EDA技术实用教程

第6章

宏功能模块与IP应用

6.1 宏功能模块概述

6.1.1 知识产权核的应用

AMPP程序

MegaCore函数

OpenCore评估功能

OpenCore Plus硬件评估功能

6.1 宏功能模块概述

6.1.2 使用MegaWizard Plug-In Manager

- <输出文件>.bsf
- <输出文件>.cmp
- <输出文件>.inc
- <输出文件>.tdf
- <输出文件>.vhd
- <输出文件>.v
- <输出文件>_ bb.v

- <输出文件>_ inst.tdf
- <输出文件>_ inst.vhd
- <输出文件>_ inst.v

6.1.3 在Quartus II中对宏功能模块进行例化

6.2 LPM计数器模块使用方法

6.2.1 LPM_COUNTER计数器模块文本文件的调用

图 6-1　定制新的宏功能块

6.2 LPM计数器模块使用方法

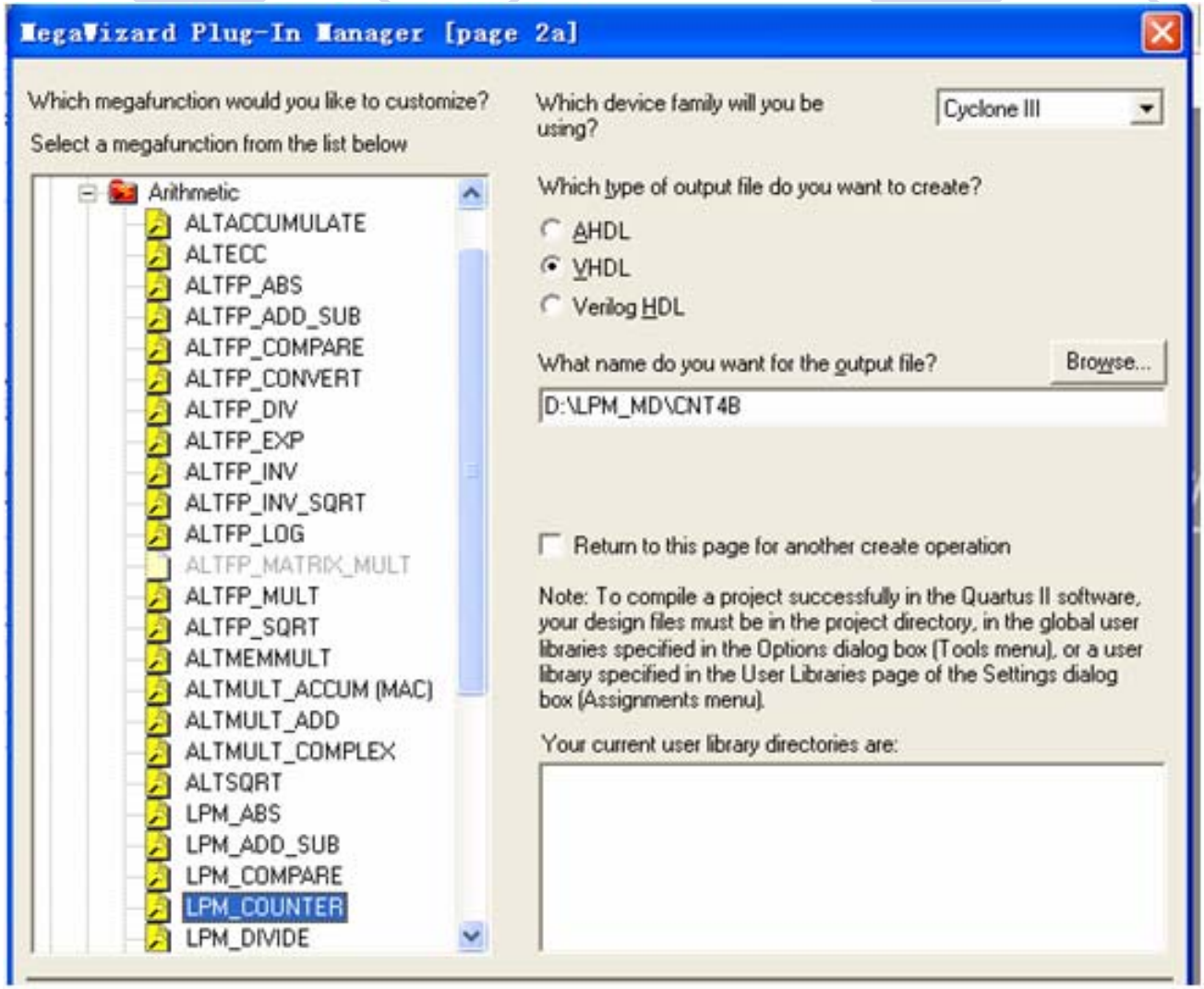

图 6-2　LPM 宏功能块设定

6.2 LPM计数器模块使用方法

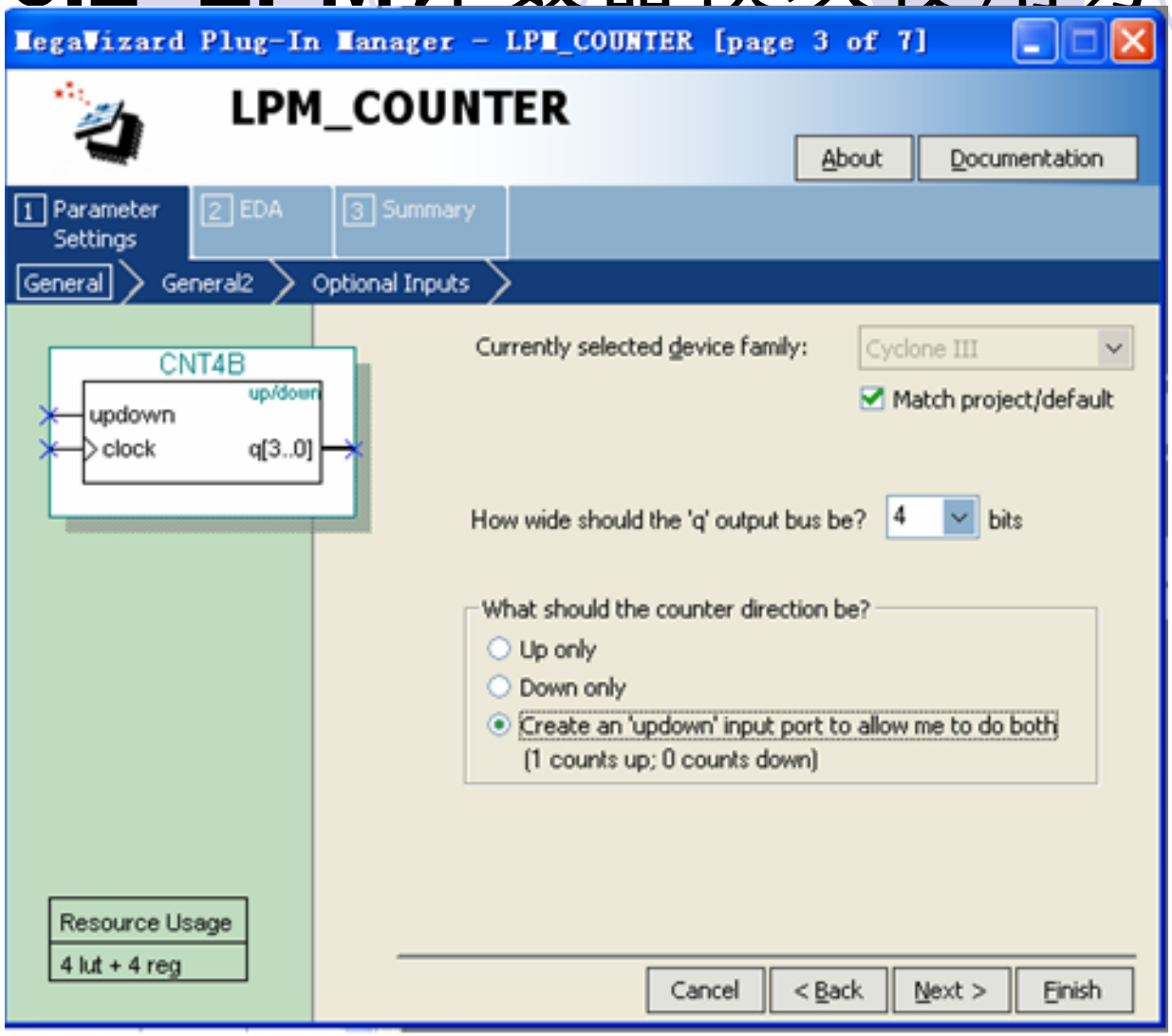

图 6-3 设 4 位可加减计数器

6.2 LPM计数器模块使用方法

6.2.1 LPM_COUNTER计数器模块文本文件的调用

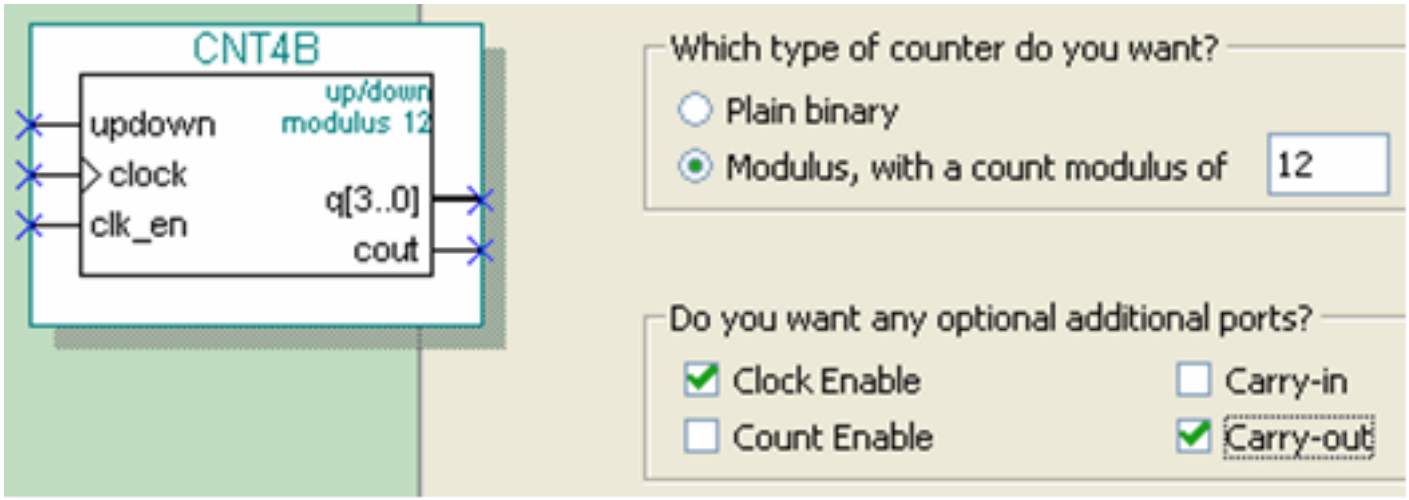

图 6-4 设定模 12 计数器，含时钟使能和进位输出

6.2 LPM计数器模块使用方法

6.2.1 LPM_COUNTER计数器模块文本文件的调用

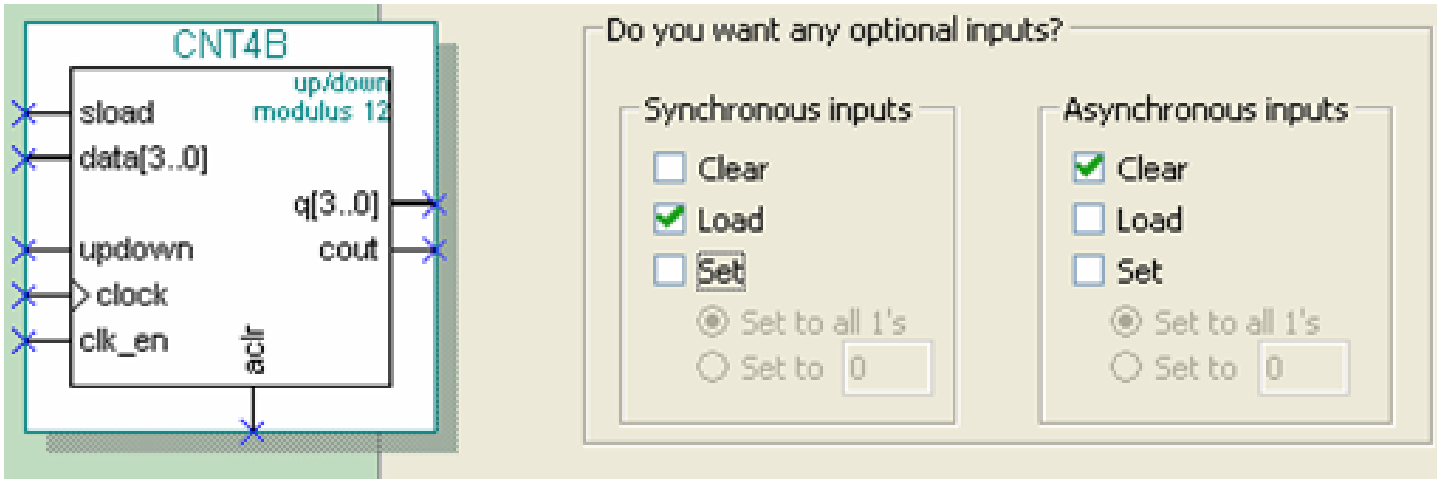

图 6-5 加入 4 位并行数据预置功能

6.2 LPM计数器模块使用方法

【例 6-1】 Quartus II 生成的计数器文件 CNT4B.vhd

```
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;--定义字符串类型
                  lpm_modulus, lpm_width : NATURAL );  --定义正整数类型
```

6.2 LPM计数器模块使用方法

接上页

```
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;
```

【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;
```

6.2 LPM计数器模块使用方法

6.2.2 创建工程与仿真测试

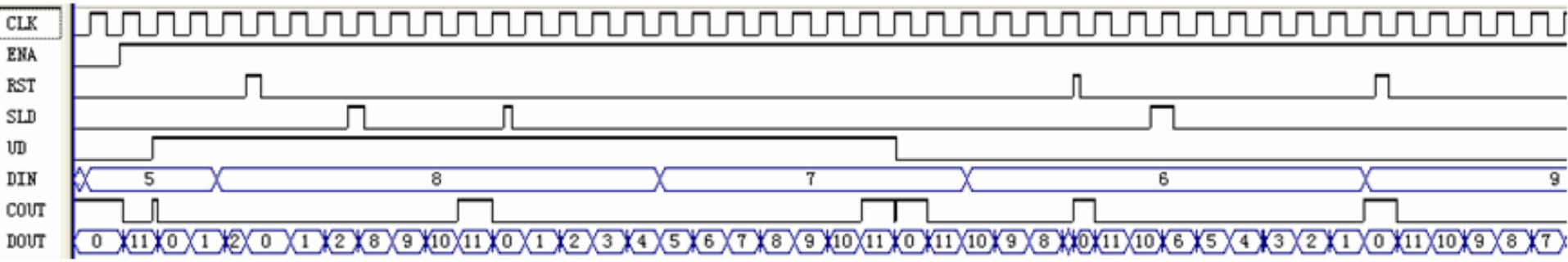

图 6-6 CNT4BIT.v 的仿真波形

6.2 LPM计数器模块使用方法

6.2.2 创建工程与仿真测试

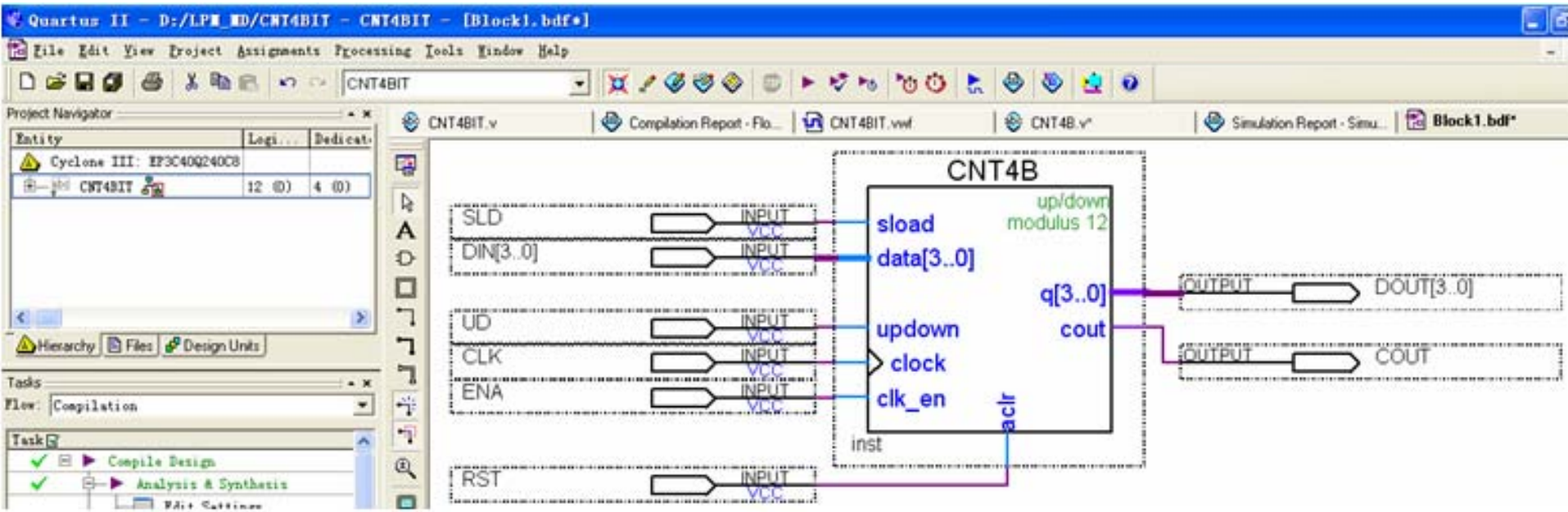

图 6-7　原理图输入设计

6.3 基于LPM的流水线乘法累加器设计

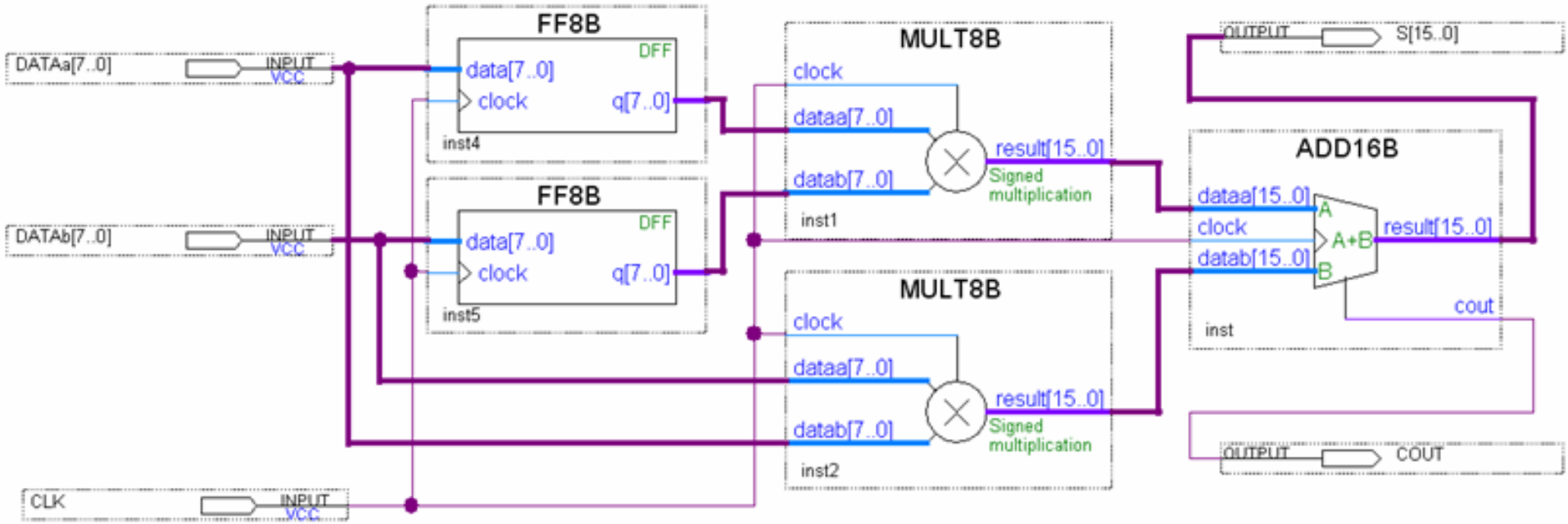

图 6-8　8 位乘法累加器顶层设计

6.3 基于LPM的流水线乘法累加器设计

6.3.1 LPM加法器模块设置调用

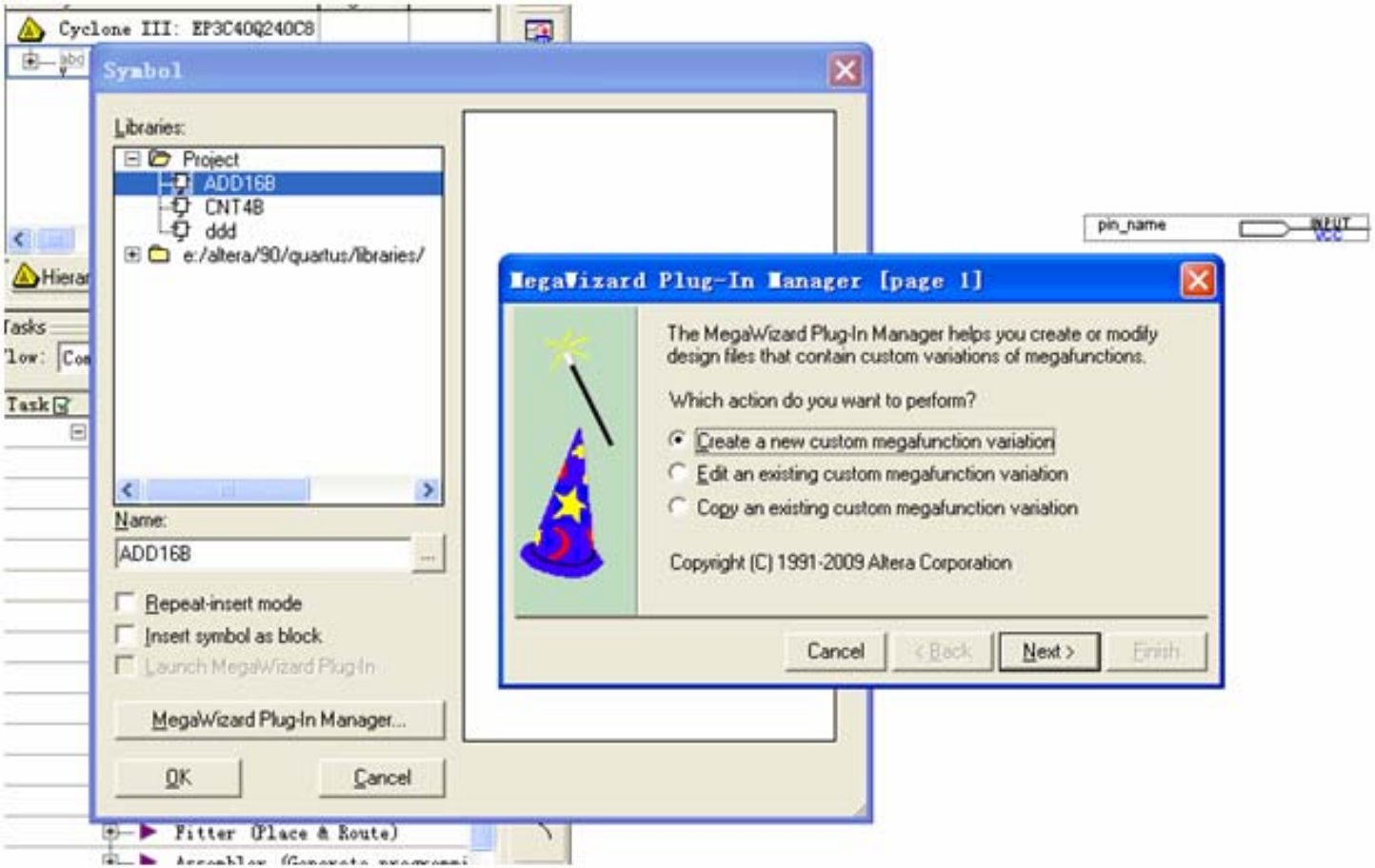

图 6-9　从原理图编辑窗进入 MegaWizard Plug-In Manager 管理器

6.3 基于LPM的流水线乘法累加器设计

6.3.1 LPM加法器模块设置调用

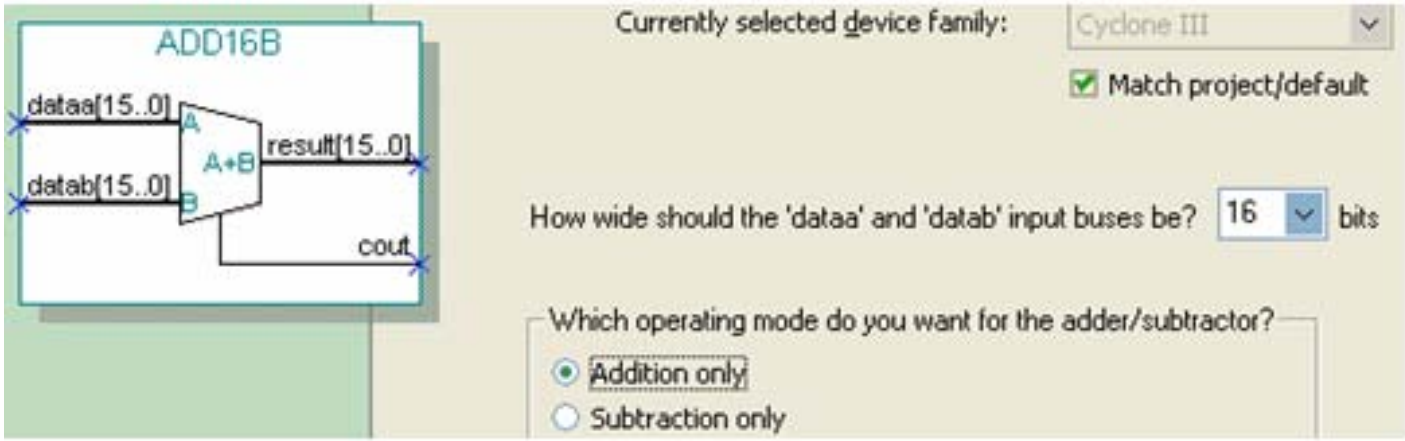

图 6-10 选择 16 位加法工作方式

6.3 基于LPM的流水线乘法累加器设计

6.3.1 LPM加法器模块设置调用

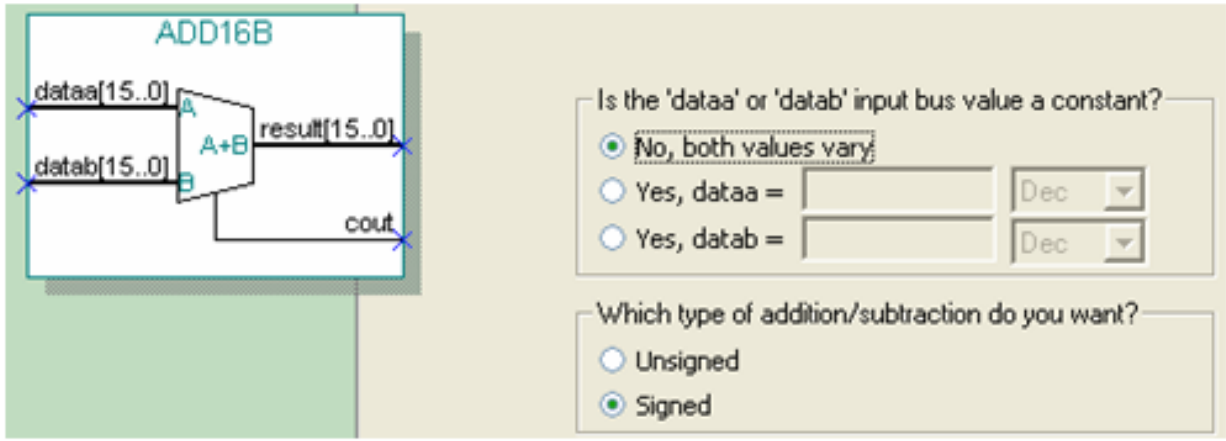

图 6-11 选择有符号加法操作类型输入

6.3 基于LPM的流水线乘法累加器设计

6.3.1 LPM加法器模块设置调用

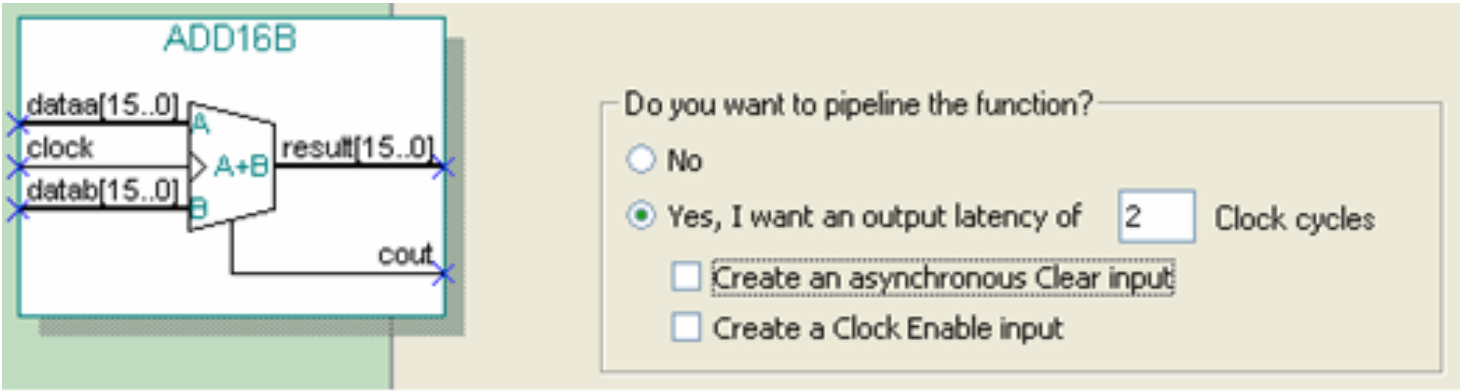

图 6-12　选择流水线方式

6.3 基于LPM的流水线乘法累加器设计

6.3.2 LPM乘法器模块设置调用

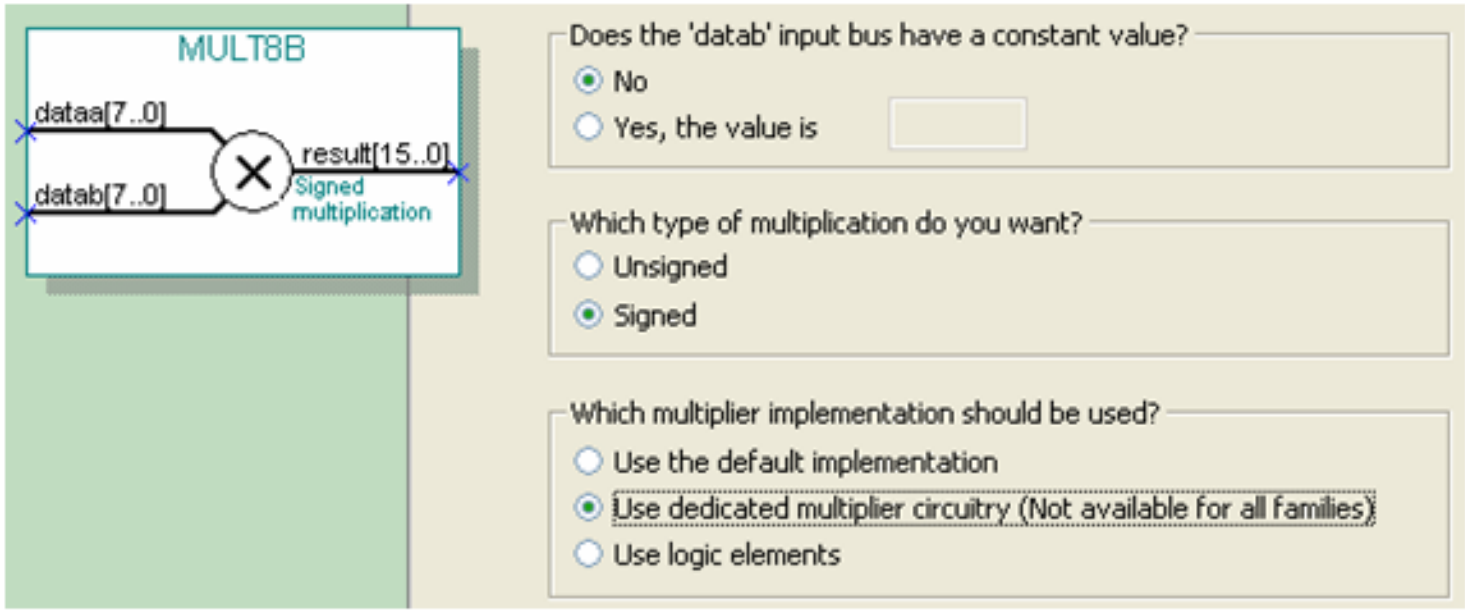

图 6-13　选择有符号乘法模式，并用专用乘法器模块构建乘法器

6.3 基于LPM的流水线乘法累加器设计

6.3.2 LPM乘法器模块设置调用

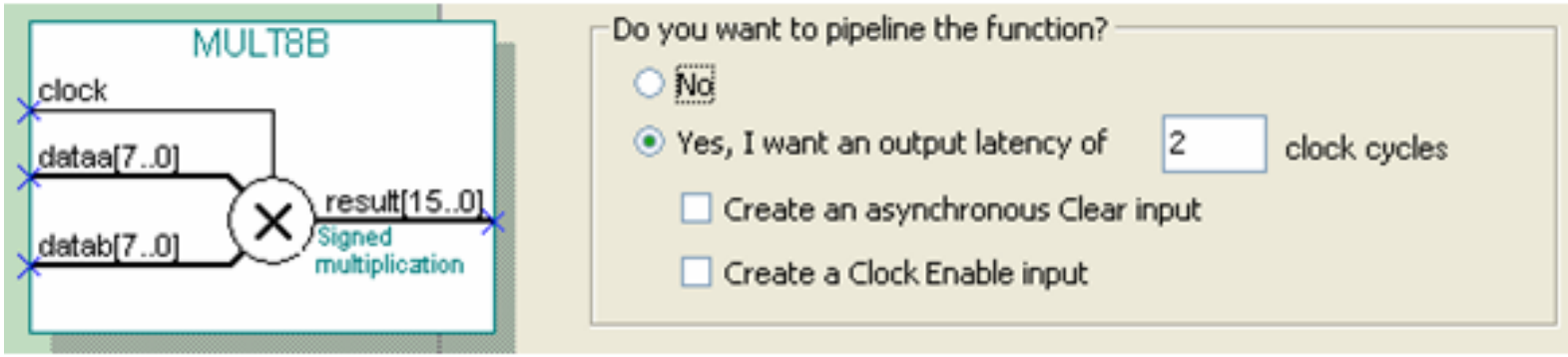

图 6-14　选择 2 级流水线乘法模式

6.3 基于LPM的流水线乘法累加器设计

6.3.3 乘法累加器的仿真测试

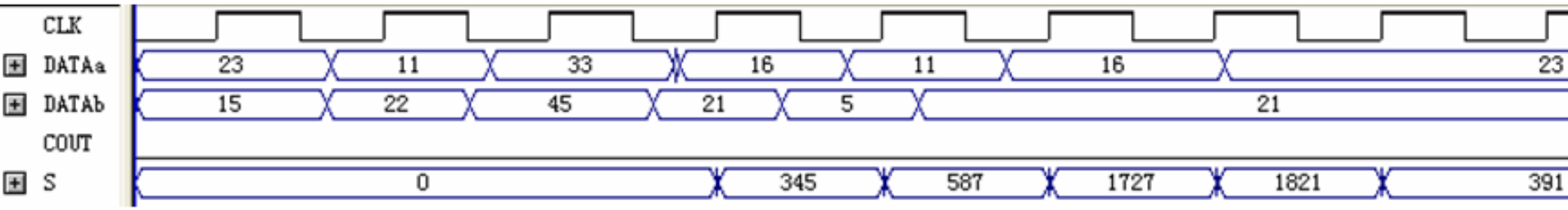

图 6-15　电路图图 6-8 的 MULTADD 工程仿真波形

6.3 基于LPM的流水线乘法累加器设计

6.3.3 乘法累加器的仿真测试

Device	EP3C40Q240C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	50 / 39,600 (< 1 %)
Total combinational functions	17 / 39,600 (< 1 %)
Dedicated logic registers	50 / 39,600 (< 1 %)
Total registers	50
Total pins	34 / 129 (26 %)
Total virtual pins	0
Total memory bits	0 / 1,161,216 (0 %)
Embedded Multiplier 9-bit elements	2 / 252 (< 1 %)
Total PLLs	0 / 4 (0 %)

Device	EP3C40Q240C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	238 / 39,600 (< 1 %)
Total combinational functions	212 / 39,600 (< 1 %)
Dedicated logic registers	188 / 39,600 (< 1 %)
Total registers	188
Total pins	34 / 129 (26 %)
Total virtual pins	0
Total memory bits	0 / 1,161,216 (0 %)
Embedded Multiplier 9-bit elements	0 / 252 (0 %)
Total PLLs	0 / 4 (0 %)

图 6-16　对乘法器的构建模式选择不同设置后的编译报告

6.3 基于LPM的流水线乘法累加器设计

6.3.3 乘法累加器的仿真测试

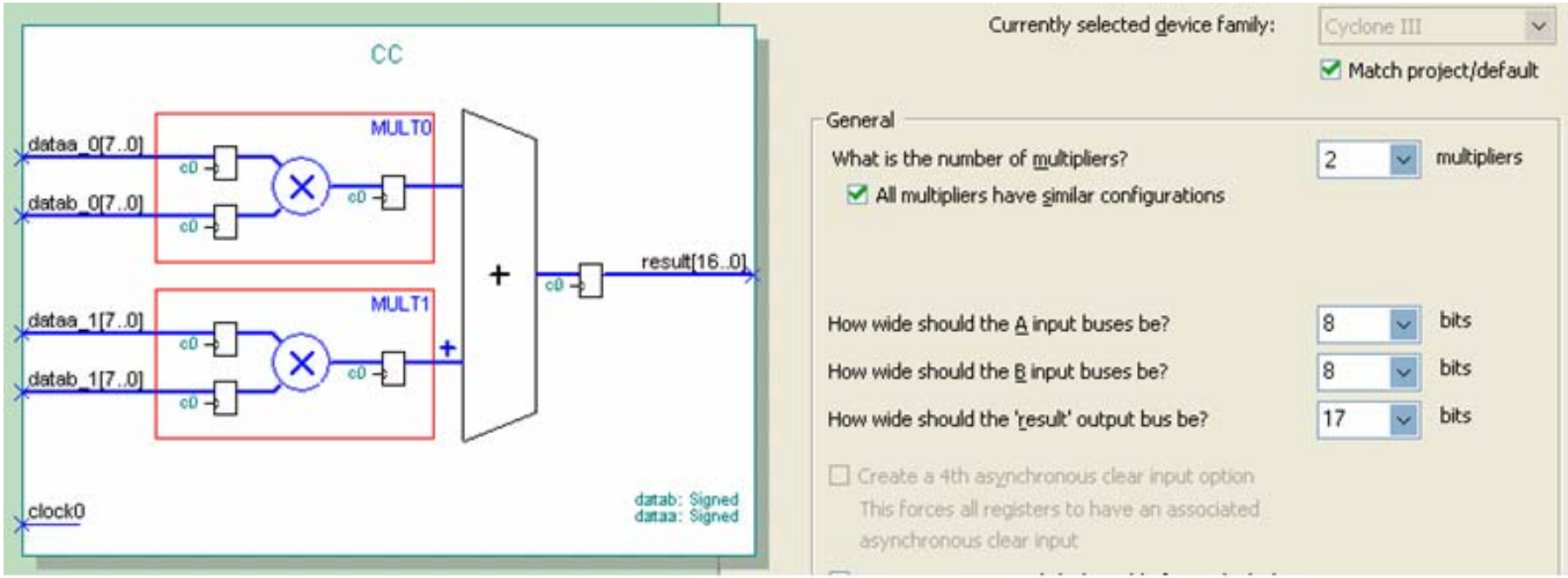

图 6-17　ALTMULT_ADD 模块设置对话框

6.3 基于LPM的流水线乘法累加器设计

6.3.4 乘法器的VHDL文本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 ;
USE IEEE.STD_LOGIC_SIGNED.ALL ;
ENTITY MULT8 IS
PORT (A1,B1,A2,B2 : IN  SIGNED(7 DOWNTO 0) ;--定义有符号位矢，参考第 9 章
            R1,R2 : OUT SIGNED(15 DOWNTO 0) ) ;
END ;
ARCHITECTURE bhv OF MULT8 IS
BEGIN
    R1 <= A1 * B1  ;      R2 <= A2 * B2  ;
END bhv;
```

6.3 基于LPM的流水线乘法累加器设计

6.3.4 乘法器的VHDL文本表述和相关属性设置

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	64 / 95 (67 %)
Total virtual pins	0
Total memory bits	0 / 423,936 (0 %)
Embedded Multiplier 9-bit elements	2 / 46 (4 %)
Total PLLs	0 / 2 (0 %)

图 6-18 例 6-3 的编译报告

6.3 基于LPM的流水线乘法累加器设计

6.3.4 乘法器的VHDL文本表述和相关属性设置

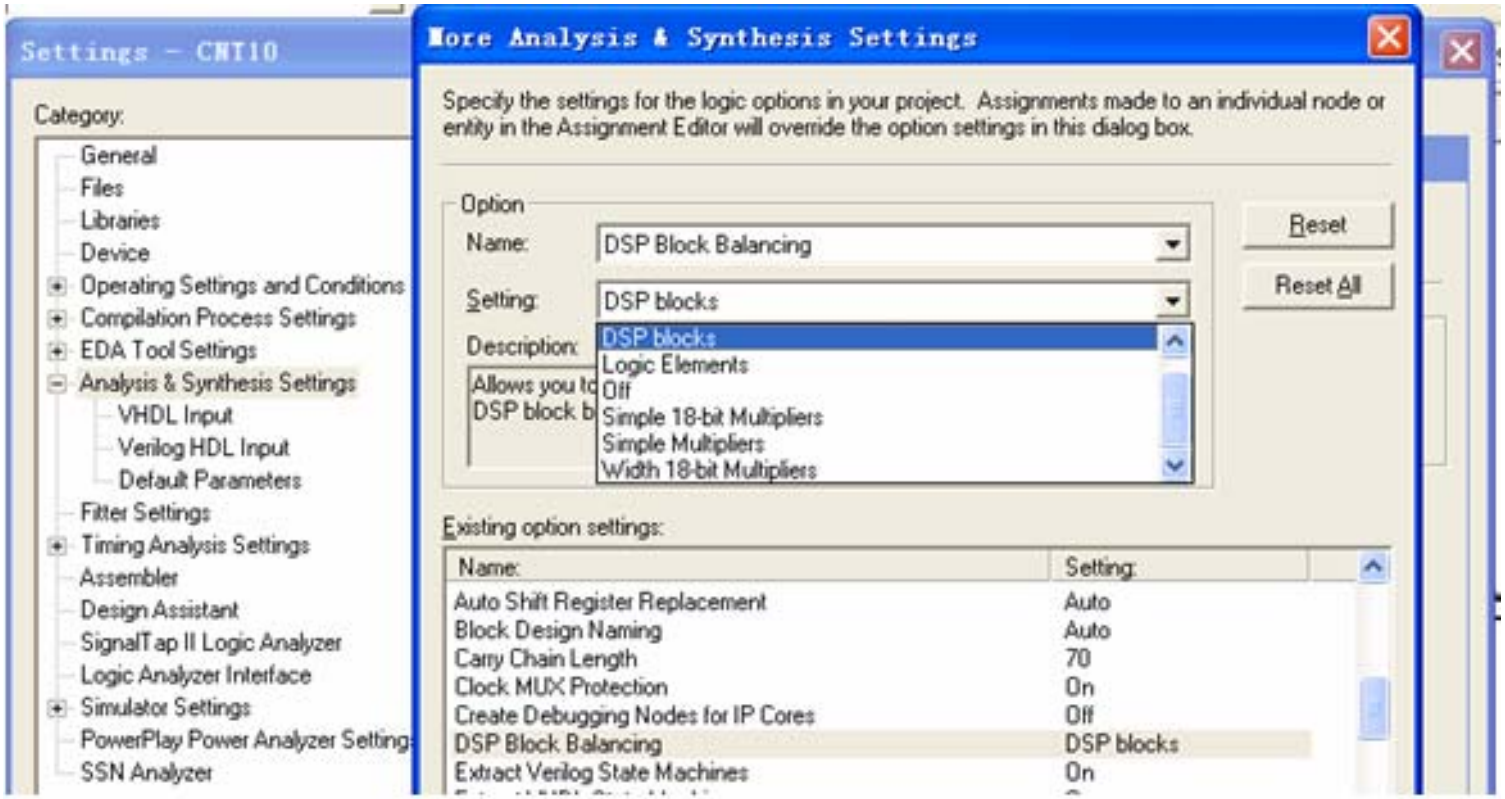

图 6-19　设置乘法器用 DSP 模块构建

6.3 基于LPM的流水线乘法累加器设计

6.3.4 乘法器的VHDL文本表述和相关属性设置

【例 6-4】

```
ARCHITECTURE bhv OF MULT8 IS
 attribute multstyle : string;
 attribute multstyle of R1,R2 : signal is "DSP";
 BEGIN
    R1 <= A1 * B1  ;      R2 <= A2 * B2  ;
END bhv;
```

6.4 LPM 随机存储器的设置和调用

6.4.1 存储器初始化文件生成

1. 建立.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 6-20 mif 文件编辑窗

6.4 LPM 随机存储器的设置和调用

（2）文件编辑法

【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;
```

6.4 LPM 随机存储器的设置和调用

6.4.1 存储器初始化文件生成

1. 建立.mif格式文件

（3）C等软件生成

【例 6-6】

```
#include <stdio.h>
#include "math.h"
main()
{int i;float s;
for(i=0;i<1024;i++)
    { s = sin(atan(1)*8*i/1024);
      printf("%d : %d;\n",i,(int)((s+1)*1023/2));
    }     }
```

6.4 LPM 随机存储器的设置和调用

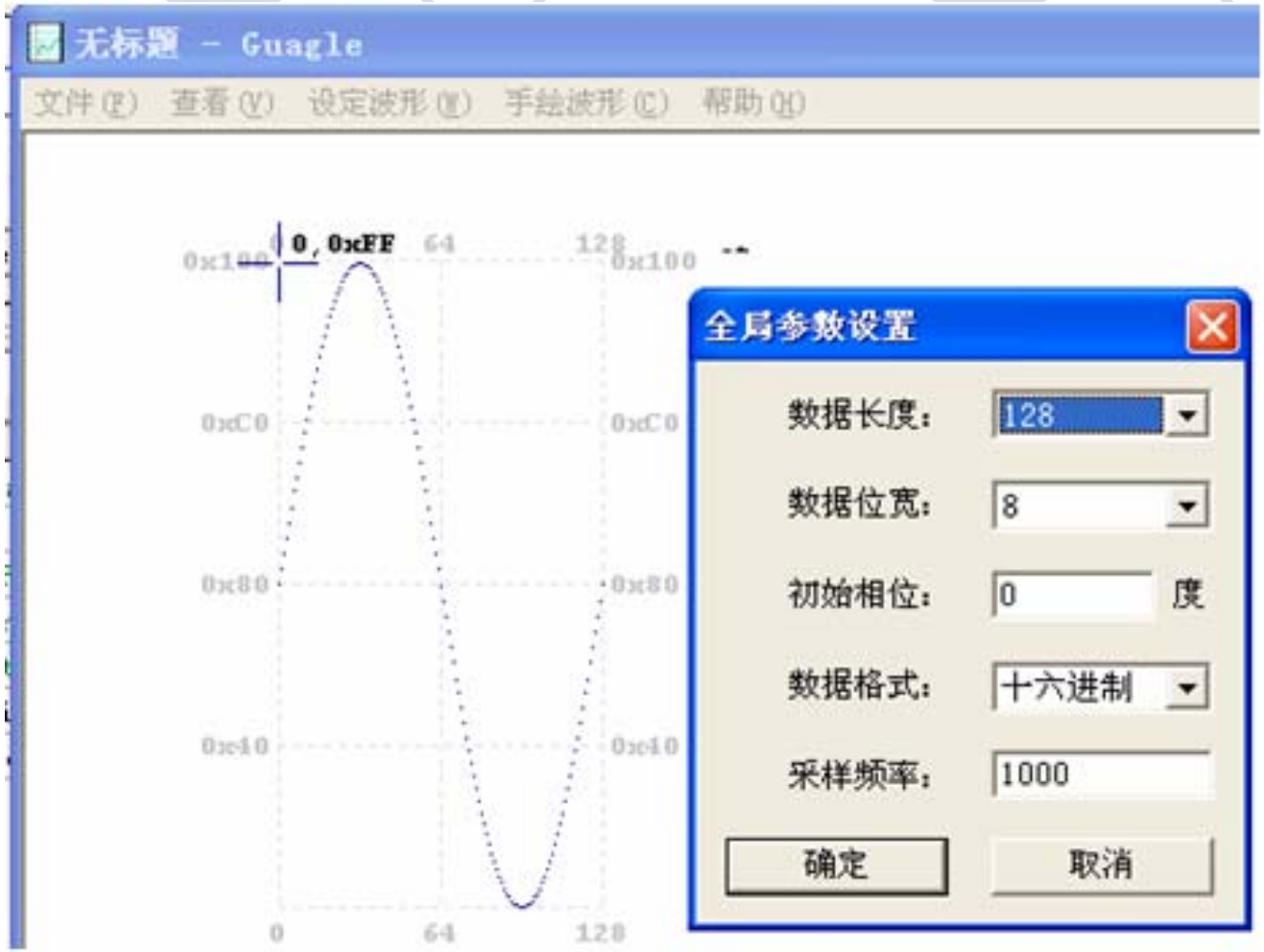

图 6-22　利用康芯 mif 生成器生成 mif 正弦波数据文件

6.4 LPM 随机存储器的设置和调用

6.4.1 存储器初始化文件生成

1. 建立.mif格式文件

（4）专用mif文件生成器

data7x8.mif - 记事本

文件(F) 编辑(E) 格式(O) 查

```
DEPTH = 128;
WIDTH = 8;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN
0000 : 0080;
0001 : 0086;
0002 : 008C;
0003 : 0092;
0004 : 0098;
0005 : 009E;
...
007B : 0061;
007C : 0067;
007D : 006D;
007E : 0073;
007F : 0079;
END ;
```

图 6-23 打开 mif 文件

6.4 LPM 随机存储器的设置和调用

6.4.1 存储器初始化文件生成

2．建立.hex格式文件

```
ORG  0000H
DB   255 , 254, 252 , 249
DB   245 , 239 , 233 , 225
DB   217 , 207 , 197 , 186
DB   174 , 162 , 150 , 137
DB   124 , 112 ,  99 ,  87
DB    75 ,  64 ,  53 ,  43
DB    34 ,  26 ,  19 ,  13
DB     8 ,   4 ,   1 ,   0
DB     0 ,   1 ,   4 ,   8
DB    13 ,  19 ,  26 ,  34
DB    43 ,  53 ,  64 ,  75
DB    87 ,  99 , 112 , 124
DB   137 , 150 , 162 , 174
DB   186 , 197 , 207 , 217
DB   225 , 233 , 239 , 245
DB   249 , 252 , 254 , 255
END
```

图 6-21 用汇编器生成 hex 文件

6.4 LPM 随机存储器的设置和调用

6.4.2 LPM_RAM的设置和调用

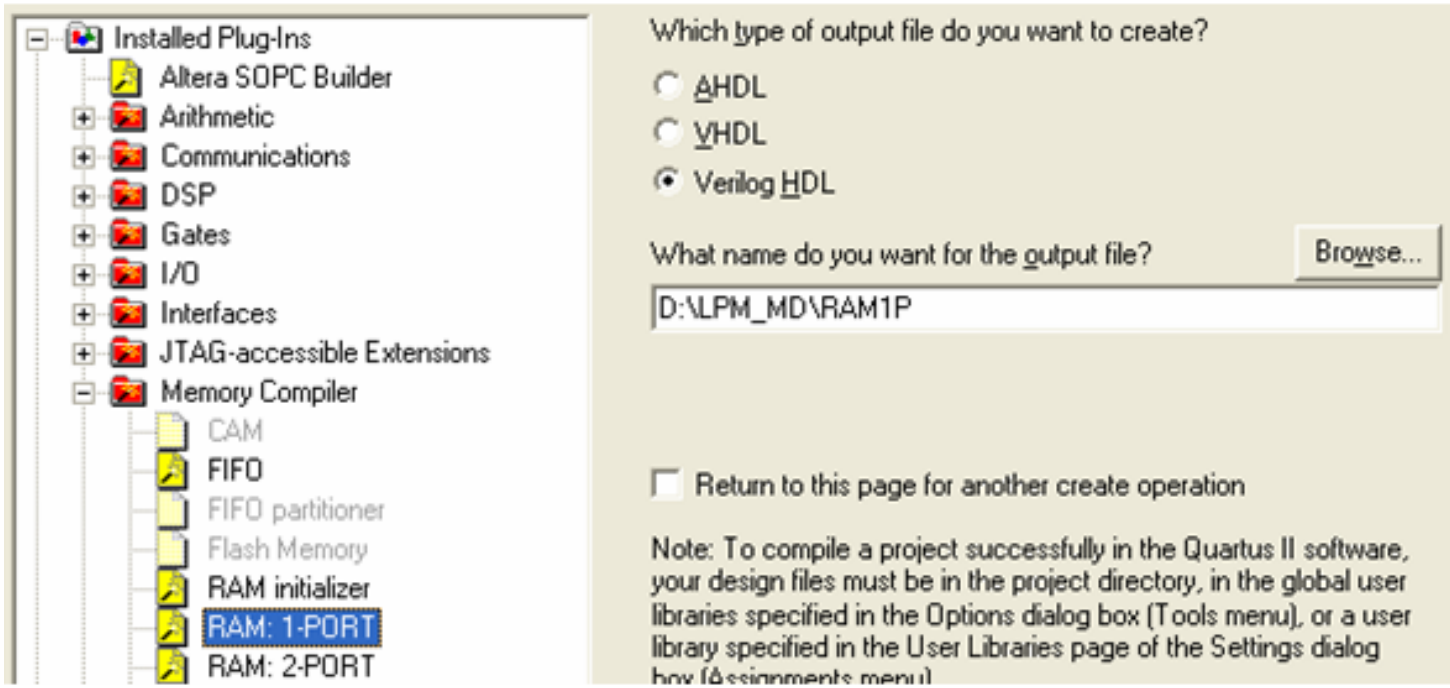

图 6-24　调用单口 LPM RAM

6.4 LPM 随机存储器的设置和调用

6.4.2 LPM_RAM的设置和调用

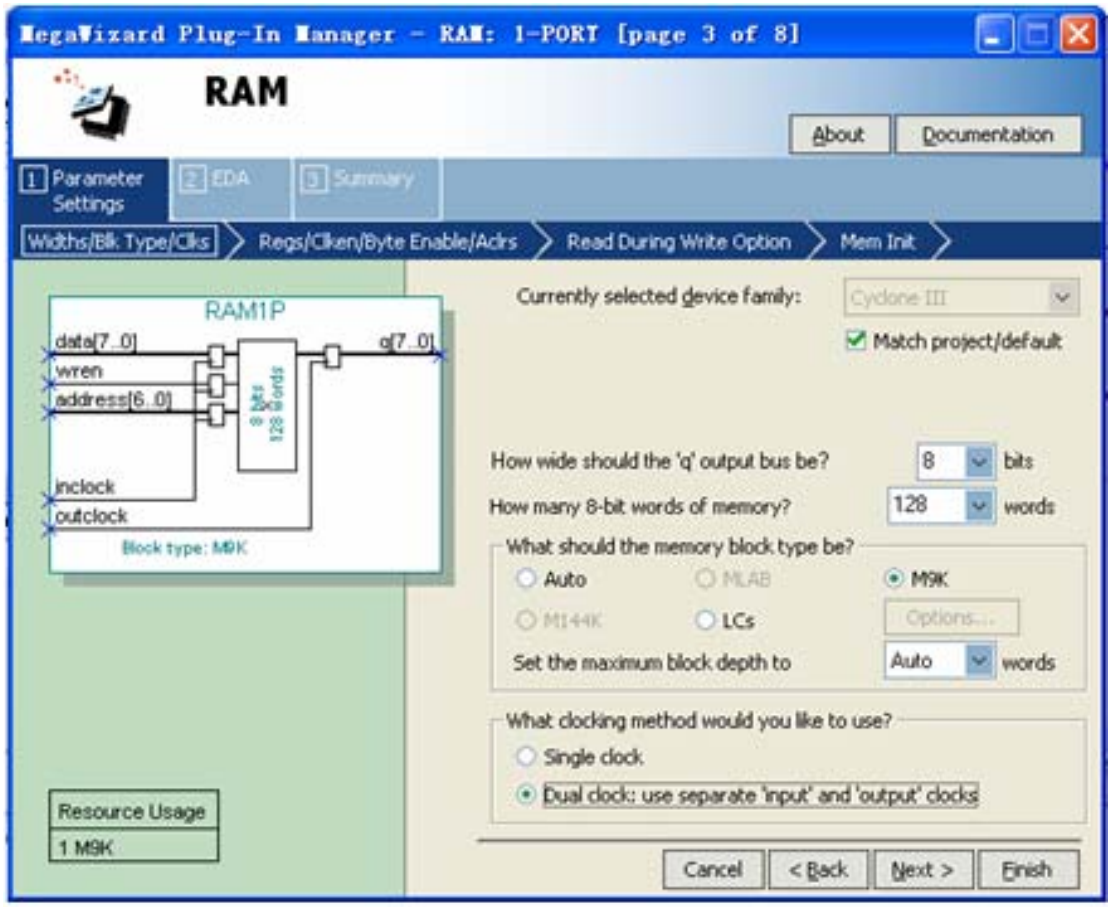

图 6-25　设定 RAM 参数

6.4 LPM 随机存储器的设置和调用

6.4.2 LPM_RAM的设置和调用

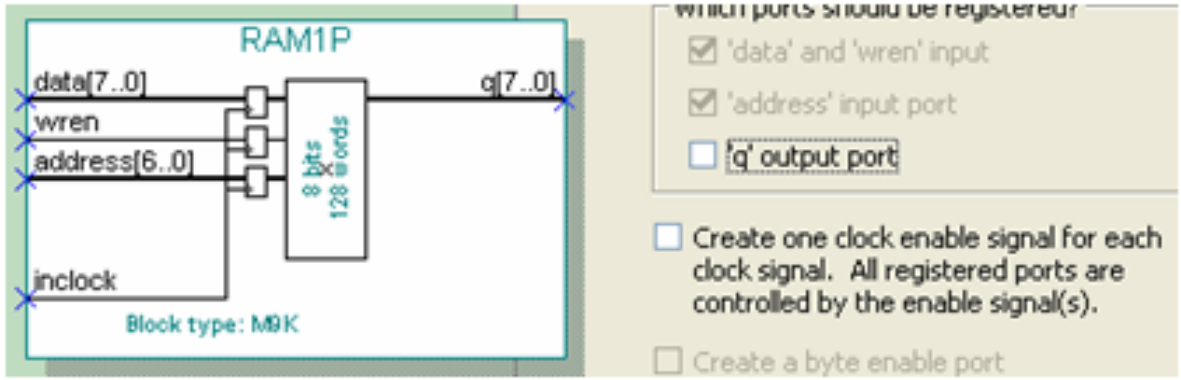

图 6-26　设定 RAM 仅输入时钟控制

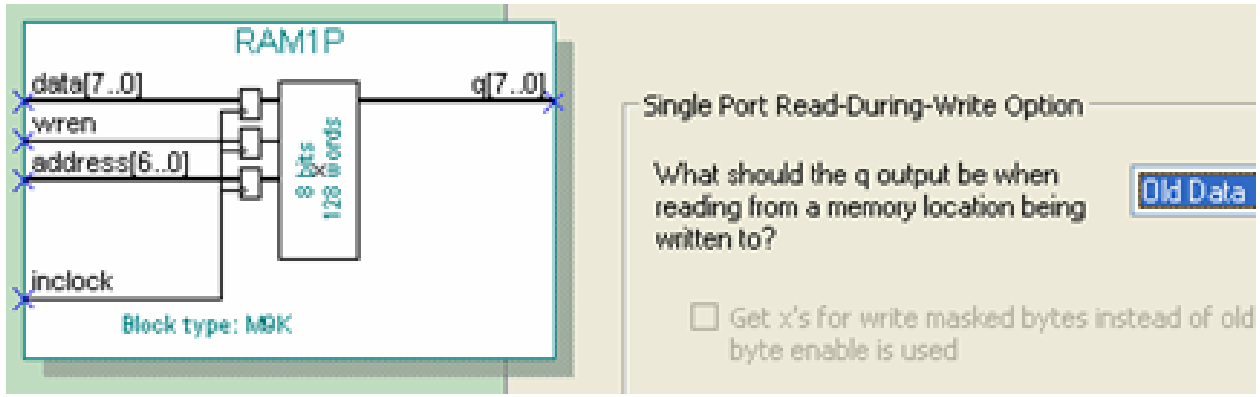

图 6-27 设定在写入同时读出原数据

6.4 LPM 随机存储器的设置和调用

6.4.2 LPM_RAM的设置和调用

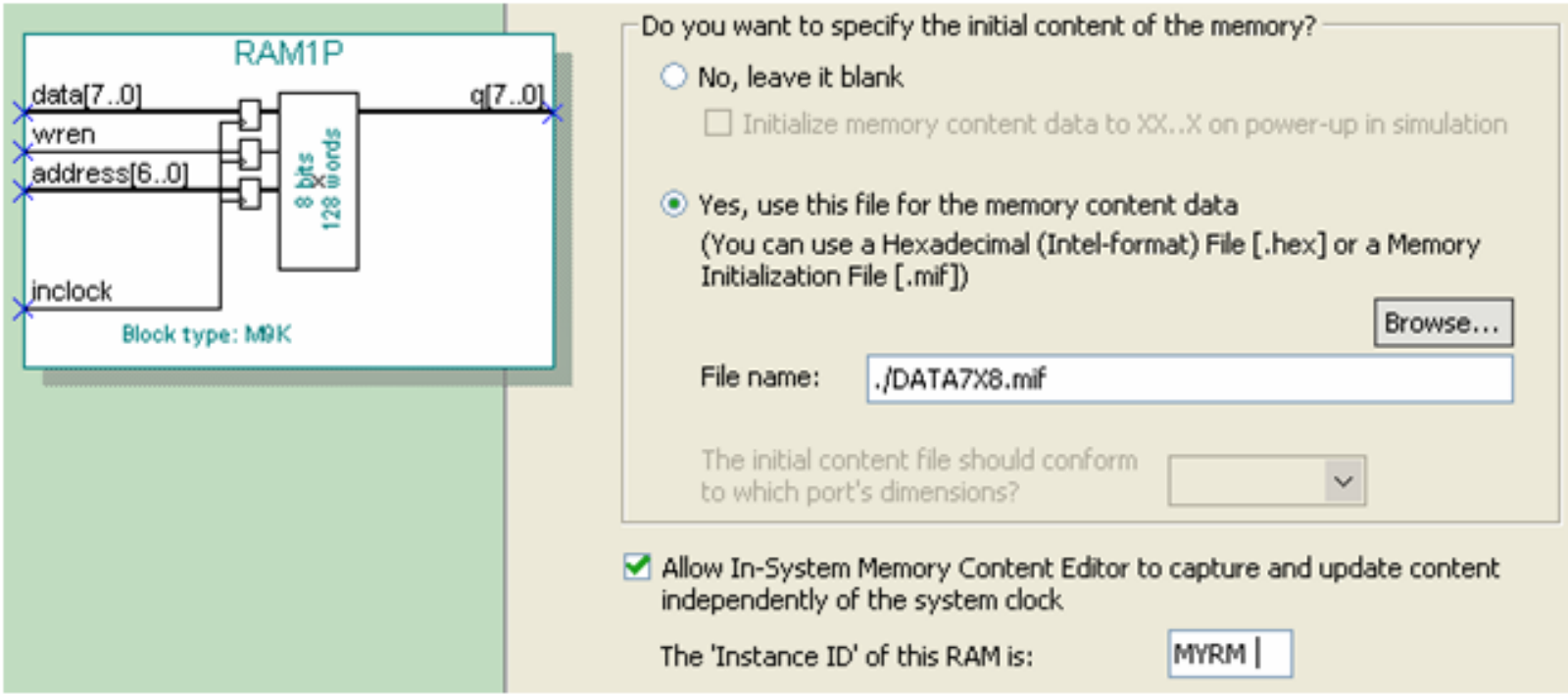

图 6-28 设定初始化文件和允许在系统编辑

6.4 LPM 随机存储器的设置和调用

6.4.2 LPM_RAM的设置和调用

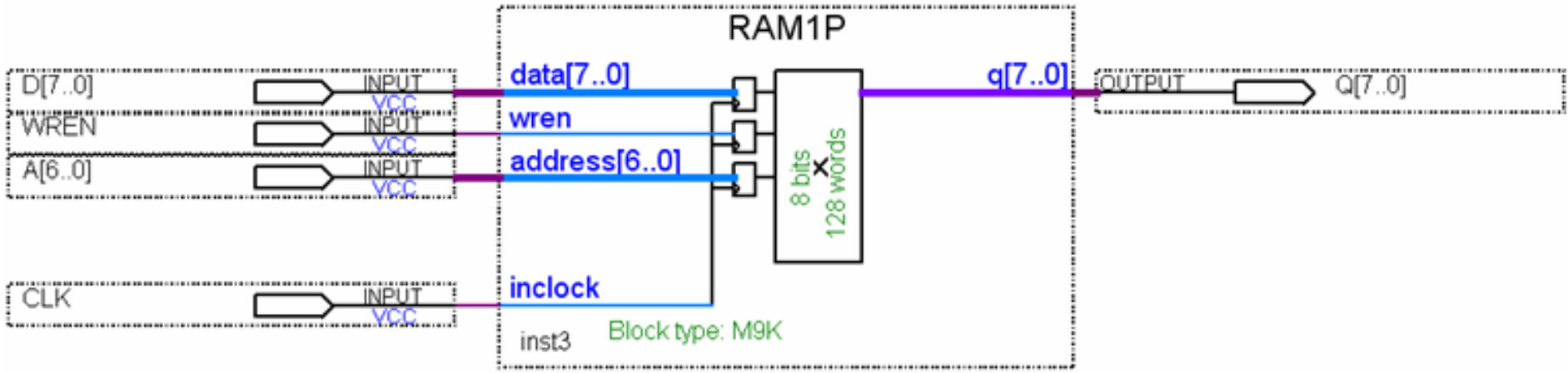

图 6-29 在原理图上连接好的 RAM 模块

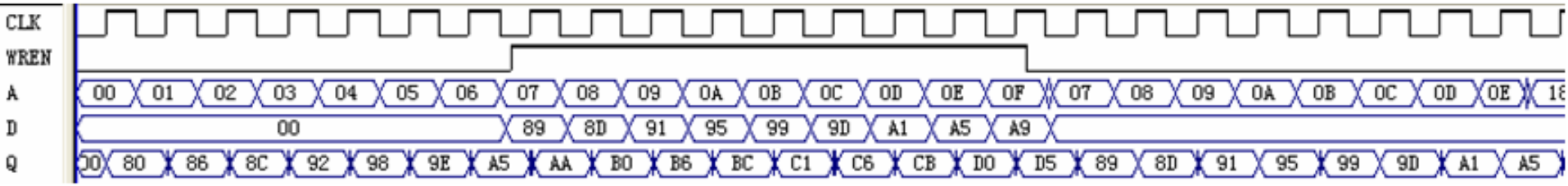

图 6-30 图 6-29 的 RAM 的仿真波形

6.4 LPM 随机存储器的设置和调用

6.4.4 VHDL的存储器描述及相关属性

【例 6-7】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ; --此程序包包含转换函数 CONV_INTEGER(A)
USE IEEE.STD_LOGIC_UNSIGNED.ALL ; --此程序包包含算符重载函数
ENTITY RAM78 IS
PORT (CLK,WREN : IN STD_LOGIC ; --定义时钟和写允许控制
        A : IN STD_LOGIC_VECTOR(6 DOWNTO 0) ; --定义 RAM 的 7 位地址输入端口
      DIN : IN STD_LOGIC_VECTOR(7 DOWNTO 0) ; --定义 RAM 的 8 位数据输入端口
       Q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)); --定义 RAM 的 8 位数据输出端口
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
```

接下页

6.4 LPM 随机存储器的设置和调用

6.4.4 VHDL的存储器描述及相关属性

接上页

```
BEGIN
  PROCESS (CLK)
  BEGIN
      IF  RISING_EDGE(CLK)  THEN
        IF  WREN='1' THEN --如果时钟有上升沿出现，且写使能为高电平，则
          MEM(CONV_INTEGER(A))<= DIN;--RAM 数据口的数据被写入指定地址的单元
        END IF;      END IF;
IF (FALLING_EDGE(CLK)) THEN  Q<=MEM(CONV_INTEGER(A));--读出存储器中的数据
     END IF;
  END PROCESS ;
 END BHV;
```

6.4 LPM 随机存储器的设置和调用

6.4.5 数据类型定义语句

1. 限定性数组型数据类型定义

```
TYPE 数组名 IS ARRAY(数组范围)OF 基本数据类型 ;

TYPE stb IS ARRAY (7 DOWNTO 0)of STD_LOGIC ;

TYPE MATRIX IS ARRAY (127 DOWNTO 0)of STD_LOGIC_VECTOR(7 DOWNTO 0) ;


  TYPE G_ARRAY IS ARRAY(0 TO 127) OF STD_LOGIC_VECTOR(7 DOWNTO 0) ;
  SIGNAL MEM : G_ARRAY;
```

6.4 LPM 随机存储器的设置和调用

6.4.5 数据类型定义语句

2. 非限定性数组型数据类型定义

```
TYPE 数组名 IS ARRAY (数组下标名 RANGE <>)OF 数据类型 ;
```

```
Type bit is ('0','1');
Type bit_vector is array(natural rang<>) of bit;
```

6.4 LPM 随机存储器的设置和调用

6.4.5 数据类型定义语句

3. 枚举型数据类型定义　TYPE BOOLEAN IS (FALSE,TRUE) ;

```
TYPE my_logic IS ( '1' ,'Z' ,'U' ,'0' ) ;
SIGNAL s1 : my_logic ;
s1 <= 'Z' ;
```

TYPE 数据类型名 IS 数据类型定义表述

```
TYPE week IS (sun,mon,tue,wed,thu,fri,sat) ;
TYPE x is (low, high);
TYPE data_bus IS ARRAY (0 TO 7, x)of BIT ;
TYPE m_state IS ( st0,st1,st2,st3,st4,st5 ) ;
SIGNAL present_state,next_state : m_state ;
```

6.4 LPM 随机存储器的设置和调用

6.4.5 数据类型定义语句

4. 枚举型子类型数据类型定义

```
SUBTYPE  子类型名 IS 基本数据类型 RANGE 约束范围;
```

```
SUBTYPE digits IS INTEGER RANGE 0 to 9 ;
```

6.4 LPM 随机存储器的设置和调用

6.4.6 存储器配置文件属性定义和结构设置

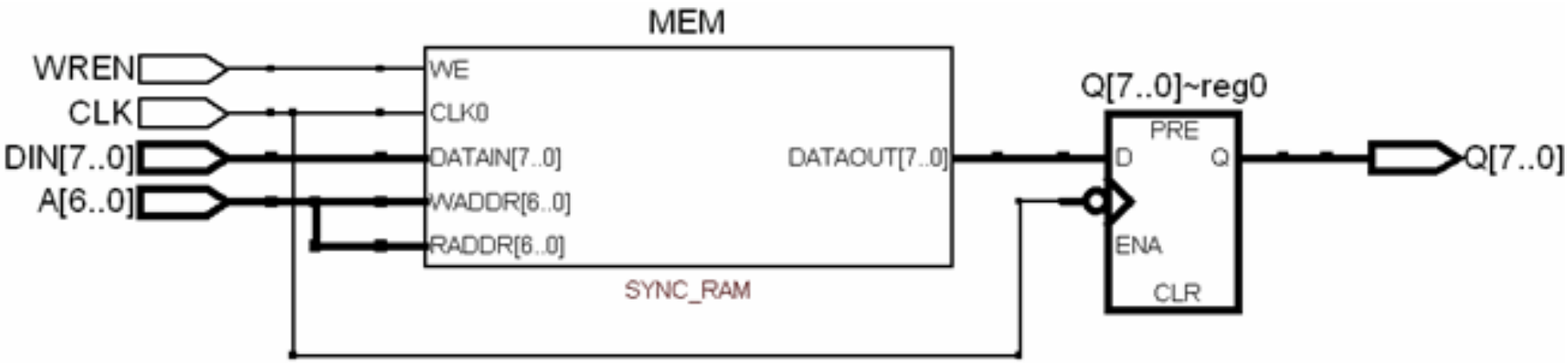

图 6-31 例 6-7 的 RAM78 的 RTL 图

6.4 LPM 随机存储器的设置和调用

6.4.6 存储器配置文件属性定义和结构设置

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 6-32 例 6-7 的编译报告

6.4 LPM 随机存储器的设置和调用

6.4.6 存储器配置文件属性定义和结构设置

【例 6-8】

```
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
```

6.4 LPM 随机存储器的设置和调用

6.4.6 存储器配置文件属性定义和结构设置

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 6-33 例 6-8 的编译报告

6.5 LPM_ROM的定制和使用示例

6.5.1 LPM_ROM的定制调用和测试

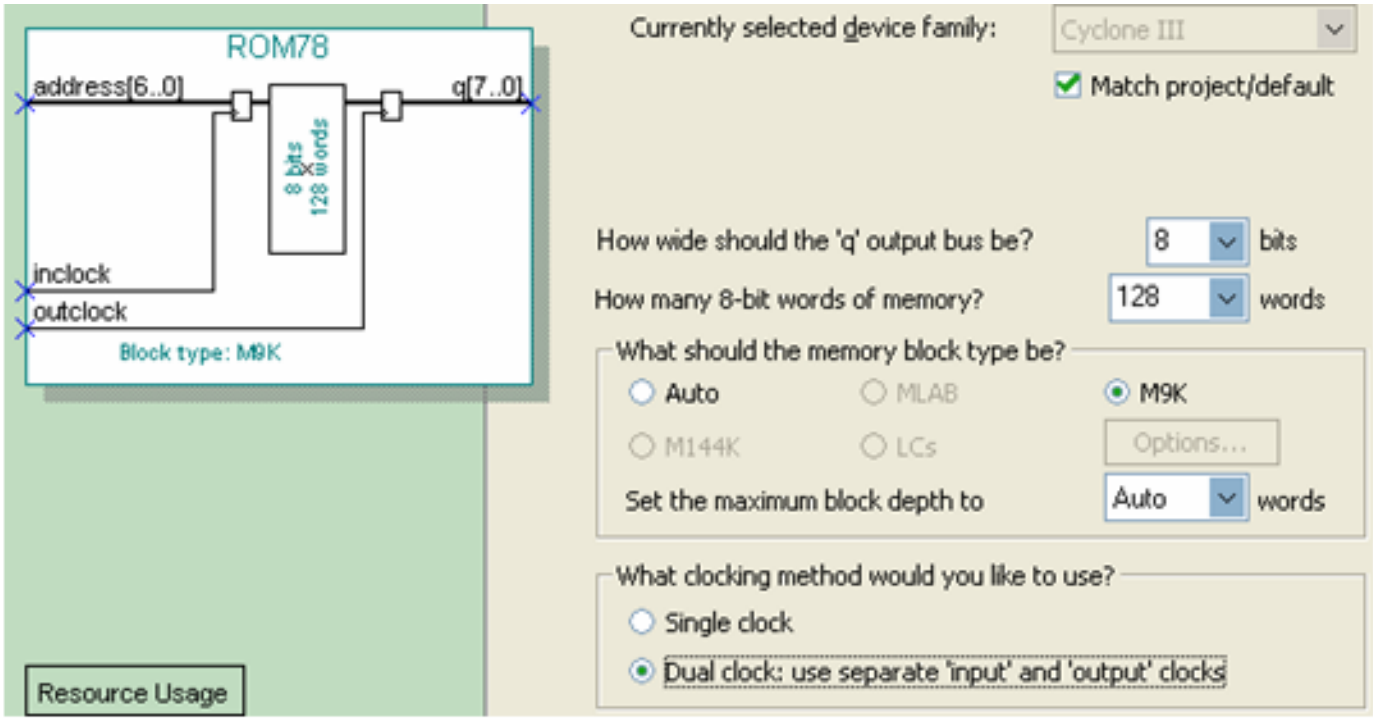

图 6-34 调用 LPM ROM 之参数设置

6.5 LPM_ROM的定制和使用示例

6.5.1 LPM_ROM的定制调用和测试

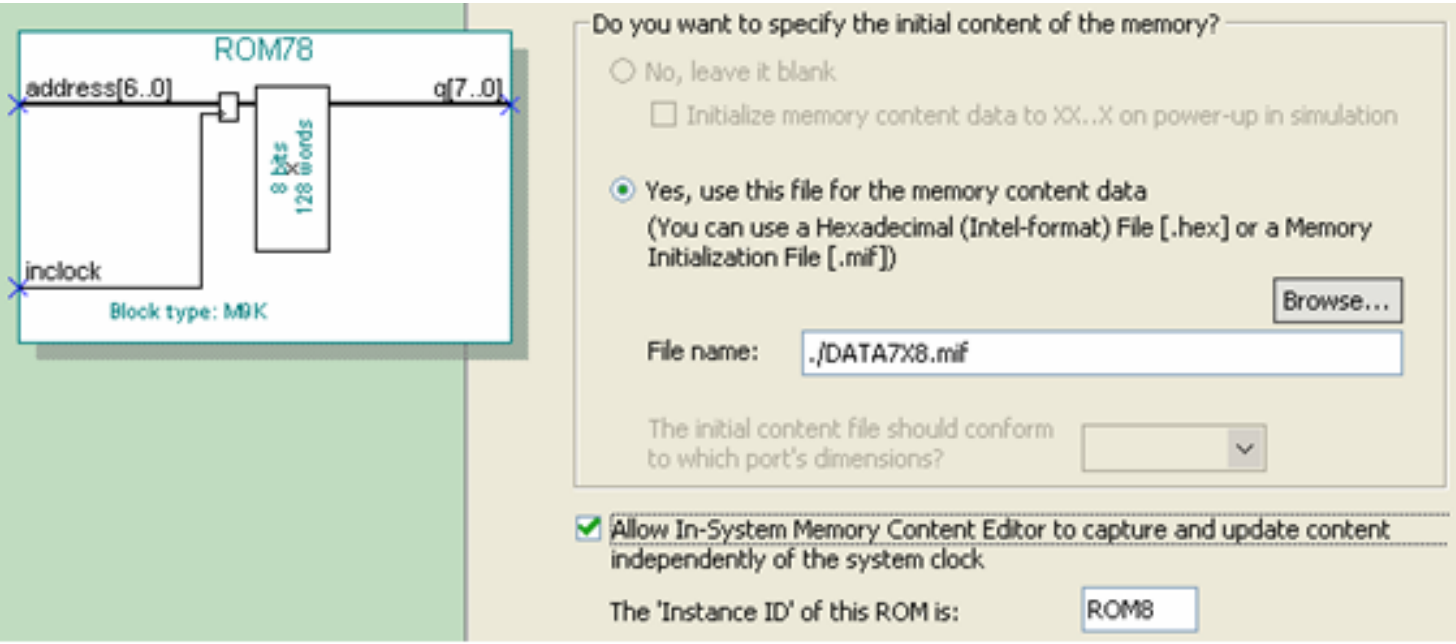

图 6-35 加入初始化配置文件

6.5 LPM_ROM的定制和使用示例

6.5.1 LPM_ROM的定制调用和测试

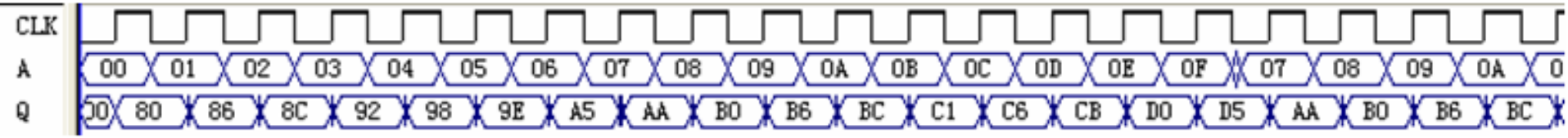

图 6-36 LPM ROM 仿真测试

6.5 LPM_ROM的定制和使用示例

6.5.2 LPM存储器模块取代设置

6.5.3 简易正弦信号发生器设计

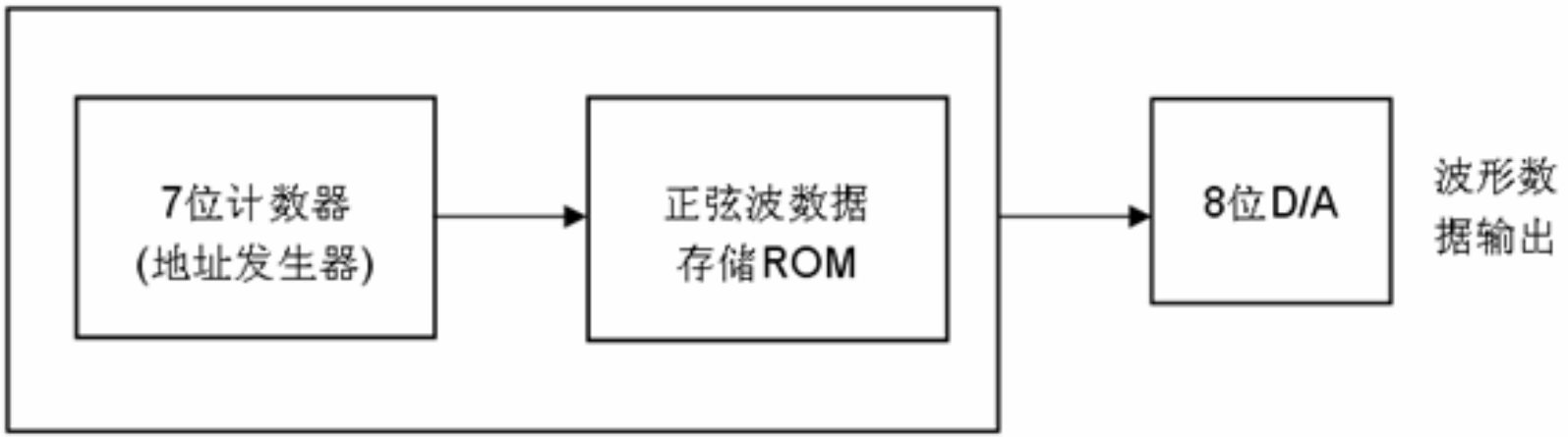

图 6-37　正弦信号发生器结构框图

6.5 LPM_ROM的定制和使用示例

6.5.3 简易正弦信号发生器设计

【例 6-9】

```
LIBRARY IEEE;  --正弦信号发生器源文件
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SIN_GNT IS
    PORT ( RST,CLK,EN  : IN STD_LOGIC;--复位信号，时钟，计数使能
           AR : OUT STD_LOGIC_VECTOR (6 DOWNTO 0);--7位地址测试输出
           Q  : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));--8位波形数据输出
END;
ARCHITECTURE ONE OF SIN_GNT IS
COMPONENT ROM78    --调用波形数据存储器LPM_ROM文件：rom78.vhd声明
   PORT(address : IN STD_LOGIC_VECTOR (6 DOWNTO 0);--7位地址信号
        inclock  : IN STD_LOGIC ;--地址锁存时钟
          q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)  );
```

接下页

6.5 LPM_ROM的定制和使用示例

6.5.3 简易正弦信号发生器设计

接上页

```
END COMPONENT;
    SIGNAL Q1 :STD_LOGIC_VECTOR(6 DOWNTO 0);--设定内部节点作为地址计数器
BEGIN
PROCESS(CLK,RST,EN )    BEGIN        --LPM_ROM地址发生器进程
IF (RST='0') THEN   Q1 <="0000000";
 ELSIF CLK'EVENT AND CLK = '1' THEN
   IF (EN='1')  THEN Q1<=Q1+1;  END IF ; END IF;
END PROCESS;
AR<=Q1;
u1 : ROM78 PORT MAP(address=>Q1, q => Q,inclock=>CLK);--例化
END;
```

6.5 LPM_ROM的定制和使用示例

6.5.3 简易正弦信号发生器设计

【例 6-10】 ROM78.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.all;
ENTITY ROM78 IS
 PORT (address   : IN STD_LOGIC_VECTOR (6 DOWNTO 0);
     inclock     : IN STD_LOGIC ;
         q       : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
END ROM78;
 ...
```

6.5 LPM_ROM的定制和使用示例

6.5.3 简易正弦信号发生器设计

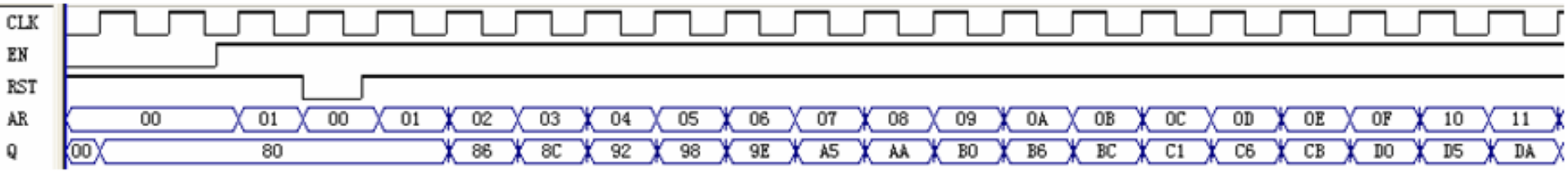

图 6-38　例 6-9 的仿真波形输出

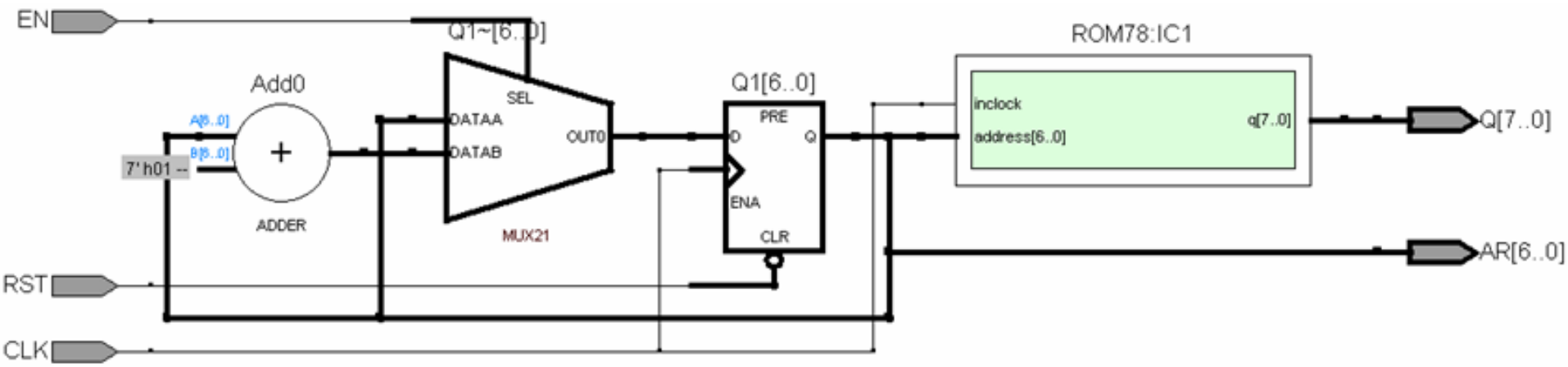

图 6-39　例 6-9 的 RTL 电路图

6.5 LPM_ROM的定制和使用示例

6.5.4 正弦信号发生器硬件实现和测试

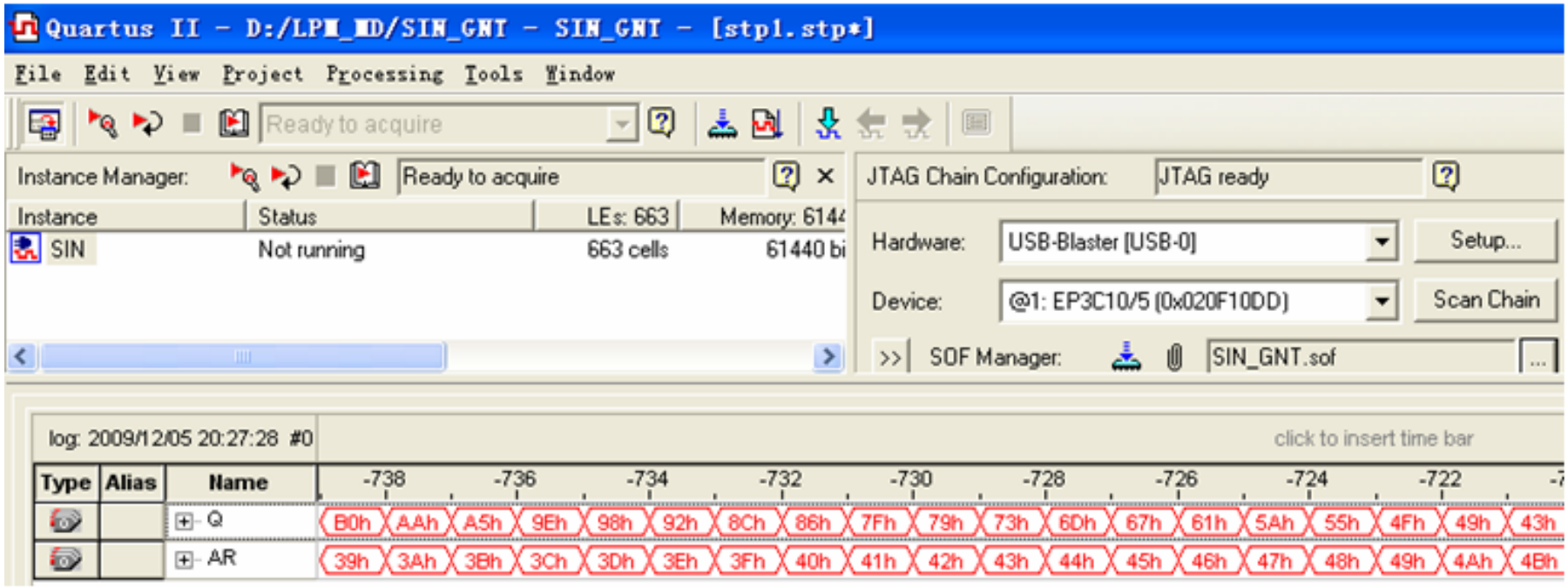

图 6-40 正弦信号发生器数据输出的 SignalTapII 测试图

6.5 LPM_ROM的定制和使用示例

6.5.4 正弦信号发生器硬件实现和测试

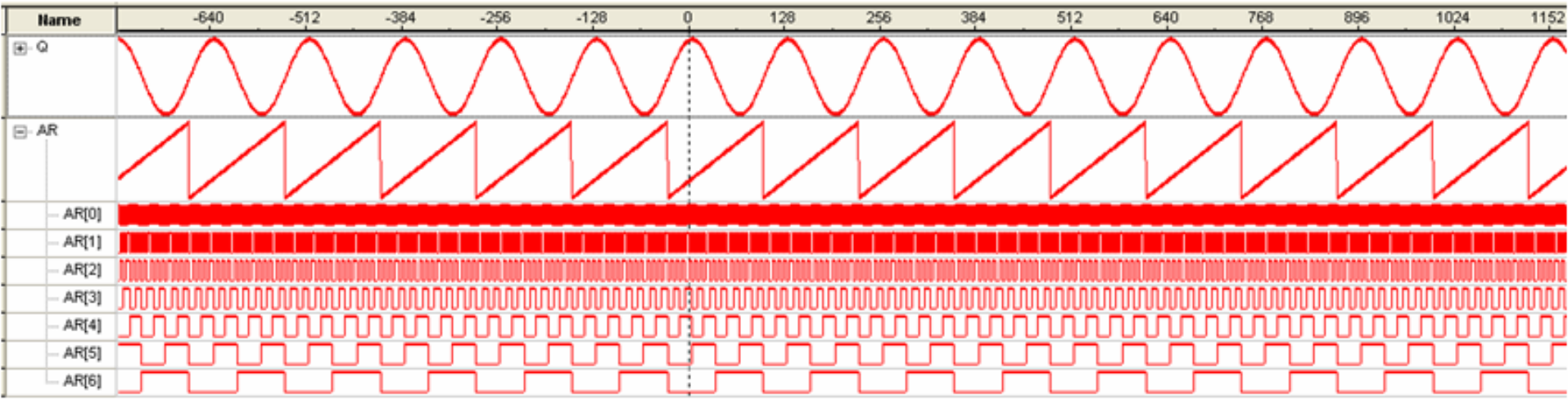

图 6-41 正弦信号发生器的 SignalTapII 的波形显示图

6.6 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

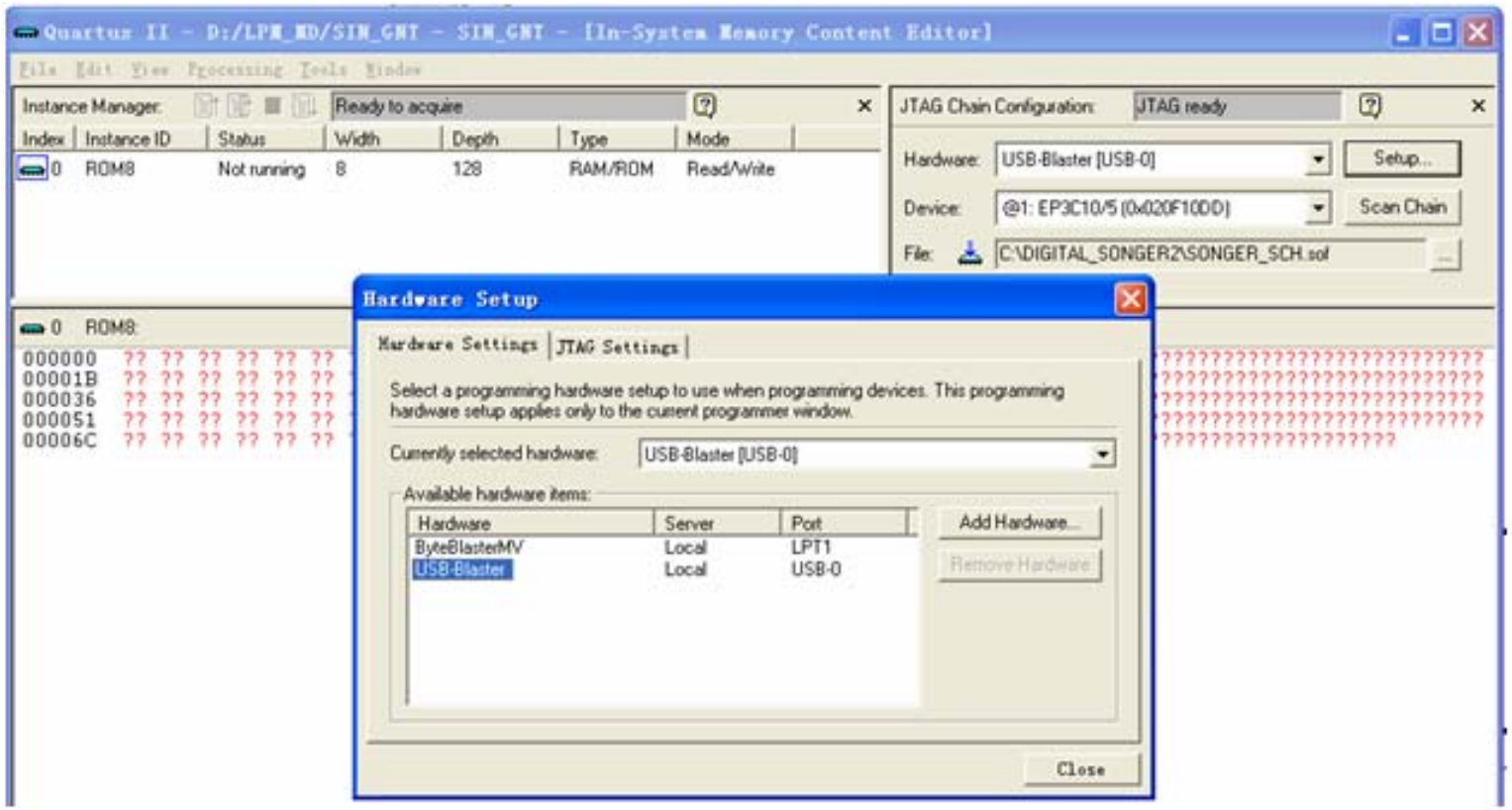

图 6-42 In-System Memory Content Editor 编辑窗

6.6 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

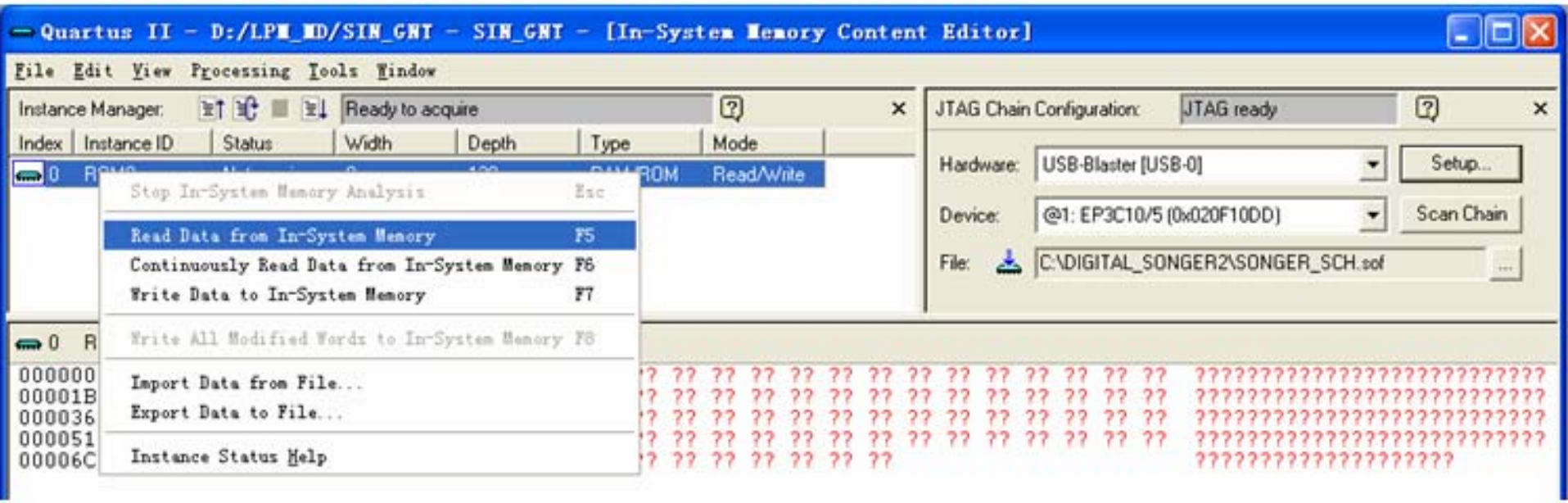

图 6-43　与实验系统上的 FPGA 通信正常情况下的编辑窗口界面

6.6 在系统存储器数据读写编辑器应用

（2）读取ROM中的波形数据

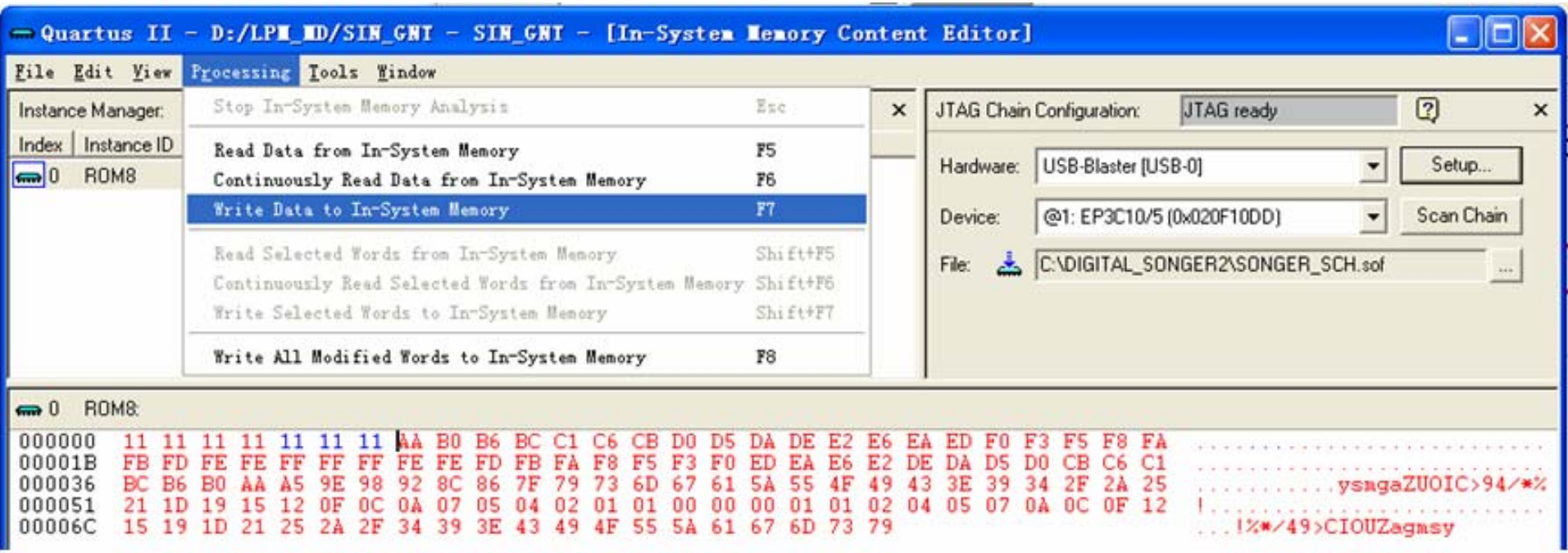

图 6-44　从 FPGA 中的 ROM 读取波形数据并编辑数据

6.6 在系统存储器数据读写编辑器应用

（3）写数据

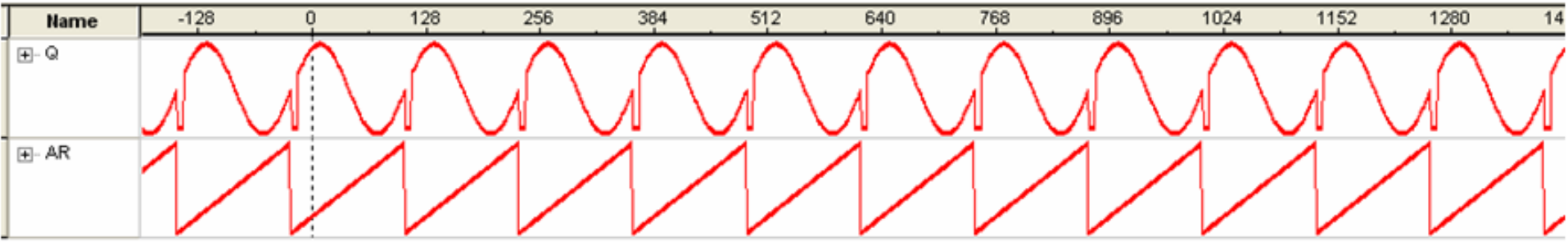

图 6-45　下载编辑数据后的 SignalTap Ⅱ 采样波形

（4）输入输出数据文件

6.7 FIFO定制

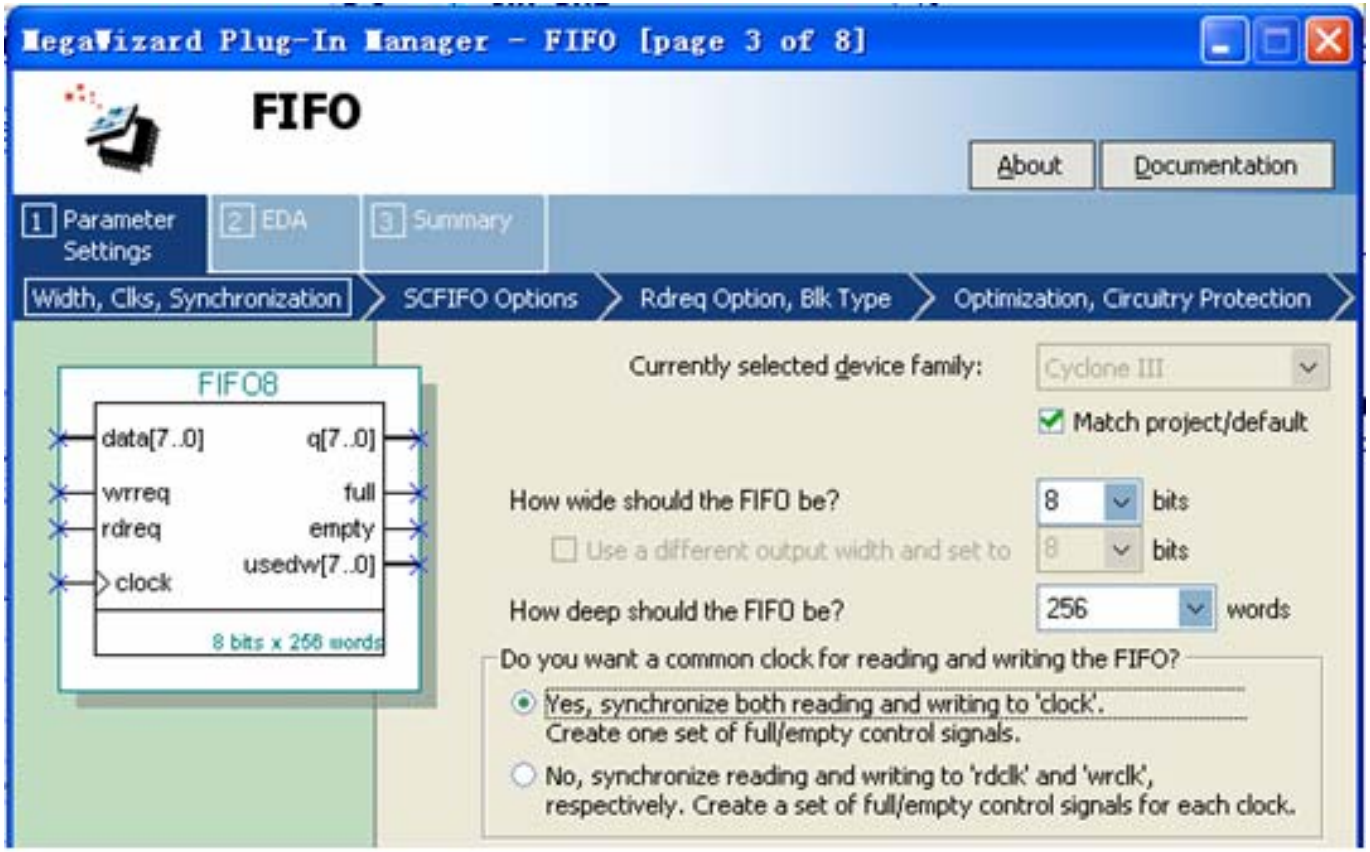

图 6-46 设置 FIFO 数据位宽和深度

6.7 FIFO定制

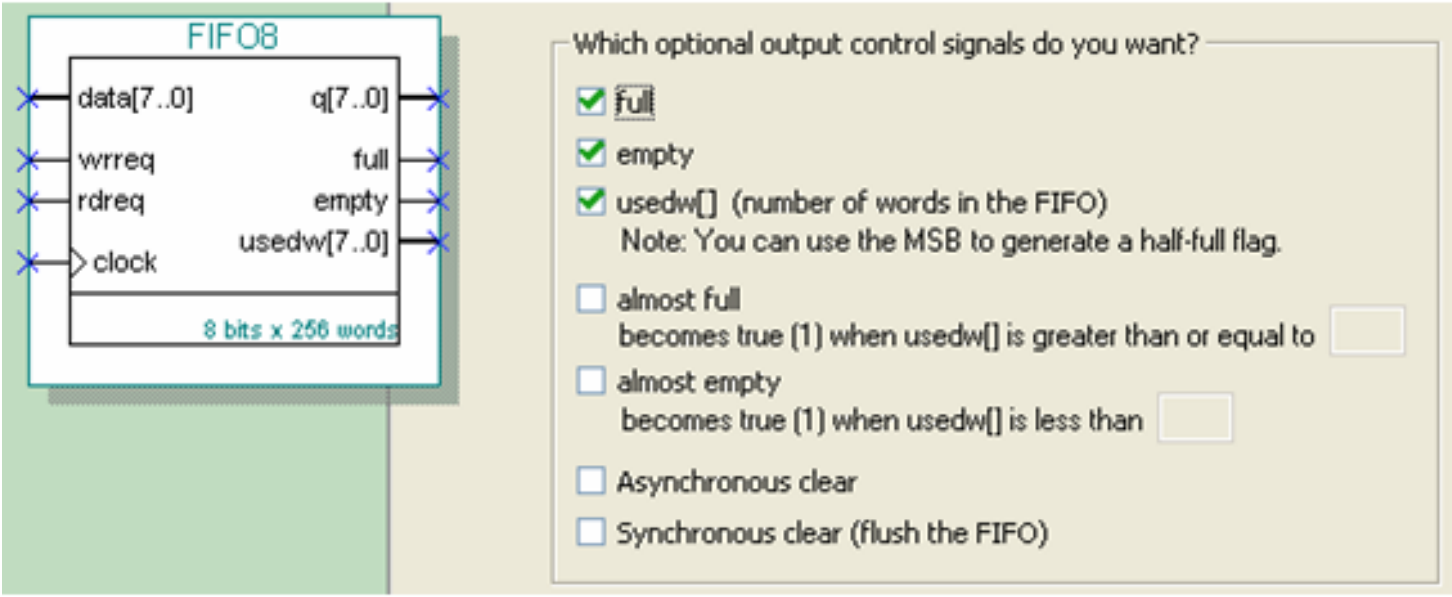

图 6-47 设置 FIFO 各种输出标志信号

6.7 FIFO定制

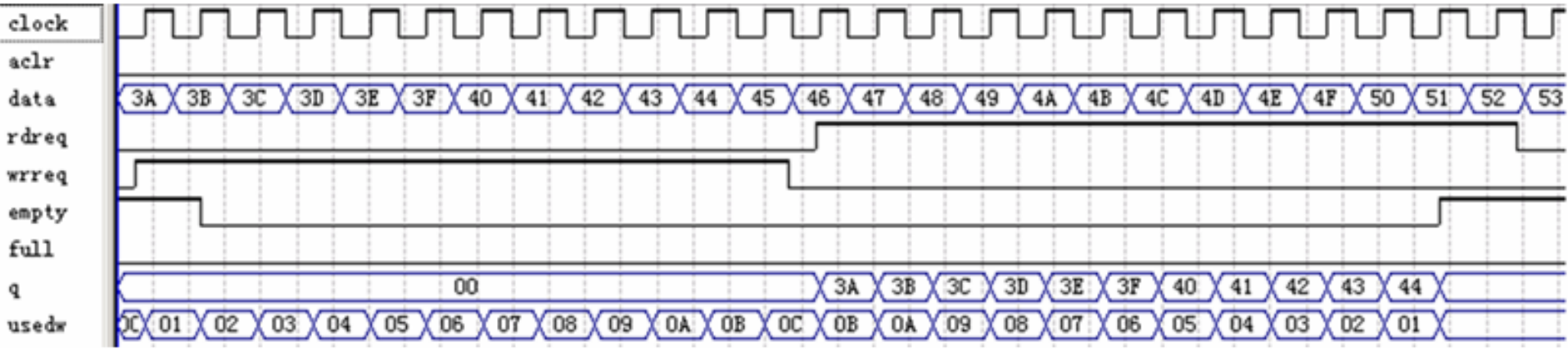

图 6-48　FIFO 的仿真波形

6.8 LPM嵌入式锁相环调用

6.8.1 建立嵌入式锁相环元件

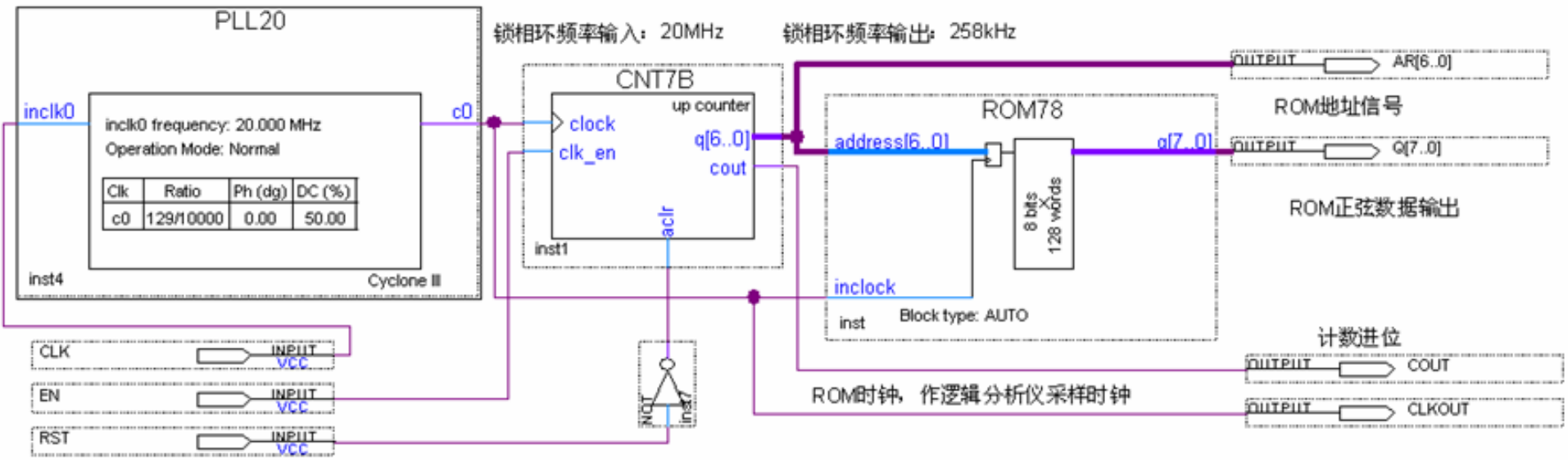

图 6-49　采用嵌入式锁相环作时钟的正弦信号发生器电路图

6.8 LPM嵌入式锁相环调用

6.8.1 建立嵌入式锁相环元件

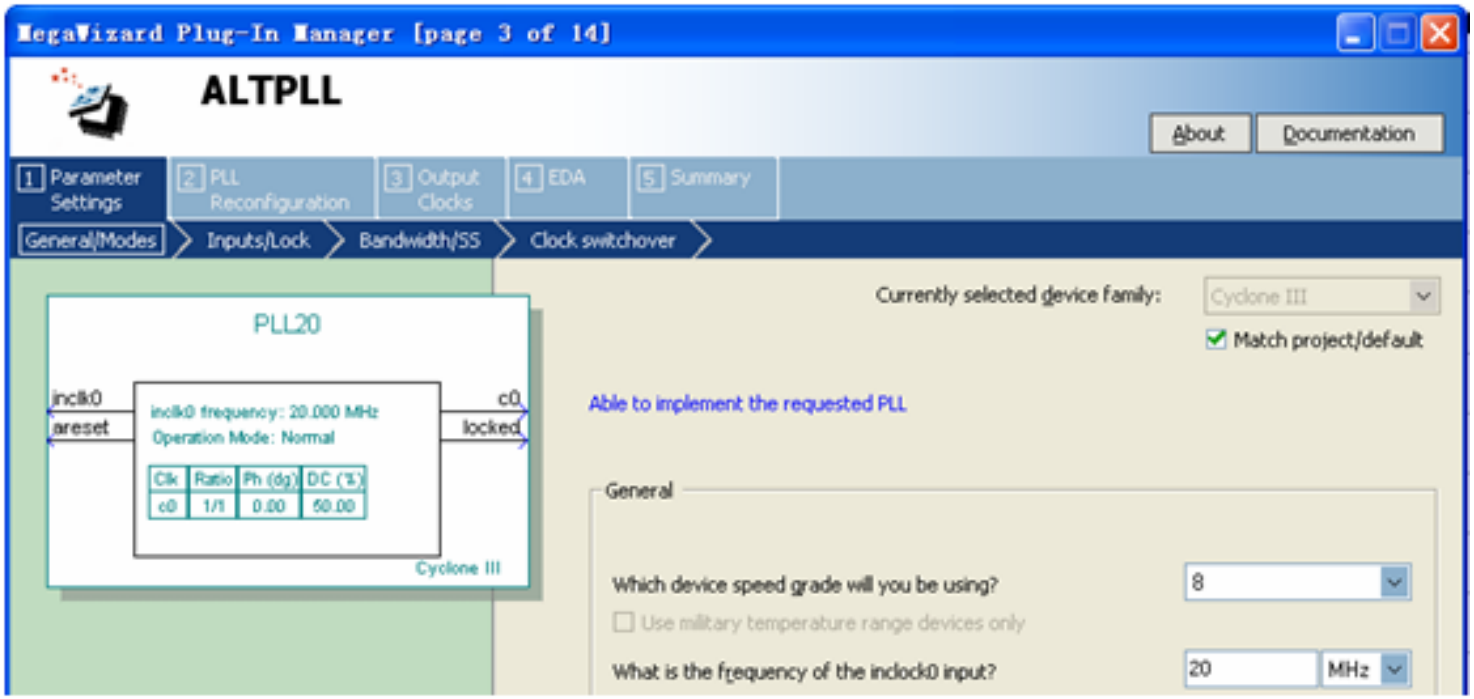

图 6-50 选择输入参考时钟 inclk0 为 20MHz

6.8 LPM嵌入式锁相环调用

6.8.1 建立嵌入式锁相环元件

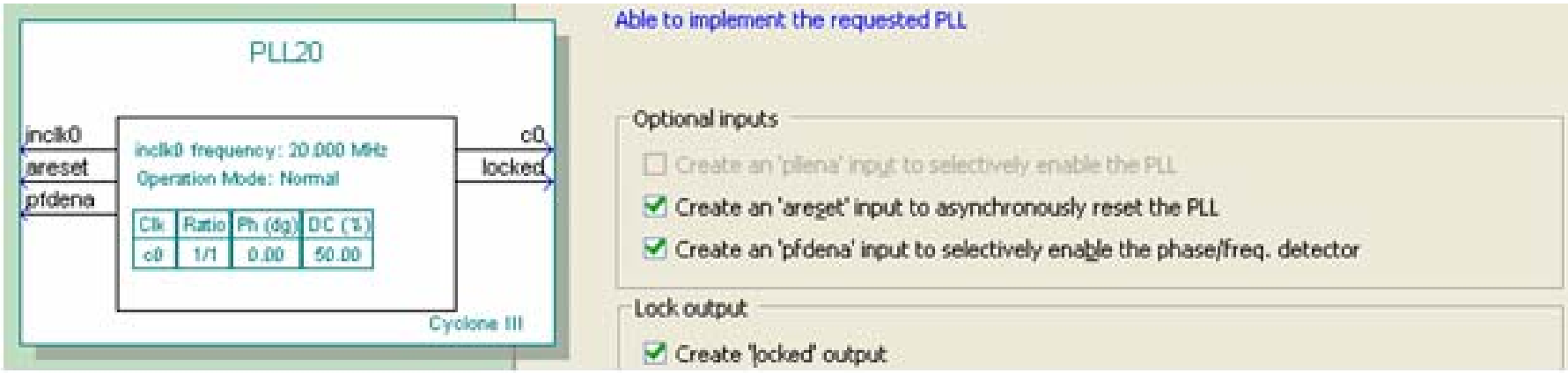

图 6-51　选择控制信号

6.8 LPM嵌入式锁相环调用

6.8.1 建立嵌入式锁相环元件

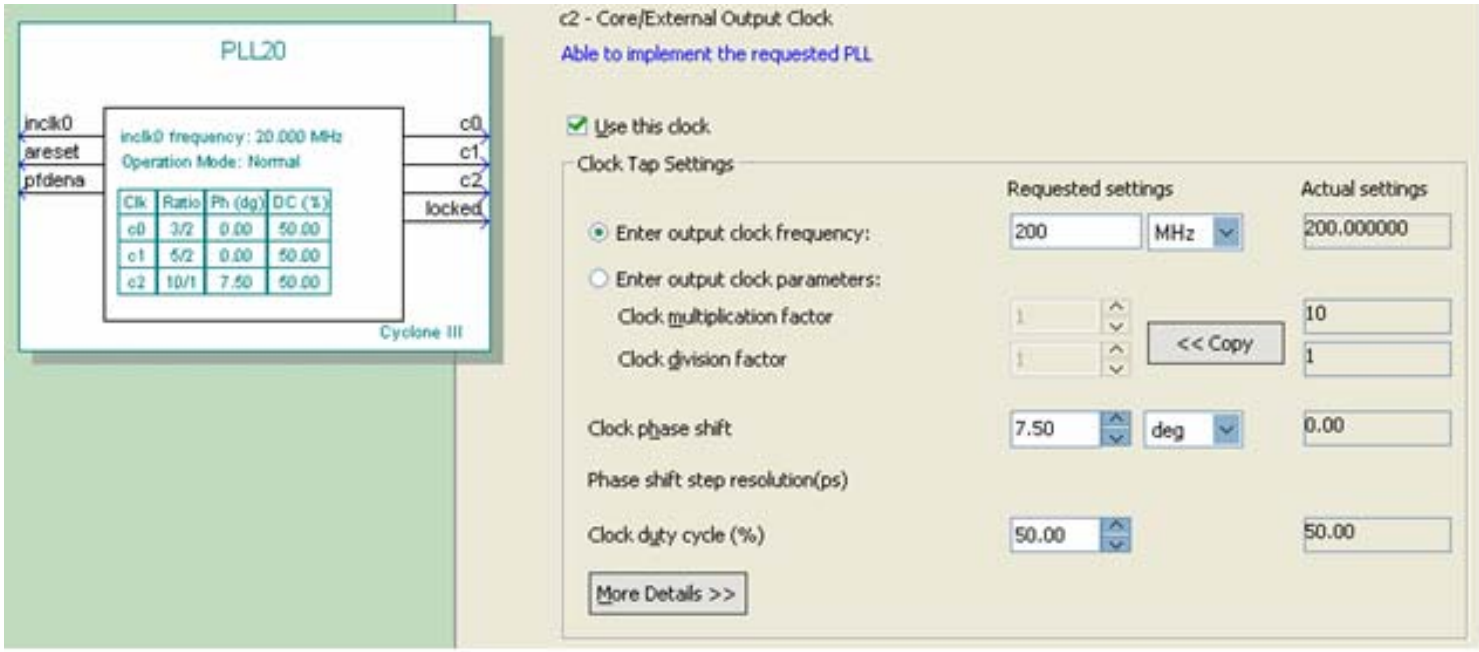

图 6-52 选择 e0 的输出频率为 200MHz

6.8 LPM嵌入式锁相环调用

6.8.2 联合设计与测试

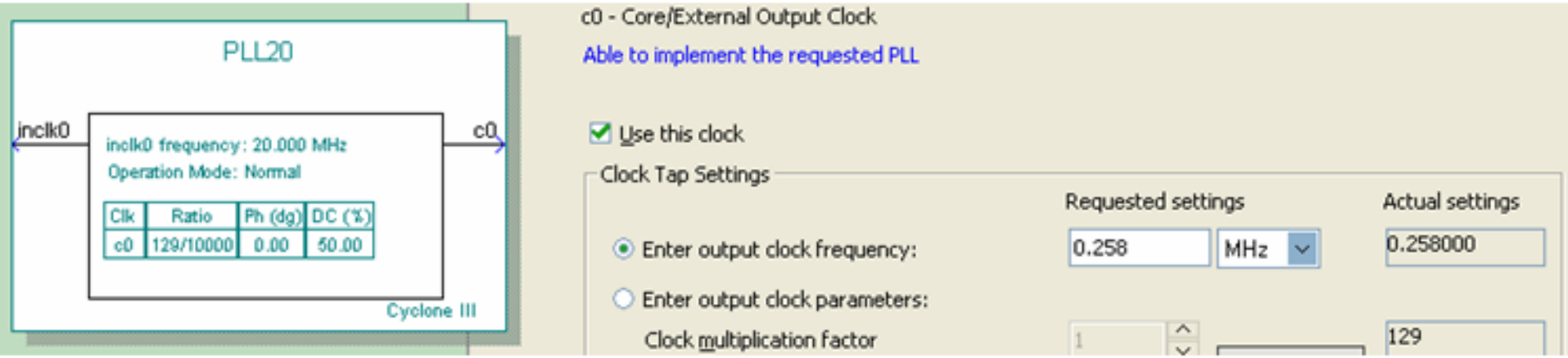

图 6-53 选择输出频率为 0.258MHz 作正弦信号发生器工作时钟

图 6-54 图 6-49 设计的逻辑分析仪实时采样输出

6.8 LPM嵌入式锁相环调用

6.8.3 测试锁相环

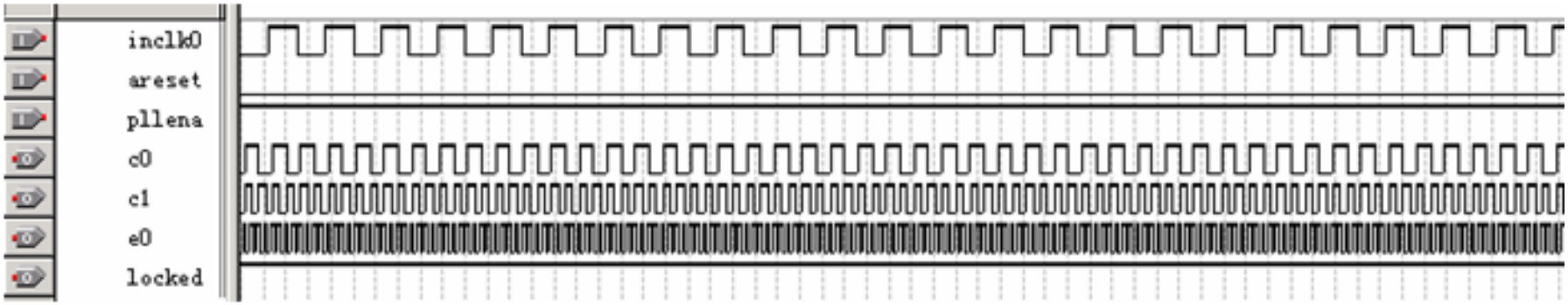

图 6-55　PLL 元件的仿真波形

6.9 NCO核数控振荡器使用方法

（1）定制NCO

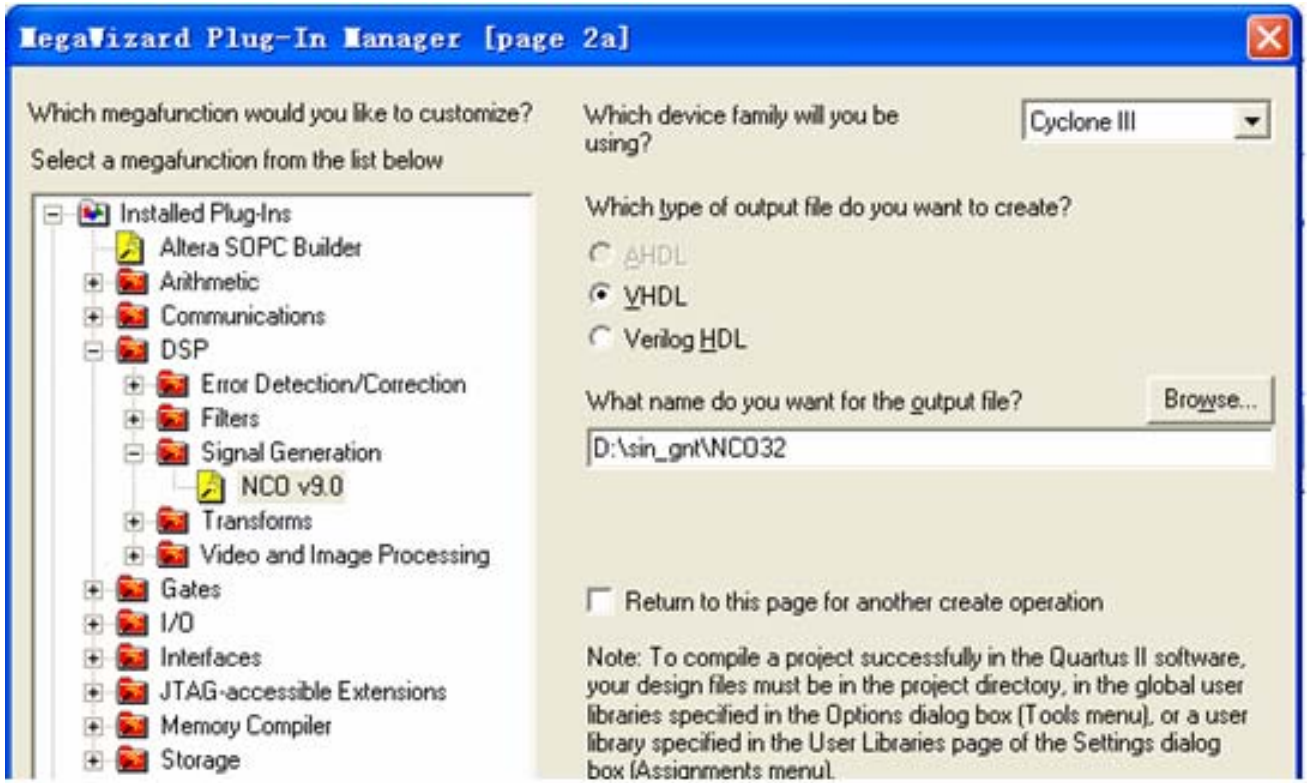

图 6-56 打开 Core 设置管理窗口选择 NCO 核

6.9 NCO核数控振荡器使用方法

（2）进入Core文件生成选择窗

图 6-57　开始进入 Core 文件生成选择窗口

6.9 NCO核数控振荡器使用方法

（3）设置参数

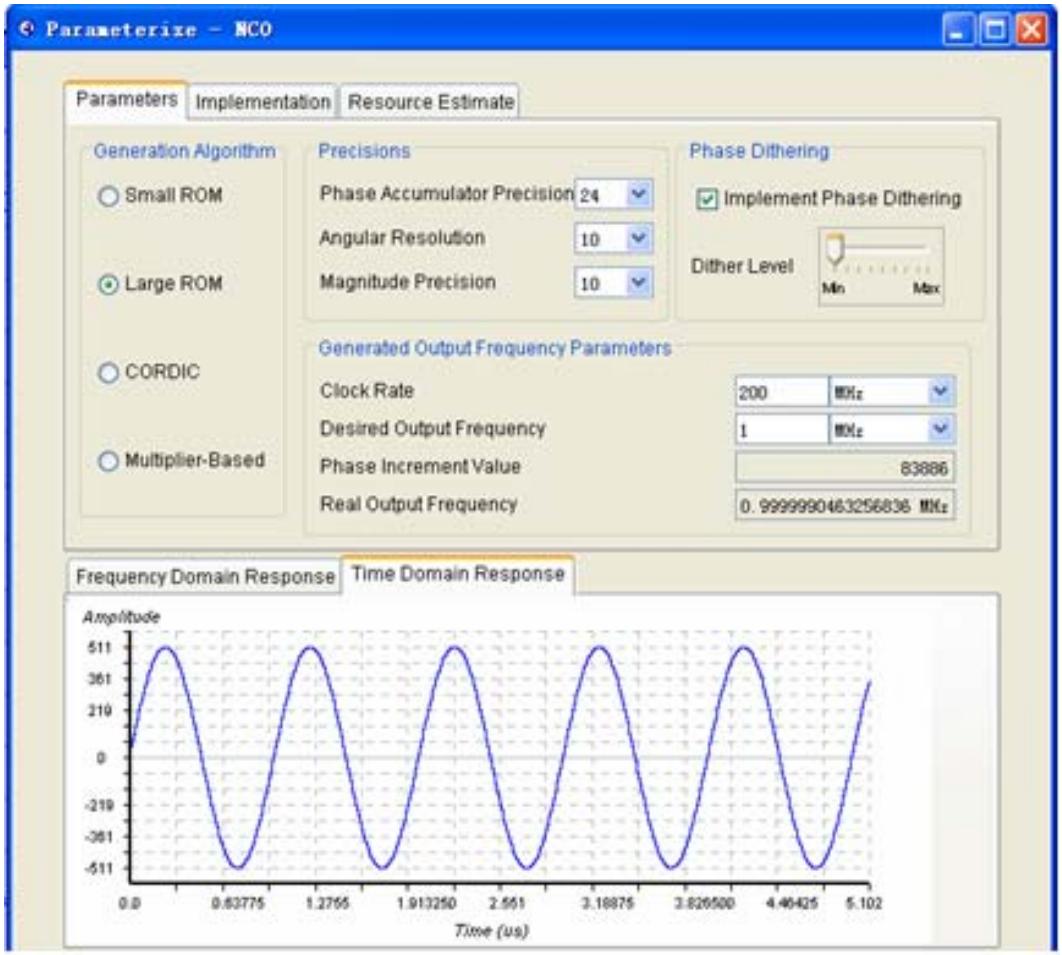

图 6-58　设置 NCO 参数（1）

6.9 NCO核数控振荡器使用方法

（3）设置参数

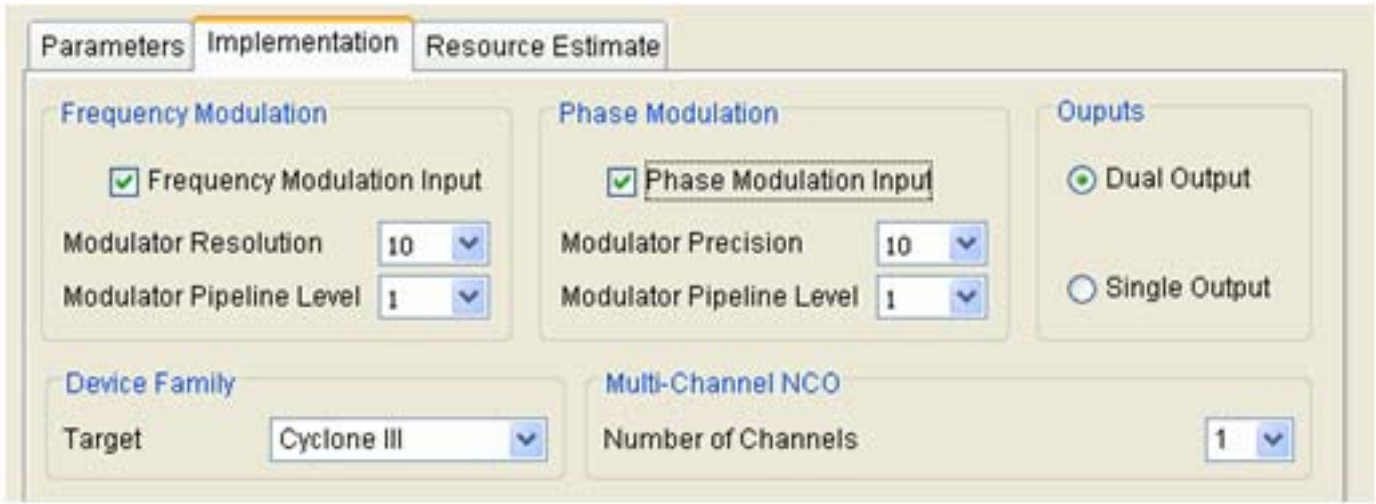

图 6-59 设置 NCO 参数（2）

6.9 NCO核数控振荡器使用方法

（4）生成仿真文件

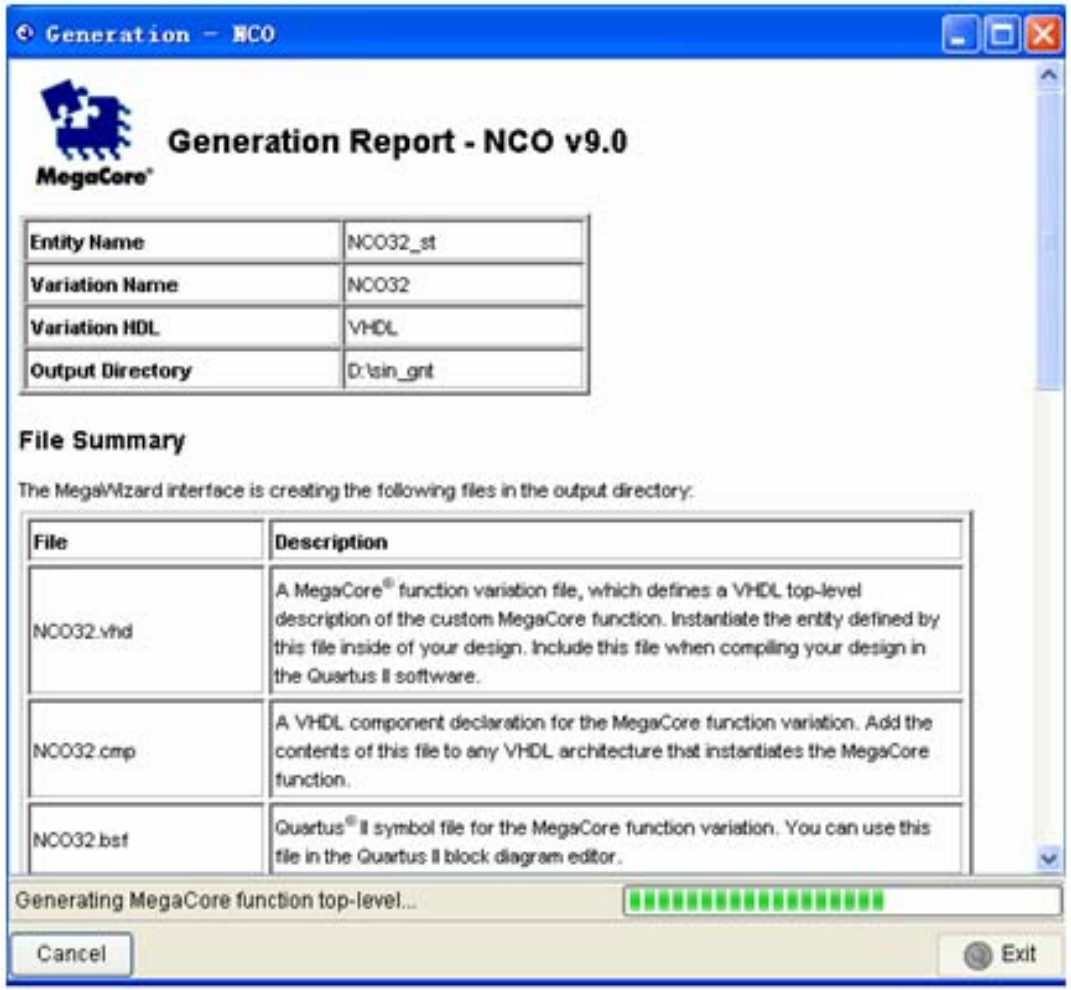

图 6-60 完成NCO参数设置并生成设计文件后的信息窗口

6.9 NCO核数控振荡器使用方法

（5）加入IP授权文件

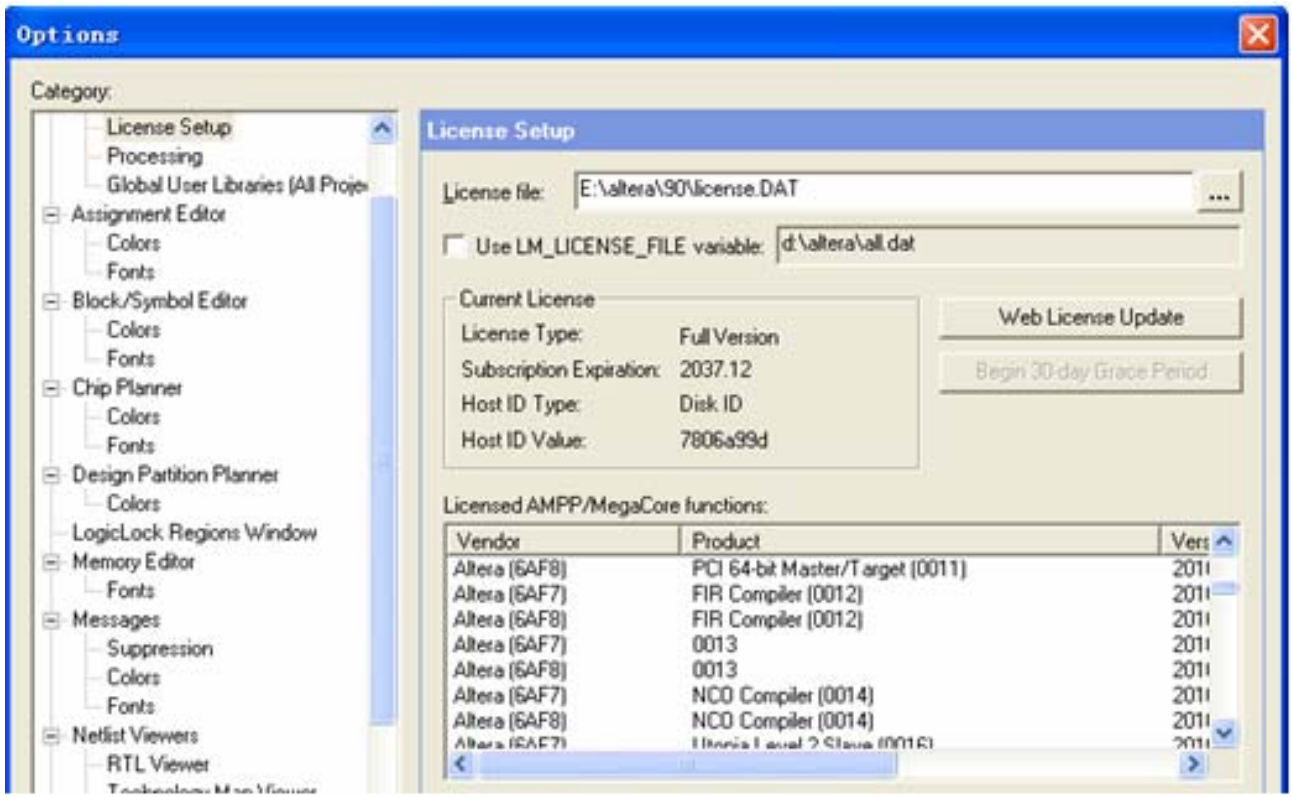

图 6-61　加入含有 NCO 等 IP 的授权文件

6.9 NCO核数控振荡器使用方法

（6）选择目标器件，然后对生成的模块进行编译及功能检测

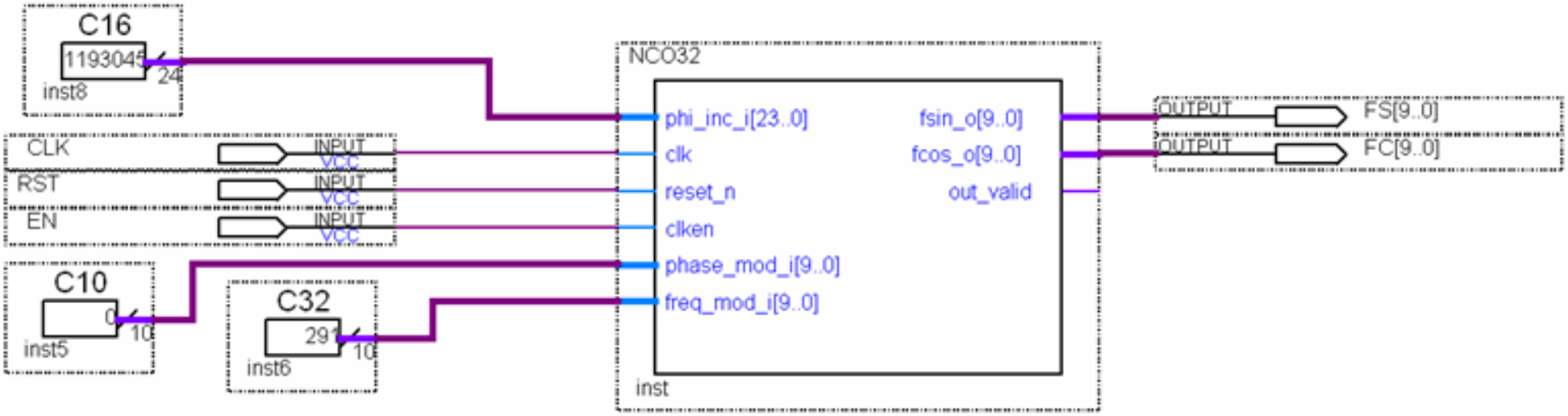

图 6-62 测试 NCO 的电路

6.9 NCO核数控振荡器使用方法

（6）选择目标器件，然后对生成的模块进行编译及功能检测

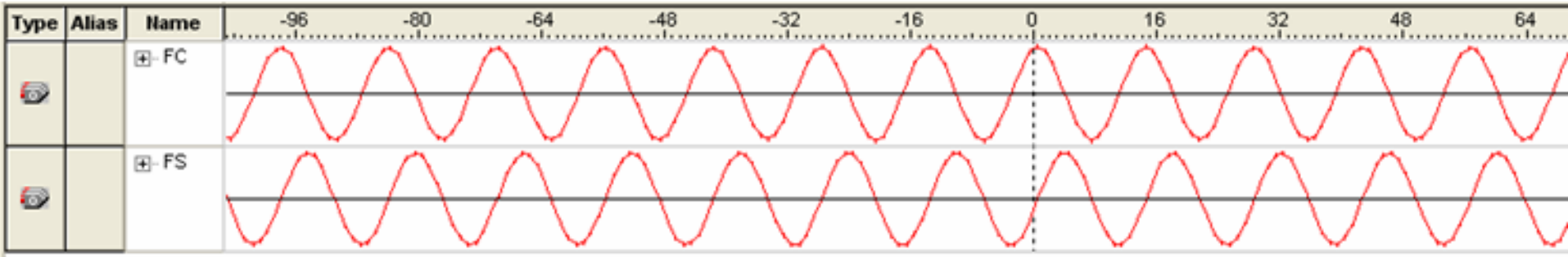

图 6-63 当前 NCO 的逻辑分析仪测试波形

6.10 使用IP Core设计FIR滤波器

$$H(z)=\sum_{n=0}^{N-1} h(n)z^{-n} \qquad (6\text{-}1)$$

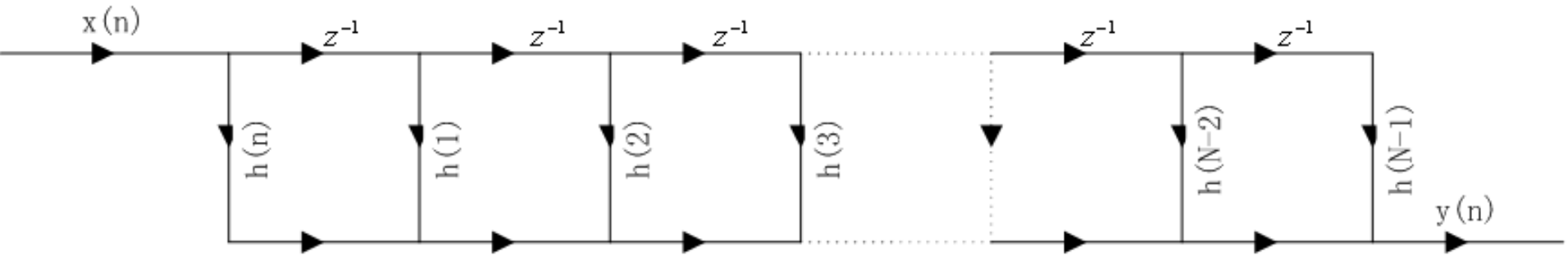

图 6-64 直接型 FIR 滤波器结构

6.10 使用IP Core设计FIR滤波器

$$Y(n)=\sum_{m=0}^{N-1}h(m)x(n-m) \qquad (6\text{-}2)$$

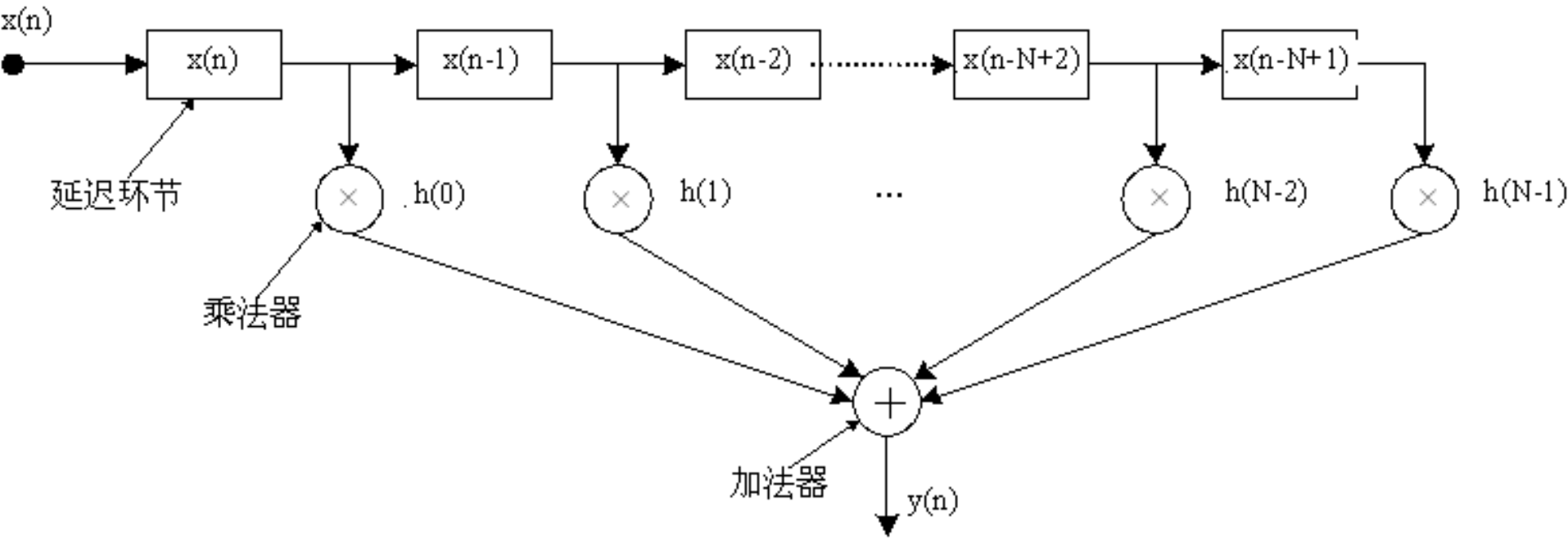

图 6-65 直接型 FIR 实现结构

6.10 使用IP Core设计FIR滤波器

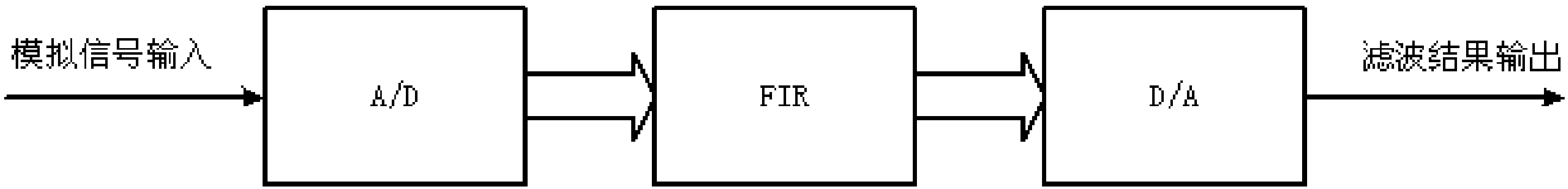

图 6-66 FIR 滤波器设计示意

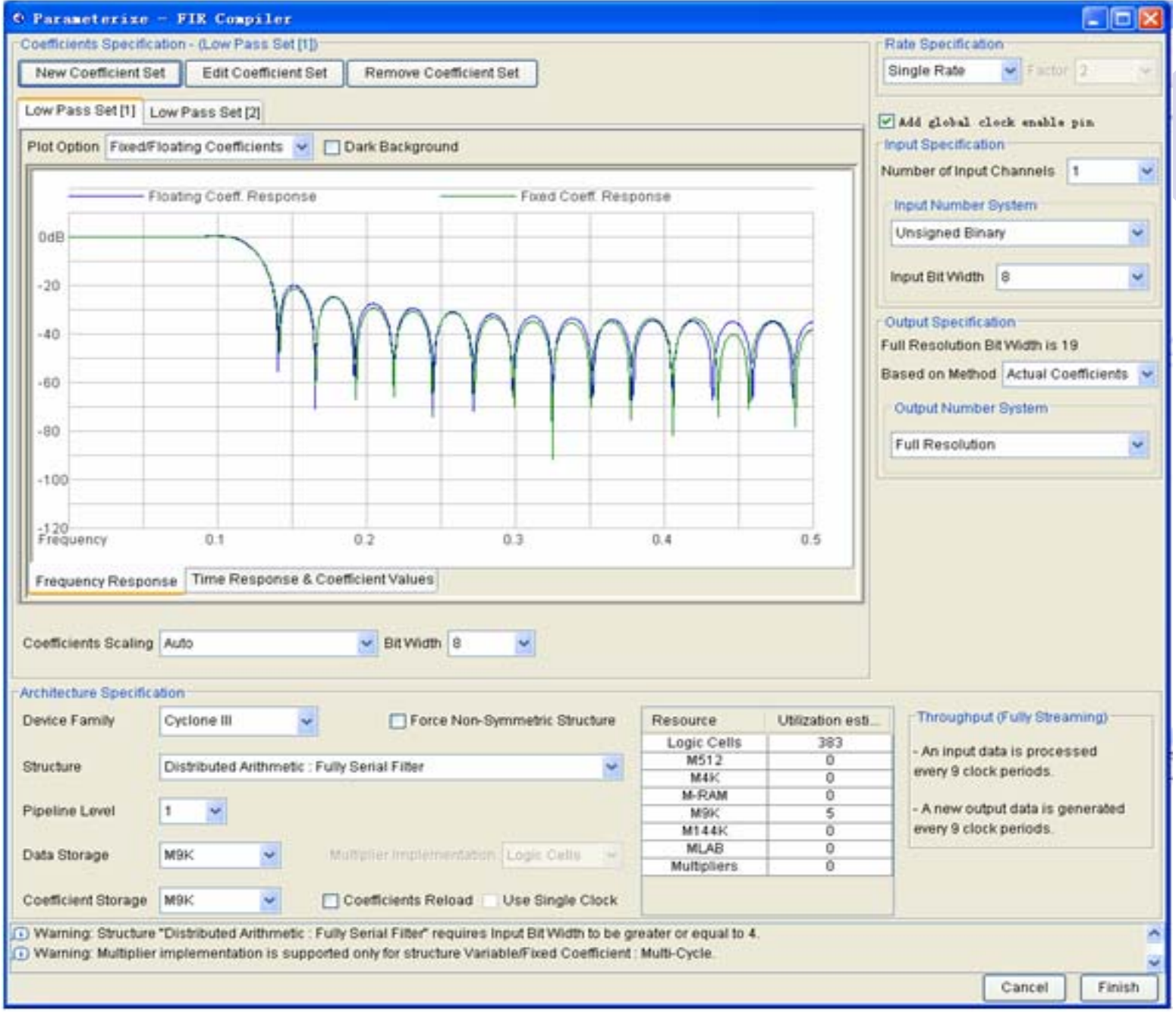

图 6-67 FIR 滤波器系数确定

6.11 8051单片机IP核应用

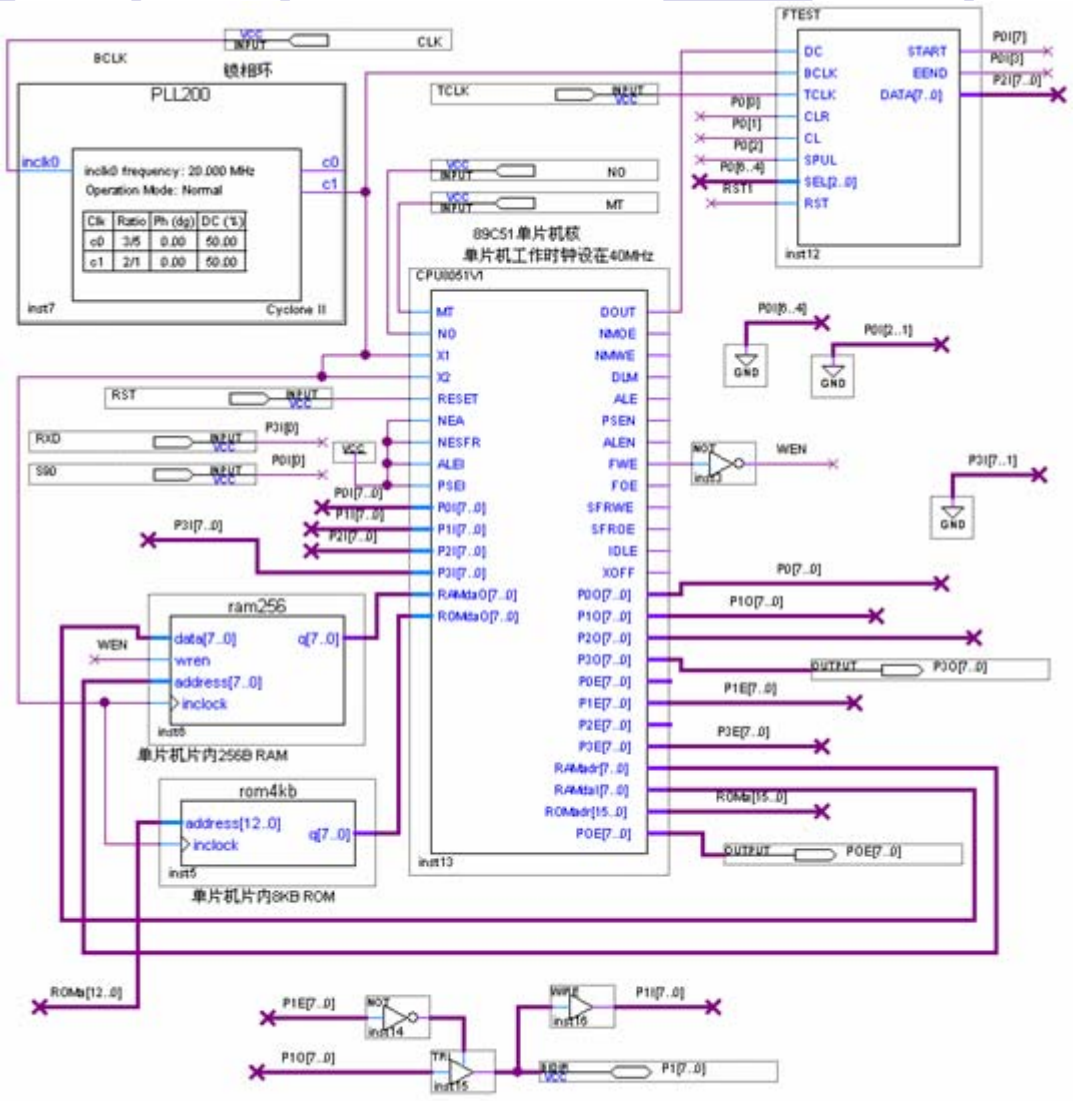

图 6-68 基本 8051CPU 核应用电路示例

6.11 8051单片机IP核应用

```
        ORG  0000H
MAIN :  MOV   SP,#60H
        MOV   24H,#00H
        MOV   30H,#01H
ROUND:  LCALL  DELAY1
        MOV   A,24H
        INC   A
        MOV   24H,A
        MOV   P1,A
        MOV   A,30H
        RR    A
        MOV   P0,A
        MOV   30H,A
        NOP
        NOP
        MOV   A,P0
        MOV   B,P3
        ADD   A,B
        MOV   P2,A
        LCALL  DELAY1
        SJMP  ROUND
```

图 6-69　test1.asm 汇编程序

6.11 8051单片机IP核应用

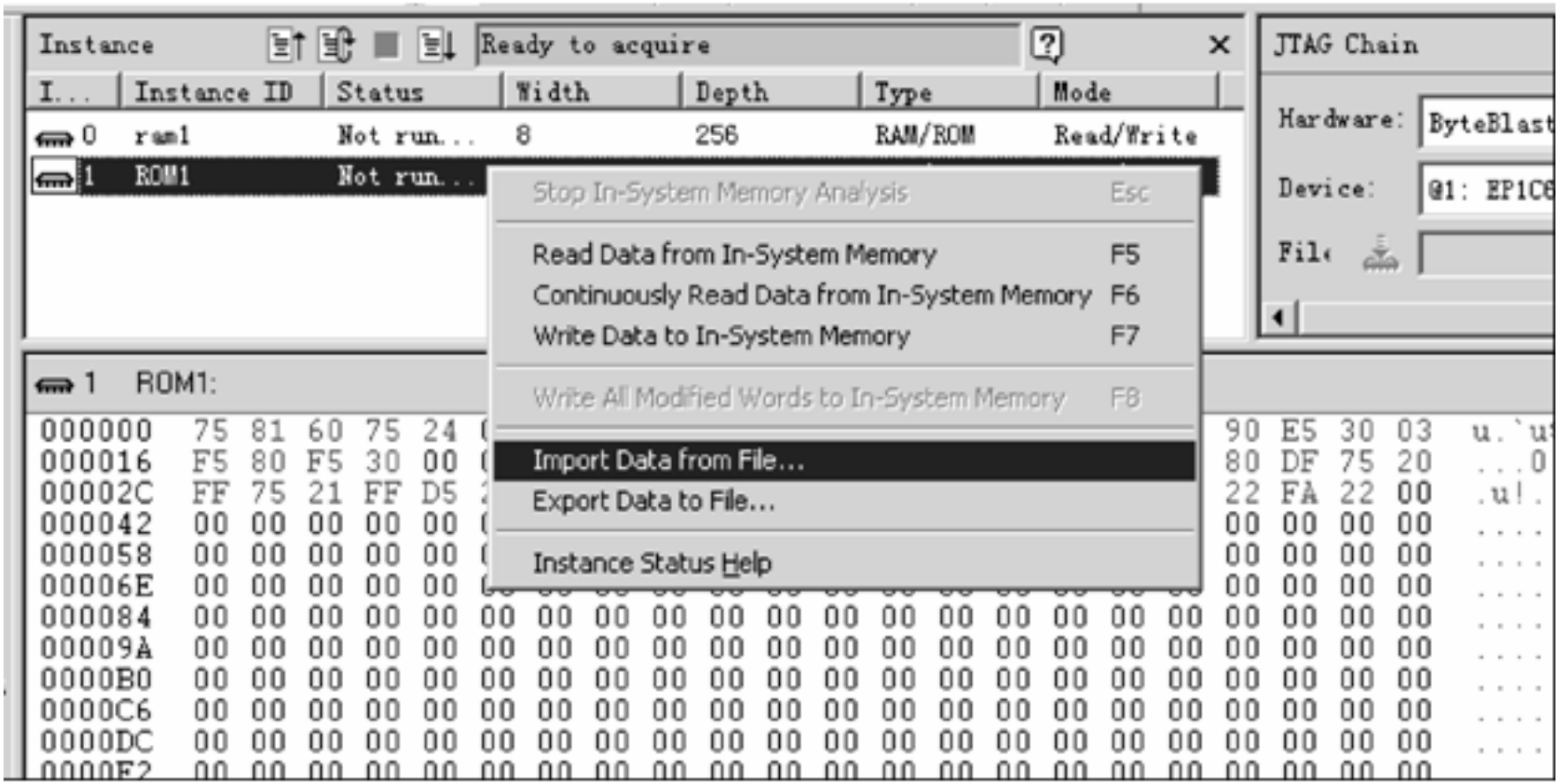

图 6-70 用 In-System Memory Content Editor 下载汇编程序代码

6.12 DDS实现原理与应用

6.12.1 DDS实现原理

$$S_{out} = A\sin \omega t = A\sin(2\pi f_{out} t) \qquad (6\text{-}3)$$

$$\theta = 2\pi f_{out} t \qquad (6\text{-}4)$$

$$\Delta\theta = 2\pi f_{out} T_{clk} = \frac{2\pi f_{out}}{f_{clk}} \qquad (6\text{-}5)$$

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

6.12 DDS实现原理与应用

6.12.1 DDS实现原理

$$S_{out} = A\sin(\theta_{k-1} + \Delta\theta) = A\sin\left[\frac{2\pi}{2^N}\cdot\left(B_{\theta_{k-1}} + B_{\Delta\theta}\right)\right] = Af_{sin}\left(B_{\theta_{k-1}} + B_{\Delta\theta}\right) \quad (6\text{-}7)$$

$$B_{\theta_{k-1}} \approx \frac{\theta_{k-1}}{2\pi}\cdot 2^N \quad (6\text{-}8)$$

$$f_{out} = \frac{B_{\Delta\theta}}{2^N}\cdot f_{clk} \quad (6\text{-}9)$$

$$f_{out} = \frac{f_{clk}}{2^N} \quad (6\text{-}10)$$

6.12 DDS实现原理与应用

6.12.1 DDS实现原理

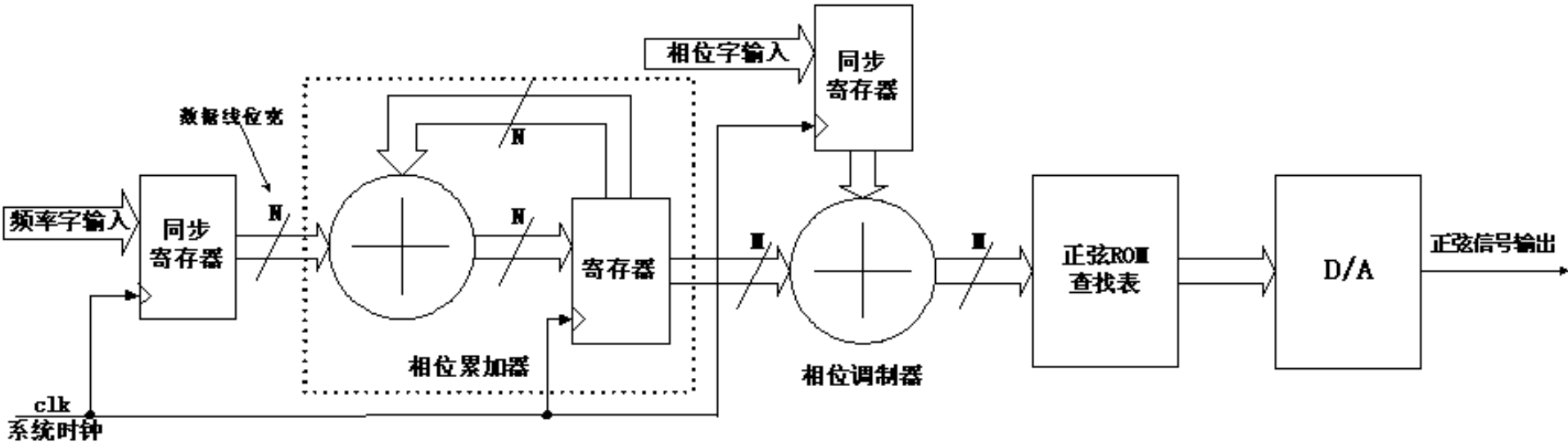

图 6-71 基本 DDS 结构

6.12 DDS实现原理与应用

6.12.2 DDS信号发生器设计

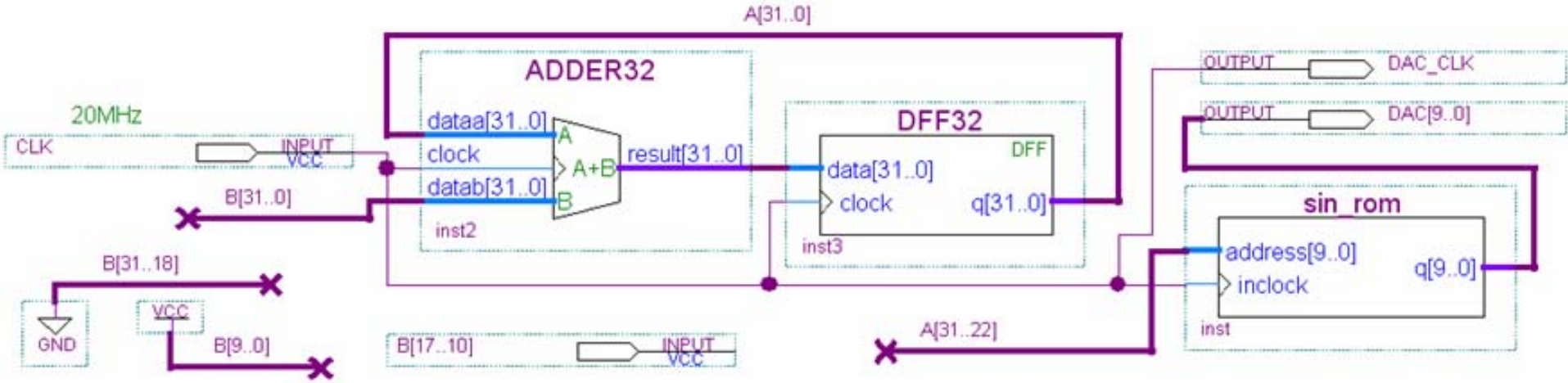

图 6-72　DDS 信号发生器电路顶层原理图

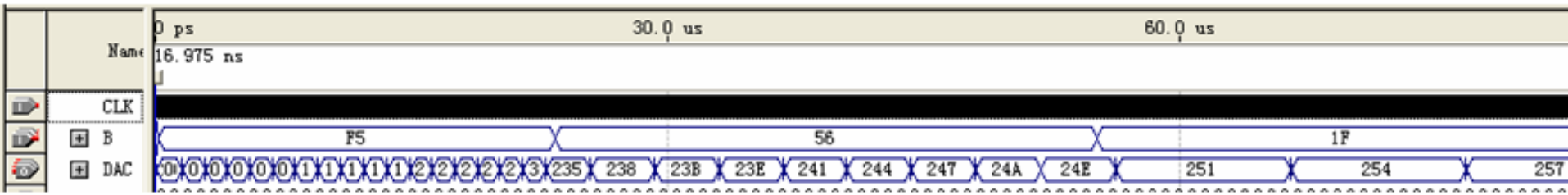

图 6-73　图 6-72 电路的仿真波形

习　题

6-1 如果不使用**MegaWizard Plug-In Manager**工具，如何在自己的设计中调用**LPM**模块？以计数器**lpm_counter** 为例，写出调用该模块的程序，其中参数自定。

6-2 LPM_ROM/RAM/FIFO等模块与**FPGA**中嵌入的**EAB**、**M9K**有怎样的联系？

6-3 参考**Quartus II**的**Help**（**Contents**），详细说明**LPM**元件**altcam**、**altsyncram**、**lpm_fifo** 、**lpm_shiftreg**的使用方法，以及其中各参量的含义和设置方法。

6-4 如果要设计一**8051**单片机应用系统，如何为它配置含有汇编程序代码的**ROM**（文件）？

实验与设计

6-1．查表式硬件运算器设计

(1) 实验原理：

(2)实验内容**1**：

(3) 实验内容**2** ：

$$f(x,y)=\sqrt{8SIN^2(2\pi xy)+2COS^2(\frac{\pi}{2}x)}$$

实验与设计

【例 6-11】

```
  WIDTH = 8 ;
  DEPTH = 256 ;
  ADDRESS_RADIX = HEX ;
  DATA_RADIX = HEX ;
  CONTENT  BEGIN
00:00; 01:00; 02:00; 03:00; 04:00; 05:00; 06:00; 07:00; 08:00; 09:00;
10:00; 11:01; 12:02; 13:03; 14:04; 15:05; 16:06;  17:07; 18:08; 19:09;
20:00; 21:02; 22:04; 23:06; 24:08; 25:10; 26:12; 27:14; 28:16; 29:18;
30:00; 31:03; 32:06; 33:09; 34:12; 35:15; 36:18; 37:21; 38:24; 39:27;
40:00; 41:04; 42:08; 43:12; 44:16; 45:20; 46:24; 47:28; 48:32; 49:36;
50:00; 51:05; 52:10; 53:15; 54:20; 55:25; 56:30; 57:35; 58:40; 59:45;
60:00; 61:06; 62:12; 63:18; 64:24; 65:30; 66:36; 67:42; 68:48; 69:54;
70:00; 71:07; 72:14; 73:21; 74:28; 75:35; 76:42; 77:49; 78:56; 79:63;
80:00; 81:08; 82:16; 83:24; 84:32; 85:40; 86:48; 87:56; 88:64; 89:72;
90:00; 91:09; 92:18; 93:27; 94:36; 95:45; 96:54; 97:63; 98:72; 99:81;
  END ;
```

实验与设计

6-2 简易正弦信号发生器设计

（1）实验目的：
（2）实验原理：
（3）实验内容1：
（4）实验内容2：
（5）实验内容3：
（6）实验报告：

下载：/KX_7C5EE+/EXPERIMENTs/EXP8_SINGT_DAC0832/SNGT

实验与设计

6-3 八位数码显示频率计设计

（1）实验目的：

（2）实验原理：

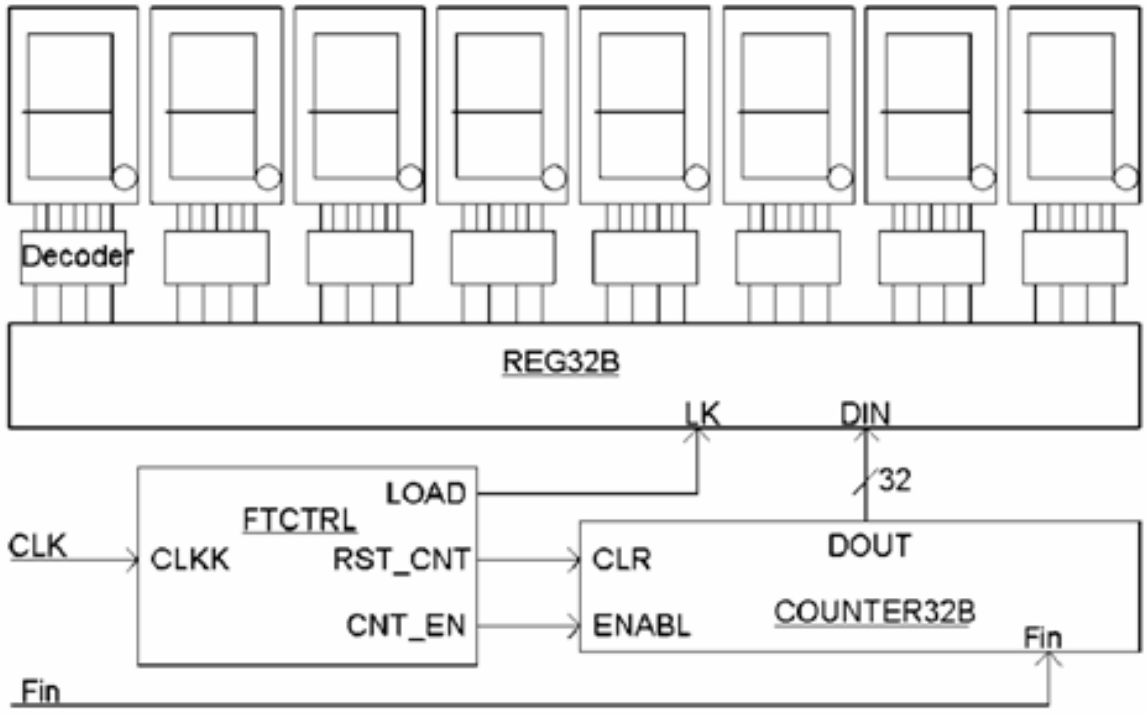

图 6-74 频率计电路框图

实验与设计

6-3 八位数码显示频率计设计

（3）实验内容1：

（4）实验内容2：

（5）实验内容3：

演示示例：**/KX_7C5EE+/EXPERIMENTs/EXP36_FTEST_HEX/F_TESTER**。

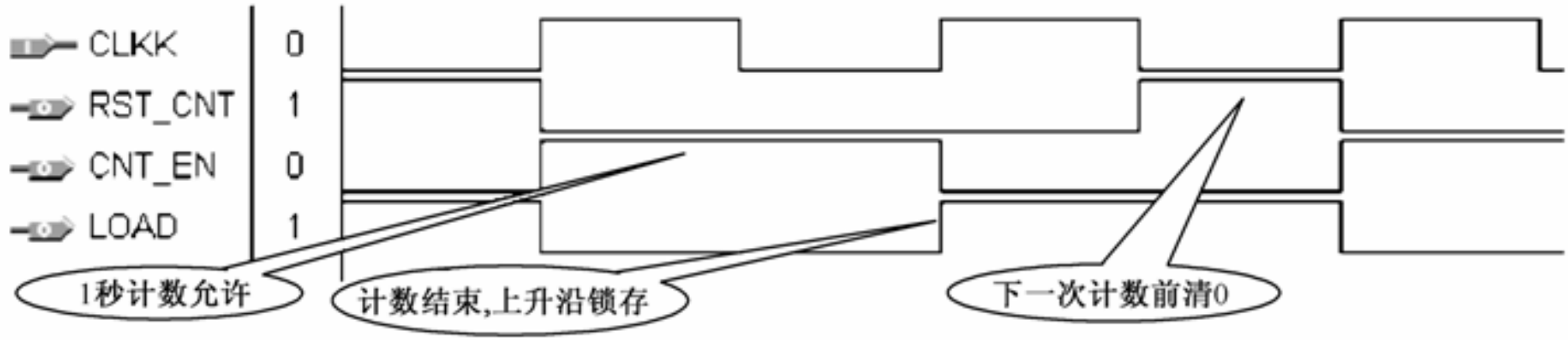

图 6-75 频率计测频控制器 FTCTRL 测控时序图

【例 6-12】

```
LIBRARY IEEE;  --测频控制电路
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY FTCTRL IS
    PORT (CLKK : IN STD_LOGIC;                    -- 1Hz
        CNT_EN : OUT STD_LOGIC;                   -- 计数器时钟使能
       RST_CNT : OUT STD_LOGIC;                   -- 计数器清零
         Load : OUT STD_LOGIC    );               -- 输出锁存信号
 END FTCTRL;
ARCHITECTURE behav OF FTCTRL IS
    SIGNAL Div2CLK : STD_LOGIC;
BEGIN
    PROCESS( CLKK )    BEGIN
        IF CLKK'EVENT AND CLKK = '1' THEN      -- 1Hz 时钟 2 分频
            Div2CLK <= NOT Div2CLK;
        END IF;
    END PROCESS;
    PROCESS (CLKK, Div2CLK)    BEGIN
    IF CLKK='0' AND Div2CLK='0' THEN RST_CNT<='1';-- 产生计数器清零信号
         ELSE RST_CNT <= '0';  END IF;
    END PROCESS;
    Load  <= NOT Div2CLK;    CNT_EN <= Div2CLK;
END behav;
```

实验与设计

6-4. 简易逻辑分析仪设计

（1）实验原理：

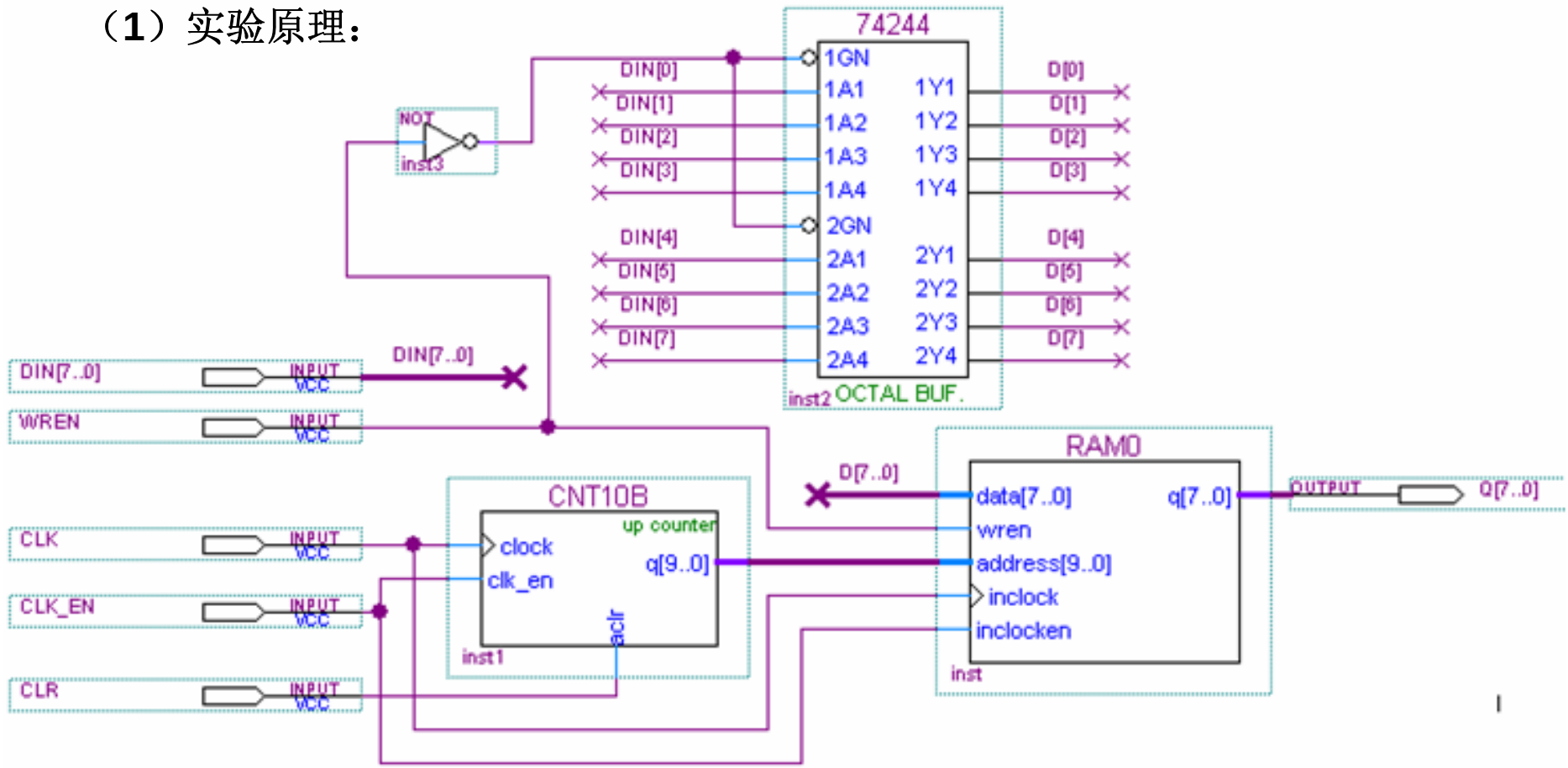

图 6-76 逻辑数据采样电路顶层设计

实验与设计

（2）实验任务1：

（3）实验任务2：

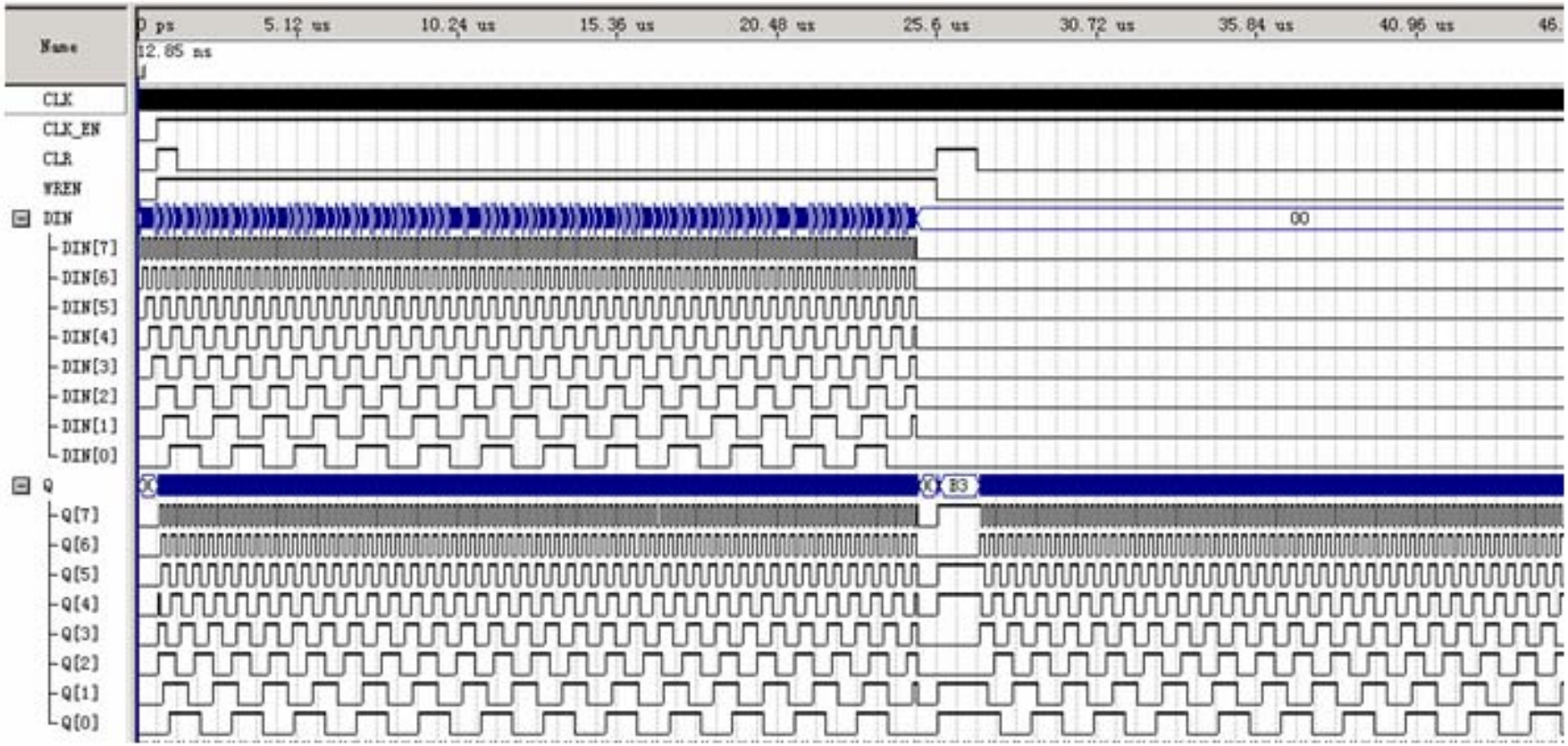

图 6-77 逻辑数据采样电路时序仿真波形

实验与设计

6-5 DDS信号发生器设计

（1）实验目的：
（2）实验原理：
（3）实验内容1：
（4）实验内容2：
（5）实验内容3：
（6）实验内容4：
（7）实验内容5：
（8）思考题：

演示示例：/KX_7C5EE+/EXPERIMENTs/EXP23_DDS/DDSP。

实验与设计

6-6 DDS移相信号发生器设计

（1）实验原理：

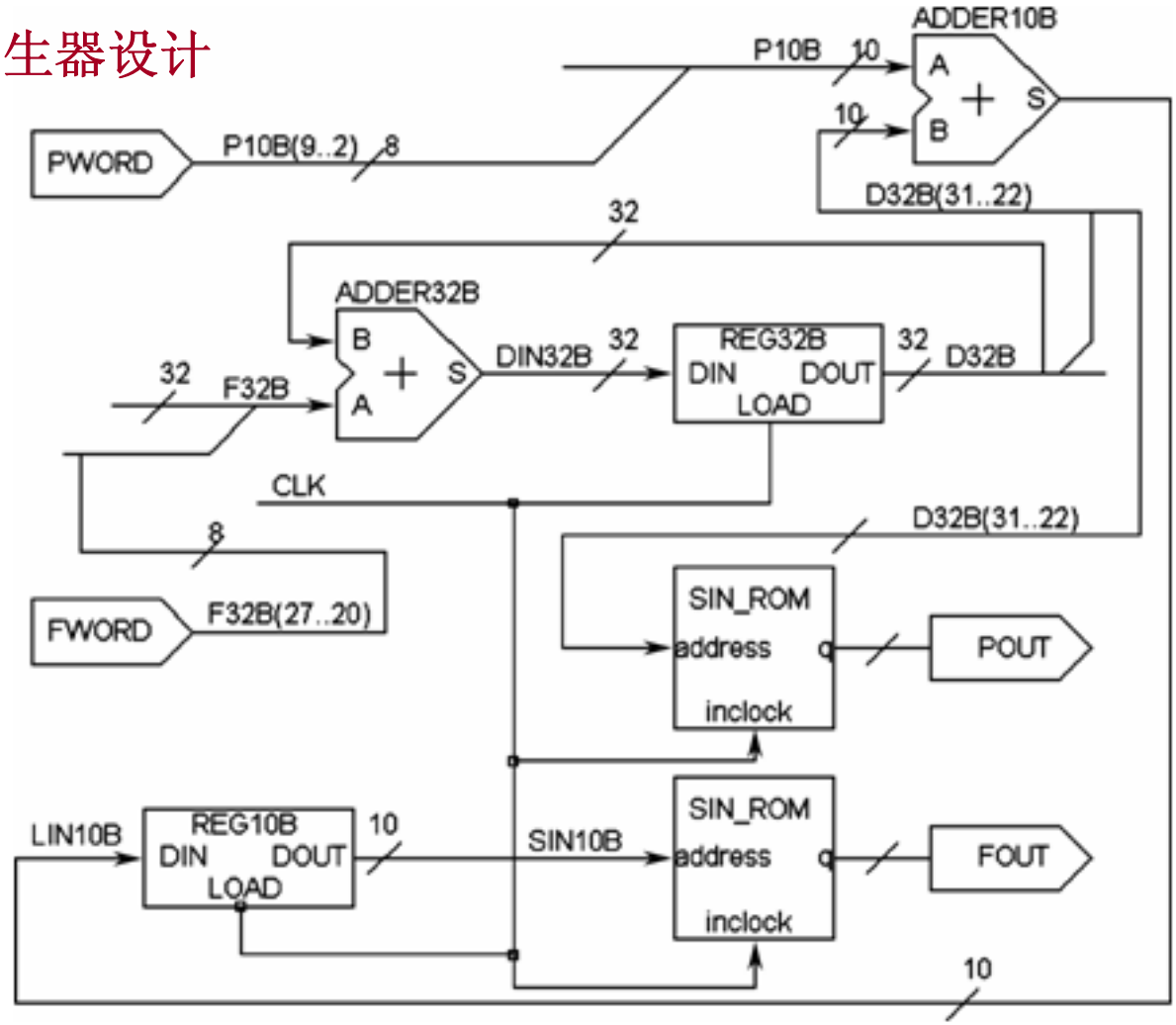

图 6-78 数字移相信号发生器电路模型图

实验与设计

（2）实验内容1：
（3）实验内容2：
（4）实验内容3：
（5）思考题：
（6）实验报告：
演示示例：/KX_7C5EE+/EXPERIMENTs/EXP9_DDS_PHASE_2DAC/DDSP。

实验与设计

6-7 4X4阵列键盘键信号检测电路设计

（1）实验原理：

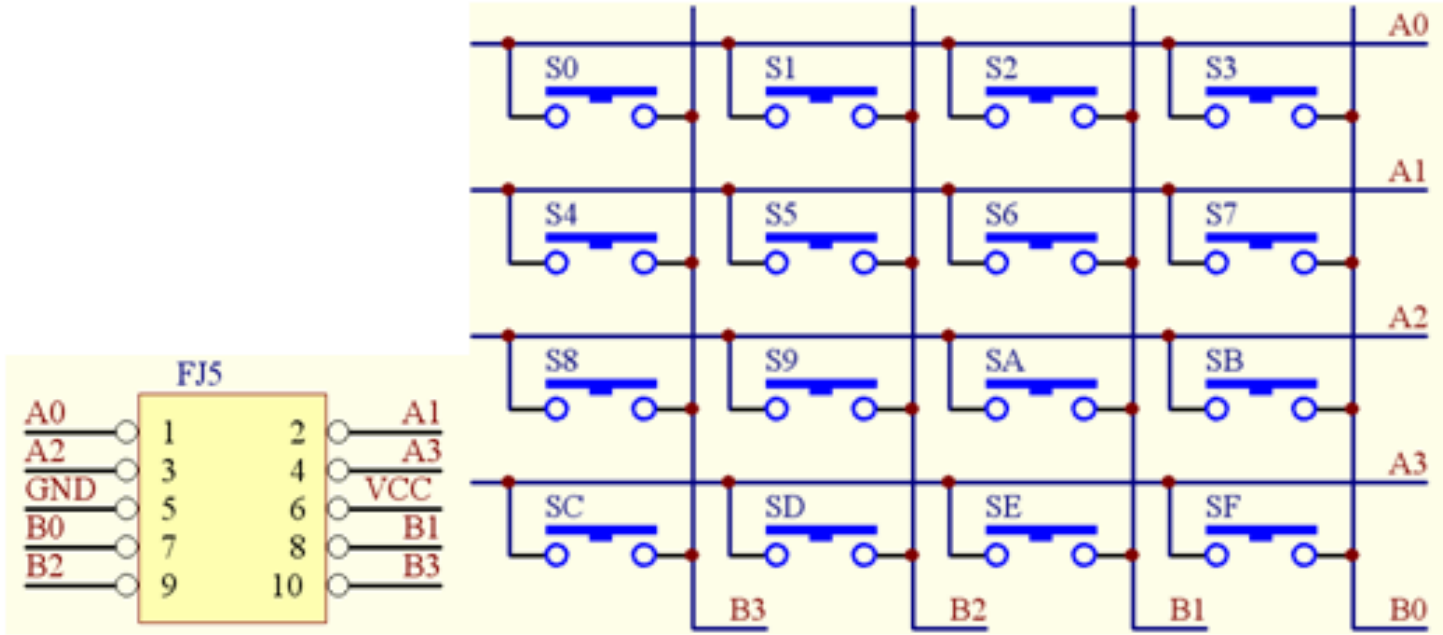

图 6-79 4X4 键盘电路和 10 芯接口

实验与设计

6-7 4X4阵列键盘键信号检测电路设计

（1）实验原理：

图 6-80 设置端口上拉

【例 6-13】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
 USE IEEE.STD_LOGIC_1164.ALL ;
 ENTITY K44 IS
  PORT (CLK  : IN  STD_LOGIC;
        A  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
      B,R  : OUT  STD_LOGIC_VECTOR(3 DOWNTO 0)) ;
 END ;
 ARCHITECTURE one OF K44 IS
  SIGNAL C  : STD_LOGIC_VECTOR(1 DOWNTO 0);
  SIGNAL BA : STD_LOGIC_VECTOR(7 DOWNTO 0);
  SIGNAL E  : STD_LOGIC_VECTOR(3 DOWNTO 0);
  BEGIN
      BA<=E & A;   B<=E;
  PROCESS( A )   BEGIN
   IF RISING_EDGE(CLK)  THEN    C<=C+1;
  CASE  C  IS
```

接下页

实验与设计

```
 WHEN "00" =>  E <= "0111" ; WHEN "01" =>  E <= "1011" ;
 WHEN "10" =>  E <= "1101" ; WHEN "11" =>  E <= "1110" ;
 WHEN OTHERS =>  NULL ;
 END CASE ;
 CASE  BA  IS
 WHEN "01111110" => R<="0000" ; WHEN "01111101" => R<="0001" ;
 WHEN "01111011" => R<="0010" ; WHEN "01110111" => R<="0011" ;
 WHEN "10111110" => R<="0100" ; WHEN "10111101" => R<="0101" ;
 WHEN "10111011" => R<="0110" ; WHEN "10110111" => R<="0111" ;
 WHEN "11011110" => R<="1000" ; WHEN "11011101" => R<="1001" ;
 WHEN "11011011" => R<="1010" ; WHEN "11010111" => R<="1011" ;
 WHEN "11101110" => R<="1000" ; WHEN "11101101" => R<="1001" ;
 WHEN "11101011" => R<="1010" ; WHEN "11100111" => R<="1011" ;
 WHEN OTHERS =>  NULL ;
 END CASE ;      END IF;
 END PROCESS ;
END ;
```

实验与设计

（2）实验任务1：
（3）实验任务2：
（4）实验任务3：
（5）实验任务4：
（6）实验任务5：
（7）实验任务6：

5E+系统演示示例：/KX_7C5EE+/EXPERIMENTs/EXP31_SCAN_4X4KEY/。

实验与设计

6-8 8051单片机IP核SOC片上系统设计实验

（1）实验内容1：

（2）实验内容2：

（3）实验内容3：

（4）实验内容4：

5E+系统演示示例：

/KX_7C5EE+/EXPERIMENTs/EXP15_8051_Core_BASIC/MCU8951。

实验与设计

6-9 VGA简单图像显示控制模块设计

（1）实验原理：

（2）实验内容1：

（3）实验内容2：

（4）实验内容3：

此项设计基于5E+系统的演示示例：

/KX_7C5EE+/EXPERIMENTs/EXP12_VGA_img/VGA，和

/KX_7C5EE+/EXPERIMENTs/EXP27_VGA_PCT/VGA_PCT。

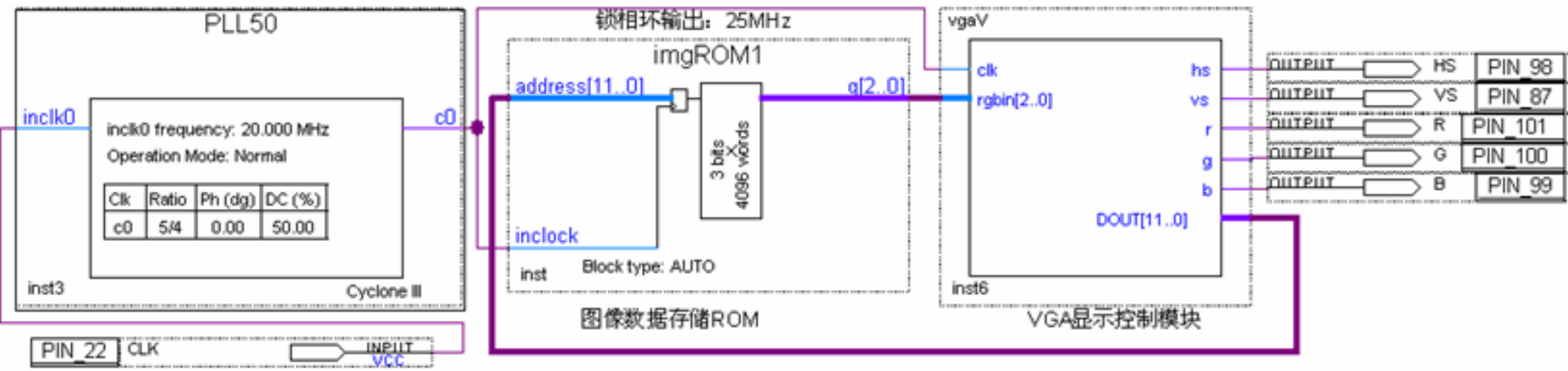

图 6-81 VGA 图像显示控制模块原理图