EDA 技术实用教程

第 7 章

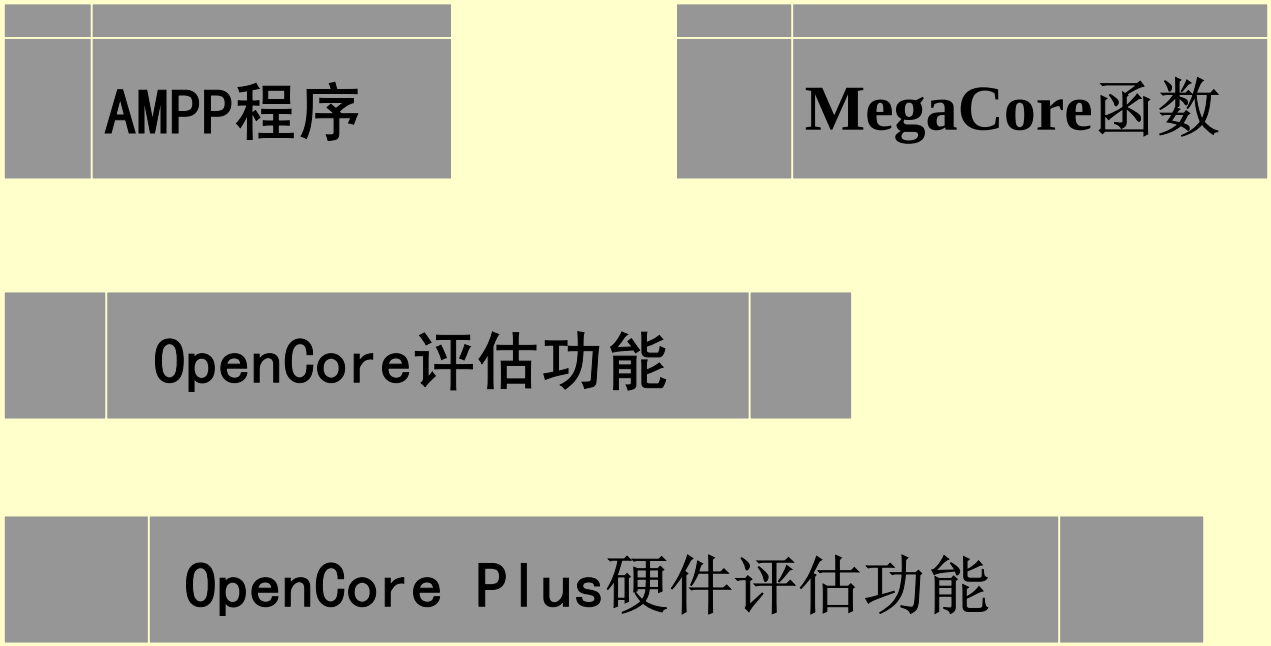
宏功能模块与IP应用

7.1 宏功能模块概述

算术组件	累加器、加法器、乘法器和LPM算术函数
门电路	多路复用器和LPM门函数
I/O组件	时钟数据恢复(CDR)、锁相环(PLL)、双数据速率(DDR)、千兆位收发器块(GXB)、LVDS接收器和发送器、PLL重新配置和远程更新宏功能模块
存储器编译器	FIFO Partitioner、RAM和ROM宏功能模块
存储组件	存储器、移位寄存器宏模块和LPM存储器函数

7.1 宏功能模块概述

7.1.1 知识产权核的应用



7.1 宏功能模块概述

7.1.2 使用MegaWizard Plug-In Manager

<输出文件>.bsf ： Block Editor中使用的宏功能模块的符号（元件）。

<输出文件>.cmp： 组件申明文件。

<输出文件>.inc ： 宏功能模块包装文件中模块的AHDL包含文件。

<输出文件>.tdf： 要在AHDL设计中实例化的宏功能模块包装文件。

<输出文件>.vhd： 要在VHDL设计中实例化的宏功能模块包装文件。

<输出文件>.v ： 要在VerilogHDL设计中实例化的宏功能模块包装文件。

<输出文件>_bb.v： VerilogHDL设计所用宏功能模块包装文件中模块的空体或black-box申明，用于在使用EDA 综合工具时指定端口方向。

<输出文件>_inst.tdf： 宏功能模块包装文件中子设计的AHDL例化示例。

<输出文件>_inst.vhd： 宏功能模块包装文件中实体的VHDL例化示例。

<输出文件>_inst.v ： 宏功能模块包装文件中模块的VerilogHDL例化示例。

7.1 宏功能模块概述

7.1.3 在QuartusII中对宏功能模块进行例化

1、在VerilogHDL和VHDL中例化

2、使用端口和参数定义

3、使用端口和参数定义生成宏功能模块

计数器	乘-累加器和乘-加法器
加法/减法器	RAM
乘法器	移位寄存器

7.2 宏模块应用实例

7.2.1 工作原理

$$f = f_0 / 64$$

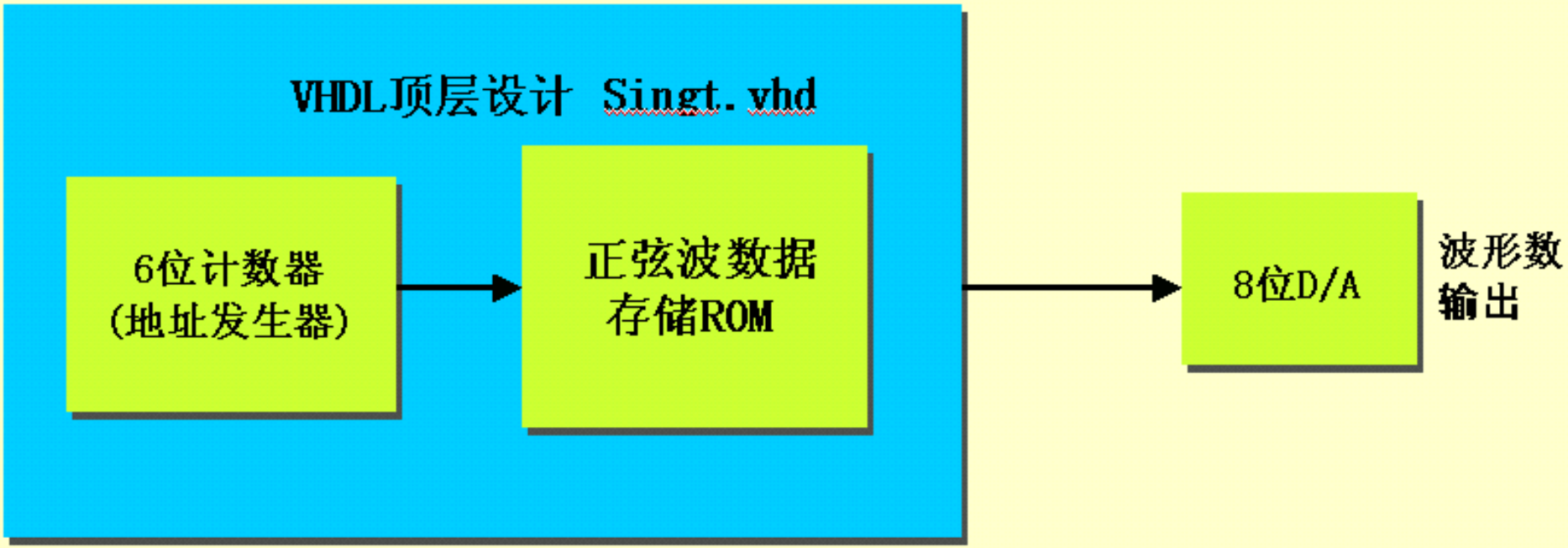


图7-1 正弦信号发生器结构框图

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

1. 建立.mif格式文件

【例7-1】

```
WIDTH = 8;
DEPTH = 64;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT BEGIN
0        :       FF;
1        :       FE;
2        :       FC;
3        :       F9;
4        :       F5;
...（数据略去）
3D        :        FC;
3E        :        FE;
3F        :        FF;
END;
```

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

1. 建立.mif格式文件

【例7-2】

```
#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));
    }
}
```

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

2. 建立.hex格式文件

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	255	254	252	249	245	239	233	225
8	217	207	197	186	174	162	150	137
16	124	112	99	87	75	64	53	43
24	34	26	19	13	8	4	1	0
32	0	1	4	8	13	19	26	34
40	43	53	64	75	87	99	112	124
48	137	150	162	174	186	197	207	217
56	225	233	239	245	249	252	254	255

图7-2 将波形数据填入mif文件表中

2. 建立.hex格式文件

图7-3 ASM格式建hex文件

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

2. 建立.hex格式文件

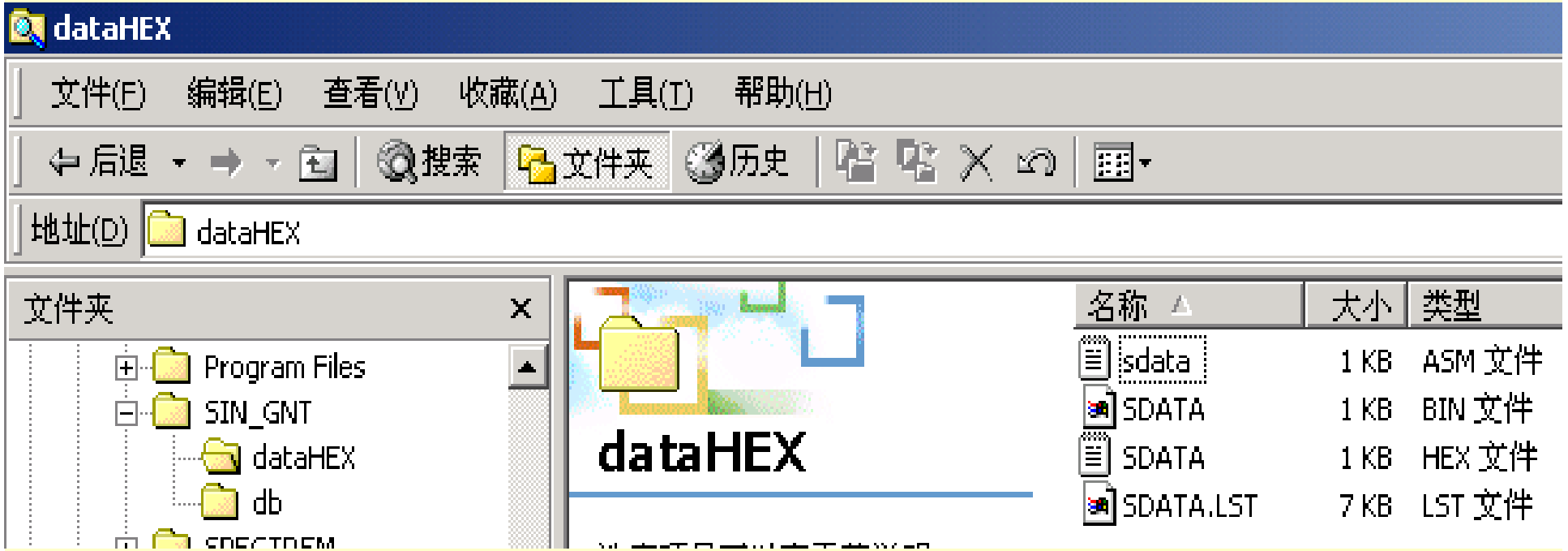


图7-4 sdata.hex文件的放置路径

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.3 定制LPM_ROM元件

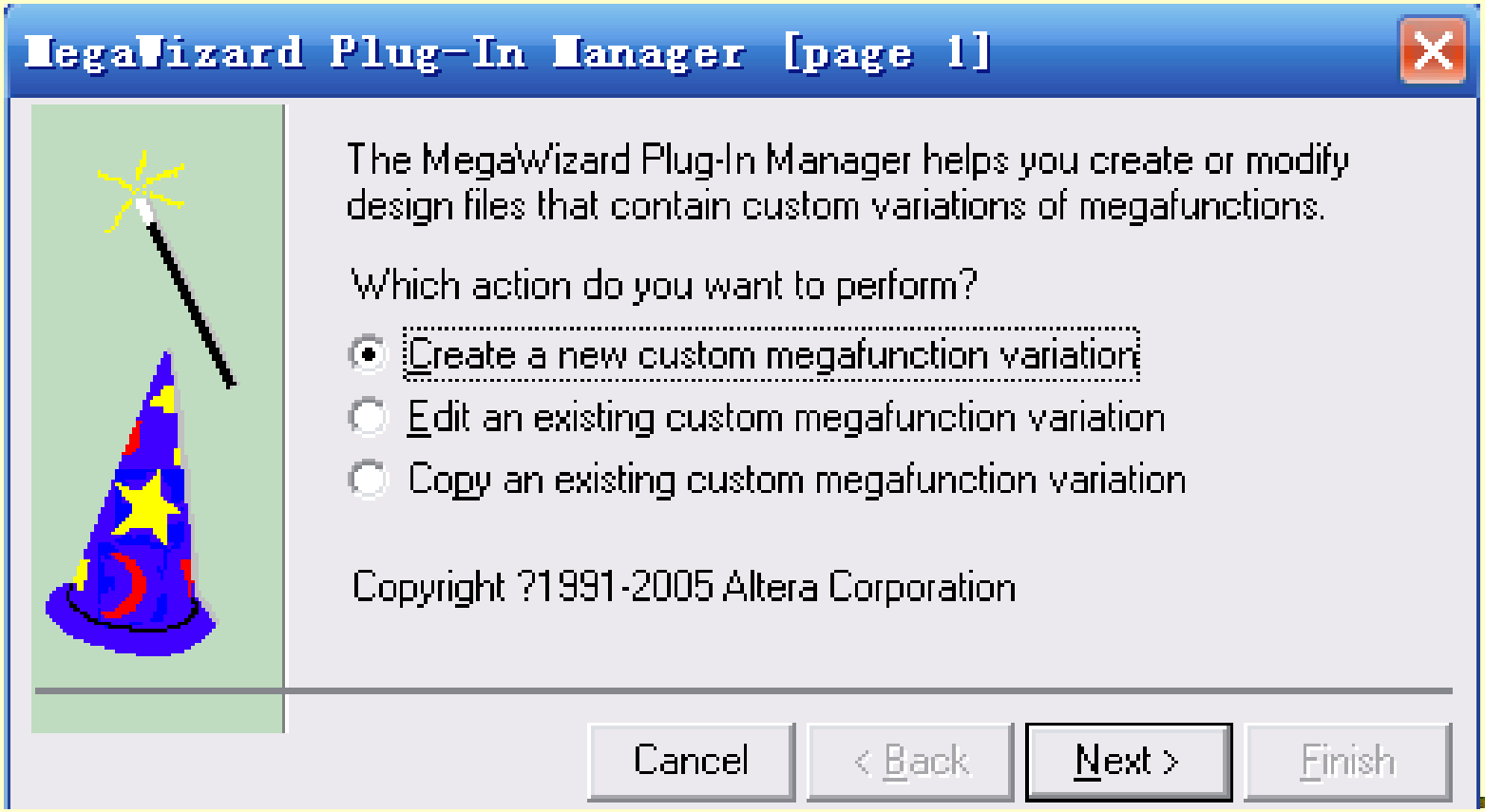


图7-5 定制新的宏功能块

康芯科技

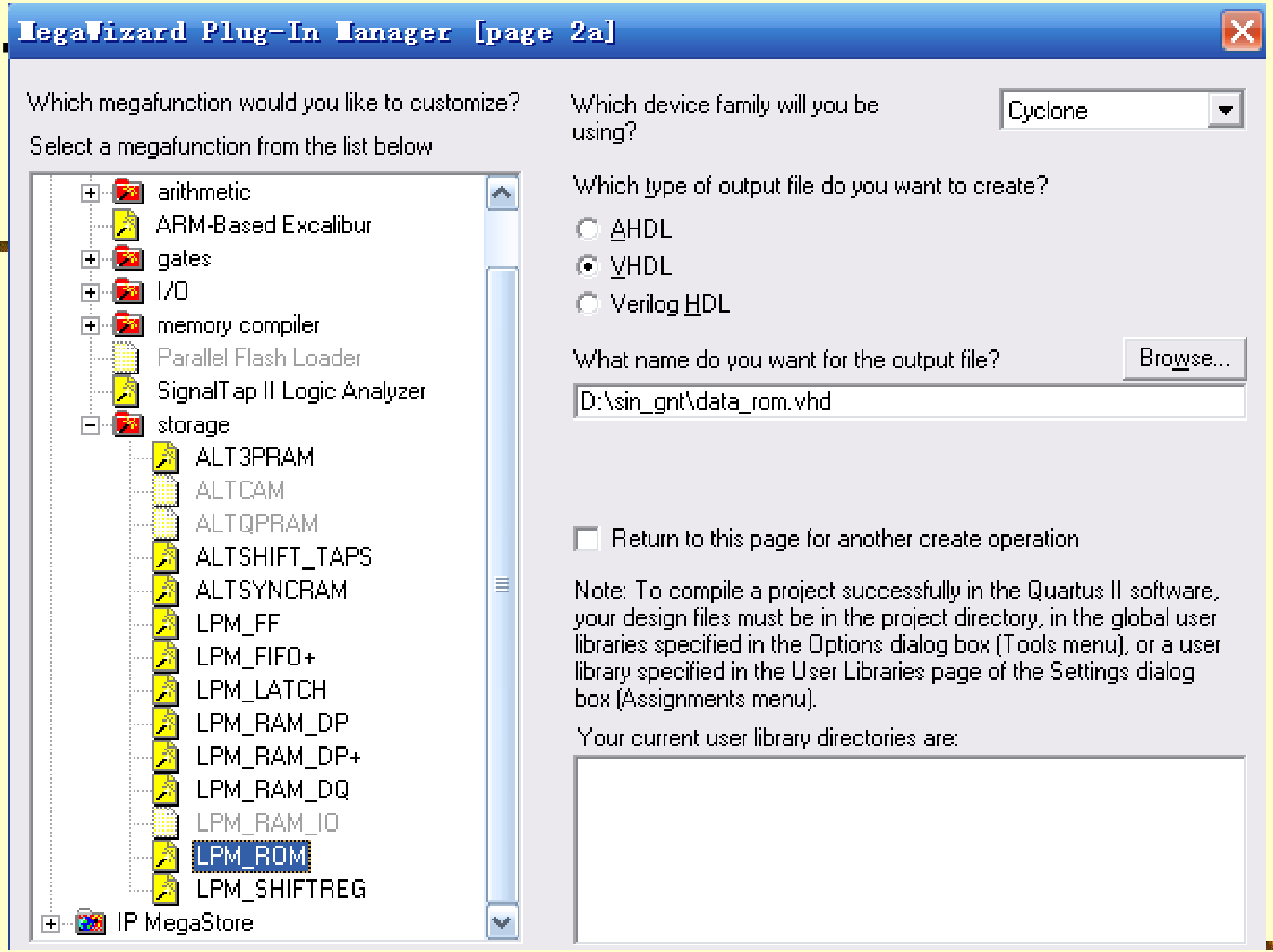


图7-6 LPM宏功能块设定

KX 康芯科技

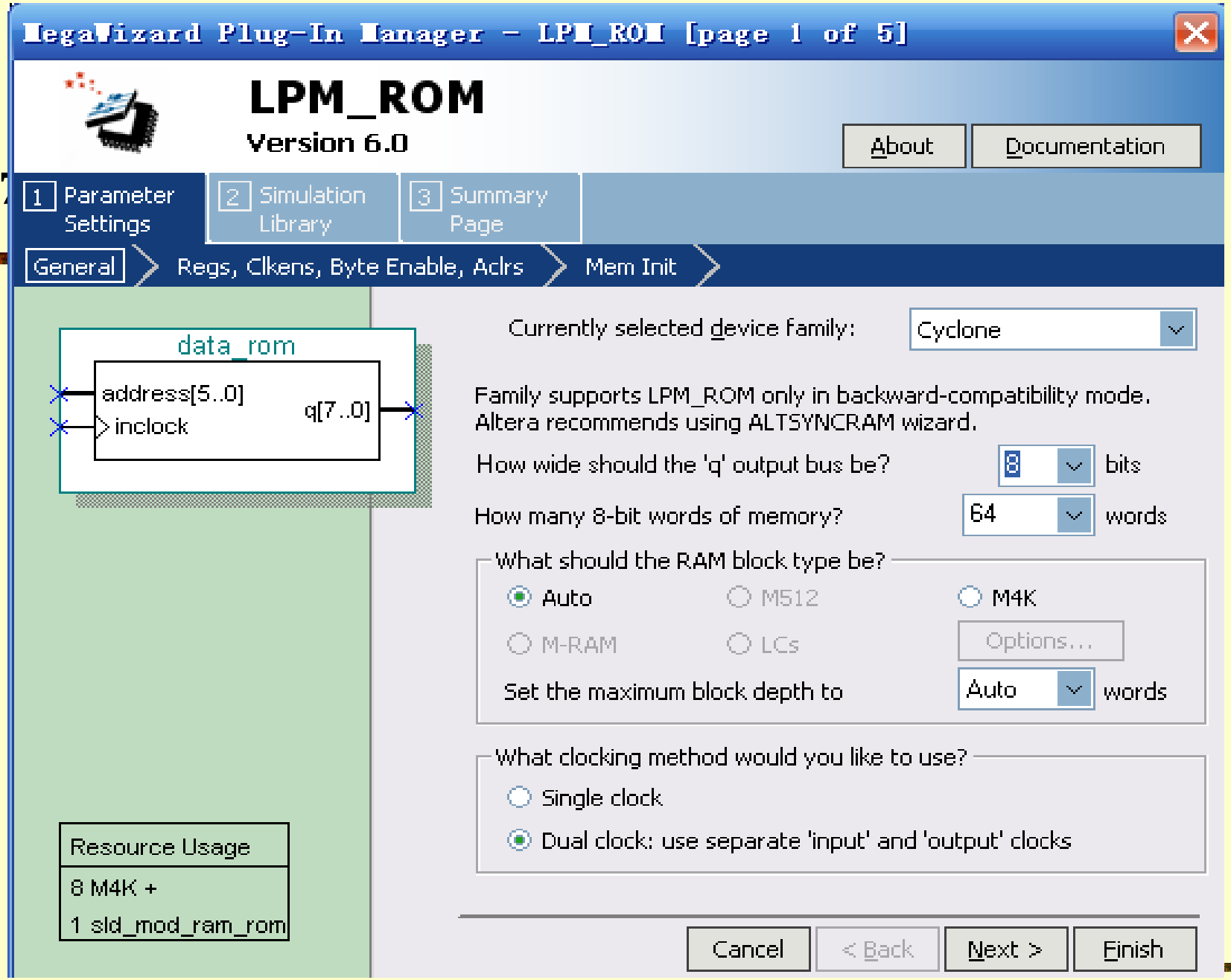


图7-7 选择data_rom模块数据线和地址线宽度

KX 康芯科技

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.3 定制LPM_ROM元件

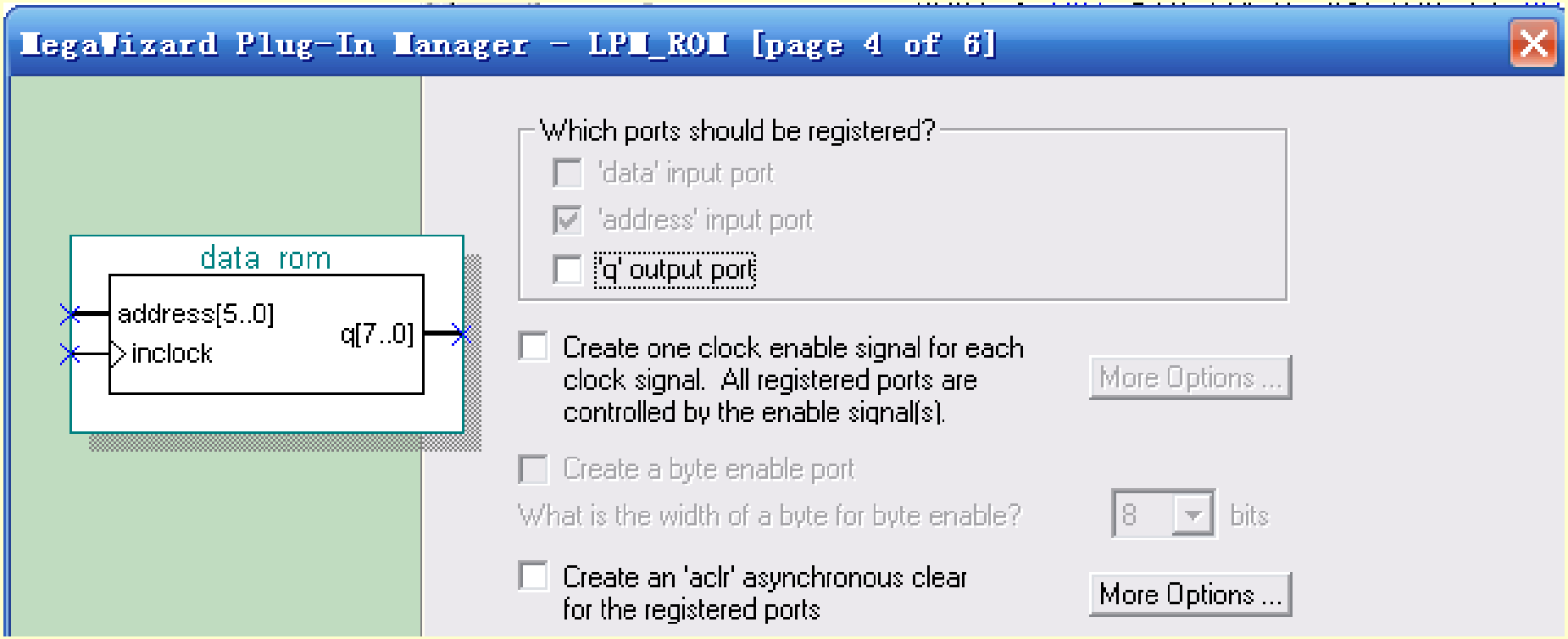


图7-8 选择地址锁存信号inclock

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.3 定制LPM_ROM元件

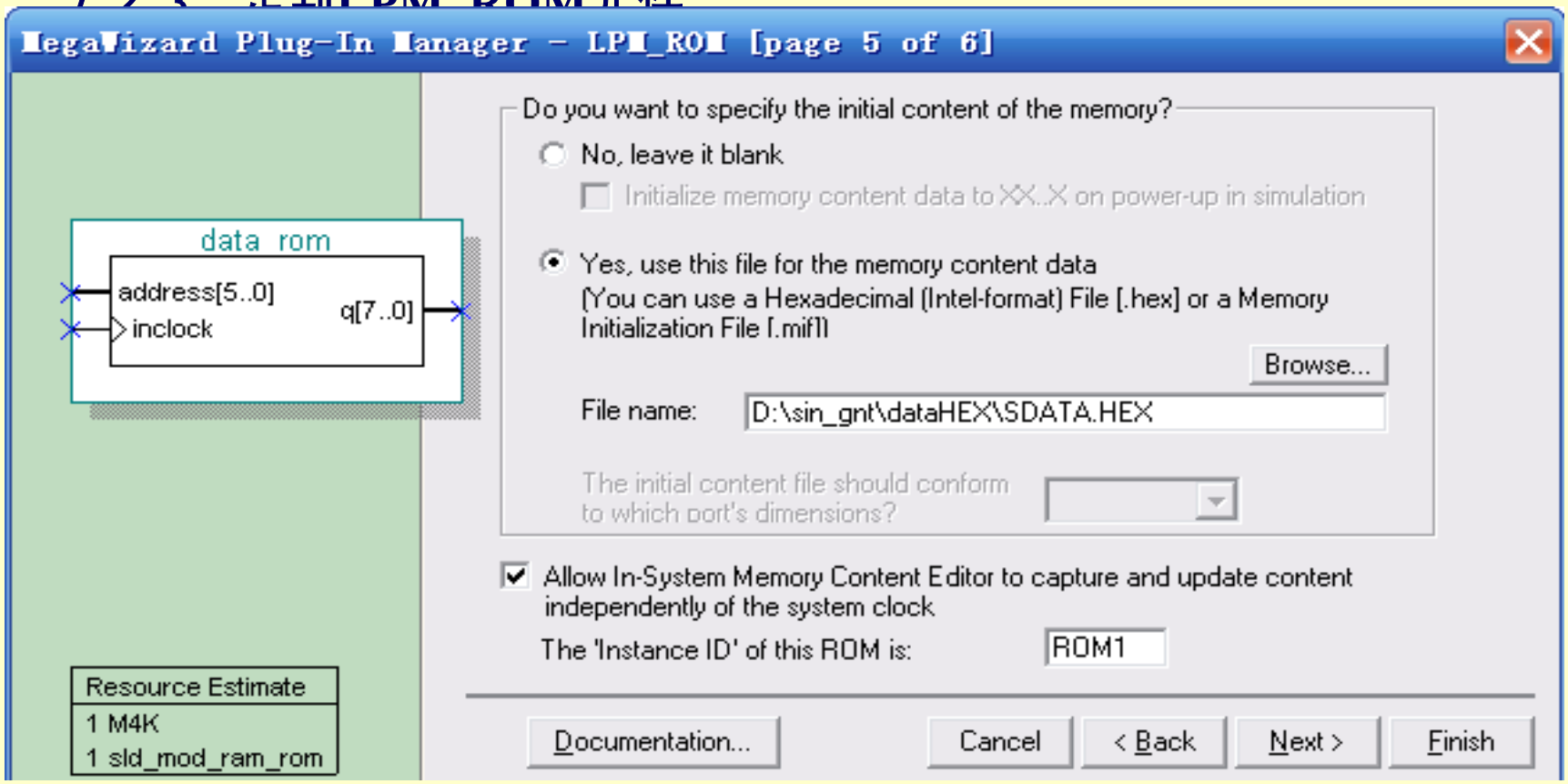


图7-9 调入ROM初始化数据文件并选择在系统读写功能

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.3 定制LPM_ROM元件

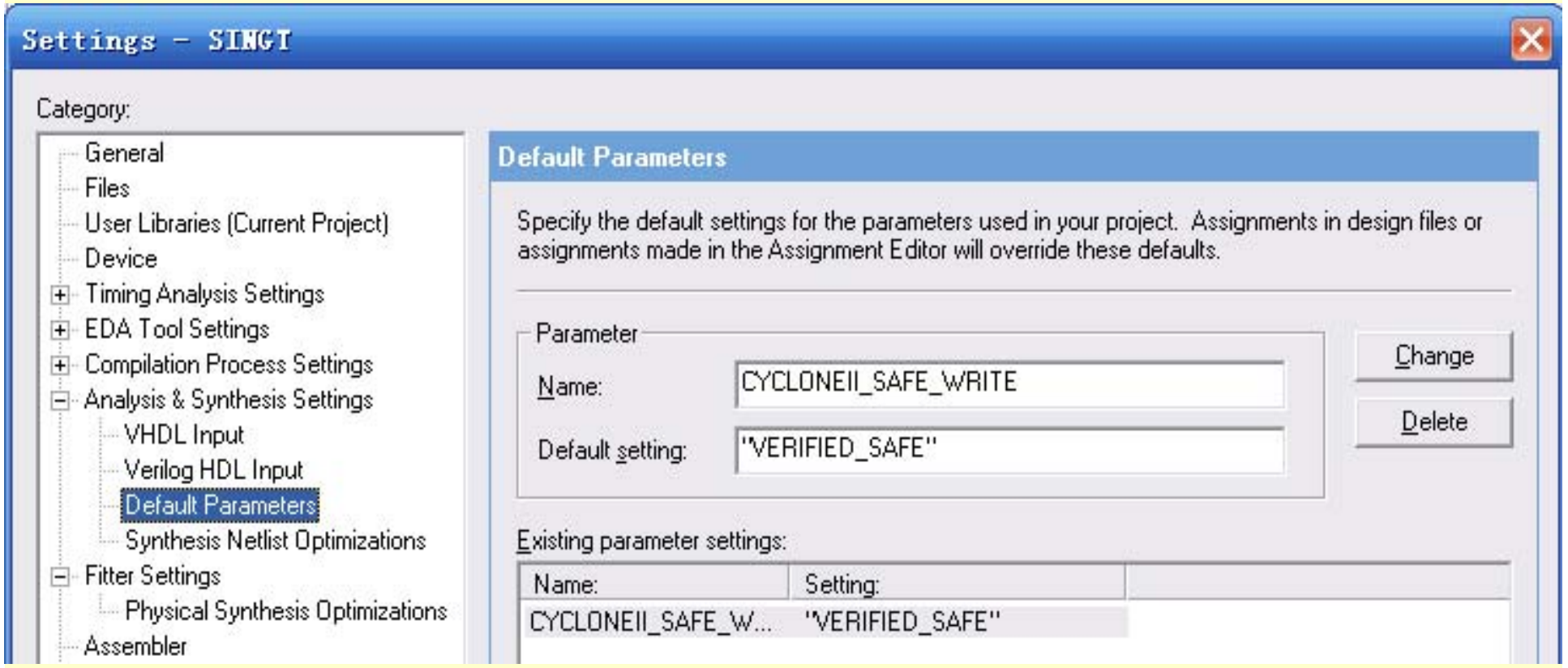


图7-10 LPM_ROM设计完成

【例7-3】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;    --使用宏功能库中的所有元件
ENTITY data_rom IS
       PORT (address    : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
               inclock   : IN STD_LOGIC ;
              q          : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
END data_rom;
ARCHITECTURE SYN OF data_rom IS
       SIGNAL sub_wire0     : STD_LOGIC_VECTOR (7 DOWNTO 0);
       COMPONENT altsyncram      --例化altsyncram元件，调用了LPM模块
altsyncram
       GENERIC (                                  --参数传递语句
              intended_device_family     : STRING;   --类属参量数据类
型定义
              width_a                : NATURAL;
              widthad_a              : NATURAL;
              numwords_a             : NATURAL;
              operation_mode               : STRING;
              outdata_reg_a          : STRING;
              address_aclr_a               : STRING;
```

接下页

```
                outdata_aclr_a                  : STRING;
                width_byteena_a                 : NATURAL;
                init_file               : STRING;
                lpm_hint                : STRING;
                lpm_type                : STRING      );
        PORT ( clock0 : IN STD_LOGIC ;        --altsyncram元件接口声明
                address_a     : IN STD_LOGIC_VECTOR (5 DOWNTO 0);
                    q_a       : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );
        END COMPONENT;
BEGIN
        q    <= sub_wire0(7 DOWNTO 0);
        altsyncram_component : altsyncram
        GENERIC MAP ( intended_device_family => "Cyclone",  --参数
传递映射
                width_a => 8,                           --数据线宽度8
                widthad_a => 6,                         --地址线宽度6
                numwords_a => 64,                       --数据数量64
                operation_mode => "ROM",                --LPM模式ROM
                outdata_reg_a => "UNREGISTERED",  --输出无锁存
                address_aclr_a => "NONE",               --无异步地址清0
                outdata_aclr_a => "NONE",               --无输出锁存异步清0
                width_byteena_a => 1,             -- byteena_a输入口宽度1
                init_file => "./dataHEX/SDATA.hex",  --ROM初始化数据文
件，此处已修改过
```

接下页

接上页

```
                    lpm_hint => "ENABLE_RUNTIME_MOD=YES,
INSTANCE_NAME=NONE",
              lpm_type => "altsyncram" )             --LPM类型
       PORT MAP ( clock0 => inclock, address_a => address,q_a =>
sub_wire0 );
```

7.2.4 完成顶层设计

【例7-4】正弦信号发生器顶层设计

```
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 SINGT IS
    PORT ( CLK  : IN STD_LOGIC;                          --信号源时钟
           DOUT : OUT STD_LOGIC_VECTOR (7 DOWNTO 0) );--8位波形数据输出
END;
ARCHITECTURE DACC OF SINGT IS
COMPONENT data_rom    --调用波形数据存储器LPM_ROM文件：data_rom.vhd声明
   PORT(address : IN STD_LOGIC_VECTOR (5 DOWNTO 0);--6位地址信号
        inclock : IN STD_LOGIC ;--地址锁存时钟
              q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)    );
END COMPONENT;
     SIGNAL Q1 : STD_LOGIC_VECTOR (5 DOWNTO 0); --设定内部节点作为地址计数器
    BEGIN
PROCESS(CLK )                              --LPM_ROM地址发生器进程
    BEGIN
IF CLK'EVENT AND CLK = '1' THEN  Q1<=Q1+1; --Q1作为地址发生器计数器
END IF;
END PROCESS;
u1 : data_rom PORT MAP(address=>Q1, q => DOUT,inclock=>CLK);--例化
END;
```

KX 康芯科技

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.4 完成顶层设计

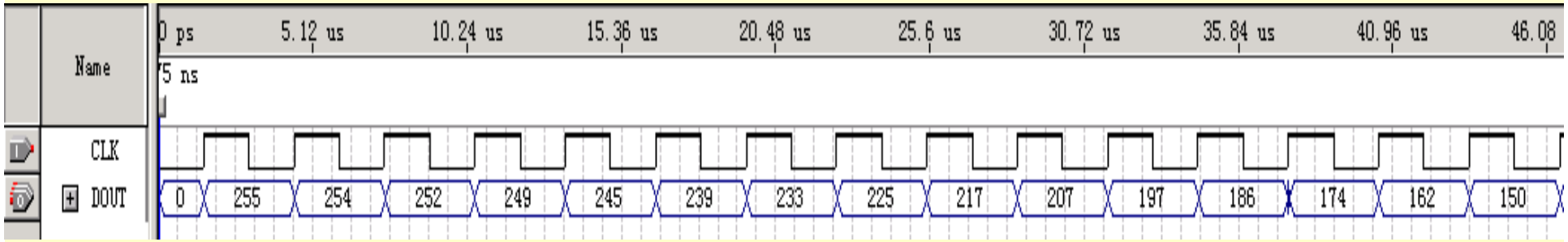


图7-11 仿真波形输出

7.2 宏模块应用实例

7.2.2 定制初始化数据文件

7.2.4 完成顶层设计

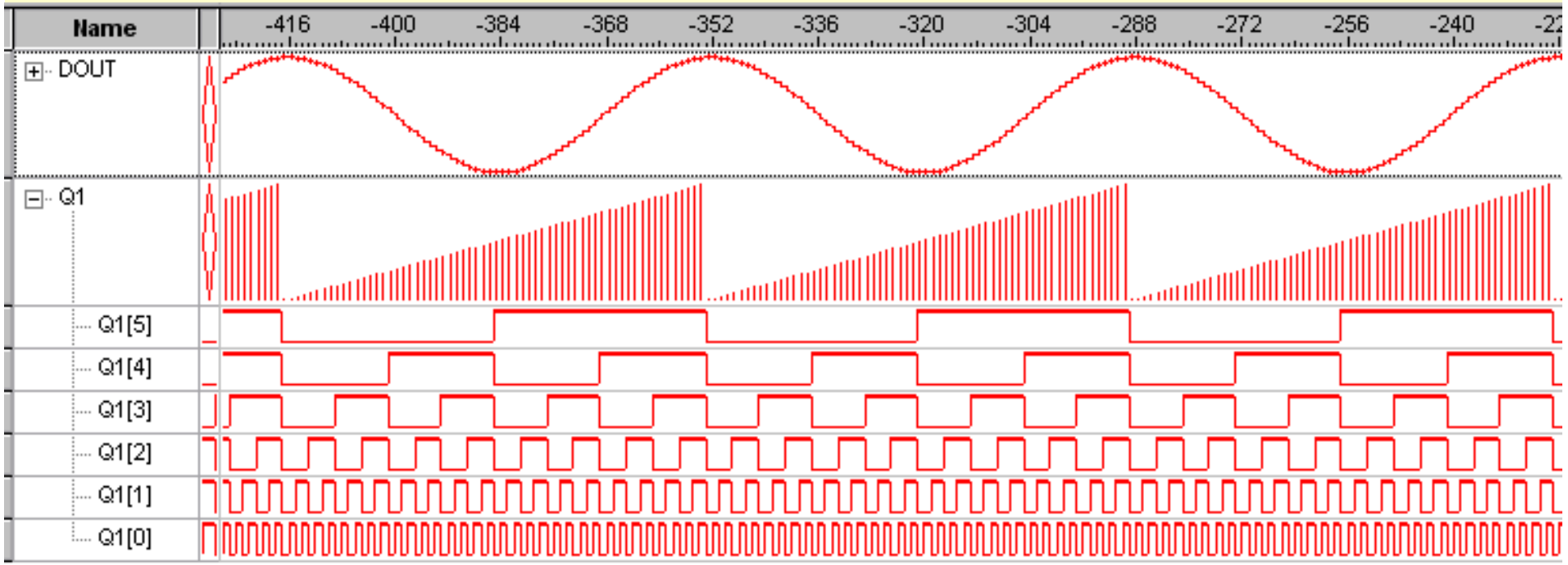


图7-12 嵌入式逻辑分析仪获得的波形

7.3 在系统存储器数据读写编辑器应用

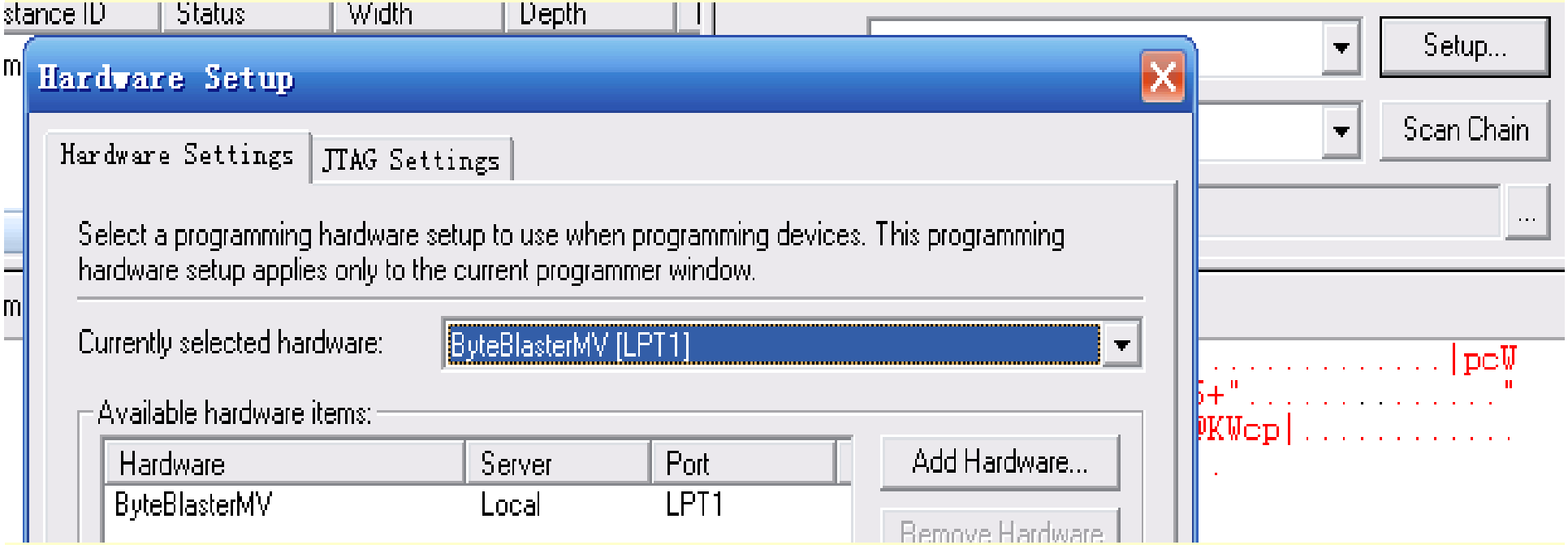


图7-13 In-System Memory Content Editor编辑窗

7.3 在系统存储器数据读写编辑器应用

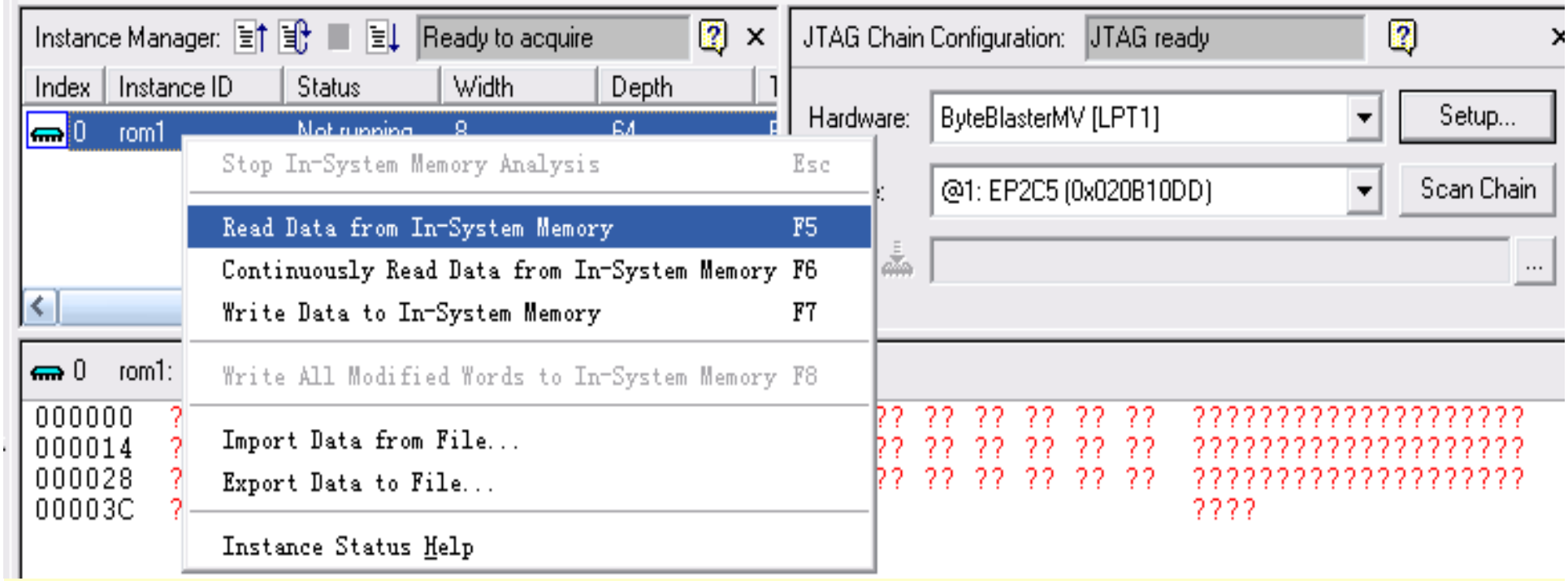


图7-14 与实验系统上的FPGA通信正常情况下的编辑窗界面

7.3 在系统存储器数据读写编辑器应用

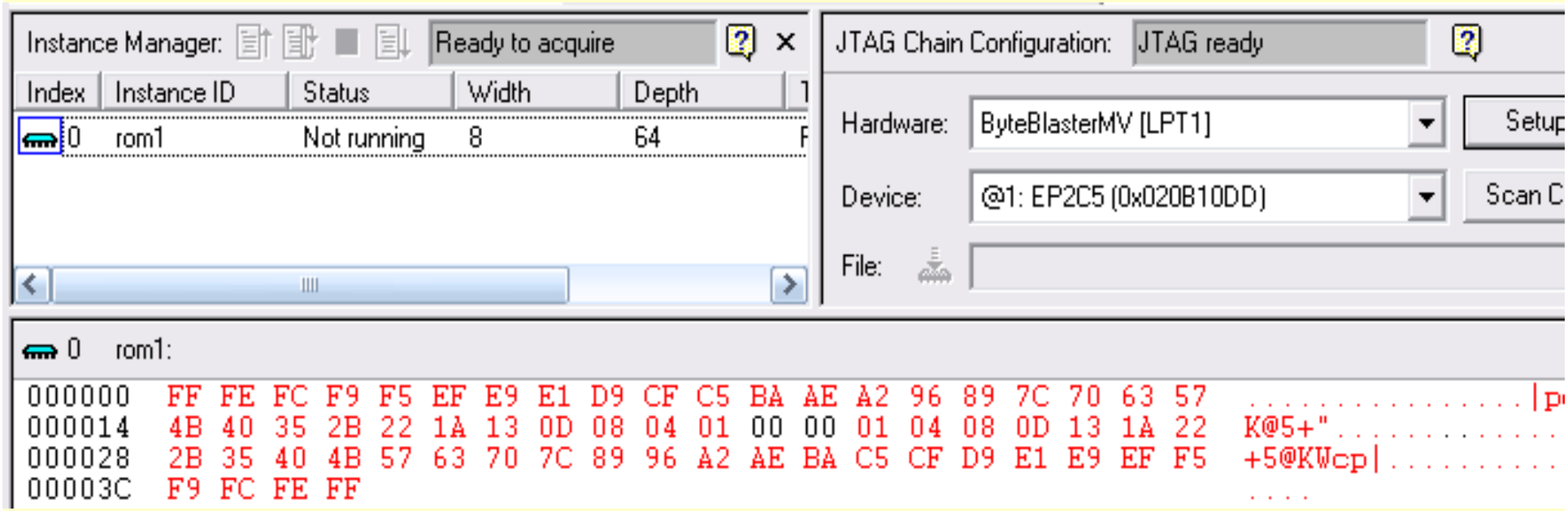


图7-15 从FPGA中的ROM读取波形数据

7.3 在系统存储器数据读写编辑器应用

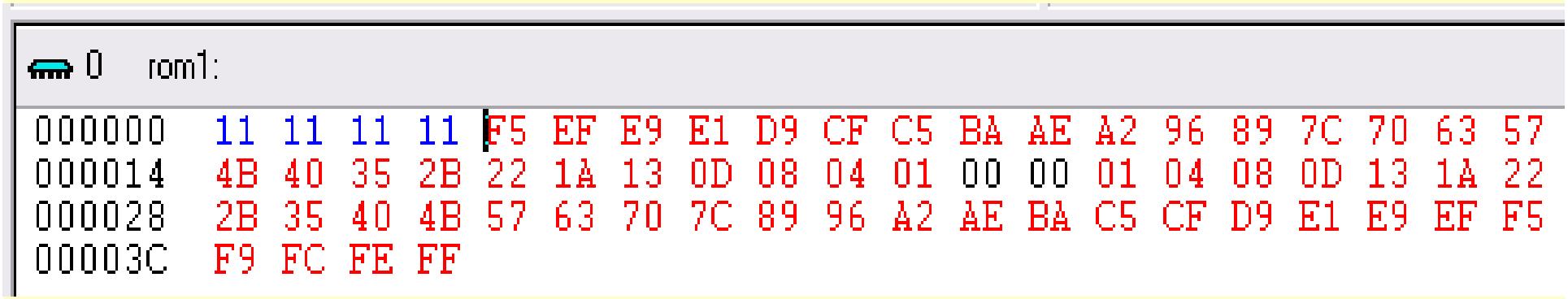


图7-16 编辑波形数据

7.3 在系统存储器数据读写编辑器应用

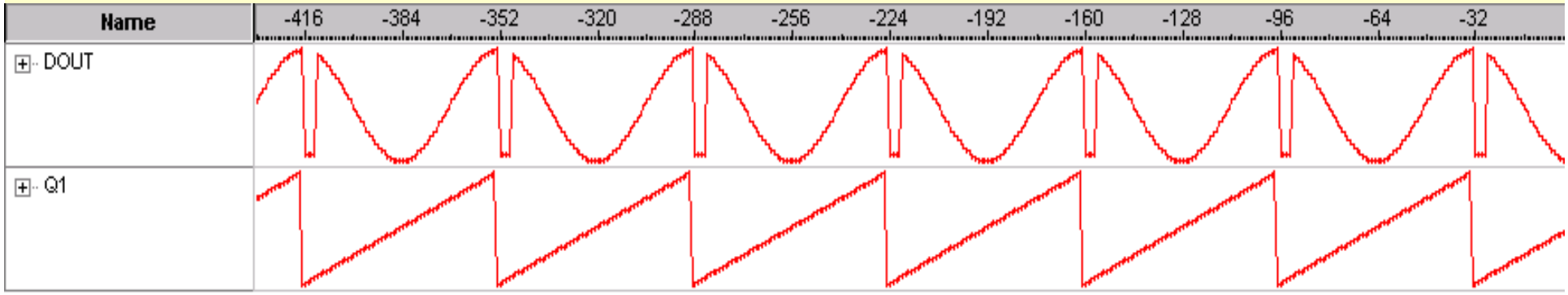


图7-16下载编辑数据后的SignalTap II采样波形

7.4 编辑SignalTapII的触发信号

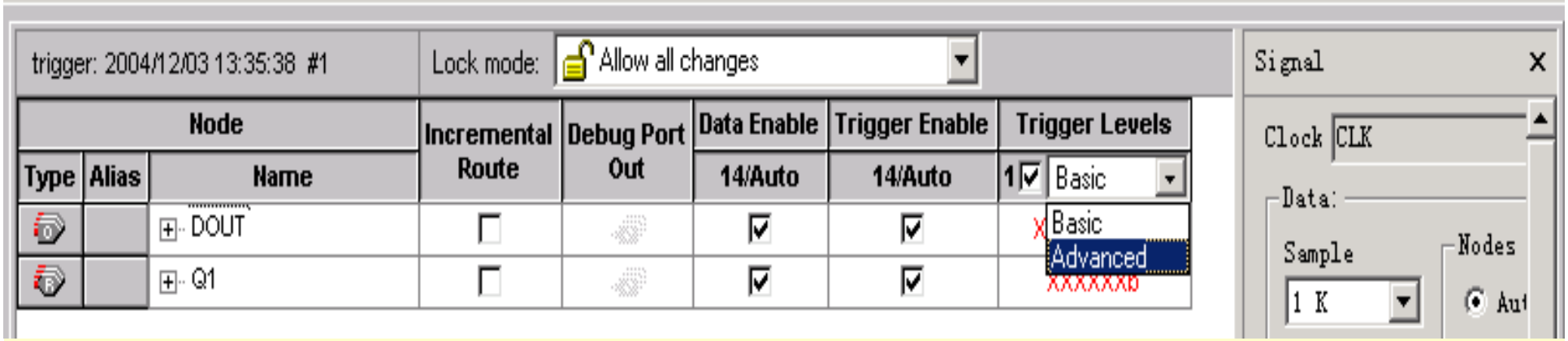


图7-17 选择高级触发条件

7.4 编辑SignalTapII的触发信号

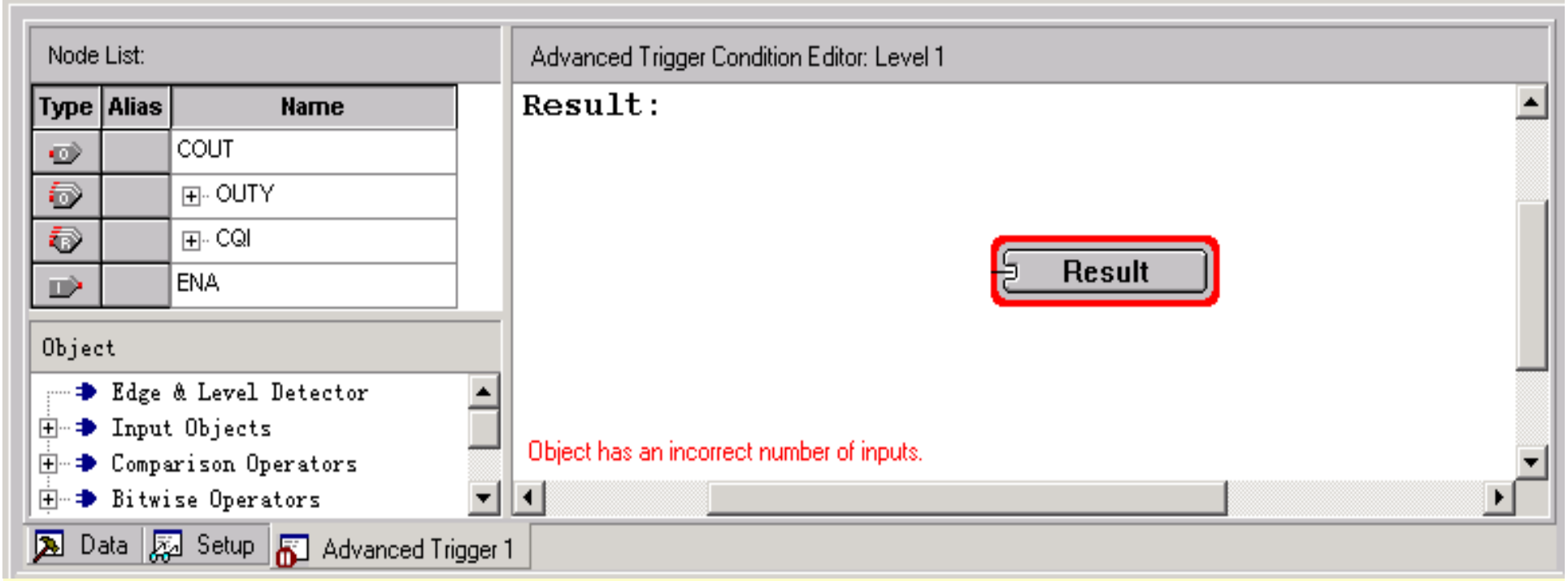


图7-18 进入“触发条件函数编辑”窗口

7.4 编辑SignalTapII的触发信号

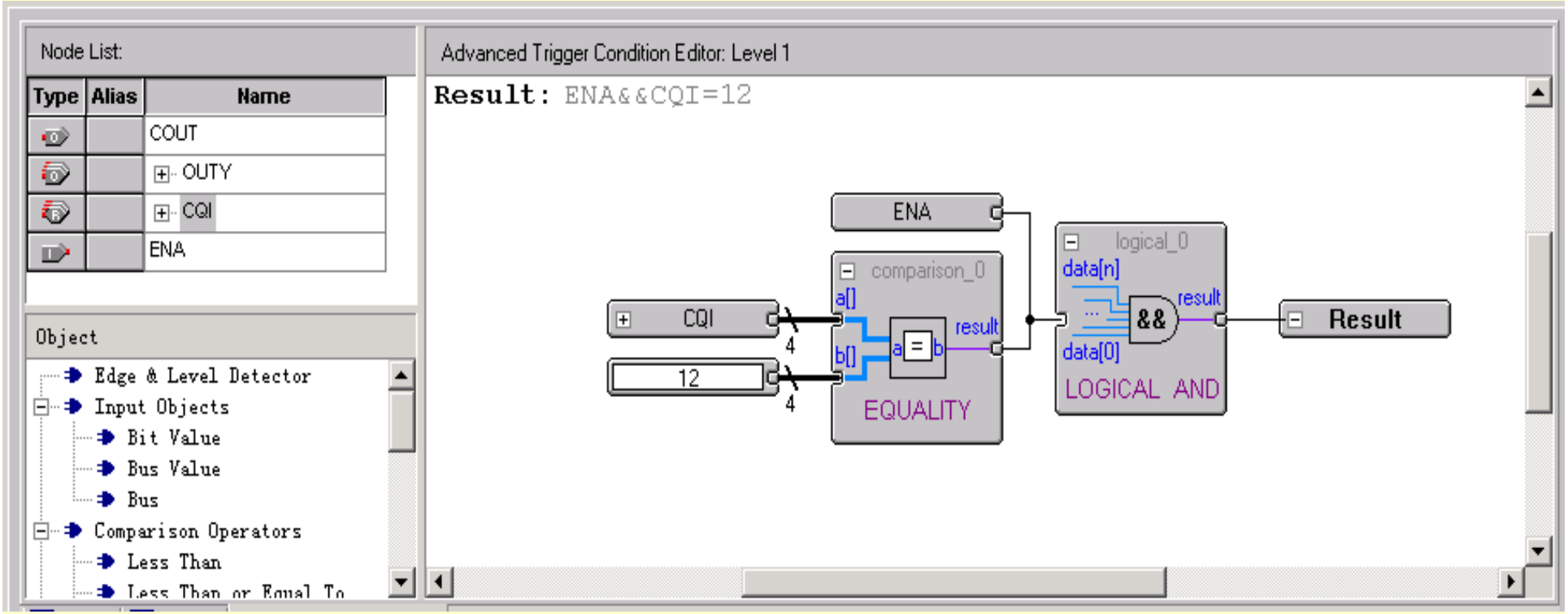


图7-19 编辑触发函数

7.5 其它存储器模块的定制与应用

7.5.1 RAM定制

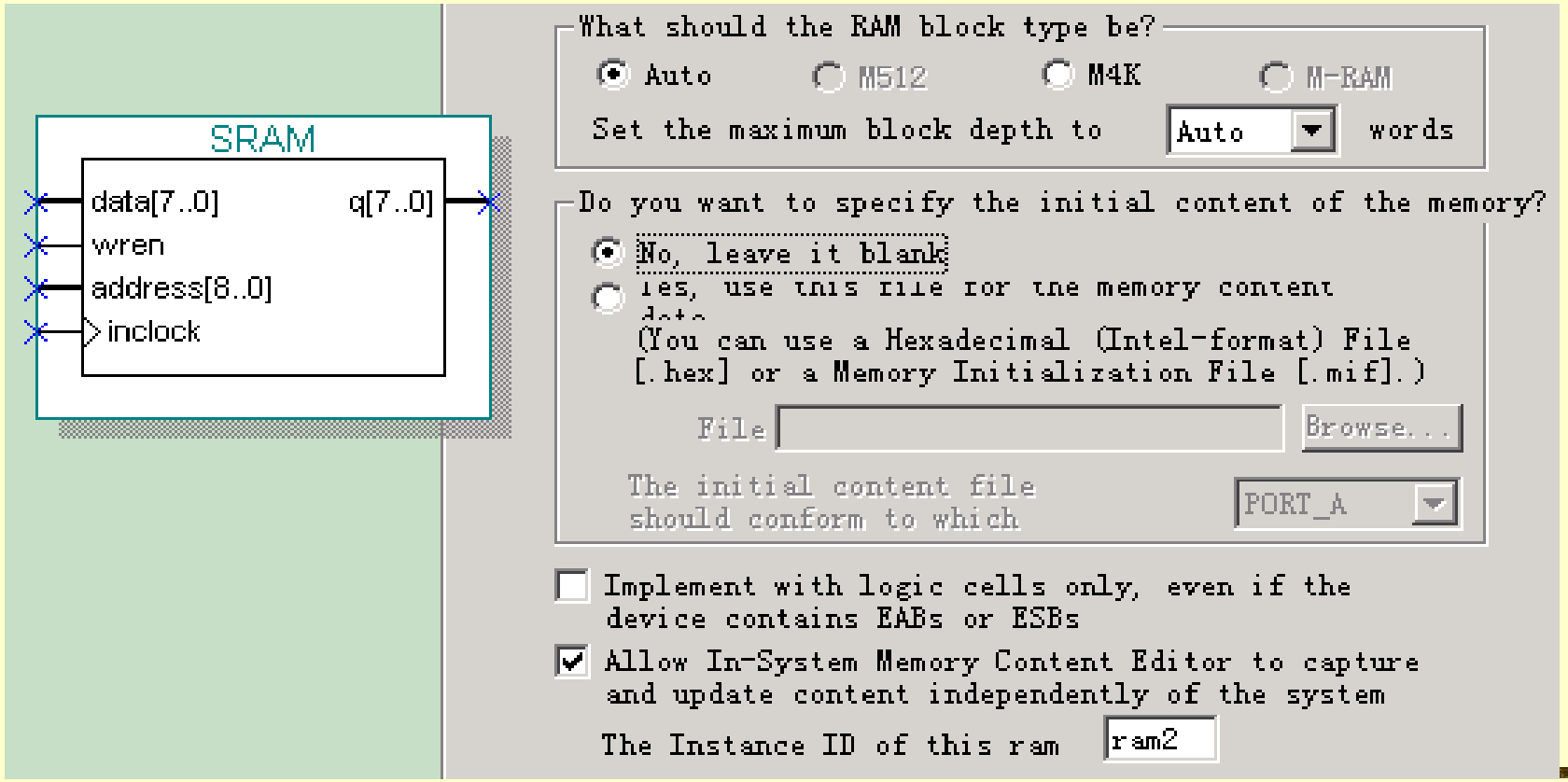


图7-20 编辑定制RAM

7.5 其它存储器模块的定制与应用

7.5.1 RAM定制

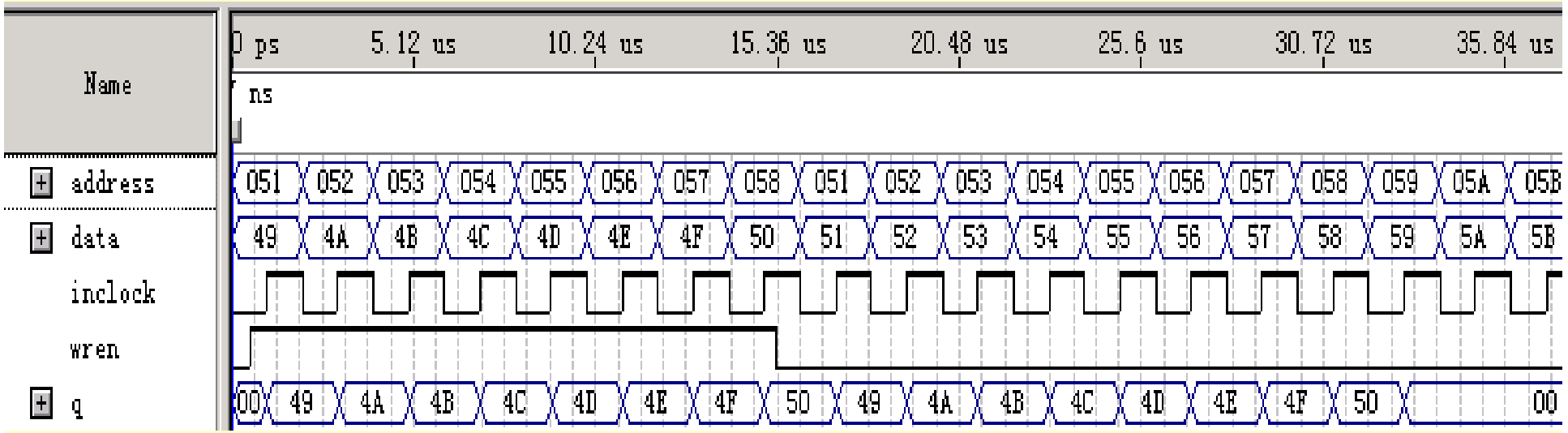


图7-21 LPM_RAM的仿真波形

7.5 其它存储器模块的定制与应用

7.5.2 FIFO定制

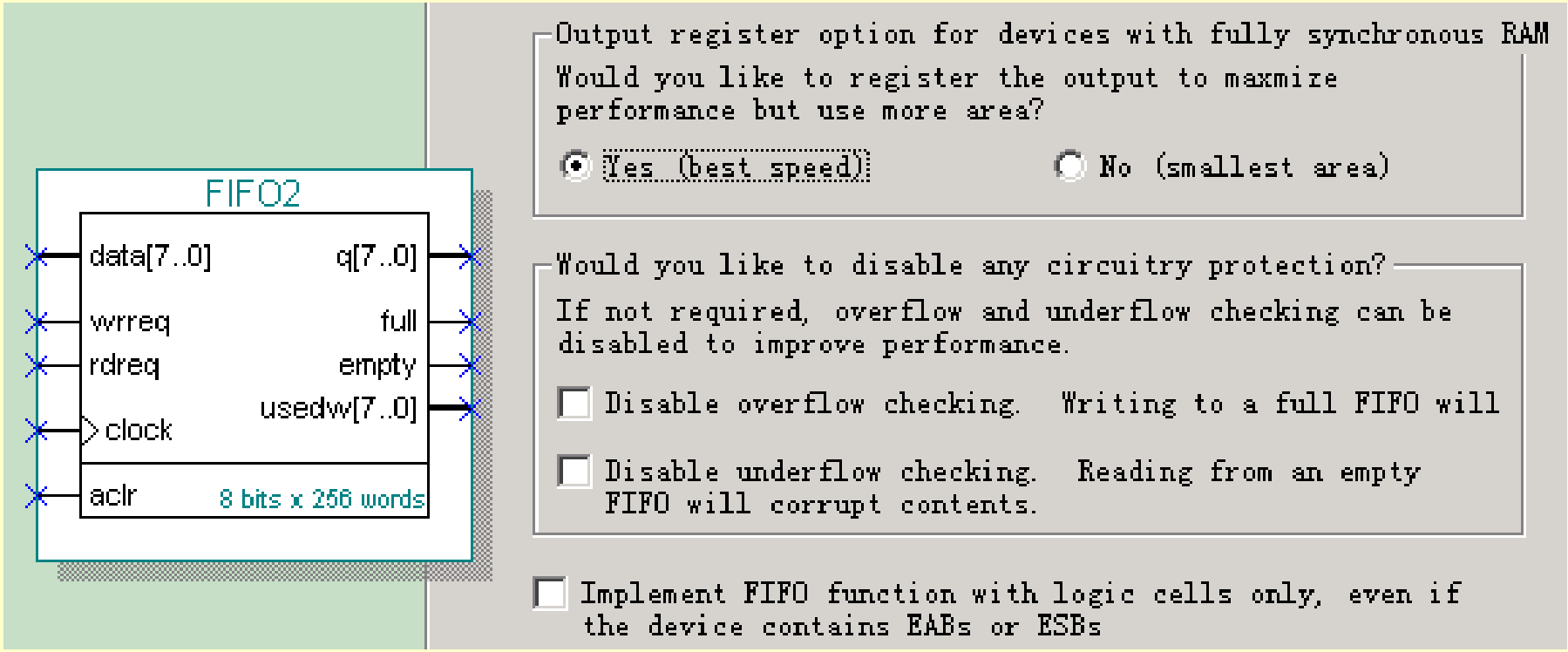


图7-22 FIFO编辑窗

7.5 其它存储器模块的定制与应用

7.5.2 FIFO定制

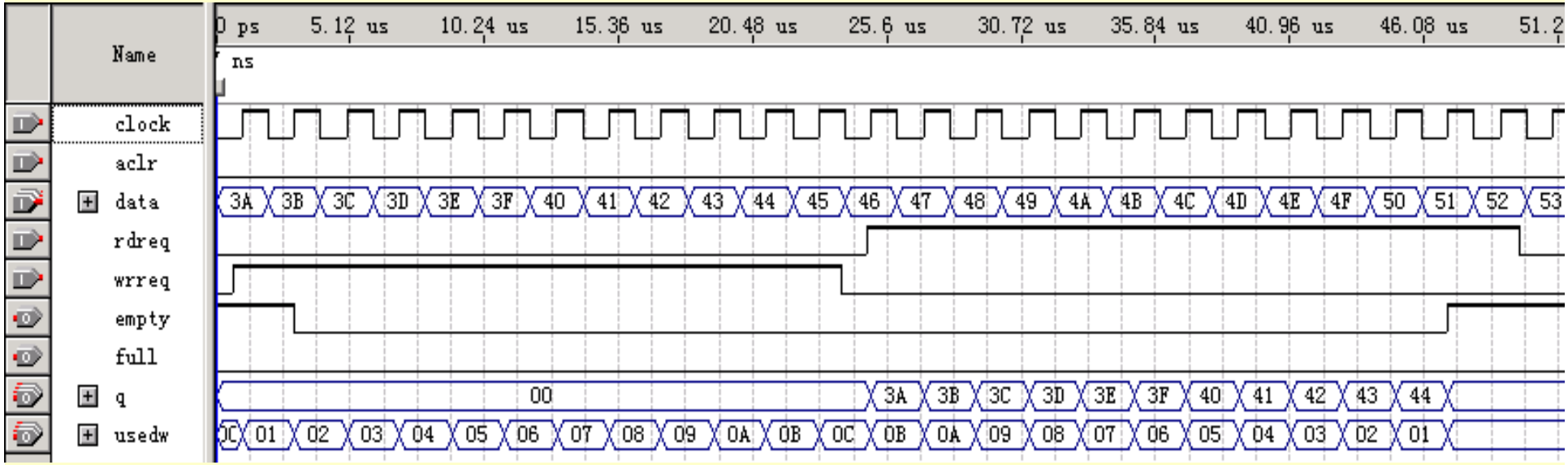


图7-23 FIFO的仿真波形

7.6流水线乘法累加器的混合输入设计

（1）用VHDL设计16位加法器。

【例7-5】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY ADDER16B IS
    PORT ( CIN : IN STD_LOGIC;
             A, B : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
             S : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
          COUT : OUT STD_LOGIC    );
END ADDER16B;
ARCHITECTURE behav OF ADDER16B IS
    SIGNAL SINT : STD_LOGIC_VECTOR(16 DOWNTO 0);
       SIGNAL AA,BB : STD_LOGIC_VECTOR(16 DOWNTO 0);
BEGIN
       AA<='0'&A;     BB<='0'& B;
    SINT <= AA + BB + CIN;   S <= SINT(15 DOWNTO 0);   COUT <= SINT(4);
END behav;
```

7.6流水线乘法累加器的混合输入设计

（2）顶层原理图文件设计。

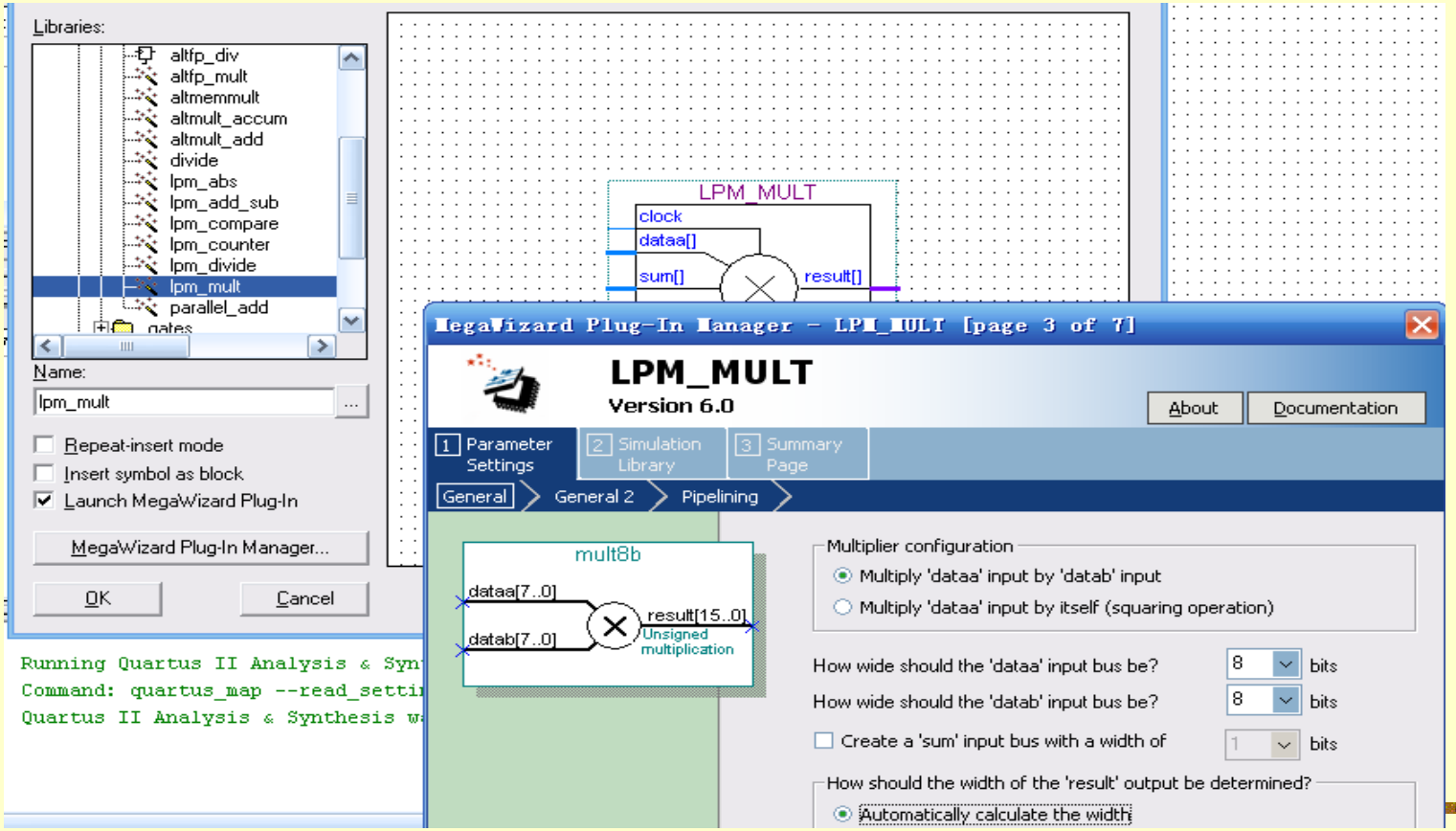


图7-24 在原理图编辑窗加入LPM元件

康芯科技

7.6流水线乘法累加器的混合输入设计

（2）顶层原理图文件设计。

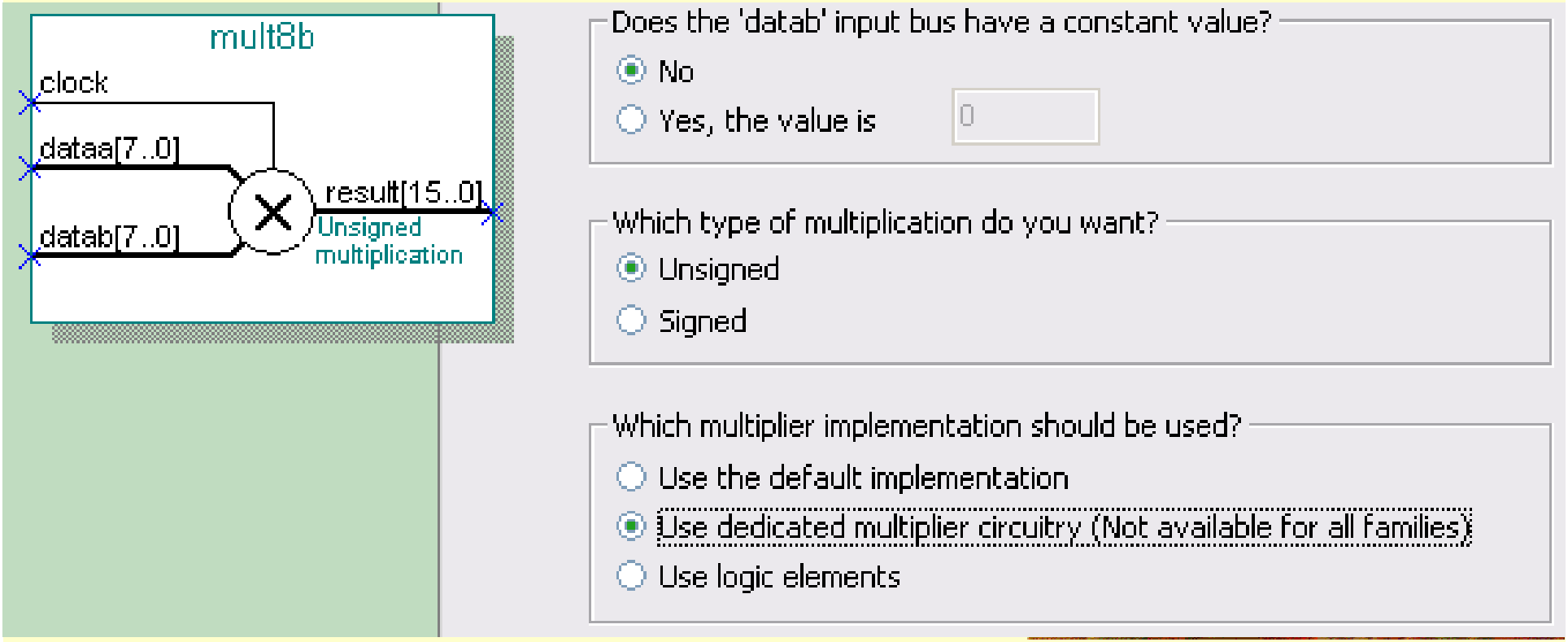


图7-25 将LPM乘法器设置为流水线工作方式

KX 康芯科技

7.6流水线乘法累加器的混合输入设计

（2）顶层原理图文件设计。

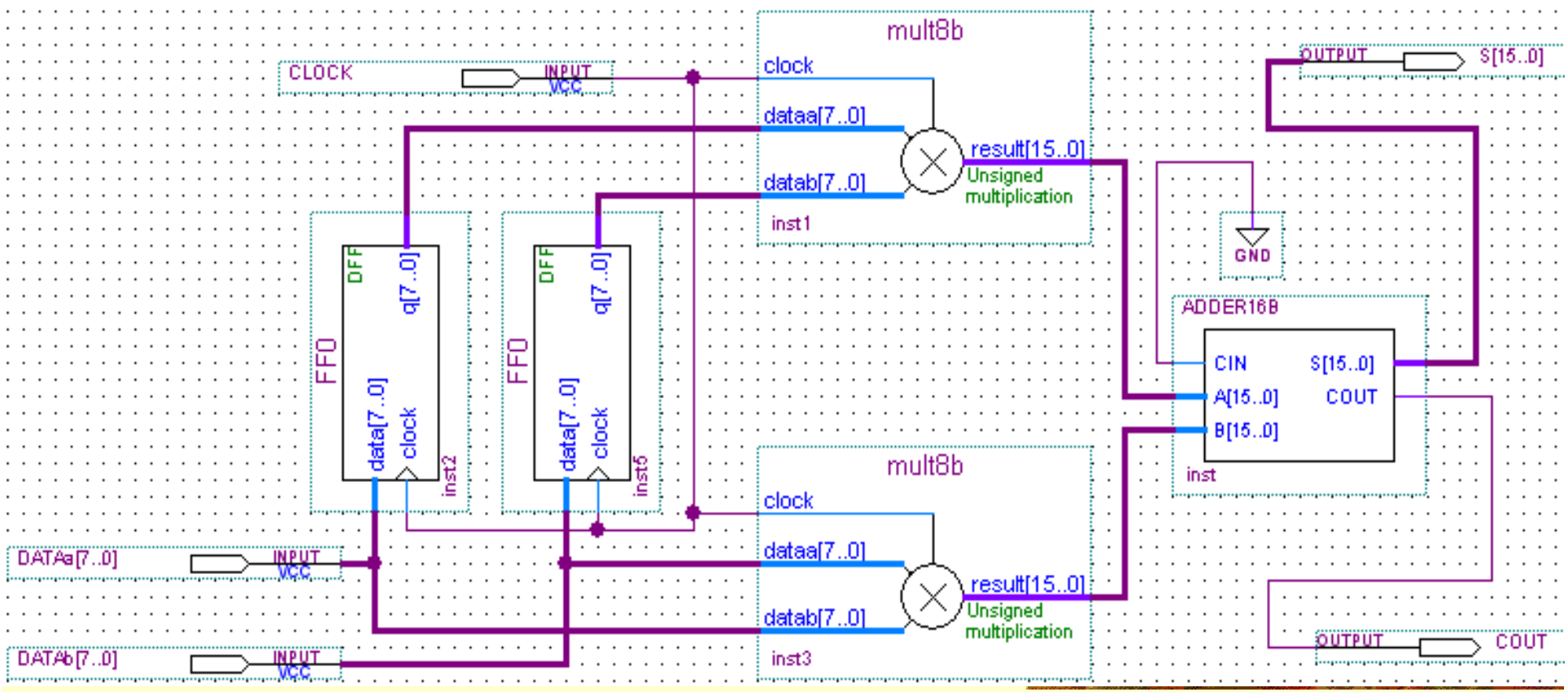


图7-26 乘法累加器电路

康芯科技

7.6流水线乘法累加器的混合输入设计

（3）仿真。

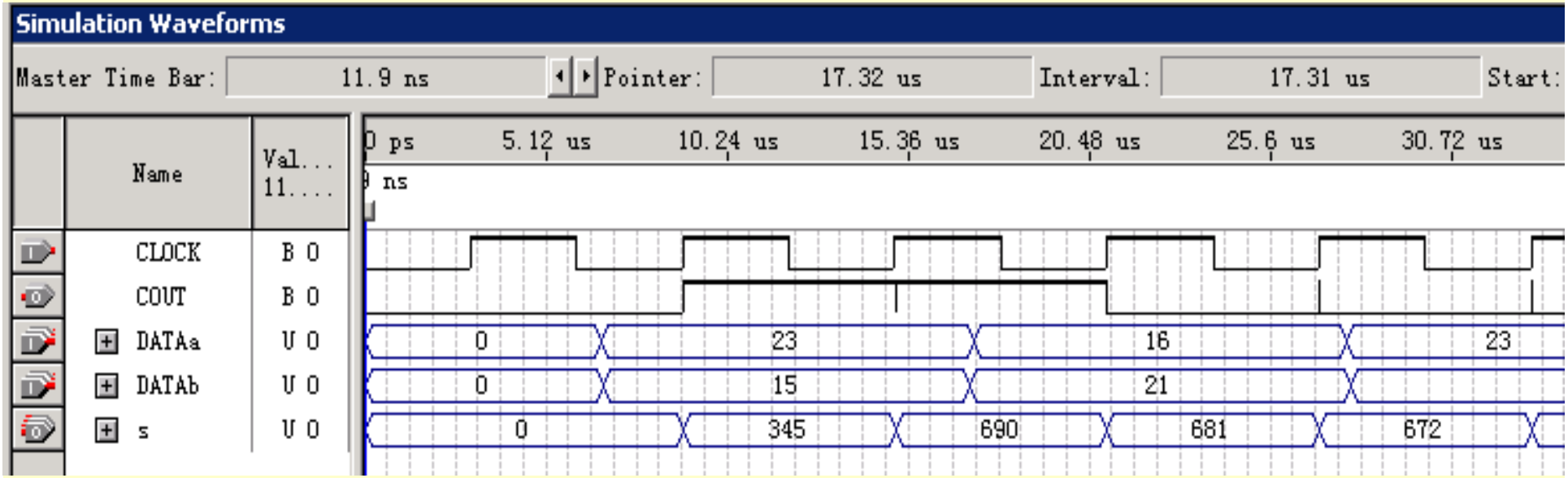


图7-27 muladd工程仿真波形

KX 康芯科技

7.6流水线乘法累加器的混合输入设计

（4）图7-28是对于图7-25在进行不同项目的选择后，编译报告给出的不同资源利用情况。

Total logic elements	224 / 4,608 (5 %
Total registers	110
Total pins	34 / 89 (38 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

Total logic elements	17 / 4,608 (< 1 %)
Total registers	0
Total pins	34 / 89 (38 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	2 / 26 (8 %)
Total PLLs	0 / 2 (0 %)

图7-28 对乘法器选择不同设置后的编译报告

7.7 LPM嵌入式锁相环调用

7.7.1 建立嵌入式锁相环元件

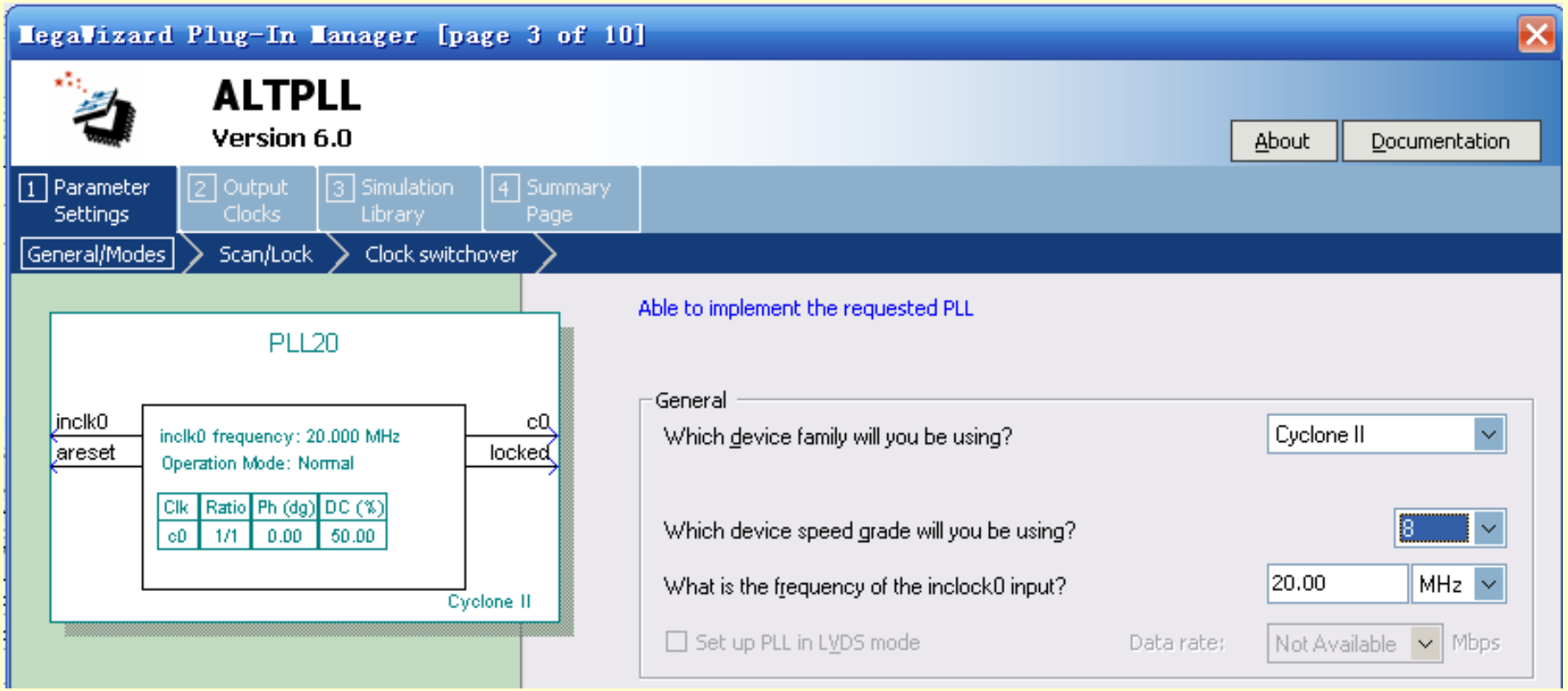


图7-29 选择参考时钟为20MHz

KX 康芯科技

7.7 LPM嵌入式锁相环调用

7.7.1 建立嵌入式锁相环元件

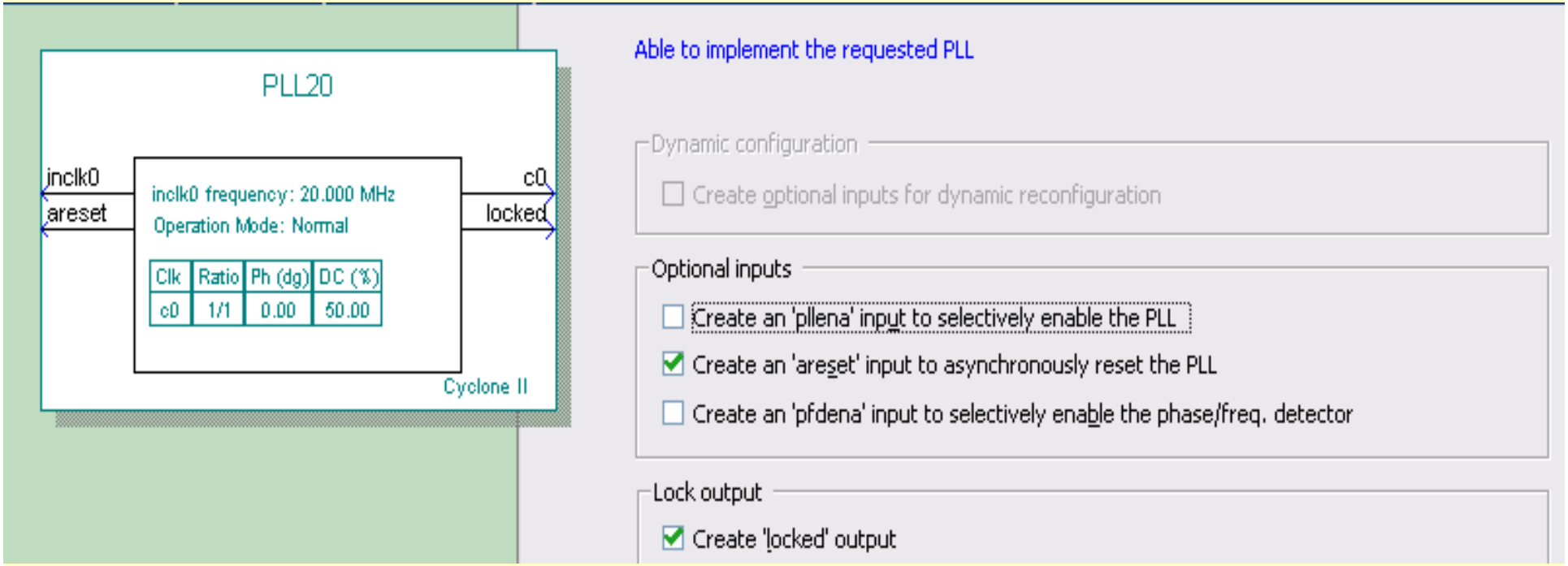


图7-30 选择控制信号

7.7 LPM嵌入式锁相环调用

7.7.1 建立嵌入式锁相环元件

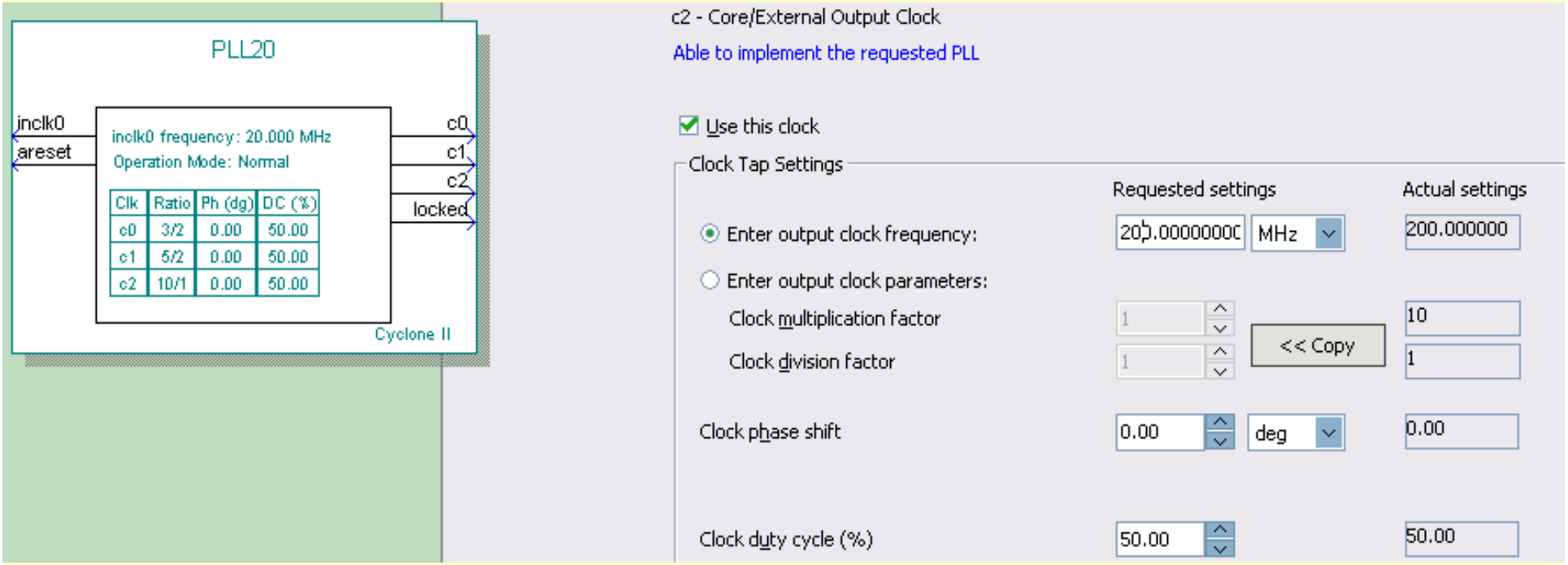


图7-31 选择e0的输出频率为210MHz

KX 康芯科技

7.7 LPM嵌入式锁相环调用

7.7.2 测试锁相环

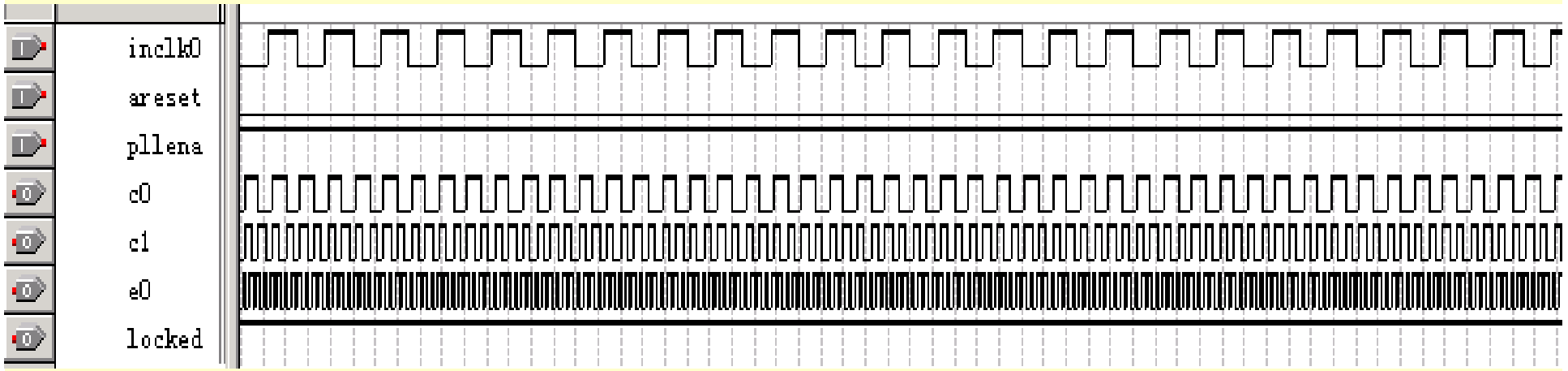


图7-32 PLL元件的仿真波形

7.7.2 测试锁相环

单频率输出的应用PLL的示例：

```
...;
ENTITY DDS_VHDL IS
    PORT (     CLKK : IN  STD_LOGIC;  --此时钟进入锁相环
               FWORD : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
...;
ARCHITECTURE one OF DDS_VHDL IS
   COMPONENT PLLU                              --调入PLL声明
        PORT (    inclk0 : IN STD_LOGIC  := '0';
                    c0 : OUT STD_LOGIC );
    END COMPONENT;
     COMPONENT REG32B
...;
BEGIN
...;
  u6 :  SIN_ROM PORT MAP( address=>D32B(31 DOWNTO 22), q=>POUT,
inclock=>CLK );
  u7 :    PLL20  PORT MAP( inclk0=> CLKK,c0=>CLK);  --例化
END;
```

7.8 IP核NCO数控振荡器使用方法

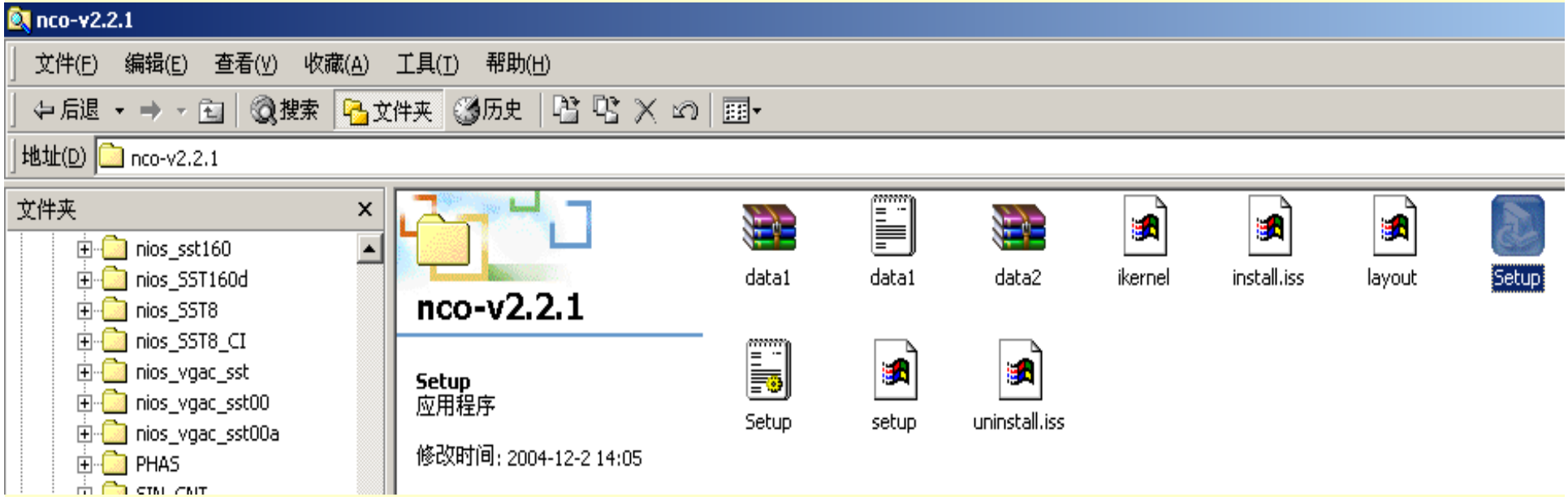


图7-33 安装NCO核

7.8 IP核NCO数控振荡器使用方法

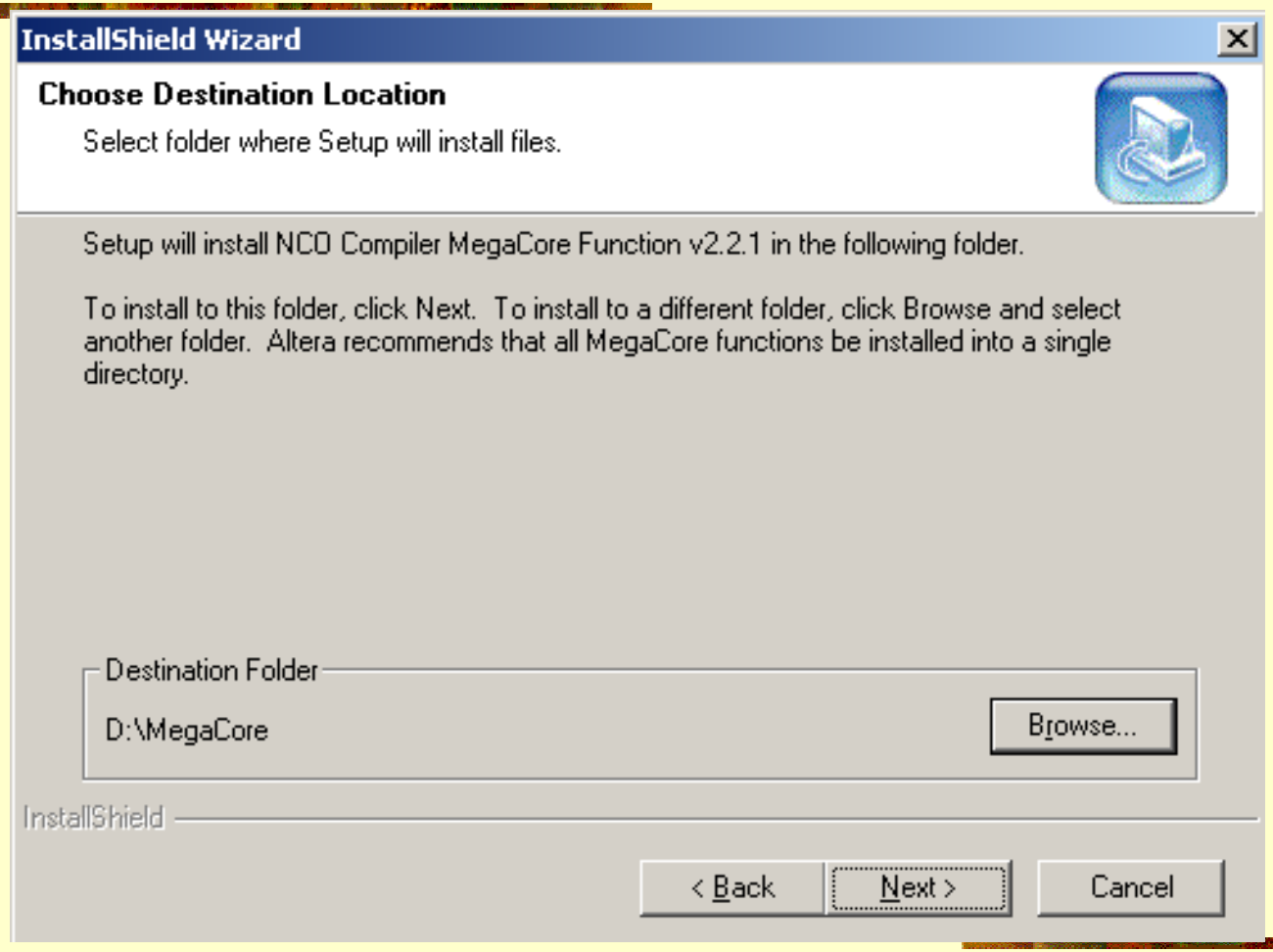


图7-34 确定安装路径

7.8 IP核NCO数控振荡器使用方法

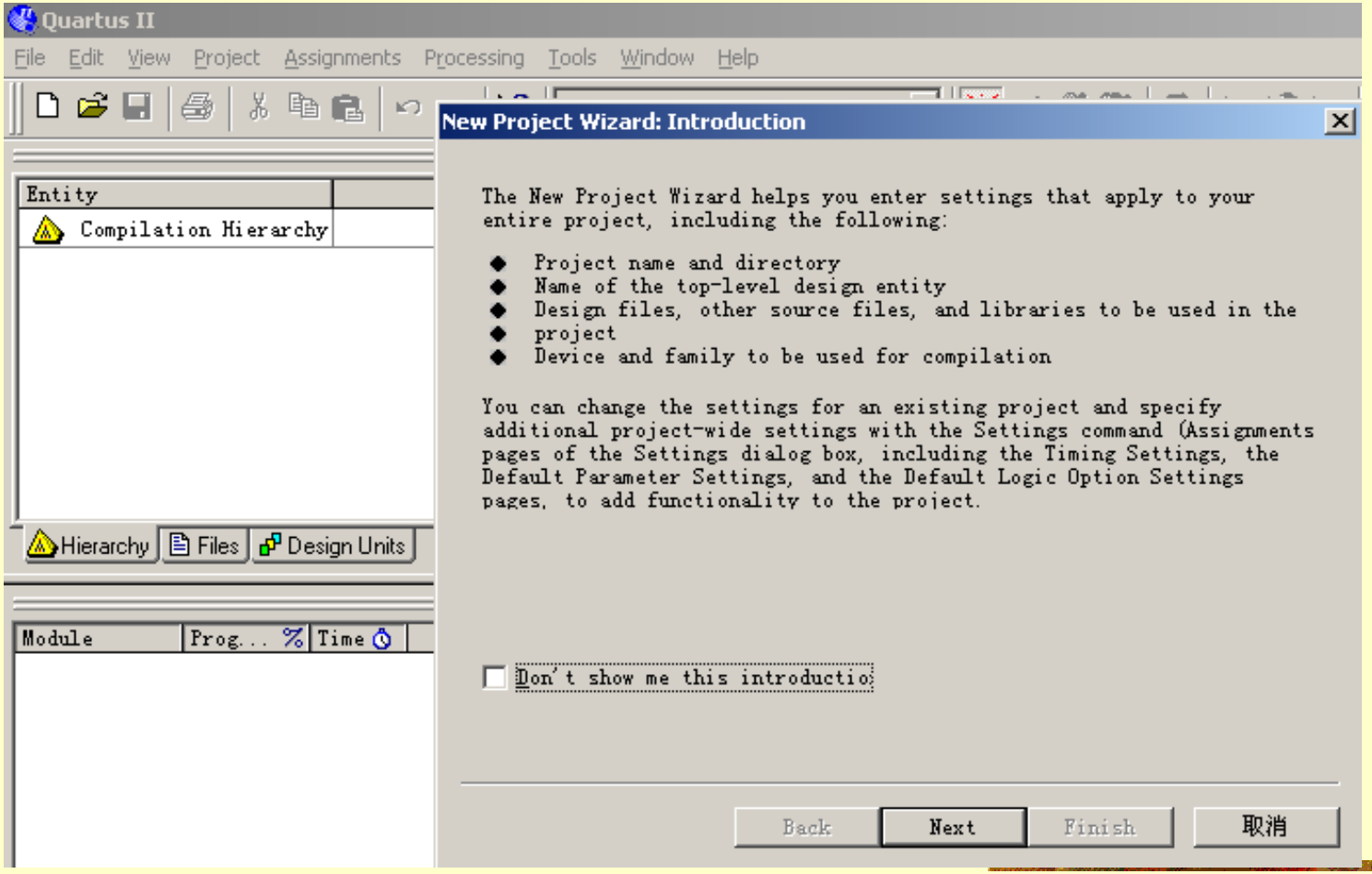


图7-35 开始Core的工程路径

7.8 IP核NCO数控振荡器使用方法

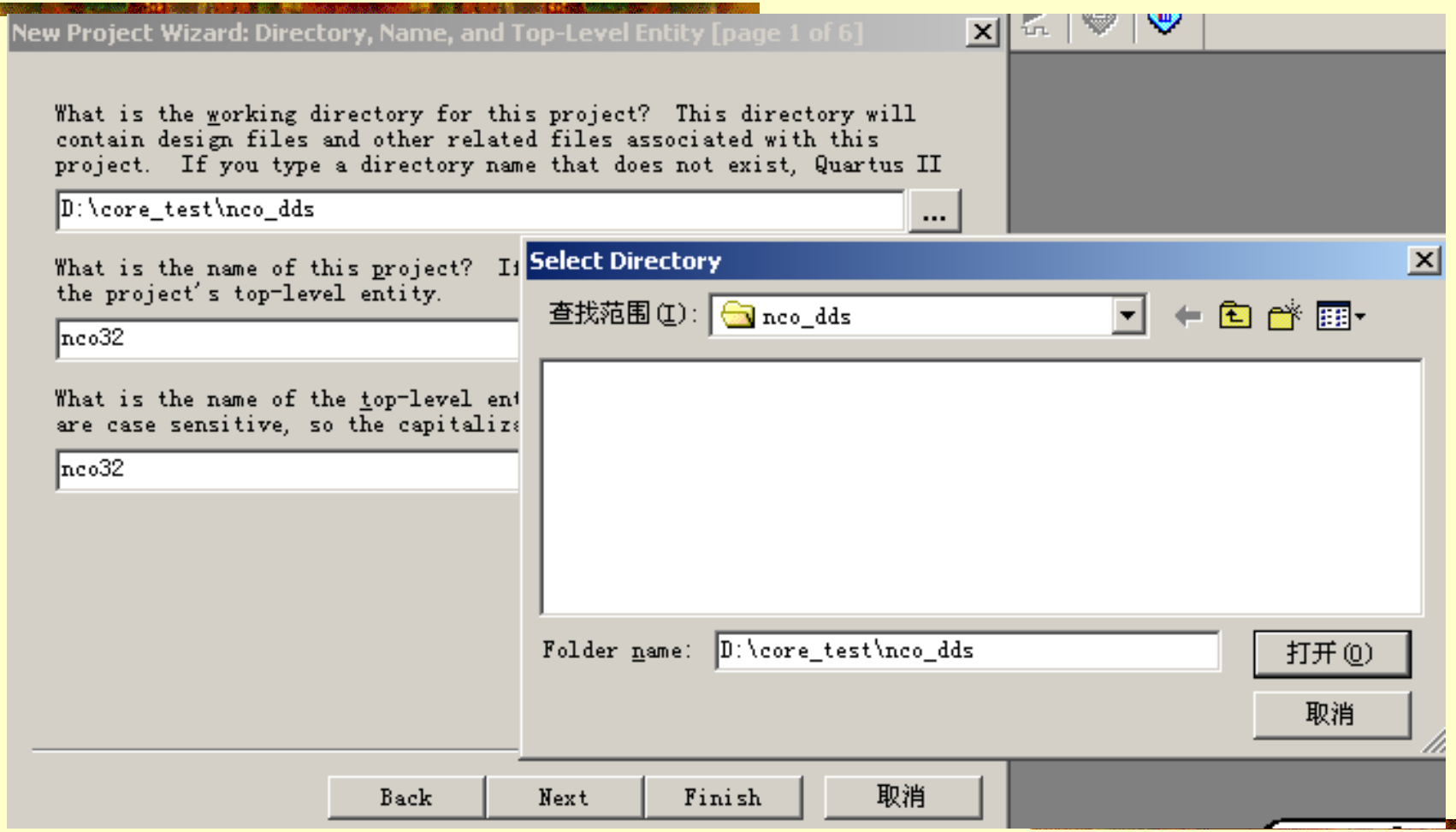


图7-36 确定工程路径和工程名

7.8 IP核NCO数控振荡器使用方法

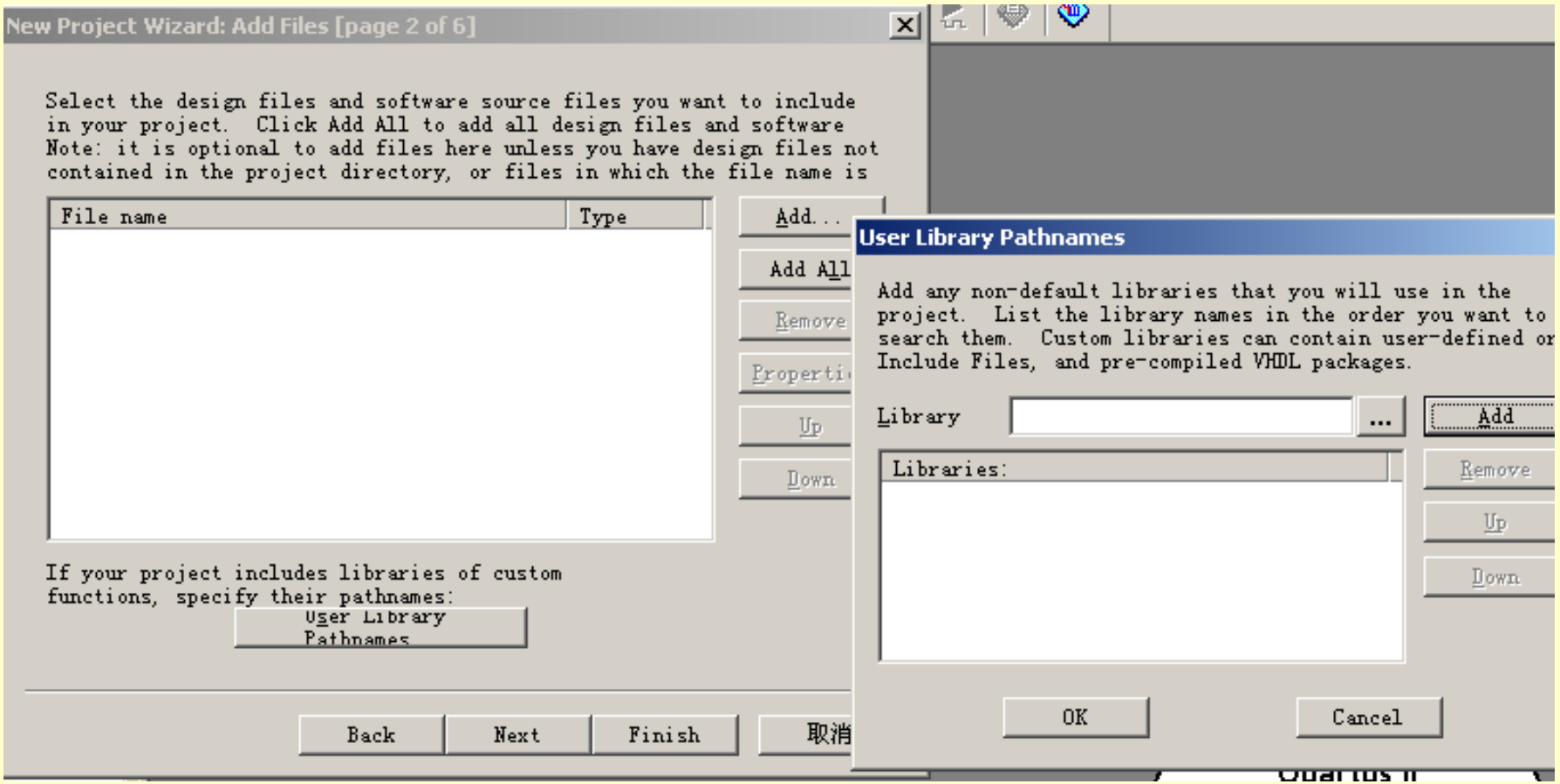


图7-37 打开Core用户库设置窗

7.8 IP核NCO数控振荡器使用方法

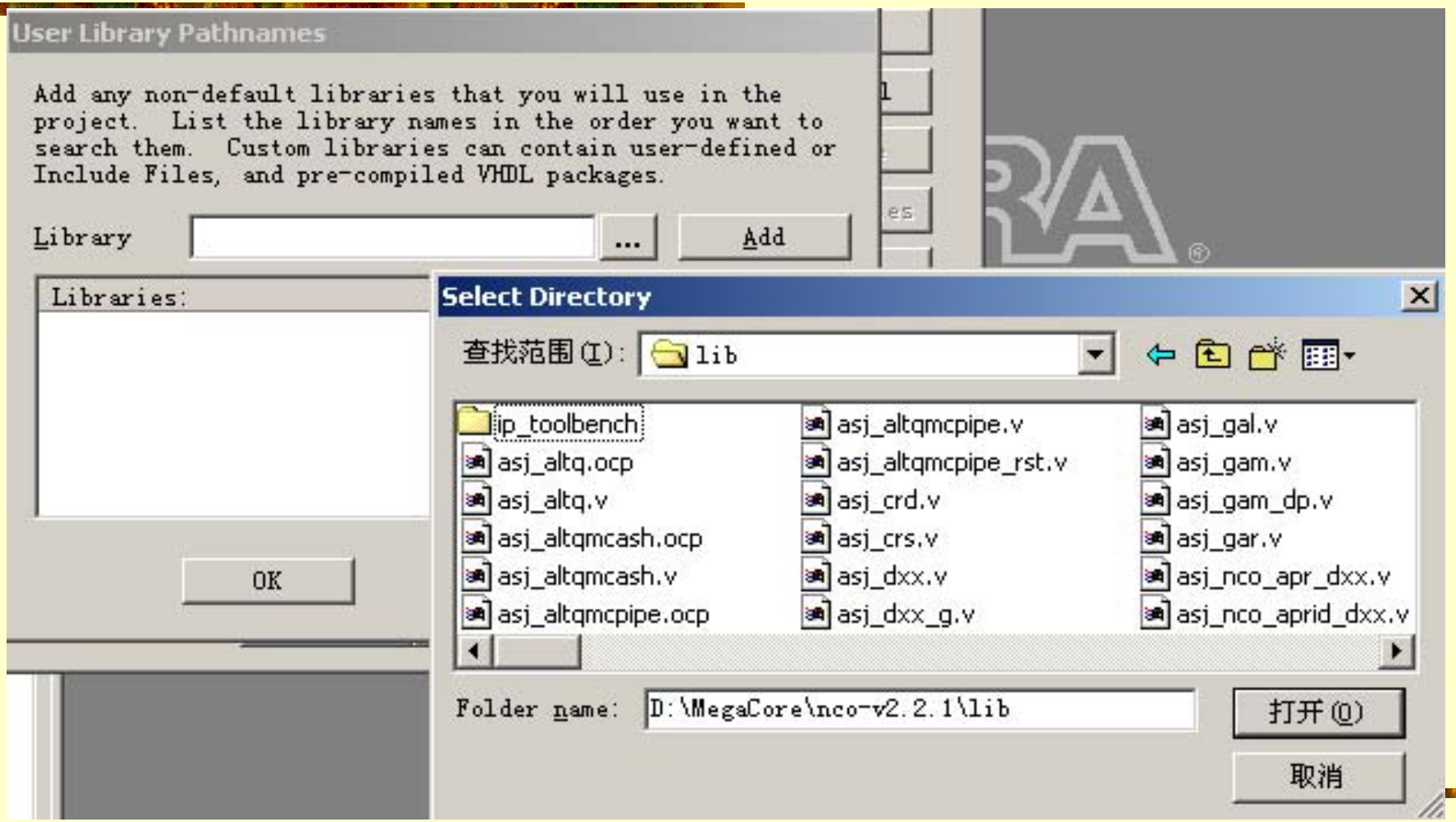


图7-38 选中确定路径上的NCO库

7.8 IP核NCO数控振荡器使用方法

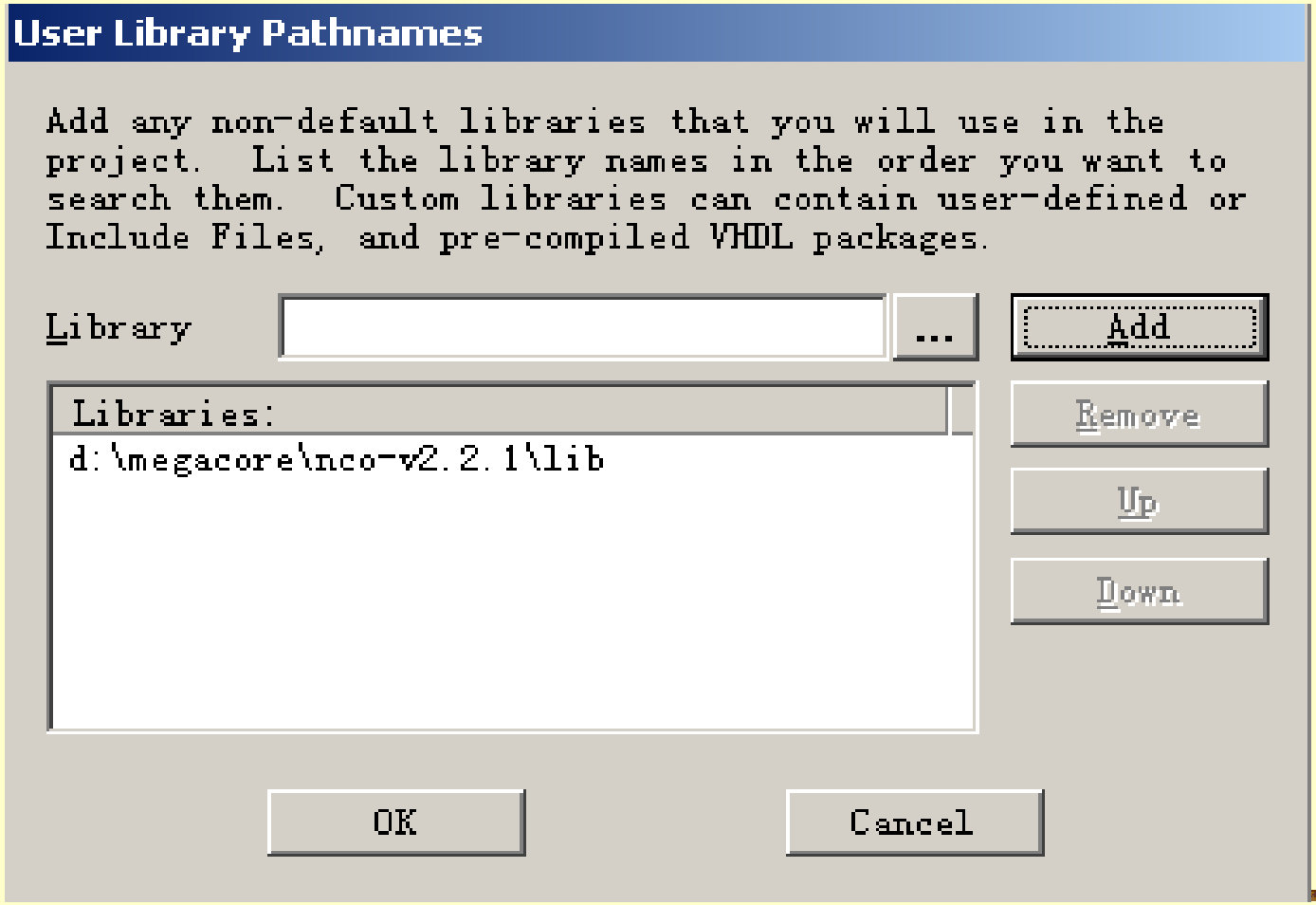


图7-39 加入NCO库

7.8 IP核NCO数控振荡器使用方法

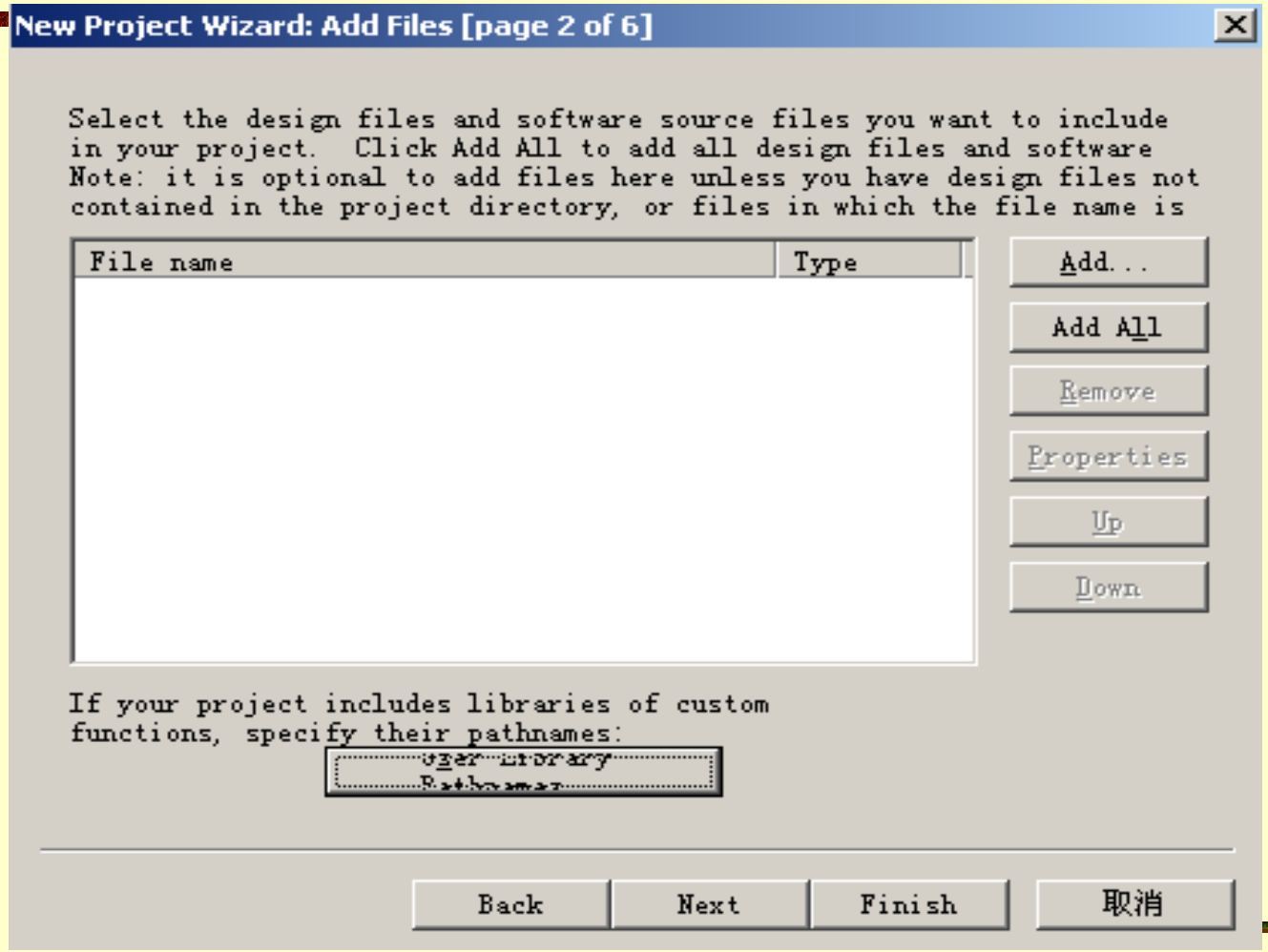


图7-40 已经在工程中加入NCO库

7.8 IP核NCO数控振荡器使用方法

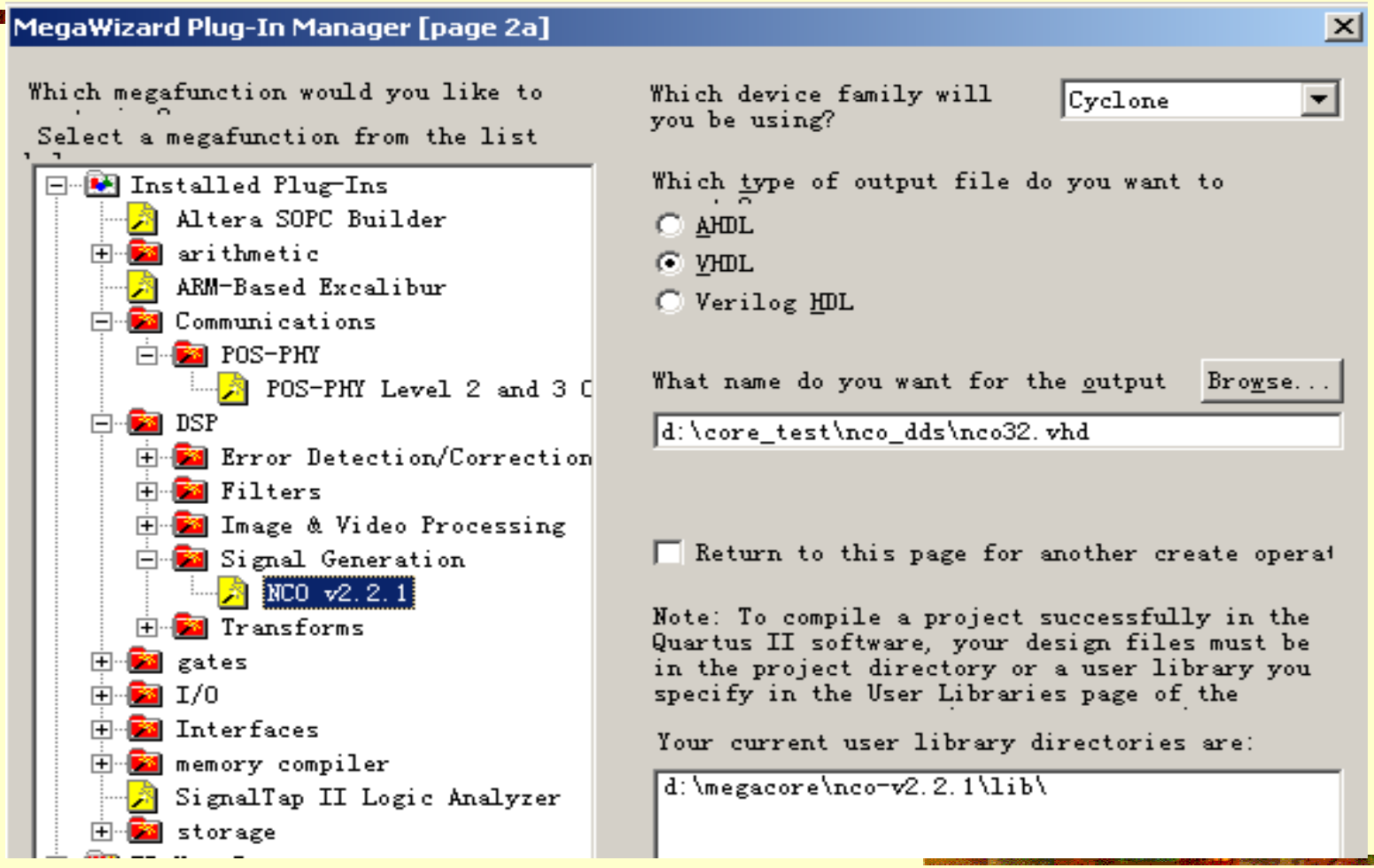


图7-41 打开Core设置管理窗

7.8 IP核NCO数控振荡器使用方法

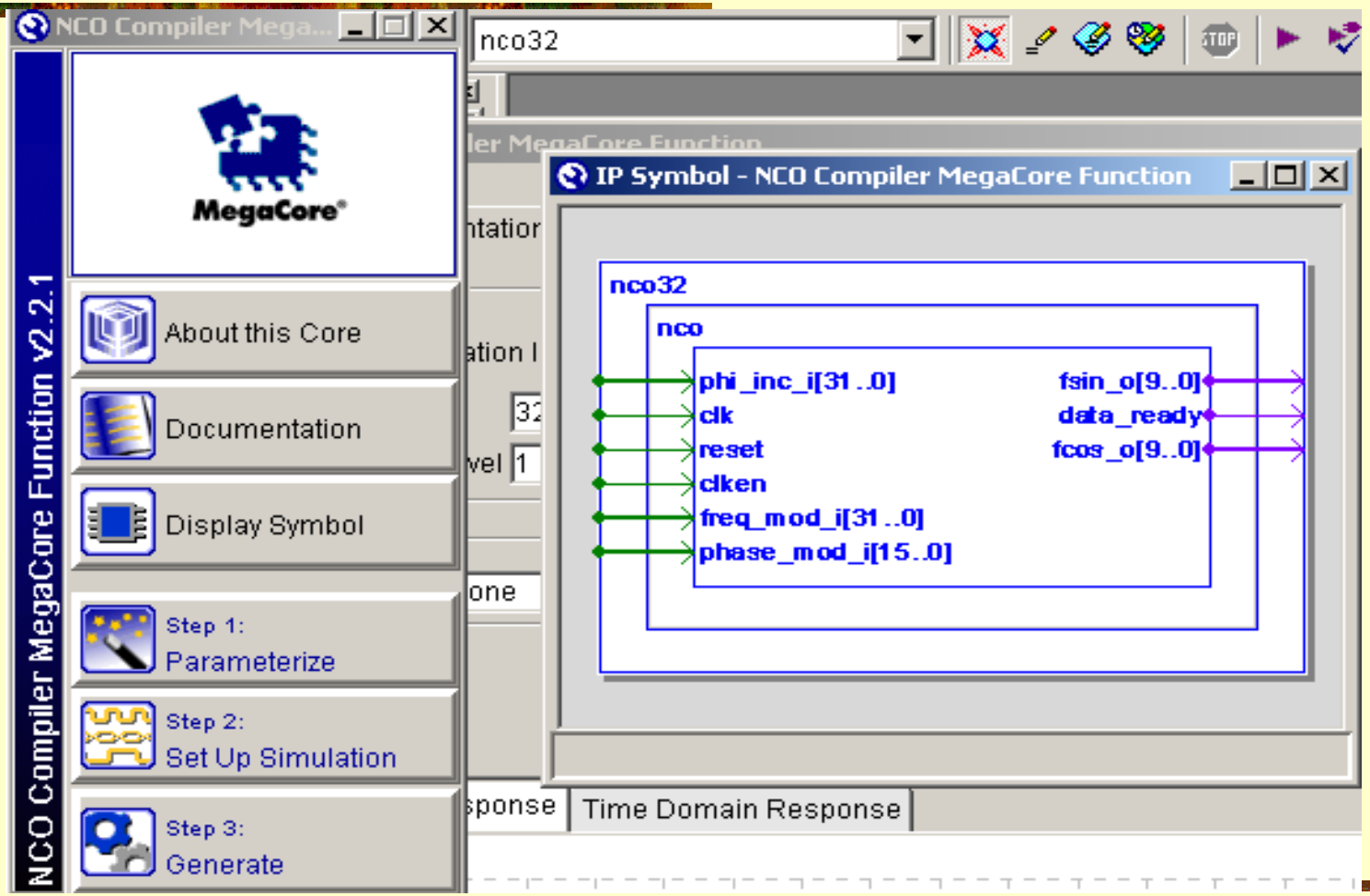


图7-42 开始进入Core参数设置窗Toolbench

康芯科技

7.8 IP核NCO数控振荡器使用方法

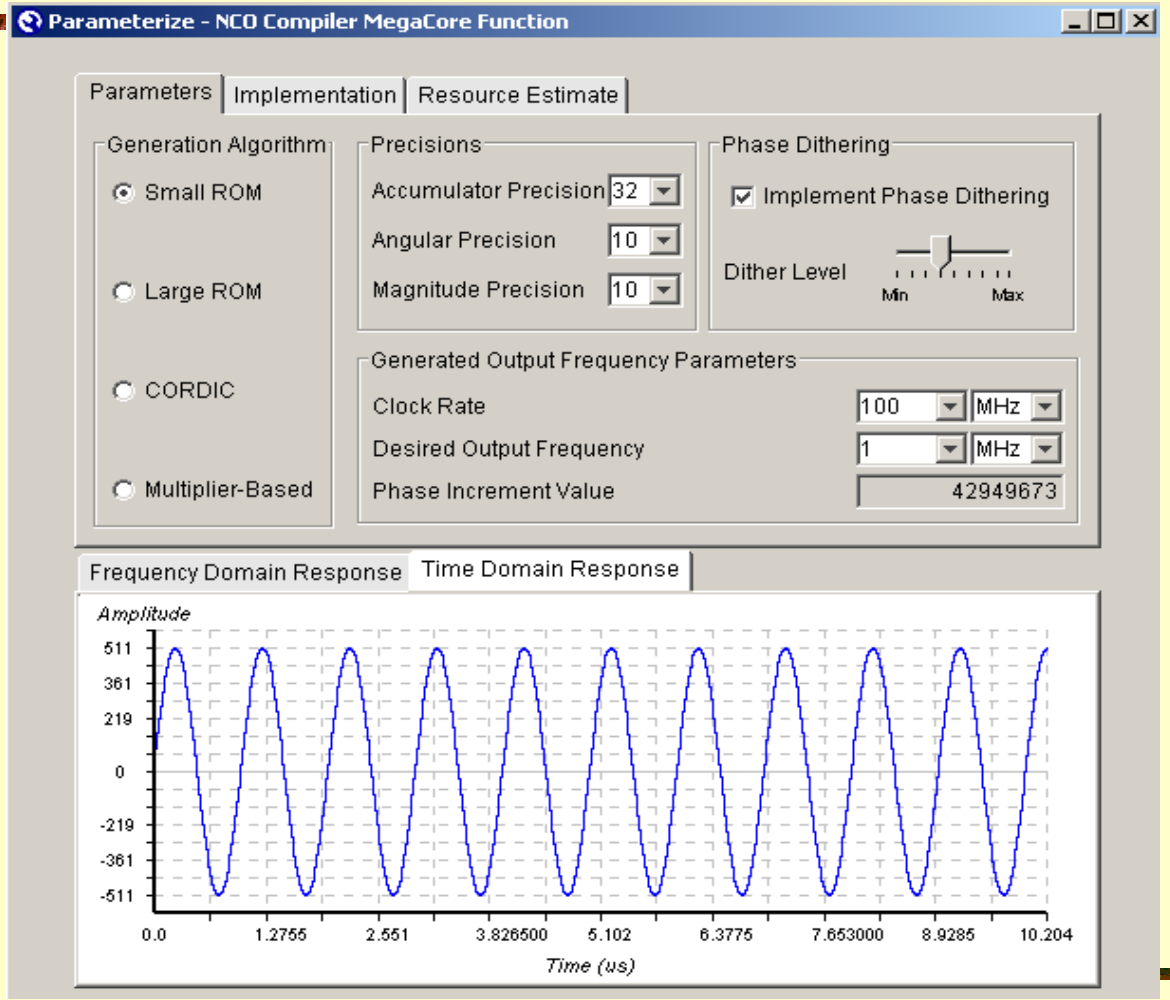


图7-43 设置NCO参数

7.8 IP核NCO数控振荡器使用方法

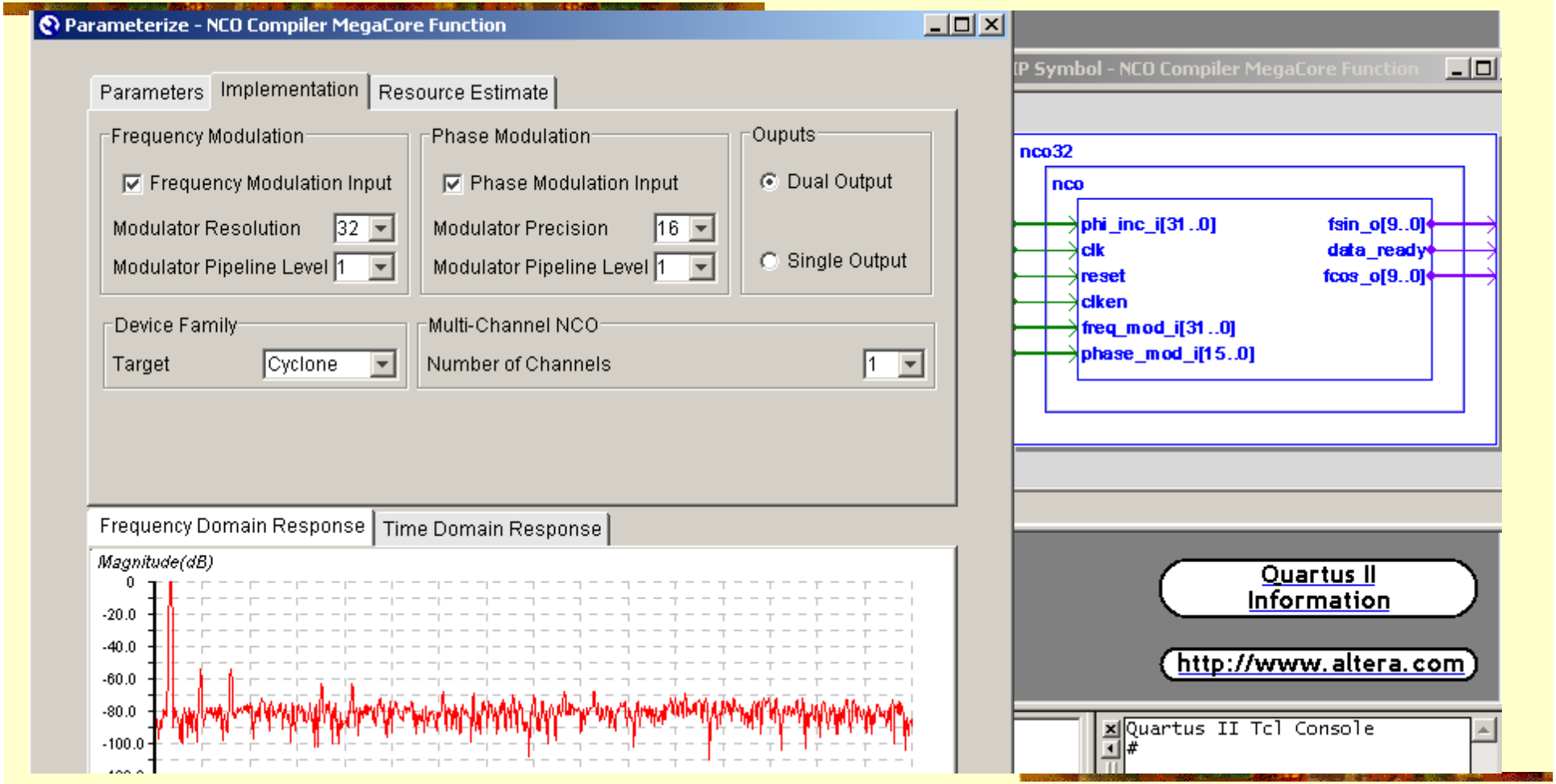


图7-44设置NCO参数

7.8 IP核NCO数控振荡器使用方法

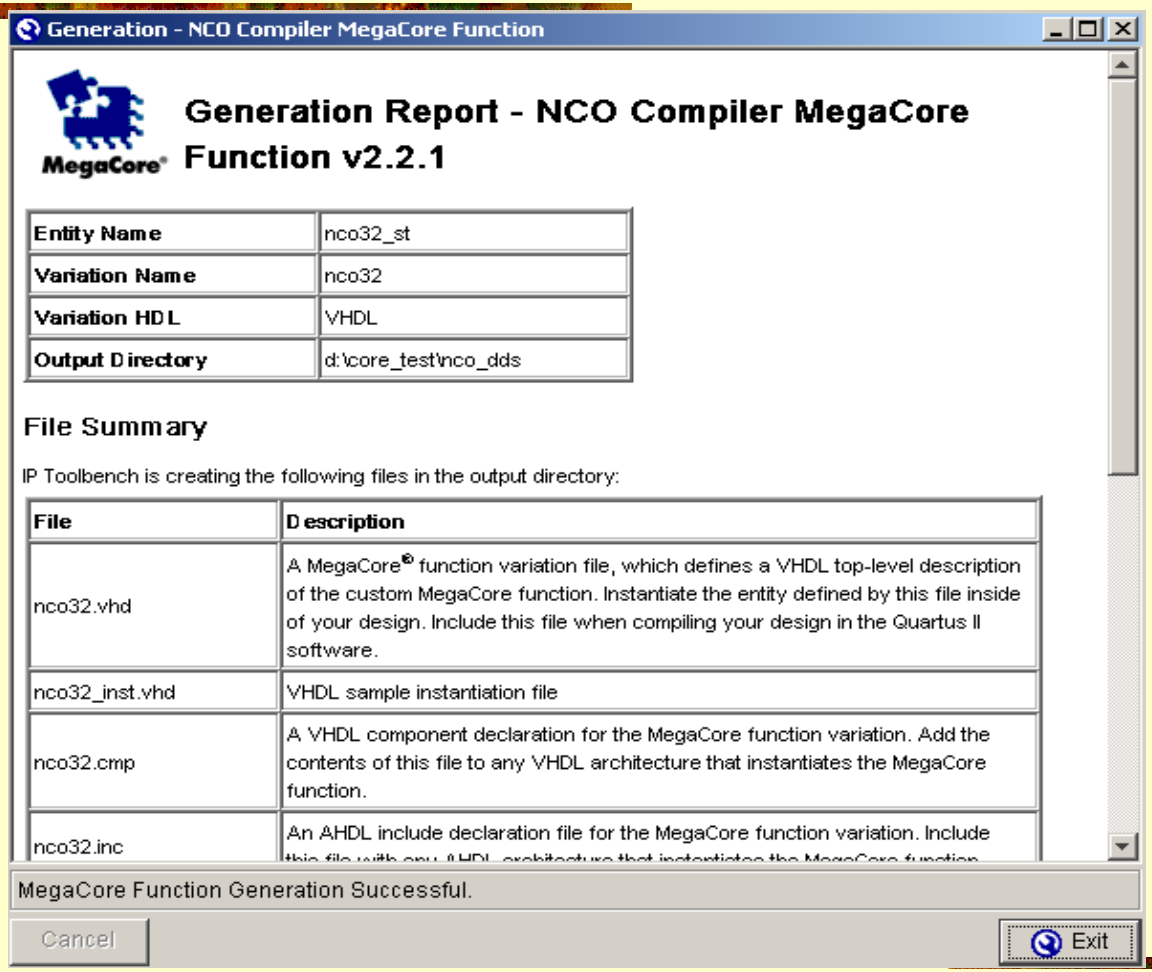


图7-45完成NCO参数设置并生成设计文件后的信息窗

KX 康芯科技

7.8 IP核NCO数控振荡器使用方法

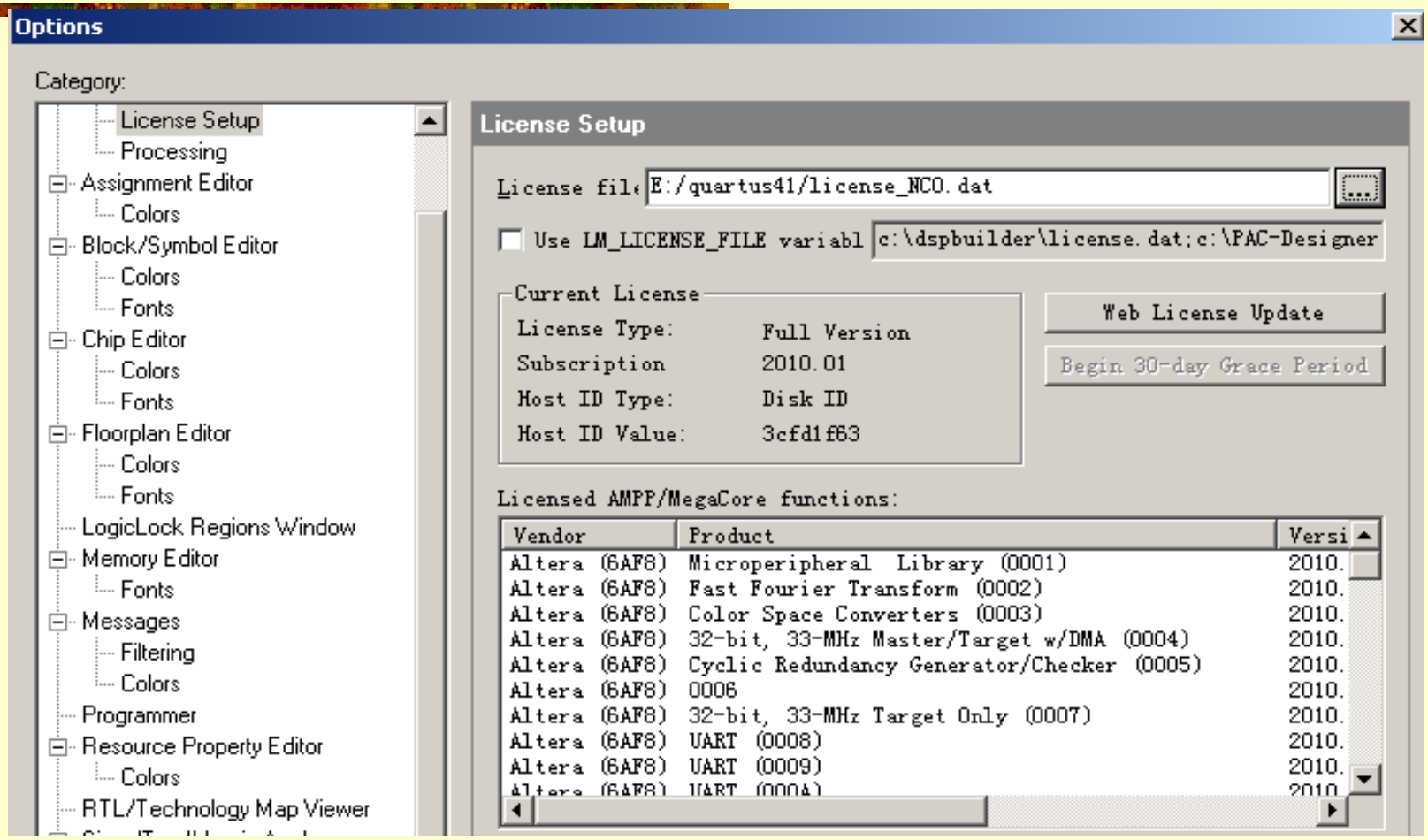


图7-46 加入NCO的授权文件

7.8 IP核NCO数控振荡器使用方法

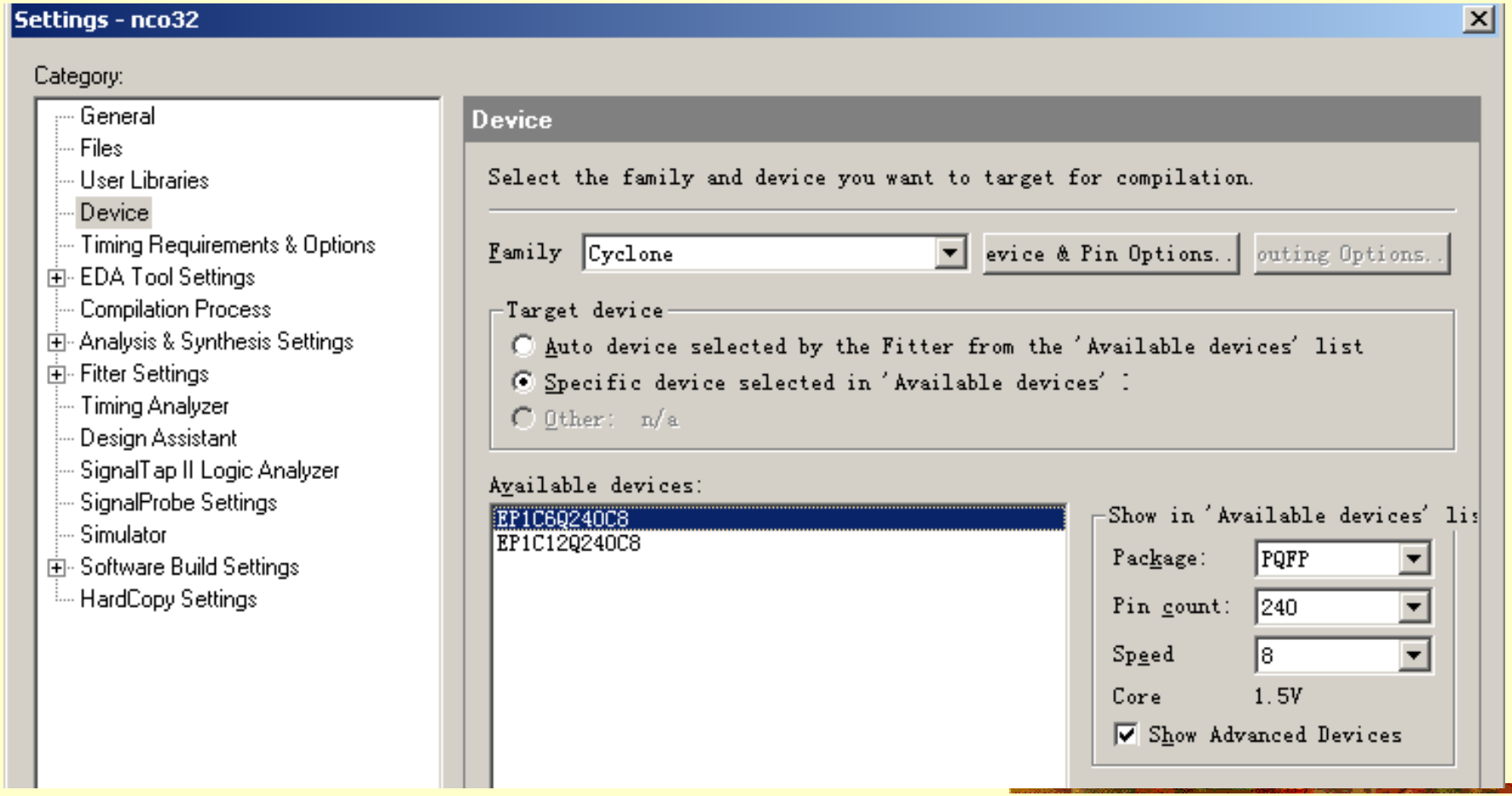


图7-47 选定FPGA目标器件

KX 康芯科技

7.8 IP核NCO数控振荡器使用方法

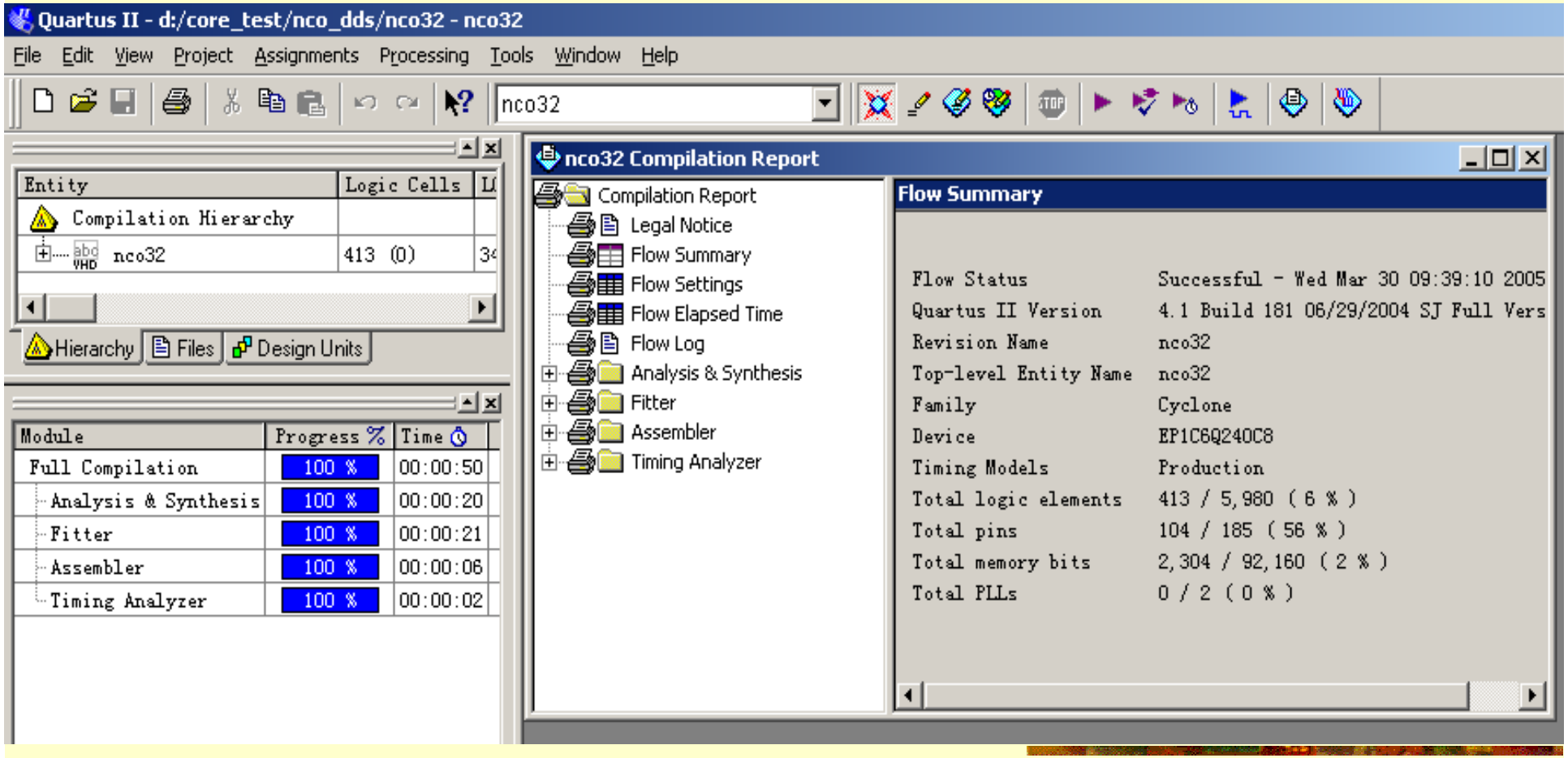


图7-48 设定工程后进行全程编译

KX 康芯科技

7.9 8051单片机IP核应用

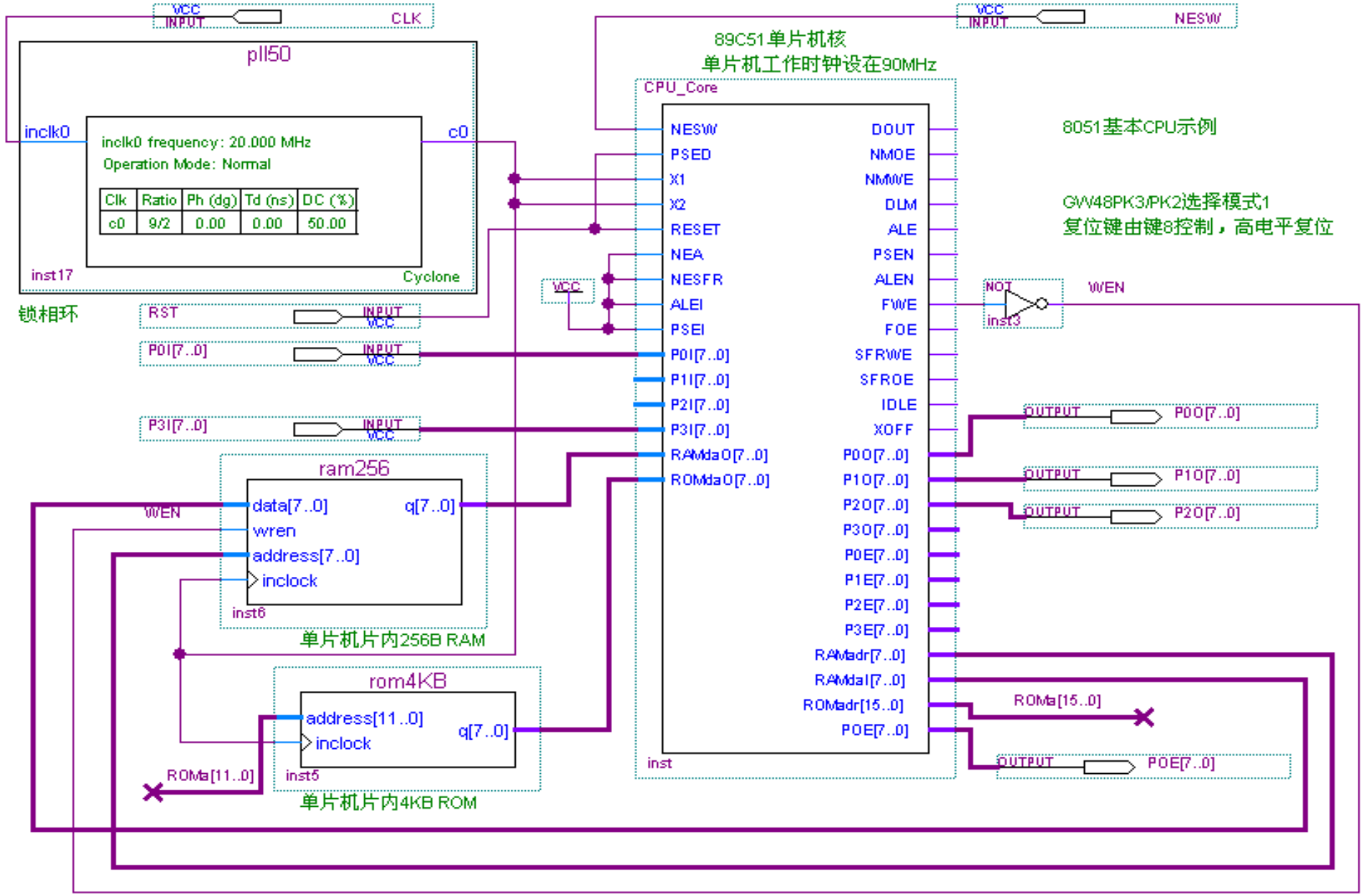


图7-49 基本8051CPU核应用电路示例

7.9 8051单片机IP核应用

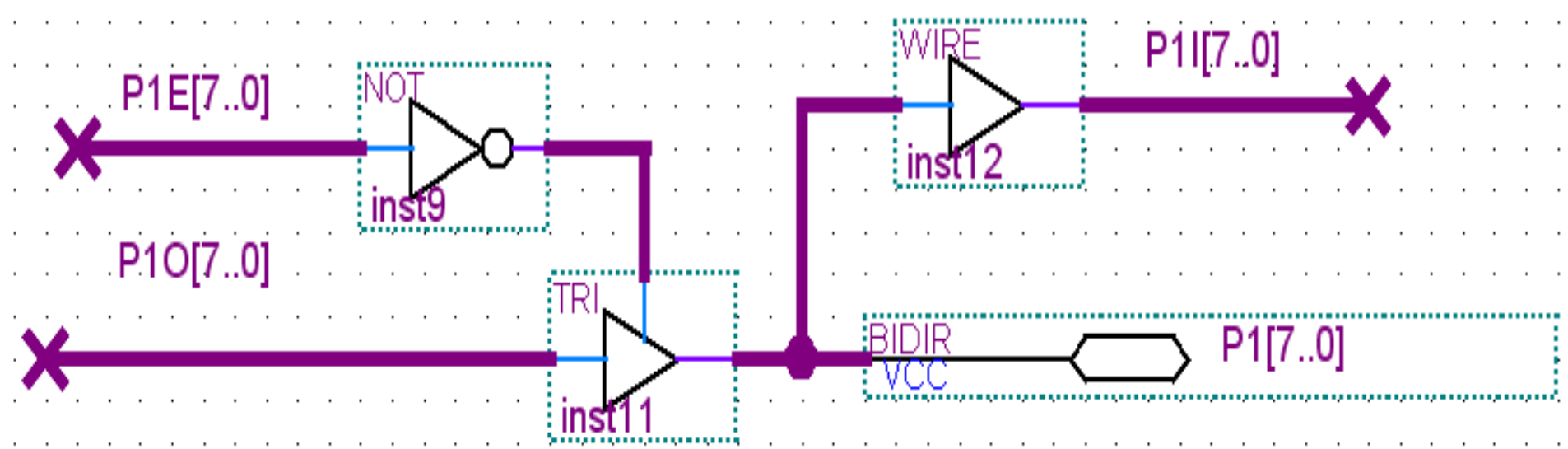


图7-50 单片机I/O口设置成双向口的电路

7.9 8051单片机IP核应用

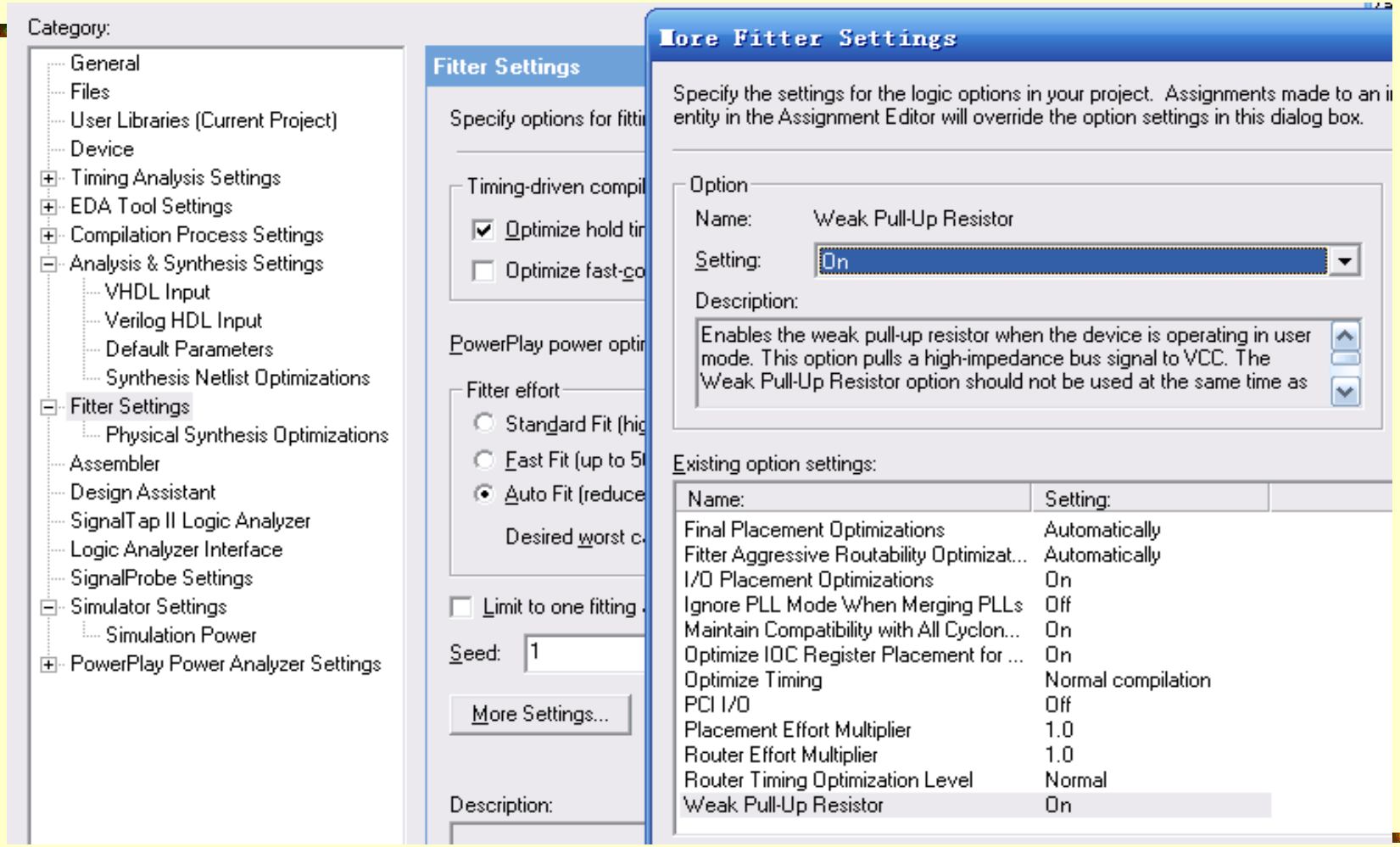


图7-51 设置FPGA的总线口输出为上拉

KX 康芯科技

7.9 8051单片机IP核应用

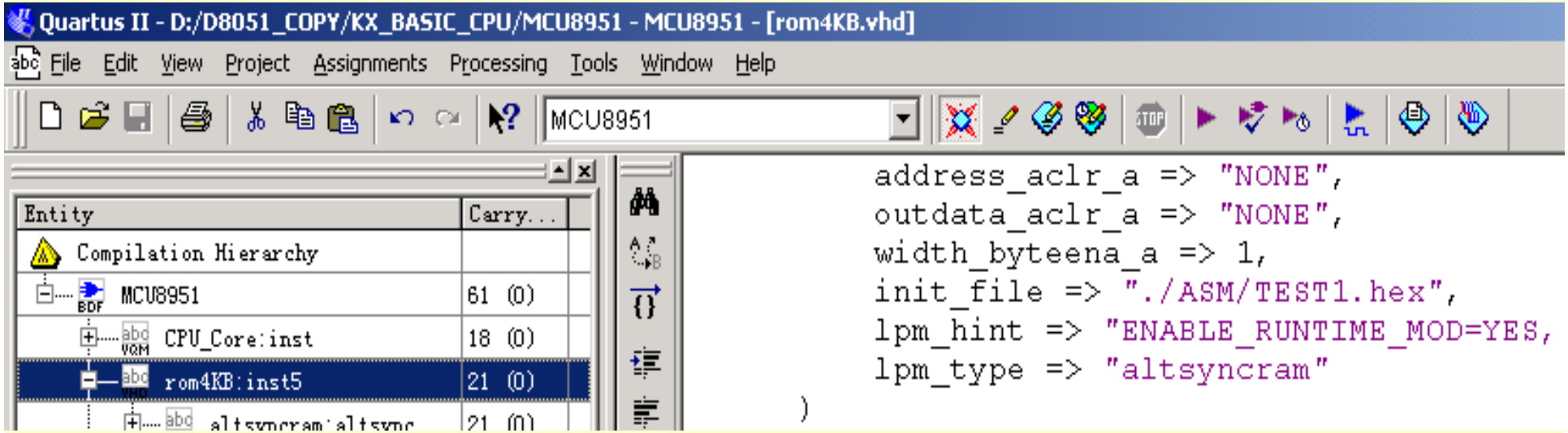


图7-52 LPM_ROM初始化文件路径

```
                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
DELAY:          MOV     20H,#0FFH
    W1:         MOV     21H,#0FFH
    W2:         DJNZ    21H,W2
                DJNZ    20H,W1
                RET
DELAY1:         MOV     22H,#08H
     W3:        LCALL   DELAY
                DJNZ    22H,W3
                RET
                END
```

图7-53 TEST1.asm汇编程序

KX 康芯科技

7.9 8051单片机IP核应用

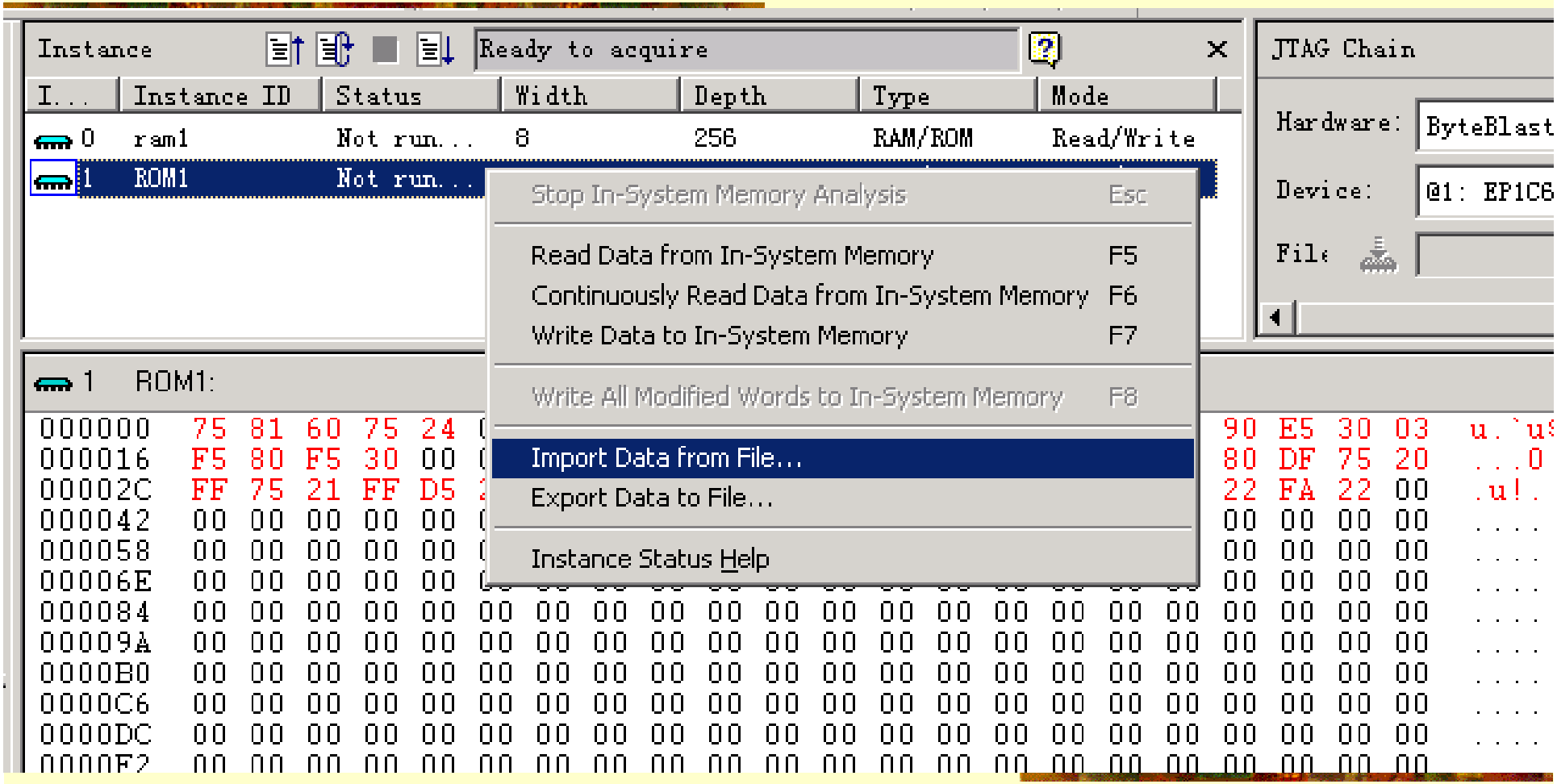


图7-54 下载汇编程序HEX代码

习 题

7-1. 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。

7-2. LPM_ROM、LPM_RAM、LPM_FIFO等模块与FPGA中嵌入的EAB，ESB，M4K有怎样的联系关系？

7-3. 参考QuartusII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。

7-4. 如果要设计一8051单片机，如何为它配置含有汇编程序代码的ROM（文件）？

7-5. 将例**7-4**的顶层程序和例**7-3**的**ROM**程序合并成为一个程序，要求用例化语句直接调用**LPM**模块altsyncram。编译验证，使之功能与原设计相同。

实验与设计

7-1. 正弦信号发生器设计

（1）实验目的：进一步熟悉QuartusII及其LPM_ROM与FPGA硬件资源的使用方法。

（2）实验原理：参考本章相关内容。

（3）实验内容1：根据例7-4，在Quartus II上完成正弦信号发生器设计，包括仿真和资源利用情况了解（假设利用Cyclone器件）。最后在实验系统上实测，包括SignalTap II测试、FPGA中ROM的在系统数据读写测试和利用示波器测试。最后完成EPCSx配置器件的编程。

（4）实验内容2：按照图7-49所示，用原理图方法设计正弦信号发生器，要调用3个LPM模块来构成：1、PLL，输入频率20MHz，32MHz单频率输出；2、6位二进制计数器；3、LPM ROM，加载的波形数据同上。注意，硬件实现时可以通过SignalTapII观察波形，但不能用0832输出，波形必须用高速DAC输出。

实验与设计

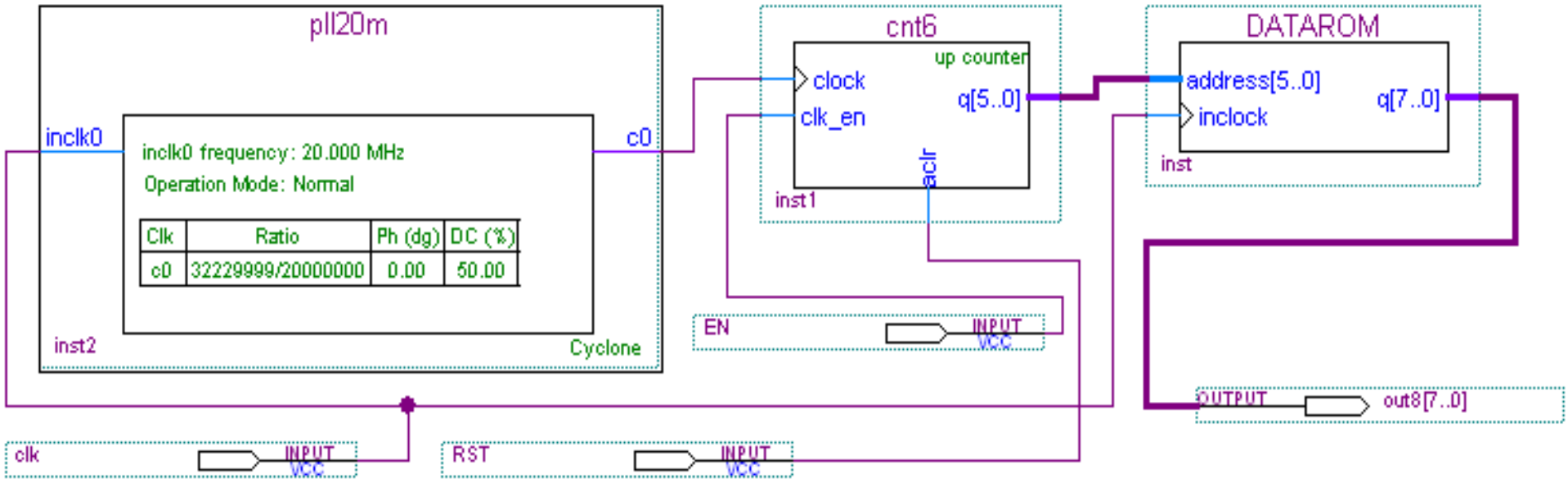


图7-55 调用了PLL元件信号发生器原理图

实验与设计

7-1. 正弦信号发生器设计

（5）实验内容3：修改例7-3的数据ROM文件，设其数据线宽度为8，地址线宽度也为8，初始化数据文件使用MIF格式，用C程序产生正弦信号数据，最后完成以上相同的实验。

（6）实验内容4：设计一任意波形信号发生器，可以使用LPM双口RAM担任波形数据存储器，利用单片机产生所需要的波形数据，然后输向FPGA中的RAM（可以利用GW48系统上与FPGA接口的单片机完成此实验，D/A可利用系统上配置的0832或5651高速器件）。

（7）实验报告：根据以上的实验内容写出实验报告，包括设计原理、程序设计、程序分析、仿真分析、硬件测试和详细实验过程。

实验与设计

7-2. 8位16进制频率计设计

(1) 实验目的：设计8位16进制频率计，学习较复杂的数字系统设计方法。

(2) 实验原理：根据频率的定义和频率测量的基本原理，测定信号的频率必须有一个脉宽为1秒的输入信号脉冲计数允许的信号；1秒计数结束后，计数值被锁入锁存器，计数器清0，为下一测频计数周期作好准备。测频控制信号可以由一个独立的发生器来产生，即图7-57中的FTCTRL。根据测频原理，测频控制时序可以如图7-56所示。

设计要求是：FTCTRL的计数使能信号CNT_EN能产生一个1秒脉宽的周期信号，并对频率计中的32位二进制计数器COUNTER32B（图7-57）的ENABL使能端进行同步控制。当CNT_EN高电平时允许计数；低电平时停止计数，并保持其所计的脉冲数。在停止计数期间，首先需要一个锁存信号LOAD的上跳沿将计数器在前1秒钟的计数值锁存进锁存器REG32B中，并由外部的16进制7段译码器译出，显示计数值。设置锁存器的好处是数据显示稳定，不会由于周期性的清0信号而不断闪烁。锁存信号后，必须有一清0信号RST_CNT对计数器进行清零，为下1秒的计数操作作准备。

实验与设计

7-2. 8位16进制频率计设计

(3) 实验内容1：分别仿真测试模块例7-7、例7-8和例7-9，再结合例7-10完成频率计的完整设计和硬件实现，并给出其测频时序波形及其分析。建议选实验电路模式5；8个数码管以16进制形式显示测频输出；待测频率输入FIN由clock0输入，频率可选4Hz、256HZ、3Hz...50MHz等；1HZ测频控制信号CLK1HZ可由clock2输入(用跳线选1Hz)。注意，这时8个数码管的测频显示值是16进制的。

(4) 实验内容2：参考例4-22，将频率计改为8位10进制频率计，注意此设计电路的计数器必须是8个4位的10进制计数器，而不是1个。此外注意在测频速度上给予优化。

(5) 实验内容3：用LPM模块取代例7-8和例7-9，再完成同样的设计任务。

(6) 实验报告：给出频率计设计的完整实验报告。

【例7-7】

```
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;
```

KX 康芯科技

【例7-8】

```
LIBRARY IEEE;  --32位锁存器
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG32B IS
     PORT (    LK : IN STD_LOGIC;
               DIN : IN STD_LOGIC_VECTOR(31
DOWNTO 0);
              DOUT : OUT STD_LOGIC_VECTOR(31
DOWNTO 0) );
END REG32B;
ARCHITECTURE behav OF REG32B IS
BEGIN
     PROCESS(LK, DIN)
    BEGIN
    IF LK'EVENT AND LK = '1' THEN  DOUT <= DIN;
         END IF;
     END PROCESS;
END behav;
```

【例7-9】

```
LIBRARY IEEE;  --32位计数器
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COUNTER32B IS
    PORT (FIN : IN STD_LOGIC;                    -- 时钟信号
          CLR : IN STD_LOGIC;                      -- 清零信号
         ENABL : IN STD_LOGIC;                  -- 计数使能信号
          DOUT :  OUT STD_LOGIC_VECTOR(31 DOWNTO 0)); -- 计数结果
   END COUNTER32B;
ARCHITECTURE behav OF COUNTER32B IS
    SIGNAL CQI :  STD_LOGIC_VECTOR(31 DOWNTO 0);
BEGIN
    PROCESS(FIN, CLR, ENABL)
      BEGIN
        IF CLR = '1' THEN   CQI <= (OTHERS=>'0');   -- 清零
        ELSIF FIN'EVENT AND FIN = '1' THEN
            IF ENABL = '1' THEN CQI <= CQI + 1; END IF;
        END IF;
    END PROCESS;
    DOUT <= CQI;
END behav;
```

【例7-10】

```
LIBRARY IEEE;  --频率计顶层文件
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FREQTEST IS
    PORT ( CLK1HZ : IN STD_LOGIC;
           FSIN : IN STD_LOGIC;
           DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
END FREQTEST;
ARCHITECTURE struc OF FREQTEST IS
COMPONENT FTCTRL
    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 COMPONENT;
COMPONENT COUNTER32B
    PORT (FIN : IN STD_LOGIC;           -- 时钟信号
          CLR : IN STD_LOGIC;           -- 清零信号
        ENABL : IN STD_LOGIC;           -- 计数使能信号
         DOUT :  OUT STD_LOGIC_VECTOR(31 DOWNTO 0)); -- 计数结果
END COMPONENT;
```

接下页

```
  COMPONENT REG32B
      PORT (    LK : IN STD_LOGIC;
                DIN : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
               DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
  END COMPONENT;
      SIGNAL TSTEN1 : STD_LOGIC;
      SIGNAL CLR_CNT1 : STD_LOGIC;
      SIGNAL Load1 : STD_LOGIC;
      SIGNAL DTO1 : STD_LOGIC_VECTOR(31 DOWNTO 0);
      SIGNAL CARRY_OUT1 : STD_LOGIC_VECTOR(6 DOWNTO 0);
  BEGIN
    U1 :     FTCTRL PORT MAP(CLKK =>CLK1HZ,CNT_EN=>TSTEN1,
  RST_CNT =>CLR_CNT1,Load =>Load1);
    U2 :     REG32B PORT MAP(  LK => Load1,   DIN=>DTO1, DOUT =>
DOUT);
    U3 : COUNTER32B PORT MAP( FIN => FSIN, CLR => CLR_CNT1,
   ENABL => TSTEN1, DOUT=>DTO1 );
  END struc;
```

实验与设计

7-2. 8位16进制频率计设计

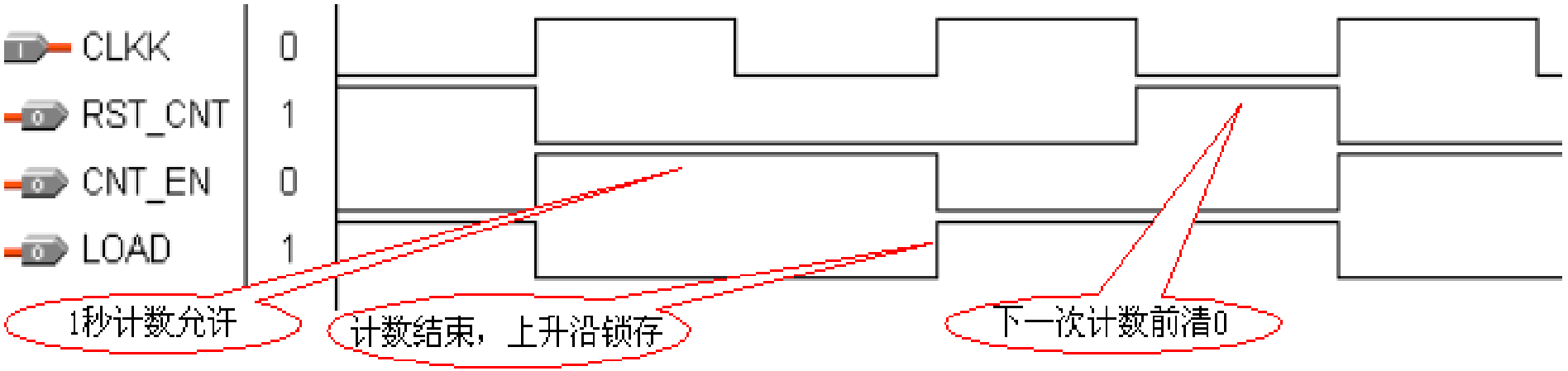


图7-56 频率计测频控制器FTCTRL测控时序图

实验与设计

7-2. 8位16进制频率计设计

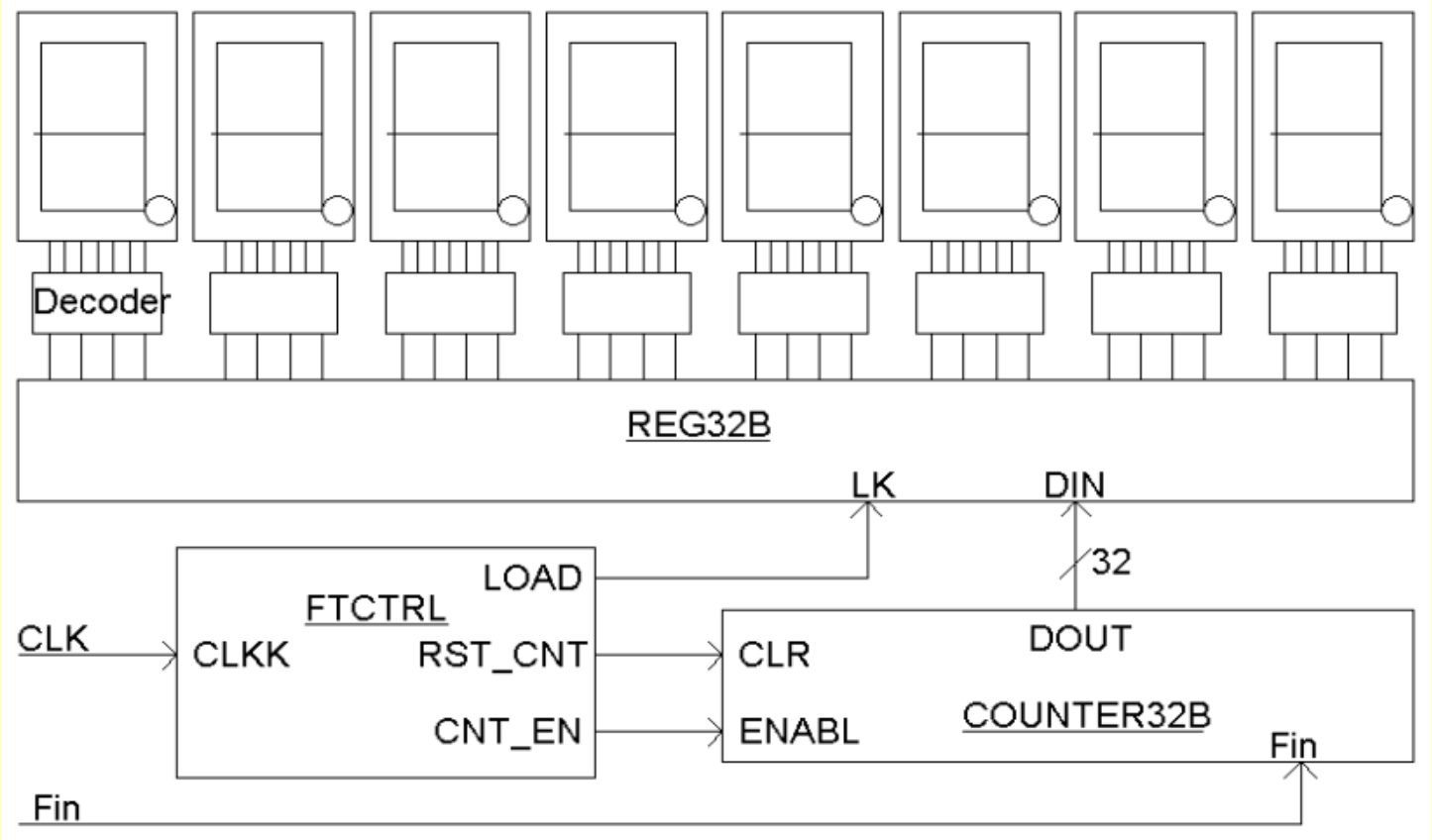


图7-57 频率计电路框图

实验与设计

7-3. 利用LPM_ROM设计乘法器

(1) 实验原理：硬件乘法器有多种设计方法，但相比之下，由**LPM_ROM**构成的乘法表方式的乘法器的运算速度最快。这里定制**LPM_ROM**的地址位宽为**8**；地址输入由时钟**inclock**的上升沿锁入；数据位宽也为**8**。最后为**ROM**配置乘法表数据文件。

LPM_ROM中作为乘法表的数据文件**rom_data.mif**如例**7-11**所示。其中的地址/数据表达方式是，冒号左边写**ROM**地址值，冒号右边写对应此地址放置的**16**进制数据。如**47**：**28**，表示**47**为地址，**28**为该地址中的数据，这样，地址高**4**位和低**4**位可以分别看成是乘数和被乘数，输出的数据可以看成是它们的乘积。

【例7-11】

```
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 ;
```

注意，以上“**CONTENT BEGIN**”下所示的数据格式只是为了节省篇幅，实用中应该使每一数据组（如**01:00 ;**）占一行。

实验与设计

7-3. 利用LPM_ROM设计乘法器

(2) 实验内容： 利用**LPM_ROM**设计**4X4**和**8X8**乘法器各一个，再利用**VHDL**语言描述，由逻辑宏单元构成同类乘法器各一，比较这两类乘法器的运行速度和资源耗用情况。

实验与设计

7-4. IP核应用实验

利用**IP**核完成如下**2**项设计：

1、利用**NCO**核分别设计：

（**1**）**FSK**；（**2**）**PSK**；（**3**）**DDS**；（**4**）移相信号发生器；　（**5**）扫频信号源；（**6**）全数字式锁相环。

2、利用**NCO**和**FIR**核设计数字正交调制解调器（参考清华大学出版社《**SOPC**技术实用教程》中的实验**6-5**）。

实验与设计

7-5. 8051单片机IP核应用实验

(1) 实验内容1：参考**7.9**节，在图**7-49**所示的基本电路平台上增加一些**LPM**或**VHDL**表述硬件模块（如锁存器、译码器、**PWM**发生器、**A/D**采样控制模块、液晶控制模块等），及与单片机的接口电路，利用单片机进行控制，再编辑对应的汇编软件，完成进一步的实验。

(2) 实验内容2：选择不同模式，和引脚锁定情况，协调软件与硬件设计，完成较大的软硬件综合设计模块。

(3) 实验内容3：编辑一段用于测试的汇编程序，利用时序仿真和逻辑分析仪，了解**8051**单片机的数据总线、指令总线、不同指令执行、地址总线、**ALE**、**PSEN**、个**IO**端口、不同指令对应下的端口方向控制信号**P0E**、**P1E**、**P2E**、**P3E**等信号间的时序情况，给出分析报告。