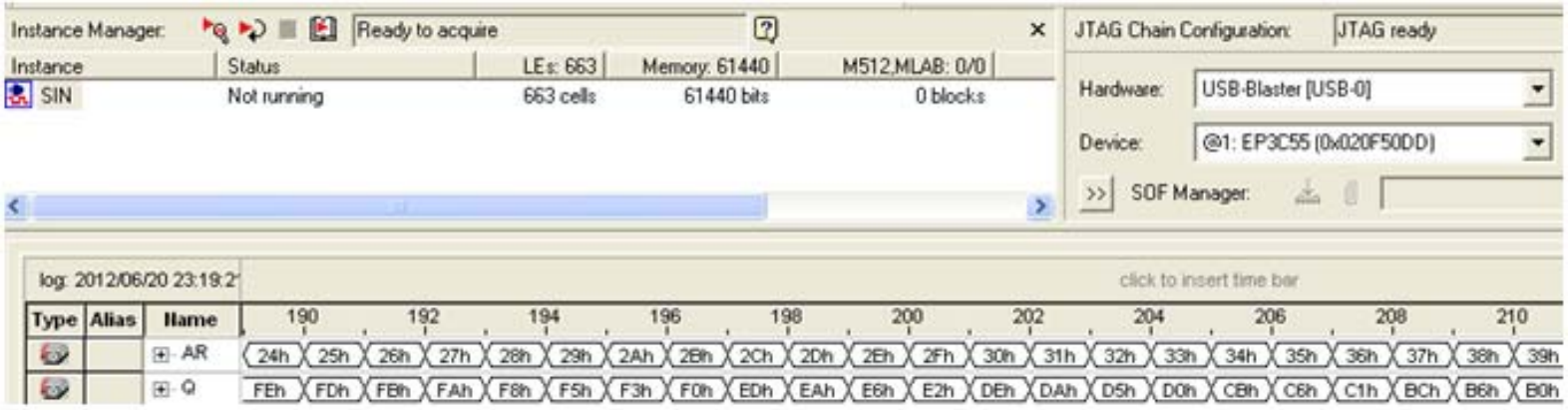

图 6-27 正弦信号发生器数据输出的 SignalTapII 实时测试图

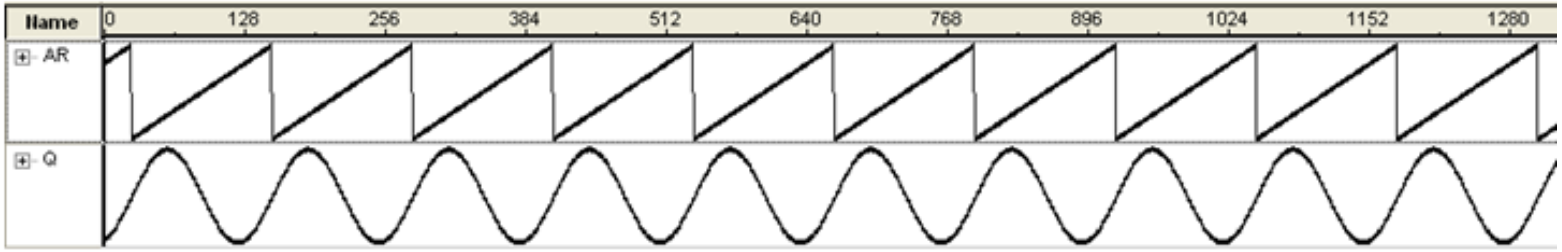

图 6-28 正弦信号发生器的 SignalTapII 的波形显示图

6.5 在系统存储器数据读写编辑器应用

（1）打开在系统存储单元编辑窗口。

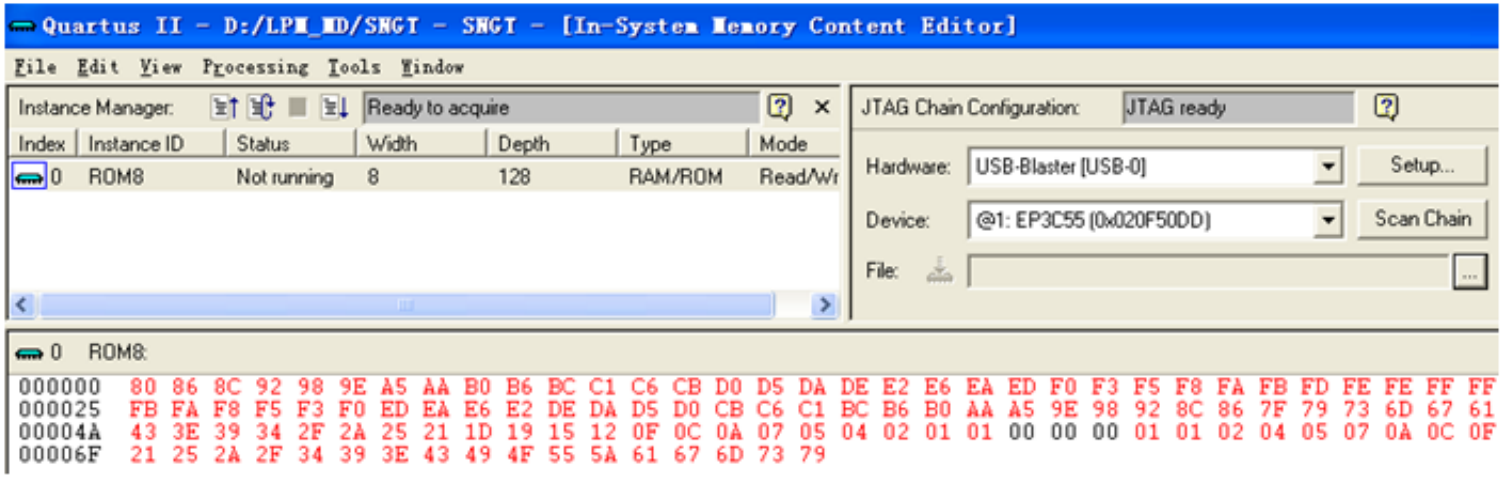

图 6-29 In-System Memory Content Editor 编辑窗

（2）读取ROM中的数据。

6.5 在系统存储器数据读写编辑器应用

（3）写数据。

0 ROM8:

```
000000  11 11 11 11 11 11 11 AA B0 B6 BC C1 C6 CB D0 D5 DA DE E2 E6 EA ED F0 F3 F5 F8 FA FB FD FE FE FF FF FF FE FE FD
000025  FB FA F8 F5 F3 F0 ED EA E6 E2 DE DA D5 D0 CB C6 C1 BC B6 B0 AA A5 9E 98 92 8C 86 7F 79 73 6D 67 61 5A 55 4F 49
00004A  43 3E 39 34 2F 2A 25 21 1D 19 15 12 0F 0C 0A 07 05 04 02 01 01 00 00 00 01 01 02 04 05 07 0A 0C 0F 12 15 19 1D
00006F  21 25 2A 2F 34 39 3E 43 49 4F 55 5A 61 67 6D 73 79
```

图 6-30 从 FPGA 中的 ROM 读取波形数据并编辑数据

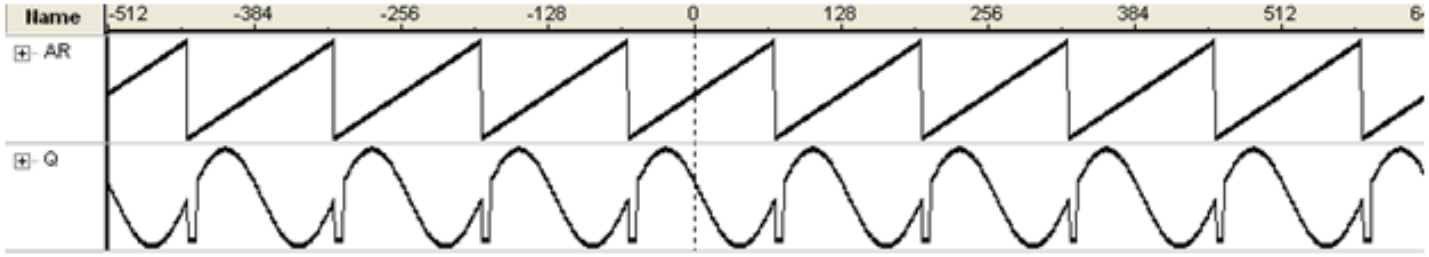

图 6-31 下载编辑数据后的 SignalTap Ⅱ 采样波形

（4）输入输出数据文件。

6.6 LPM嵌入式锁相环调用

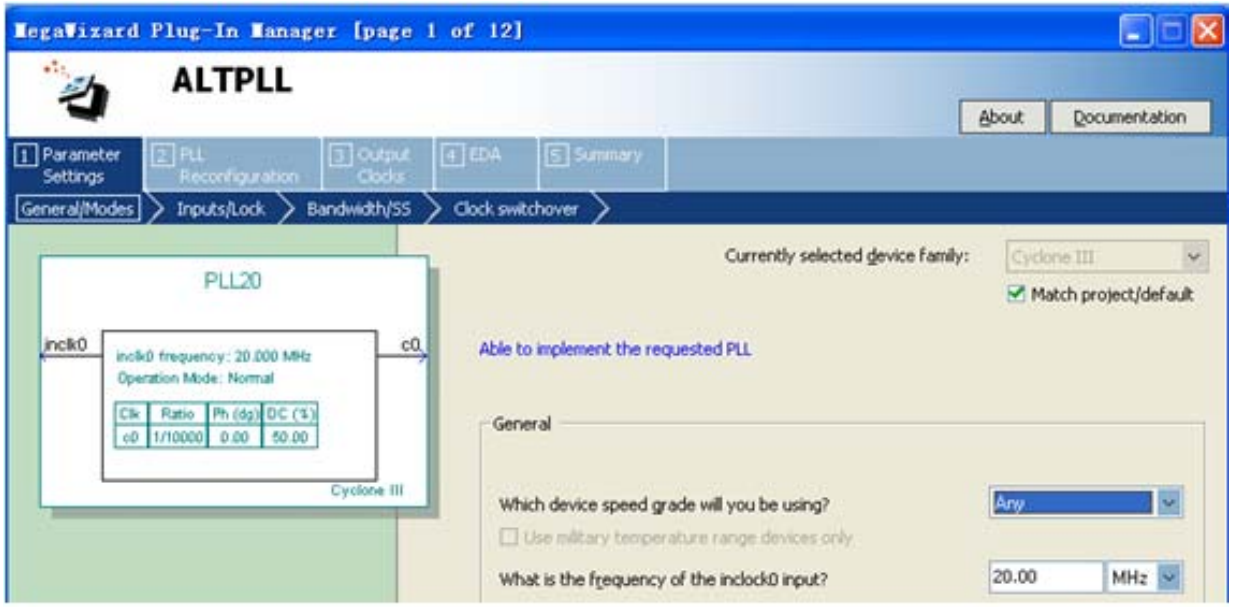

图 6-32 选择输入参考时钟 inclk0 为 20MHz

6.6 LPM嵌入式锁相环调用

图 6-33 选择控制信号

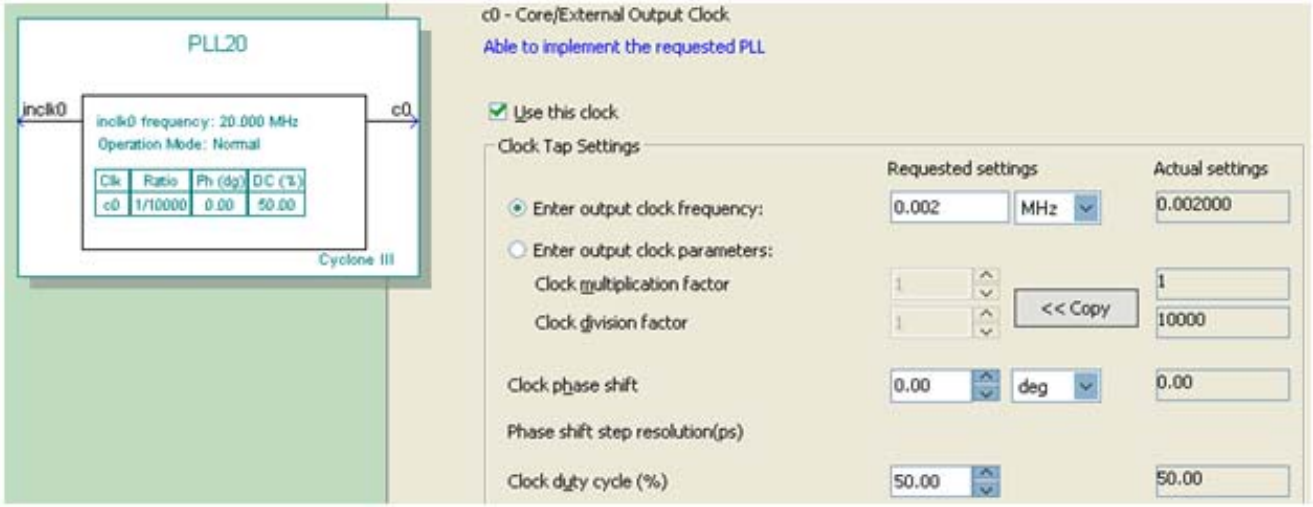

图 6-34 选择 c0 的输出频率为 0.002MHz

6.6 LPM嵌入式锁相环调用

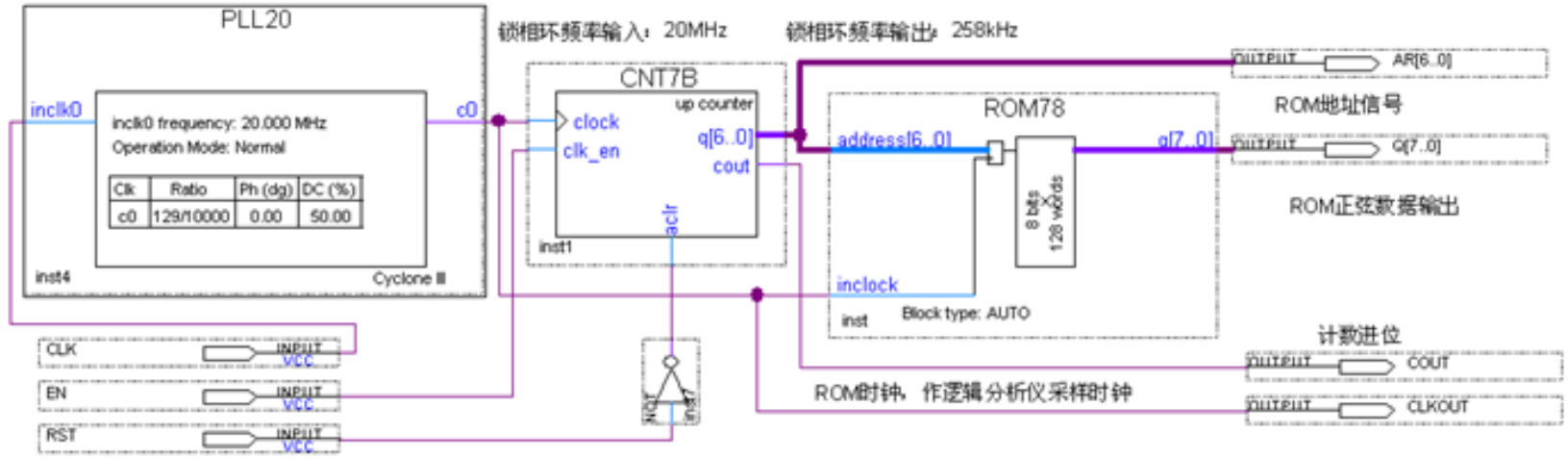

图 6-35 采用嵌入式锁相环作时钟的正弦信号发生器电路图

6.7 In-System Sources and Probes Editor使用方法

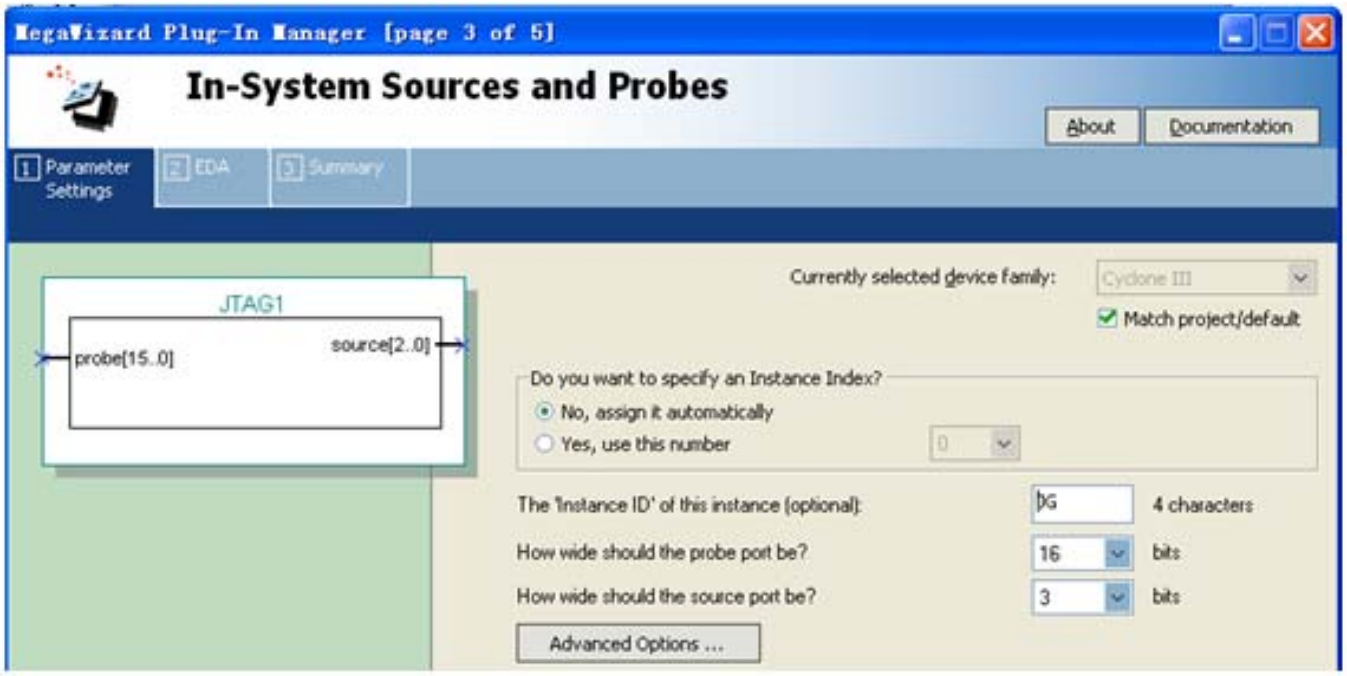

图 6-36 为 In-System Sources and Probes 模块设置参数

6.7 In-System Sources and Probes Editor使用方法

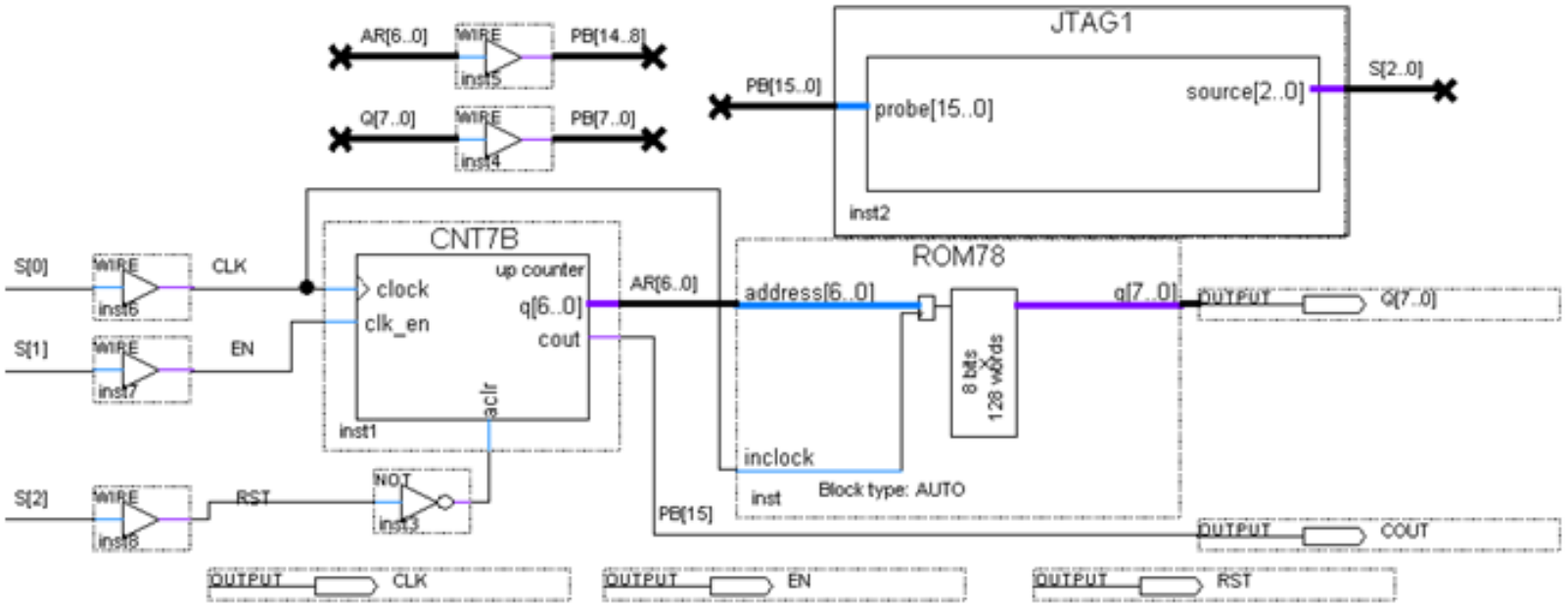

图 6-37 在双十进制计数器设计电路中加入 In-System Sources and Probes 测试模块

6.7 In-System Sources and Probes Editor使用方法

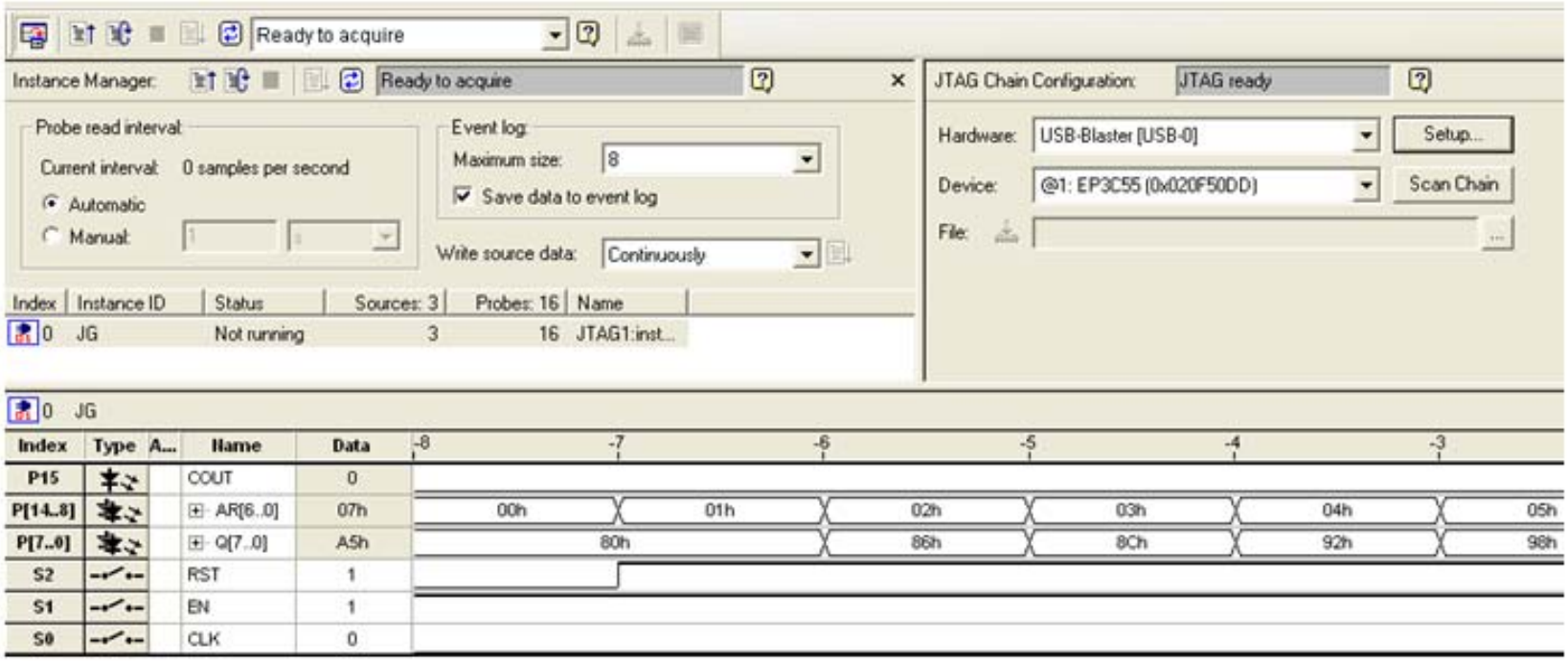

图 6-38 In-System Sources and Probes Editor 的测试情况

6.8 DDS实现原理与应用

6.8.1 DDS原理

$$S_{out} = A\sin\omega t = A\sin(2\pi f_{out} t) \tag{6-1}$$

$$\theta = 2\pi f_{out} t \tag{6-2}$$

$$\Delta\theta = 2\pi f_{out} T_{clk} = \frac{2\pi f_{out}}{f_{clk}} \tag{6-3}$$

$$B_{\Delta\theta} \approx \frac{\Delta\theta}{2\pi}\cdot 2^N \qquad \frac{B_{\Delta\theta}}{2^N} = \frac{f_{out}}{f_{clk}},\quad B_{\Delta\theta} = 2^N \cdot \frac{f_{out}}{f_{clk}} \tag{6-4}$$

$$S_{out} = A\sin(\theta_{k-1} + \Delta\theta) = A\sin\left[\frac{2\pi}{2^N}\cdot\left(B_{\theta_{k-1}} + B_{\Delta\theta}\right)\right] = Af_{sin}\left(B_{\theta_{k-1}} + B_{\Delta\theta}\right) \tag{6-5}$$

$$B_{\theta_{k-1}} \approx \frac{\theta_{k-1}}{2\pi}\cdot 2^N \tag{6-6}$$

6.8 DDS实现原理与应用

6.8.1 DDS原理

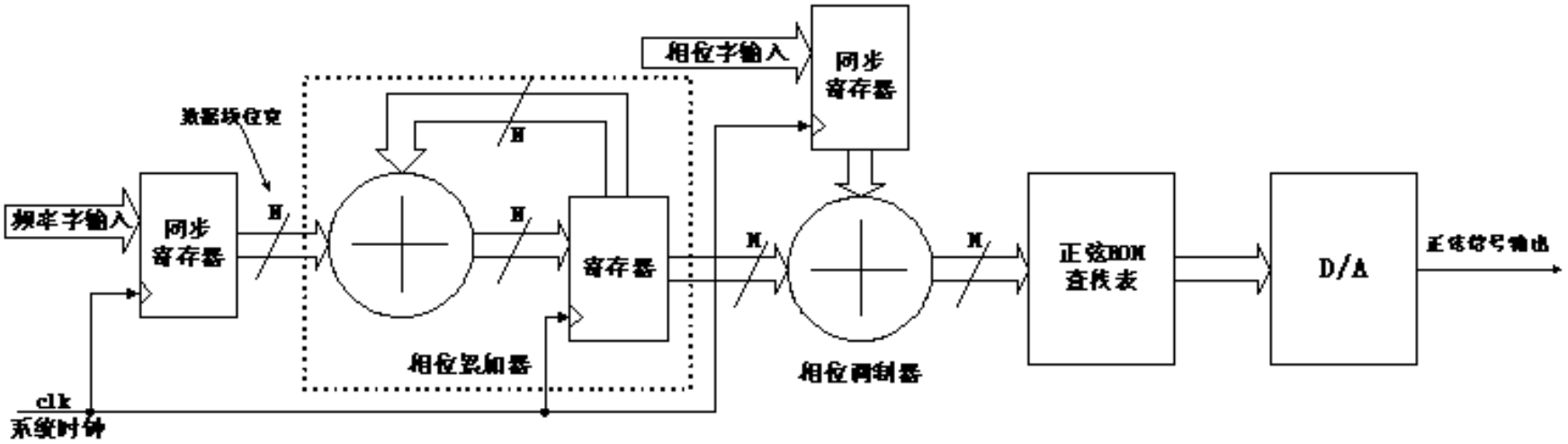

图 6-39 基本 DDS 结构

$$f_{out} = \frac{B_{\Delta\theta}}{2^N} \cdot f_{clk} \qquad (6\text{-}7)$$

$$f_{out} = \frac{f_{clk}}{2^N} \qquad (6\text{-}8)$$

6.8 DDS实现原理与应用

6.8.2 DDS信号发生器设计示例

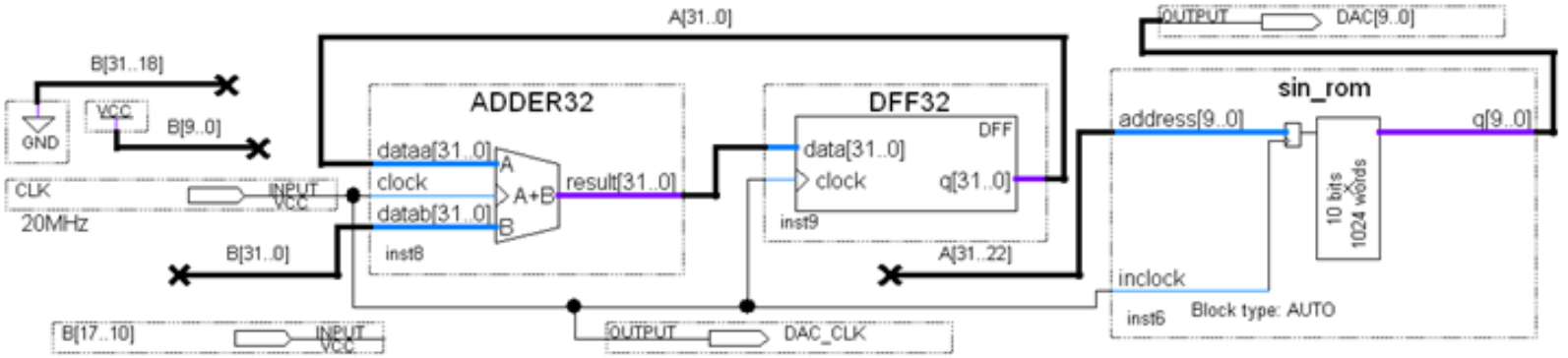

图 6-40　DDS 信号发生器电路顶层原理图

6.8 DDS实现原理与应用

6.8.2 DDS信号发生器设计示例

$$f_{out} = \frac{B[31..0]}{2^{32}} \cdot f_{clk} \qquad (6\text{-}9)$$

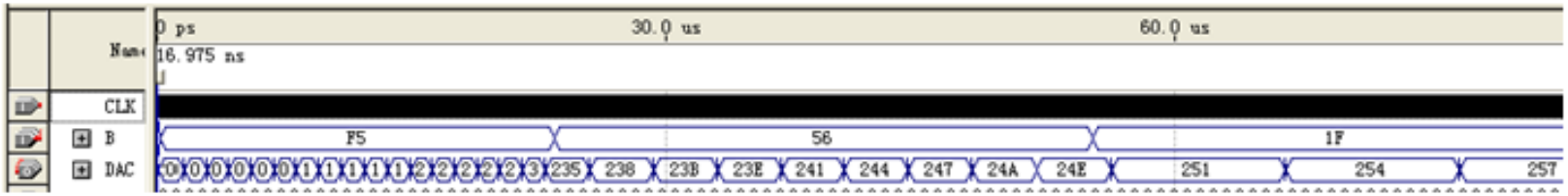

图 6-41 电路原理图图 6-40 的仿真波形

实验与设计

6-1．查表式硬件运算器设计

6-2 正弦信号发生器设计

6-3 DDS正弦信号发生器设计

实验与设计

6-4. 简易逻辑分析仪设计

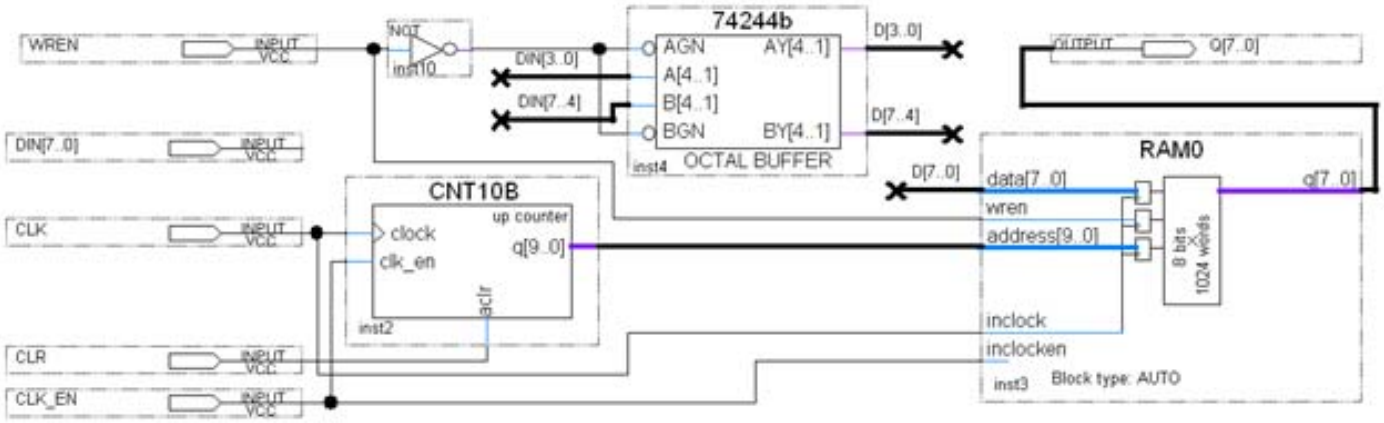

图 6-42　逻辑数据采样电路顶层设计

实验与设计

6-4. 简易逻辑分析仪设计

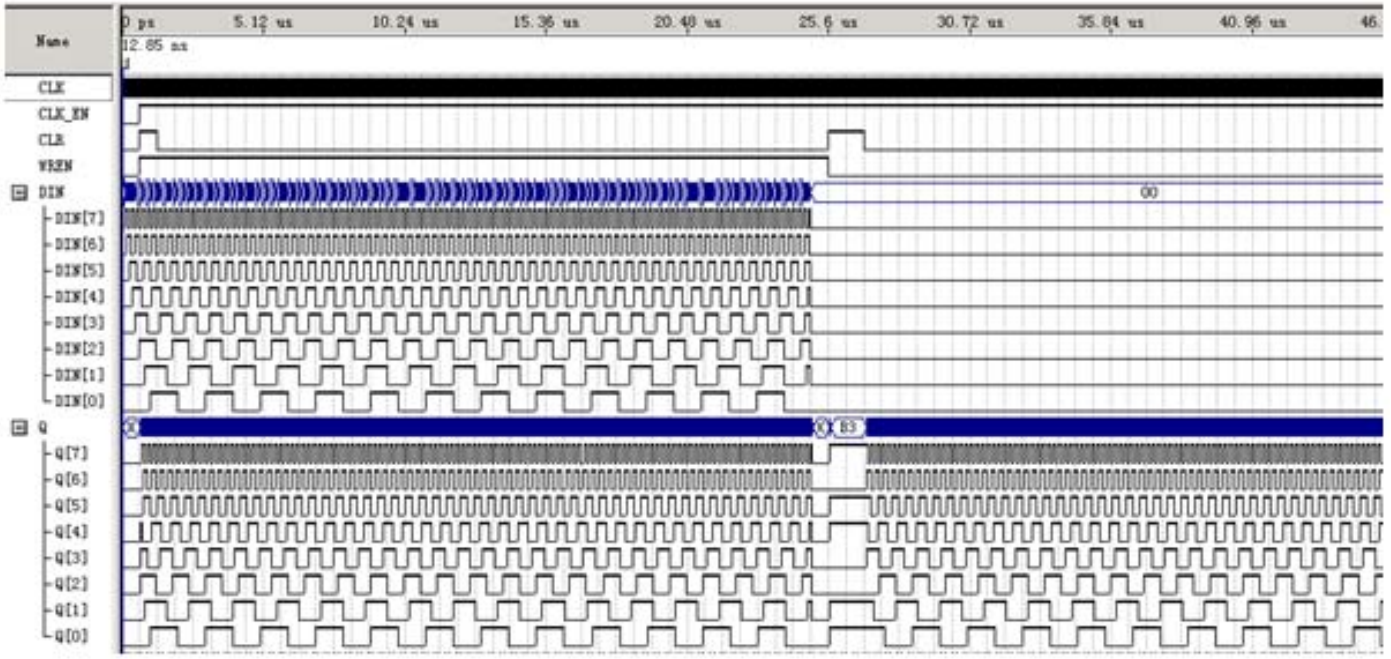

图 6-43 逻辑数据采样电路时序仿真波形

实验与设计

6-5 移相信号发生器设计

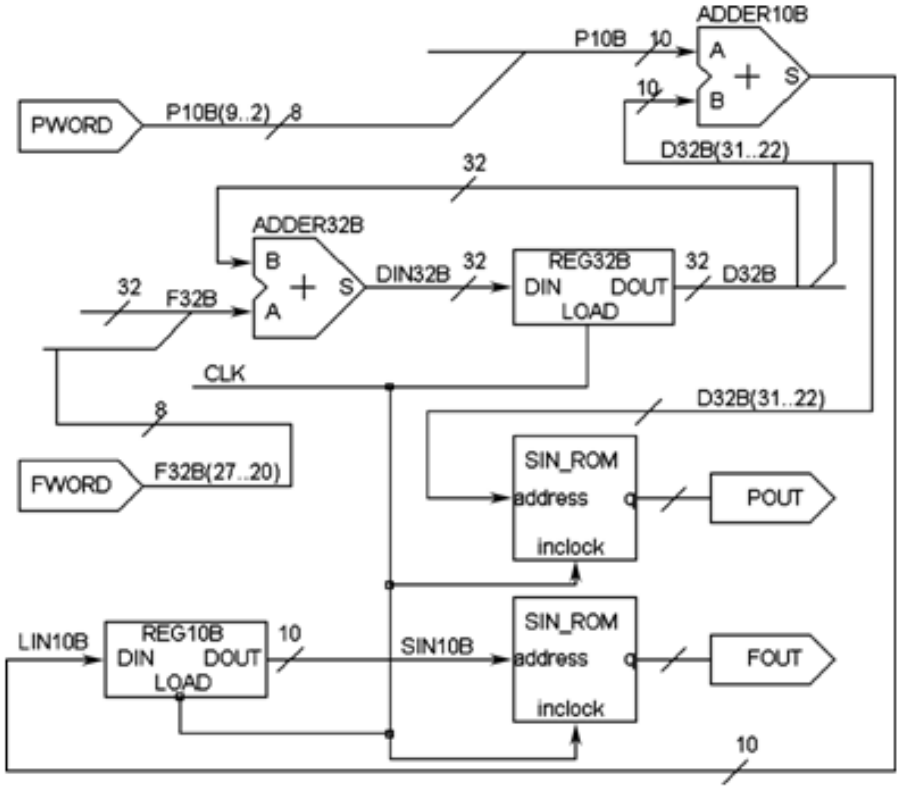

图 6-44 全数字移相信号发生器电路模型图

6-6 VGA简单图像显示控制模块设计

【例 6-7】

```
LIBRARY ieee;  --图象显示顶层程序
USE ieee.std_logic_1164.all;
ENTITY vgaV IS
 Port ( clk50MHz :  IN  STD_LOGIC;
        hs, vs, r, g, b :  OUT  STD_LOGIC );
END vgaV;
ARCHITECTURE modelstru OF vgaV IS
 component vga640480              --VGA 显示控制模块
 PORT(clk : IN STD_LOGIC;
      rgbin : IN STD_LOGIC_VECTOR(2 downto 0);
      hs, vs, r, g, b : OUT STD_LOGIC;
      hcntout, vcntout : OUT STD_LOGIC_VECTOR(9 downto 0)   );
end component;
component imgrom                 --图象数据 ROM，数据线 3 位；地址线 12 位
 PORT(inclock : IN STD_LOGIC;
      address : IN STD_LOGIC_VECTOR(11 downto 0);
      q : OUT STD_LOGIC_VECTOR(2 downto 0)  );
end component;
signal   rgb :  STD_LOGIC_VECTOR(2 downto 0);
signal   clk25MHz : std_logic;
signal   romaddr :  STD_LOGIC_VECTOR(11 downto 0);
signal   hpos, vpos  :  std_logic_vector(9 downto 0);
BEGIN
romaddr <= vpos(5 downto 0) & hpos(5 downto 0);
process(clk50MHz) begin
if clk50MHz'event and clk50MHz='1' then clk25MHz<=not clk25MHz; end if;
end process;
i_vga640480 : vga640480 PORT MAP(clk => clk25MHz, rgbin => rgb, hs => hs,
vs => vs, r => r, g => g, b => b, hcntout => hpos, vcntout => vpos);
i_rom : imgrom PORT MAP(inclock => clk25MHz, address => romaddr, q => rgb);
end;
```

实验与设计

6-6 VGA简单图像显示控制模块设计

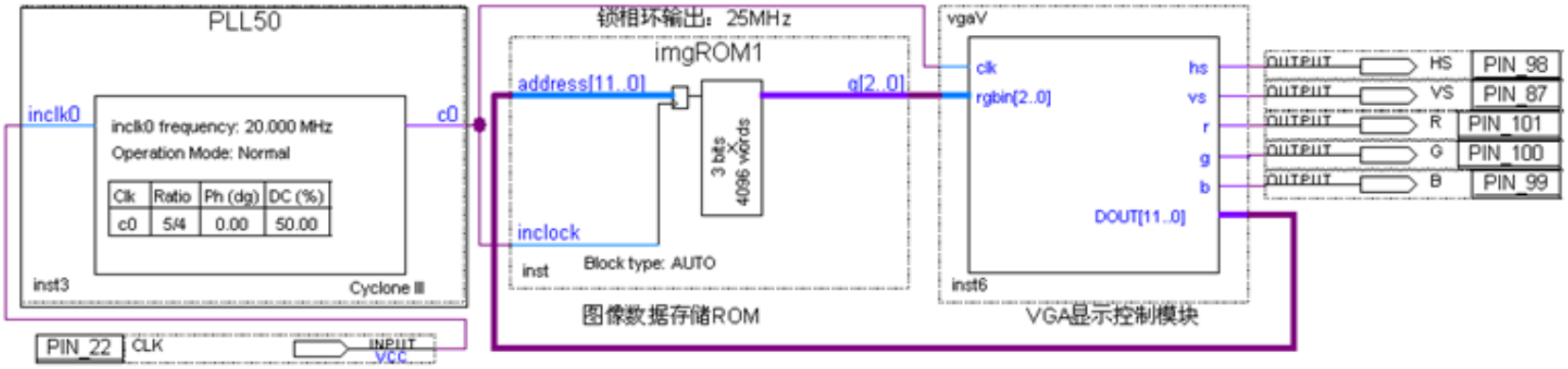

图 6-45　VGA 图像显示控制模块原理图

实验与设计

6-7 AM幅度调制信号发生器设计

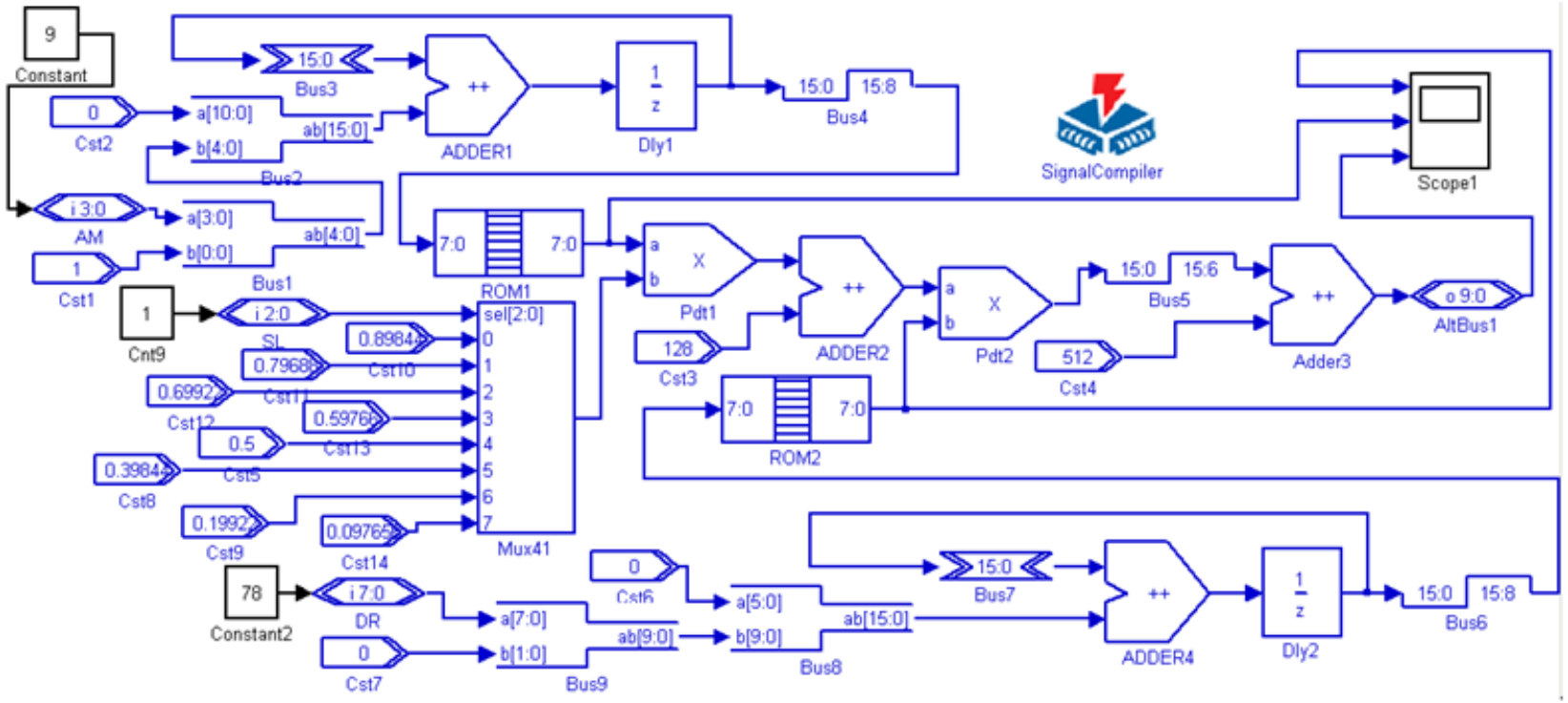

图 6-46　AM 信号发生器 DSP Builder/MATLAB Simulink 模型

实验与设计

6-7 AM幅度调制信号发生器设计

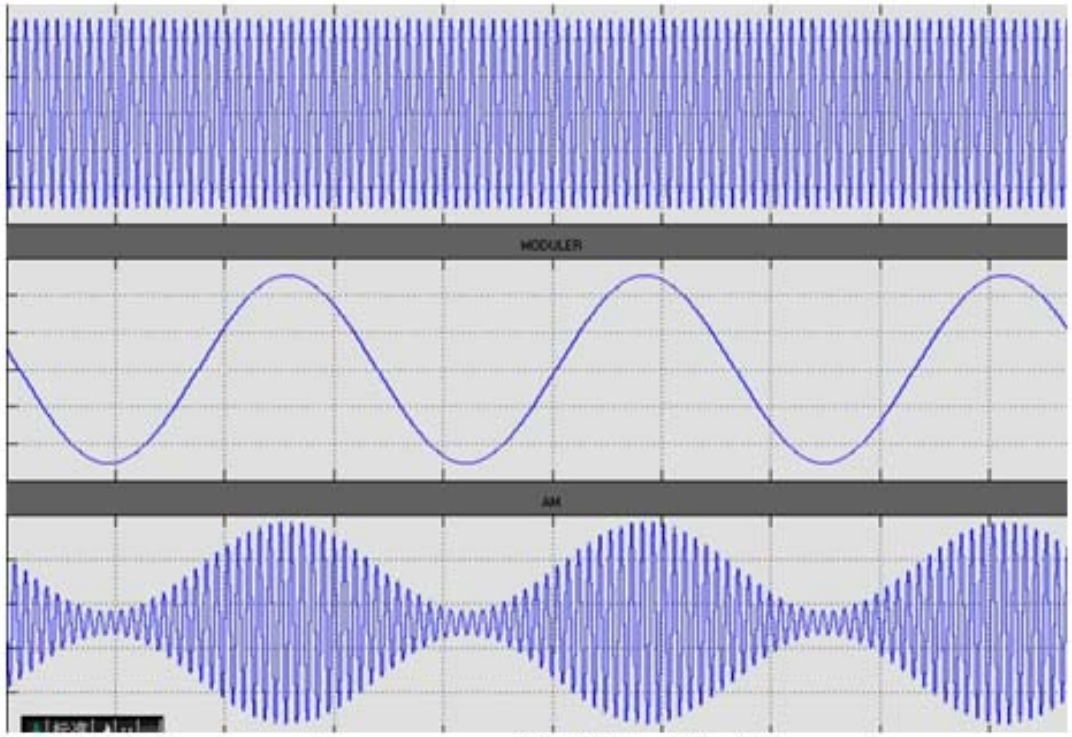

图 6-47　AM 模型仿真波形