基于 Nios II IP 核的 VGA 功能设计

VGA Function IP design based on NiosII

Revision.0

http://XiaomaGee.cnblogs.com

http://i-board.taobao.com

http://www.heijin.org

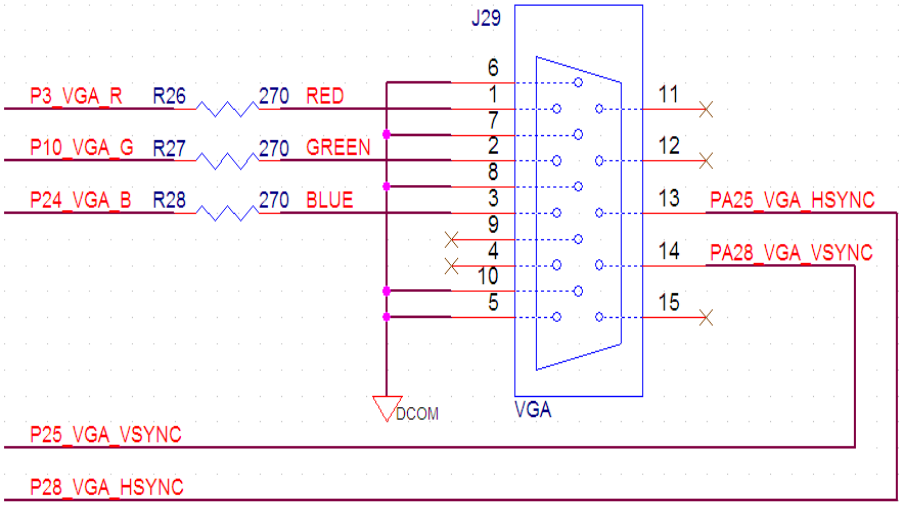
E.V. Stdio. 2012/10/17

本文基于 iCore 组合板创建，详细信息请点击上面网址信息。

目　录

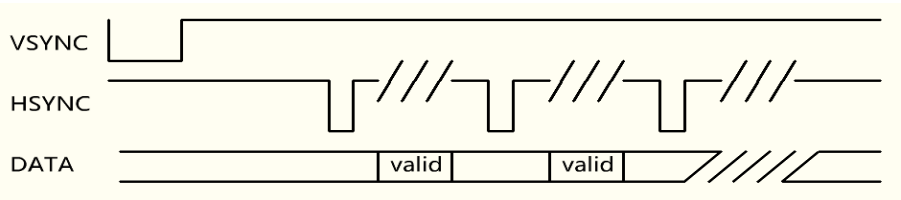
第一章 VGA IP 实现原理

VGA(Video Graphics Array)即视频图形阵列，是 IBM 在 1987 年随 PS/2 一起推出。通用 VGA 系统主要由控制电路、显示缓存区和基本输入输出系统程序三部分做成。标准 VGA 接口共有 15 个接口（如下图所示），真正能用到的只有 5 个接口。HSYNC 是行同步信号，VSYNC 是场同步信号，同步信号就是让 VGA 显示器接收部分知道送来的数据是对应哪一行哪一列的哪一个像素点。VGA_R、VGA_G、VGA_B 是三原色信号，这三个信号接口的输入都是模拟信号（标准为 0～0.7V），所以它们都有相应的地线需要连接。本文档基于 icore 开发板的接口做的比较简单，直接用 CPLD 的 I/O 口连接 VGA 的5个信号接口 ,且三原色信号接口输入的只可能是数字信号(0或1)，因此驱动液晶屏上显示的颜色最多有 8 种。



一、 VGA 时序

VGA 接口时序如下图所示，场同步信号 VSYNC 在每帧开始的时候产生一个固定宽度的低脉冲，行同步信号 HSYNC 在每行开始的时候产生一个固定宽度的低脉冲，数据在某些固定的行和列交汇处有效。



本文档以 800*600@60Hz 分辨率详细分析 VGA 时序：

800*600*60Hz	a 段	b 段	c 段	d 段	e 段-共 n 个列像素
HSYNC	128	88	800	40	1056
800*600*60Hz	o 段	p 段	q 段	r 段	s 段-共 n 个行像素
VSYNC	4	23	600	1	628

HSYNC 是用来控制“列填充”，而一个 HSYNC 可以分为 4 段：a（同步段）、b(后廊段)、c（激活段）、d（前廊段）。HSYNC 的 a 是拉低的 128 个列像素，b 是拉高的 88 个列像素，c 是拉高的 800 个列像素，最后的 d 是拉高的 40 个列像素。列总共有 1056 个列像素。

VSYNC 是用来控制“行扫描”。而一个 VSYNC 同样可以分为 4 段：o（同步段）、p（后廊段）、q（激活段）、r（前廊段）。o 是拉低的 4 个行像素，p 是拉高的 23 个行像素，q 是拉高的 600 个行像素，最后的 r 是拉高的 1 个行像素。行总共有 628 个行像素。

HSYNC 只有在 c 段同时 VSYNC 在 q 段即 HSYNC 信号和 VSYNC 信号同时是激活段时，输入数据才有效。

Vga.v 和 Vga_timing.v 是构建 VGA 内核必不可少的两个核心文件。Vga.v 主要用于，在此不作详细介绍，详见本文档附带压缩包。Vga_timing.v 主要定义了 VGA 各种逻辑关系之间的逻辑关系。

```
module vga_timing (
    input  wire clk_i,
    input  wire reset_i,           //输入复位信号
    output wire vga_pixel_flag,    //输出像素有效
    output reg  vga_line_o,        //输出水平信号
    output reg  vga_field_o,       //输出垂直信号
    output reg  vga_frame_o        //输出帧开始信号
);

reg [12:0] line_sync_count;        //行同步计数器
reg [12:0] field_sync_count;       //场同步计数器

//60Hz 800x600
parameter  line_sync_head    = 128; //行同步头宽度
parameter  line_pixel_begin  = 216; //行有效像素开始位置
parameter  line_pixel_end    = 1016;//行有效像素结束位置
parameter  line_time         = 1056;//行周期

parameter  field_sync_head   = 4;   //场同步头宽度
parameter  field_pixel_begin = 27;  //场有效像素开始位置
```

```
21 parameter  field_pixel_end   = 627; //场有效像素结束位置
22 parameter  field_time        = 628; //场周期
23
24
25 wire   line_pixel_flag;
26 wire   field_pixel_flag;
27 //计算行像素有效位置
28
assign line_pixel_flag  = (line_sync_count  >= line_pixel_begin)
&& (line_sync_count  < line_pixel_end);
29 //计算场像素有效位置
30                                          assign field_pixel_flag
= (field_sync_count      >= field_pixel_begin) && (field_sync_count
< field_pixel_end);
31 //计算像素有效位置
32 assign vga_pixel_flag   = line_pixel_flag & field_pixel_flag;
33
34 always @ (posedge clk_i or negedge reset_i) begin
35     if (!reset_i) begin
36         //复位时清 0 行计数器
37         line_sync_count <= 0;
38     end else begin
39         if (line_sync_count == line_time) begin
40             //如果到达行周期就重新开始计数
41             line_sync_count <= 0;
42         end else begin
43             //每个脉冲行计数器加 1
44             line_sync_count <= line_sync_count + 1;
45         end
46     end
47 end
48
49 //场同步计数
50 always @ (posedge clk_i or negedge reset_i) begin
51     if (!reset_i) begin
52         //复位时清 0 场计数器
53         field_sync_count <= 0;
54     end else begin
55         if (field_sync_count == field_time) begin
56             //如果到达场周期就重新开始计数
57             field_sync_count <= 0;
58         end else if (line_sync_count == line_time) begin
```

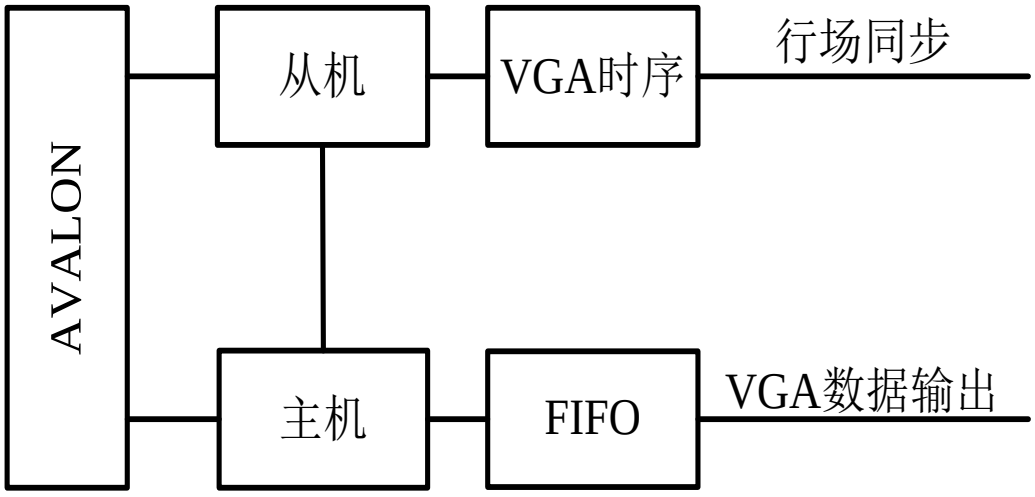
```
59              //如果行输出完毕场计算器加 1
60              field_sync_count <= field_sync_count + 1;
61          end
62      end
63  end
64
65  //帧开始
66  always @ (posedge clk_i or negedge reset_i) begin
67      if (!reset_i) begin
68          vga_frame_o <= 0;
69      end else if (field_sync_count < 2) begin
70          vga_frame_o <= 1;
71      end else begin
72          vga_frame_o <= 0;
73      end
74  end
75
76  //时序控制
77  always @ (posedge clk_i or negedge reset_i) begin
78      if (!reset_i) begin
79          //设置复位时行场引脚电平
80          vga_line_o  <= 1;
81          vga_field_o <= 1;
82      end else begin
83          //输出行同步头信号
84          if (line_sync_count  == 0)      vga_line_o  <= 0;
85          //输出场同步头信号
86          if (field_sync_count == 0)      vga_field_o <= 0;
87          //停止输出行同步头信号
88         if (line_sync_count  == line_sync_head)
                                            vga_line_o  <= 1;
89          //停止输出场同步头信号
90          if (field_sync_count == field_sync_head)
                                           vga_field_o <= 1;
91      end
92  end
93  endmodule
```

以上代码主要解析了 VGA 时序的逻辑关系，程序从第 2 行～第 7 行分别定义了 clk_i 输入时钟信号、reset_i 输入复位信号、vga_pixel_flag 输出像

素有效、vga_line_0 输出水平信号、vga_field_0 输出垂直信号、vga_frame_0 输出帧开始信号。程序从第 10 行～第 11 行定义了 line_sync_count 行同步计数器和 field_synv_count 场同步计数器两个变量。程序从第 13 行～第 22 行声明了 VGA 的相关参数及参数设置，相关参数的详细内容及设置请参考：http://www.docin.com/p-68197623.html 。程序第 28 行、第 30 行、第 32 行分别定义了计算行像素有效位置、计算场像素有效位置、计算像素有效位置。程序从第 34 行～第 47 行是对行计数器的定义。程序从第 49 行～第 63 行是对场计数器的定义。程序从第 65 行～第 74 行是帧的定义。程序从第 76 行～第 93 行是对 VGA 时序控制的定义。

二、 VGA IP 核原理

1、VGA IP 核原理框图



2、IP 核实现 vga.v

```
module vga(
    input wire clk,
    input wire reset,

    input wire slave_chipselect,
    input wire [1 : 0]  slave_address,
    input wire slave_read,
    output wire [31 : 0] slave_readdata,
    input wire slave_write,
    input wire [31 : 0] slave_writedata,

    output wire [31 : 0] master_address,
    output wire master_byteenable,
    output wire master_read,
    input wire [7 : 0]  master_readdata,
    input wire master_waitrequest,
    input wire master_readdatavalid,

    input wire vga_clk,
    output wire vga_line_sync,
    output wire vga_field_sync,
    output wire [2 : 0]  vga_rgb
    );

wire vga_pixel_flag;
wire [7 : 0] fifo_data;
wire vga_frame_o;

assign vga_rgb = vga_pixel_flag ? fifo_data[2 : 0] : 3 'b000;

vga_timing vga_11(
    .clk_i(vga_clk),
    .reset_i(reset),
    .vga_pixel_flag(vga_pixel_flag),
    .vga_line_o(vga_line_sync),
    .vga_field_o(vga_field_sync),
    .vga_frame_o(vga_frame_o)
);


//slave
reg        vga_go_r;
reg [31:0] readdata_r;
reg [31:0] vga_base_address;
```

```
45
46 always @ (posedge clk or negedge reset) begin
47     if (!reset) begin
48         vga_go_r <= 0;
49     end else begin
50         if (slave_chipselect & slave_write) begin
51             case (slave_address)
52                 2'b00 : begin
53          vga_base_address <= slave_writedata;
54 end
55 2 'b01: begin
56                     vga_go_r<= slave_writedata[0];
57                 end
58             endcase
59         end
60     end
61 end
62
63 always @ (posedge clk) begin
64     if (reset) begin
65         if (slave_chipselect & slave_read) begin
66             case (slave_address)
67                 2'b00 : begin
68 readdata_r <= vga_base_address;
69 end
70 2 'b01: begin
71          readdata_r
<= {31'b0000_0000_0000_0000_0000_0000_0000_000, vga_go_r };
72 end
73 endcase
74 end
75 end
76 end
77
78 //master
79     reg  [21 : 0] input_data_count;
80
81 reg vga_read;
82 wire [16 : 0] fifo_count;
83 wire fifo_clear;
84
85 assign master_read = vga_read;
86 assign master_address = vga_base_address + input_data_count;
```

```
 87
 88 assign fifo_clear = vga_frame_o;
 89
 90 assign master_byteenable = 1 'b1;
 91
 92 always @ (posedge clk or negedge reset) begin
 93     if (!reset) begin
 94         input_data_count <= 0;
 95     end else if (vga_go_r) begin
 96         if (!master_waitrequest) begin
 97             if (fifo_clear) begin
 98                 input_data_count <= 0;
 99             end else begin
100                 if (input_data_count < 22'd480000)begin
101                if (vga_read)
                    input_data_count <= input_data_count + 1 'b1;
102                 end
103             end
104         end
105     end else begin
106         input_data_count <= 0;
107     end
108 end
109
110 always @ (posedge clk or negedge reset) begin
111     if (!reset) begin
112         vga_read <= 0;
113     end else if (vga_go_r) begin
114         if (!master_waitrequest) begin
115             if (fifo_clear) begin
116                 vga_read <= 0;
117             end else begin
118                 if ((fifo_count < 2000) && (input_data_count
< 22'd480000)) begin
 119     vga_read <= 1;
120 end else if (fifo_count >= 30000) begin
121     vga_read <= 0;
122 end
123 end
124 end
125     end else begin
126     vga_read <= 0;
127 end
```

```
128 end
129
130 vga_fifo fifo_12(
131     .aclr(fifo_clear),
132     .data(master_readdata),
133     .rdclk(~vga_clk),
134     .rdreq(vga_pixel_flag),
135     .wrclk(clk),
136     .wrreq(master_readdatavalid),
137     .q(fifo_data),
138     .wrusedw(fifo_count)
139     );
140
141 endmodule
```

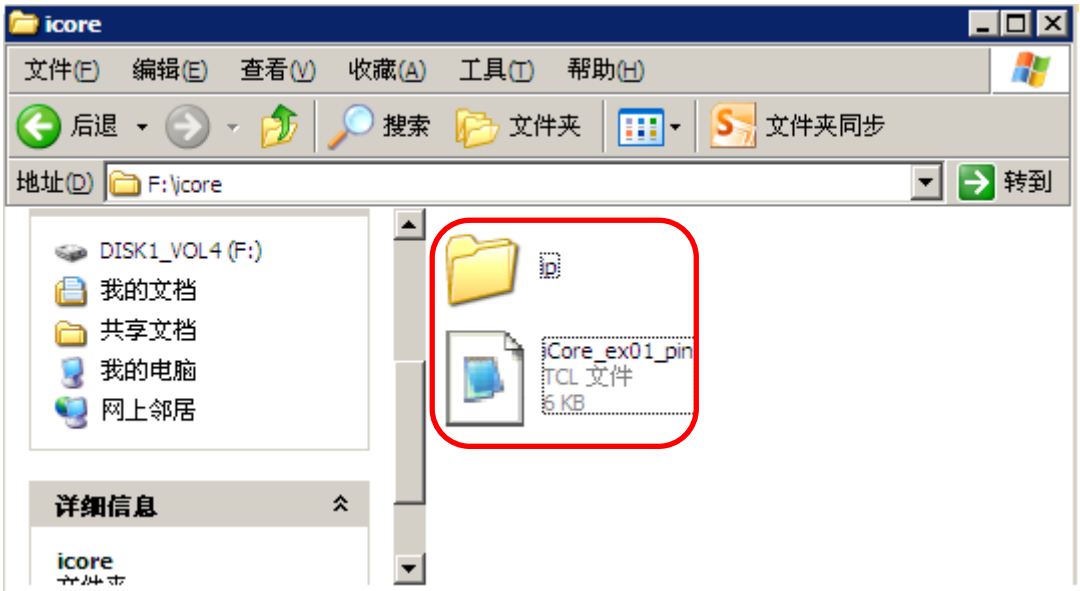
程序从第 41～第 77 行，实现 VGA 模块 AVALON 总线从机模块的功能：地址 0 写入 VGA 数据在 AVALON 总线的地址。（如果连接 SDRAM 就是 SDRAM 为 VGA 开辟的一个显示缓冲器的首地址）地址 1 的 0 位控制 VGA 是否工作，1 工作，0 停止工作。

程序的 78～128 行，实现 VGA 模块 AVALON 总线主机模块的功能：通过从机模块设置的 VGA 数据缓冲区地址来向 AVALONG 索要数据，输出到 VGA 的 FIFO 上。VGA 时序控制器有效的读取 FIFO 的数据到 VGA 上。

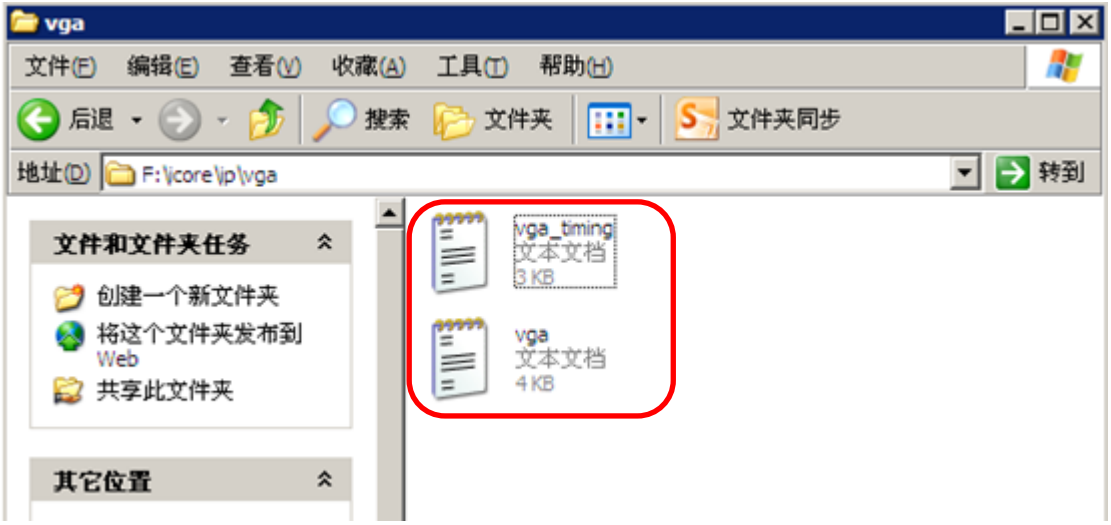
第二章　VGA IP 核的构建

一、 建立 QuartusII 工程

1、首先在 F 盘下建立一个 icore 文件夹用于存储工程，icore 文件夹下新建一个 ip 文件夹用于储存 vga 的 ip 核文件，另将 icore 的脚本文件复制到 icore 文件夹下。如下图所示：



2、在 ip 文件夹下新建一个 vga 文件夹，将 vga.v 和 timing.v 这两个文件存放 vga 文件夹下。如下图所示：

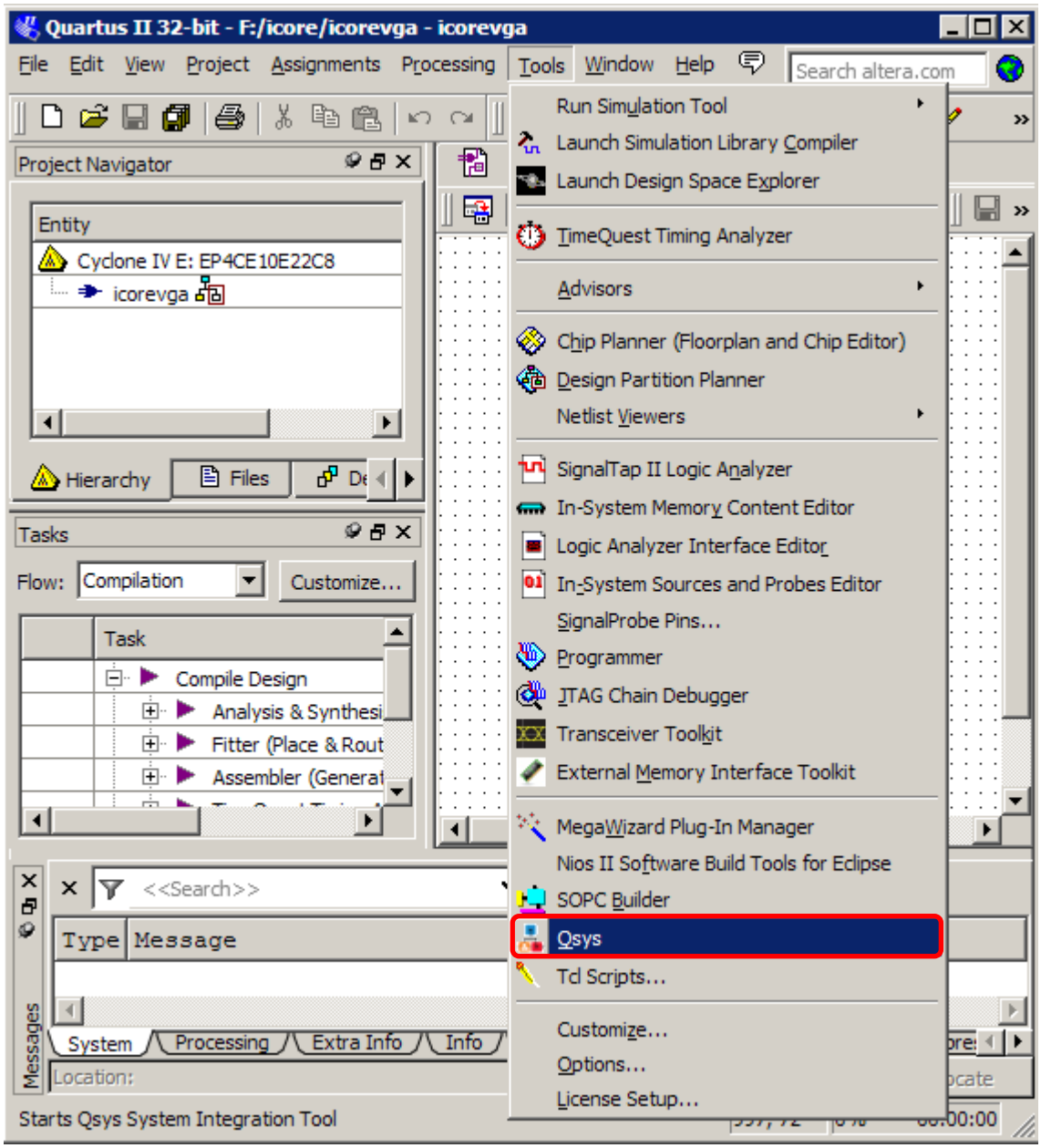


3、打开 Quartus II12.0 软件，建立一个空工程：将工程命名为 icorevga，选择 Cyclone IV E 系列内的 EP4CE10E22C8 为主器件，此步操作较为简单在此不作详细介绍，详见基于 icore 新建工程的文档。

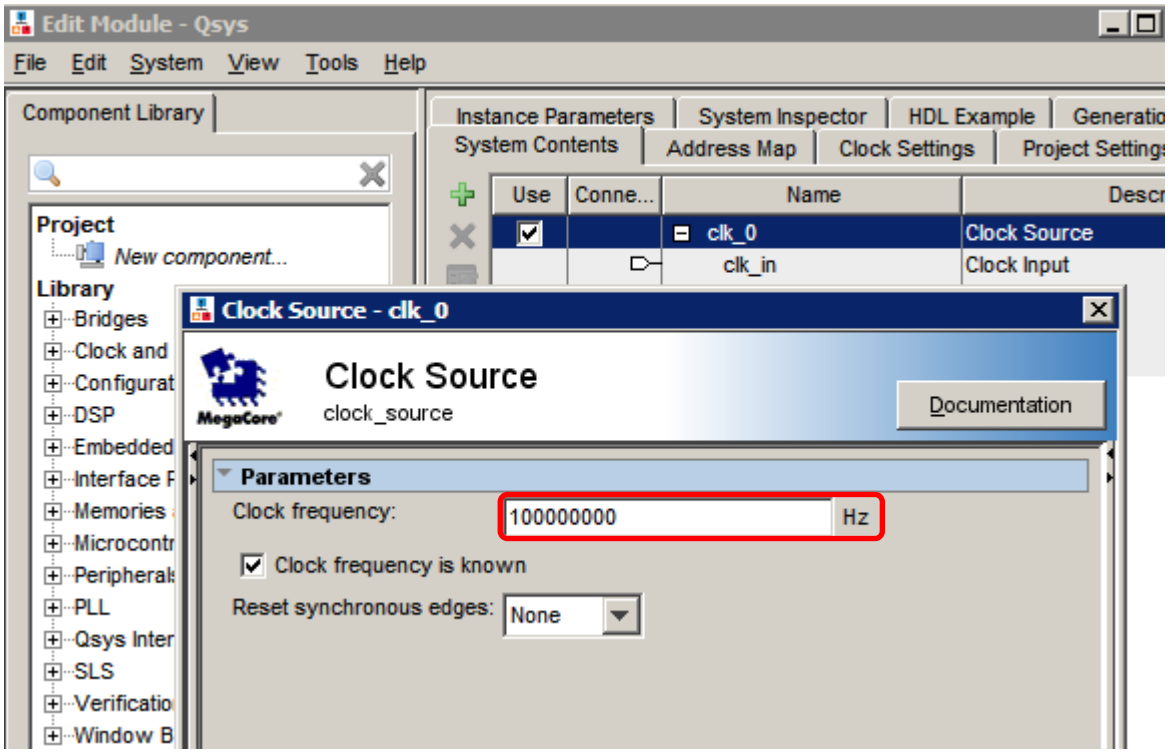
4、建立一个 Block.bdf 文件。至此 Quartus 工程和顶层设计文件已经建立完成。

二、 构建 Nios II 软核

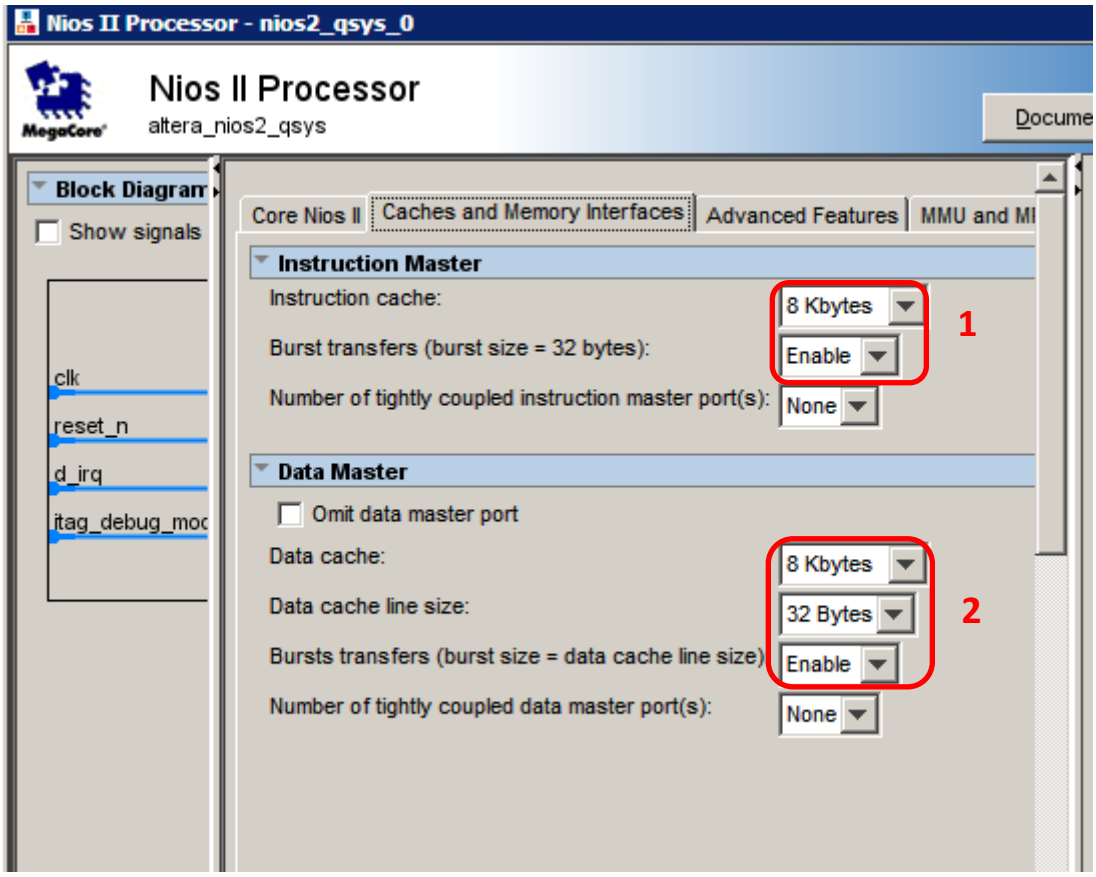
5、点击“Tools/Qsys”。如下图所示：



6、将 clk 的频率设置为 100Mhz。如下图所示：



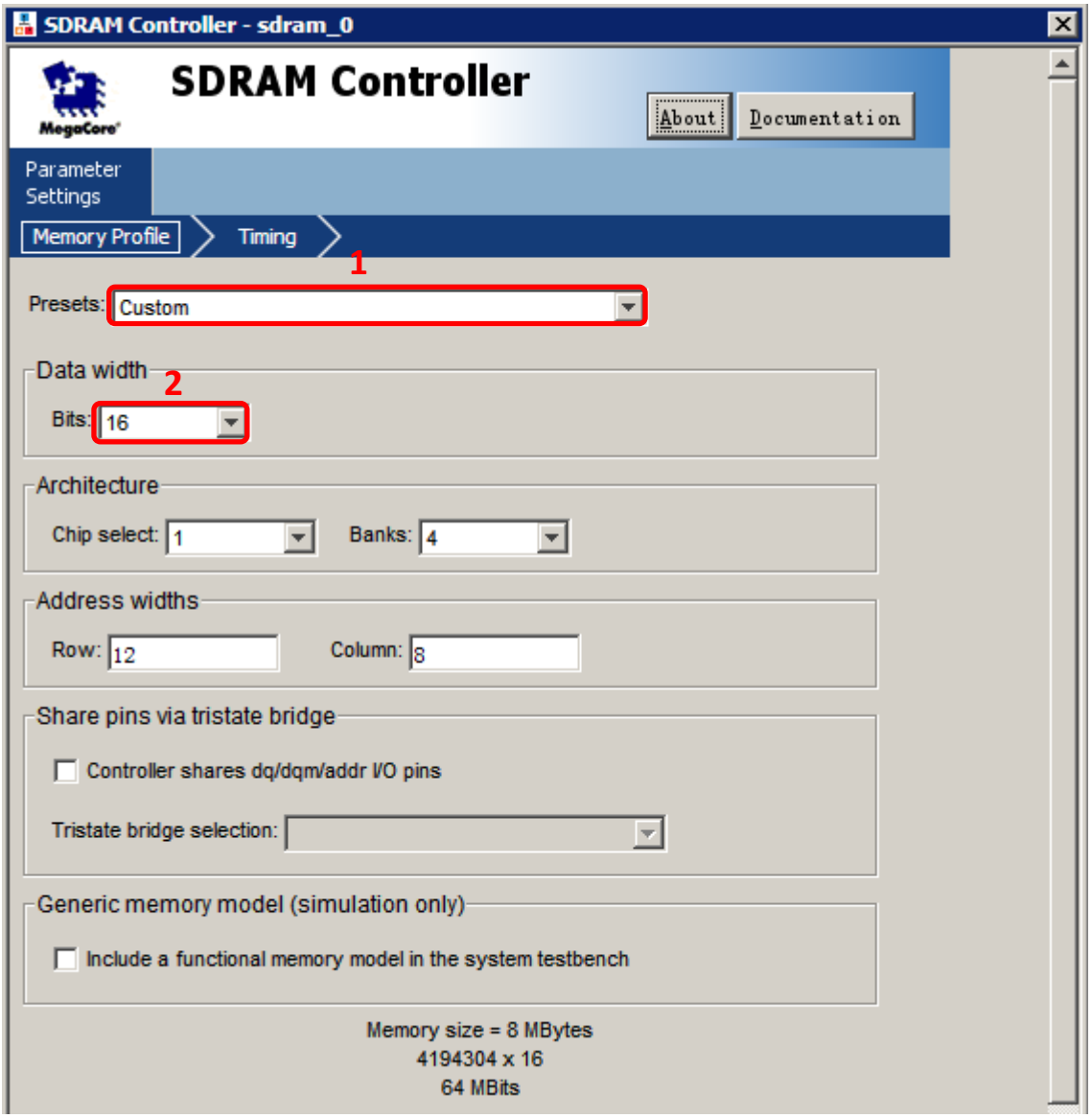
7、构建 CPU，修改参数。如下图所示：



8、构建 JTAG UART。

9、构建 FLASH。

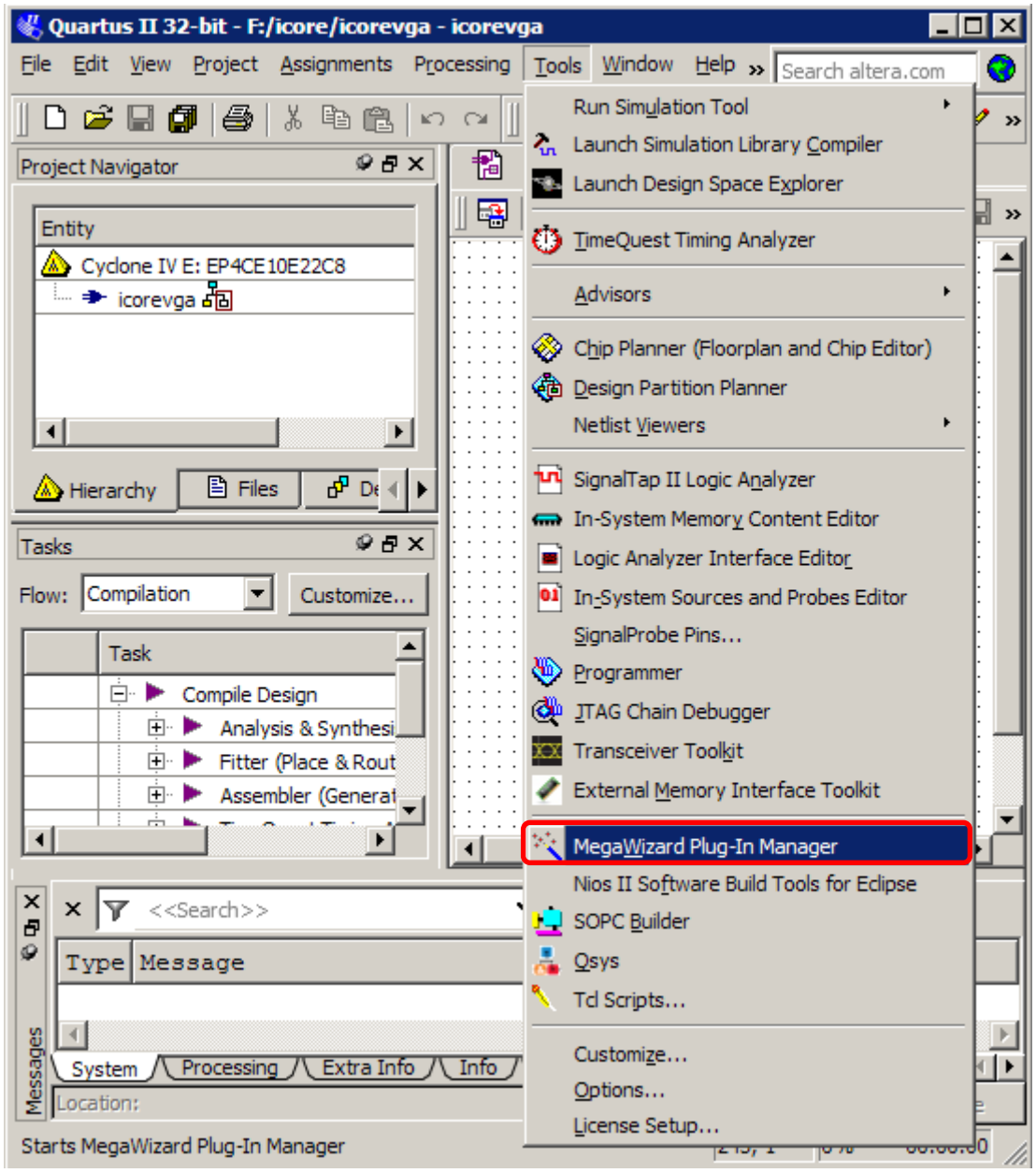
10、构建 SDRAM。修改相关参数，如下图所示：



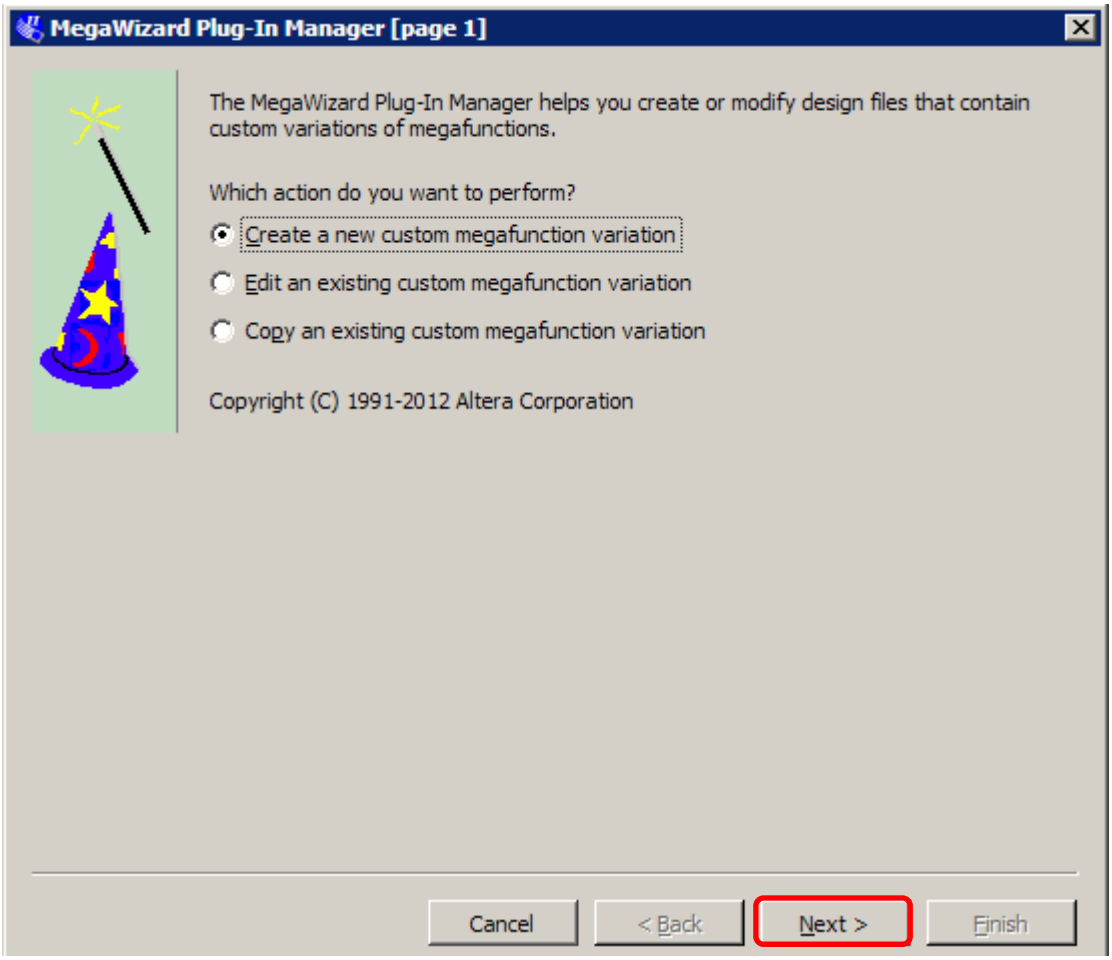
11、构建 System ID。

三、 构建 VGA 的 IP 核

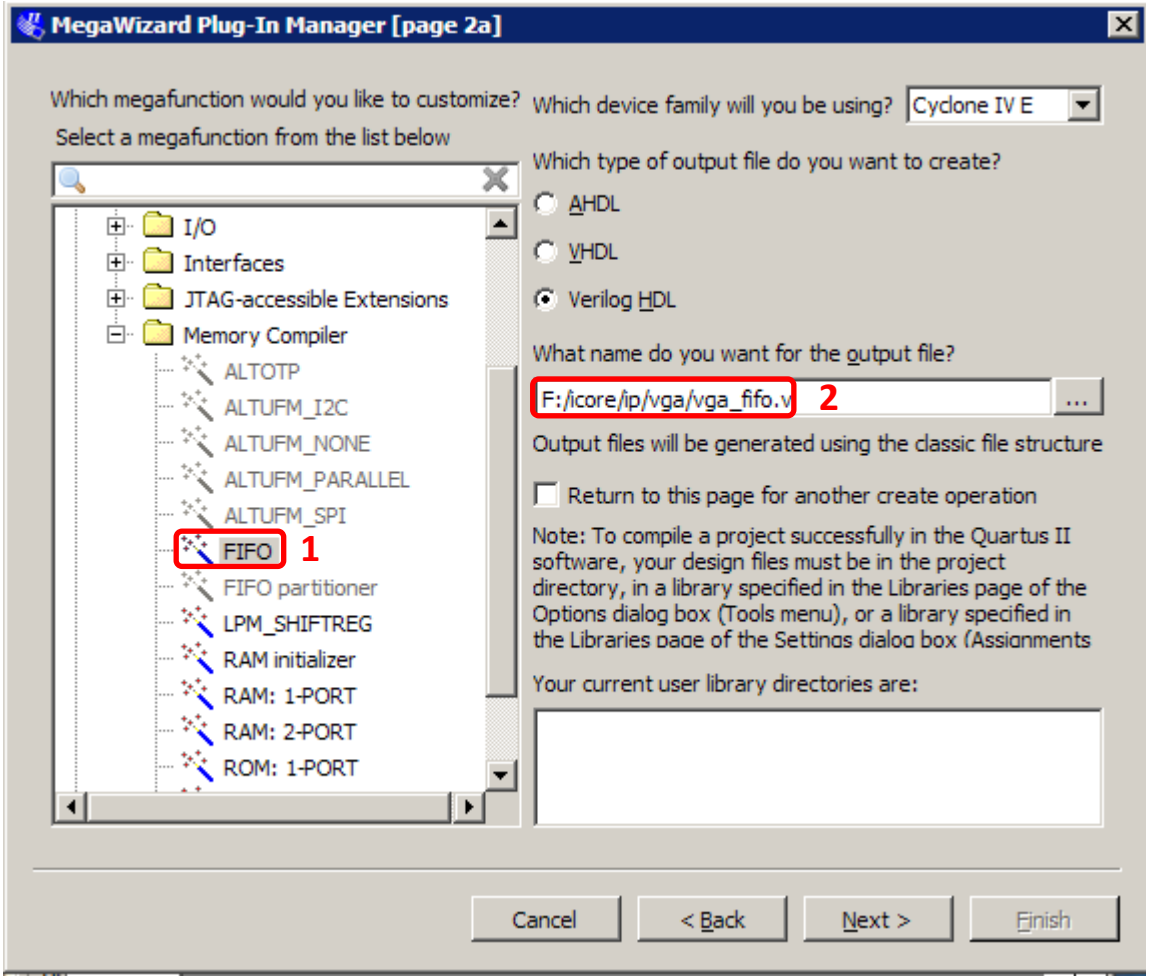
12、在 Quartus 界面下点击“Tool/MegaWizard Plug_In Manager”。如下图所示：



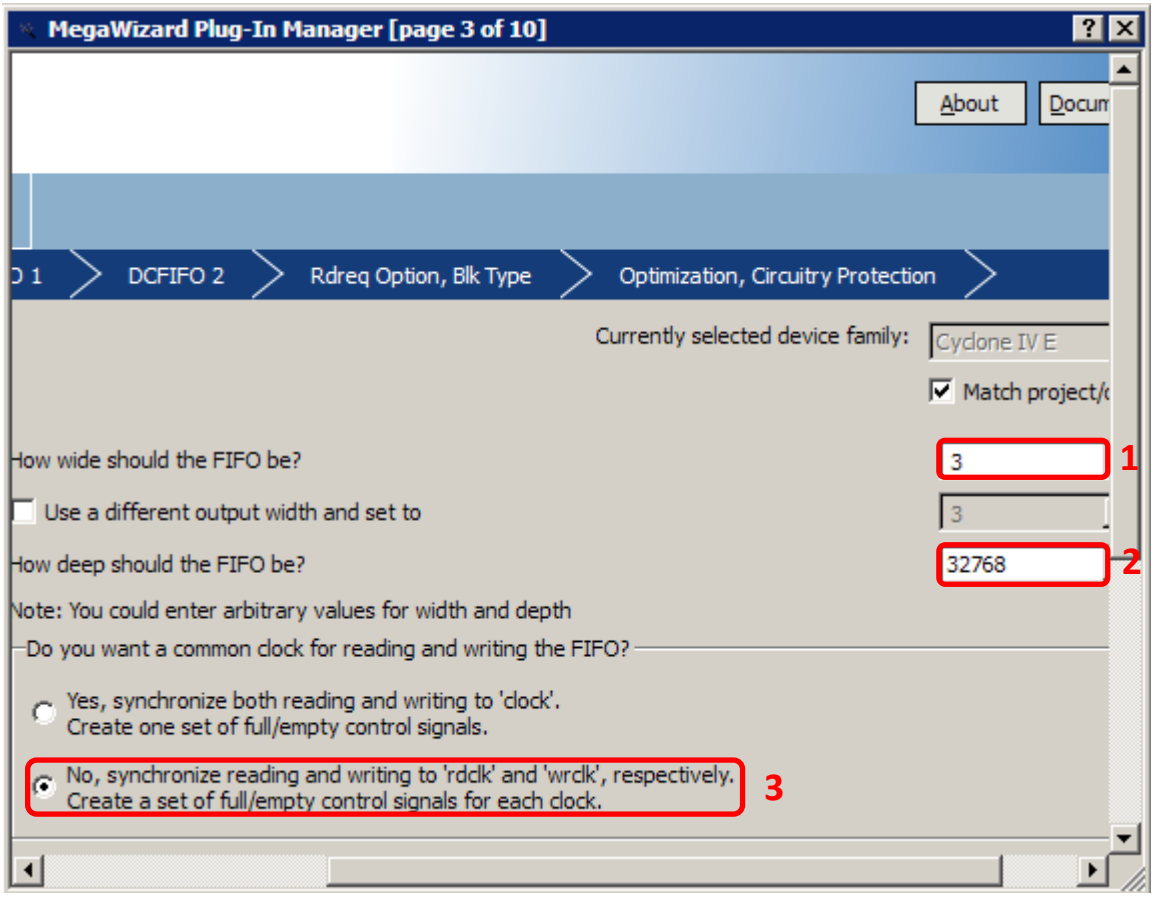
13、直接点击“Next”。如下图所示：



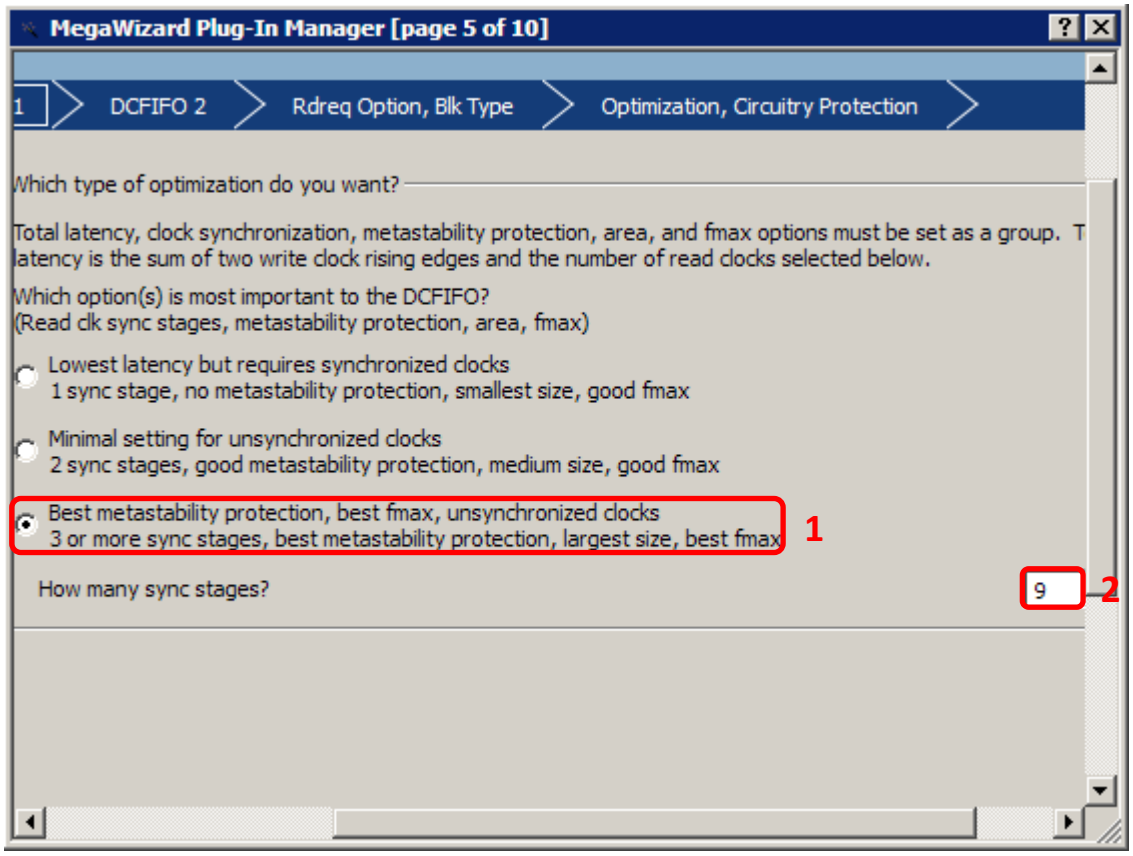
14、点击“Memory Compiler/FIFO”,按照下图中的“2”对输出文件进行命名,并注意输出文件的路径,它是与 vga.v 和 timing.v 的路径是一致的,然后点击“Next”。如下图所示：



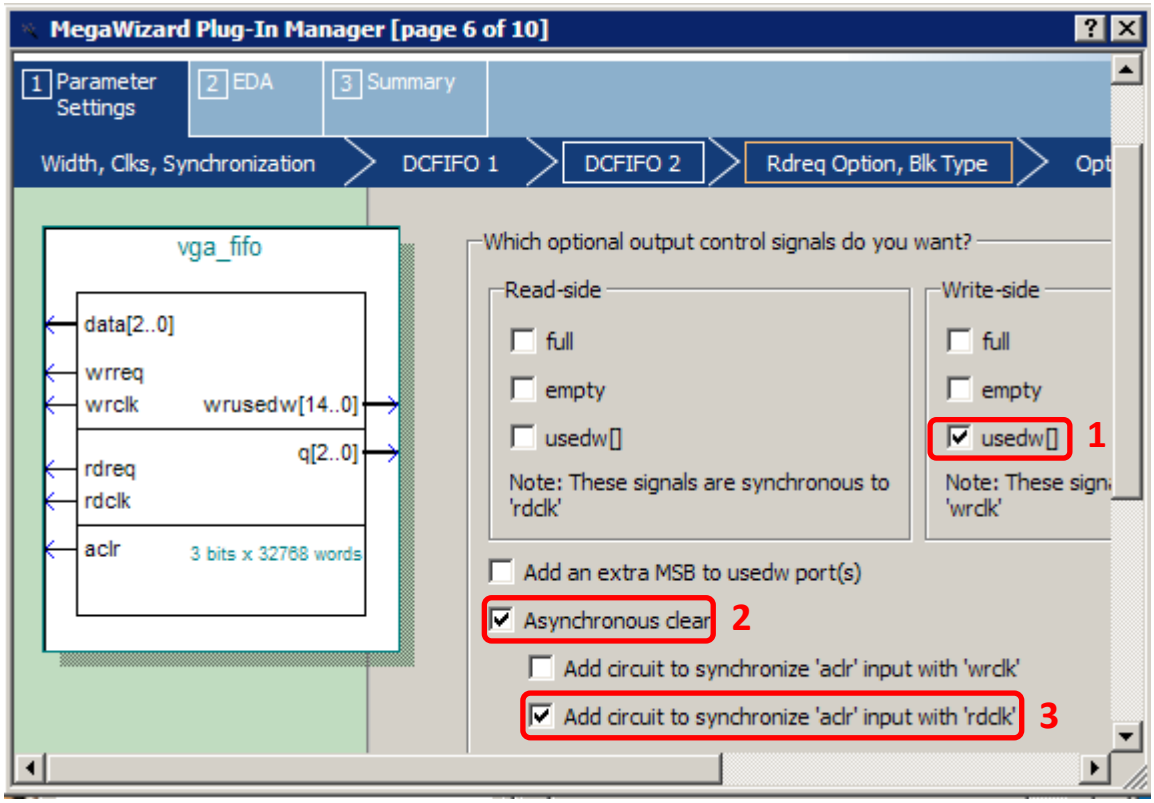
15、将 FIFO 文件的输出设置为“3”位，深度设置为“32768”，读和写的频率设置为不是同一个频率，然后点击“Next”。如下图所示：



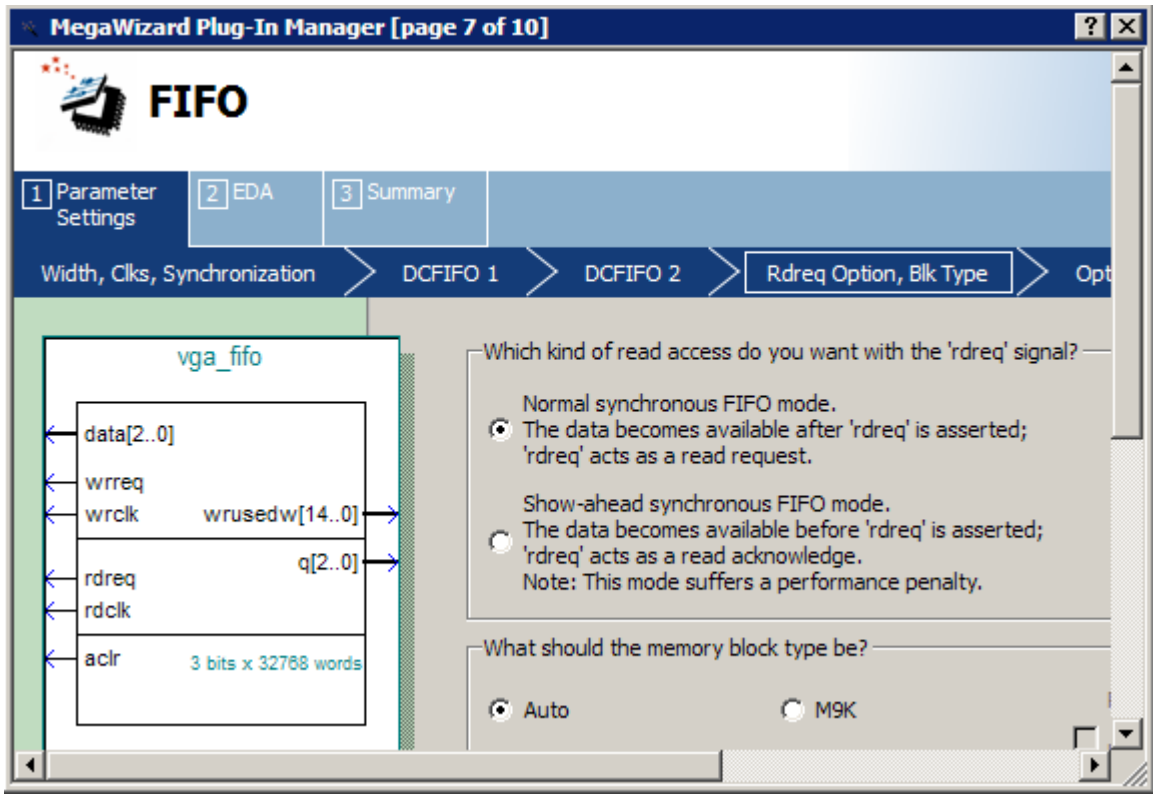
16、按照图中红框内的内容修改参数，然后点击“Next”。如下图所示：



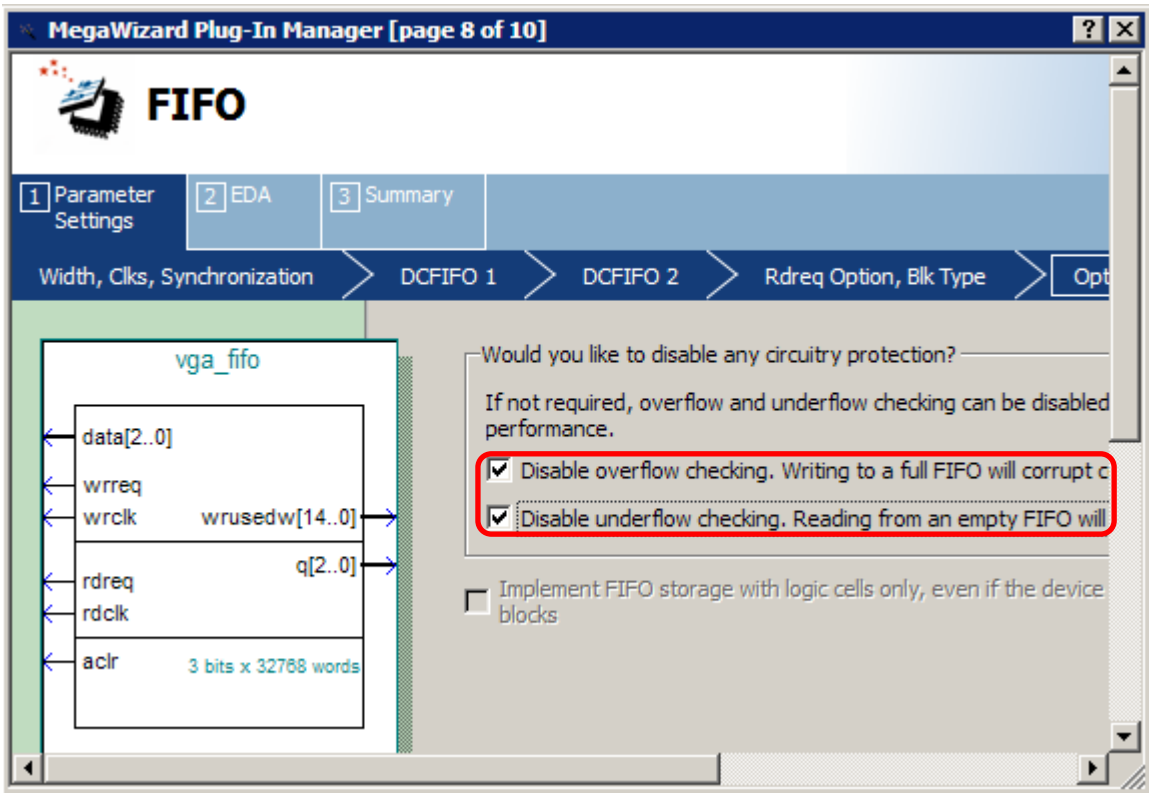
17、按照图中红框内的内容修改参数，然后点击“Next”。如下图所示：



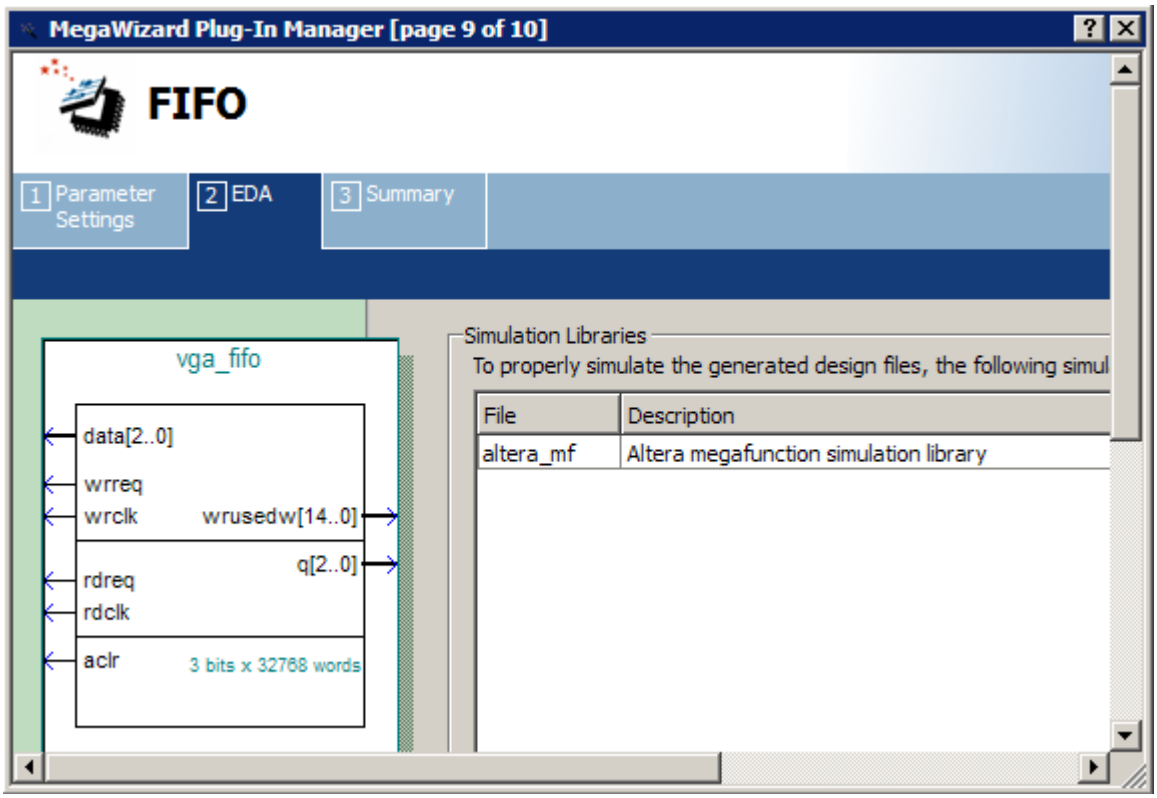
18、此步不作任何修改，直接点击“Next”。如下图所示：



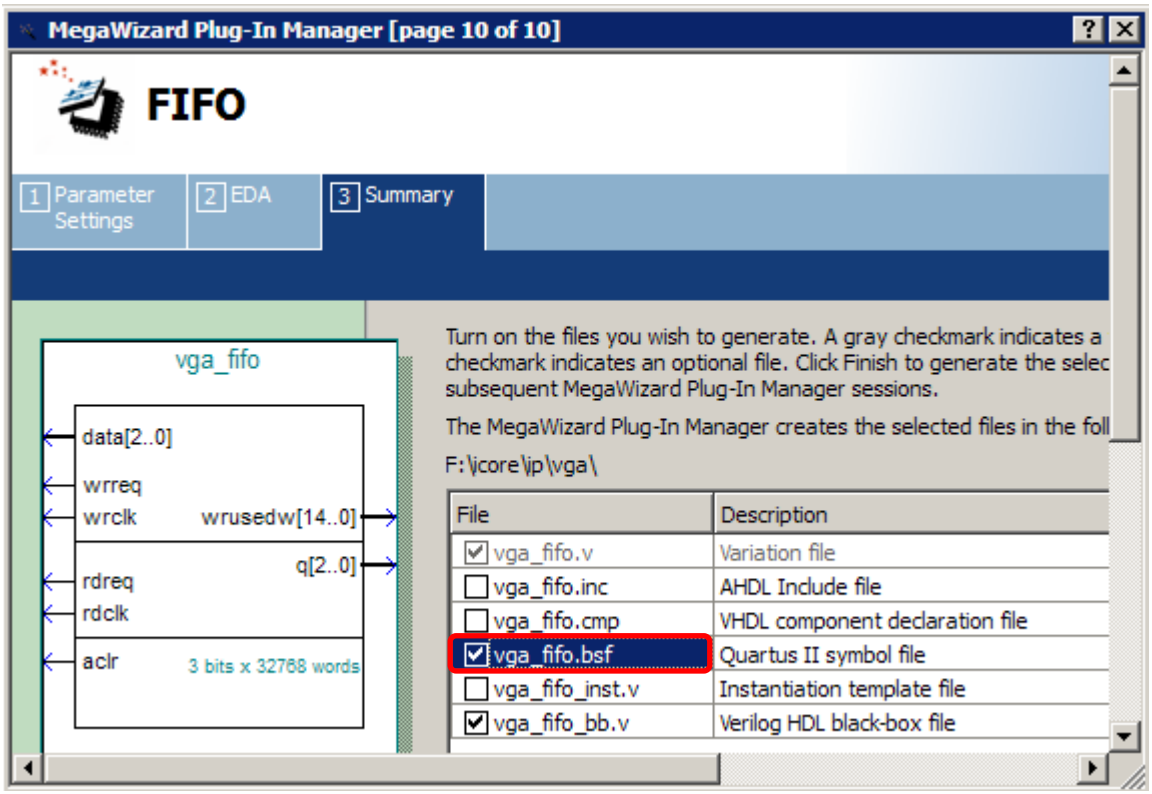
19、按照图中红框内的内容修改参数，然后点击“Next”。如下图所示：



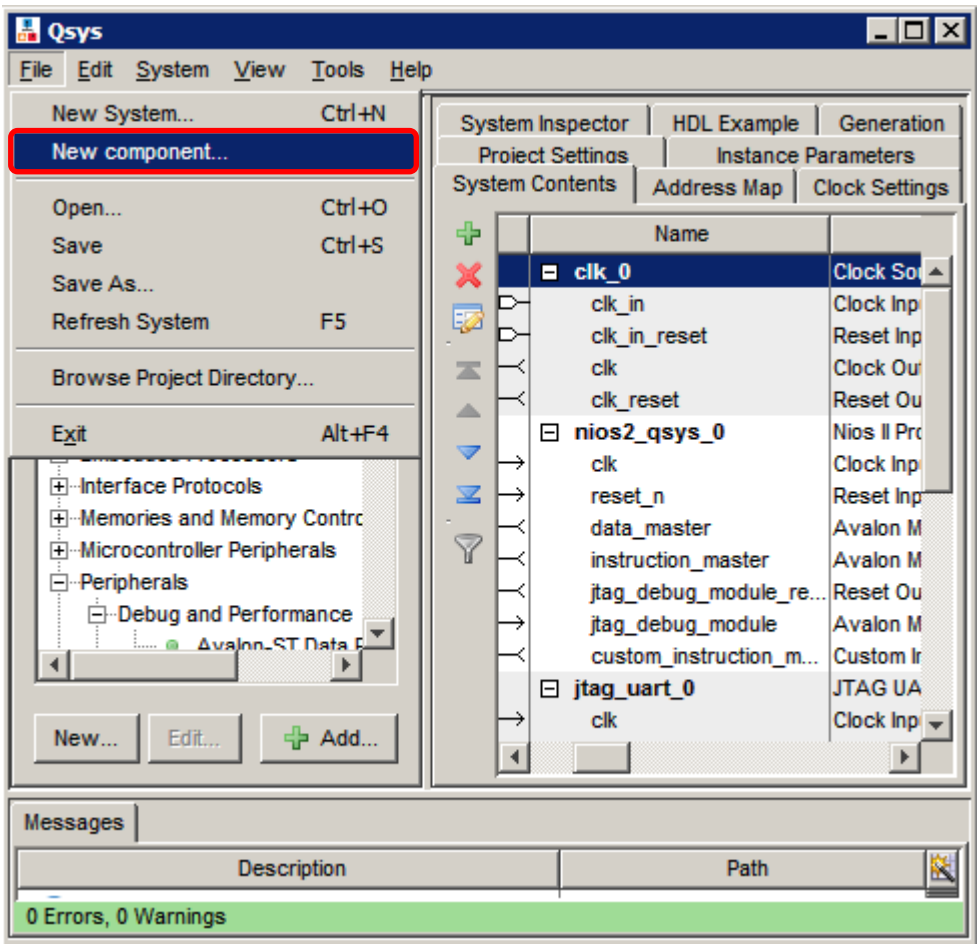
20、此步不作任何修改，直接点击“Next”。如下图所示：



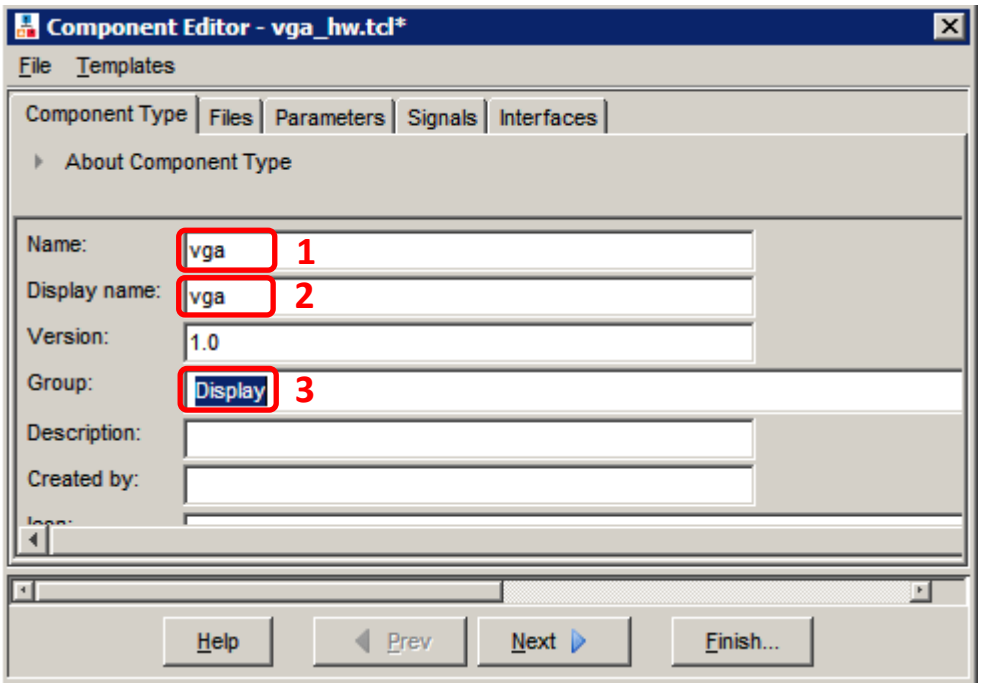
21、按照图中红框内的内容修改参数，然后点击“Finish”。如下图所示：



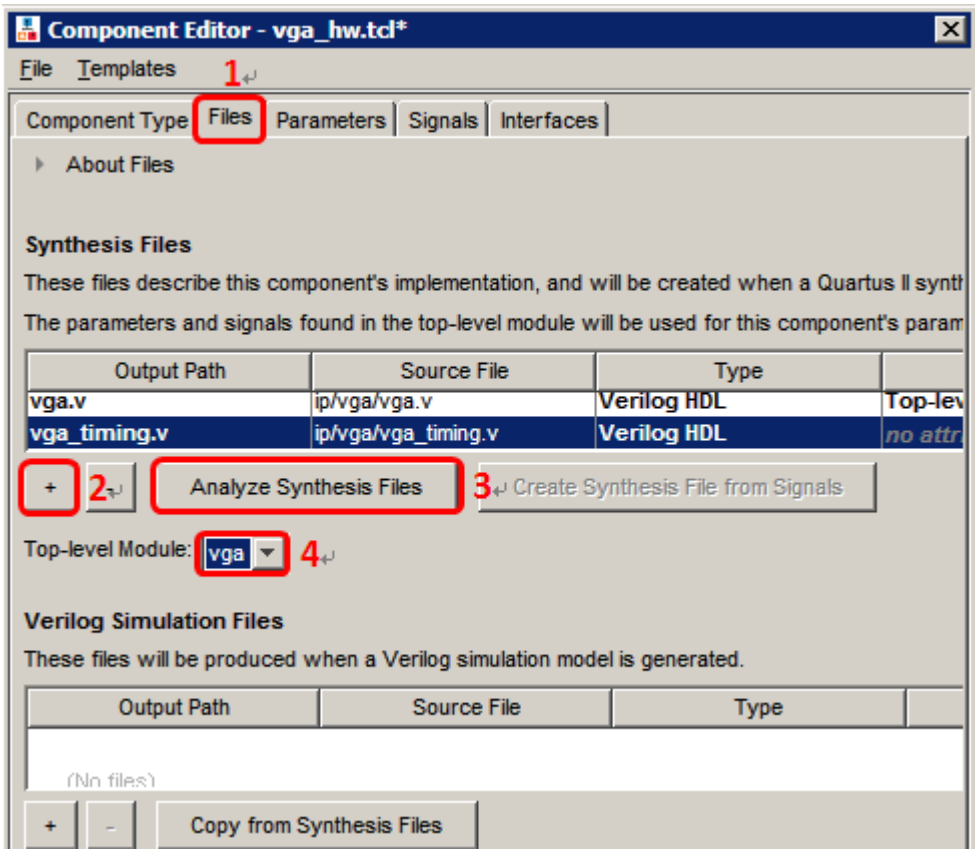
22、回到 Qsys 界面下，点击“File/new component”。如下图所示：



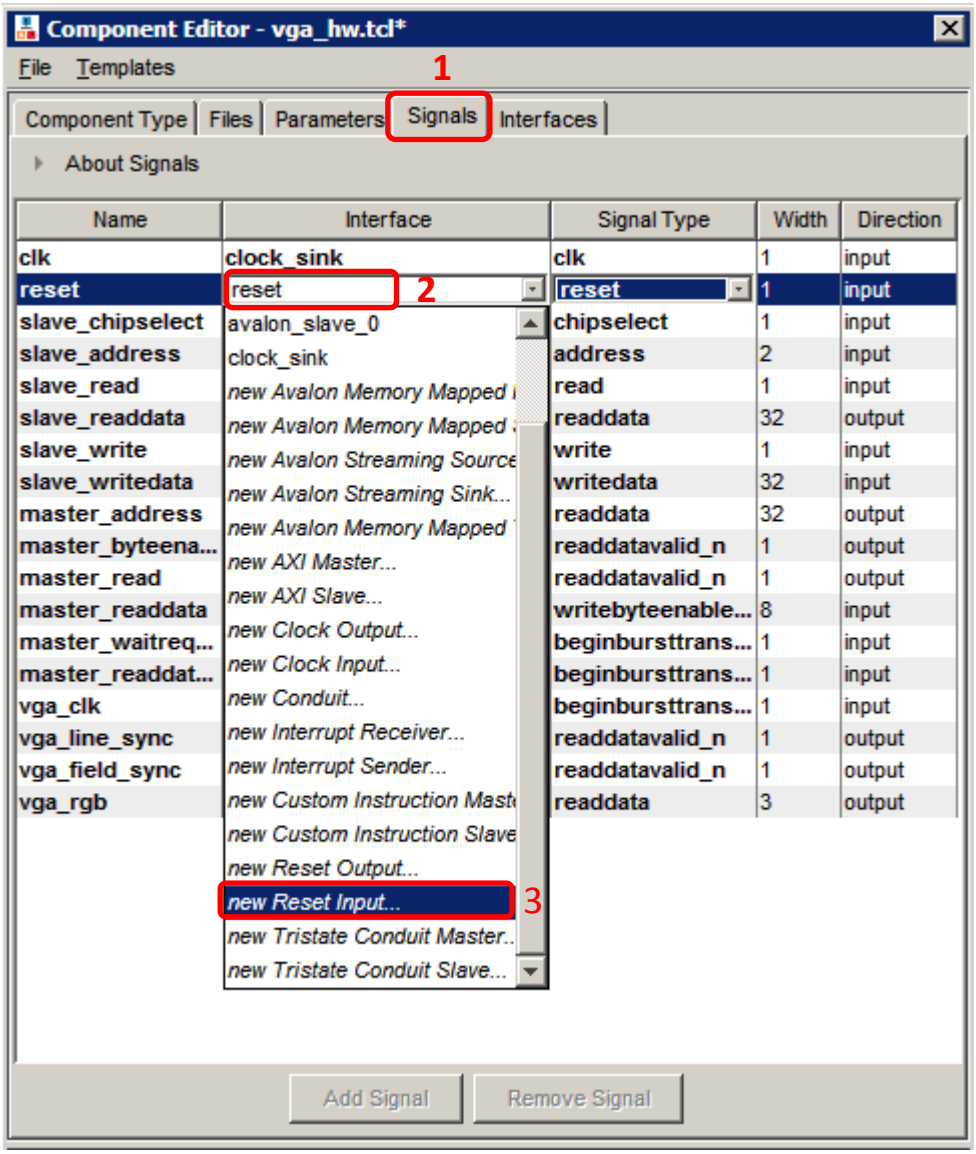
23、按照图中红框内的内容修改参数。如下图所示：



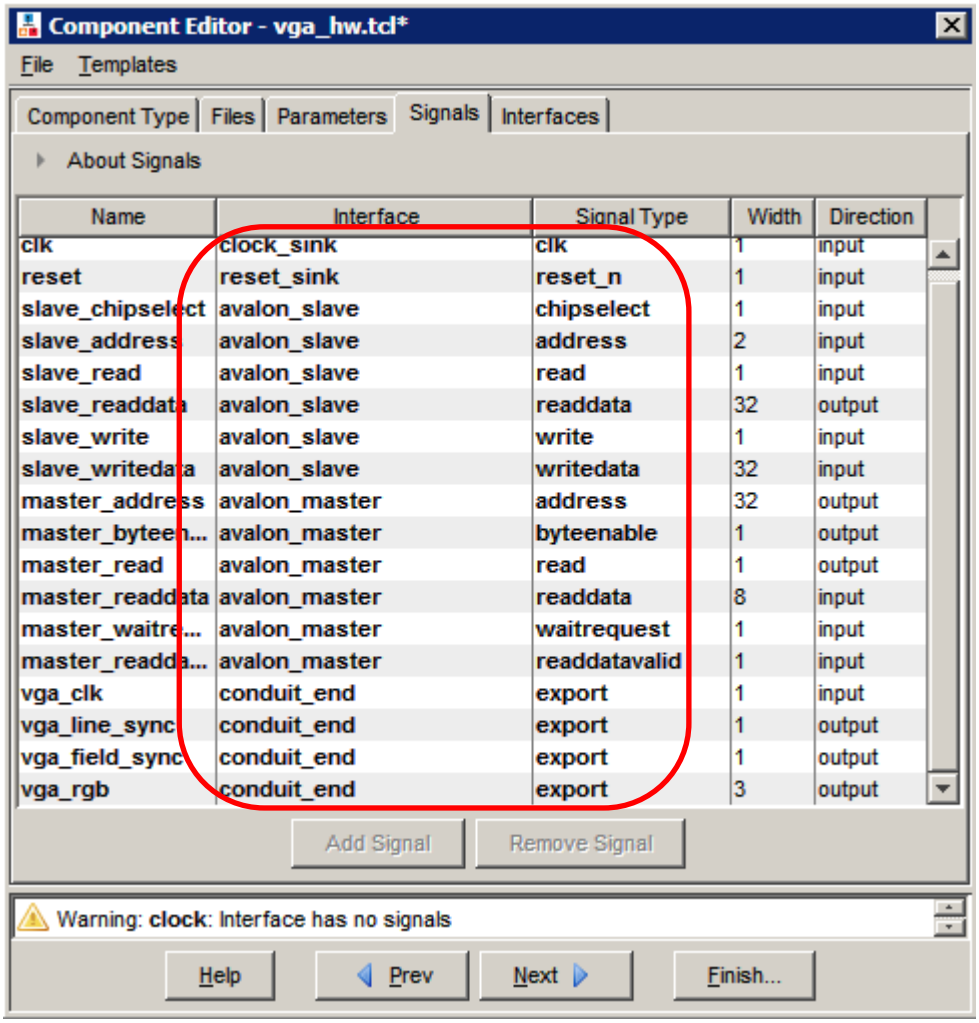
24、首先点击“1”处的“Files”。然后点击“2”处的“+”号，把 F/icore/ip/vga 文件夹下的 vga.v 和 timing.v 两个文件添加到 Synthesis Files 内，再点击“3”进行综合分析，最后将“4”设置成“vga”。如下图所示：



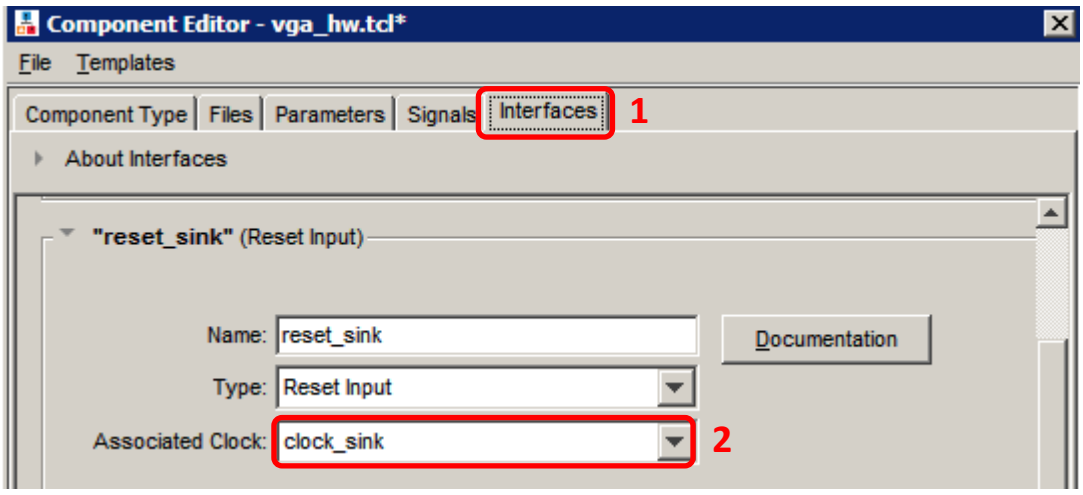
25、点击“Signals”，设置 vga 的时钟、复位等相关参数。如下图所示：

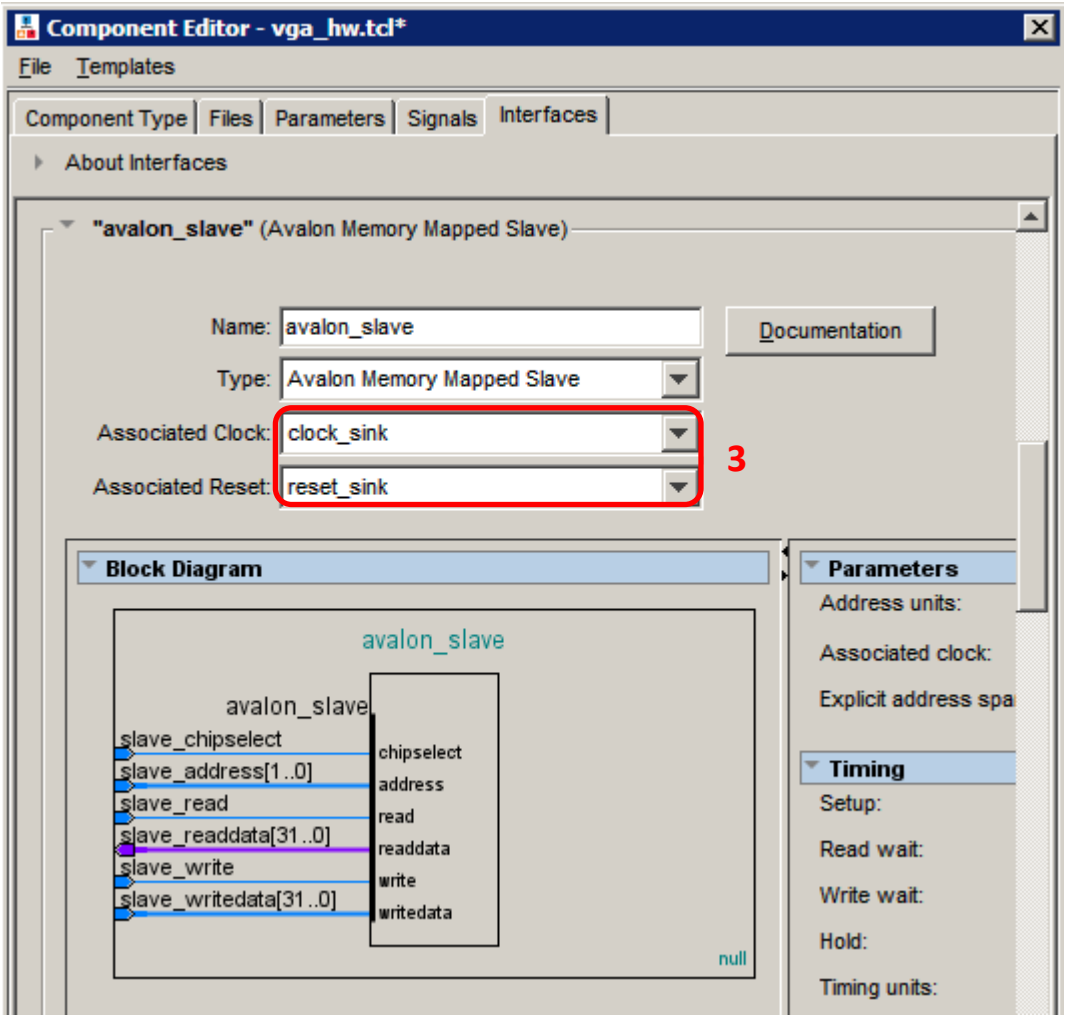


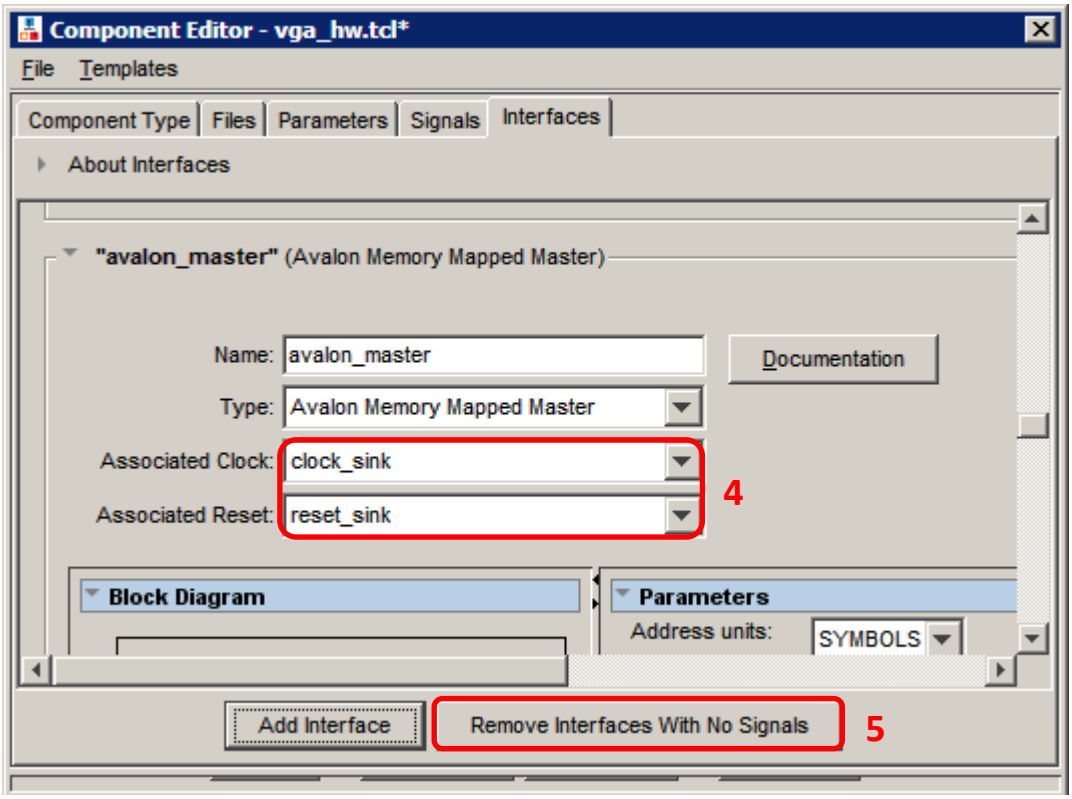
26、将 vga 的相关参数设置成红框内的结果。如下图所示：



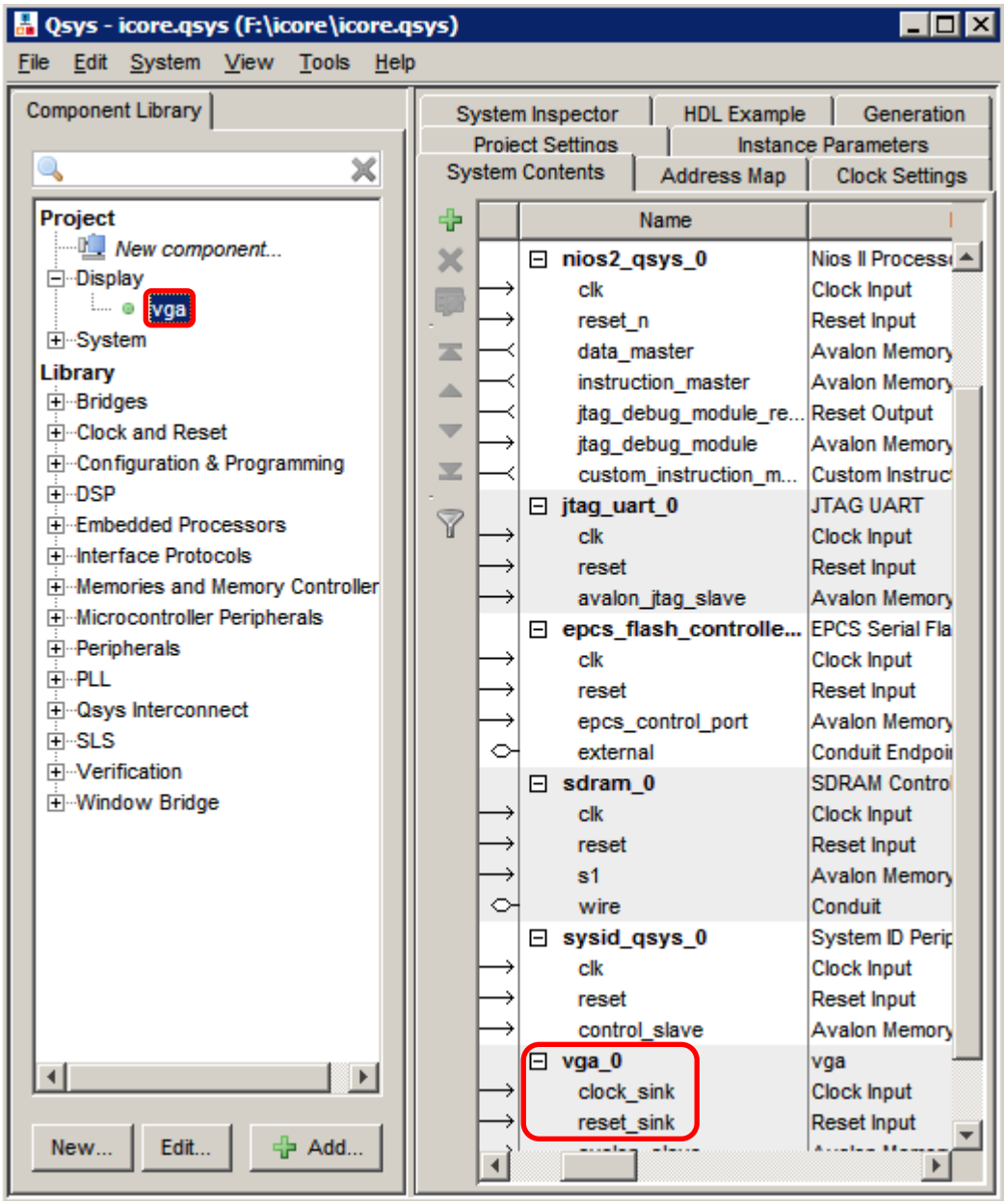
27、点击“Interfaces”，按照图中画红框的内容修改参数，然后点击“Remove interfcces with no signals”，最后点击“Finish”。如下图所示：



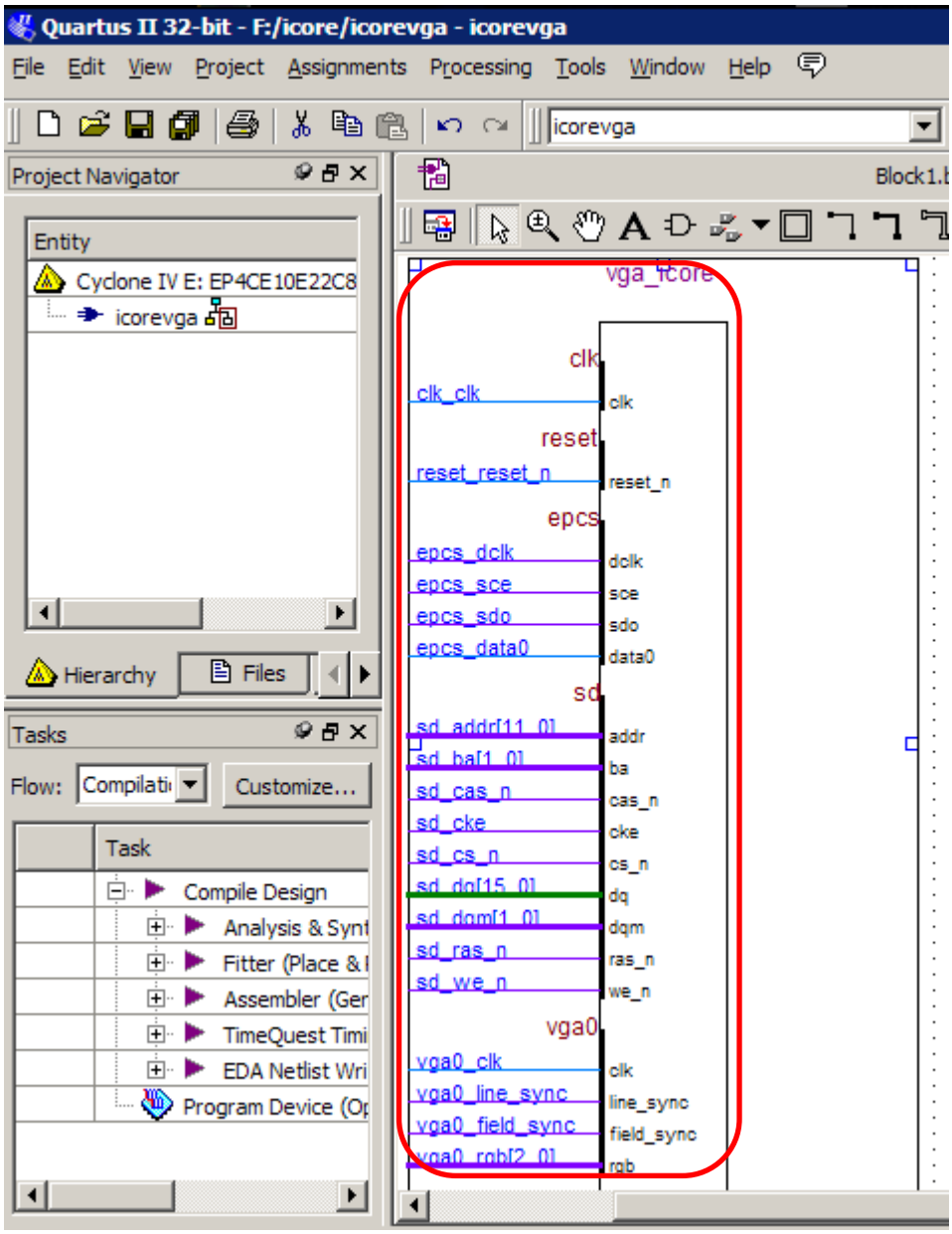

Component Editor - vga_hw.tcl*
File
Templates
Component Type
Files
Parameters
Signals
Interfaces
About Interfaces
"avalon_slave" (Avalon Memory Mapped Slave)
Name:
avalon_slave
Documentation
Type:
Avalon Memory Mapped Slave
Associated Clock:
clock_sink
Associated Reset:
reset_sink
3
Block Diagram
avalon_slave
avalon_slave
slave_chipselect
slave_address[1..0]
slave_read
slave_readdata[31..0]
slave_write
slave_writedata[31..0]
chipselect
address
read
readdata
write
writedata
null
Parameters
Address units:
Associated clock:
Explicit address spa
Timing
Setup:
Read wait:
Write wait:
Hold:
Timing units:



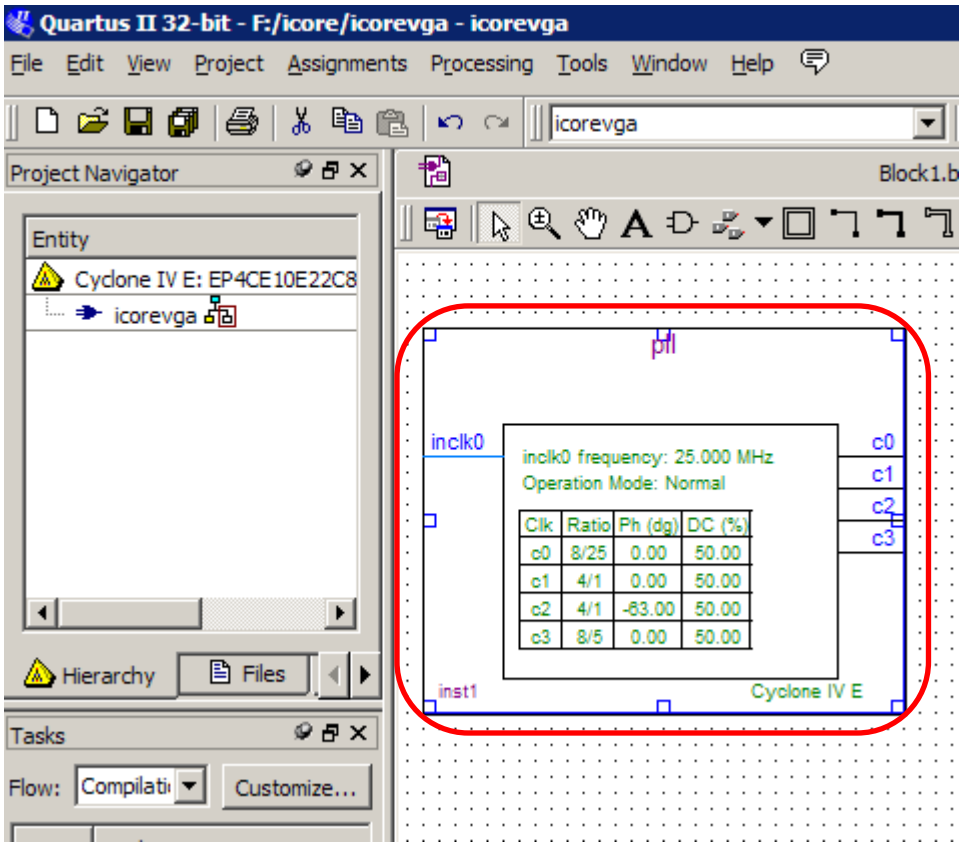
28、按照上述的设置，Qsys 界面下就会出现一个 Display，这样 vga 核到此就构建完成。然后把 vga 核添加到 Nios 软核内，再设置时钟、复位、中断等相关设置，必须达到零错误零警告。最后对 Nios 软核进行编译，编译完成后退出即可。如下图所示：



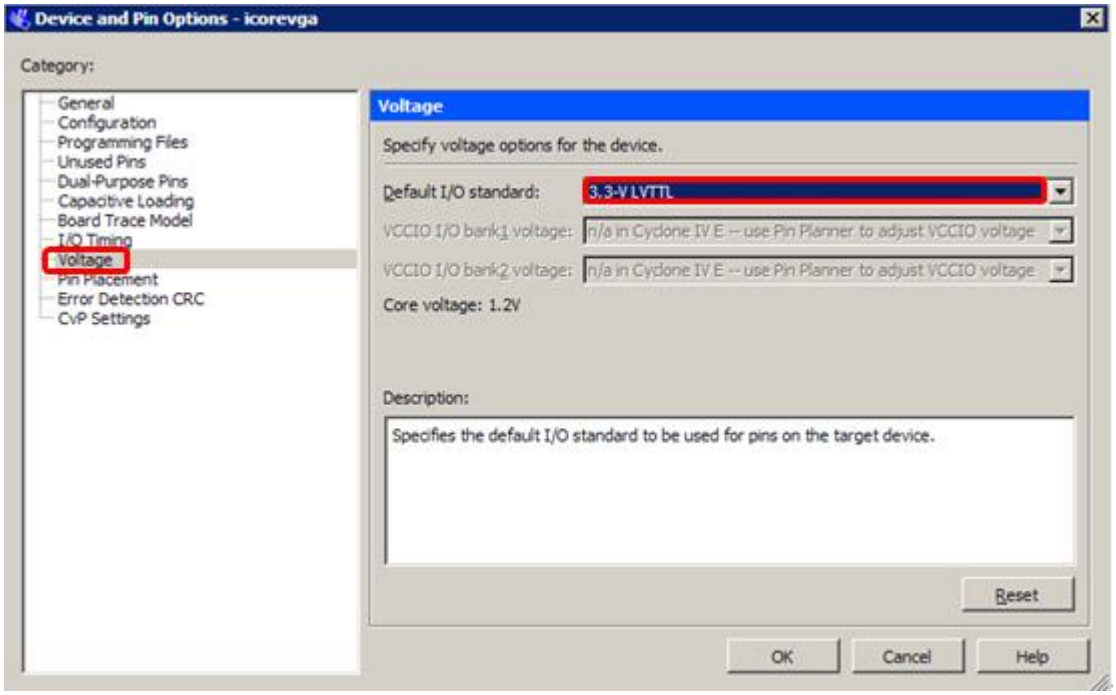
29、将 vga_icore 软核导入 Block.bdf 文件。如下图所示：



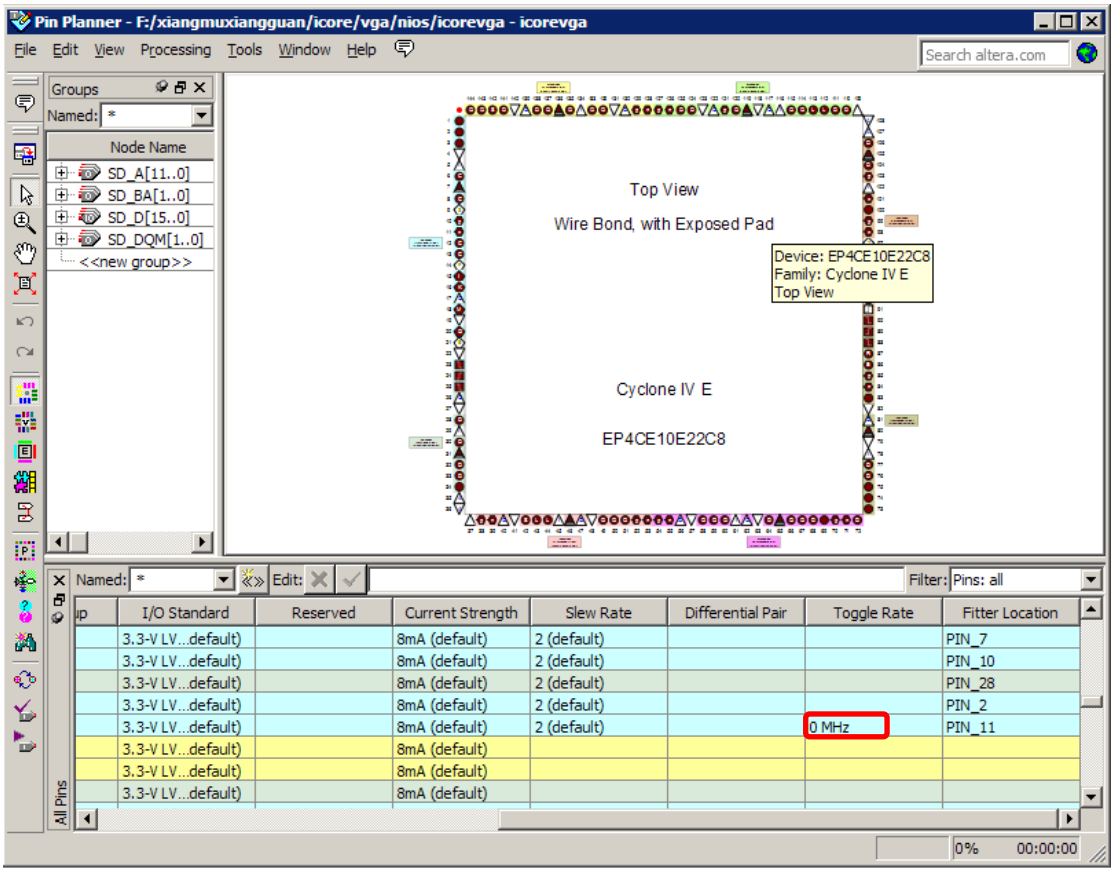
30、构建锁相环（操作步骤不作详细介绍）。锁相环输出四个时钟频率，分别为 arm、nios、vga 和 sdram 提供时钟。如下图所示：



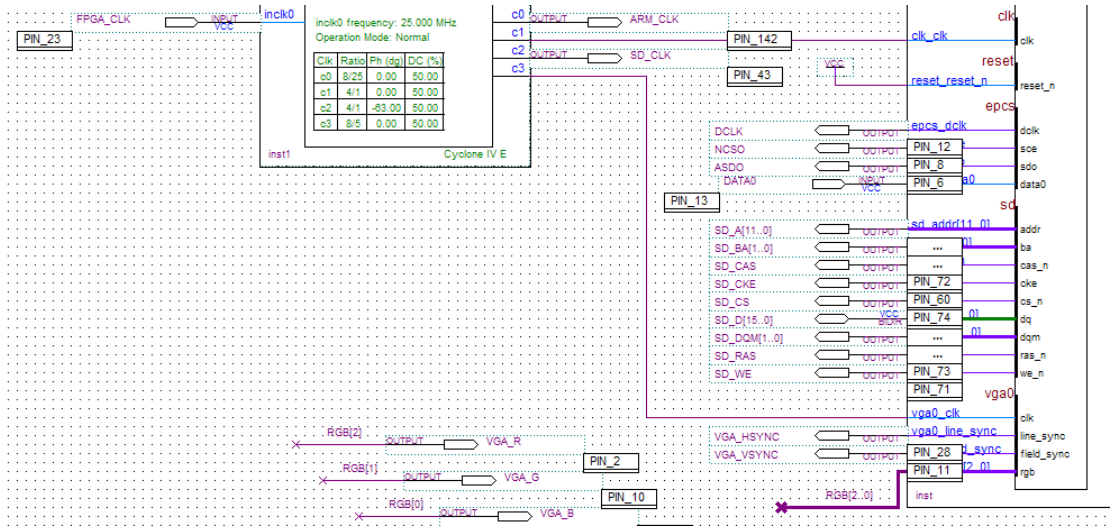
31、点击“Assignments/Device/ Device and Pin Options”设置 I/O 口的标准电压值为 3.3-VLTTL。（其他相关设置不作详细介绍，详见 icore 开发板新建工程文档）。如下图所示：



32、设置 11 引脚的“Toggle Rate”为 0Hz。如下图所示：

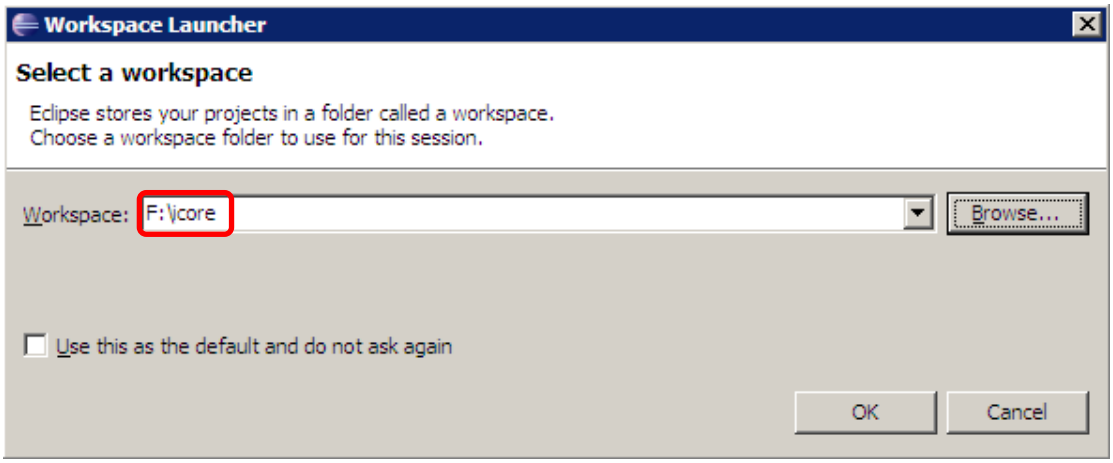


33、分配引脚(相关操作不作详细介绍 ,详见 icore 开发板新建工程文档), 最后编译、下载。如下图所示：

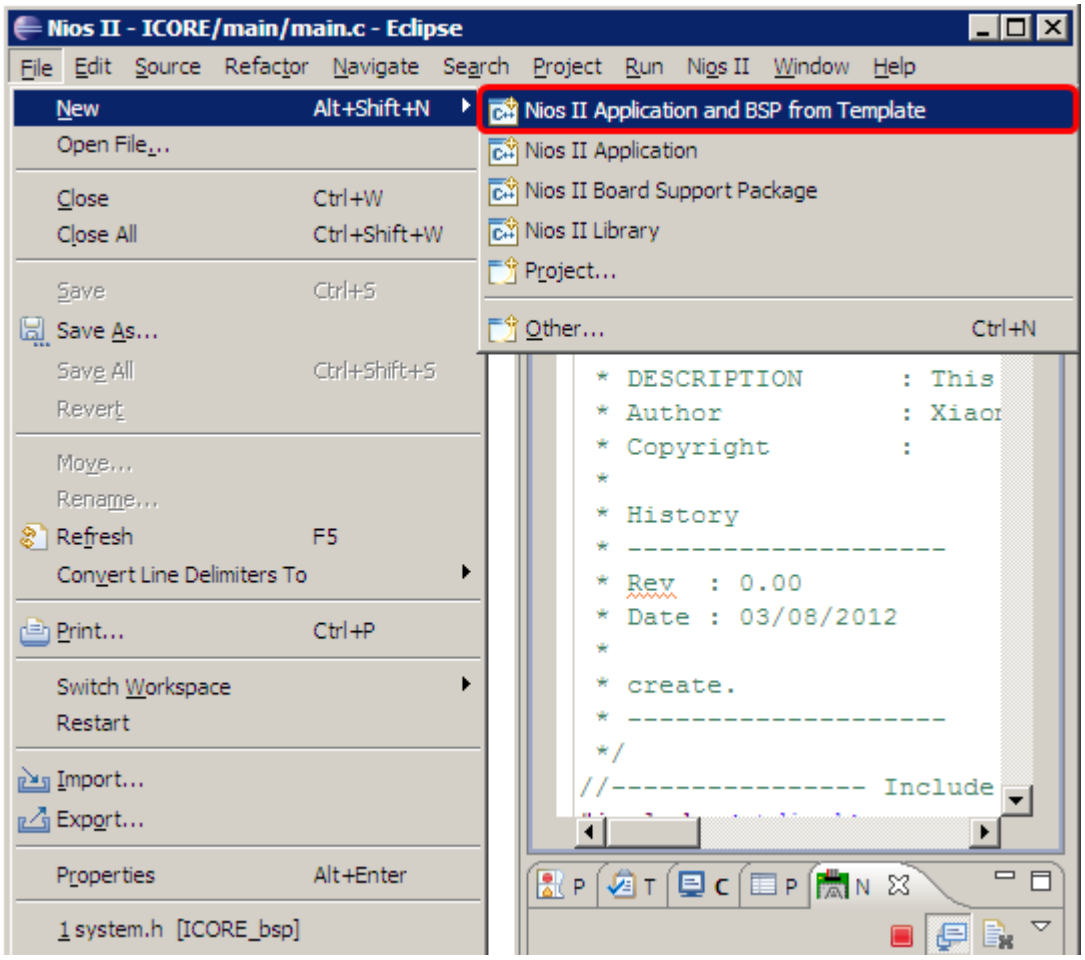


第三章 编写 NiosII 程序

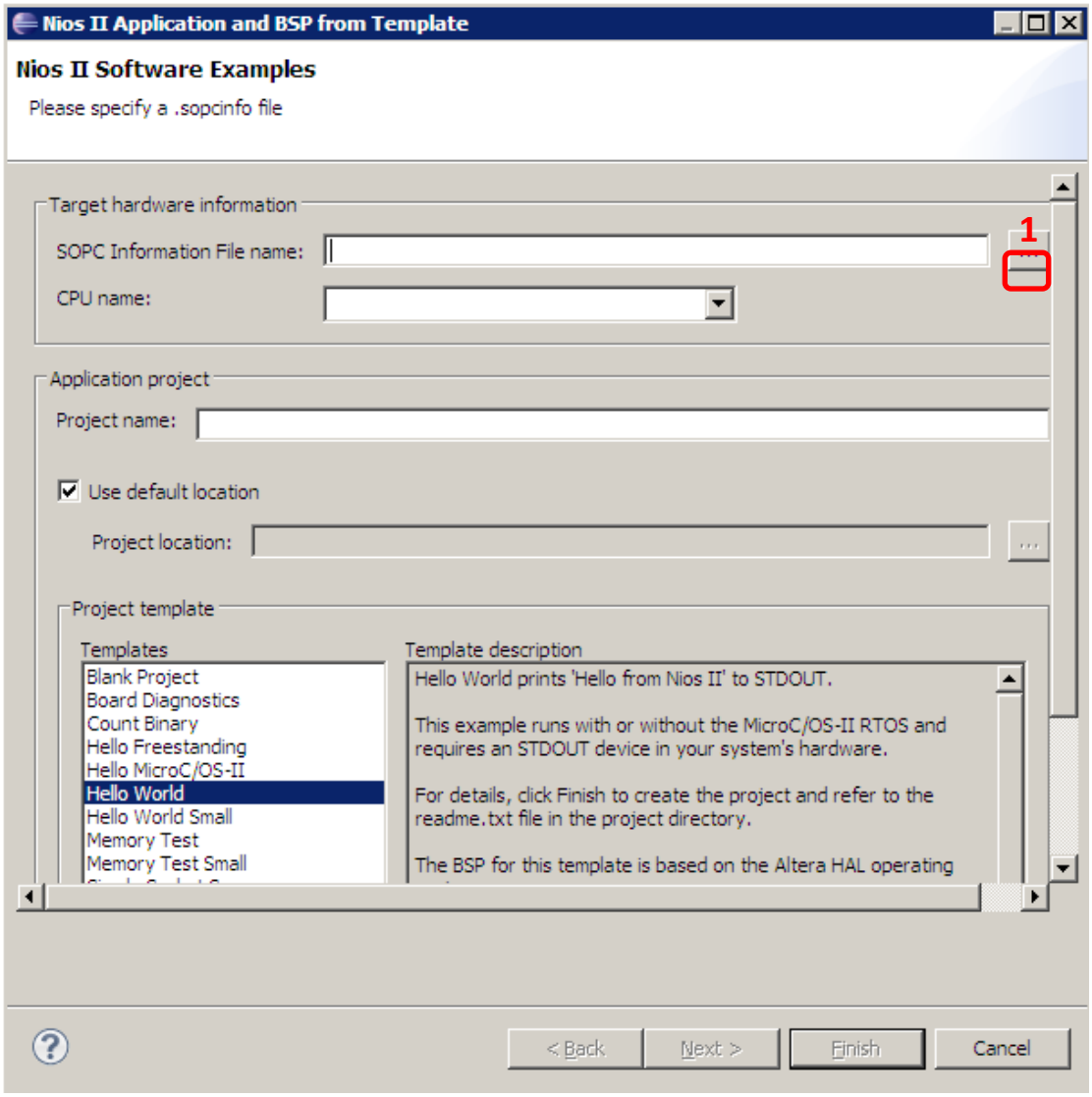
1、打开 NiosII 12.0，将 Nios 的工作空间指向 F/icore 下。如下图所示：



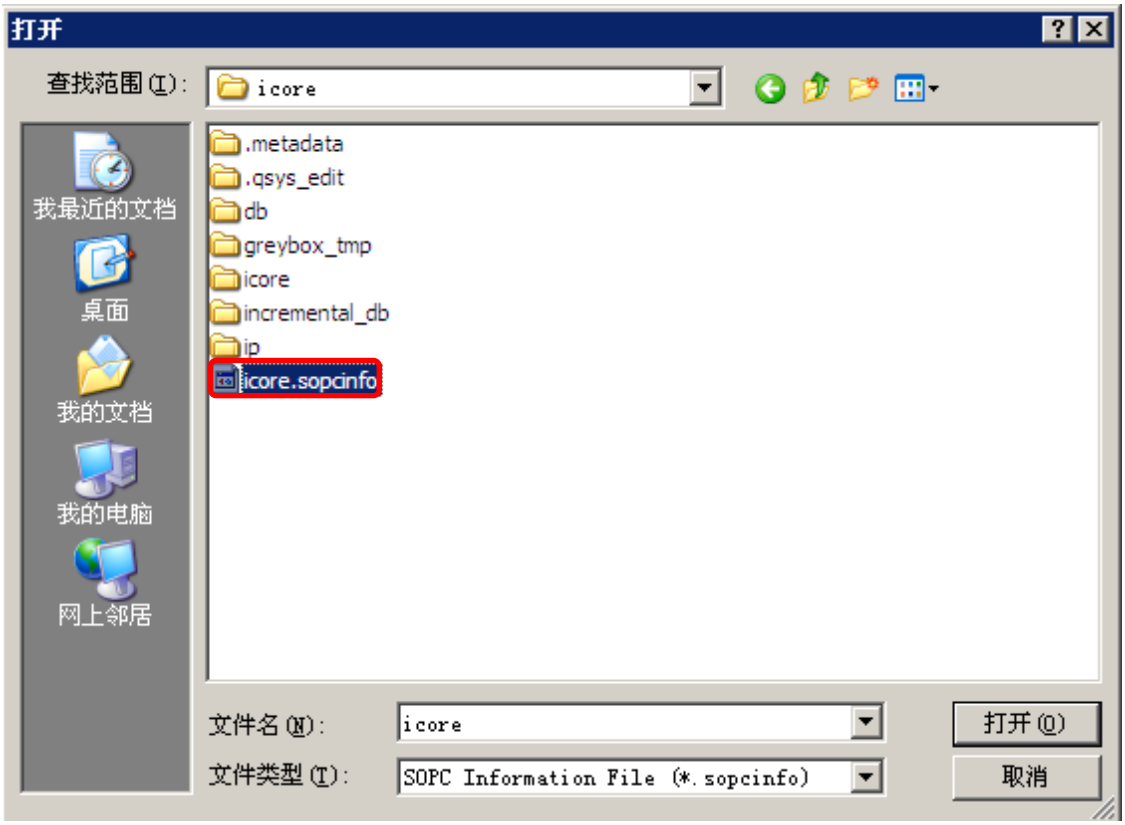
2、点击“File/new/Nios II Application and BSP from Template”。如下图所示：


Nios II - ICORE/main/main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Nios II Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename...
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 system.h [ICORE_bsp]
Nios II Application and BSP from Template
Nios II Application
Nios II Board Support Package
Nios II Library
Project...
Other... Ctrl+N
* DESCRIPTION : This
* Author : Xiao
* Copyright :
*
* History
* ---
* Rev : 0.00
* Date : 03/08/2012
*
* create.
* ---
*/
//---------- Include

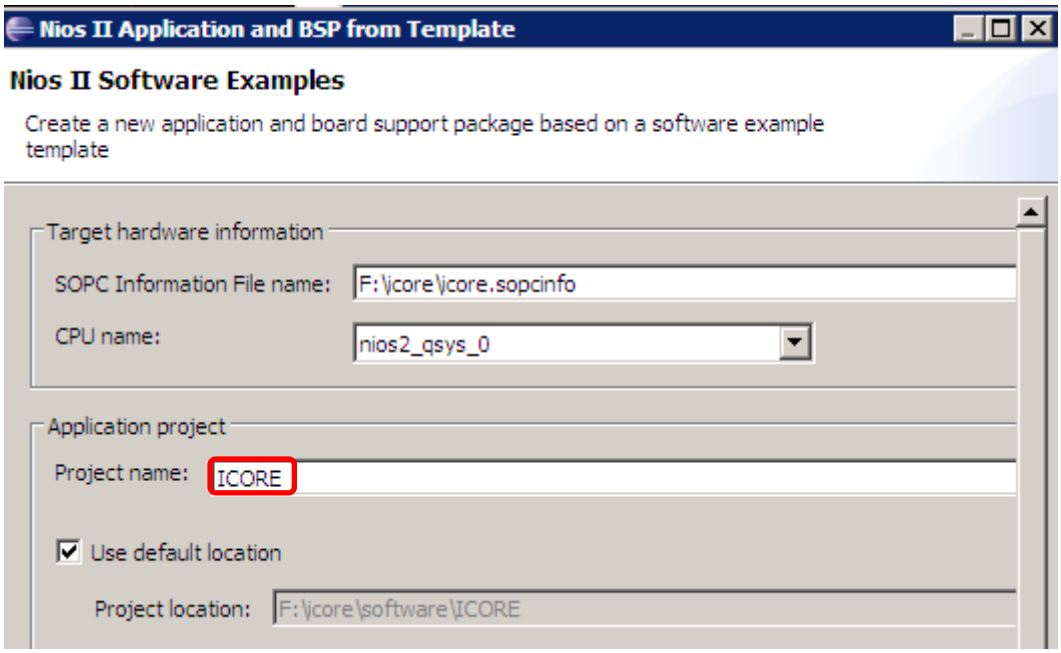
3、点击“1”处。如下图所示：



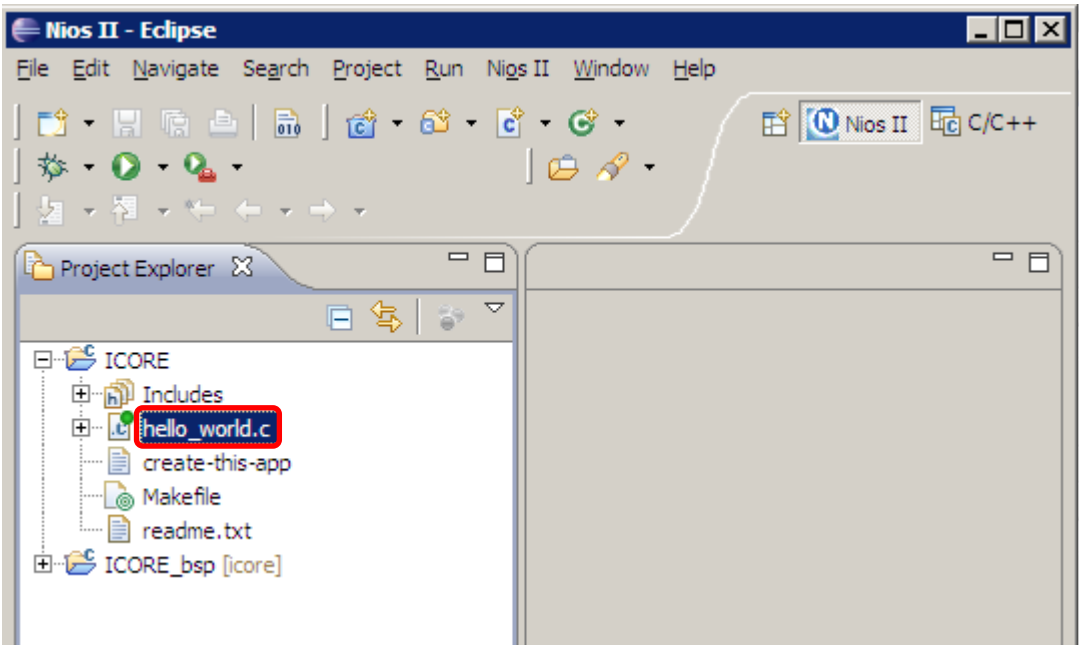
4、点击“File/icore/icore.sopcinfo”，然后点击“打开”。如下图所示：



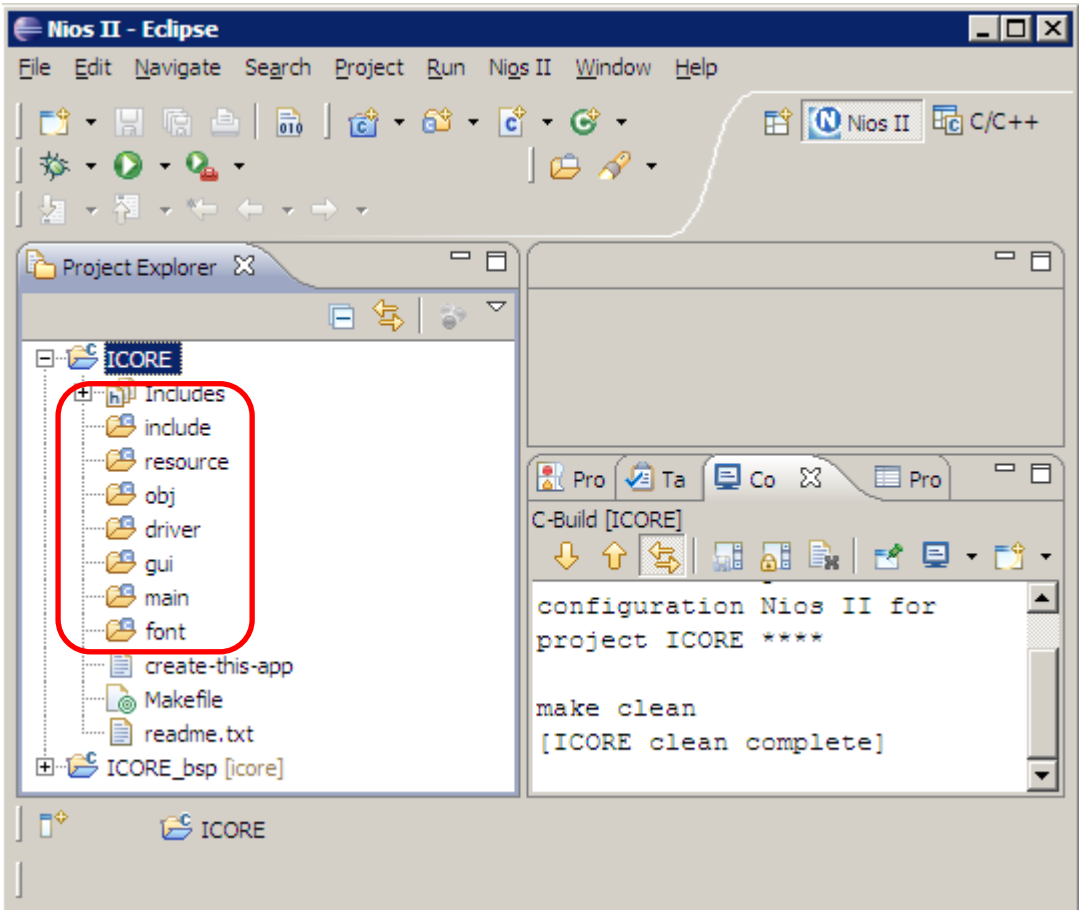
5、将 Nios II 工程命名为“ICORE”。如下图所示：



6、将“ICORE/hello_world.c”重新命名为“main.c”。如下图所示：

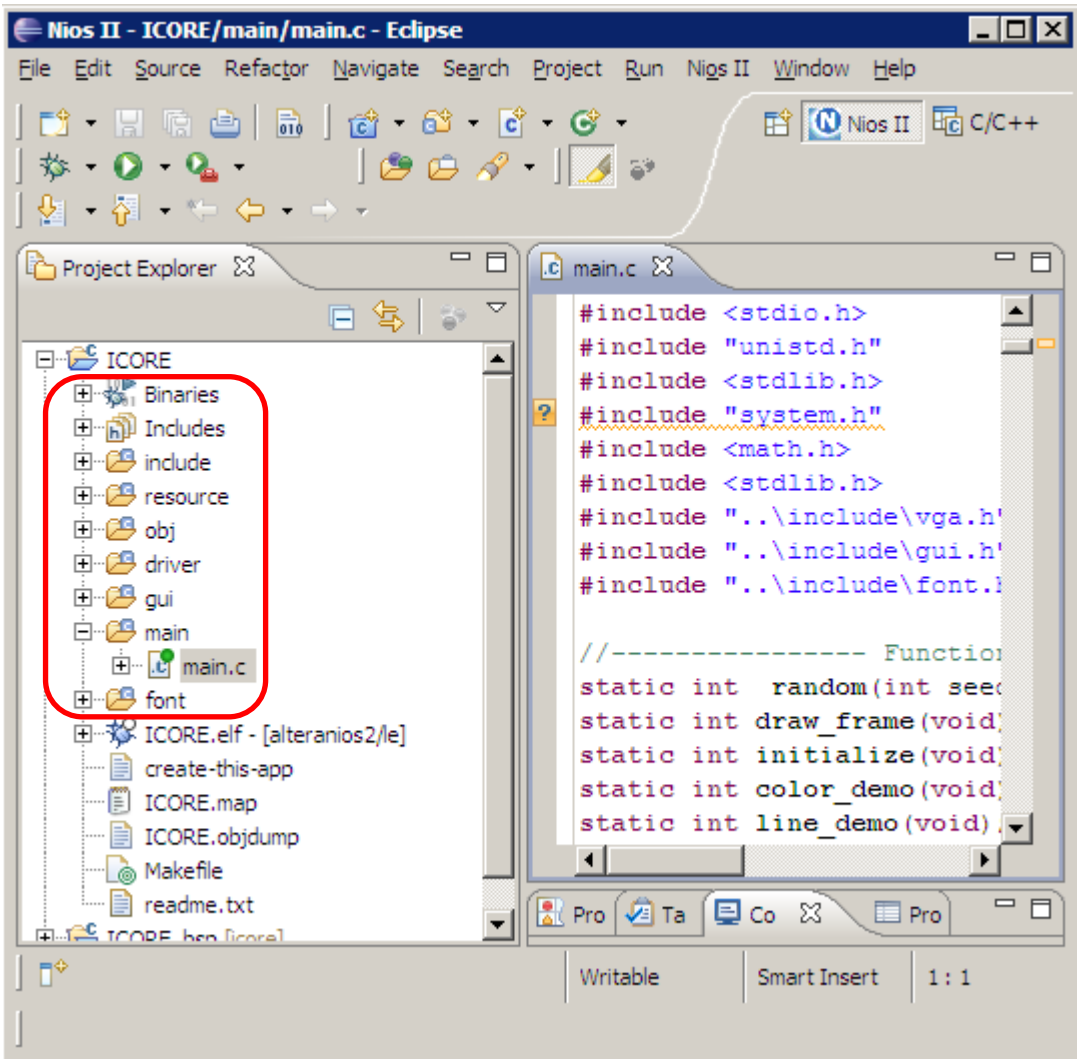


7、在 ICORE 文件夹下新建相关文件夹。如下图所示：



8、把相关的代码写入对应文件夹内（相关操作不作详细介绍，详见 icore

开发板新建工程文档），如下图所示：



9、编译程序代码，不出现任何错误然后下载到开发板内即可（相关操作不作详细介绍，详见 icore 开发板新建工程文档）。

第四章 程序解析

本文档主要解析 gui.c 和 main.c 中的程序代码：解析 gui.c 中的底层代码实现什么样的功能及解析 main.c 实现在显示器上显示的功能。

```
56 static int
57 line(int x1, int y1, int x2, int y2, int c)
58 {
59     int dx = x2 - x1;
60     int dy = y2 - y1;
61     int ux = ((dx > 0) << 1) - 1; //x 的增量方向，取或-1
62     int uy = ((dy > 0) << 1) - 1; //y 的增量方向，取或-1
63     int x = x1, y = y1, eps;      //eps 为累加误差
64
65     eps = 0; dx = abs(dx); dy = abs(dy);
66     if (dx > dy) {
67         for (x = x1; x != x2; x += ux) {
68             vga.dot(x, y, c);
69             eps += dy;
70             if ((eps << 1) >= dx) {
71                 y += uy; eps -= dx;
72             }
73         }
74     }else {
75         for (y = y1; y != y2; y += uy) {
76             vga.dot(x, y, c);
77             eps += dx;
78             if ((eps << 1) >= dy) {
79                 x += ux; eps -= dy;
80             }
81         }
82     }
83
84     return 0;
85 }
86
87 /*
88  * Name             : rect
89  * Description      : ---
90  * Author           : XiaomaGee.
91  *
92  * History
```

```
 93  * ----------------------
 94  * Rev  : 0.00
 95  * Date : 03/08/2012
 96  *
 97  * create.
 98  * ----------------------
 99  */
100 static int
101 rect(RECT_T * r)
102 {
103     int i;
104
105     if (r->fill_flag == 1) {
106         for (i = 0; i < r->height; i++) {
107             line(r->x, r->y + i, r->x + r->width, r->y + i,
r->color);
108         }
109     }else { //no fill
110         line(r->x, r->y, r->x + r->width - 1, r->y, r->color);
111         line(r->x, r->y + r->height - 1, r->x + r->width - 1, r->y
+ r->height - 1, r->color);
112         line(r->x, r->y, r->x, r->y + r->height - 1, r->color);
113         line(r->x + r->width - 1, r->y, r->x + r->width - 1, r->y
+ r->height, r->color);
114     }
115
116     return 0;
117 }
118 /*
119  * Name             : _draw_circle_8
120  * Description      : ---
121  * Author           : XiaomaGee.
122  *
123  * History
124  * ----------------------
125  * Rev  : 0.00
126  * Date : 03/08/2012
127  *
128  * create.
129  * ----------------------
130  */
131 static
132 void _draw_circle_8(int xc, int yc, int x, int y, int c)
133 {
```

```
134     vga.dot(xc + x, yc + y, c);
135     vga.dot(xc - x, yc + y, c);
136     vga.dot(xc + x, yc - y, c);
137     vga.dot(xc - x, yc - y, c);
138     vga.dot(xc + y, yc + x, c);
139     vga.dot(xc - y, yc + x, c);
140     vga.dot(xc + y, yc - x, c);
141     vga.dot(xc - y, yc - x, c);
142 }
143
144 /*
145  * Name             : circle
146  * Description      : ---
147  * Author           : XiaomaGee.
148  *
149  * History
150  * ----------------------
151  * Rev  : 0.00
152  * Date : 03/08/2012
153  *
154  * create.
155  * ----------------------
156  */
157 static int
158 circle(int xc, int yc, int r, int fill, int c)
159 {
160     int x = 0, y = r, yi, d;
161     d = 3 - 2 * r;
162
163     if (fill) {
164         // 如果填充（画实心圆）
165
166         while (x <= y) {
167             for (yi = x; yi <= y; yi++)
168                 _draw_circle_8(xc, yc, x, yi, c);
169
170             if (d < 0) {
171                 d = d + 4 * x + 6;
172             } else {
173                 d = d + 4 * (x - y) + 10;
174                 y--;
175             }
176             x++;
177         }
```

```
178     } else {
179         // 如果不填充（画空心圆）
180 
181         while (x <= y) {
182             _draw_circle_8(xc, yc, x, y, c);
183             if (d < 0) {
184                 d = d + 4 * x + 6;
185             } else {
186                 d = d + 4 * (x - y) + 10;
187                 y--;
188             }
189             x++;
190         }
191     }
192     return 0;
193 }
194 
195 /*
196  * Name             : line_to
197  * Description      : ---
198  * Author           : XiaomaGee.
199  *
200  * History
201  * ----------------------
202  * Rev  : 0.00
203  * Date : 03/08/2012
204  *
205  * create.
206  * ----------------------
207  */
208 static int
209 line_to(int x, int y, int c)
210 {
211     line(cursor_x, cursor_y, x, y, c);
212     set_cursor(x, y);
213 
214     return 0;
215 }
216 /*
217  * Name             : set_cursor
218  * Description      : ---
219  * Author           : XiaomaGee.
220  *
```

```
221  * History
222  * ----------------------
223  * Rev   : 0.00
224  * Date  : 03/08/2012
225  *
226  * create.
227  * ----------------------
228  */
229 static int
230 set_cursor(int x, int y)
231 {
232     cursor_x = x;
233     cursor_y = y;
234
235     return 0;
236 }
```

以上程序是 gui 的底层代码，程序从第 56 行～第 85 行是函数 line(画线)的底层代码，程序从第 100 行～第 117 行是函数 rect(画矩形)的底层代码，程序从第 157 行～第 193 行是函数 circle(画圆)的底层代码 ,程序从第 209 行～第 215 行是函数 line_to(连线)的底层代码，程序从第 229 行～第 236 行是函数 set_cursor(设置光标)的底层代码。

```
196 static int
197 color_demo(void)
198 {
199     int i;
200     STRING_T s;
201     RECT_T r;
202     char  * color_string[] = {
203         "蓝色 BLUE", "绿色 GREEN", "青色 CYAN", "红色 RED", "品
红 FUCHSINE", "黄色 YELLOW", "白色 WHITE"
204     };
205
206     draw_frame();
207     for (i = 0; i < 7; i++) {
208         r.x = 30;
209         r.y = 70 * i + 80;
210         r.width = 500;
211         r.height = 60;
212         r.fill_flag = 1;
219
220         s.x = 550;
```

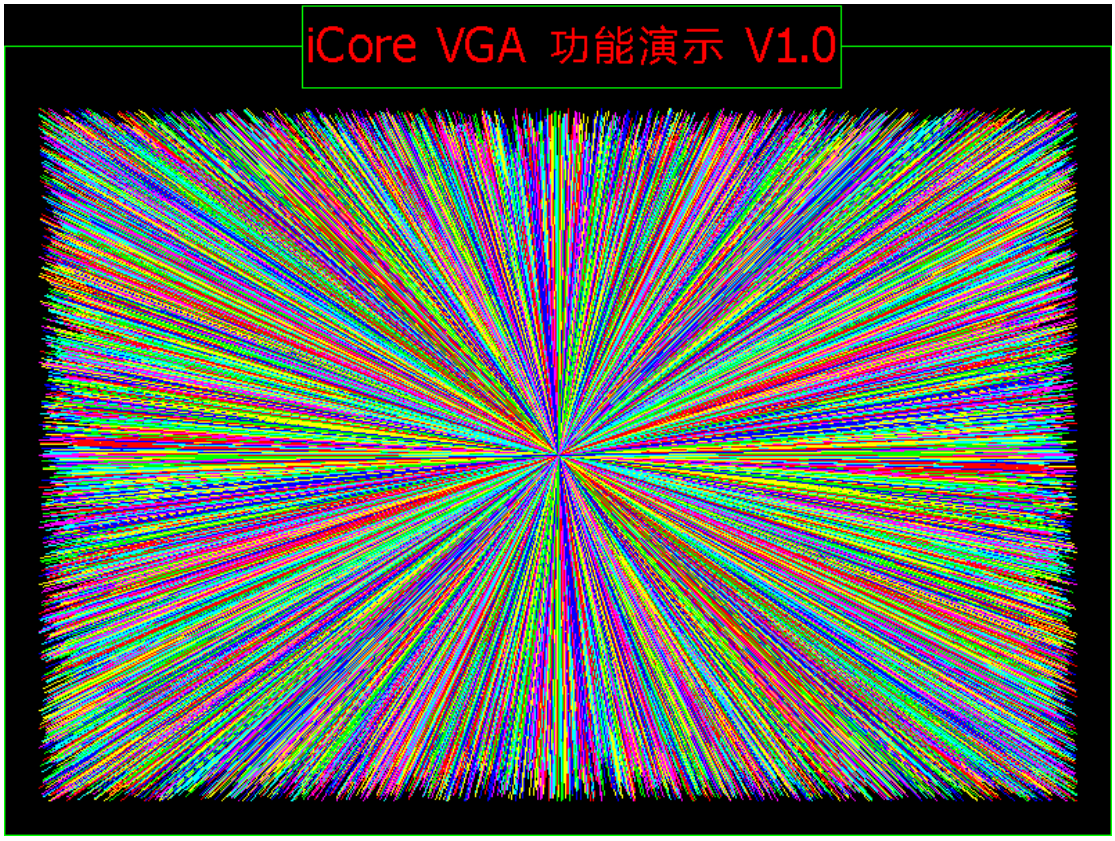
```
221         s.y = 70 * i + 90;
222         s.inverse = NULL;
223         s.color = i + 1;
224         s.background_color = COLOR_BLACK;
225         s.space.line = 5;
226         s.space.word = 0;
227         font.printf(&s, "%s", color_string[i]);
228     }
229     usleep(5000000);
230
231     return 0;
232 }
```

以上代码表示通过 vga 界面显示各种不同颜色的矩形条，程序运行结果如下图所示：


```
247 static int
248 line_demo(void)
249 {
250     int i, x0, y0, x1, y1;
251     int color;
```

```
252
253     draw_frame();
254
255     for (i = 0; i < 20000; i++) {
256         x0 = random(750);
257         y0 = random(500);
258
259         x1 = 750 - x0;
260         y1 = 500 - y0;
261
262         gui.line(x0 + 25, y0 + 75, x1 + 25, y1 + 75, i % 6 + 1);
263     }
264
265     return 0;
266 }
```

以上代码表示通过 vga 界面显示各种不同颜色的画线，程序运行结果如下图所示：

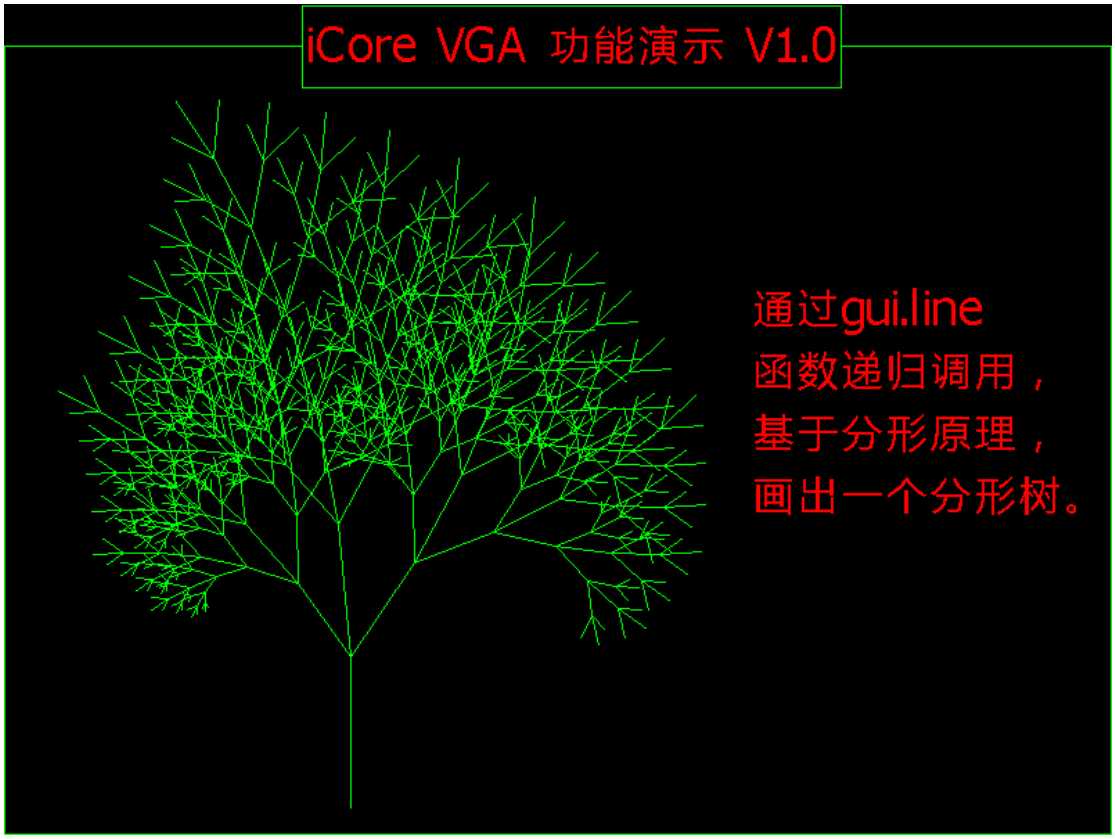



```
281 static int
282 _draw_tree(int x_start, int y_start, double length,
double angle, int num)
283 {
```

```
284     int x_end, y_end;
285 
286     if (num == 0) return 0;          // 画完
287 
288     x_end = x_start + (int)(length * cos(angle));
289     y_end = y_start - (int)(length * sin(angle));
290 
291     // 画树干
292     gui.line(x_start, y_start, x_end, y_end, COLOR_GREEN);
293 
294     // 画左叉树枝
295     _draw_tree(x_end, y_end,
296            length * 0.6,
297            angle + 0.624,
298            num - 1);
299 
300     // 画中叉树枝
301     _draw_tree(x_end, y_end, length * .88, angle + 0.1, num - 1);
302     _draw_tree(x_end, y_end, length * .75, angle - 0.6, num - 1);
303 
304     return 0;
305 }
306 /*
307  * Name                     : draw_tree
308  * Description      : ---
309  * Author                   : XiaomaGee.
310  *
311  * History
312  * ----------------------
313  * Rev  : 0.00
314  * Date : 03/05/2012
315  *
316  * create.
317  * ---------------------
318  */
319 static int
320 tree_demo(void)
321 {
322     int i;
323     STRING_T s;
324 
325     draw_frame();
326 
327     _draw_tree(250, 580, 110, 1.57, 7);
```

```
328
329     font._default.single_byte = &tahoma26;
330     font._default.double_byte = &yahei32;
331
332     s.x = 540;
333     s.y = 200;
334     s.inverse = NULL;
335     s.color = COLOR_RED;
336     s.background_color = COLOR_BLACK;
337     s.space.line = 5;
338     s.space.word = 0;
339     font.printf(&s,"通过 gui.line \n 函数递归调用，\n 基于分形原理，\n 画出一个分形树。");
340
341     usleep(5000000);
342
343     return 0;
344 }
```

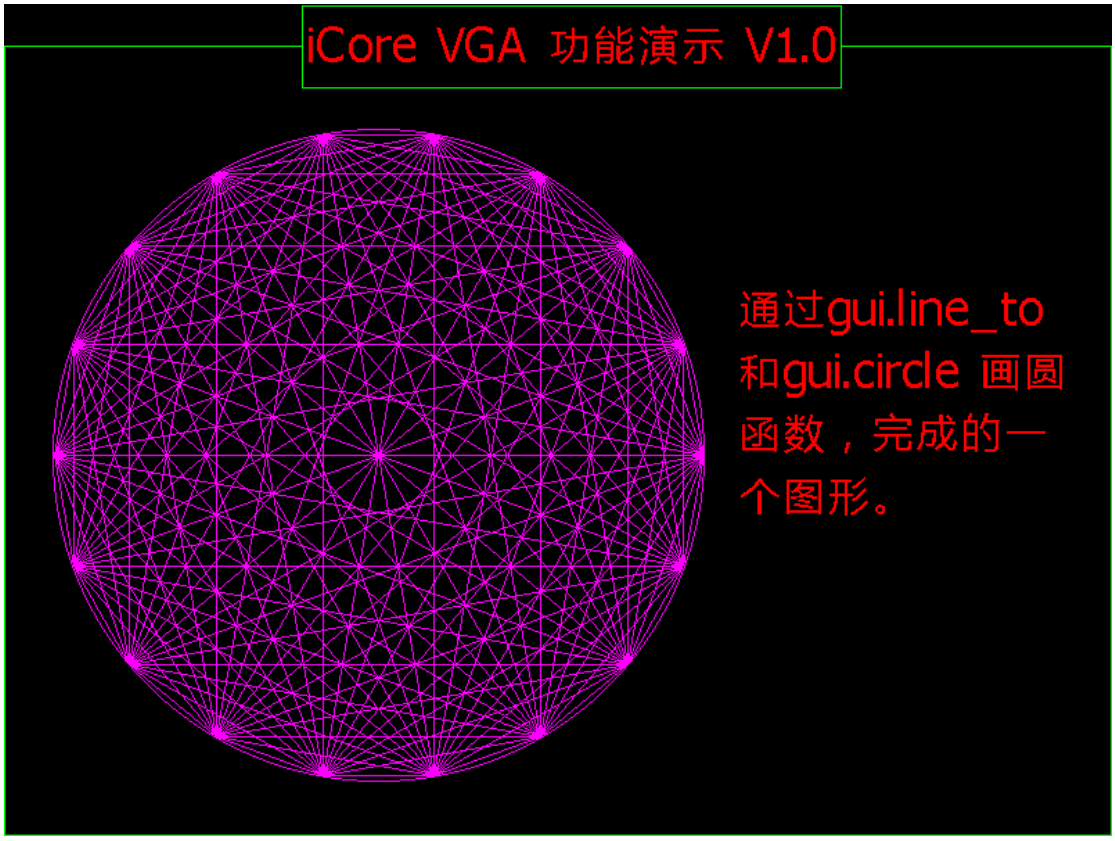
以上代码表示通过 vga 界面实现 gui 的连线功能，程序运行结果如下图所示：



```
360 #define PI 3.1415926
361
362 struct POINT_T {
363     int x, y;
364 };
365
366 static int
367 line_to_demo(int pt)
368 {
369     struct POINT_T points[20];
370     int i, j, h, w, xcenter, ycenter;
371     int radius, angle, step;
372     double rads;
373     STRING_T s;
374
375     draw_frame();
376
377     h = 500;
378     w = 700;
379
380     xcenter = w / 2 - 80;    /* Determine the center of circle */
381     ycenter = h / 2 + 75;
382     radius = (h - 30) / (2);
383     step = 360 / pt;         /* Determine # of increments   */
384
385     angle = 0;               /* Begin at zero degrees   */
386     for (i = 0; i < pt; ++i) { /* Determine circle intercepts */
387         rads = (double)angle * PI /
180.0;                           /* Convert angle to radians    */
388         points[i].x = xcenter + (int)(cos(rads) * radius);
389         points[i].y = ycenter - (int)(sin(rads) * radius);
390         angle += step;       /* Move to next increment */
391     }
392
393     gui.circle(xcenter, ycenter, radius, 0,
COLOR_FUCHSINE);                 /* Draw bounding circle    */
394
395     for (i = 0; i < pt; ++i) { /* Draw the cords to the circle */
396         for (j = i;j<pt;++j){/* For each remaining intersect */
397             gui.set_cursor(points[i].x,
points[i].y);                    /* Move to beginning of cord   */
398             gui.line_to(points[j].x, points[j].y,
COLOR_FUCHSINE);                 /* Draw the cord      */
399         }
```

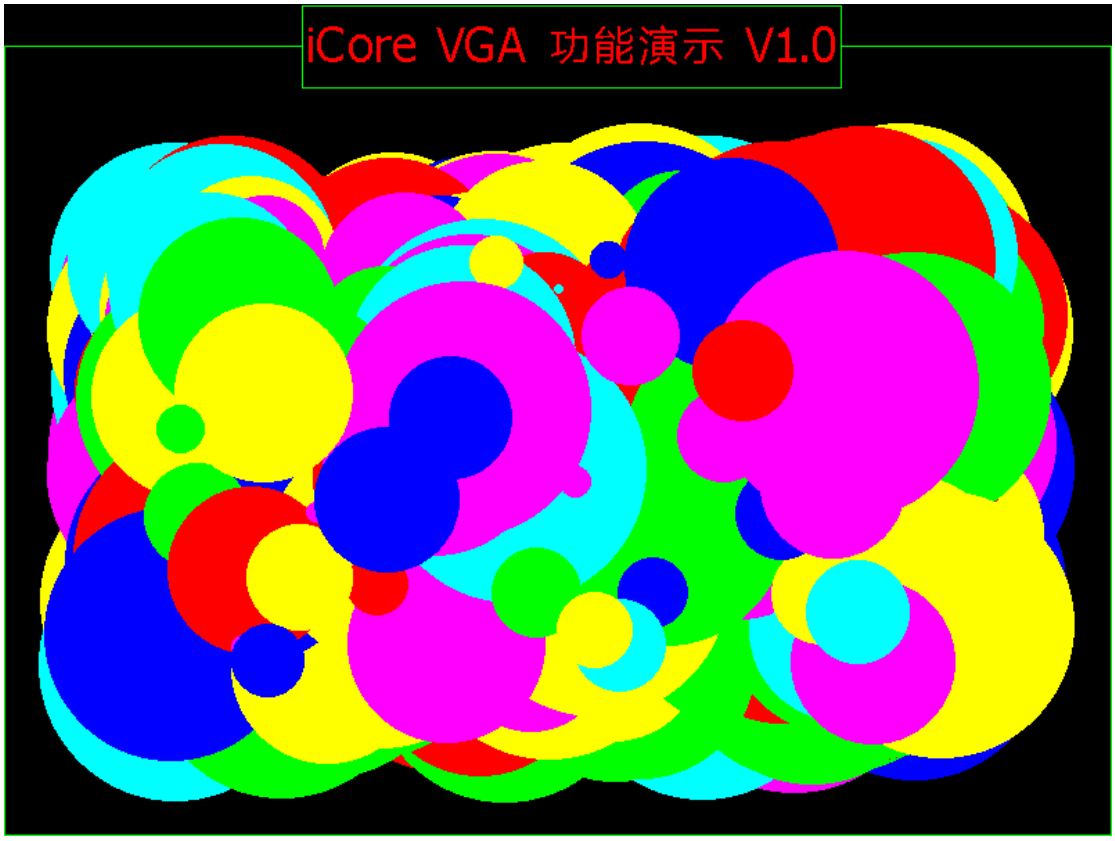
```
400     }
401     font._default.single_byte = &tahoma26;
402     font._default.double_byte = &yahei32;
403
404     s.x = 530;
405     s.y = 200;
406     s.inverse = NULL;
407     s.color = COLOR_RED;
408     s.background_color = COLOR_BLACK;
409     s.space.line = 5;
410     s.space.word = 0;
411     font.printf(&s, "通过 gui.line_to \n 和 gui.circle 画圆\n 函
数，完成的一\n 个图形。");
412
413
414     usleep(5000000);
415
416     return 0;
417 }
```

以上代码表示通过 vga 界面实现 gui 的连线功能，程序运行结果如下图所示：




```
431 static int
432 circle_demo(void)
433 {
434     int i;
435     STRING_T s;
436     int x, y, r, color;
437 
438     draw_frame();
439     for (i = 0; i < 2000; i++) {
440         x = random(560);
441         y = random(300);
442         r = random(100);
443         color = random(6) + 1;
444 
445         gui.circle(x + 120, y + 180, r, 1, color);
446     }
447 
448     return 0;
449 }
```

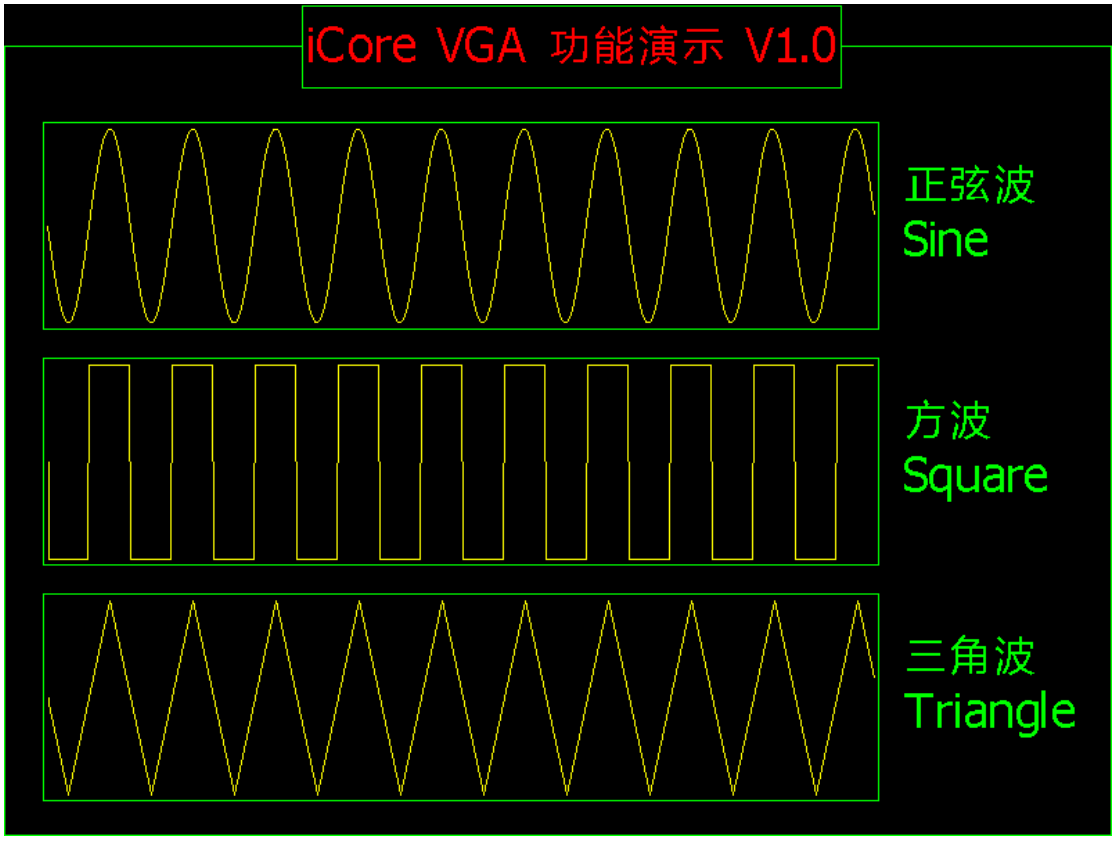
以上代码表示通过 vga 界面实现画圆功能，程序运行结果如下图所示：


iCore VGA 功能演示 V1.0

```
463 static int
464 wave_demo(void)
465 {
466     float temp;
467     int i, y;
468     RECT_T r;
469     STRING_T s;
470     char * wave_string[] = {
471         "正弦波\nSine", "方波\nSquare", "三角波\nTriangle"
472     };
473 
474     draw_frame();
475 
476     for (i = 0; i < 3; i++) {
477         r.x = 28;
478         r.y = i * 170 + 85;
479         r.width = 604;
480         r.height = 150;
481         r.fill_flag = 0;
482         r.color = COLOR_GREEN;
483         gui.rect(&r);
484     }
485 
486     //sine
487     gui.set_cursor(31, 85 + 75);
488     for (i = 1; i < 598; i++) {
489         temp = i;
490         temp /= 598;
491         temp *= 2 * PI;
492         temp *= 10;
493 
494         temp = sin(temp);
495         temp += 1;
496         temp /= 2;
497 
498         gui.line_to(i + 31, 85 + 5 + 140 * temp, COLOR_YELLOW);
499     }
500     //squa
501     gui.set_cursor(32, 85 + 75 + 170);
502 
503     for (i = 1; i < 598; i++) {
504         temp = i % 60;
505 
506         if (temp < 30) y = 1;
```

```
507         else y = 0;
508
509         gui.line_to(i + 31, 85 + 5 + 170 + 140 * y, COLOR_YELLOW);
510     }
511
512     //squa
513     gui.set_cursor(32, 85 + 75 + 170 * 2);
514
515     for (i = 1; i < 598; i++) {
516         temp = (i + 15) % 60;
517
518         temp /= 30;
519
520         if (temp > 1) temp = 2 - temp;
521
522         gui.line_to(i + 31, 85 + 5 + 170 * 2 + 140 * temp,
COLOR_YELLOW);
523     }
524
525     font._default.single_byte = &tahoma26;
526     font._default.double_byte = &yahei32;
527
528     for (i = 0; i < 3; i++) {
529         s.x = 650;
530         s.y = i * 170 + 110;
531         s.inverse = NULL;
532         s.color = COLOR_GREEN;
533         s.background_color = COLOR_BLACK;
534         s.space.line = 0;
535         s.space.word = 0;
536         font.printf(&s, "%s", wave_string[i]);
537     }
538
539     usleep(5000000);
540
541     return 0;
542 }
```

以上代码表示通过 vga 界面显示各种不同的波形，程序运行结果如下图所示：


iCore VGA 功能演示 V1.0
正弦波
Sine
方波
Square
三角波
Triangle

```
556 static int
557 font_demo(void)
558 {
559     STRING_T s;
560     SINGLE_BYTE_FONT_T * slist[] = {
561         &tahoma8,
562         &tahoma9,
563         &tahoma10,
564         &tahoma11,
565         &monaco,
566         &courier,
567         &fixedsys,
568         &borlandTE,
569         &tahoma12,
570         &tahoma26
571     };
572     char * sname[] = {
573         "Tahoma8",
574         "Tahoma9",
575         "Tahoma10",
576         "Tahoma11",
577         "monaco",
578         "courier",
```

```
579         "fixedsys",
580         "BoarlandTE",
581         "Tahoma12",
582         "Tahoma26",
583     };
584     int i;
585 
586     draw_frame();
587 
588     font._default.double_byte = &simsun16;
589 
590     s.x = 20;
591     s.y = 75;
592     s.inverse = NULL;
593     s.color = COLOR_GREEN;
594     s.background_color = COLOR_BLACK;
595     s.space.line = 5;
596     s.space.word = 0;
597 
598     for (i = 0; i < 10; i++) {
599         font._default.single_byte = slist[i];
600         s.x = 20;
601         font.printf(&s, "%s: The quick brown fox jumps over the
lazy dog.", sname[i]);
602         font.printf(&s, "\n");
603     }
604 
605 
606     font._default.double_byte = &simsun16;
607     font._default.single_byte = &fixedsys;
608     s.x = 20;
609     font.printf(&s,"\n16 点阵宋体：床前明月光，疑是地上霜，举头
望明月，低头思故乡！\n");
610 
611     font._default.double_byte = &yahei32;
612     font._default.single_byte = &tahoma26;
613     s.x = 20;
614     font.printf(&s,"32 点阵雅黑字体：床前明月光，疑是地上霜，举
头望明月，低头思故乡！");
615     font.printf(&s, "\n");
616 
617     s.color = COLOR_RED;
618     font._default.double_byte = &simsun16;
```

```
619     font._default.single_byte = &nix48;
620     s.x = 20;
621     font.printf(&s, "0123456789.");
622     s.y += 75;
623
624     font._default.double_byte = &simsun16;
625     font._default.single_byte = &nix96;
626     s.x = 20;
627     font.printf(&s, "01234567.");
628
629     usleep(5000000);
630
631     return 0;
632 }
630
631     return 0;
632 }
```

以上代码表示通过 vga 界面显示各种不同的文字字体，程序运行结果如下图所示：

第五章 结论及参考文献

一、结论

本文档主要介绍构建 vga 的 ip 核的操作步骤，实现 vga 的驱动，达到在显示器上显示各种不同的效果，详见本文档附带的压缩包。

二、参考文献

1、VGA 驱动与实现

2、Avalon 总线规范

3、基于 NIOS-II 的 VGA IP 设计

4、http://www.docin.com/p-68197623.html

E.V.Stdio

QQ 群：

[Group A]　204255896（500 人，满）

[Group B]　165201798（500 人，满）

[Group C]　215053598（200 人，满）

[Group D]　215054675（200 人高级群）

[Group E]　215055211（200 人，满）

[Group F]　78538605　（200 人高级群）

[Group G]　158560047（500 人，满）

淘宝：http://i-board.taobao.com

博客：http://XiaomaGee.cnblogs.com

论坛：http://www.heijin.org