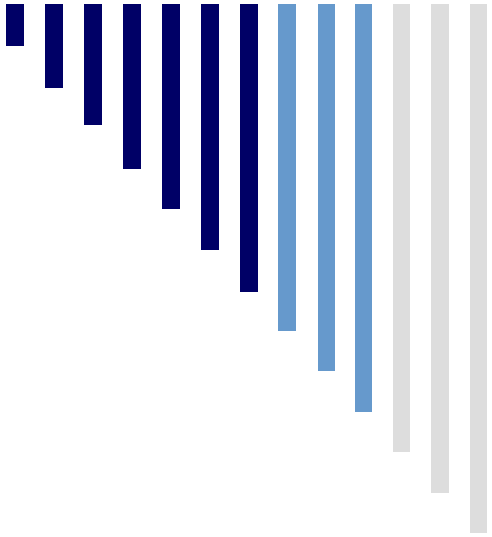

第5章

逻辑宏功能模块的应用

5.1 计数器宏模块调用

5.1.1 计数器模块文本的调用

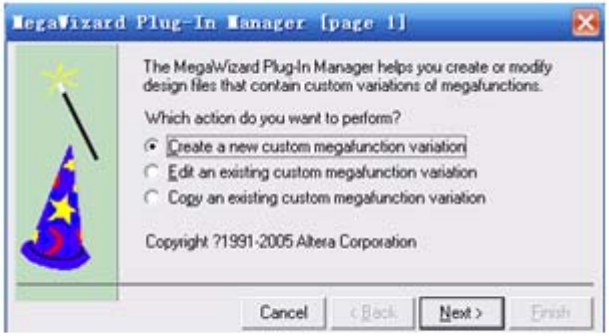

图 5-1　定制新的宏功能块

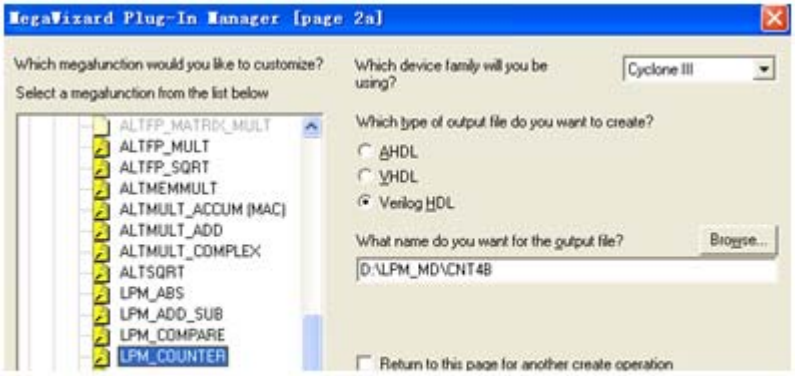

图 5-2　LPM 宏功能块设定

5.1 计数器宏模块调用

5.1.1 计数器模块文本的调用

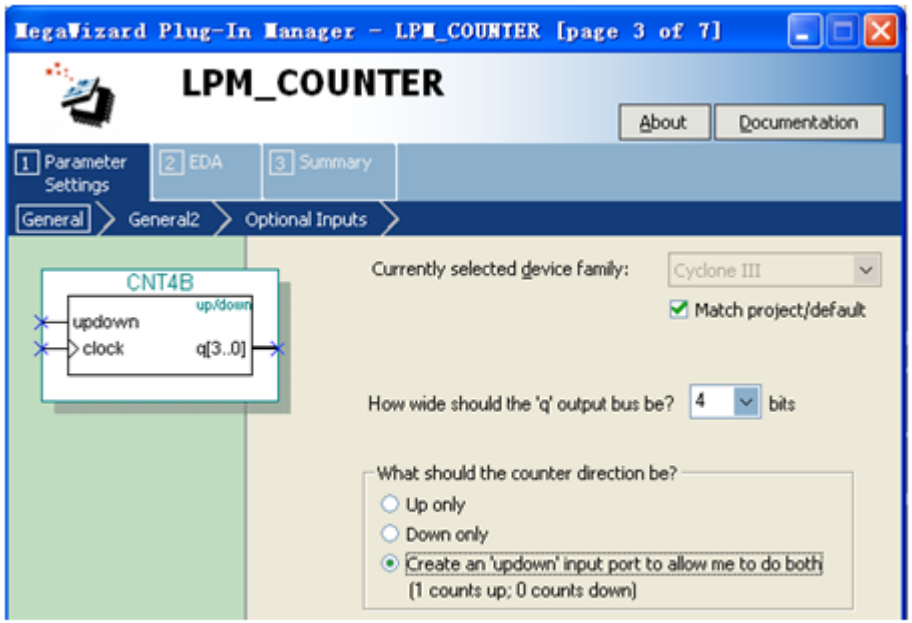

图 5-3 设 4 位可加减计数器

5.1 计数器宏模块调用

5.1.1 计数器模块文本的调用

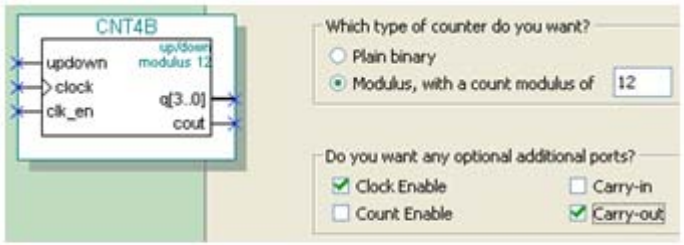

图 5-4 设定计数器，含时钟使能和进位输出

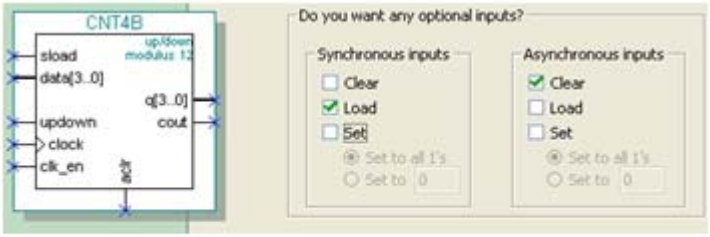

图 5-5 加入 4 位并行数据预置功能

【例 5-1】

```
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
   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
```

5.1 计数器宏模块调用

5.1.2 计数器模块程序与参数传递语句

【例 5-2】

```
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
```

5.1 计数器宏模块调用

5.1.2 计数器模块程序与参数传递语句

【例 5-3】

```
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
```

5.1 计数器宏模块调用

5.1.3 对计数器进行仿真测试

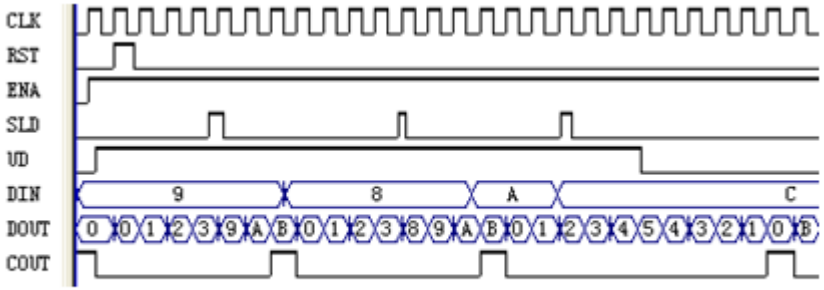

图 5-6 CNT4BIT.v 的仿真波形

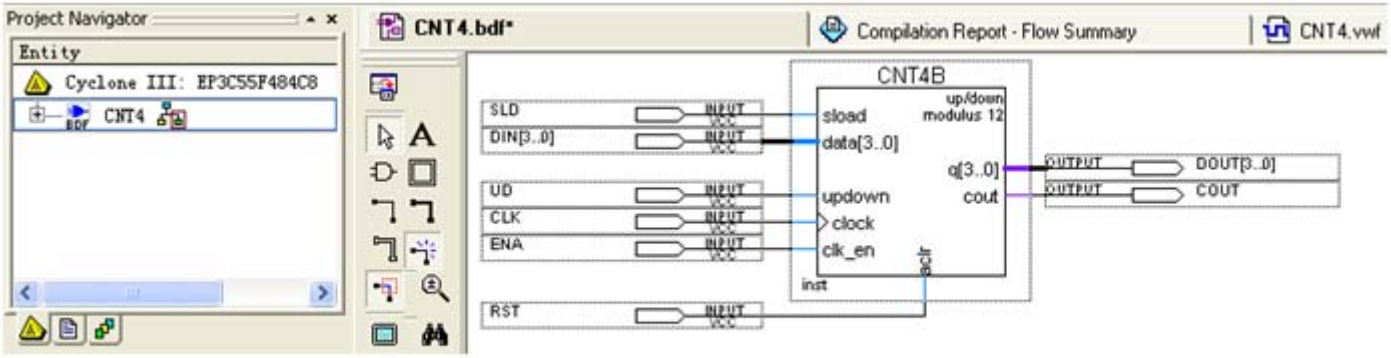

图 5-7 原理图输入设计

5.2 利用属性设置控制乘法器的构建

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

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

【例 5-4】

```
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
```

5.2 利用属性设置控制乘法器的构建

Family	Cyclone III
Device	EP3C55F484C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	0 / 55,856 (0 %)
Total combinational functions	0 / 55,856 (0 %)
Dedicated logic registers	0 / 55,856 (0 %)
Total registers	0
Total pins	64 / 328 (20 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	2 / 312 (< 1 %)
Total PLLs	0 / 4 (0 %)

图 5-8　编译报告

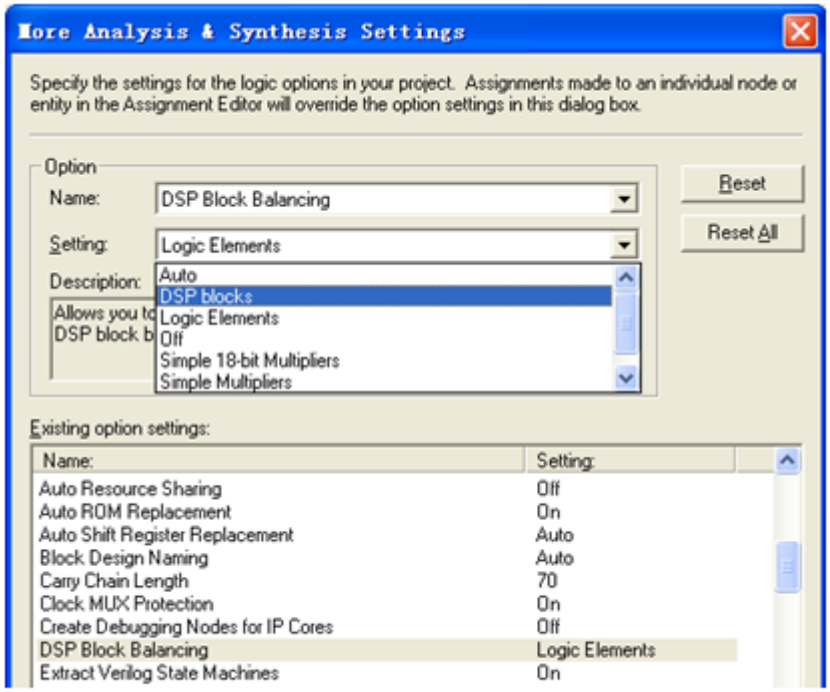

图 5-9　设置乘法器用 DSP 模块构建

5.3 RAM宏模块的使用方法

5.3.1 存储器初始化文件

【例 5-5】

```
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;
```

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 5-10 mif 文件编辑窗

5.3 RAM宏模块的使用方法

5.3.1 存储器初始化文件

【例 5-6】

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

5.3 RAM宏模块的使用方法

5.3.1 存储器初始化文件

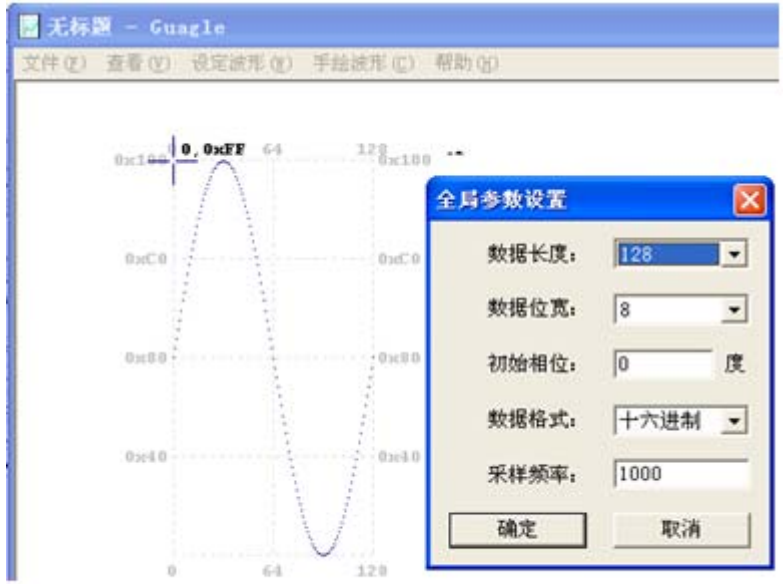

图 5-11 利用 mif 生成器生成 mif 正弦波文件

图 5-12 打开 mif 文件

5.3 RAM宏模块的使用方法

5.3.2 RAM宏模块的设置和调用

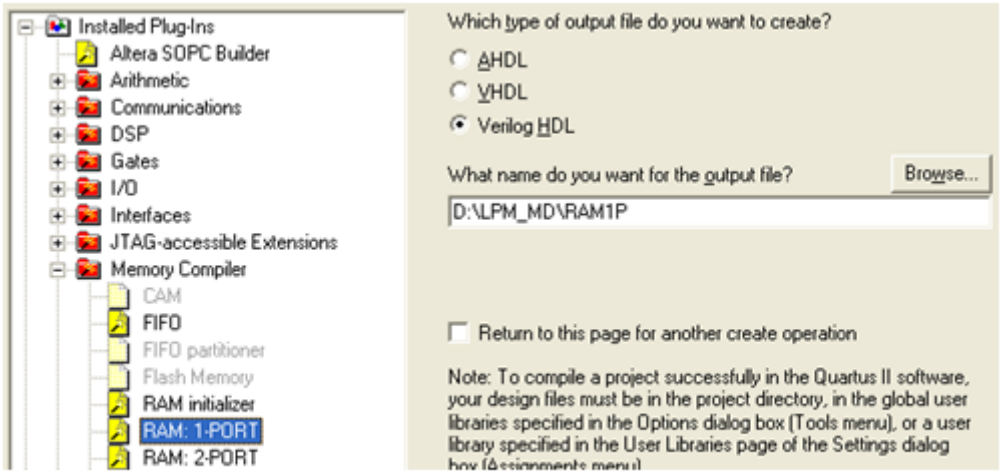

图 5-13　调用单口 LPM RAM

5.3 RAM宏模块的使用方法

5.3.2 RAM宏模块的设置和调用

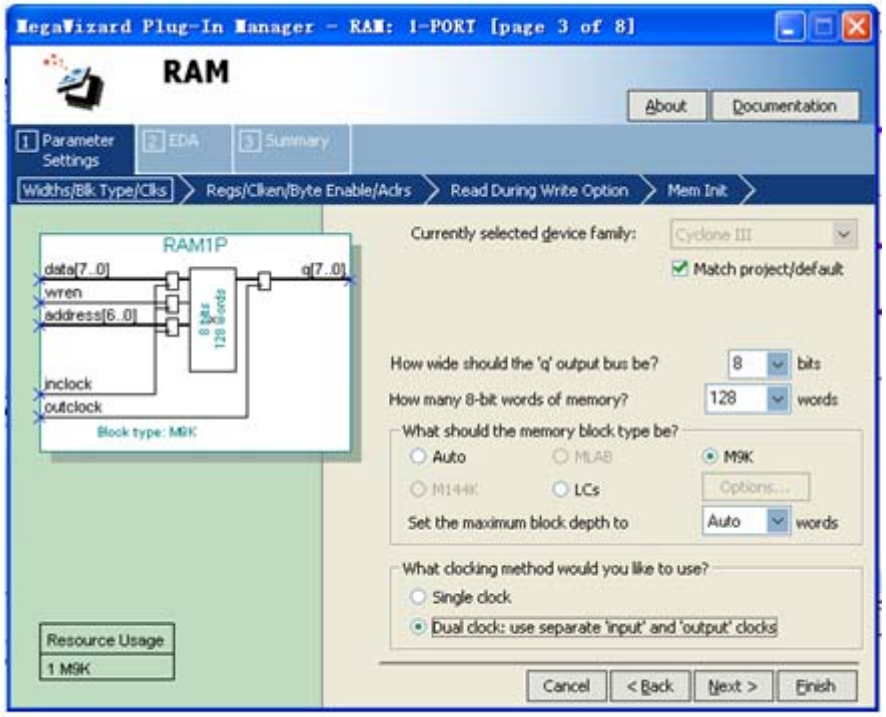

图 5-14 设定 RAM 参数

5.3 RAM宏模块的使用方法

5.3.2 RAM宏模块的设置和调用

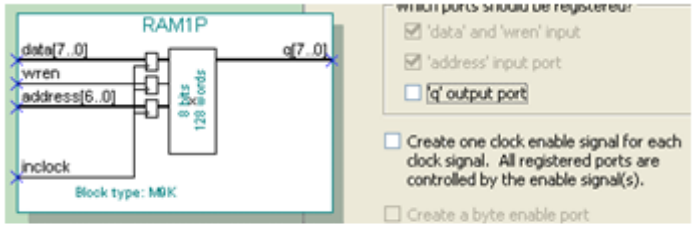

图 5-15 设定 RAM 仅输入时钟控制

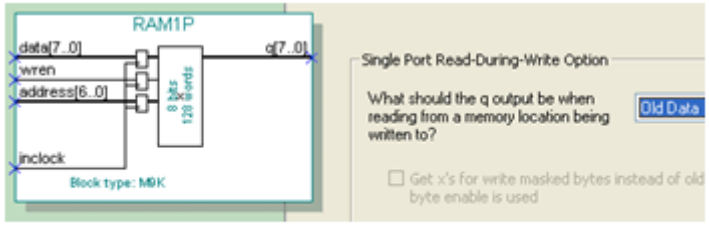

图 5-16 设定在写入同时读出原数据：Old Data

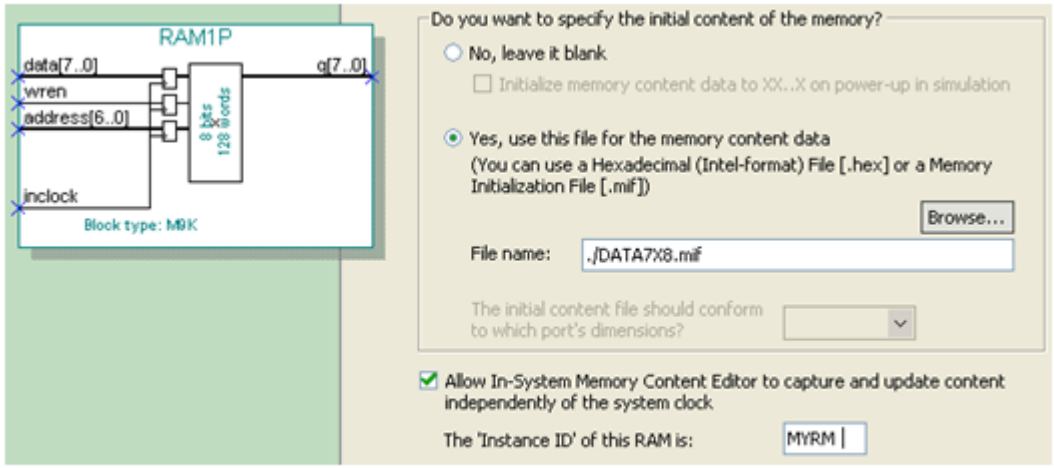

图 5-17 设定初始化文件和允许在系统编辑

5.3 RAM宏模块的使用方法

5.3.2 RAM宏模块的设置和调用

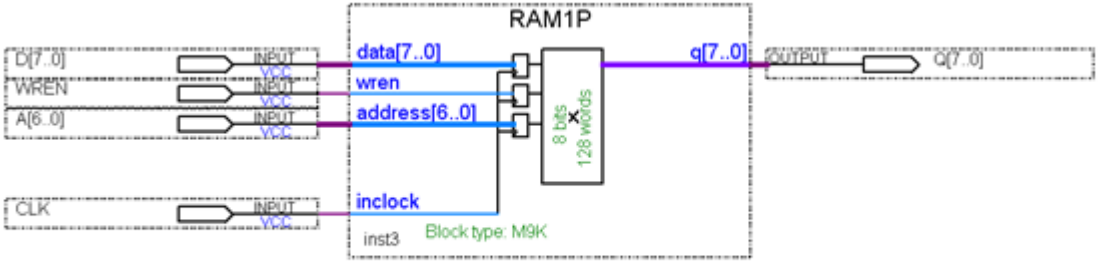

图 5-18 在原理图上连接好的 RAM 模块

5.3 RAM宏模块的使用方法

5.3.3 仿真测试RAM宏模块

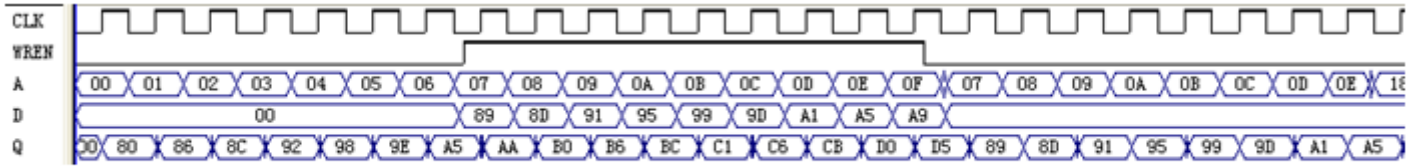

图 5-19 的 RAM 的仿真波形

5.3 RAM宏模块的使用方法

5.3.4 存储器的Verilog代码描述及初始化文件调用

【例 5-7】

```
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
```

5.3 RAM宏模块的使用方法

5.3.4 存储器的Verilog代码描述及初始化文件调用

1. 存储器端口描述

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

2. 存储器的Verilog一般描述

```
reg[7:0]  mem[127:0];
```

```
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 ;        // 允许对寄存器 A 整体赋值
            MEM = 16'hABCD;       // 错误！不允许对存储器多个或者所有单元同时赋值
```

5.3 RAM宏模块的使用方法

5.3.4 存储器的Verilog代码描述及初始化文件调用

3. 存储器中初始化文件的调用（配置）

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

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

【例 5-8】

```
module RAM78 (output[7:0]Q, input[7:0]D,input[6:0]A, input CLK,WREN);
  reg[7:0]  mem[0:127] ;
  always @(posedge CLK ) if (WREN) mem[A] <= D;
   assign Q = mem[A];
   initial $readmemh("RAM78_DAT.dat", mem );
endmodule
```

5.3 RAM宏模块的使用方法

5.3.4 存储器的Verilog代码描述及初始化文件调用

4. 语句语法说明

```
initial
  begin  语句1;  语句2; ... end
```

5.3 RAM宏模块的使用方法

5.3.5 存储器设计的结构控制

【例 5-9】

```
module RAM78(output reg[7:0] Q, input[7:0] D,input[6:0] A, input CLK,WREN);
    reg[7:0]  mem[127:0] /* synthesis ram_init_file="DATA7X8.mif" */;
    always  @(posedge CLK )  if (WREN) mem[A] <= D;
    always @(posedge CLK )  Q = mem[A];
 endmodule
```

5.3 RAM宏模块的使用方法

5.3.5 存储器设计的结构控制

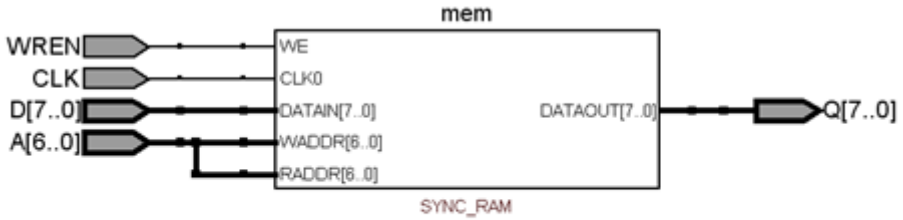

图 5-20 例 5-7 的 RTL 电路模块图

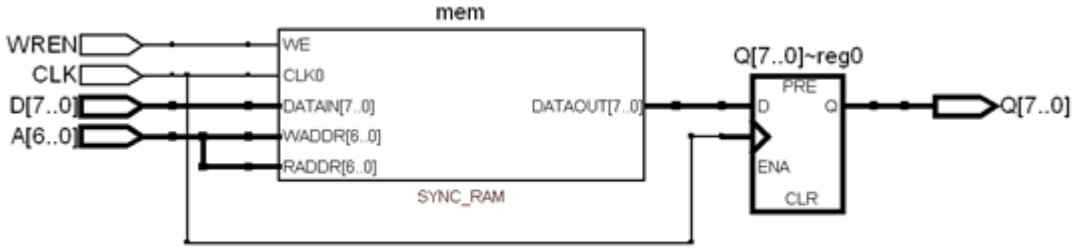

图 5-21 例 5-9 的 RTL 电路模块图

5.3 RAM宏模块的使用方法

5.3.5 存储器设计的结构控制

Top-level Entity Name	RAM78
Family	Cyclone III
Device	EP3C55F484C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	1,497 / 55,856 (3 %)
Total combinational functions	1,340 / 55,856 (2 %)
Dedicated logic registers	1,024 / 55,856 (2 %)
Total registers	1024
Total pins	25 / 328 (8 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 312 (0 %)
Total PLLs	0 / 4 (0 %)

图 5-22 例 5-7 的编译报告

Top-level Entity Name	RAM78
Family	Cyclone III
Device	EP3C55F484C8
Timing Models	Final
Met timing requirements	N/A
Total logic elements	0 / 55,856 (0 %)
Total combinational functions	0 / 55,856 (0 %)
Dedicated logic registers	0 / 55,856 (0 %)
Total registers	0
Total pins	25 / 328 (8 %)
Total virtual pins	0
Total memory bits	1,024 / 2,396,160
Embedded Multiplier 9-bit elements	0 / 312 (0 %)
Total PLLs	0 / 4 (0 %)

图 5-23 例 5-9 的编译报告

5.3 RAM宏模块的使用方法

5.3.5 存储器设计的结构控制

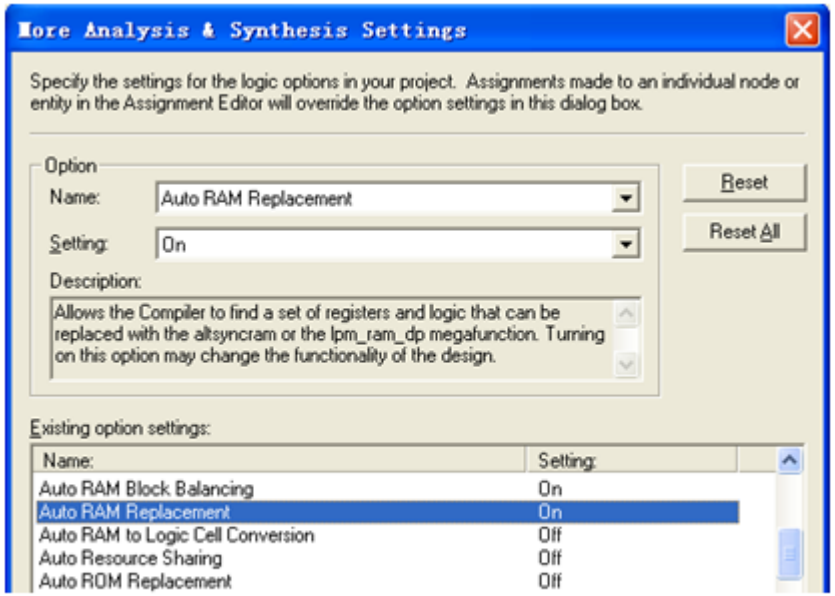

图 5-24 选择 RAM 单元自动替代控制

5.4 LPM存储器在系统读写方法

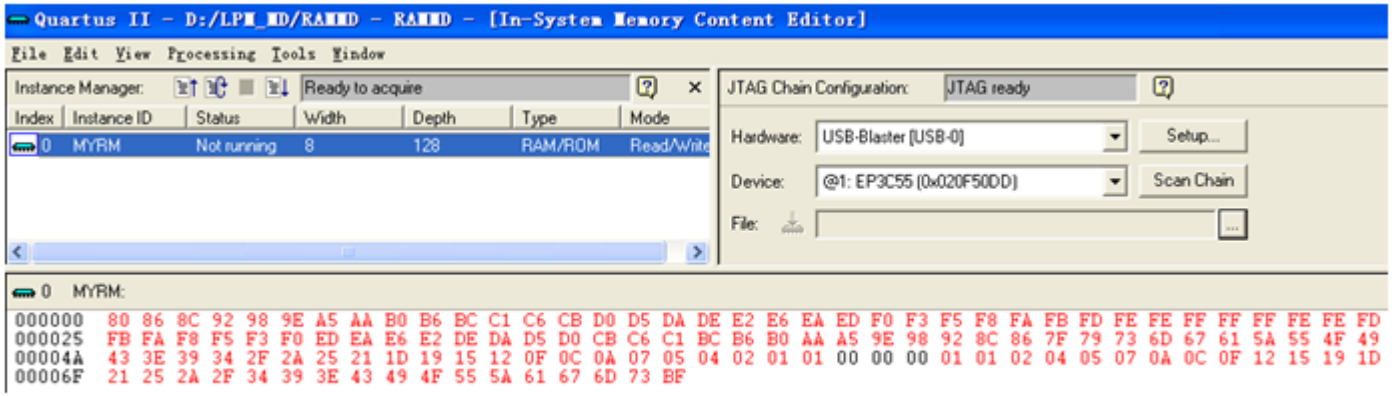

图 5-25 利用 In-System Memory Content Editor 读取 RAM 中的数据

5.5 嵌入式锁相环使用方法

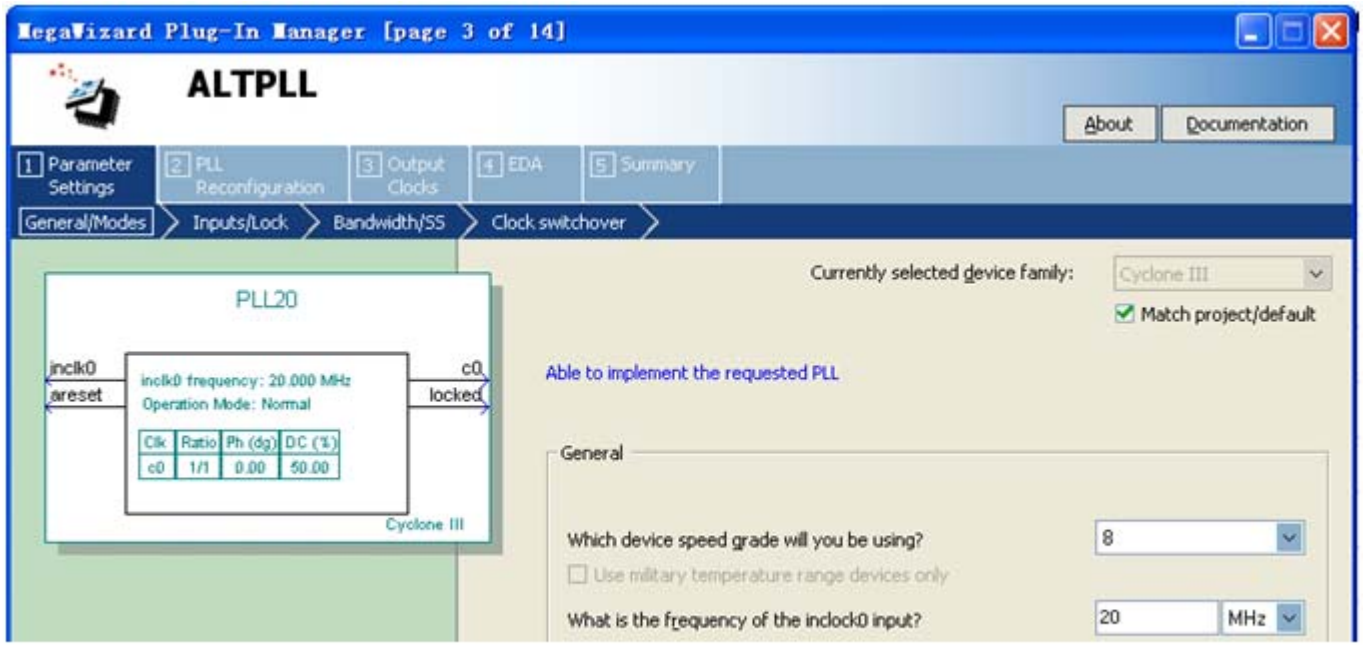

图 5-26 选择输入参考时钟 inclk0 为 20MHz

5.5 嵌入式锁相环使用方法

图 5-27 选择控制信号

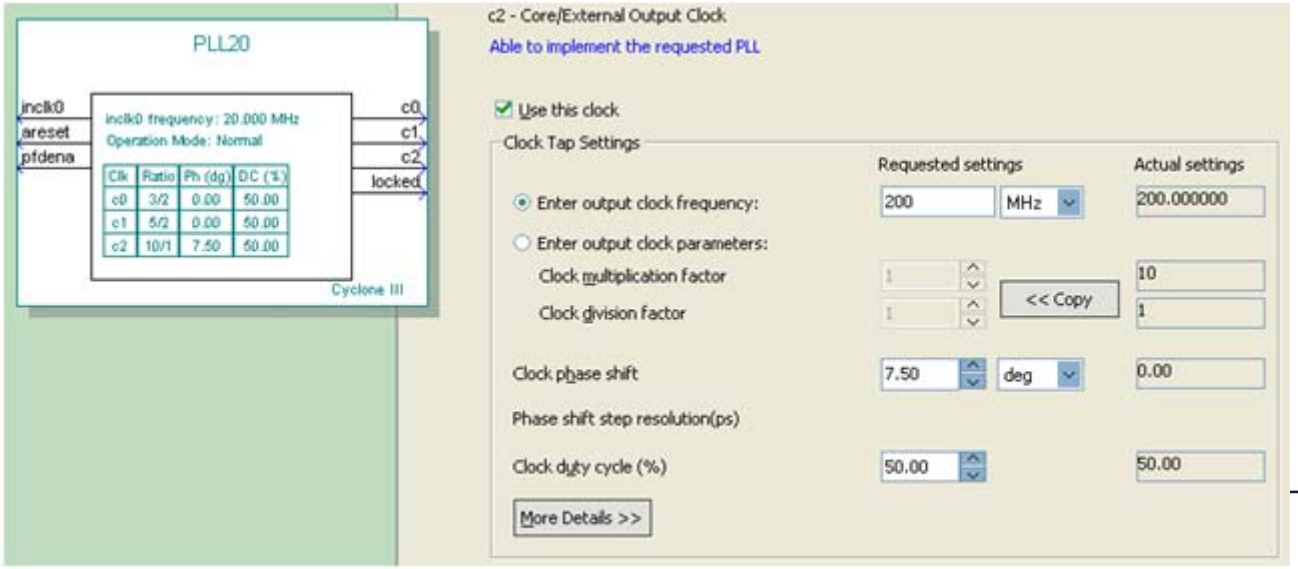

图 5-28 选择 e0 的输出频率为 200MHz

EDA实验与创新实践

5-1 查表式硬件运算器设计

【例5-10】

```
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 ;
```

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

EDA实验与创新实践

5-2 正弦信号发生器设计

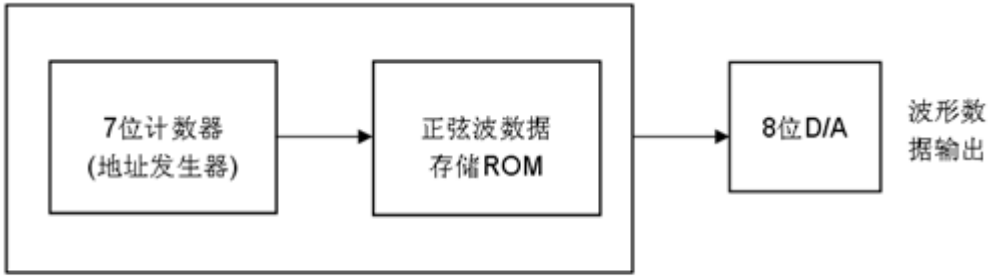

图 5-29 正弦信号发生器结构框图

EDA实验与创新实践

5-2 正弦信号发生器设计

【例 5-12】

```
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
```

【例 5-13】 ROM78.v

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

EDA实验与创新实践

5-2 正弦信号发生器设计

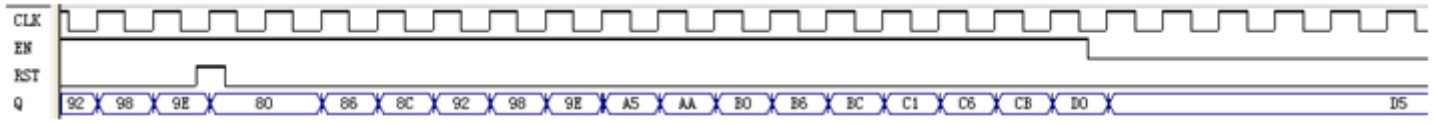

图 5-30 例 5-12 的仿真波形输出

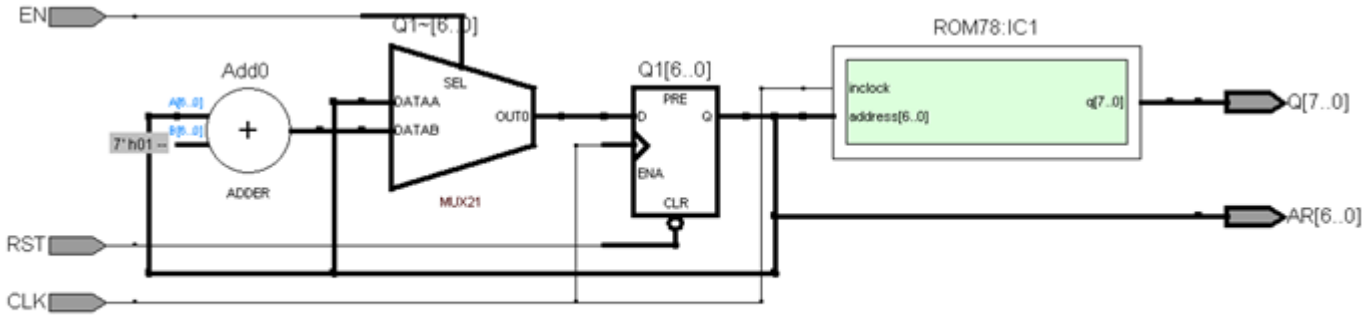

图 5-31 例 5-12 的 RTL 电路图

EDA实验与创新实践

5-2 正弦信号发生器设计

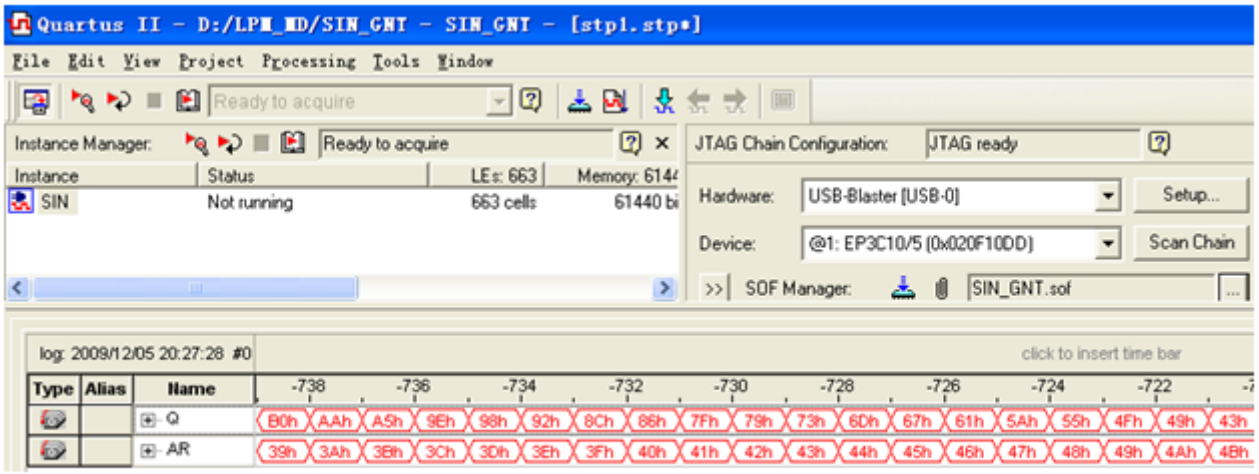

图 5-32 正弦信号发生器数据输出的 SignalTapII 测试图

EDA实验与创新实践

5-2 正弦信号发生器设计

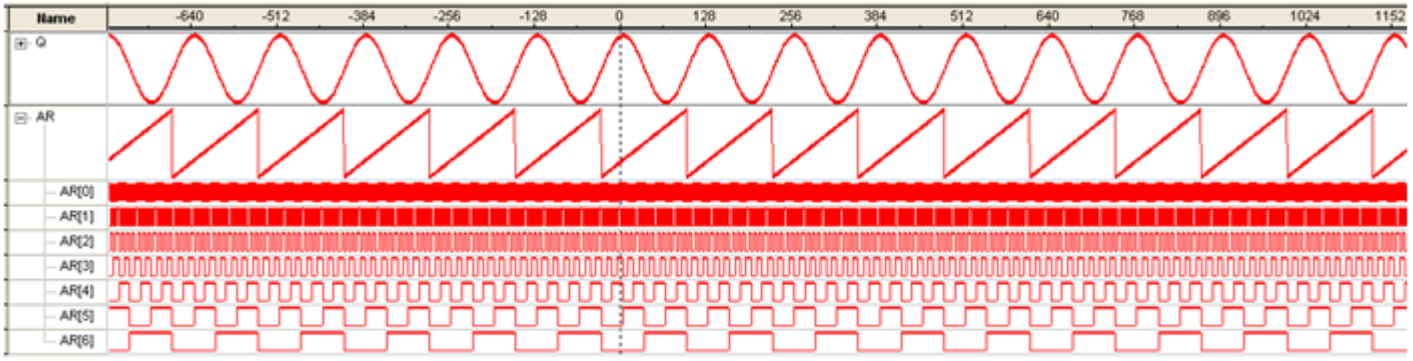

图 5-33 正弦信号发生器的 SignalTapII 的波形显示图

EDA实验与创新实践

5-3 DDS正弦信号发生器设计

$$S_{out} = A\sin\omega t = A\sin(2\pi f_{out} t) \tag{5-1}$$

$$\theta = 2\pi f_{out} t \tag{5-2}$$

$$\Delta\theta = 2\pi f_{out} T_{clk} = \frac{2\pi f_{out}}{f_{clk}} \tag{5-3}$$

$$\frac{B_{\Delta\theta}}{2^N} = \frac{f_{out}}{f_{clk}}, \quad B_{\Delta\theta} = 2^N \cdot \frac{f_{out}}{f_{clk}} \tag{5-4}$$

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

$$B_{\theta_{k-1}} \approx \frac{\theta_{k-1}}{2\pi}\cdot 2^N \tag{5-6}$$

EDA实验与创新实践

5-3 DDS正弦信号发生器设计

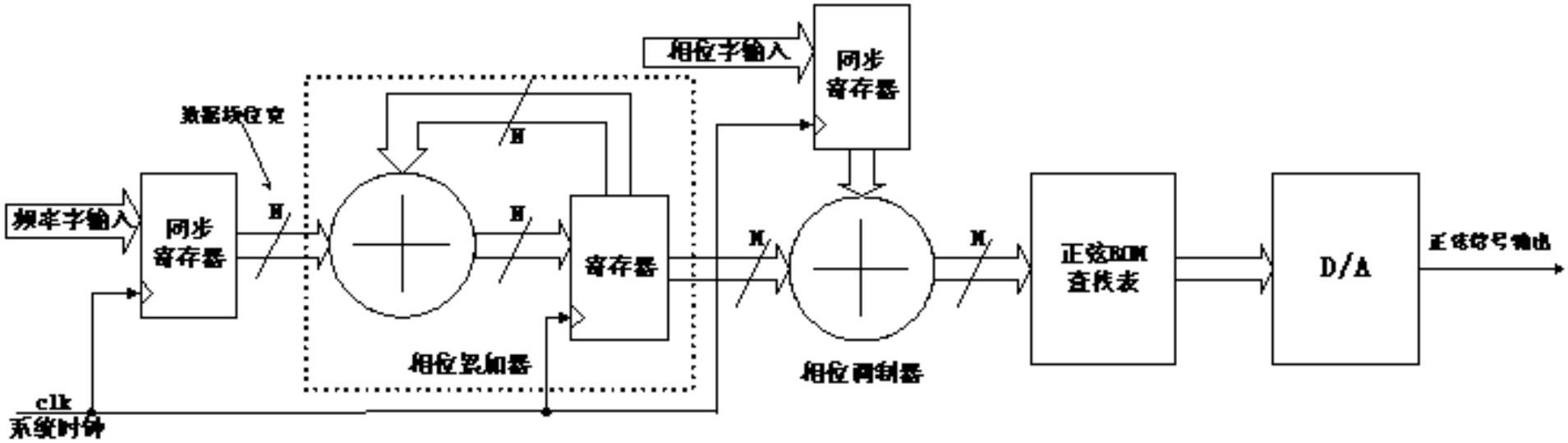

图 5-34 基本 DDS 结构

EDA实验与创新实践

5-3 DDS正弦信号发生器设计

$$f_{out} = \frac{B_{\Delta\theta}}{2^N} \cdot f_{clk} \qquad (5\text{-}7)$$

$$f_{out} = \frac{f_{clk}}{2^N} \qquad (5\text{-}8)$$

$$f_{out} = \frac{B[31..0]}{2^{32}} \cdot f_{clk} \qquad (5\text{-}9)$$

EDA实验与创新实践

5-3 DDS正弦信号发生器设计

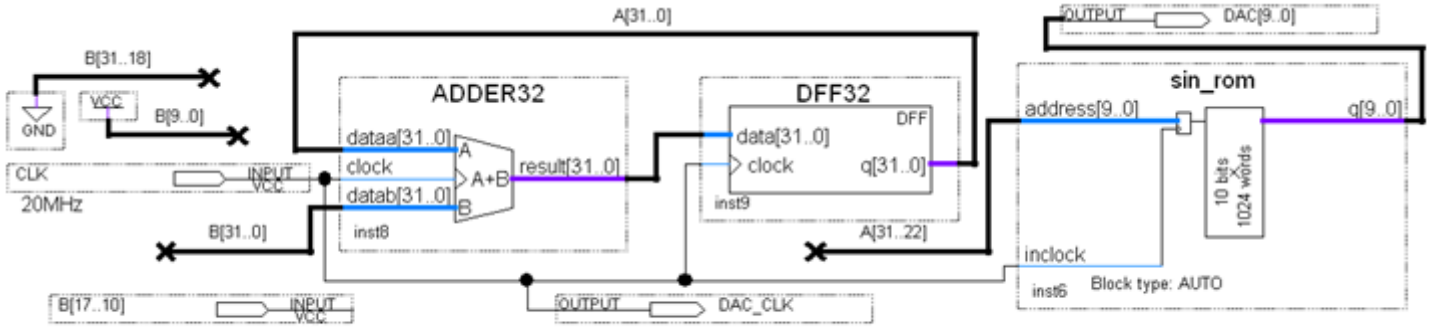

图 5-35 DDS 信号发生器电路顶层原理图

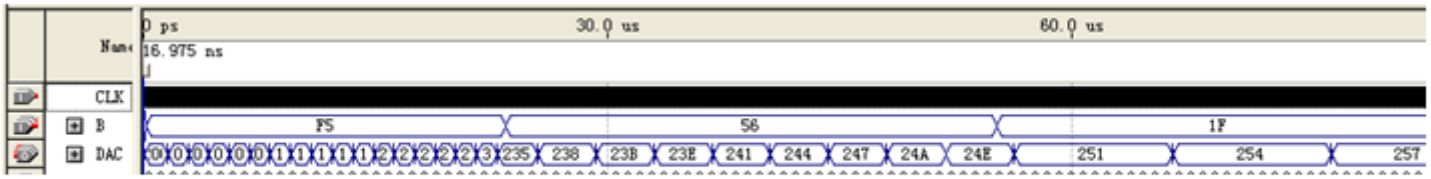

图 5-36 图 5-35 的仿真波形

EDA实验与创新实践

5-4 移相信号发生器设计

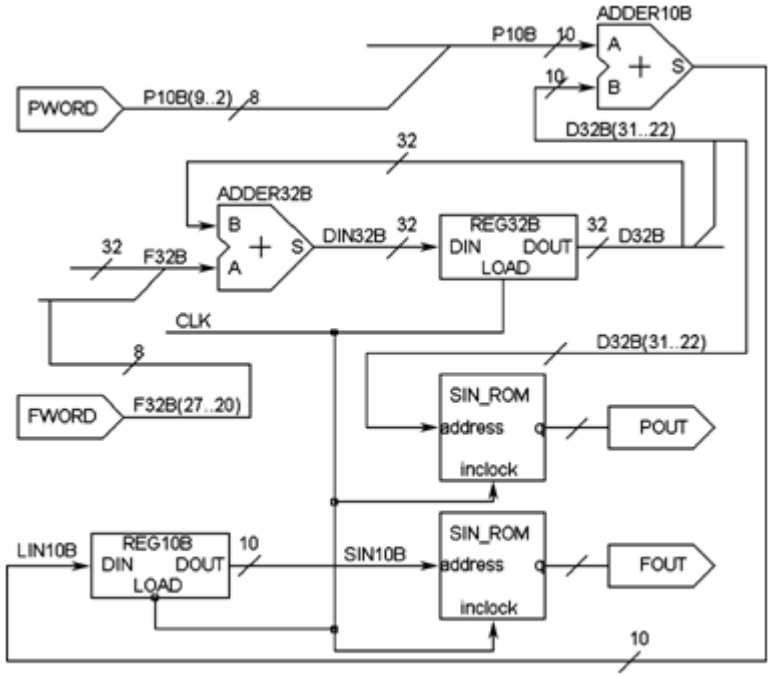

图 5-37 全数字移相信号发生器电路模型图