第6章

LPM宏模块使用方法

6.1 调用计数器宏模块示例

6.1.1 计数器LPM模块文本代码的调用

（1）打开LPM宏功能块调用管理器。

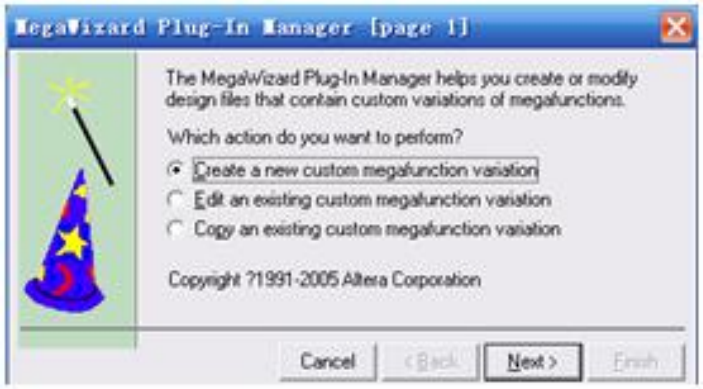

图 6-1　定制新的宏功能块

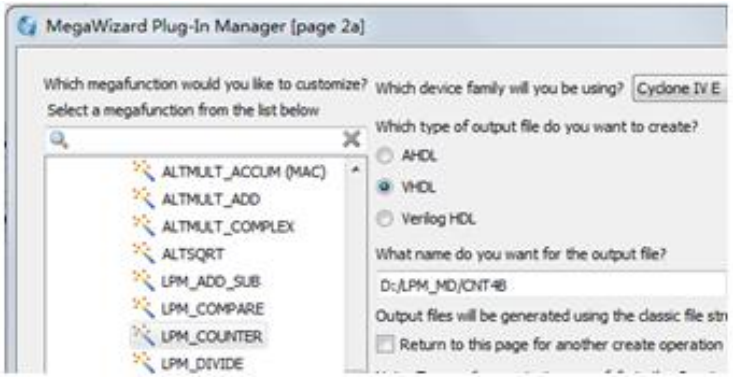

图 6-2　LPM 宏功能块设定

6.1 调用计数器宏模块示例

6.1.1 计数器LPM模块文本代码的调用

（2）单击Next按钮后打开如图6-3所示的对话框。

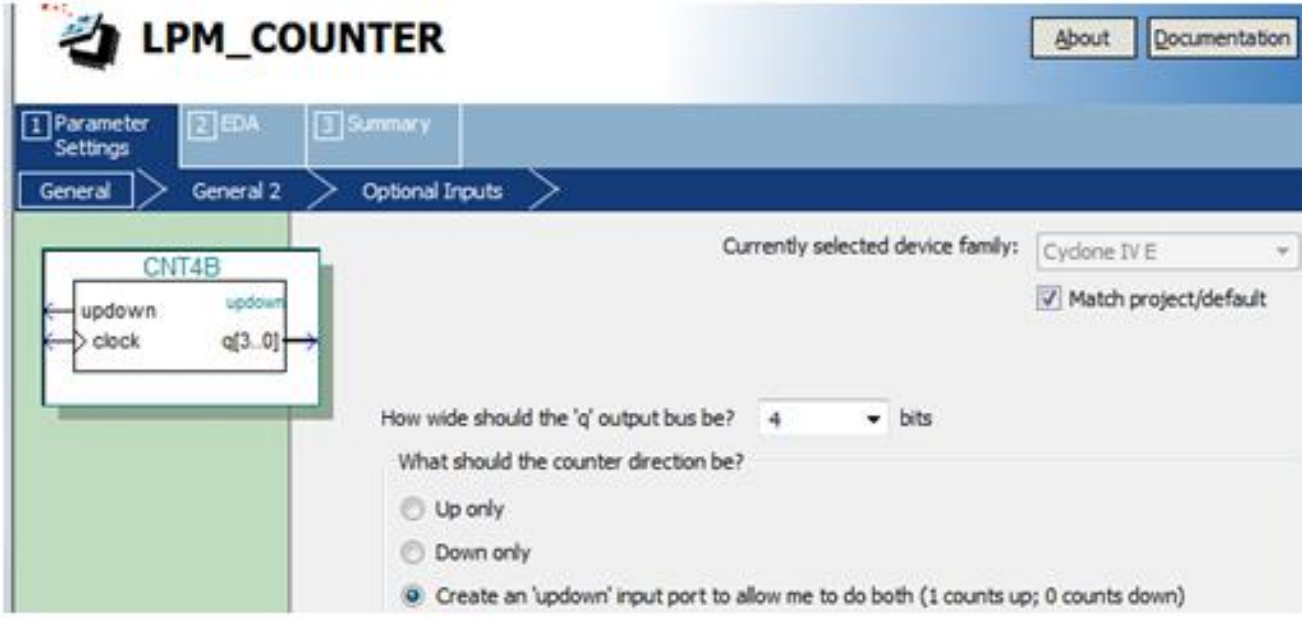

图 6-3　设 4 位可加减计数器

6.1 调用计数器宏模块示例

6.1.1 计数器LPM模块文本代码的调用

（3）再单击Next按钮，打开如图6-4所示的对话框。

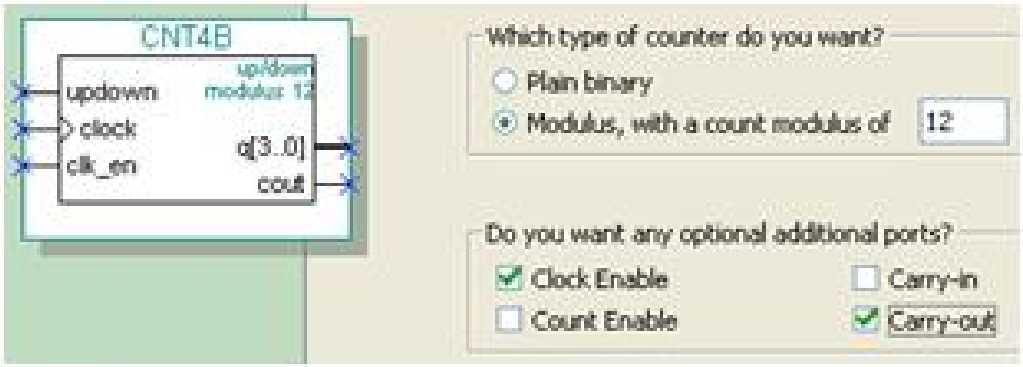

图 6-4　设定计数器，含时钟使能和进位输出

6.1 调用计数器宏模块示例

6.1.1 计数器LPM模块文本代码的调用

（4）再单击Next按钮，打开如图6-5所示的对话框。

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 调用计数器宏模块示例

6.1.2 LPM计数器代码与参数传递语句应用

【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 调用计数器宏模块示例

6.1.3 创建工程与仿真测试

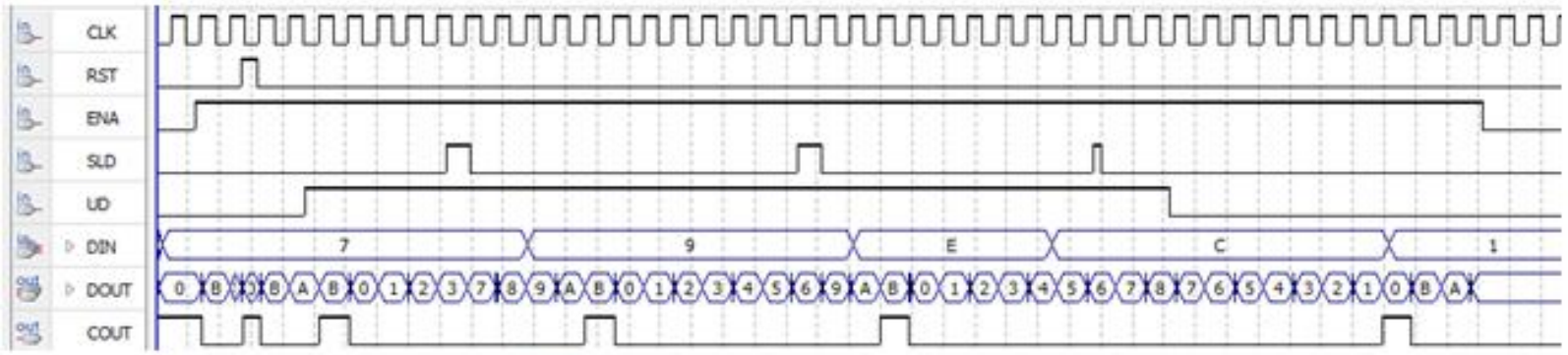

图 6-6　CNT4BIT.vhd 的仿真波形

6.2 利用属性控制乘法器构建的示例

【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;
```

图 6-7 例 6-3 的仿真波形图

6.2 利用属性控制乘法器构建的示例

Flow Status	Successful - Wed May 31 20:17:39 2017
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Full Version
Revision Name	MULT8
Top-level Entity Name	MULT8
Family	Cyclone IV E
Device	EP4CE55F23C8
Timing Models	Final
Total logic elements	95 / 55,856 (< 1 %)
Total combinational functions	95 / 55,856 (< 1 %)
Dedicated logic registers	0 / 55,856 (0 %)
Total registers	0
Total pins	32 / 325 (10 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

图 6-8　完全用逻辑宏单元构建乘法器的编译报告

Flow Status	Successful - Wed May 31 20:19:55 2017
Quartus II 64-Bit Version	13.1.0 Build 162 10/23/2013 SJ Full Version
Revision Name	MULT8
Top-level Entity Name	MULT8
Family	Cyclone IV E
Device	EP4CE55F23C8
Timing Models	Final
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 / 325 (10 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	1 / 308 (< 1 %)
Total PLLs	0 / 4 (0 %)

图 6-9　调用了 DSP 模块的编译报告

6.2 利用属性控制乘法器构建的示例

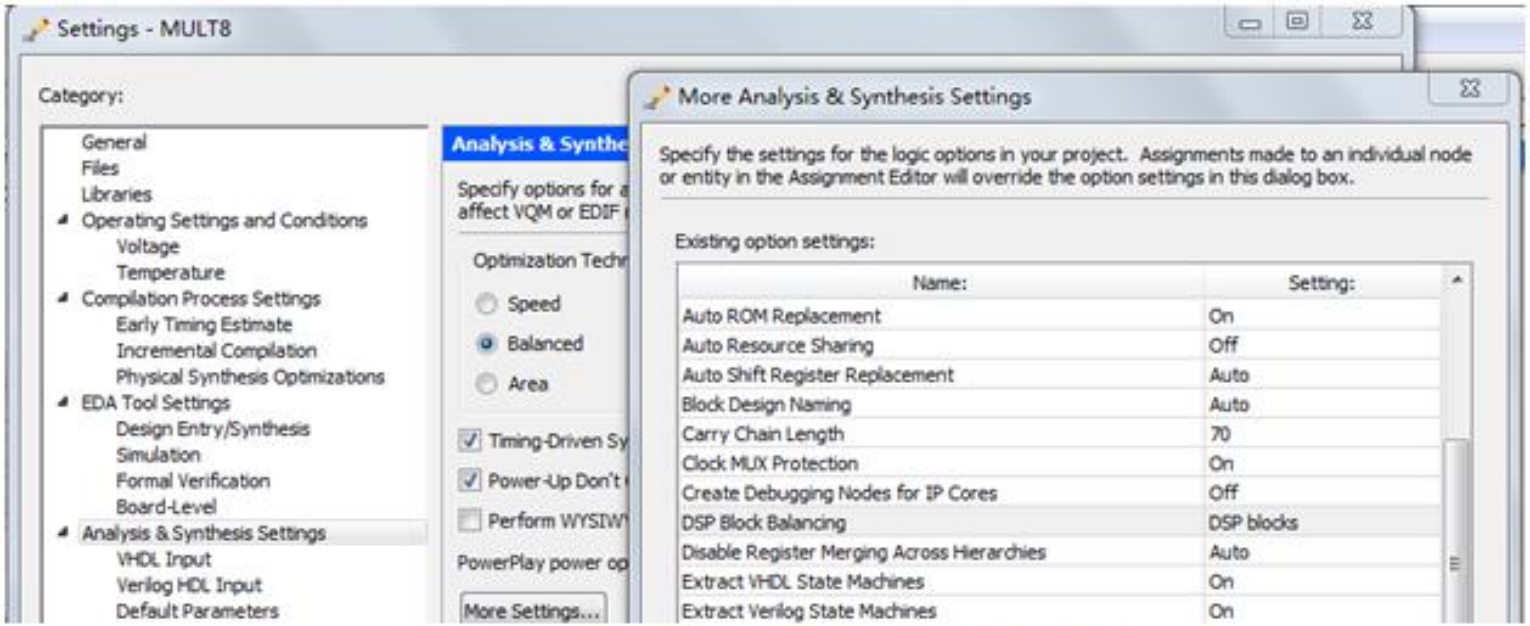

图6-10　选择DSP Block Balancing为DSP blocks

6.3 LPM_RAM宏模块用法

6.3.1 初始化文件及其生成

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 6-11 .mif 文件编辑窗口

6.3 LPM_RAM宏模块用法

6.3.1 初始化文件及其生成

1. .mif 格式文件

（2）文件直接编辑法。

【例 6-4】

```
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表示选择十六进制数据类型
DATA_RADIX  = HEX;  //存储数据类型
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_RAM宏模块用法

6.3.1 初始化文件及其生成

（3）高级语言生成。

（4）专用.mif文件生成器。

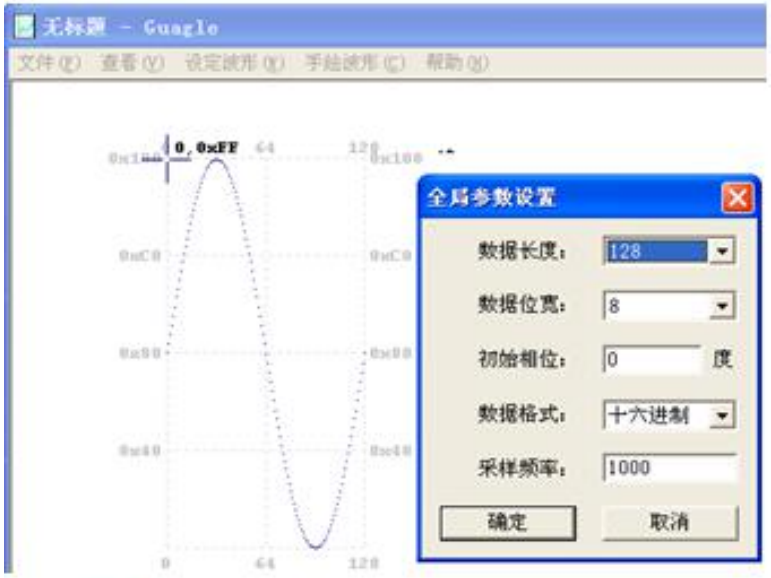

图 6-12 利用 mif 生成器生成.mif 正弦波文件

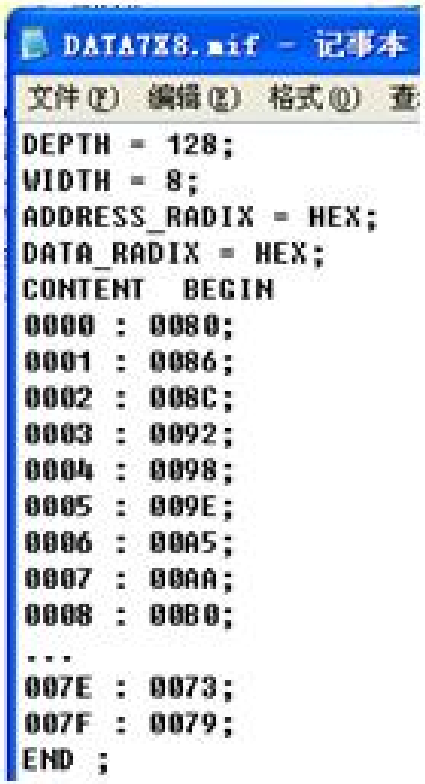

图 6-13 打开.mif 文件

6.3 LPM_RAM宏模块用法

6.3.1 初始化文件及其生成

2. .hex格式文件

6.3.2 LPM_RAM的设置与调用

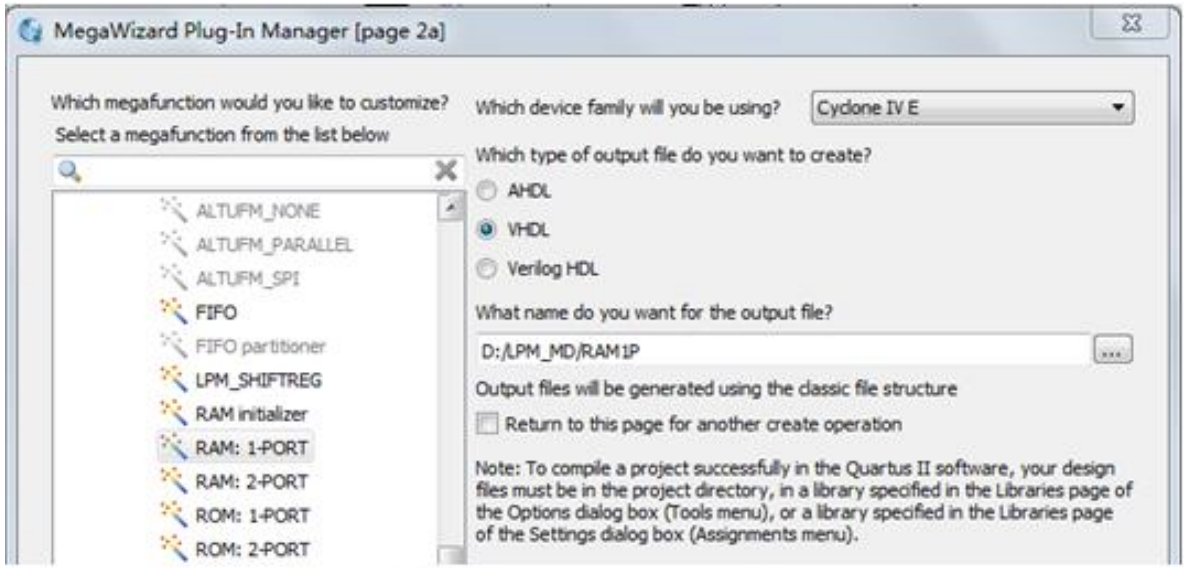

图 6-14 调用单口 LPM RAM

6.3 LPM_RAM宏模块用法

6.3.2 LPM_RAM的设置与调用

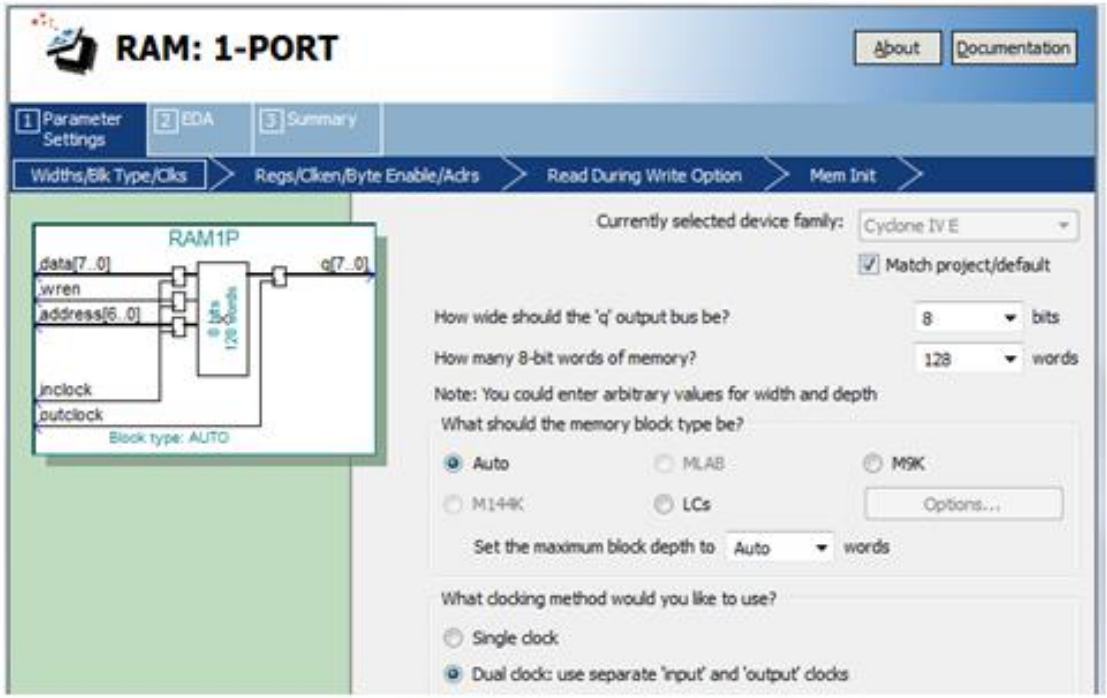

图 6-15 设定 RAM 参数

6.3 LPM_RAM宏模块用法

6.3.2 LPM_RAM的设置与调用

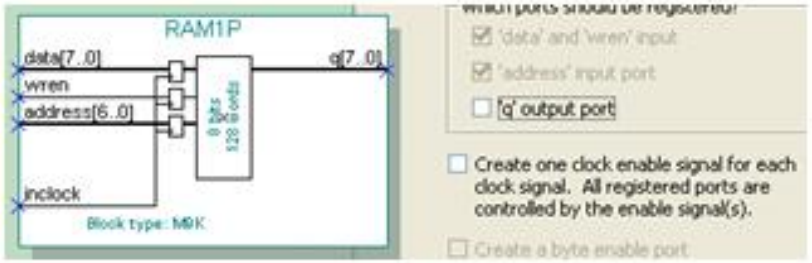

图 6-16 设定 RAM 仅输入时钟控制

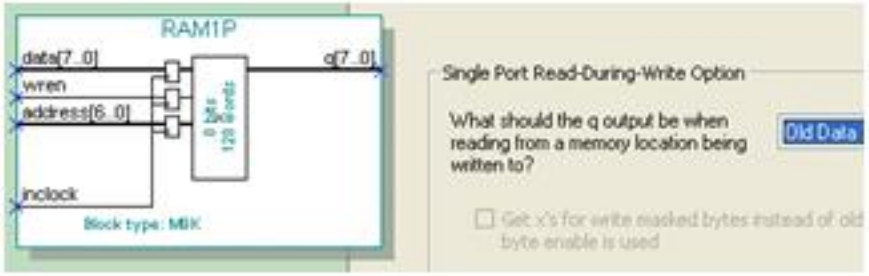

图 6-17 设定在写入同时读出原数据： Old Data

6.3 LPM_RAM宏模块用法

6.3.2 LPM_RAM的设置与调用

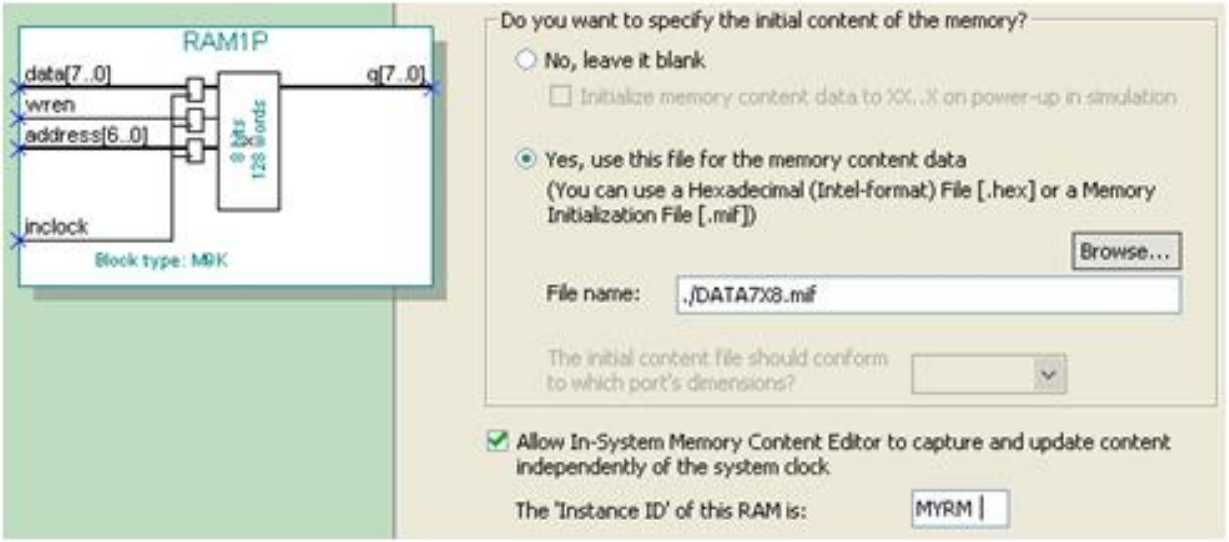

图 6-18 设定初始化文件和允许在系统编辑

6.3 LPM_RAM宏模块用法

6.3.2 LPM_RAM的设置与调用

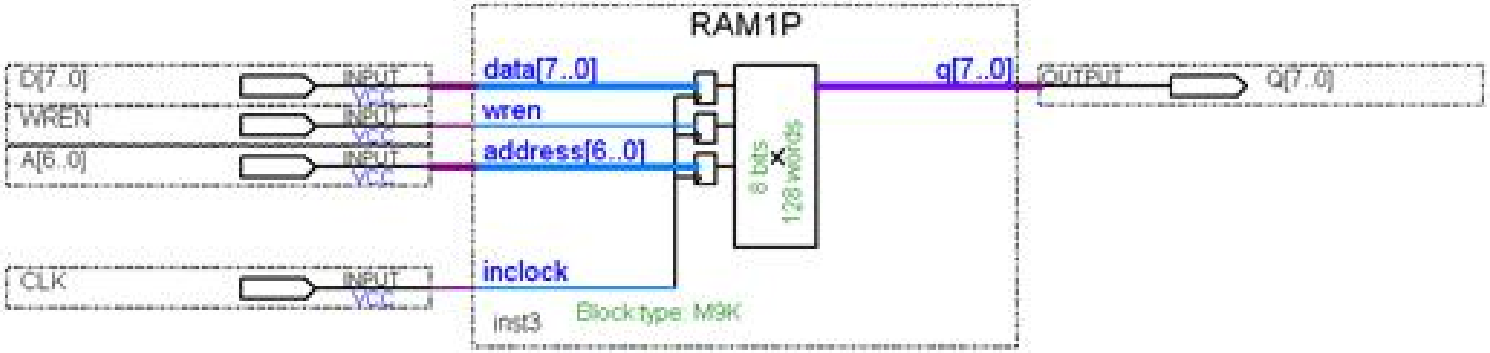

图 6-19 在原理图上连接好的 RAM 模块

6.3 LPM_RAM宏模块用法

6.3.3 测试LPM_RAM

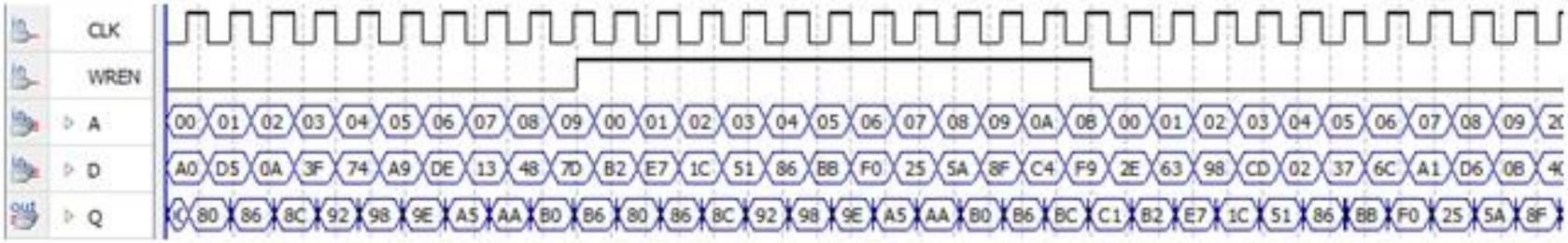

图 6-20　图 6-19 的 RAM 仿真波形

6.3 LPM_RAM宏模块用法

6.3.4 用VHDL代码描述存储器以及用初始化文件加载表述

【例 6-5】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY RAM78 IS
PORT (CLK,WREN : IN STD_LOGIC;
       A : IN STD_LOGIC_VECTOR(6 DOWNTO 0);
     DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
       Q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END;
ARCHITECTURE bhv OF RAM78 IS
TYPE G_ARRAY IS ARRAY(0 TO 127) OF STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL MEM : G_ARRAY;  --定义信号MEM的数据类型是用户新定义的类型G_ARRAY
 attribute ram_init_file : string;
 attribute ram_init_file of MEM :
 SIGNAL IS "data7x8.mif";   --加载初始化文件
BEGIN
  PROCESS (CLK)       BEGIN
     IF RISING_EDGE(CLK)  THEN
     IF WREN='1' THEN   MEM(CONV_INTEGER(A))<= DIN;  END IF;
     END IF;
  IF (FALLING_EDGE(CLK)) THEN  Q<=MEM(CONV_INTEGER(A));   END IF;
 END PROCESS;
 END BHV;
```

6.3 LPM_RAM宏模块用法

6.3.4 用VHDL代码描述存储器以及用初始化文件加载表述

Top-level Entity Name	RAM78
Family	Cyclone IV E
Device	EP4CE55F23C8
Timing Models	Final
Total logic elements	1,337 / 55,856 (2 %)
Total combinational functions	832 / 55,856 (1 %)
Dedicated logic registers	1,032 / 55,856 (2 %)
Total registers	1032
Total pins	25 / 325 (8 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

图 6-21 例 5-10 无约束条件下的编译报告

Top-level Entity Name	RAM78
Family	Cyclone IV E
Device	EP4CE55F23C8
Timing Models	Final
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	25 / 325 (8 %)
Total virtual pins	0
Total memory bits	1,024 / 2,396,160 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

图 6-22 例 5-10 使用嵌入 RAM 的编译报告

6.4 LPM_ROM使用示例

6.4.1 简易正弦信号发生器设计

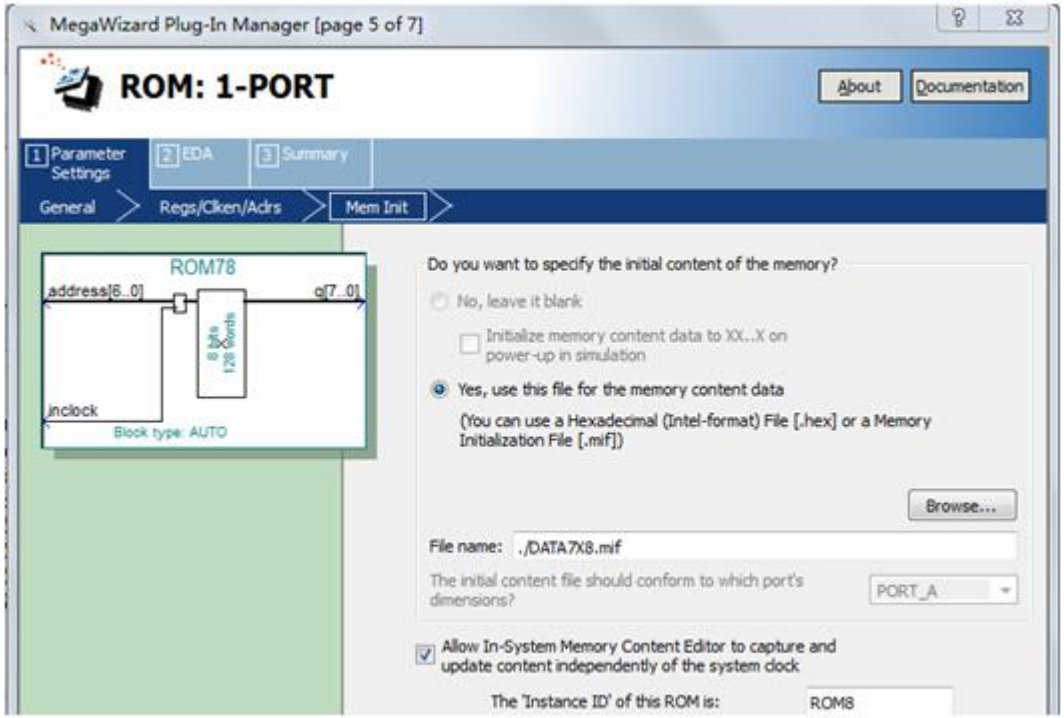

图 6-23 加入初始化配置文件并允许在系统访问 ROM 内容

6.4 LPM_ROM使用示例

6.4.1 简易正弦信号发生器设计

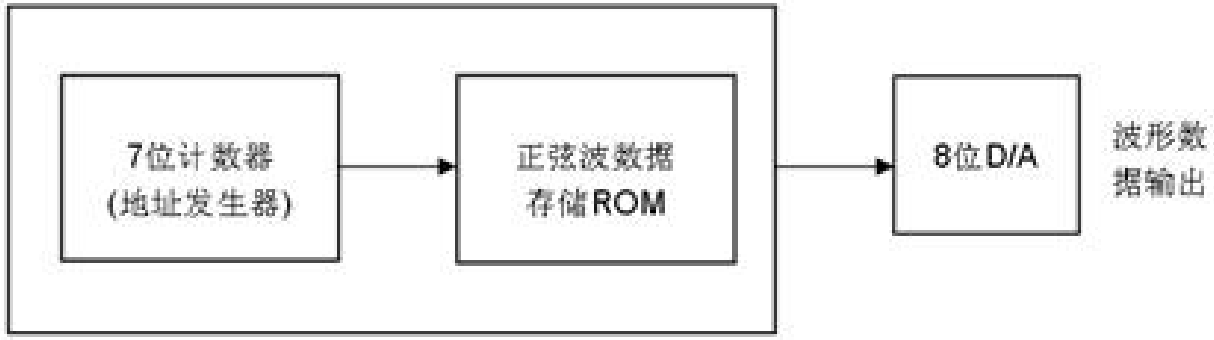

图 6-24 正弦信号发生器结构框图

6.4 LPM_ROM使用示例

6.4.1 简易正弦信号发生器设计

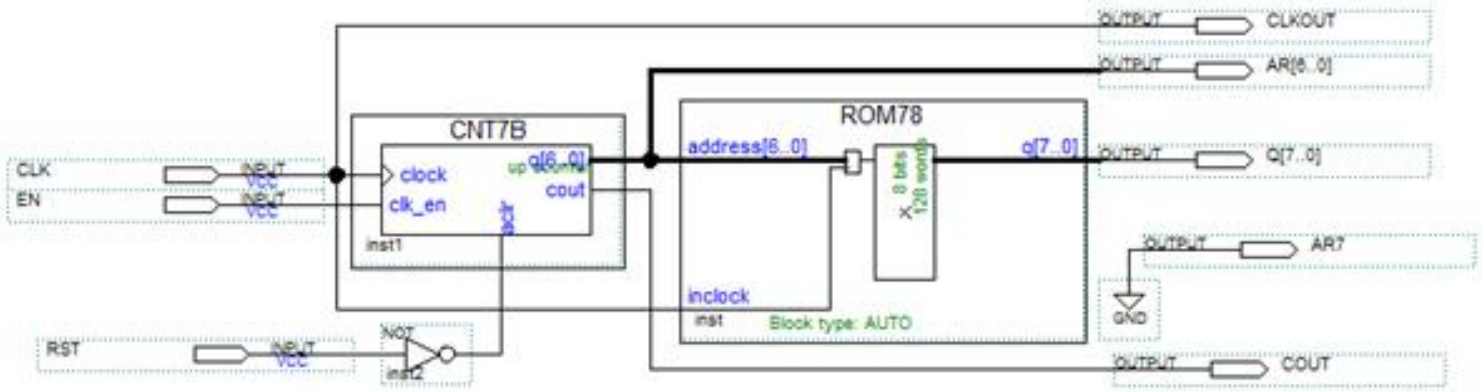

图 6-25 正弦信号发生器电路原理图

6.4 LPM_ROM使用示例

6.4.1 简易正弦信号发生器设计

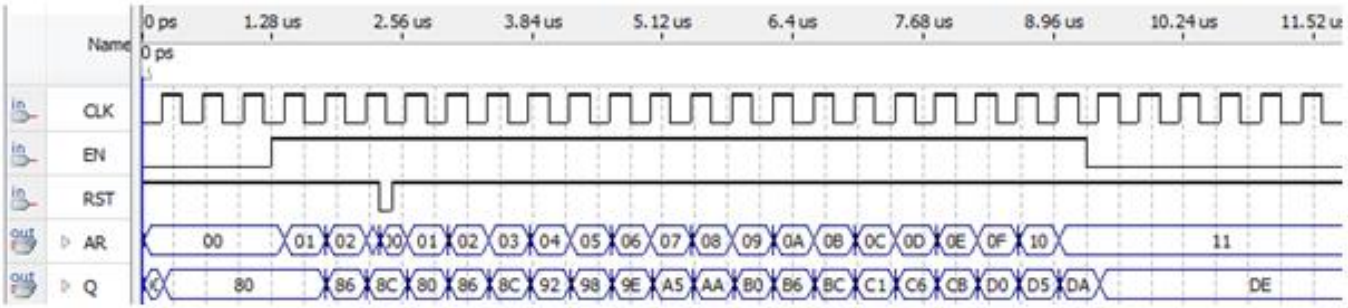

图 6-26 图 25 电路仿真波形

6.4 LPM_ROM的定制和使用示例

6.4.2 正弦信号发生器硬件实现和测试

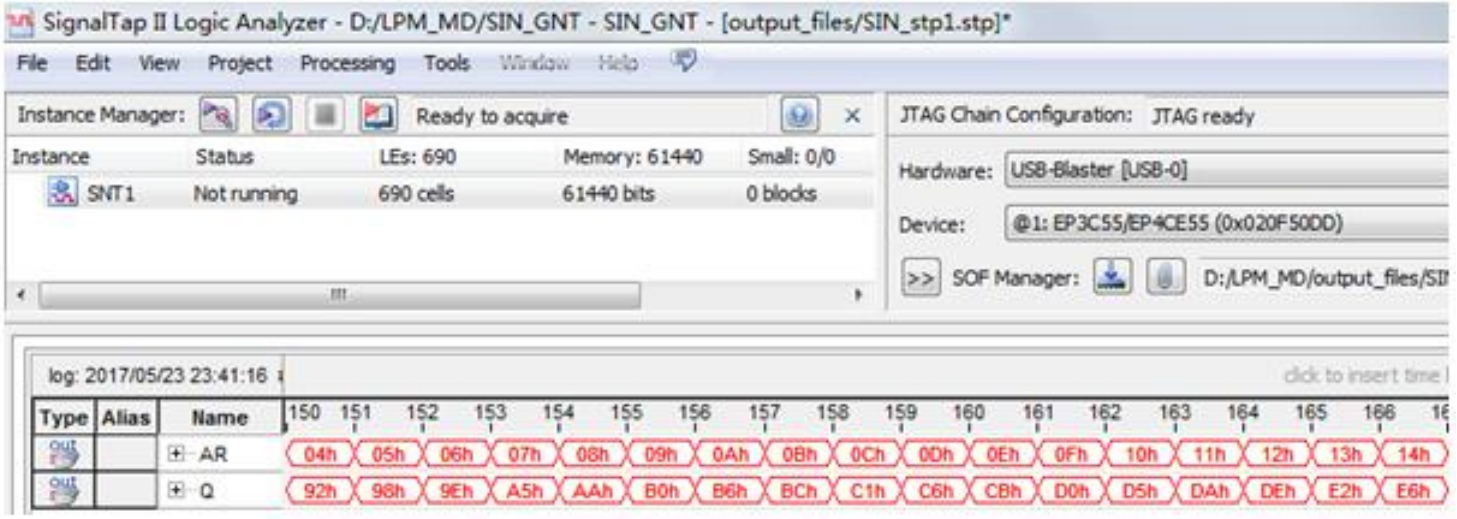

图 6-27 正弦信号发生器数据输出的 SignalTap II 实时测试界面

6.4 LPM_ROM的定制和使用示例

6.4.2 正弦信号发生器硬件实现和测试

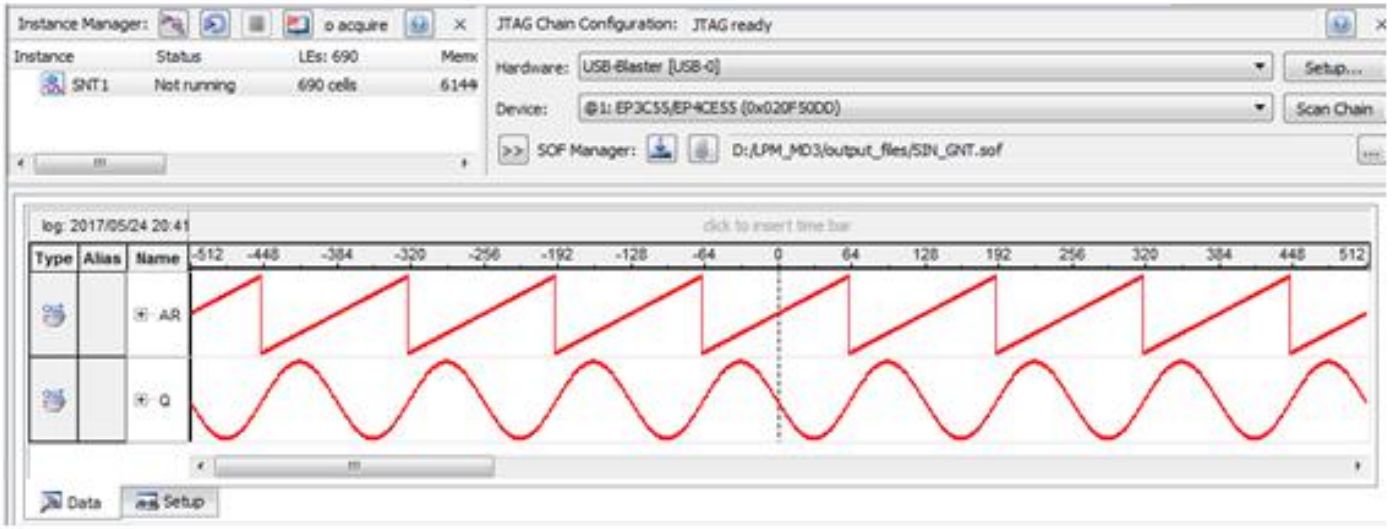

图 6-28 正弦信号发生器的 SignalTap II 的波形显示图

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。

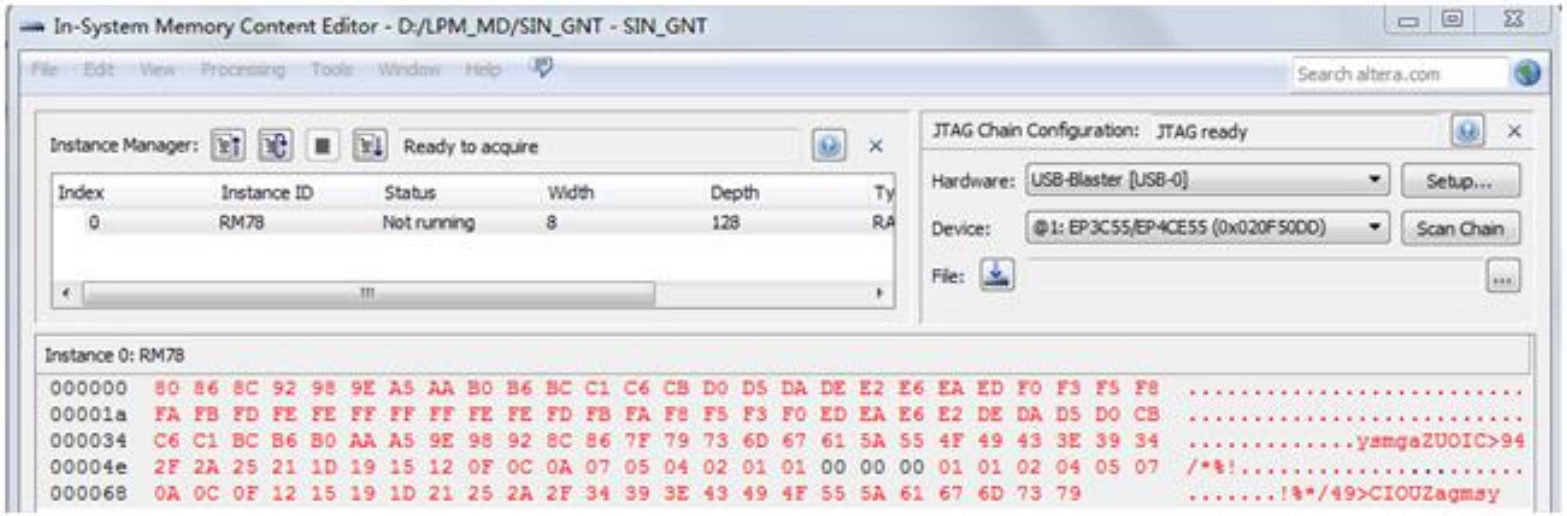

图 6-29 In-System Memory Content Editor 编辑窗，从 FPGA 中的 ROM 读取波形数据

（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。

Instance 0: RM78

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
00001a	FA	FB	FD	FE	FE	FF	FF	FF	FE	FE	FD	FB	FA	F8	F5	F3	F0	ED	EA	E6	E2	DE	DA	D5	D0	CB
000034	C6	C1	BC	B6	B0	AA	A5	9E	98	92	8C	86	7F	79	73	6D	67	61	5A	55	4F	49	43	3E	39	34
00004e	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
000068	0A	0C	0F	12	15	19	1D	21	25	2A	2F	34	39	3E	43	49	4F	55	5A	61	67	6D	73	79		

图 6-30 在此将编辑好的数据载入 FPGA 中的 ROM 内

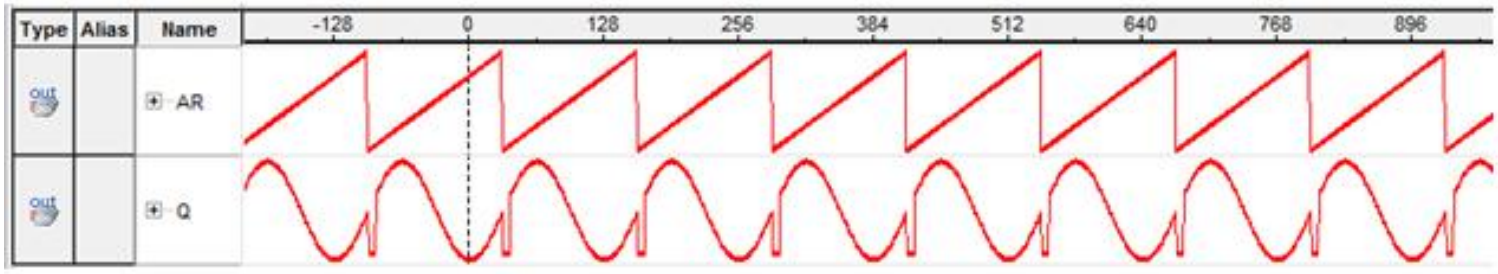

图 6-31 SignalTap II 测得的数据波形

（4）输入输出数据文件。

6.6 LPM嵌入式锁相环调用

6.6.1 建立嵌入式锁相环元件

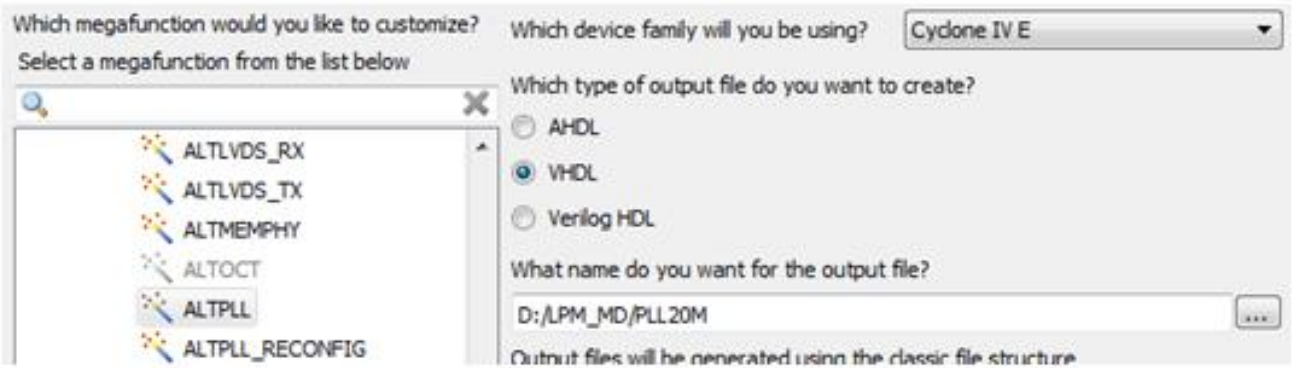

图 6-32 选择锁相环 ALTPLL

6.6 LPM嵌入式锁相环调用

6.6.1 建立嵌入式锁相环元件

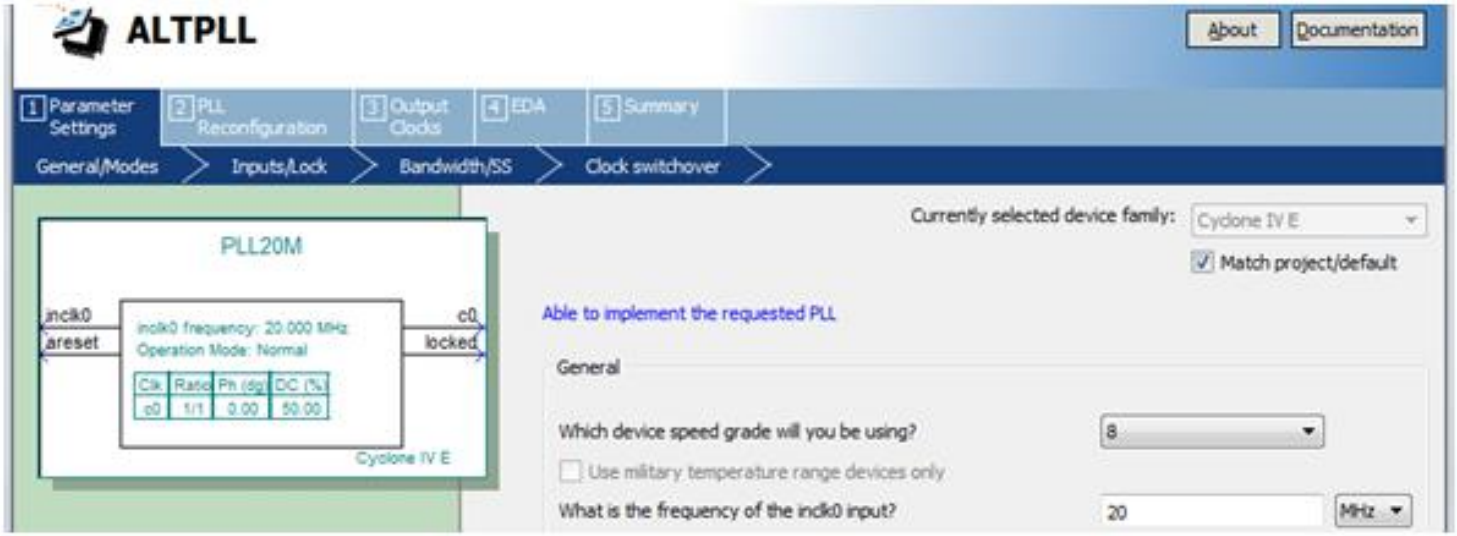

图 6-33 选择输入参考时钟 inclk0 为 20MHz

6.6 LPM嵌入式锁相环调用

6.6.1 建立嵌入式锁相环元件

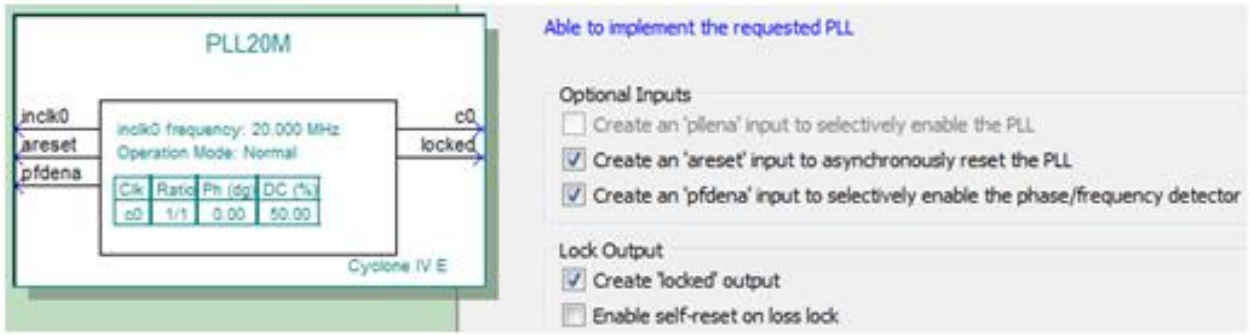

图 6-34 选择锁相环的控制信号

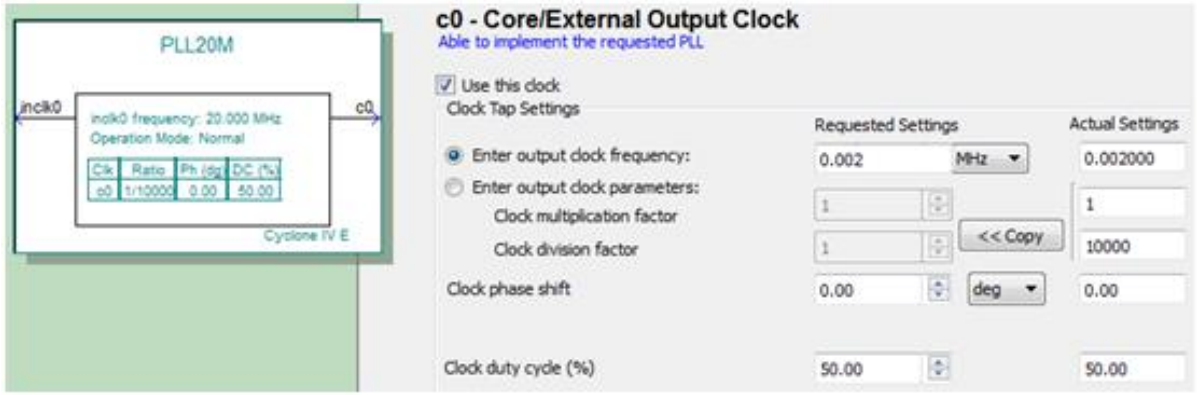

图 6-35 选择 c0 的输出频率为 0.002MHz

6.6 LPM嵌入式锁相环调用

6.6.1 建立嵌入式锁相环元件

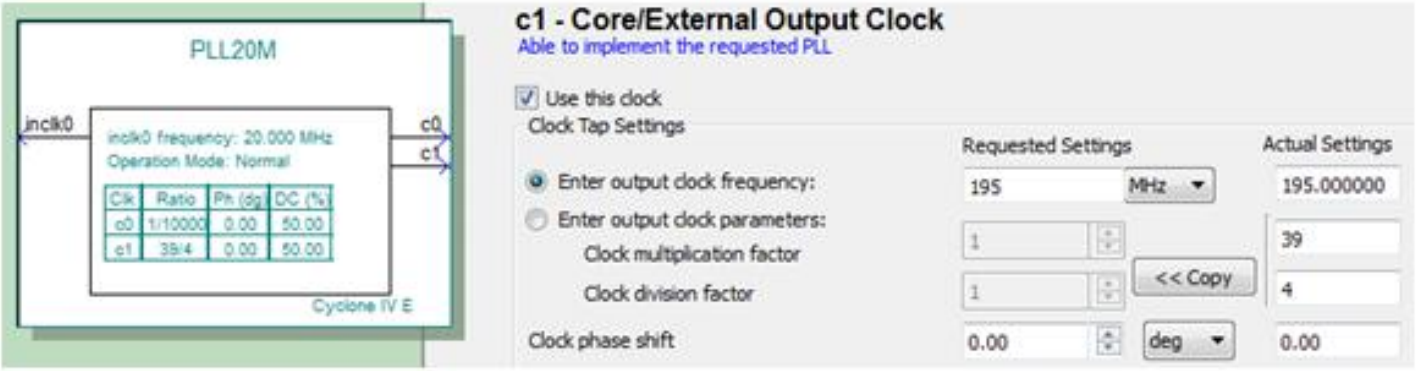

图 6-36 输出第二个时钟信号 c1

6.6 LPM嵌入式锁相环调用

6.6.1 建立嵌入式锁相环元件

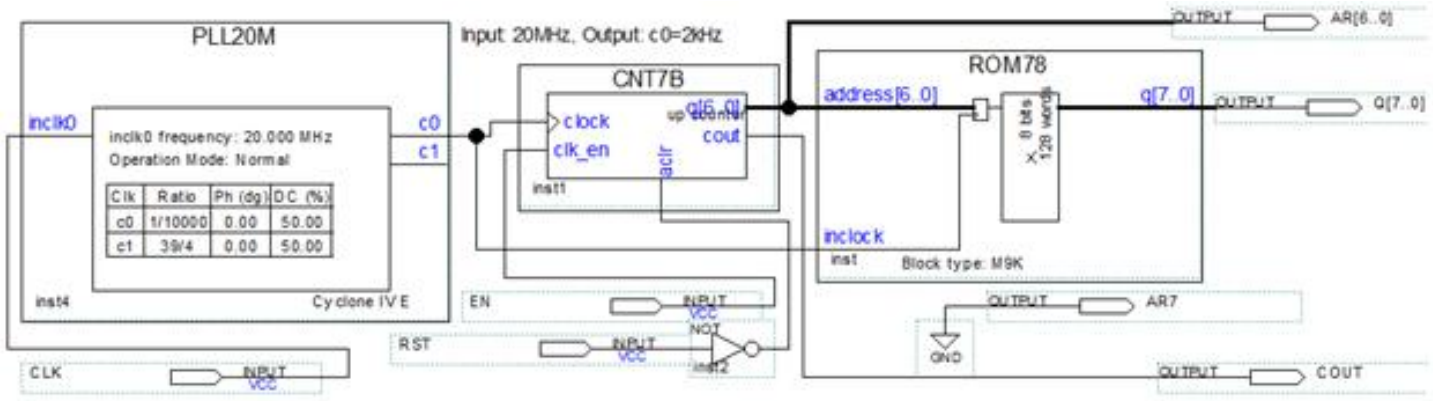

图 6-37 采用了嵌入式锁相环作时钟的正弦信号发生器电路

6.6.2 测试锁相环

6.7 In-System Sources and Probes Editor用法

（1）在顶层设计中嵌入**In-System Sources and Probes**模块。

（2）设定参数。

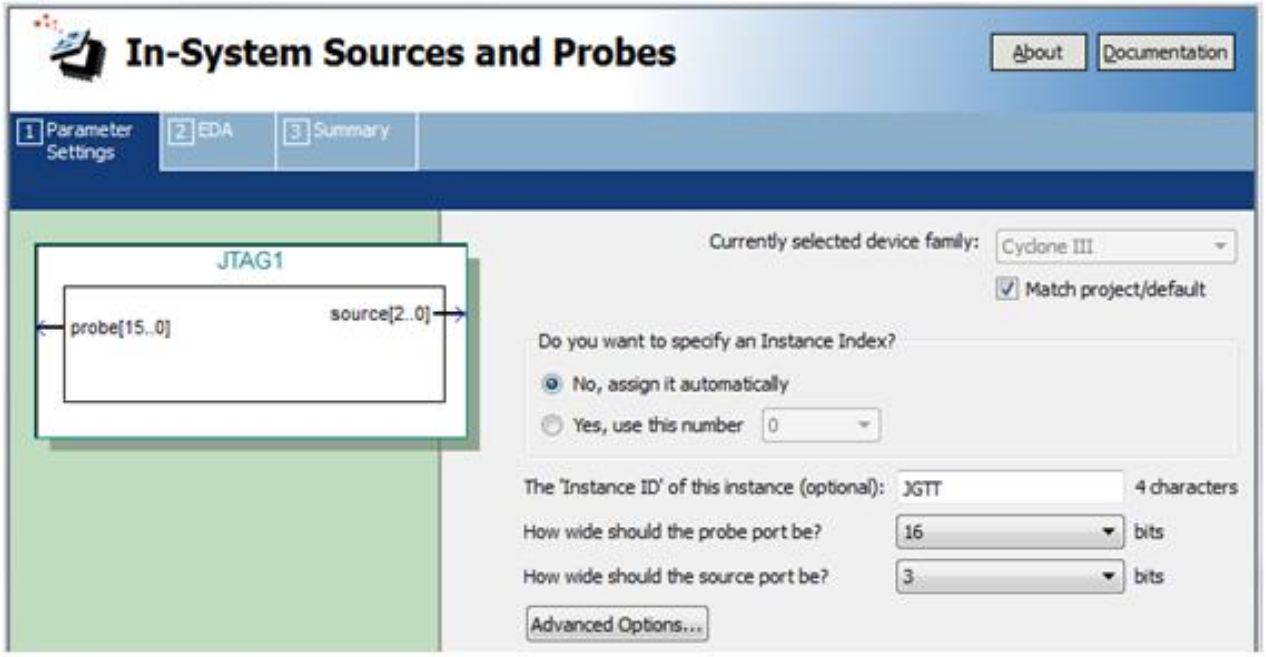

图 6-38 为 In-System Sources and Probes 模块设置参数

6.7 In-System Sources and Probes Editor用法

（3）与需要测试的电路系统连接好。

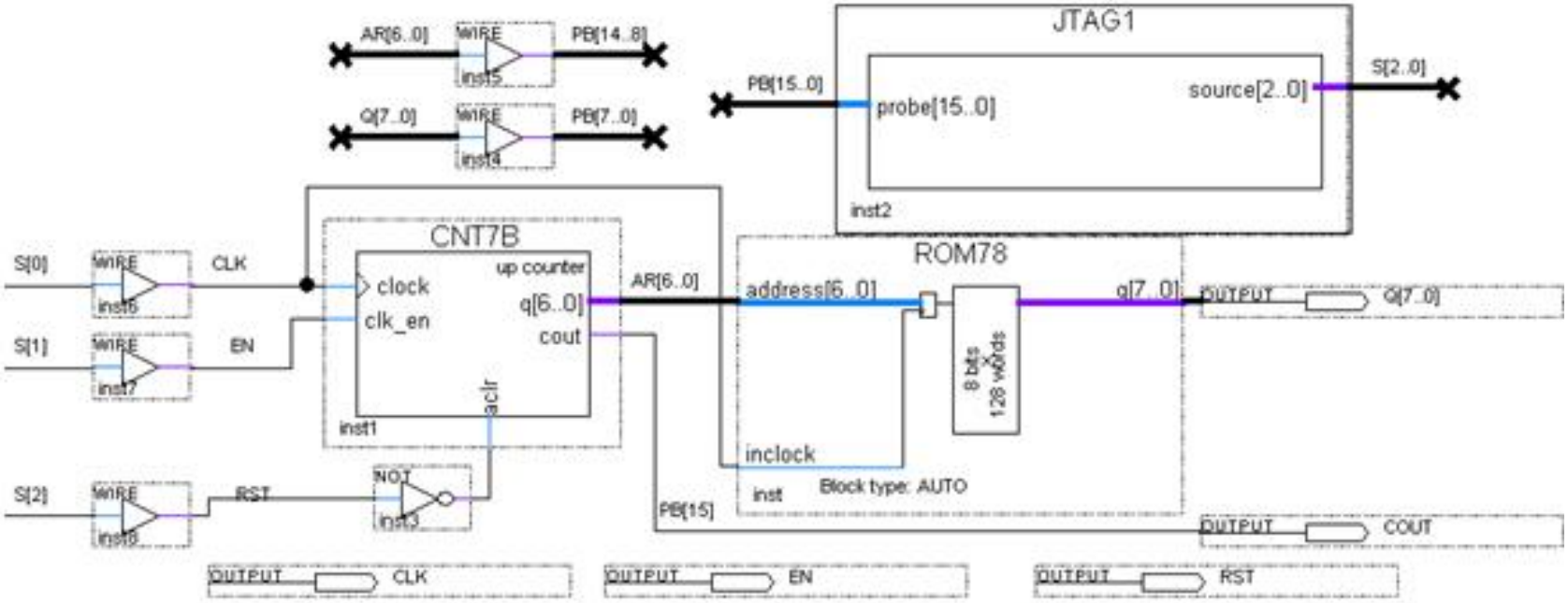

图 6-39 在电路中加入 In-System Sources and Probes 测试模块

6.7 In-System Sources and Probes Editor用法

（4）调用In-System Sources and Probes Editor。

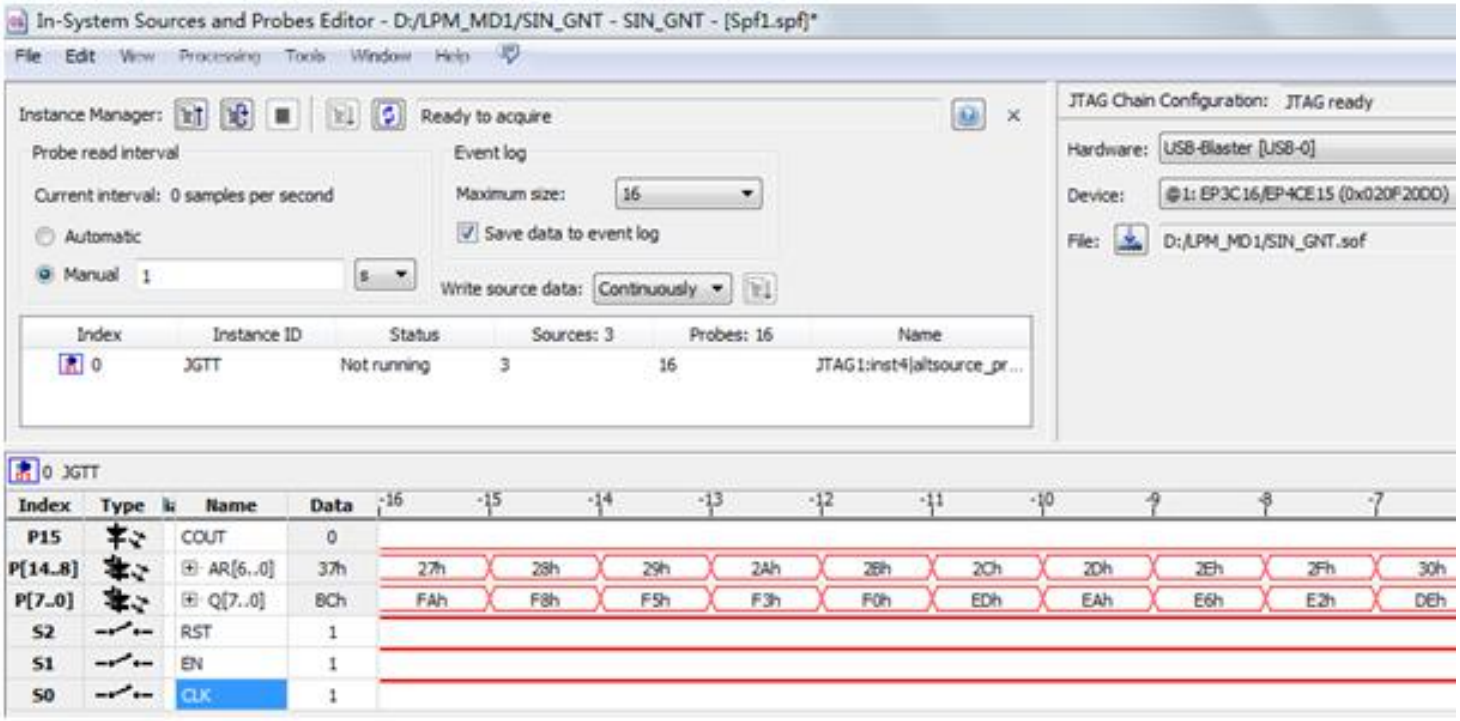

图 6-40 In-System Sources and Probes Editor 的测试情况

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

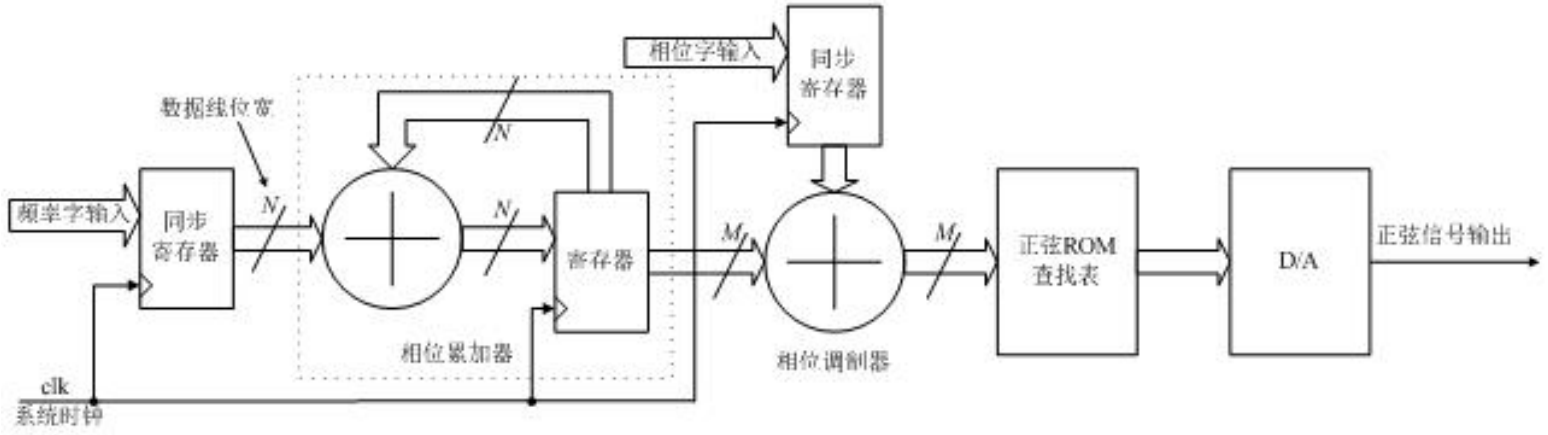

图 6-41 基本 DDS 结构

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

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

6.8 DDS实现原理与应用

6.8.2 DDS信号发生器设计示例

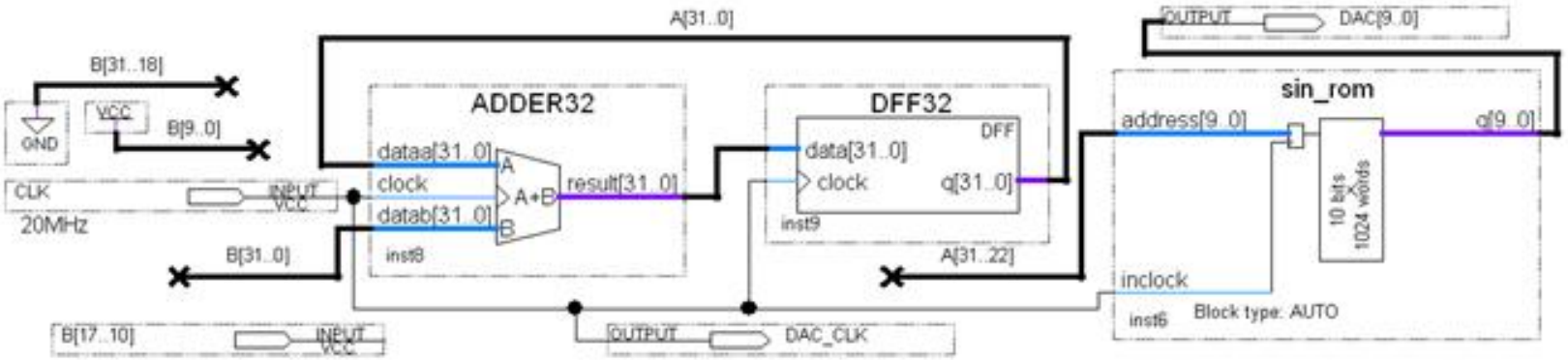

图 6-42　DDS 信号发生器电路顶层原理图

6.8 DDS实现原理与应用

6.8.2 DDS信号发生器设计示例

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

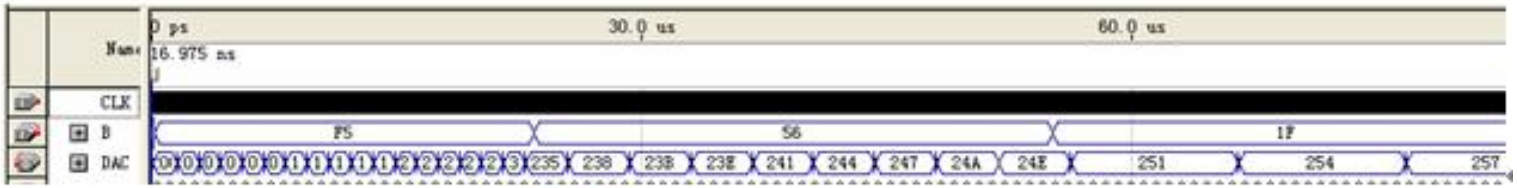

图 6-43　图 6-42 的仿真波形

实验与设计

实验6-1 查表式硬件运算器设计

实验6-2 正弦信号发生器设计

实验与设计

实验6-3 简易逻辑分析仪设计

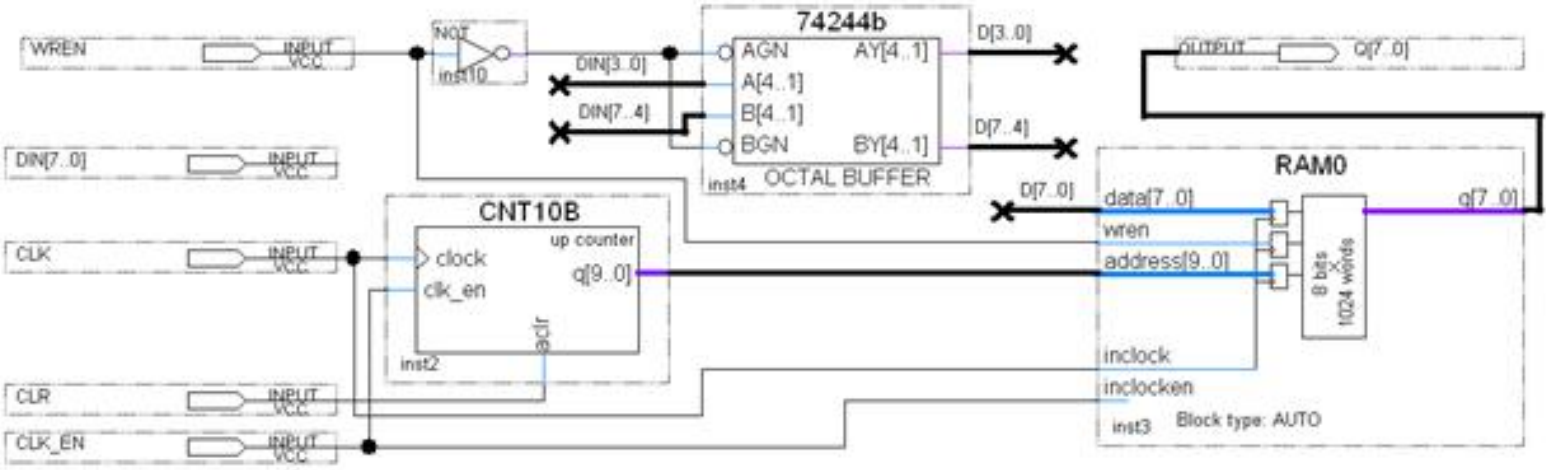

图 6-44 逻辑数据采样电路顶层设计

实验与设计

实验6-3 简易逻辑分析仪设计

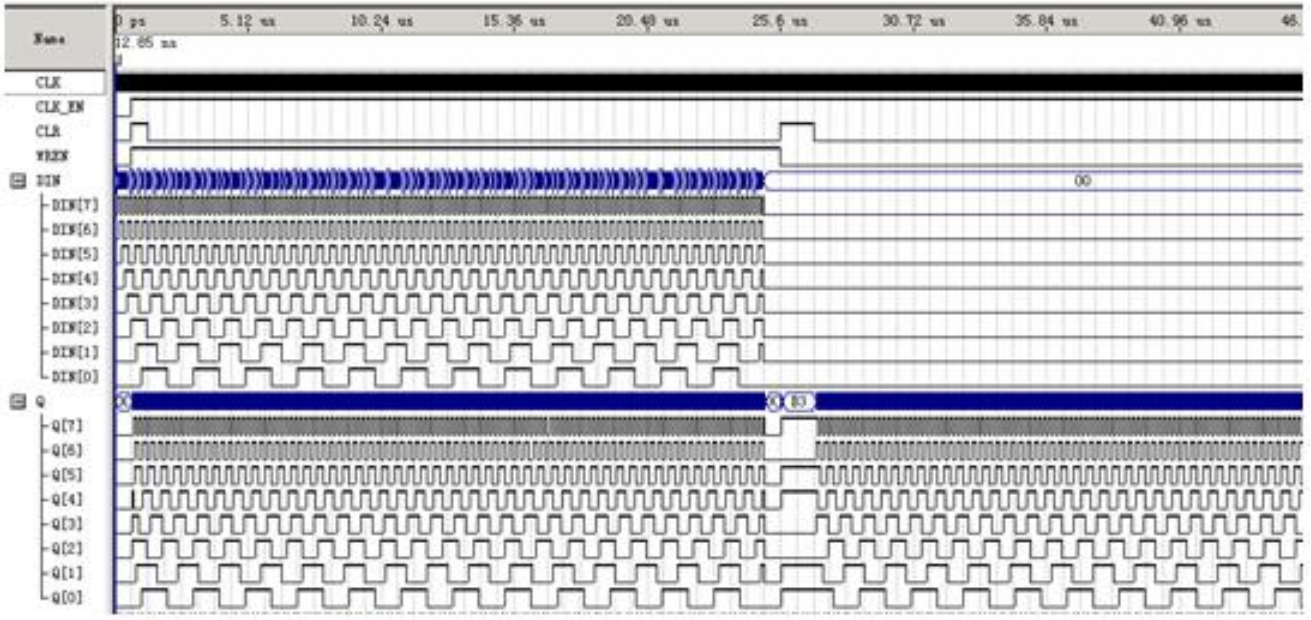

图 6-45 逻辑数据采样电路时序仿真波形

实验与设计

实验6-4 DDS正弦信号发生器设计

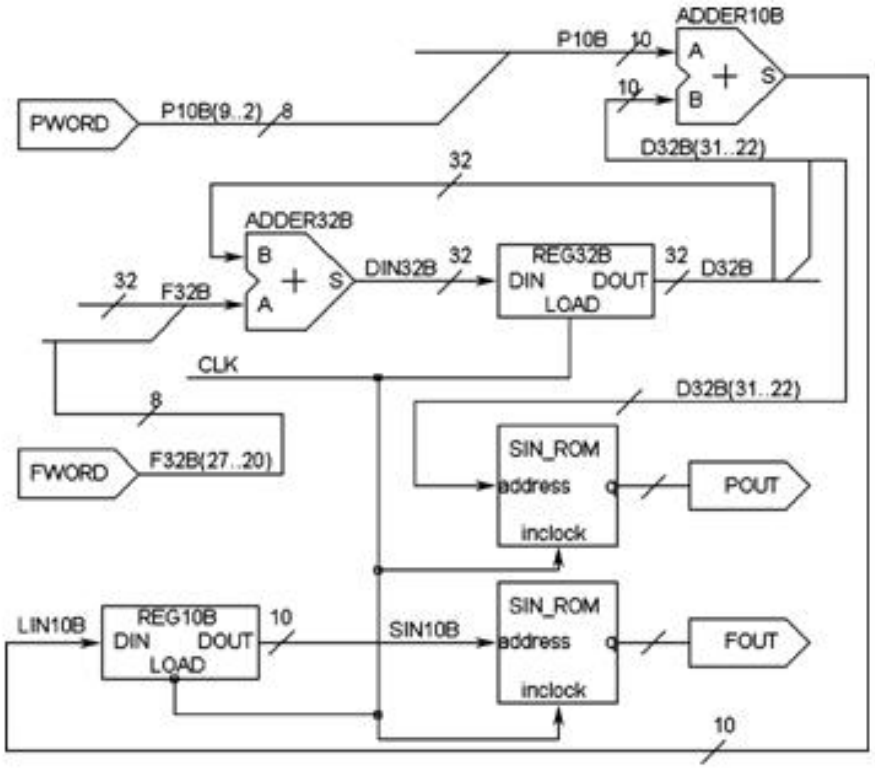

图 6-46 DDS 正弦信号发生器顶层原理图

实验6-5 移相信号发生器设计

实验6-6 VGA简单图像显示控制模块设计

【例 6-6】

```
LIBRARY ieee;                                  --图像显示顶层程序
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                               --图像数据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

实验6-6 VGA简单图像显示控制模块设计

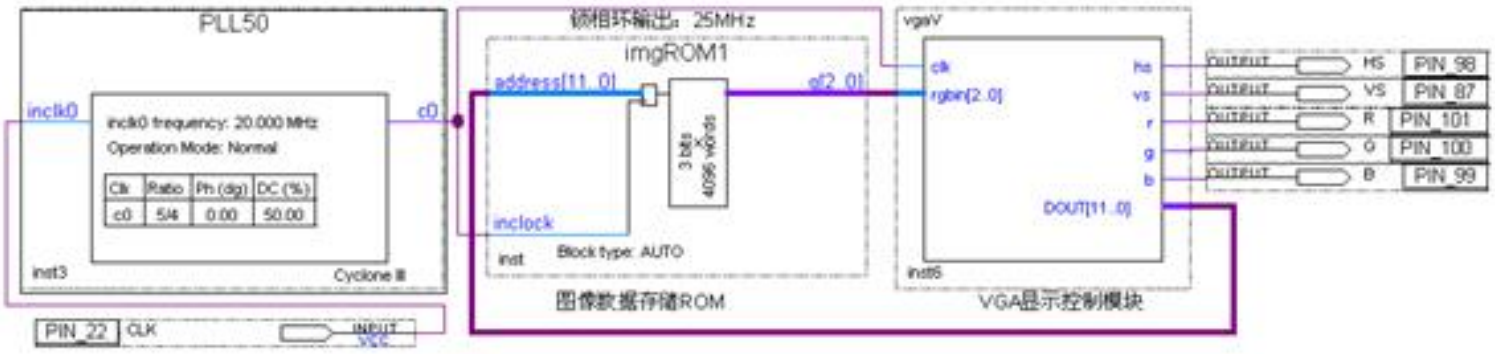

图 6-47 VGA 图像显示控制模块原理图

实验与设计

实验6-7 AM幅度调制信号发生器设计

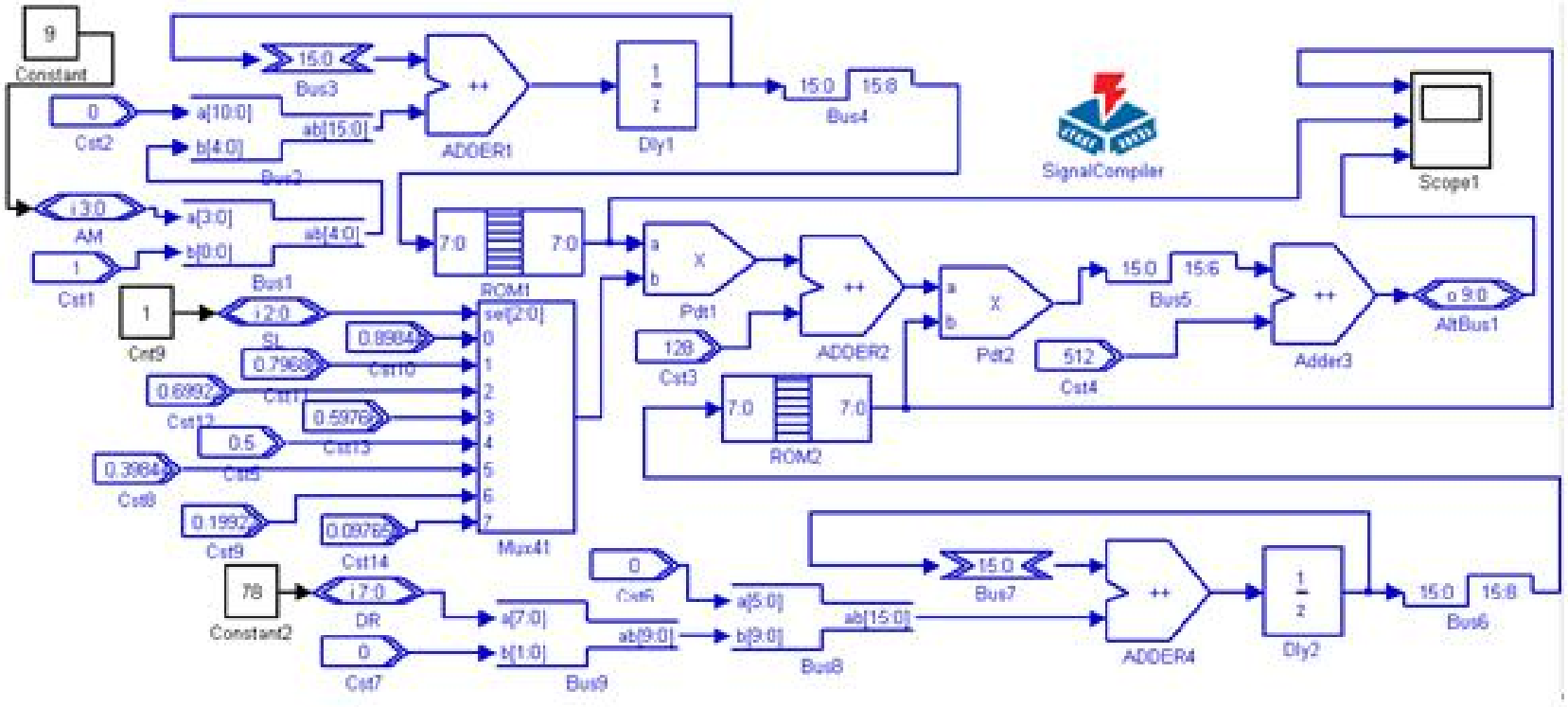

图 6-48 AM 信号发生器 DSP Builder/MATLAB Simulink 模型

实验与设计

实验6-7 AM幅度调制信号发生器设计

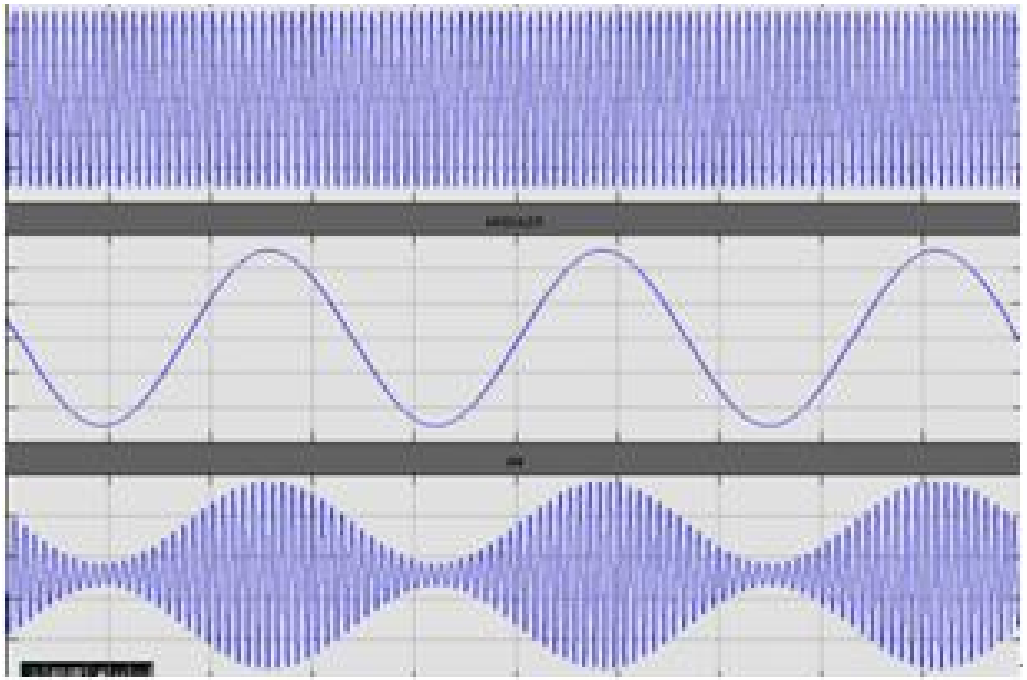

图 6-49 AM 模型仿真波形