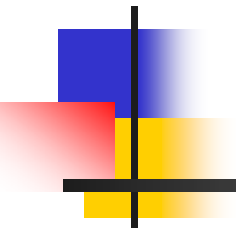


第6章

EDA工具应用深入

6.1 LPM计数器模块调用

6.1.1 计数器LPM模块文本文件的调用

（1）打开宏功能块调用管理器

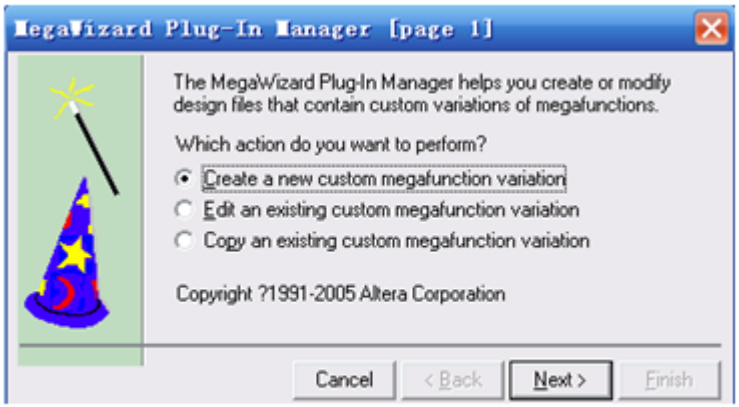


图 6-1　定制新的宏功能块

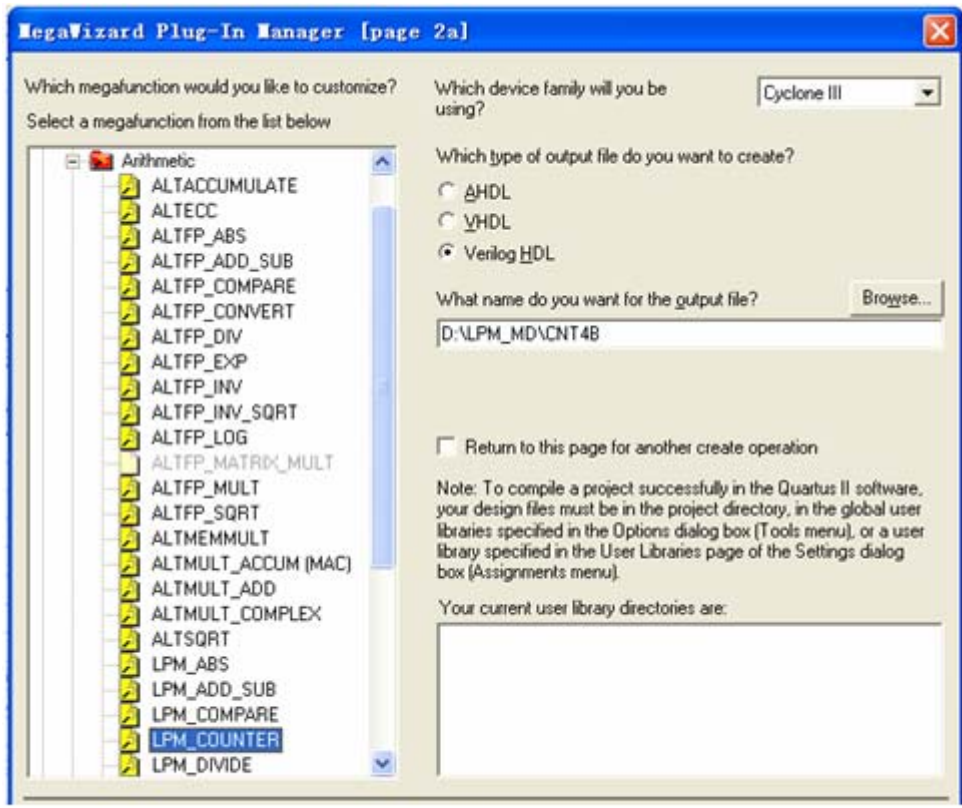


图 6-2　LPM 宏功能块设定

6.1 LPM计数器模块调用

6.1.1 计数器LPM模块文本文件的调用

（2）单击Next按钮后打开如图6-3所示的对话框

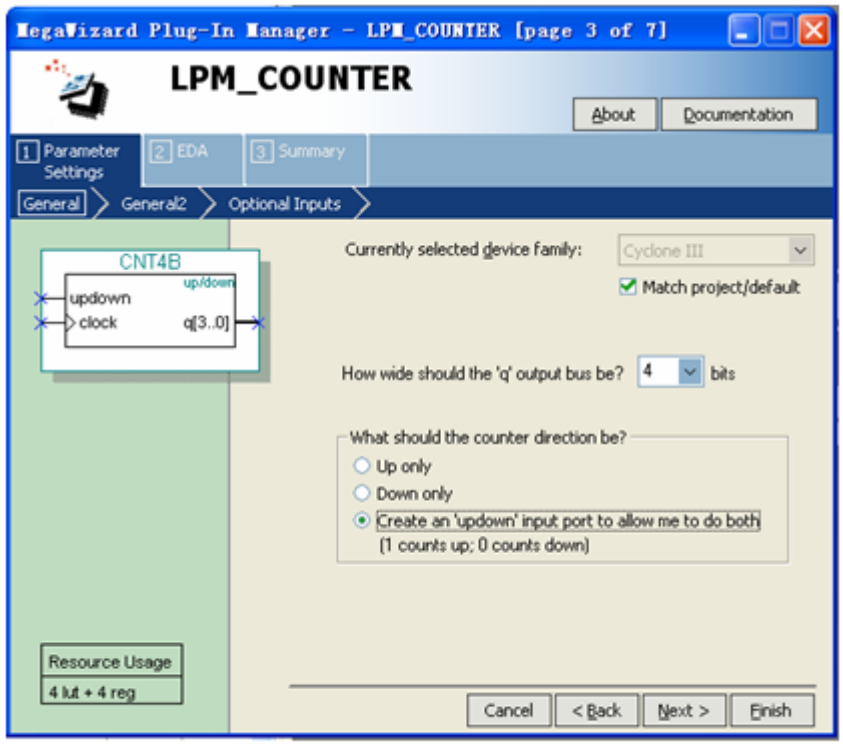


图 6-3 设 4 位可加减计数器

6.1 LPM计数器模块调用

6.1.1 计数器LPM模块文本文件的调用

（3）再单击Next按钮，打开如图7-4所示的对话框

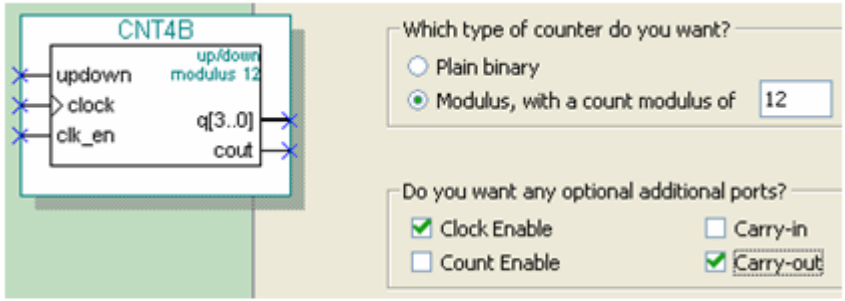


图 6-4 设定计数器，含时钟使能和进位输出

6.1 LPM计数器模块调用

6.1.1 计数器LPM模块文本文件的调用

（4）再单击Next按钮，打开如图6-5所示的对话框

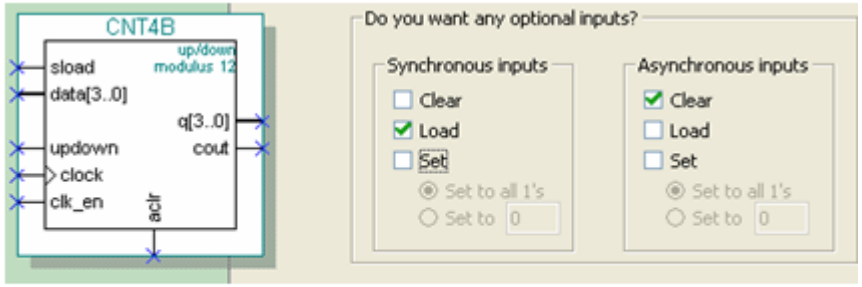


图 6-5 加入 4 位并行数据预置功能

6.1.2 LPM计数器程序与参数传递语句

【例 6-1】 Quartus II 生成的计数器文件 CNT4B.v

```
module CNT4B (aclr, clk_en, clock, data, sload, updown, cout, q);
    input  aclr, clk_en;  // 异步清 0，  1 清 0；时钟使能，1 使能，0 禁止
    input  clock, sload;  // 时钟输入；同步预置数加载控制，1 加载，0 计数
    input  [3:0]  data; input   updown; // 4 位预置数和加减控制，1 加，0 减
    output  cout;   output [3:0]  q;    // 进位输出和 // 4 位计数输出
    wire  sub_wire0;   wire [3:0] sub_wire1; // 定义内部连线
    wire  cout = sub_wire0;  // 与 assign 相同的赋值语句
    wire [3:0] q = sub_wire1[3:0]; // 与 assign 相同的赋值语句
lpm_counter  lpm_counter_component( //注意例化语句中未用端口必须接上指定电平
    .sload(sload), .clk_en(clk_en), .aclr(aclr), .data(data),
    .clock(clock), .updown(updown), .cout(sub_wire0), .q(sub_wire1),
    .aload(1'b0), .aset(1'b0), .cin(1'b1), .cnt_en(1'b1),
    .eq(),  .sclr(1'b0), .sset(1'b0));
 defparam                                          //参数传递说明语句
  lpm_counter_component.lpm_direction = "UNUSED",//单方向计数参数未用
  lpm_counter_component.lpm_modulus = 12,         //模 12 计数器
  lpm_counter_component.lpm_port_updown = "PORT_USED", //使用加减计数
  lpm_counter_component.lpm_type = "LPM_COUNTER",  //计数器类型
  lpm_counter_component.lpm_width = 4;              //计数位宽
endmodule
```

6.1 LPM计数器模块调用

6.1.2 LPM计数器程序与参数传递语句

defparam <宏模块元件例化名>.<宏模块参数名> = <参数值>

【例 6-2】

```
module REG24B (d, clk, q);
    input   [23:0] d;     input   clk;  output  [23:0] q;
    lpm_ff  U1(.q (q[11:0]), .data (d[11:0]), .clock (clk));
        defparam U1.lpm_width = 12;
    lpm_ff  U2(.q (q[23:12]), .data (d[23:12]), .clock (clk));
       defparam U2.lpm_width = 12;
endmodule
```

6.1 LPM计数器模块调用

6.1.2 LPM计数器程序与参数传递语句

【例 6-3】

```
module CNT4BIT (RST,ENA,CLK,DIN,SLD,UD,COUT,DOUT);
   input RST, ENA, CLK, SLD,UD ;   input [3:0]  DIN;
   output  COUT; output [3:0]  DOUT ;
   CNT4B  U1(.sload (SLD), .clk_en (ENA), .aclr (RST), .cout (COUT),
            .clock (CLK), .data (DIN),  .updown (UD), .q (DOUT) );
endmodule
```

6.1 LPM计数器模块调用

6.1.3 创建工程与仿真测试

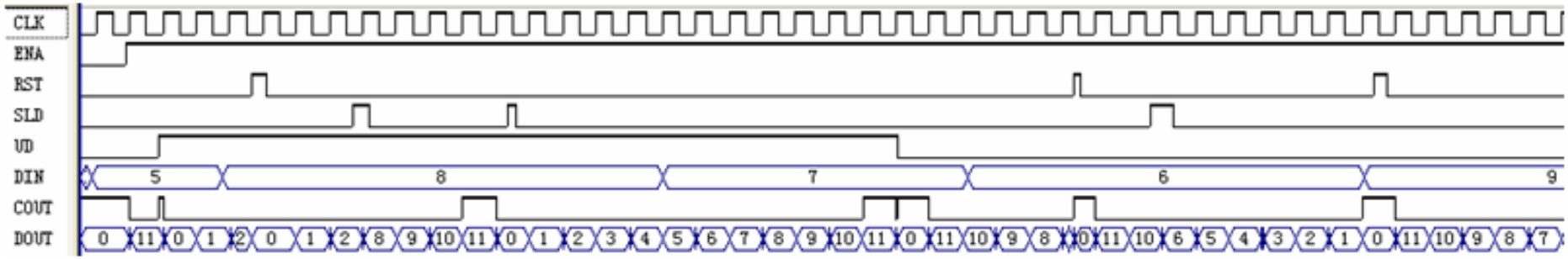


图 6-6 CNT4BIT.v 的仿真波形

6.1 LPM计数器模块调用

6.1.3 创建工程与仿真测试

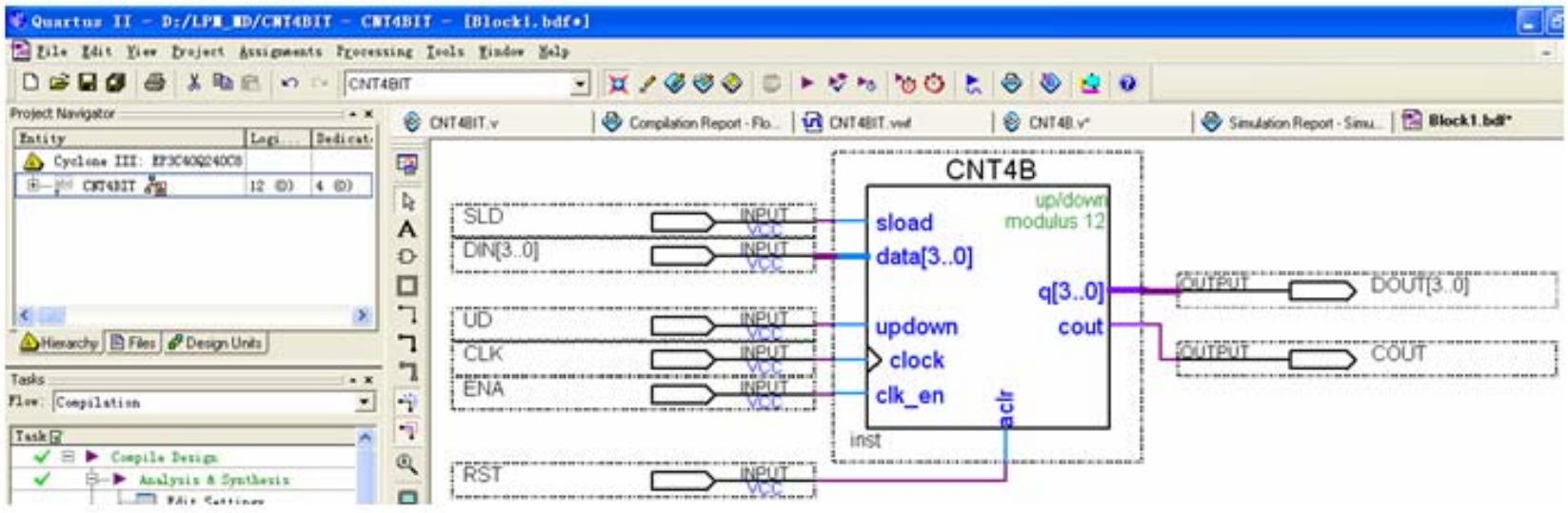


图 6-7　原理图输入设计

6.2 流水线乘法累加器设计

6.2.1 LPM加法器模块设置

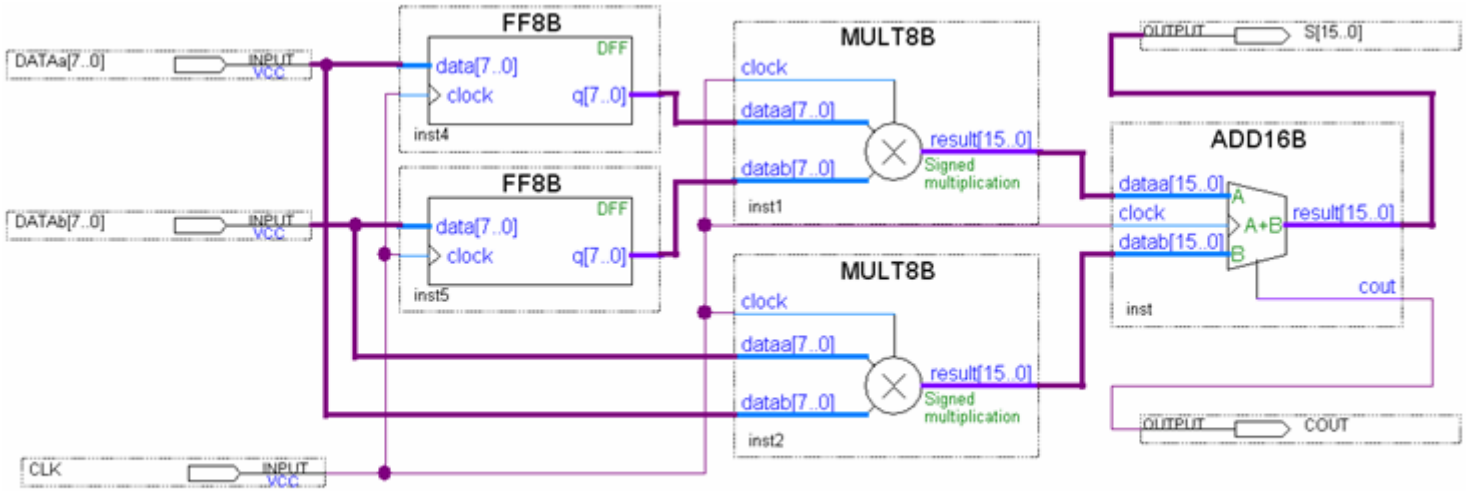


图 6-8　8 位乘法累加器顶层设计

6.2 流水线乘法累加器设计

6.2.1 LPM加法器模块设置

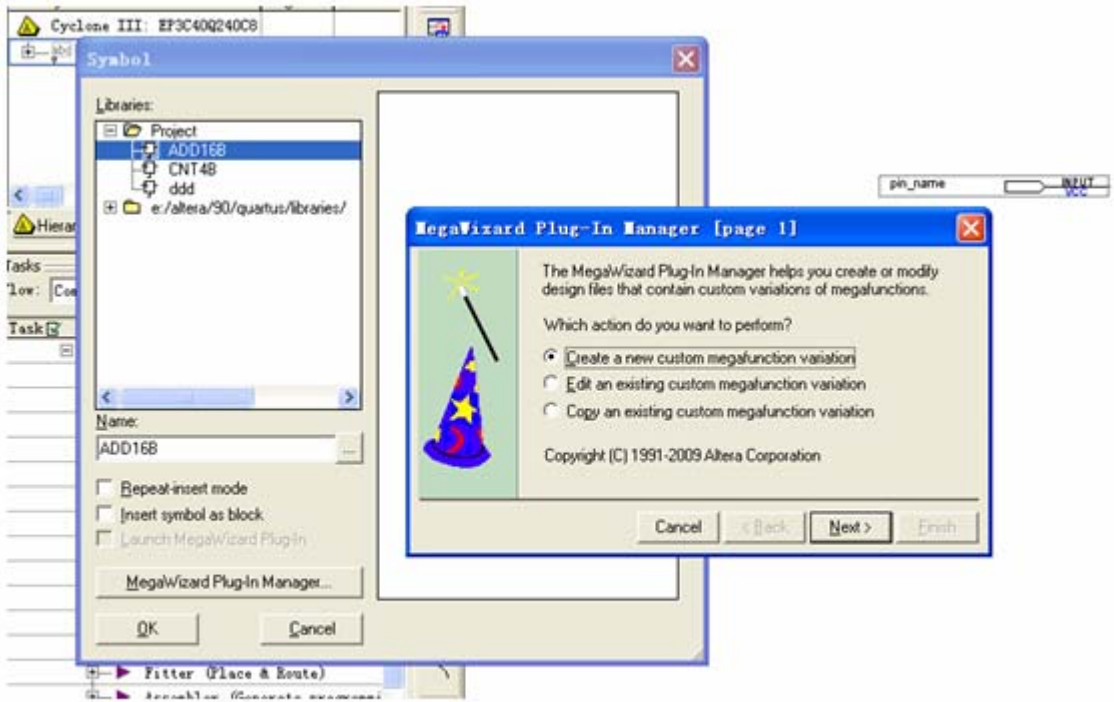


图 6-9　从原理图编辑窗进入 MegaWizard Plug-In Manager 管理器

6.2 流水线乘法累加器设计

6.2.1 LPM加法器模块设置

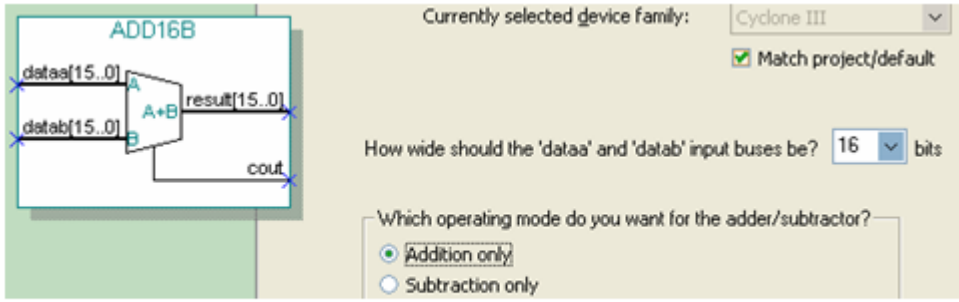


图 6-10 选择 16 位加法工作方式

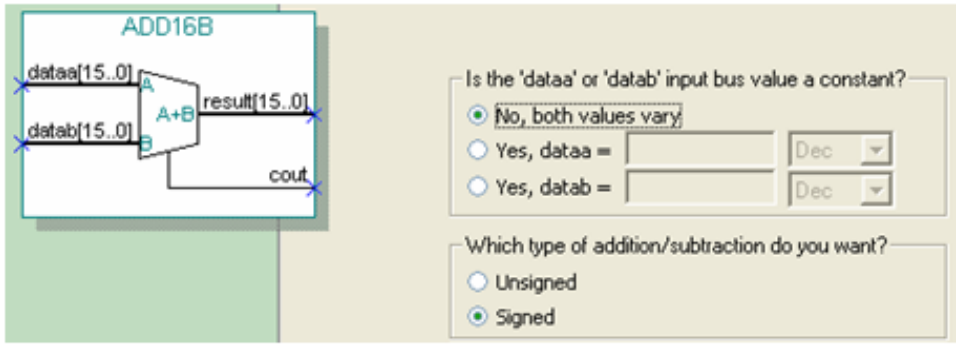


图 6-11 选择有符号加法操作类型输入

6.2 流水线乘法累加器设计

6.2.1 LPM加法器模块设置

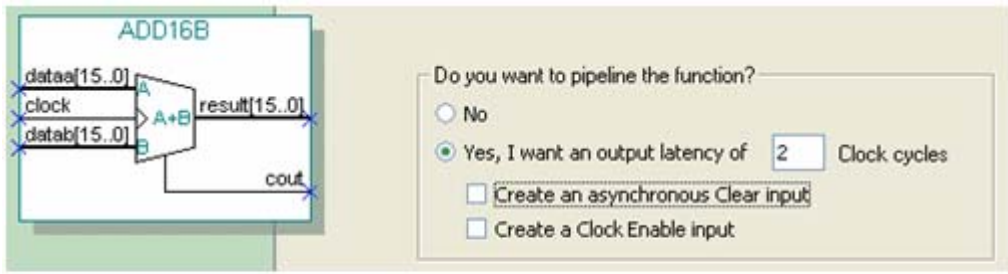


图 6-12　选择流水线方式

6.2 流水线乘法累加器设计

6.2.2 LPM乘法器模块设置

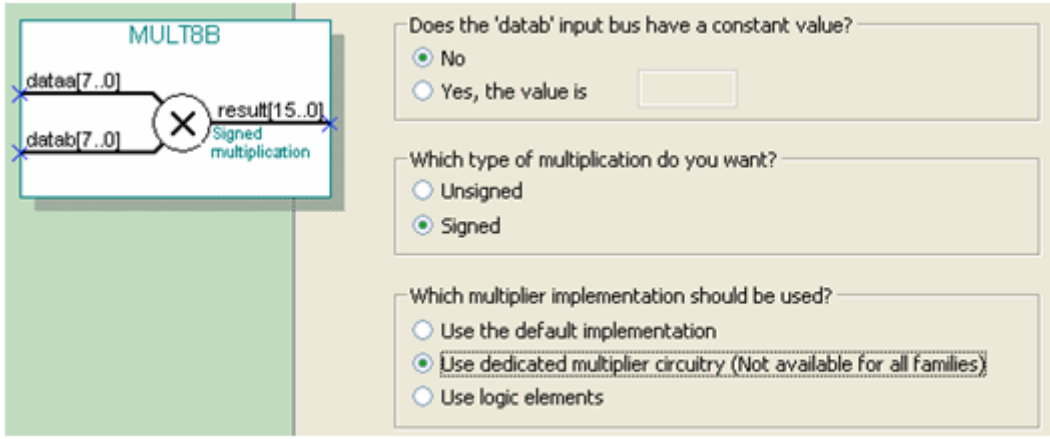


图 6-13　选择有符号乘法模式，并用专用乘法器模块构建乘法器

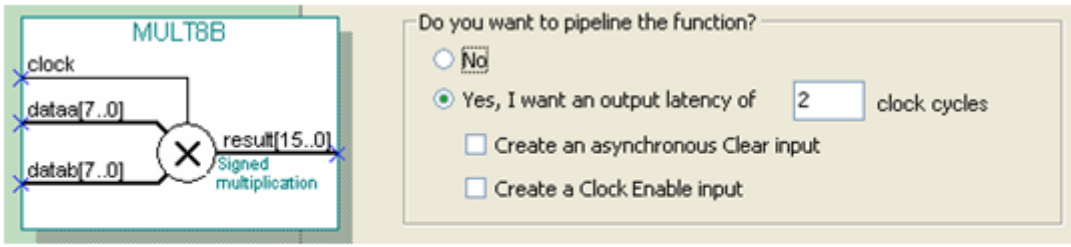


图 6-14　选择 2 级流水线乘法模式

6.2 流水线乘法累加器设计

6.2.3 仿真乘法累加器

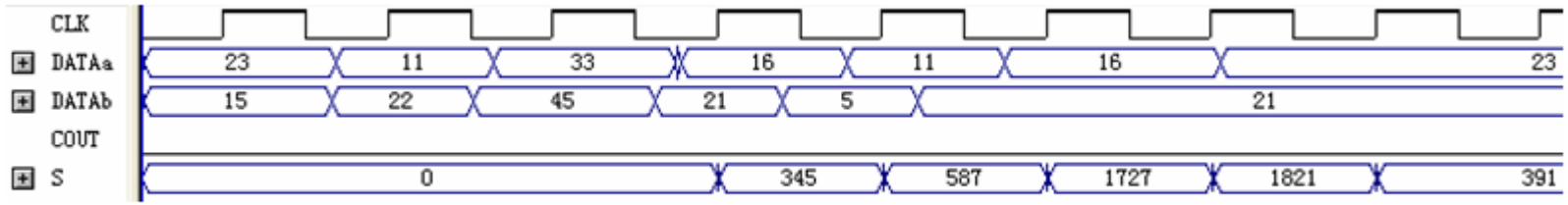


图 6-15　电路图图 6-8 的 MULTADD 工程仿真波形

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

6.2 流水线乘法累加器设计

6.2.3 仿真乘法累加器

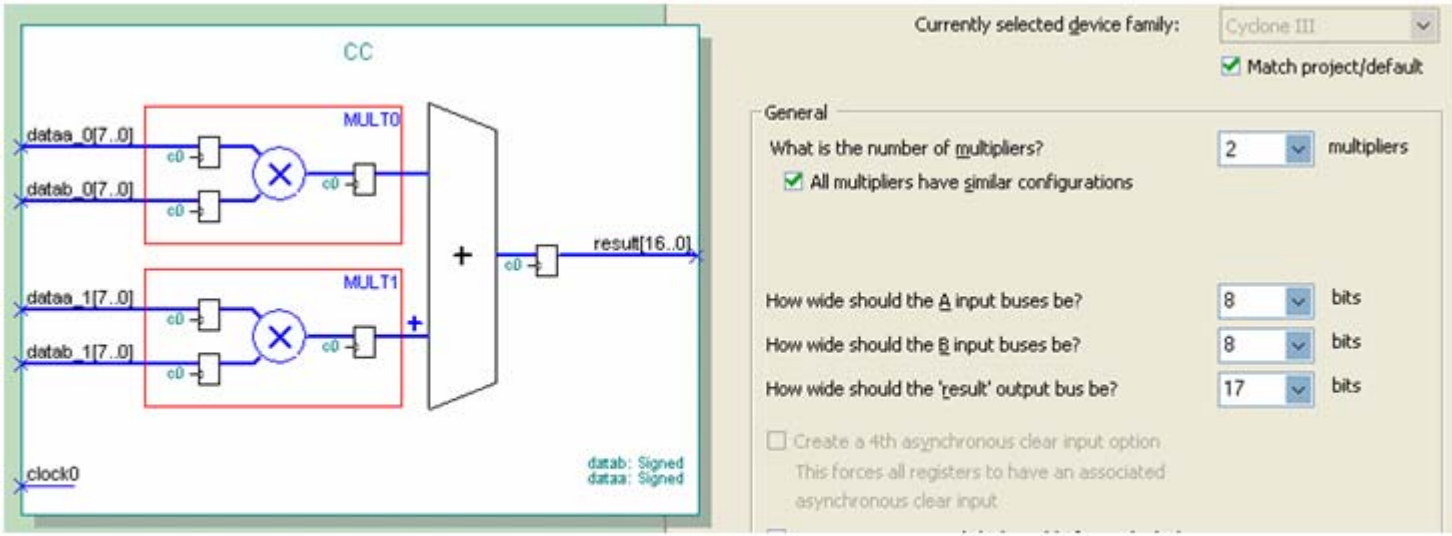


图 6-17　ALTMULT_ADD 模块设置对话框

6.2 流水线乘法累加器设计

6.2.4 乘法器的Verilog文本表述和相关属性设置

```
/* synthesis multstyle = "logic" */

/* synthesis multstyle = "dsp" */
```

【例 6-4】

```
module MULT8 (A1,B1,A2,B2,R1,R2) ;
    output signed[15:0]  R1, R2  ;  // 定义有符号数据类型输出
    input signed[7:0] A1,B1,A2,B2; // 定义有符号数据类型输入
    wire  [15:0]  R2  /* synthesis multstyle = "logic" */;
    wire  [15:0]  R1 /* synthesis multstyle = "dsp" */;
    assign R1 = A1 * B1  ;
    assign R2 = A2 * B2  ;
  endmodule

module andd(A1,B1,A2,B2,R1,R2)  /* synthesis multstyle = "dsp" */;
```

6.2 流水线乘法累加器设计

6.2.4 乘法器的Verilog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 6-4 的编译报告

6.2 流水线乘法累加器设计

6.2.4 乘法器的Verilog文本表述和相关属性设置

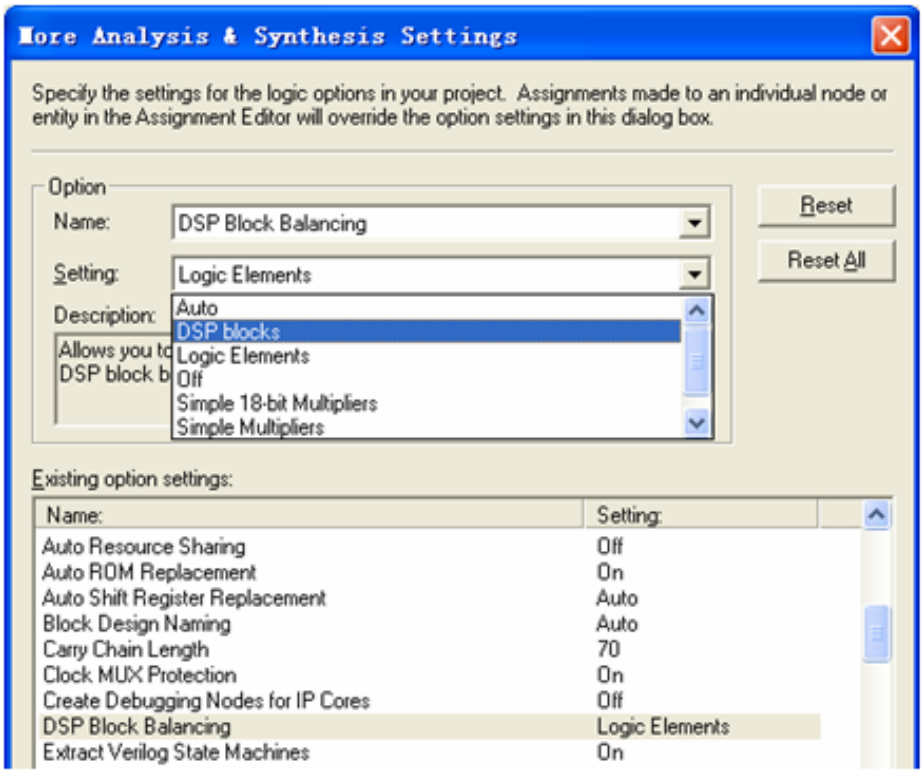


图 6-19　设置乘法器用 DSP 模块构建

6.3 LPM_RAM模块的设置

6.3.1 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

6.3 LPM_RAM模块的设置

（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;
```

6.3 LPM_RAM模块的设置

（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));
    }    }
```

6.3 LPM_RAM模块的设置

（4）专用mif文件生成器

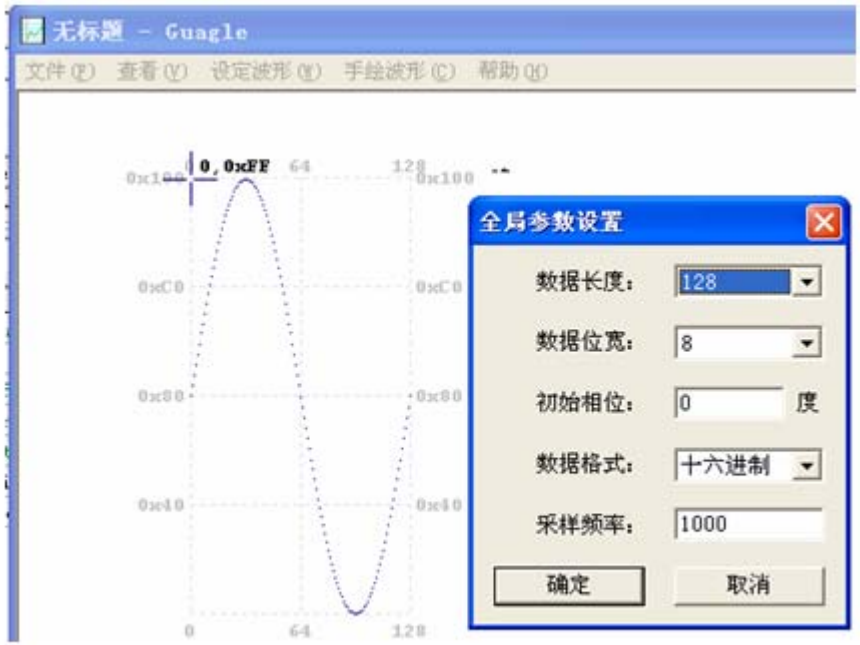


图 6-22 利用康芯 mif 生成器生成 mif 正弦波数据文件

图 6-23 打开 mif 文件

6.3 LPM_RAM模块的设置

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

6.3 LPM_RAM模块的设置

6.3.2 LPM_RAM设置和调用

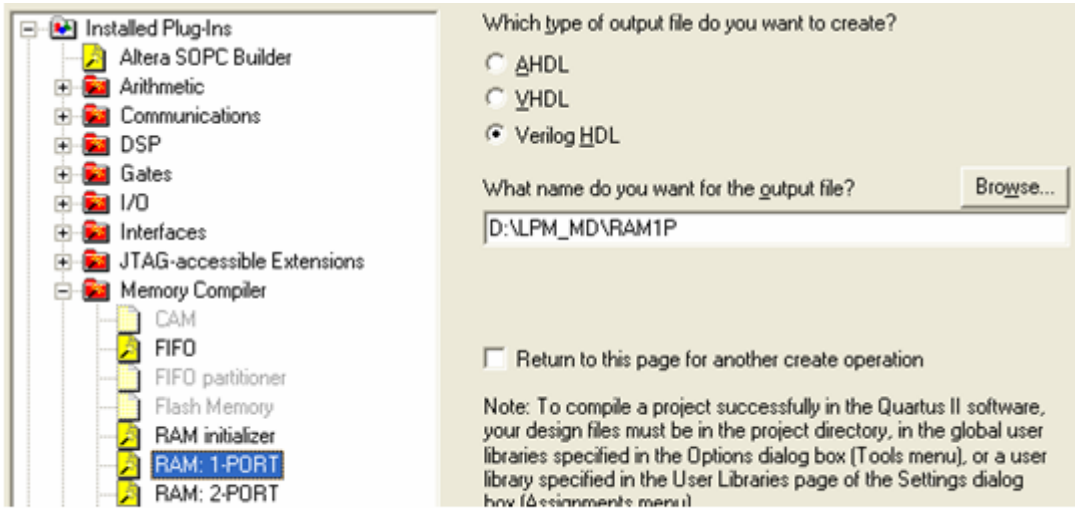


图 6-24　调用单口 LPM RAM

6.3 LPM_RAM模块的设置

6.3.2 LPM_RAM设置和调用

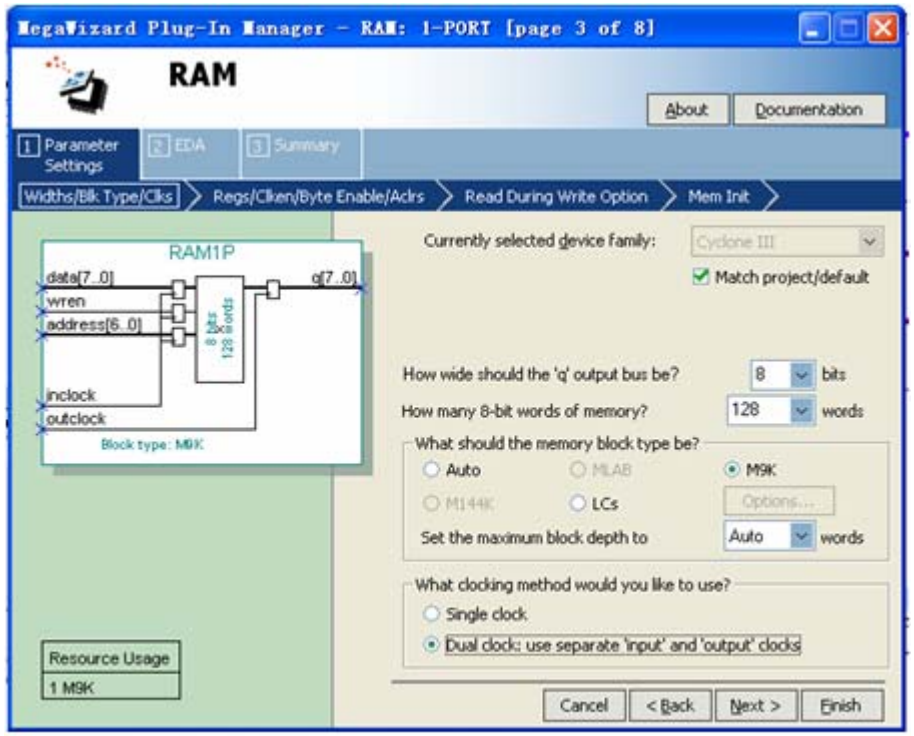


图 6-25　设定 RAM 参数

6.3 LPM_RAM模块的设置

6.3.2 LPM_RAM设置和调用

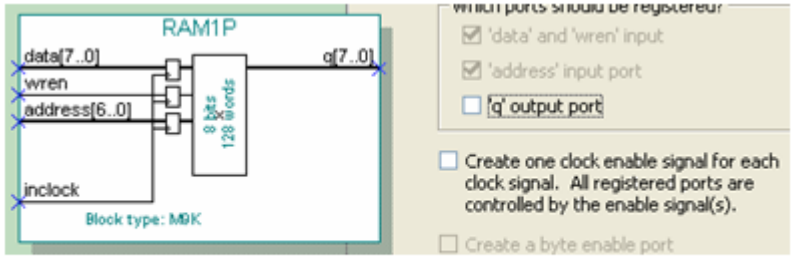


图 6-26　设定 RAM 仅输入时钟控制

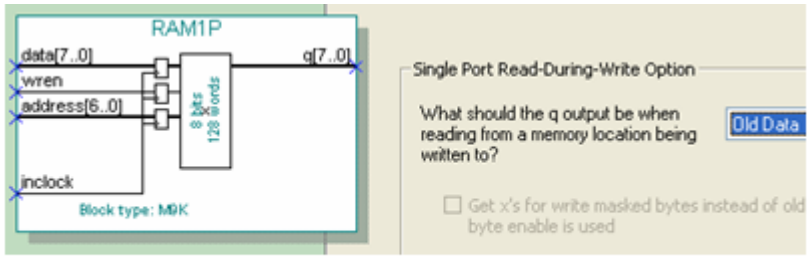


图 6-27　设定在写入同时读出原数据：Old Data

6.3 LPM_RAM模块的设置

6.3.2 LPM_RAM设置和调用

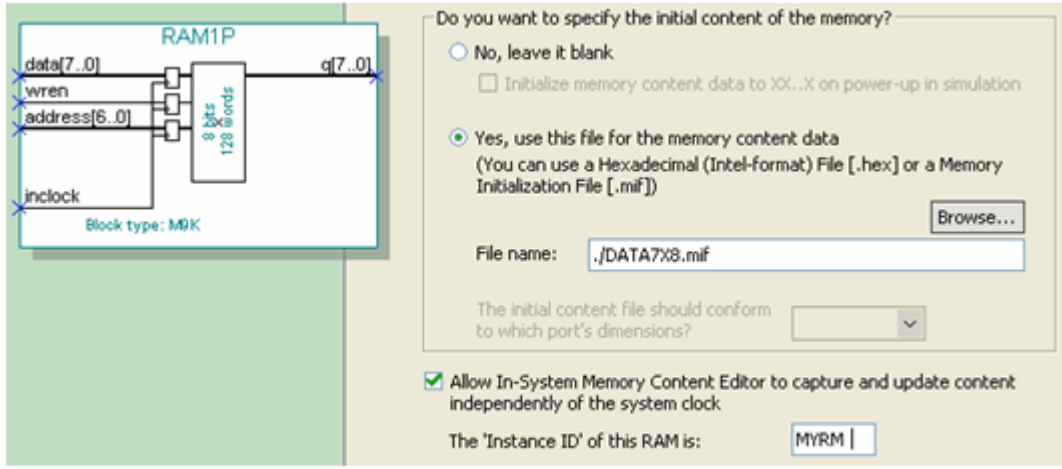


图 6-28 设定初始化文件和允许在系统编辑

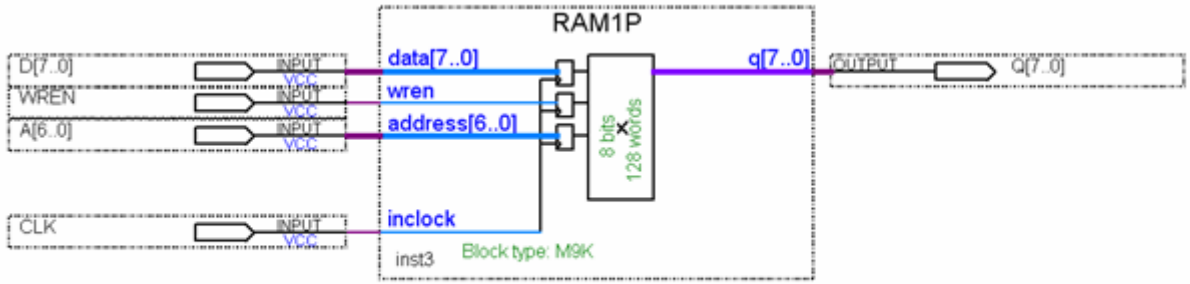


图 6-29 在原理图上连接好的 RAM 模块

6.3 LPM_RAM模块的设置

6.3.3 测试LPM_RAM

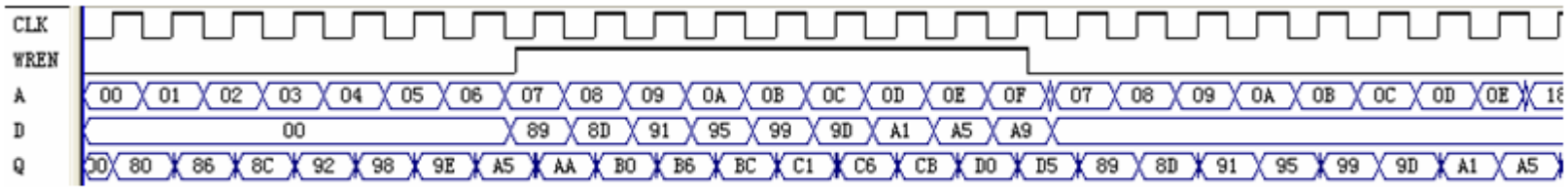


图 6-30 图 6-29 的 RAM 的仿真波形

6.3 LPM_RAM模块的设置

6.3.4 存储器的Verilog文本描述及相关属性应用

【例 6-7】

```
module RAM78 ( output wire[7:0] Q,    //定义 RAM 的 8 位数据输出端口
               input wire[7:0] D,     //定义 RAM 的 8 位数据输入端口
               input wire[6:0] A,     //定义 RAM 的 7 位地址输入端口
               input wire CLK,WREN ) ;  //定义时钟和写允许控制
   reg[7:0]  mem[127:0]  /* synthesis ram_init_file="DATA7X8.mif" */ ;
   always @(posedge CLK )
     if (WREN) mem[A] <= D;  //在 CLK 上升沿将数据口 D 的数据锁入地址对应单元中
   assign Q = mem[A];          //同时，地址对应单元的数据被输出端口
endmodule
```

6.3 LPM_RAM模块的设置

6.3.4 存储器的Verilog文本描述及相关属性应用

1. 存储器端口描述

```
module RAM78(Q, D, A, CLK, WREN);
  output[7:0] Q;   input[7:0] D;    input[6:0] A;  input  CLK,WREN  ;
module RAM78(output[7:0] Q,input[7:0] D, input[6:0] A, input CLK,WREN);
```

6.3 LPM_RAM模块的设置

2．存储器的Verilog一般描述

```
parameter width=8, msize=1024;
reg[width-1:0] MEM87[msize-1:0];
```

```
reg[7:0] mem87[128:0];
mem87[16]=8'b11001001; //mem87 存储器的第 16 单元被赋值为二进制数 11001001
mem87[122]=76;         //mem87 存储器的第 122 单元被赋值为十进制数 76。
```

```
reg [15:0] A;    //定义了一个 16 位的寄存器
reg MEM[15:0];   //定义了一个字长为 1，即 1 位的，容量深度为 16 的存储器
```

```
A[5] = 1'b0;          //允许对寄存器 A 的第 5 位赋值 0
MEM[7] = 1'b1;        //允许对存储器 MEM 的第 7 个单元赋值 1
A = 16'hABCD ;        //允许寄存器 A 整体赋值
MEM = 16'hABCD;       //不允许对存储器多个或者所有单元同时赋值
```

6.3 LPM_RAM模块的设置

3．存储器初始化文件属性应用

```
/* synthesis ram_init_file="DATA7X8.mif" */ ;
```

```
(* ram_init_file = "DATA7X8.mif" *) reg[7:0]  mem[127:0]
```

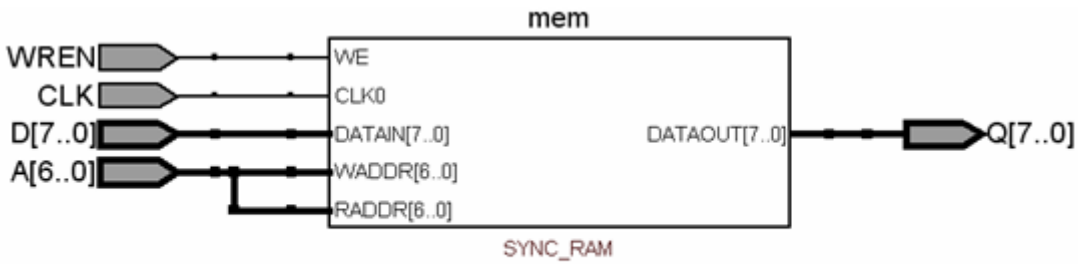


图 6-31 例 6-7 的 RAM78 的 RTL 图

Family	Cyclone III
Device	EP3C5E144C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	1,497 / 5,136 (29 %)
Total combinational functions	1,340 / 5,136 (26 %)
Dedicated logic registers	1,024 / 5,136 (20 %)
Total registers	1024
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

6.3 LPM_RAM模块的设置

4．文本方式调用存储器LPM模块

【例 6-8】

```
module RAM78(Q1, D1, A1, CLK1, WREN1);
  output[7:0] Q1;     input[7:0] D1;    input[6:0] A1;  input  CLK1,WREN1;
   RAMP  IC1(.A(A1), .D(D1), .CLK(CLK1), .Q(Q1),  .WREN(WREN1) );
endmodule
```

Family	Cyclone III
Device	EP3C5E144C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	168 / 5,136 (3 %)
Total combinational functions	156 / 5,136 (3 %)
Dedicated logic registers	98 / 5,136 (2 %)
Total registers	98
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

【例6-9】

```
module RAMP (A,D,CLK,WREN,Q); //RAM1P.v
  input[6:0] A;  input[7:0] D;  input CLK;  input WREN ;
  output[7:0] Q;
  altsyncram  U1(                          //例化 ram 模块
        .wren_a(WREN), .clock0(CLK), .address_a(A),
        .data_a(D), .q_a(Q),  .aclr0(1'b0),
        .aclr1(1'b0),  .address_b(1'b1), .addressstall_a(1'b0),
        .addressstall_b(1'b0), .byteena_a(1'b1), .byteena_b(1'b1),
        .clock1 (1'b1),  .clocken0 (1'b1), .clocken1 (1'b1),
        .clocken2(1'b1),.clocken3(1'b1),.data_b(1'b1),.wren_b(1'b0),
        .eccstatus (),   .q_b(), .rden_a(1'b1), .rden_b(1'b1)
           );
 defparam       // 参数传递
  U1.clock_enable_input_a = "BYPASS",
  U1.clock_enable_output_a = "BYPASS",
  U1.init_file = "DATA7X8.mif",        //初始化数据文件,后缀最好小写
  U1.intended_device_family = "Cyclone III",
  U1.lpm_hint = "ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=MYRM",
  U1.lpm_type = "altsyncram",          //LPM 类型
  U1.numwords_a = 128,                 //数据数量 64
  U1.operation_mode = "SINGLE_PORT",
  U1.outdata_aclr_a = "NONE",          //无异步地址清 0
  U1.outdata_reg_a = "UNREGISTERED",  //输出无锁存
  U1.power_up_uninitialized = "FALSE",
  U1.ram_block_type = "M9K",
  U1.read_during_write_mode_port_a = "OLD_DATA",
  U1.widthad_a = 7,                    //地址线宽度 6
  U1.width_a = 8,                      //数据线宽度 8
  U1.width_byteena_a = 1;              //byteena_a 输入口宽度 1
 endmodule
```

6.4 LPM_ROM的定制和使用示例

6.4.1 LPM_ROM定制和测试

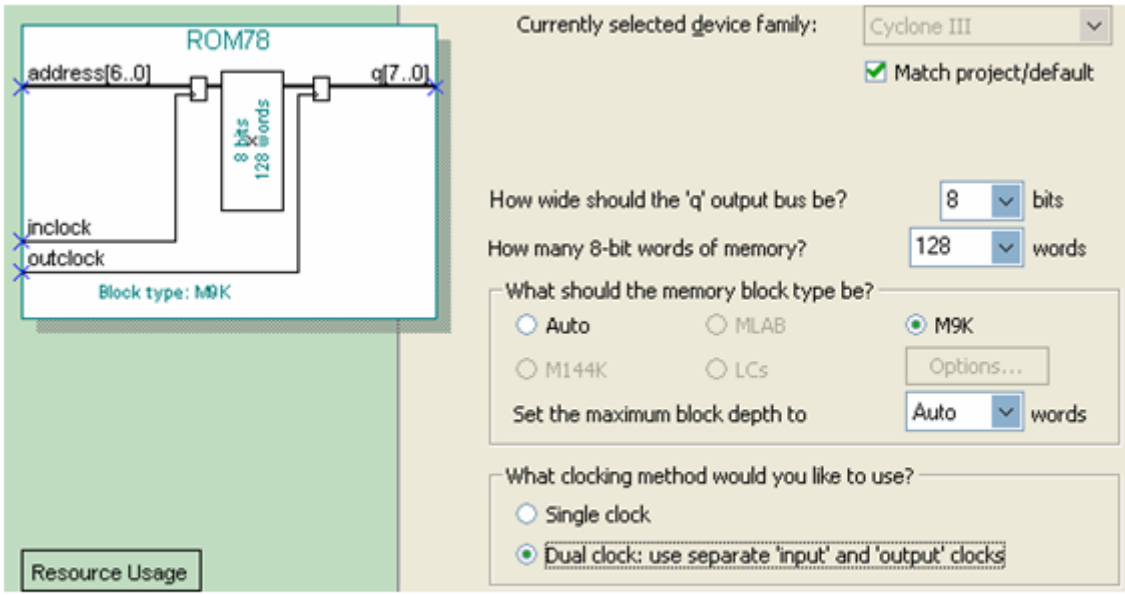


图 6-34 调用 LPM ROM 之参数设置

6.4 LPM_ROM的定制和使用示例

6.4.1 LPM_ROM定制和测试

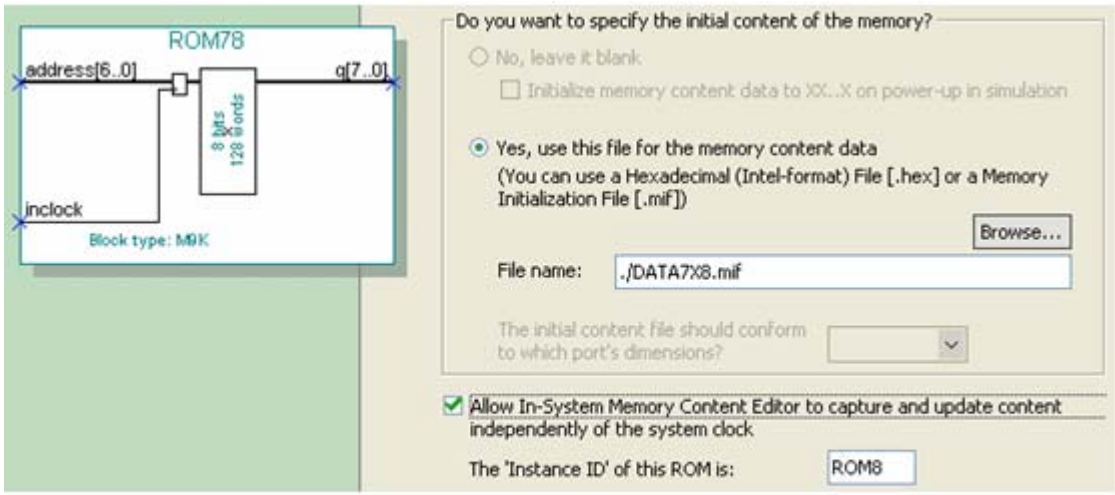


图 6-35 加入初始化配置文件

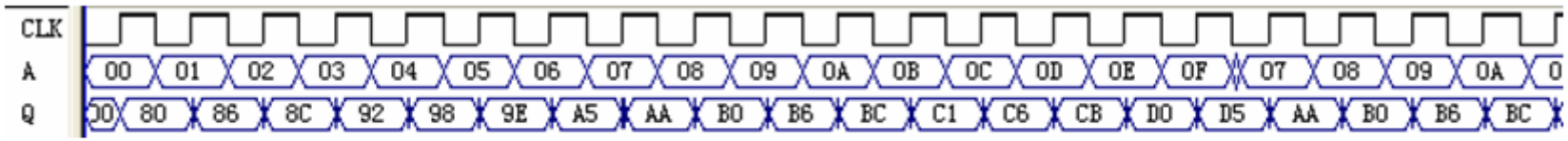
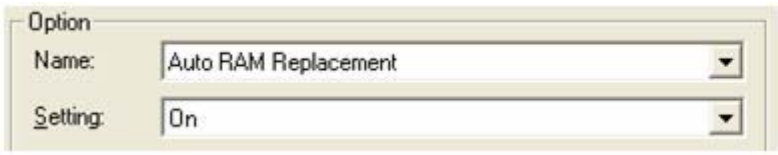
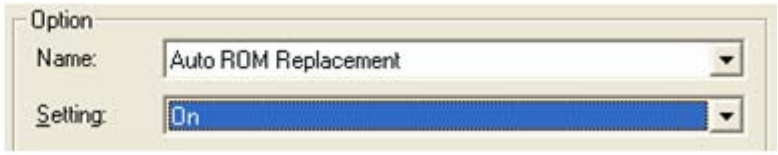


图 6-36 LPM ROM 仿真测试

6.4 LPM_ROM的定制和使用示例

6.4.2 LPM存储器模块取代设置





6.4 LPM_ROM的定制和使用示例

6.4.3 正弦信号发生器设计

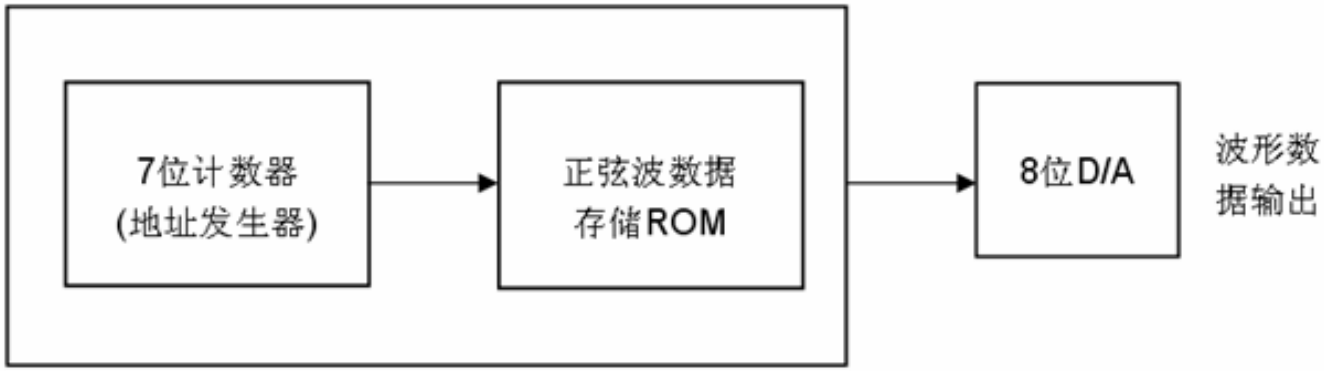


图 6-37　正弦信号发生器结构框图

6.4 LPM_ROM的定制和使用示例

6.4.3 正弦信号发生器设计

【例 7-10】

```
module SIN_GNT(RST,CLK,EN, Q,AR);
   output [7:0] Q ;  output [6:0] AR ; //AR是7位地址发生器输出测试口
   input  EN,CLK,RST ;   wire [6:0] TMP;    reg [6:0] Q1 ;
    always @(posedge CLK or negedge RST )
       if(!RST)   Q1 <=7'B0000000;
         else if (EN)   Q1 <= Q1+1 ;
         else  Q1 <= Q1;
    assign TMP=Q1;        assign  AR=TMP;
     ROM78  IC1(.address(TMP), .inclock(CLK), .q (Q) );//例化ROM78.v
endmodule
```

6.4 LPM_ROM的定制和使用示例

6.4.3 正弦信号发生器设计

【例 7-11】 ROM78.v

```
module ROM78 (address, inclock,    q);
   input   [6:0]  address; input  inclock; output[7:0]  q;
 ...
```

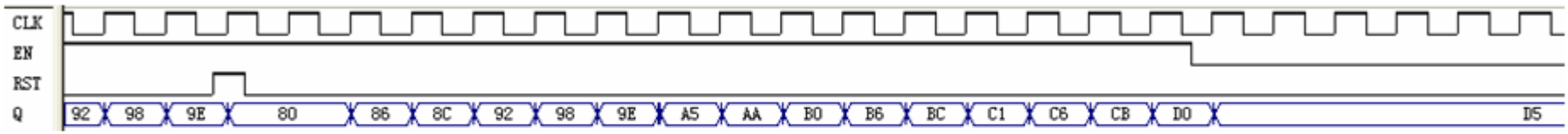


图 6-38　例 6-10 的仿真波形输出

6.4 LPM_ROM的定制和使用示例

6.4.3 正弦信号发生器设计

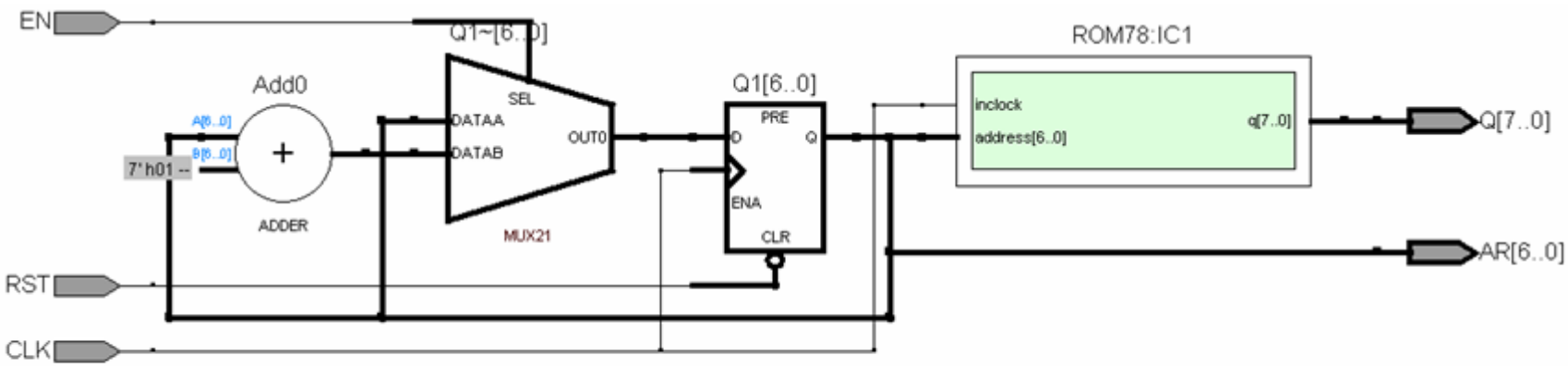


图 6-39 例 6-10 的 RTL 电路图

6.4 LPM_ROM的定制和使用示例

6.4.4 硬件实现和测试

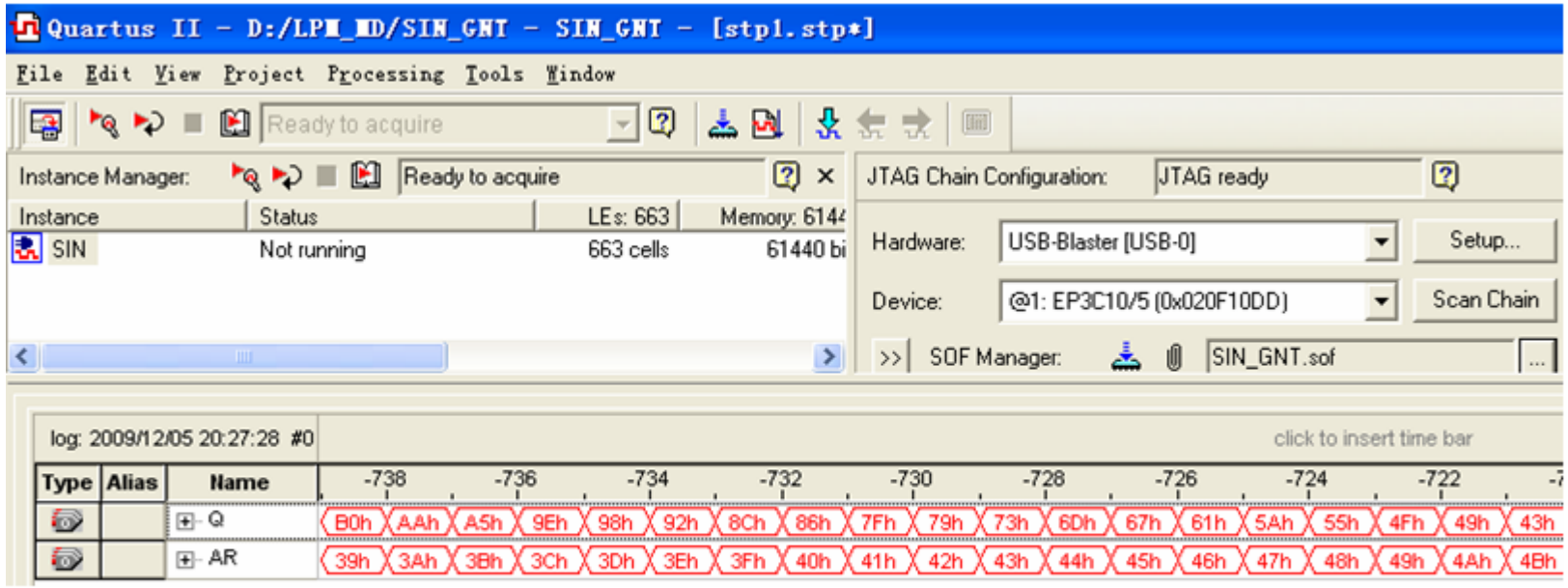


图 6-40 正弦信号发生器数据输出的 SignalTapII 测试图

6.4 LPM_ROM的定制和使用示例

6.4.4 硬件实现和测试

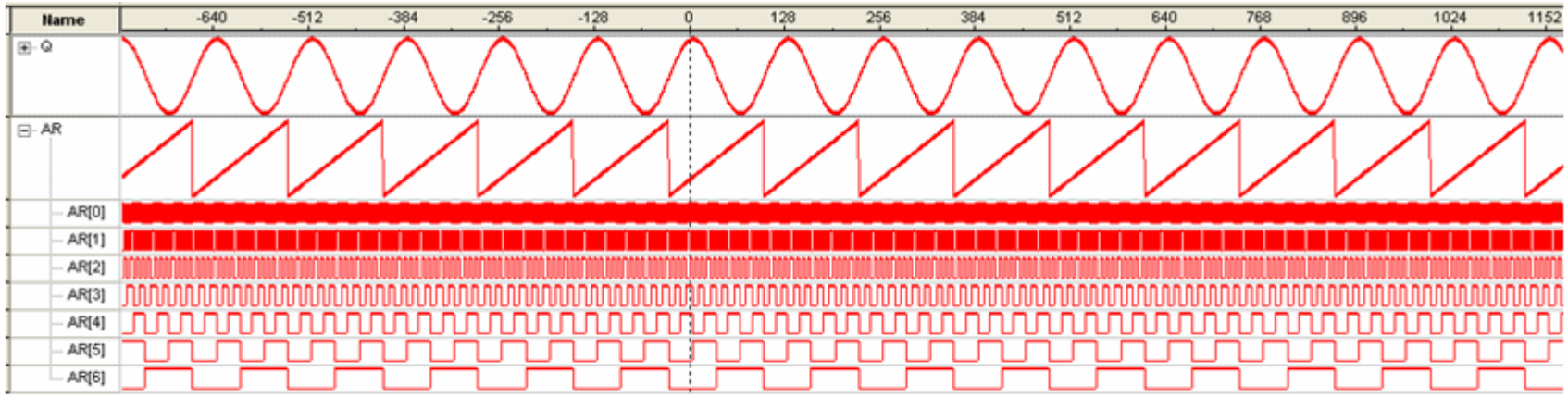


图 6-41 正弦信号发生器的 SignalTapII 的波形显示图

6.5 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

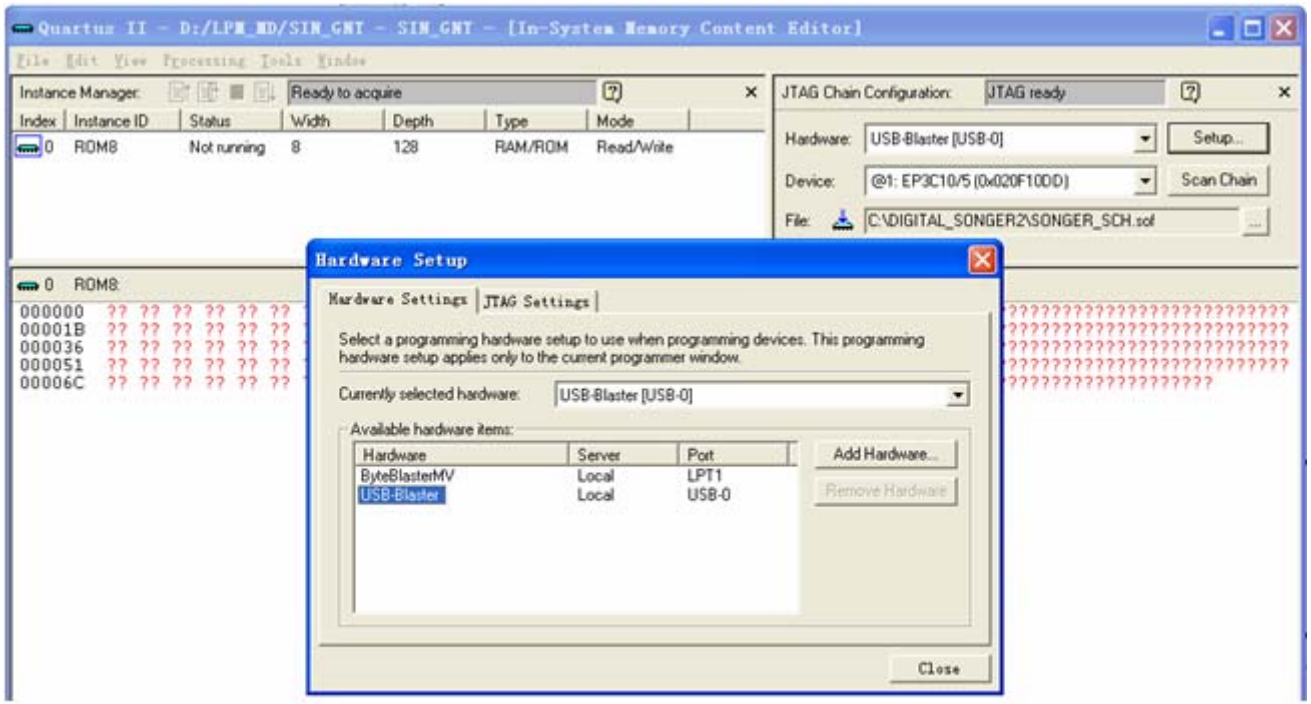


图 6-42 In-System Memory Content Editor 编辑窗

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

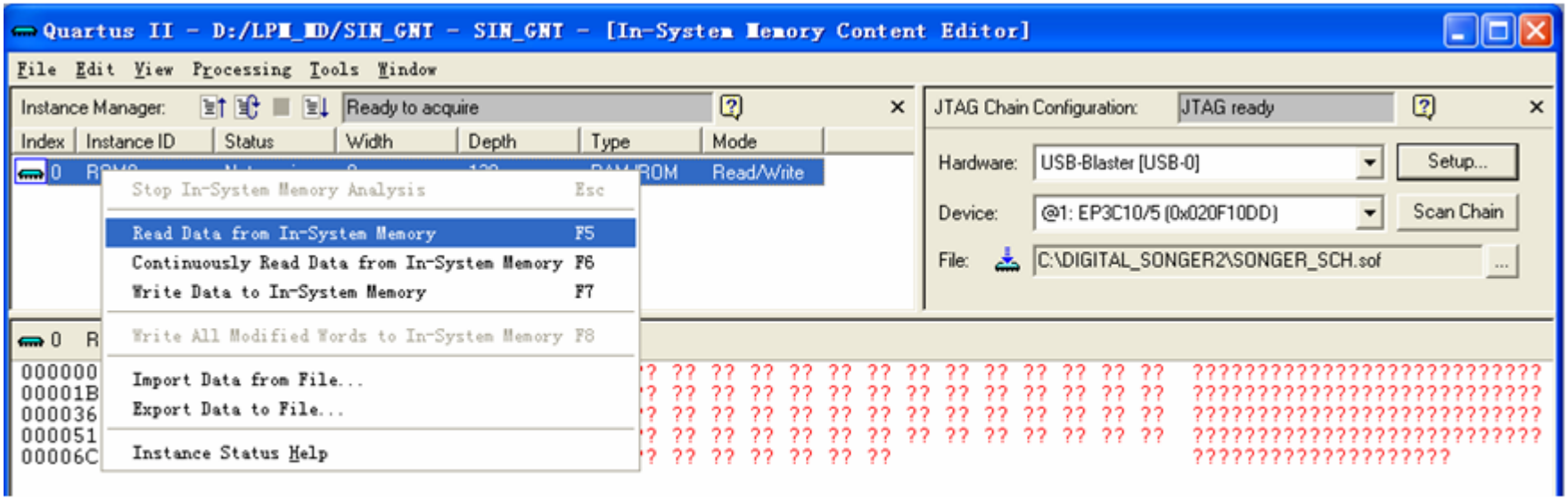


图 6-43　与实验系统上的 FPGA 通信正常情况下的编辑窗口界面

6.5 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

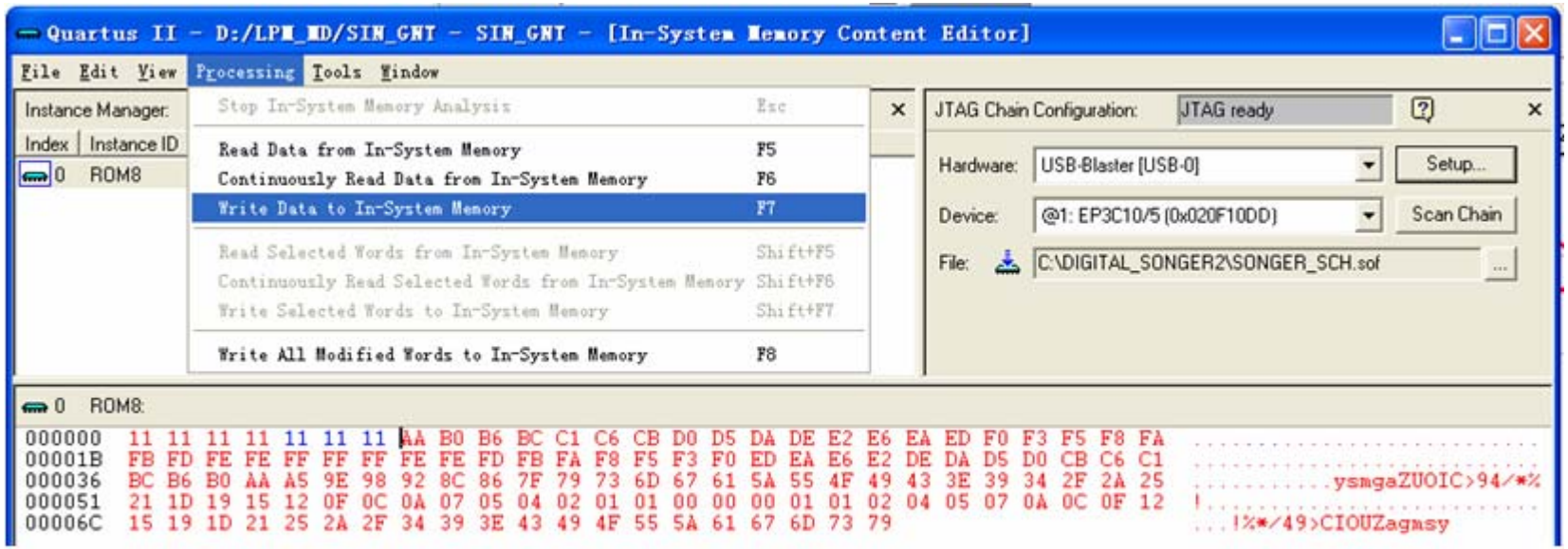


图 6-44　从 FPGA 中的 ROM 读取波形数据并编辑数据

6.5 在系统存储器数据读写编辑器应用

（3）写数据

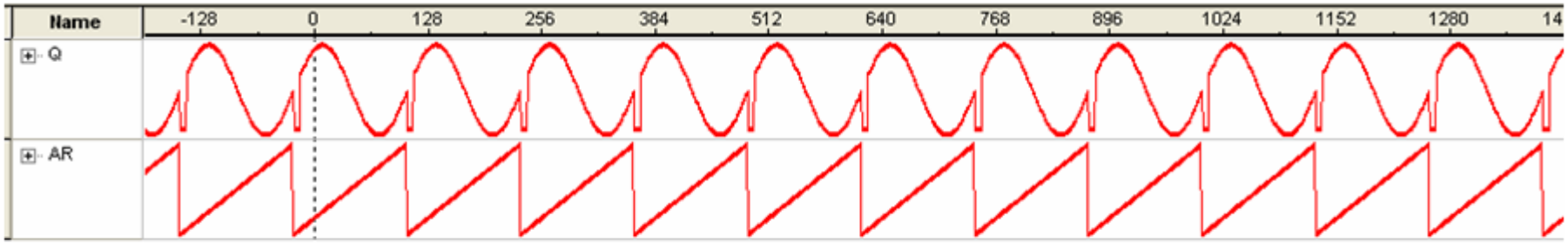


图 6-45　下载编辑数据后的 SignalTap II 采样波形

（4）输入输出数据文件

6.6 FIFO定制

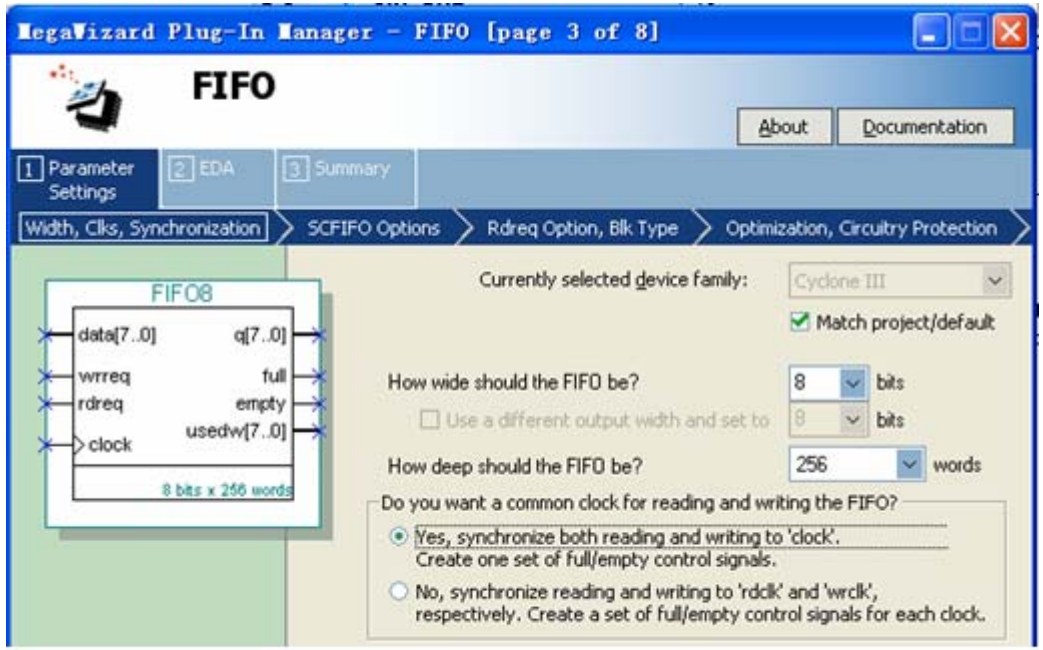


图 6-46 设置 FIFO 数据位宽和深度

6.6 FIFO定制

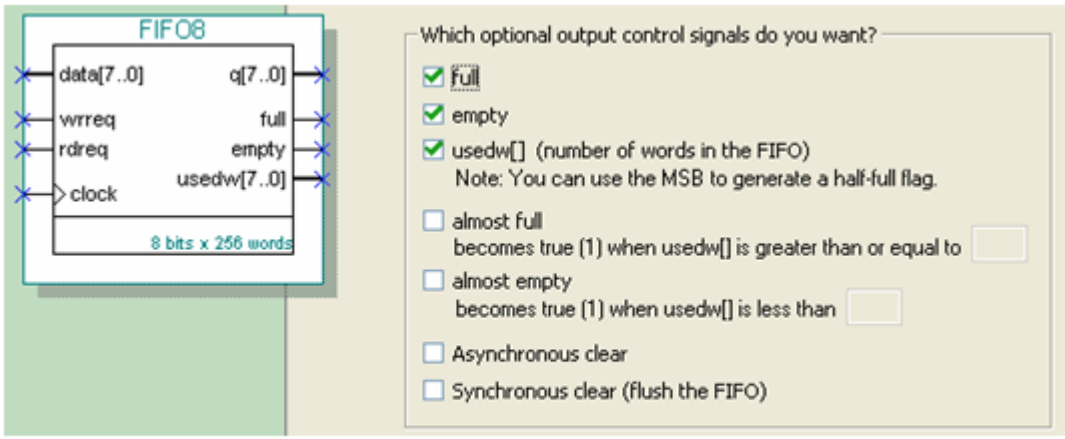


图 6-47 设置 FIFO 各种输出标志信号

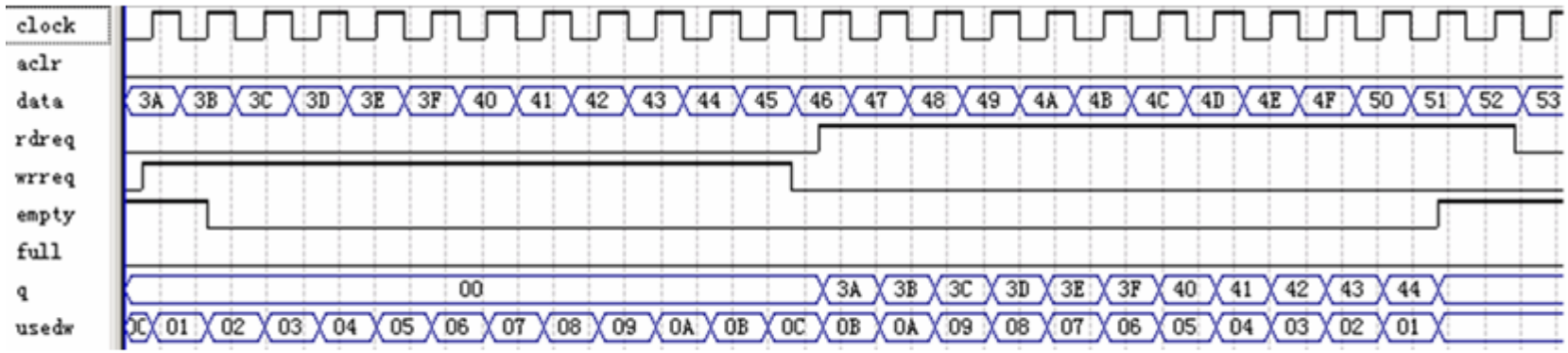


图 6-48 FIFO 的仿真波形

6.7 嵌入式锁相环ALTPLL调用

6.7.1 嵌入式锁相环参数设置

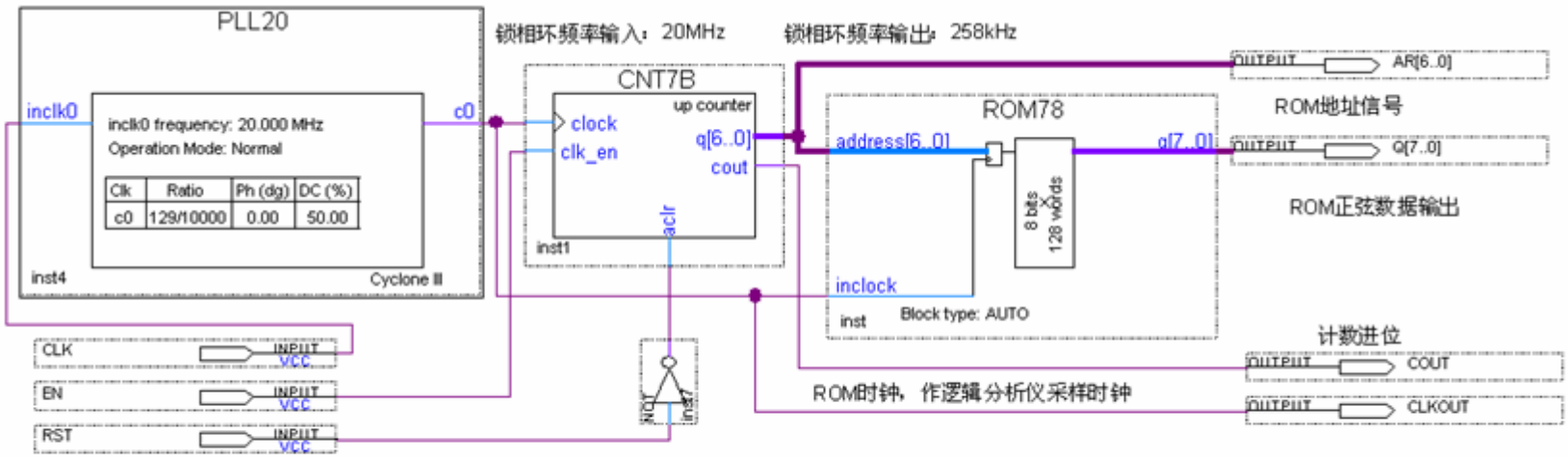


图 6-49　采用嵌入式锁相环作时钟的正弦信号发生器电路图

6.7 嵌入式锁相环ALTPLL调用

6.7.1 嵌入式锁相环参数设置

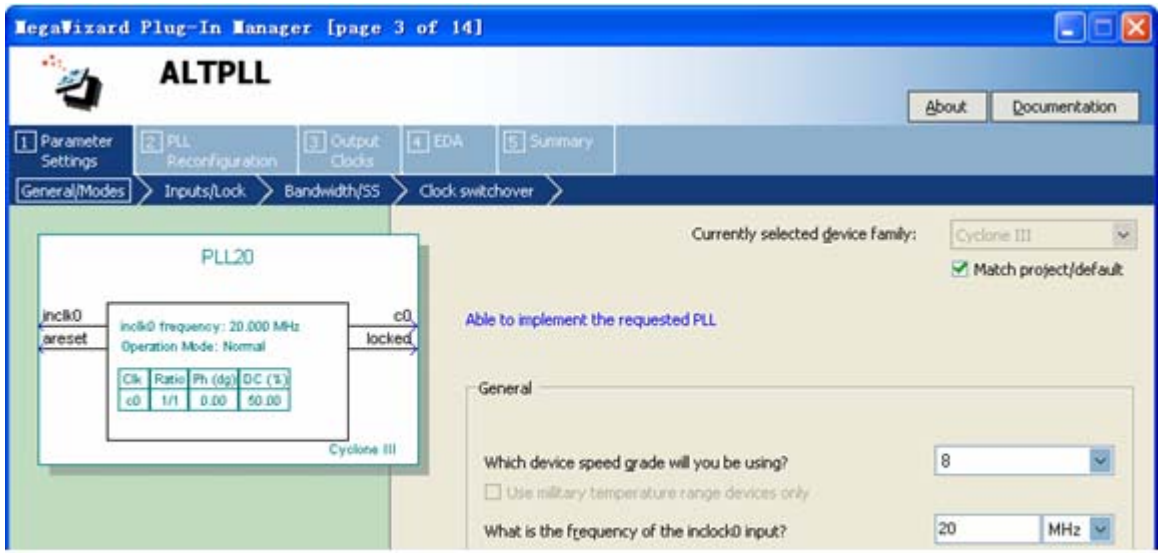


图 6-50 选择输入参考时钟 inclk0 为 20MHz

6.7 嵌入式锁相环ALTPLL调用

6.7.1 嵌入式锁相环参数设置

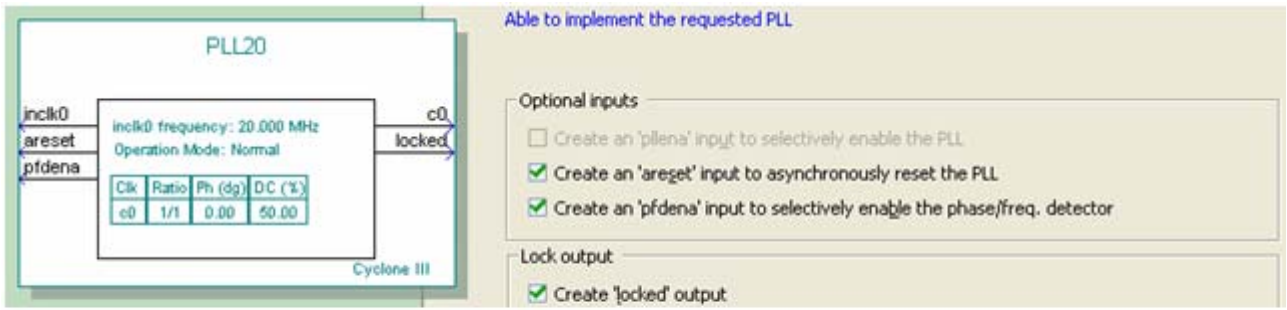


图 6-51　选择控制信号

6.7 嵌入式锁相环ALTPLL调用

6.7.1 嵌入式锁相环参数设置

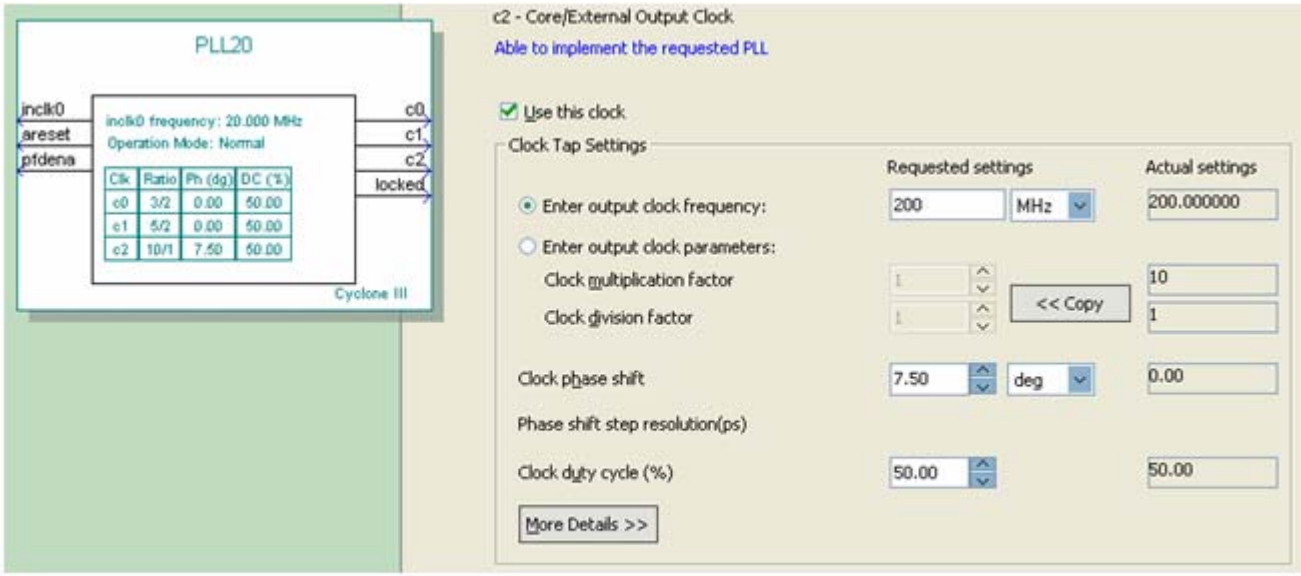


图 7-52 选择 e0 的输出频率为 200MHz

6.7 嵌入式锁相环ALTPLL调用

6.7.2 锁相环调用和测试的注意事项

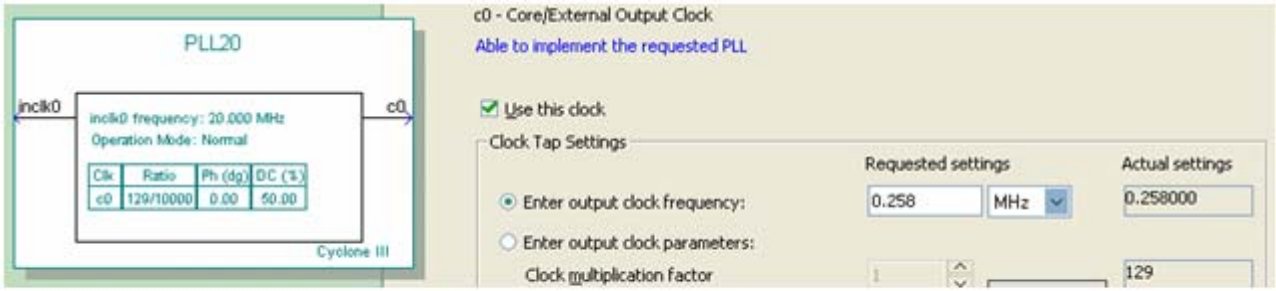


图 6-53 选择输出频率为 0.258MHz 作正弦信号发生器工作时钟

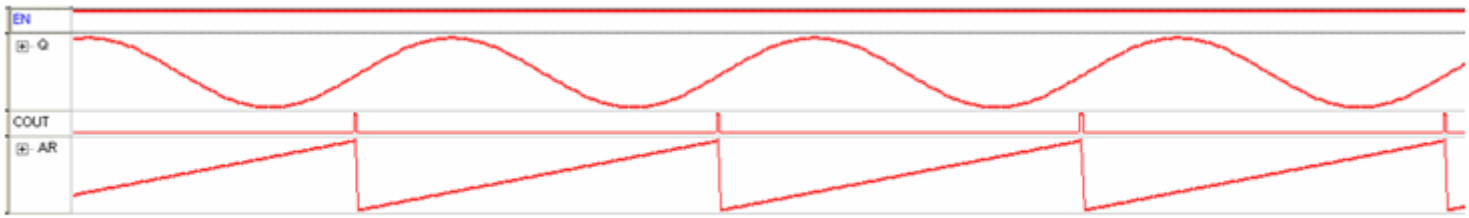


图 6-54 图 6-49 设计的逻辑分析仪实时采样输出

6.7 嵌入式锁相环ALTPLL调用

6.7.2 锁相环调用和测试的注意事项

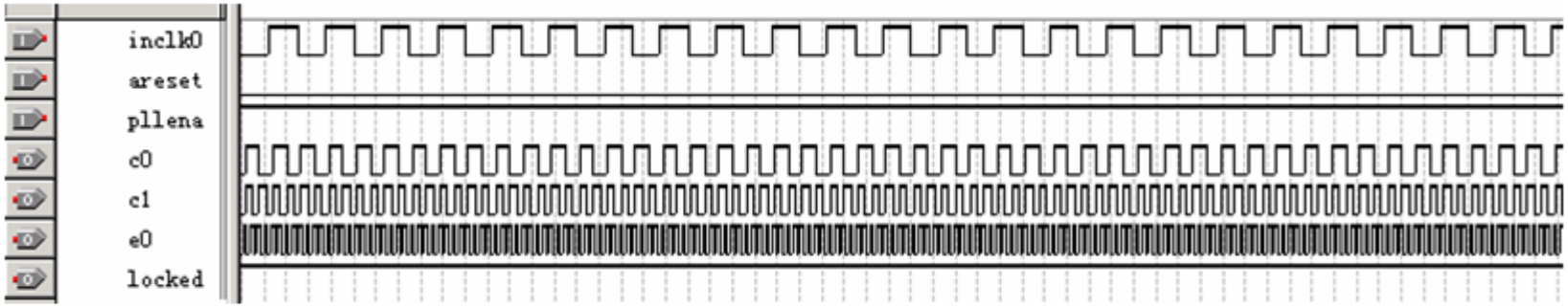


图 6-55　PLL 元件的仿真波形

6.8 数控振荡器核使用方法

（1）定制NCO

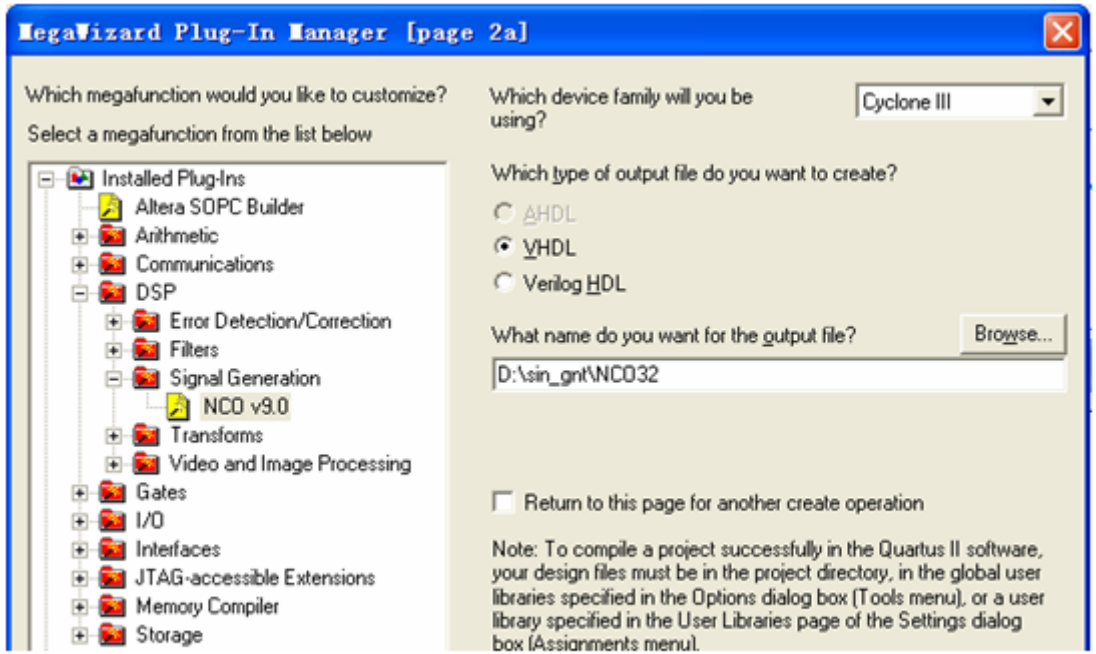


图 6-56 打开 Core 设置管理窗口选择 NCO 核

6.8 数控振荡器核使用方法

（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

6.8 数控振荡器核使用方法

（3）设置参数

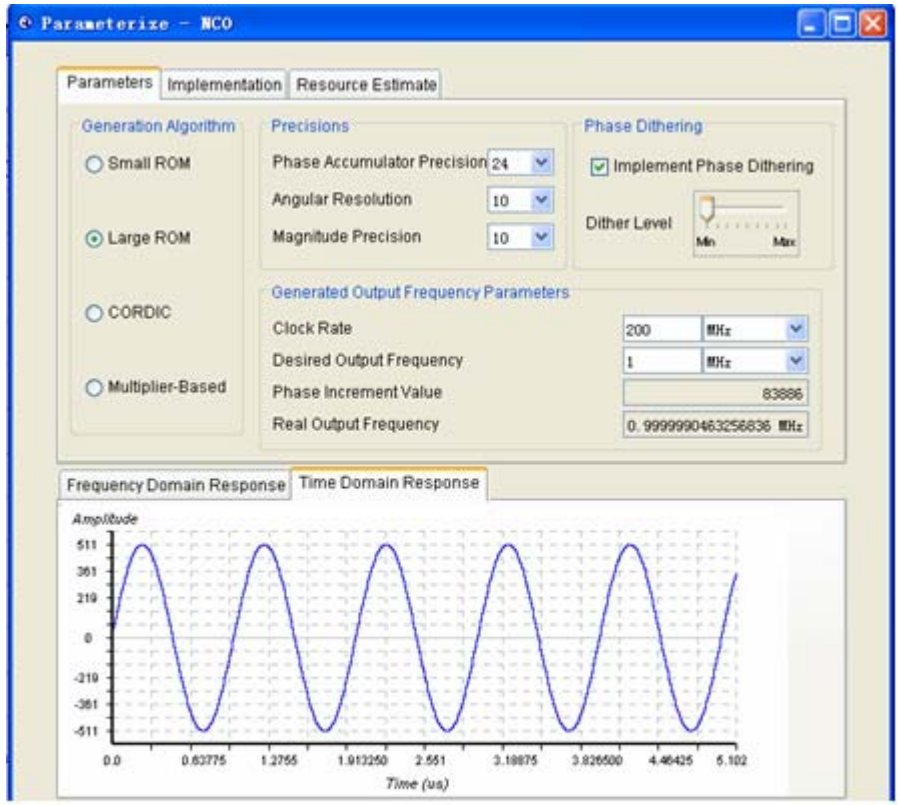


图 6-58　设置 NCO 参数

6.8 数控振荡器核使用方法

（3）设置参数

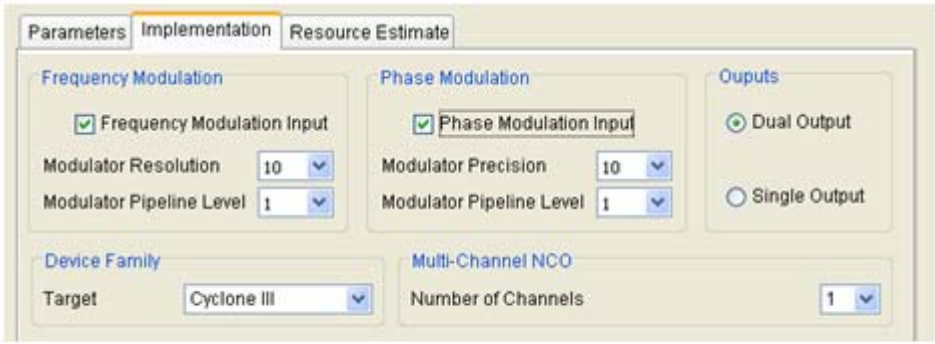


图 6-59 设置 NCO 参数

6.8 数控振荡器核使用方法

（4）生成仿真文件

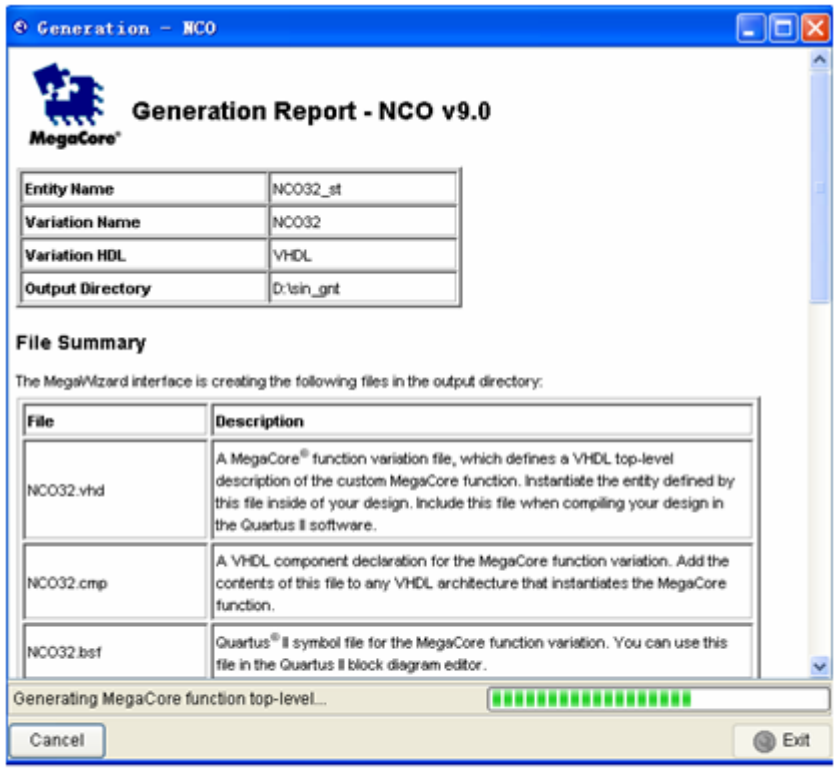


图 6-60 完成NCO 参数设置后的信息窗口

6.8 数控振荡器核使用方法

（5）加入IP授权文件

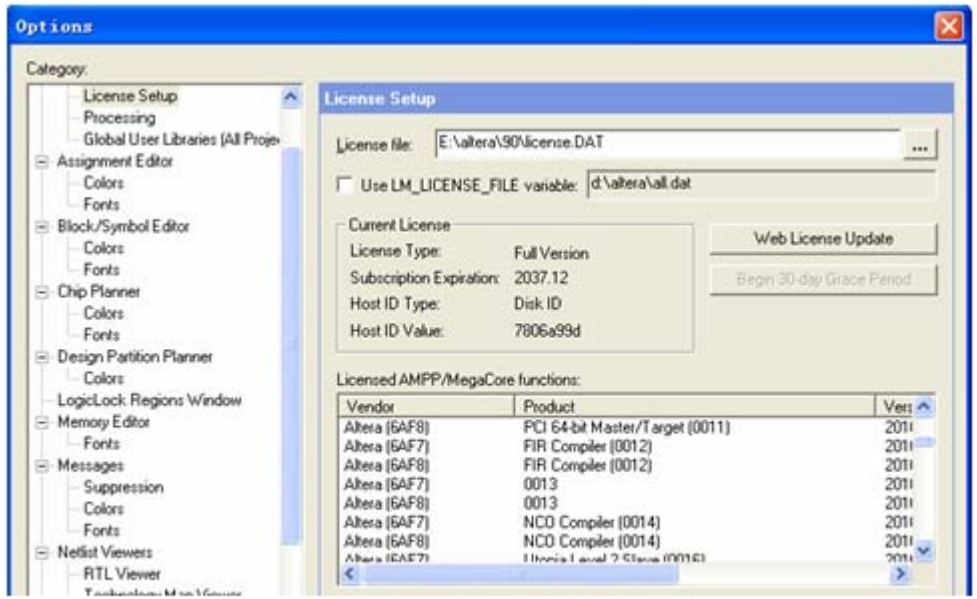


图 6-61　加入含有 NCO 等 IP 的授权文件

6.8 数控振荡器核使用方法

（6）选择目标器件，然后对生成的模块进行编译及功能检测

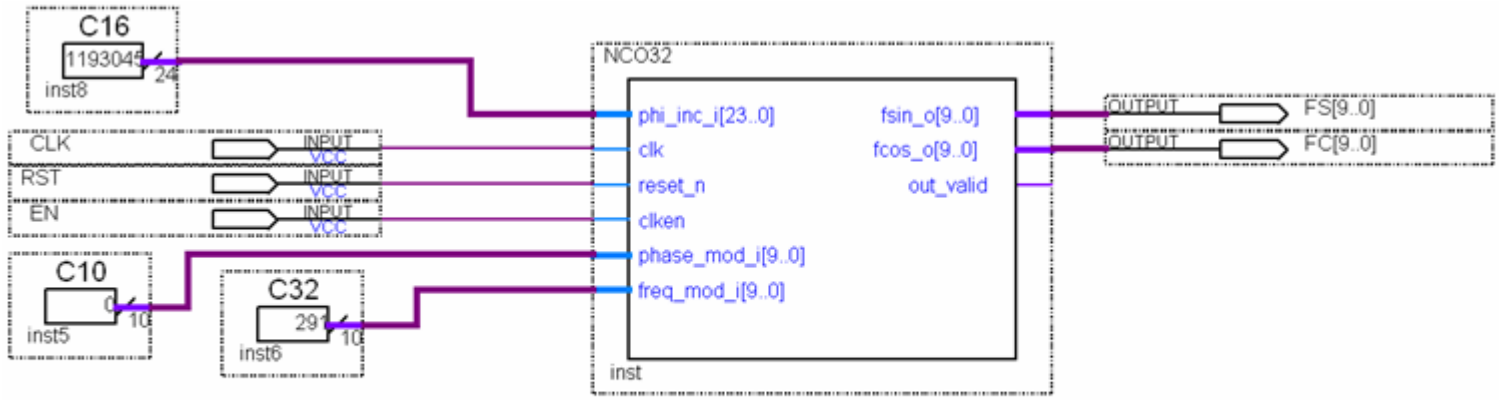


图 6-62 测试 NCO 的电路

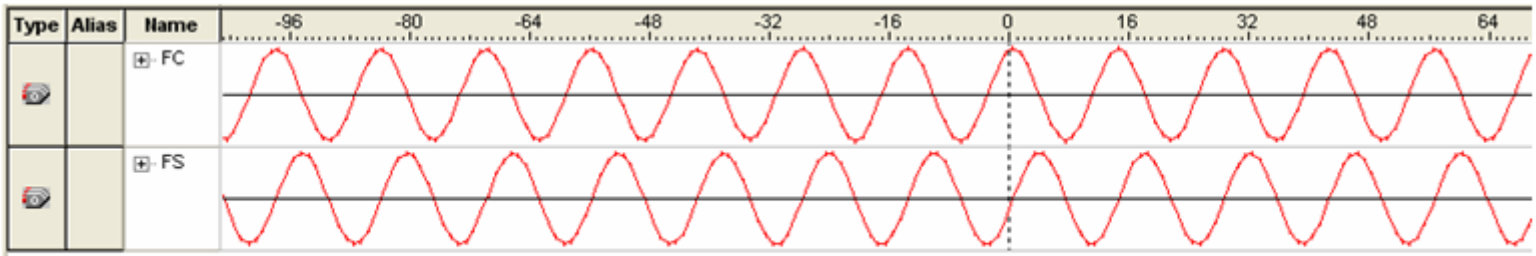


图 6-63 当前 NCO 的逻辑分析仪测试波形

6.9 FIR核使用方法

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

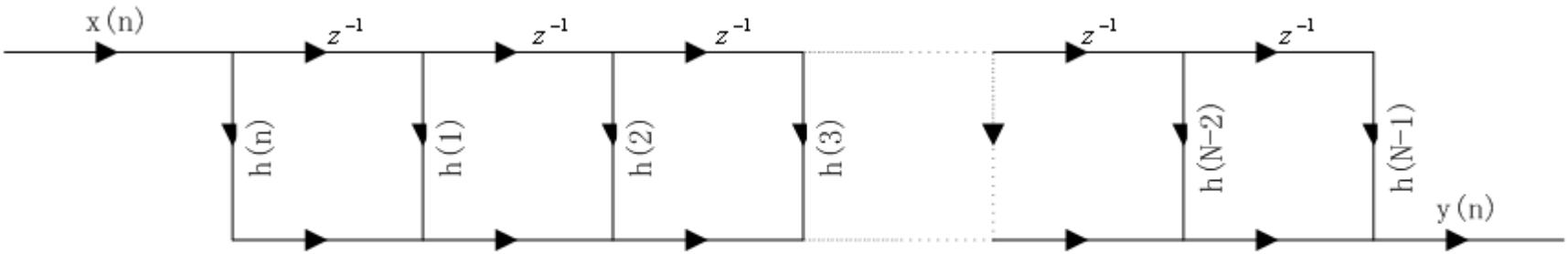


图 6-64 直接型 FIR 滤波器结构

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

6.9 FIR核使用方法

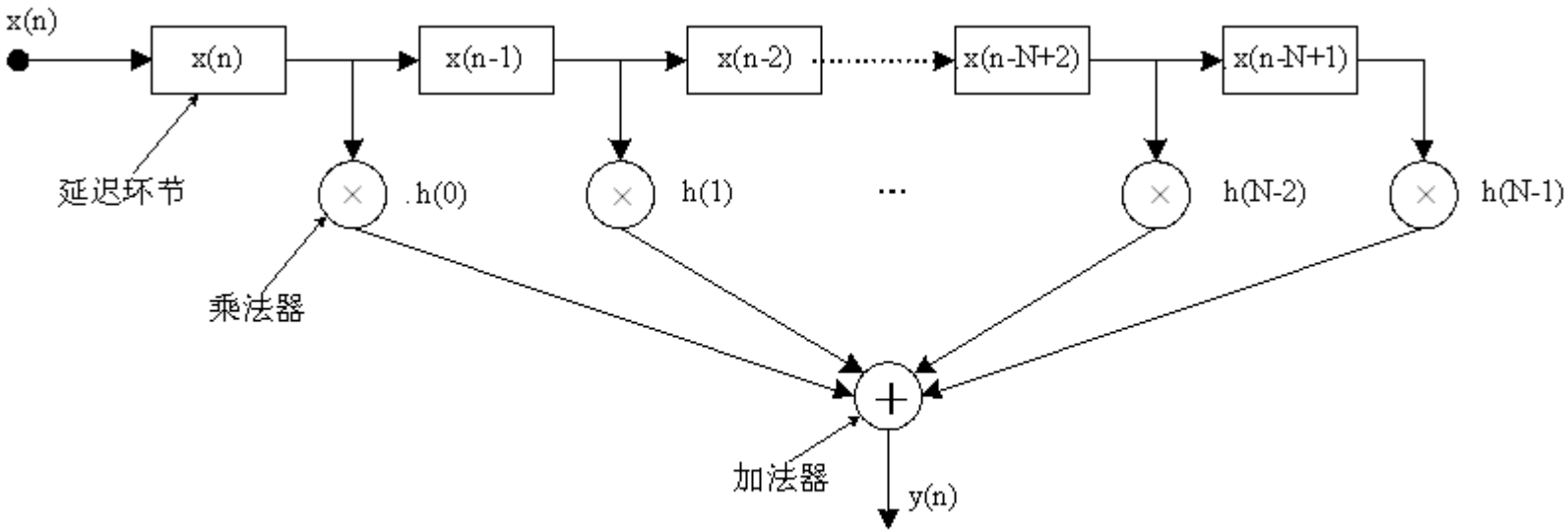


图 7-65 直接型 FIR 实现结构

图 6-66 FIR 滤波器设计示意

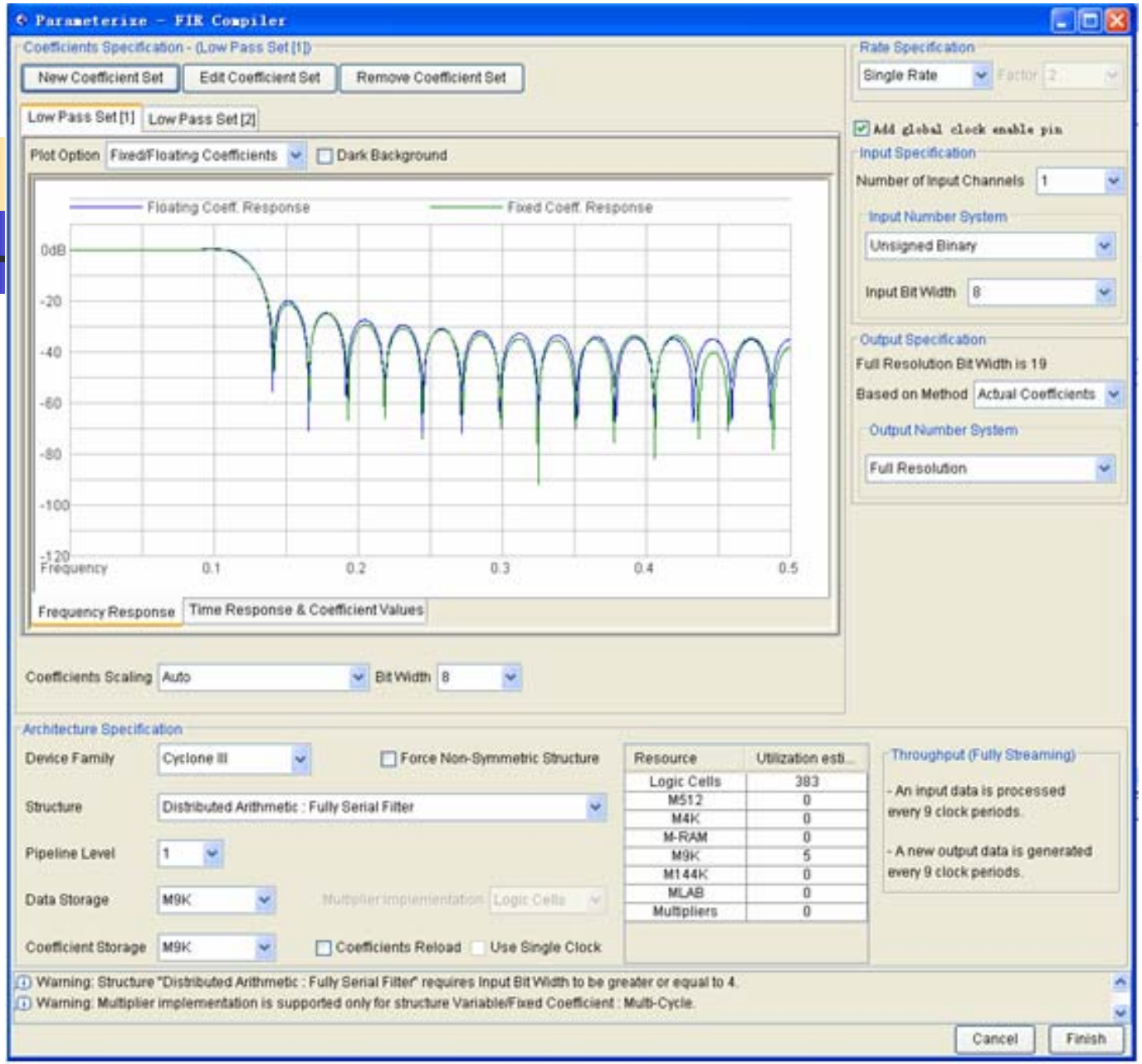
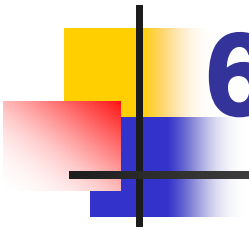


图 6-67　FIR 滤波器系数确定



6.10 单片机IP核应用

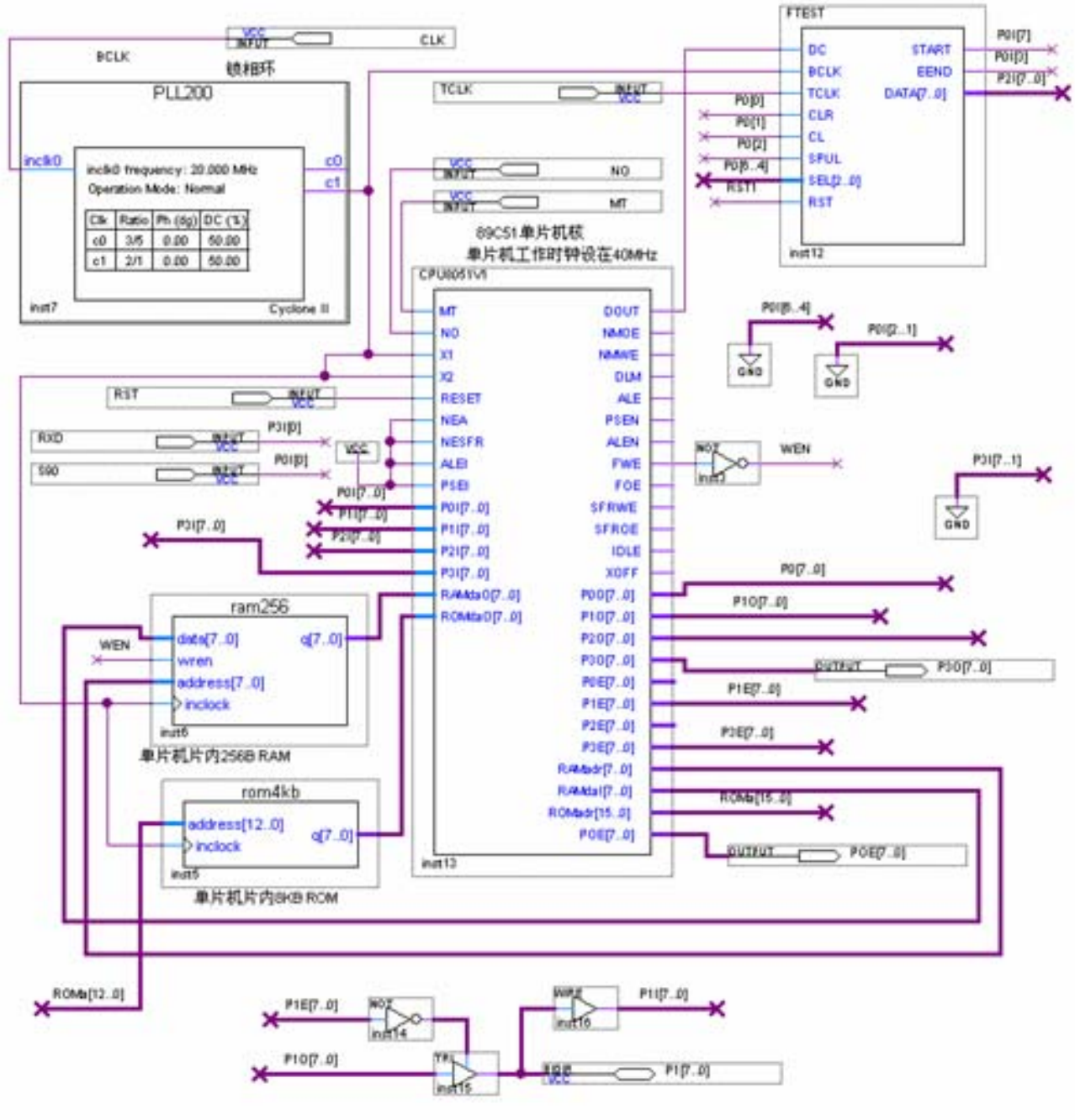


图 6-68 基本 8051CPU 核应用电路示例

6.10 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

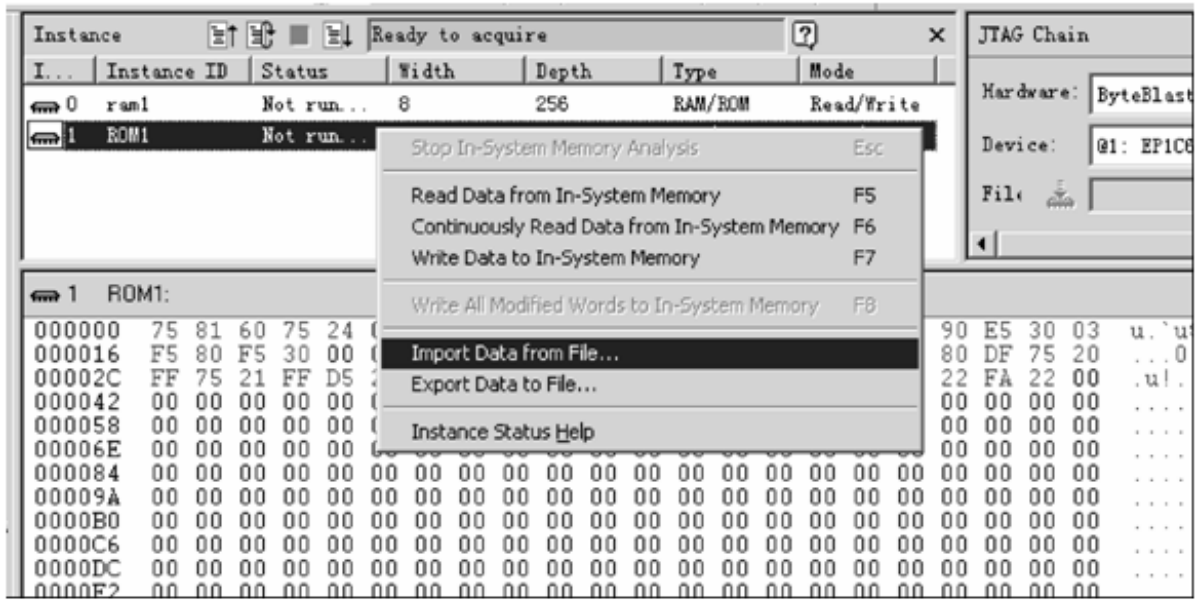


图 6-70　用 In-System Memory Content Editor 下载汇编程序代码

6.11 DDS实现原理与应用

6.11.1 DDS原理

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

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

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

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

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

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

6.11 DDS实现原理与应用

6.11.1 DDS原理

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

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

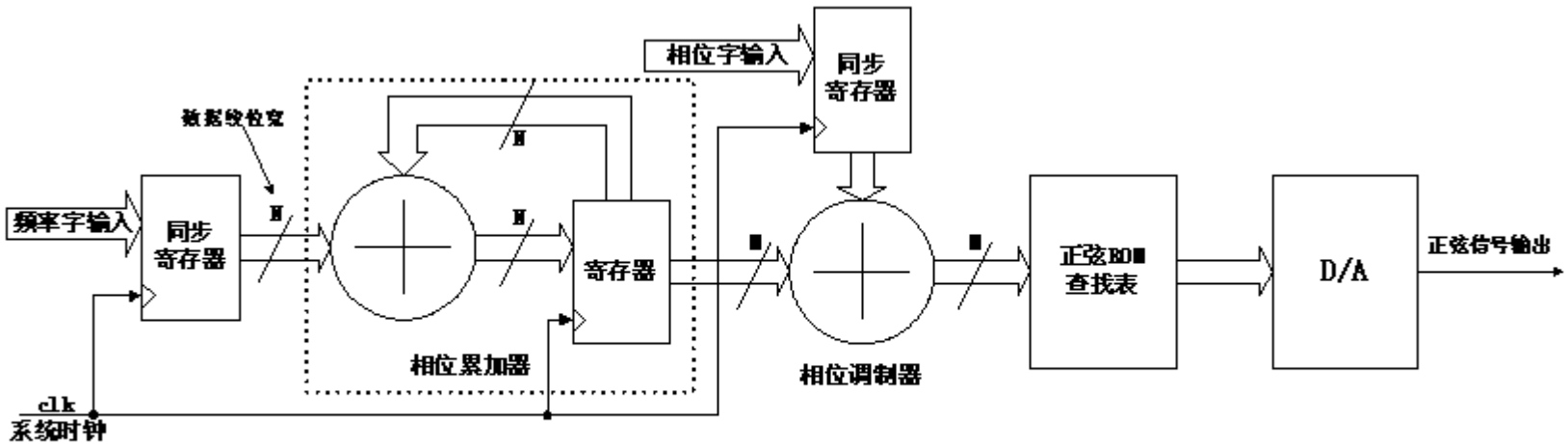


图 6-71 基本 DDS 结构

6.11 DDS实现原理与应用

6.11.2 DDS信号发生器设计

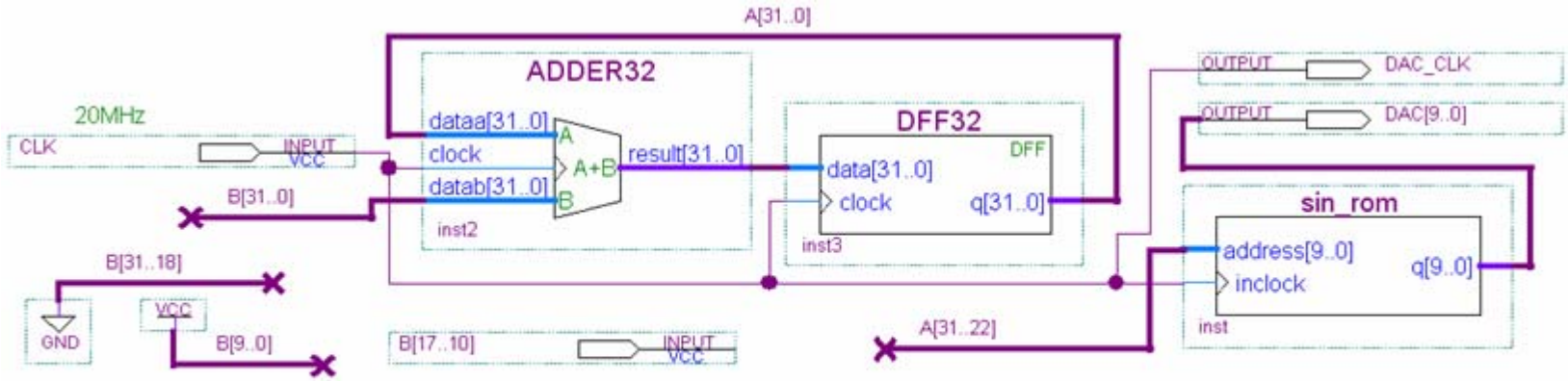


图 6-72 DDS 信号发生器电路顶层原理图

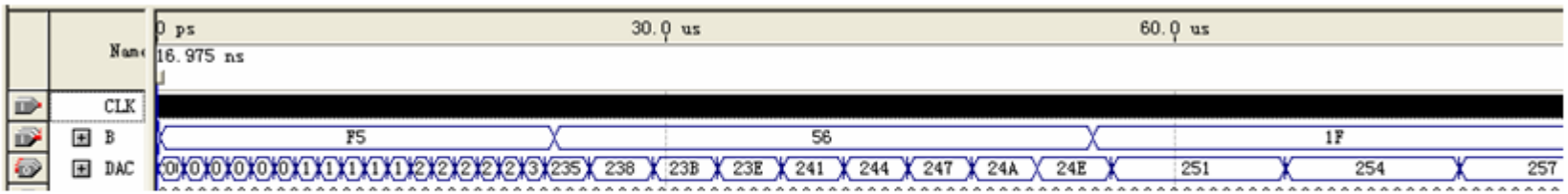


图 6-73 图 6-72 的仿真波形

实验与设计

6-1. 查表式硬件运算器设计

【例 6-12】

```
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

6-3 八位数码显示频率计设计

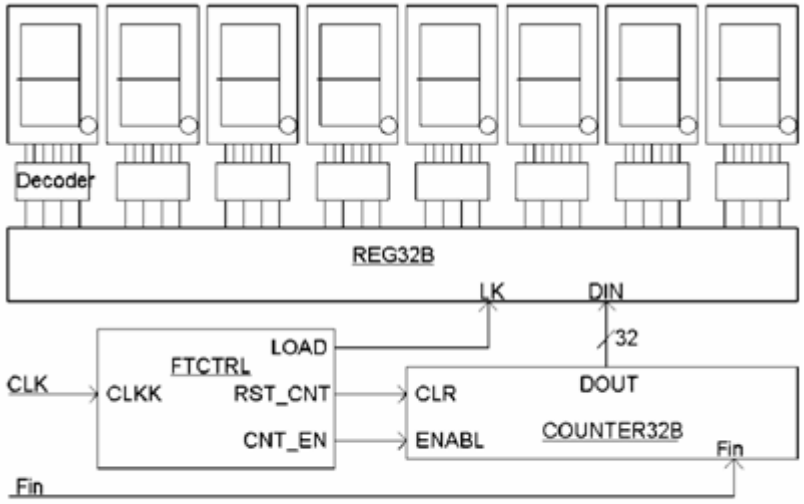


图 6-74 频率计电路框图

实验与设计

6-3 八位数码显示频率计设计

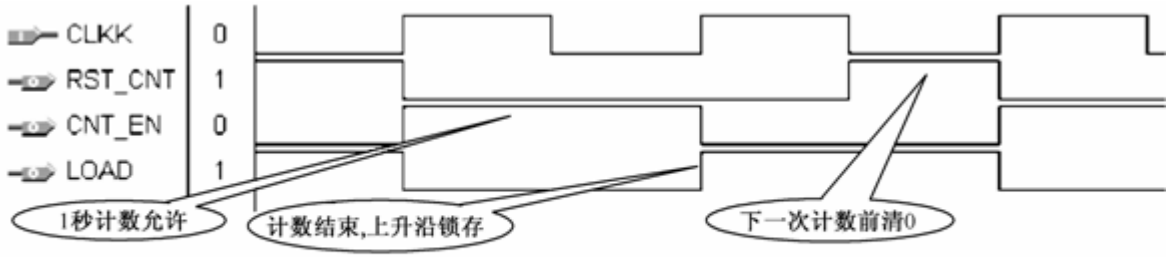


图 6-75 频率计测频控制器 FTCTRL 测控时序图

实验与设计

6-3 八位数码显示频率计设计

【例 6-13】

```
module FTCTRL (CLKK, CNT_EN, RST_CNT, LOAD);
   input CLKK;        output CNT_EN, RST_CNT,LOAD;
   wire CNT_EN, LOAD;        reg RST_CNT,Div2CLK;
   always @(posedge CLKK)
       Div2CLK <= ~Div2CLK ;
   always @(CLKK or Div2CLK)  begin
     if (CLKK==1'b0 & Div2CLK==1'b0)  RST_CNT <= 1'b1 ;
       else  RST_CNT <= 1'b0 ;       end
   assign LOAD = ~Div2CLK ;     assign CNT_EN = Div2CLK ;
endmodule
```

实验与设计

6-4. 简易逻辑分析仪设计

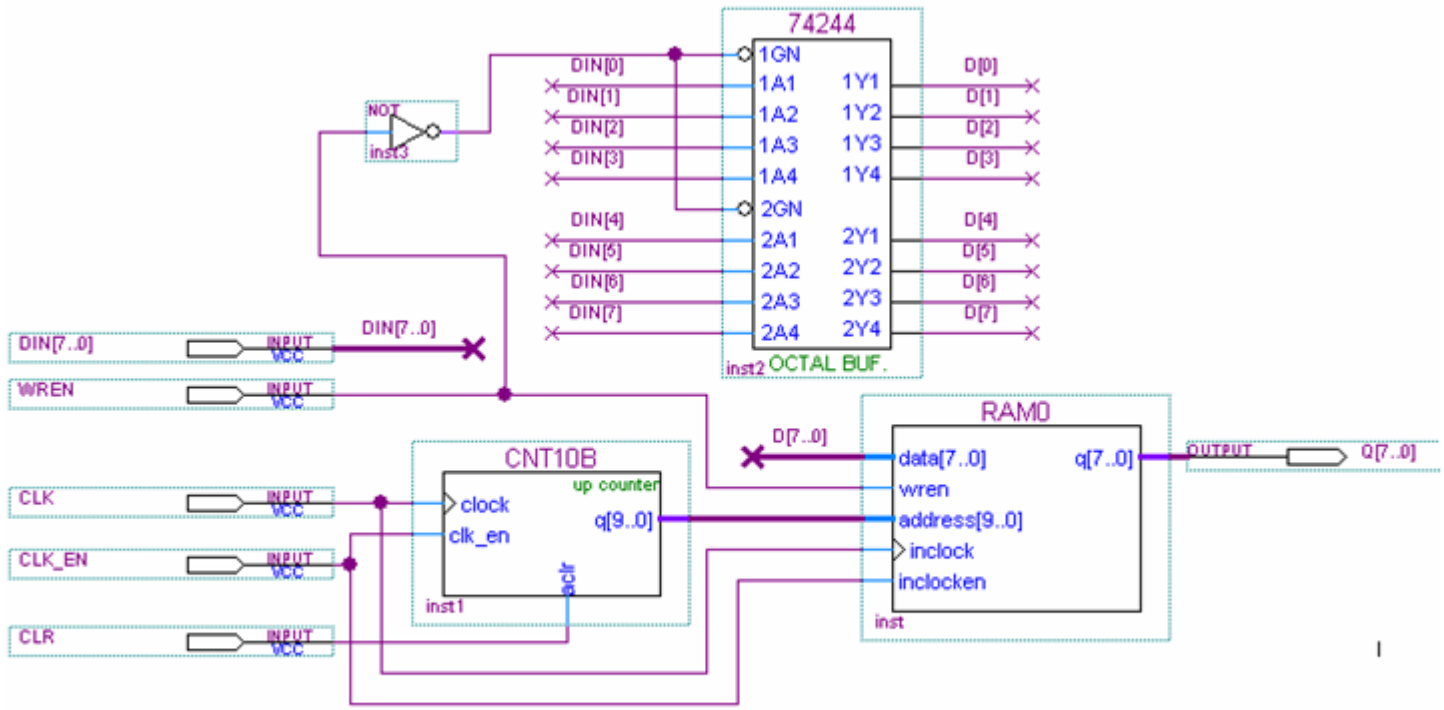


图 6-76 逻辑数据采样电路顶层设计

实验与设计

6-4. 简易逻辑分析仪设计

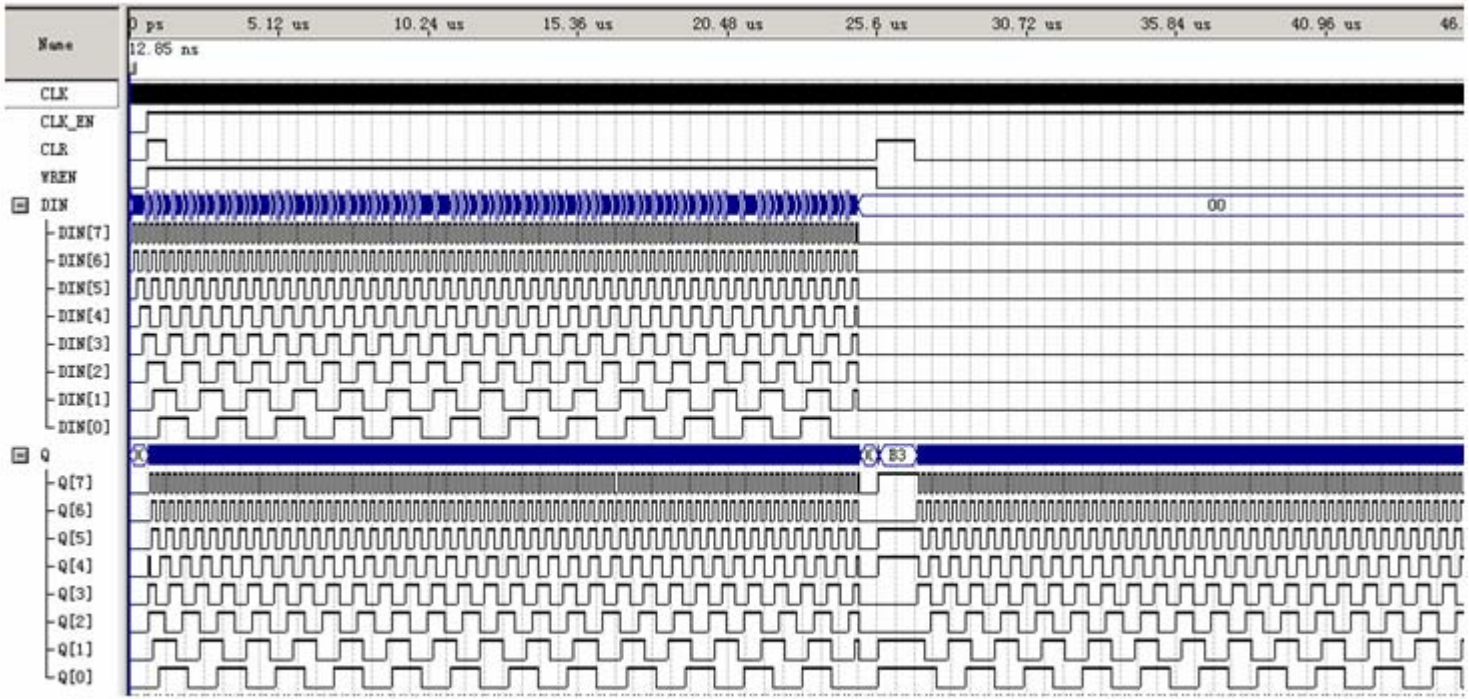


图 6-77 逻辑数据采样电路时序仿真波形

实验与设计

6-6 移相信号发生器设计

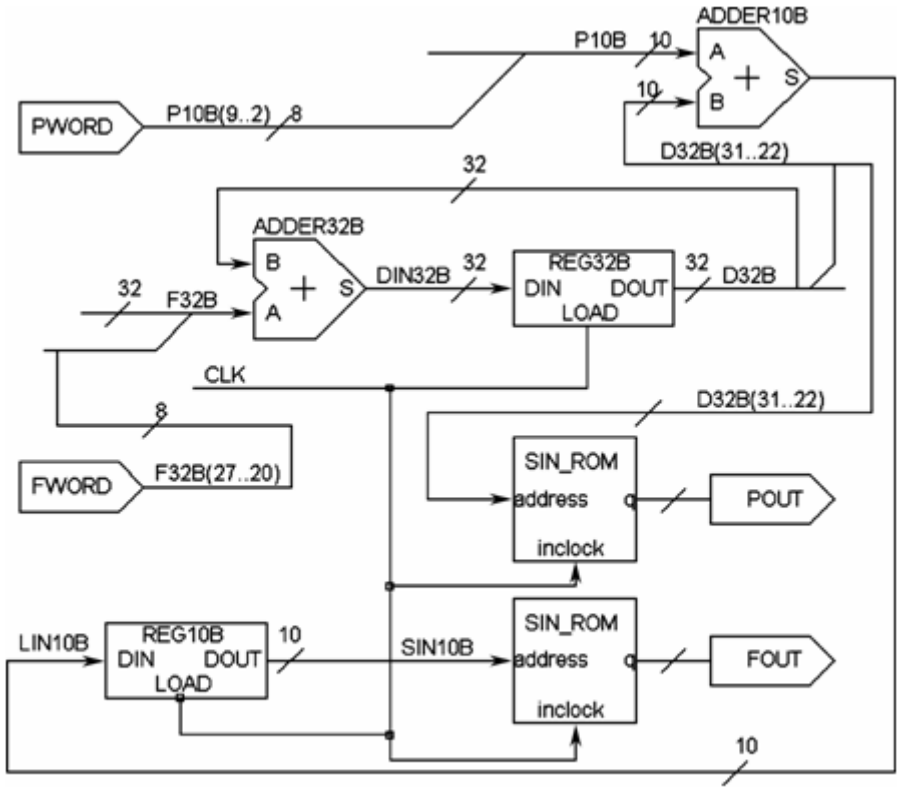


图 6-78 数字移相信号发生器电路模型图

实验与设计

6-7 4X4阵列键盘键信号检测电路设计

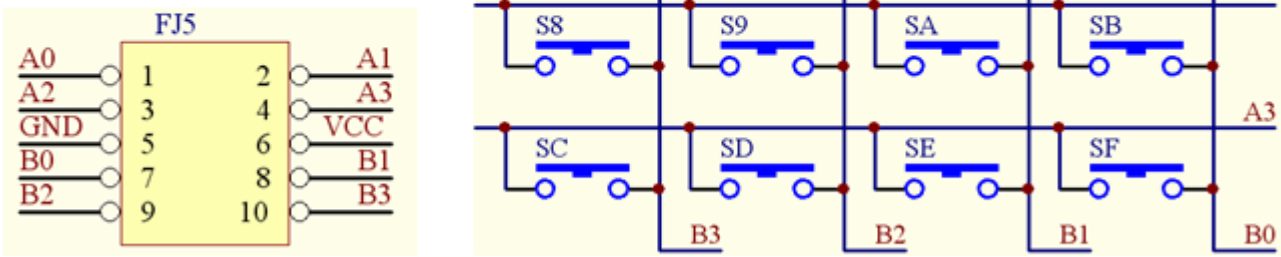


图 6-79 4X4 键盘电路和 10 芯接口

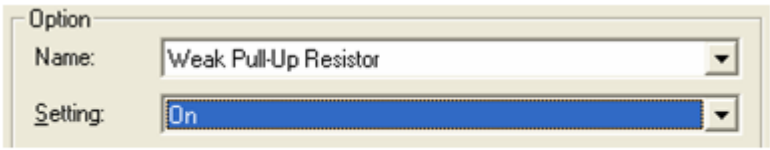


图 6-80 设置端口上拉

【例 6-14】

```
module KEY4X4 (CLK,A,B,R);
   (* chip_pin = "125" *)  input CLK;
   (* chip_pin = "76, 75, 74, 73" *)  input [3:0] A;
   (* chip_pin = "83, 80, 79, 77" *) output [3:0] B;
   (* chip_pin = "34, 38, 39, 42" *) output [3:0] R;
      reg [1:0] C ;       reg [3:0] R,B ;
     always @ (posedge CLK) begin
       C<=C+1;
        case(C)
      0: B=4'B0111; 1: B=4'B1011; 2: B=4'B1101; 3: B=4'B1110;
        endcase
        case({B,A} )
      8'B0111_1110 : R=4'H0;   8'B0111_1101 : R=4'H1;
      8'B0111_1011 : R=4'H2;   8'B0111_0111 : R=4'H3;
      8'B1011_1110 : R=4'H4;   8'B1011_1101 : R=4'H5;
      8'B1011_1011 : R=4'H6;   8'B1011_0111 : R=4'H7;
      8'B1101_1110 : R=4'H8;   8'B1101_1101 : R=4'H9;
      8'B1101_1011 : R=4'HA;   8'B1101_0111 : R=4'HB;
      8'B1110_1110 : R=4'HC;   8'B1110_1101 : R=4'HD;
      8'B1110_1011 : R=4'HE;   8'B1110_0111 : R=4'HF;
   endcase
  end
endmodule
```

实验与设计

6-8 VGA简单图像显示控制模块设计

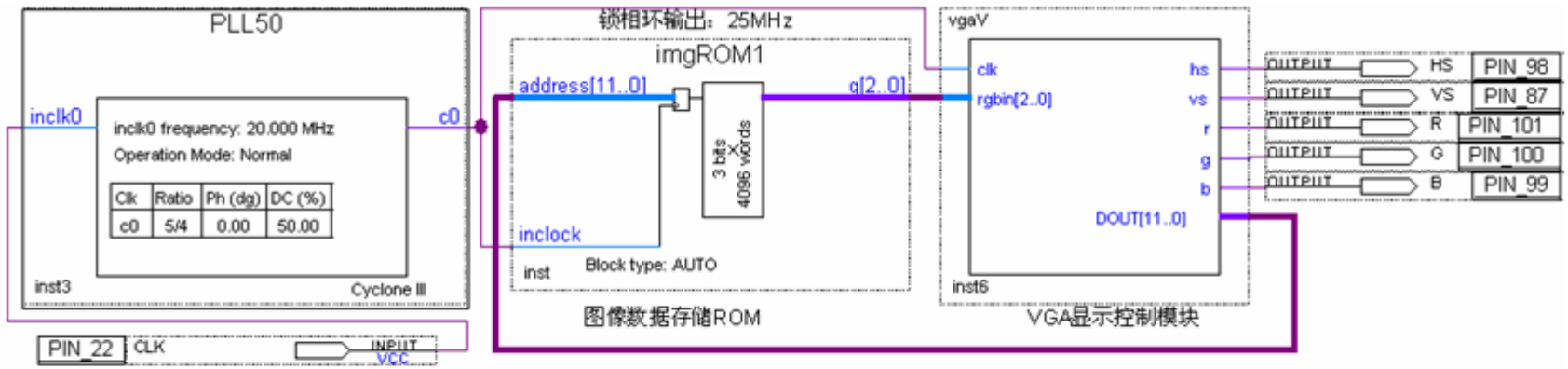
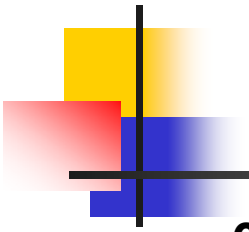


图 6-81 VGA 图像显示控制模块原理图

【例 6-15】

```
module vgaV (clk, hs, vs, r, g, b, rgbin, DOUT);//VGA
   input clk ;     //工作时钟25MHz
   output hs,vs;   //场同步，行同步信号
   output r,g,b ;  // 红，绿，蓝信号，
   input[2:0] rgbin; //像素数据
   output[11:0] DOUT; //图像数据ROM的地址信号
   reg[9:0] hcnt, vcnt;      reg r,g,b;      reg hs,vs;
   assign DOUT = {vcnt[5:0], hcnt[5:0]} ;
   always @(posedge clk)  begin  //水平扫描计数器
      if (hcnt<800)   hcnt<=hcnt+1 ;
      else            hcnt<={10{1'b0}} ;
      end
   always @(posedge clk)   begin  //垂直扫描计数器
      if (hcnt==640+8)    begin
         if (vcnt<525)  vcnt<=vcnt+1 ;
         else    vcnt<={10{1'b0}} ;  end   end
   always @(posedge clk)   begin //场同步信号发生
      if ((hcnt>=640+8+8) & (hcnt<640+8+8+96))
        hs<=1'b0 ; else  hs<=1'b1 ;    end
   always @(vcnt)   begin   //行同步信号发生
      if ((vcnt>=480+8+2) & (vcnt<480+8+2+2))
        vs<=1'b0 ;  else  vs<=1'b1 ;  end
   always @(posedge clk)   begin
      if (hcnt<640 & vcnt<480)   //扫描终止
      begin  r<=rgbin[2] ;  g<=rgbin[1] ;  b<=rgbin[0];   end
      else begin  r<=1'b0; g<=1'b0;  b<=1'b0;  end
   end    endmodule
```



实验与设计

6-8 VGA简单图像显示控制模块设计