# 数字电路与数字系统实验

**EX08:状态机及键盘输入**

191220029 傅小龙

周一5-6节班

[1830970417@qq.com](mailto:1830970417@qq.com)

2020年11月27日

# **目录**

1. [实验内容](#_实验内容)………………………………………………………………………………………3
   1. [实验要求](#_1.1实验要求)…………………………………………………………………………………3
   2. [实验工具](#_1.2实验工具)…………………………………………………………………………………3
2. [实验过程](#_实验过程)………………………………………………………………………………………3

2.1 显示不同颜色条纹………………………………………………………………………3

2.1.1 [模型概述](#_2.2.1模型概述)…………………………………………………………………………3

2.1.2 [数字抽象](#_2.2.2数字抽象)…………………………………………………………………………4

2.1.3 [建立模型](#_2.2.3建立模型)…………………………………………………………………………4

2.1.4 [分析/综合](#_2.4分析/综合)……………………………………………………………………… 5

2.1.5 [分配引脚](#_2.6分配引脚)…………………………………………………………………………6

2.1.6 [全编译](#_2.7全编译)……………………………………………………………………………8

2.2 图片显示…………………………………………………………………………………8

2.1.1 [模型概述](#_2.2.1模型概述)…………………………………………………………………………8

2.1.2 [数字抽象](#_2.2.2数字抽象)…………………………………………………………………………9

2.1.3 [建立模型](#_2.2.3建立模型)…………………………………………………………………………9

2.1.4 [分析/综合](#_2.4分析/综合)…………………………………………………………………… 11

2.1.5 [分配引脚](#_2.6分配引脚)……………………………………………………………………… 12

2.1.6 [全编译](#_2.7全编译)………………………………………………………………………… 14

2.3 自定义移动图片……………………………………………………………………… 15

2.1.1 [模型概述](#_2.2.1模型概述)……………………………………………………………………… 15

2.1.2 [建立模型](#_2.2.3建立模型)……………………………………………………………………… 15

2.1.3 [分析/综合](#_2.4分析/综合)…………………………………………………………………… 16

2.1.4 [分配引脚](#_2.6分配引脚)……………………………………………………………………… 17

2.1.5 [全编译](#_2.7全编译)………………………………………………………………………… 19

三、[实验总结](#_实验总结)…………………………………………………………………………………… 19

#### 实验内容

##### 1.1实验要求

（一）显示不同颜色条纹

在VGA控制器中，根据扫描的行或列数据，输出两种以上的不同颜色条纹（横条或竖条均可以）。

（二）图片显示

使用控制器，在显示器上显示一张静态图片。

（三）拓展要求：

·显示一张自定义的图片，自行完成图片格式到 mif 文件的转换。如有余力，可以显示一张在屏幕上按特定速度移动的图片。即图片本身大小远小于显示器分辨率，例如 100 × 100 像素大小。图片随时钟按特定方向以随机速度（x方向和 y 方向速度可不同）在屏幕内移动，当图片边界触及屏幕边界时按弹性碰撞方式改变运动方向。最终效果类似弹球游戏，图片在屏幕内不停反弹。

##### 1.2实验工具

软件环境：

设计、编译、仿真：Quartus Prime Version 17.1.0 Build 590 10/25/2017 SJ Lite Edition

DE10\_Standard\_SystemBuilder

硬件环境： DE-10 Standard开发平台

FPGA芯片： Cyclone V 5CSXFC6D6F31C6

#### 实验过程

##### 2.1显示不同颜色条纹

###### 2.1.1模型概述

在exp09.pdf表 9-2给出的VGA参考代码基础上，在顶层文件中根据该模块中的当前扫描像素点的坐标(h\_addr,v\_addr)给该控制模块输入需要的VGA颜色数据vga\_data

.

###### 2.1.2数字抽象

由于提供给VGA控制模块的VGA数据输入信号在顶层文件中进行赋值和修改，以下给出在顶层文件中相关操作所用到的wire型，reg型变量的定义.

|  |  |  |
| --- | --- | --- |
| 类型 | 变量名称 | 含义 |
| reg | [23:0]data | VGA数据，用于对VGA数据值修改 |
| wire | [23:0]vga\_data | VGA数据，用于传递VGA数据至VGA控制模块 |
| wire | [9:0] h\_addr | 接收VGA控制模块给出当前扫描位置的水平坐标 |
| wire | [9:0]v\_addr | 接收VGA控制模块给出当前扫描位置的垂直坐标 |

表 2-1-1:对顶层文件中变量的解释

###### 2.1.3建立模型

这里尝试要显示的条纹图片方向为横向, 故要根据h\_addr的值来对VGA数据data进行赋值.屏幕640\*480像素的区域内显示3个竖向条纹，各条纹的长边的起始位置分别为20,220,420,宽度取200像素，颜色分别取蓝白红，对应rgb码字为0x0000ff, 0xf8f8ff, 0xdc143c, 剩下的屏幕左右各20像素宽的长方形区域填充黑色色块，对应rgb码字为0x000000.

下面是相关实现的代码：

//=======================================================

// REG/WIRE declarations

//=======================================================

wire [23:0]vga\_data;

reg [23:0]data = 24'hffffff;

wire [9:0] h\_addr;

wire [9:0] v\_addr;

//=======================================================

// Structural coding

//=======================================================

assign VGA\_R = vga\_data[23:16];

assign VGA\_G = vga\_data[15:8];

assign VGA\_B = vga\_data[7:0];

assign VGA\_SYNC\_N = 1'b0;

clkgen #(25000000) my\_vgaclk(CLOCK\_50,SW[0],1'b1,VGA\_CLK);

//vga\_ctrl(pclk,reset,vga\_data,[9:0] h\_addr,[9:0]

v\_addr,hsync,vsync,valid,[7:0] vga\_r,[7:0] vga\_g,[7:0] vga\_b);

vga\_ctrl ctrl0(VGA\_CLK, SW[1], data, h\_addr, v\_addr, VGA\_HS, VGA\_VS, VGA\_BLANK\_N, vga\_data[23:16], vga\_data[15:8], vga\_data[7:0]);

//竖向条纹

/\*

always @(v\_addr) begin

if(v\_addr < 160)

data = 24'hff0000;

else if(v\_addr < 320)

data = 24'h90ee90;

else if(v\_addr < 480)

data = 24'h0000ff;

else;

end\*/

//横向条纹

always @(h\_addr) begin

if(h\_addr < 20)

data = 24'h000000;

else if(h\_addr < 21 + 200 && h\_addr > 19)

data = 24'h0000ff;

else if(h\_addr < 20 + 400)

data = 24'hf8f8ff;

else if(h\_addr < 20 + 600)

data = 24'hdc143c;

else data = 24'h000000;

end

endmodule

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21  vga\_ctrl ctrl0(VGA\_CLK, SW[1], data, h\_addr, v\_addr, VGA\_HS, VGA\_VS, VGA\_BLANK\_N, vga\_data[23:16], vga\_data[15:8], vga\_data[7:0]);  //竖向条纹  /\*  always @(v\_addr) begin  if(v\_addr < 160)  data = 24'hff0000;  else if(v\_addr < 320)  data = 24'h90ee90;  else if(v\_addr < 480)  data = 24'h0000ff;  else;  end\*/  //横向条纹  always @(h\_addr) begin  if(h\_addr < 20)  data = 24'h000000;  else if(h\_addr < 21 + 200 && h\_addr > 19)  data = 24'h0000ff;  else if(h\_addr < 20 + 400)  data = 24'hf8f8ff;  else if(h\_addr < 20 + 600)  data = 24'hdc143c;  else data = 24'h000000;  end  endmodule |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |
| 37 |  |
| 38 |  |
| 39 |  |
| 40 |  |
| 41 |  |
| 42 |  |
| 43 |  |
| 44 |  |
| 45 |  |
| 45 |  |

\*Line16处的clkgen模块的实现参考exp09.pdf表 9-1通用时钟生成代码

###### 2.1.4分析/综合

分析/综合实验成功，如下图所示：

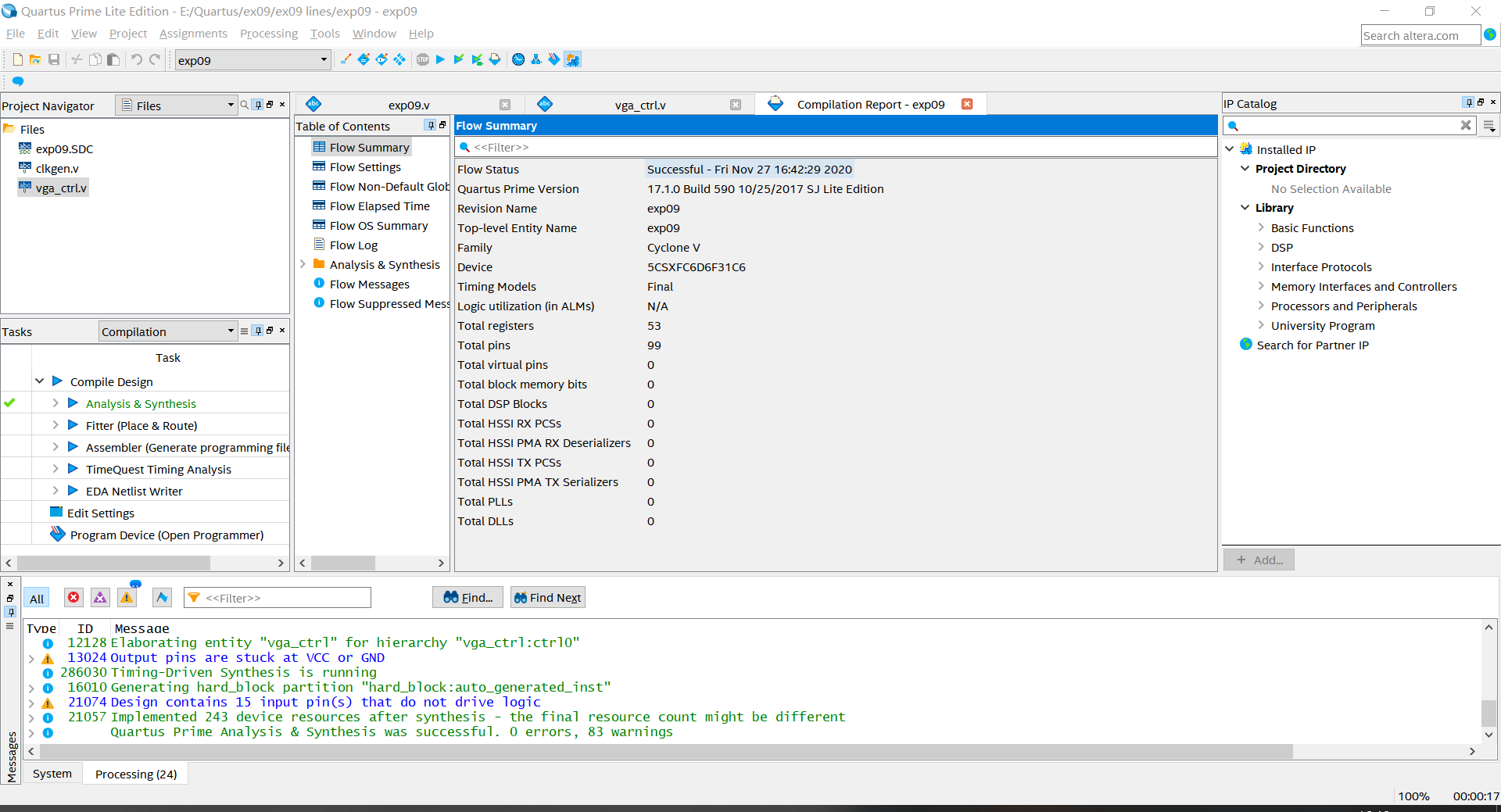


图2-1-1：分析/综合成功

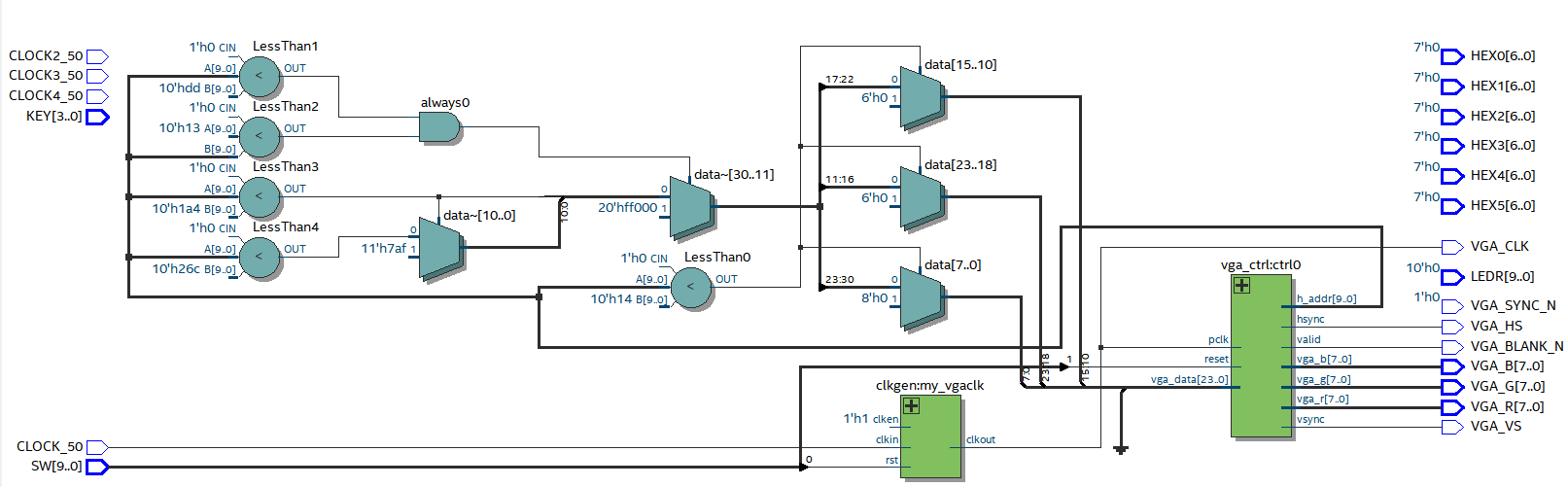
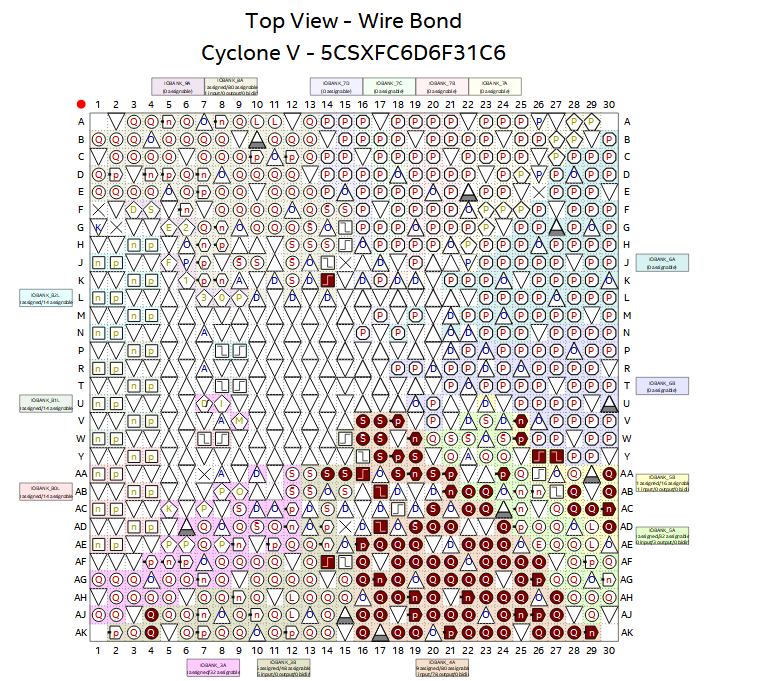
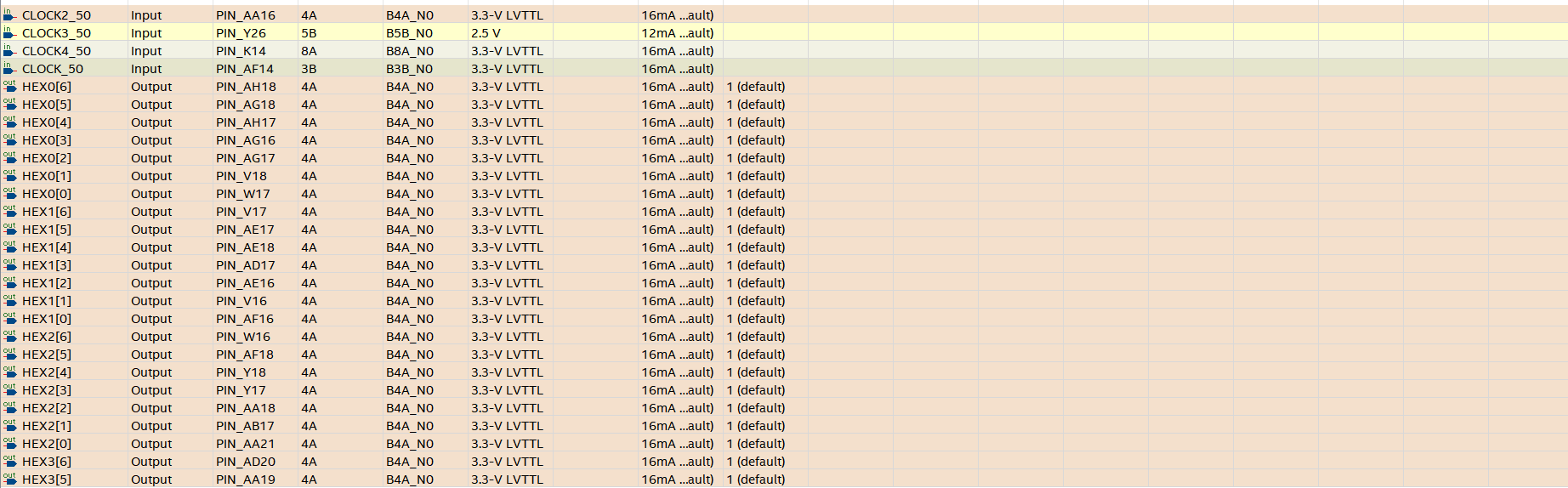


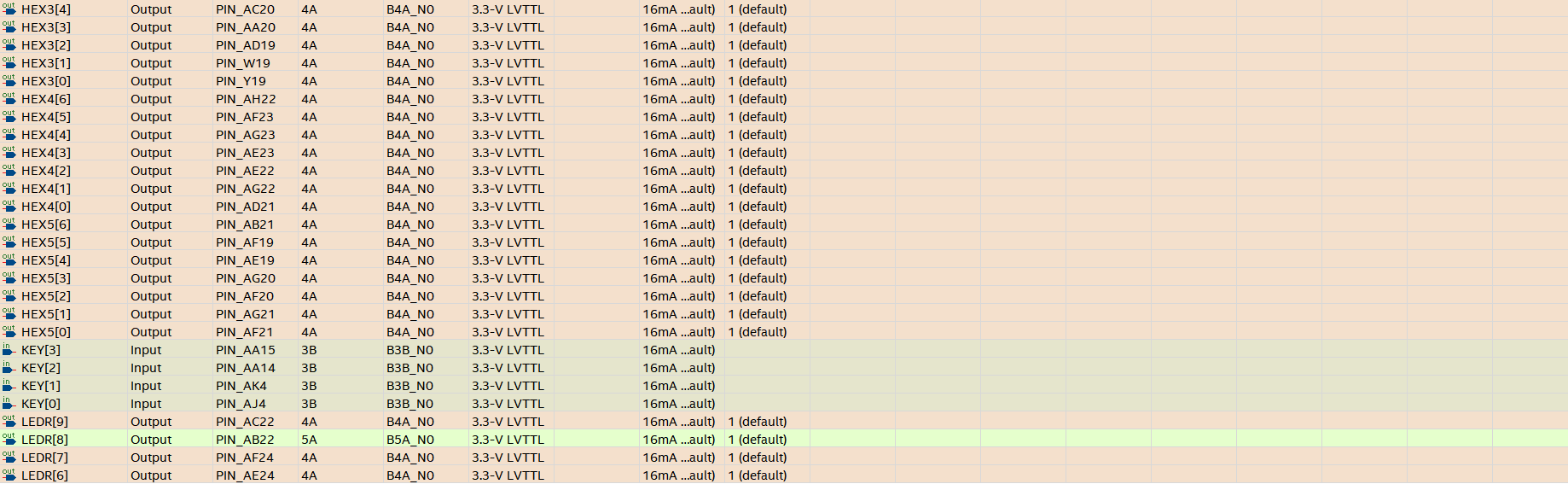
图 2-1-2:RTL视图

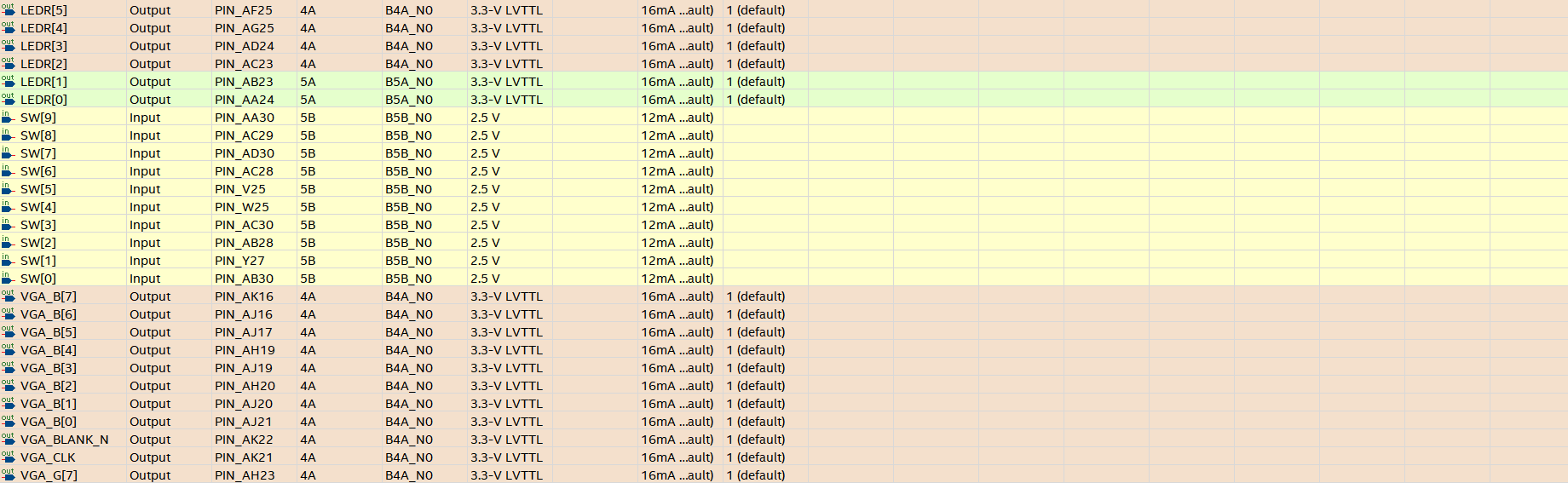
###### 2.1.5分配引脚

引脚分配使用DE10\_Standard\_SystemBuilder生成。









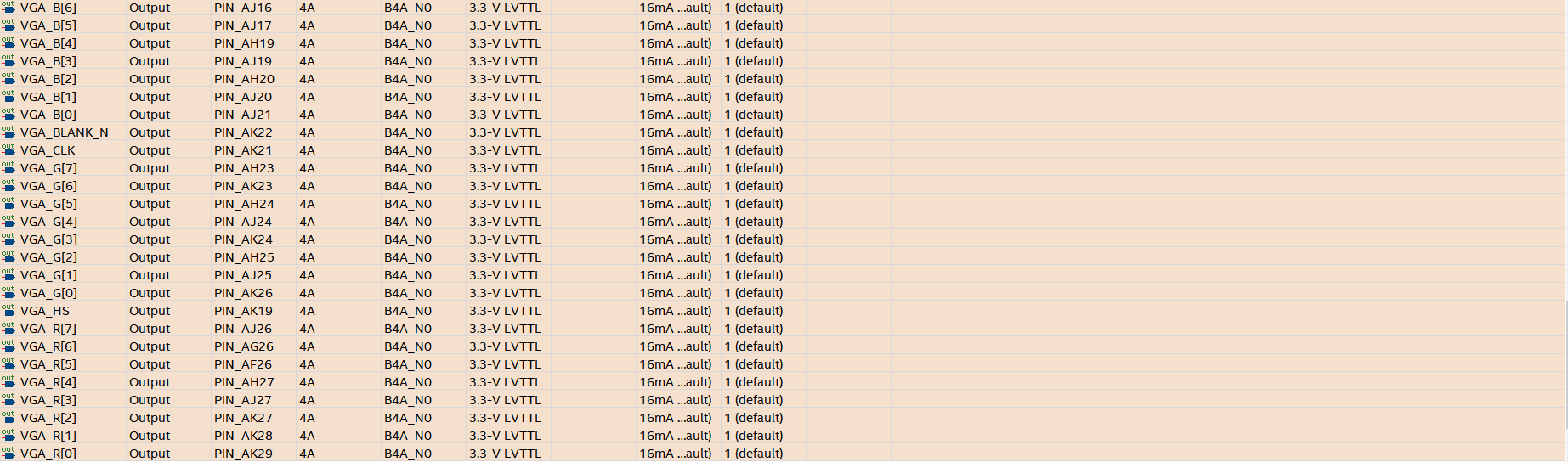


图2-1-3引脚分配图

###### 2.1.6全编译

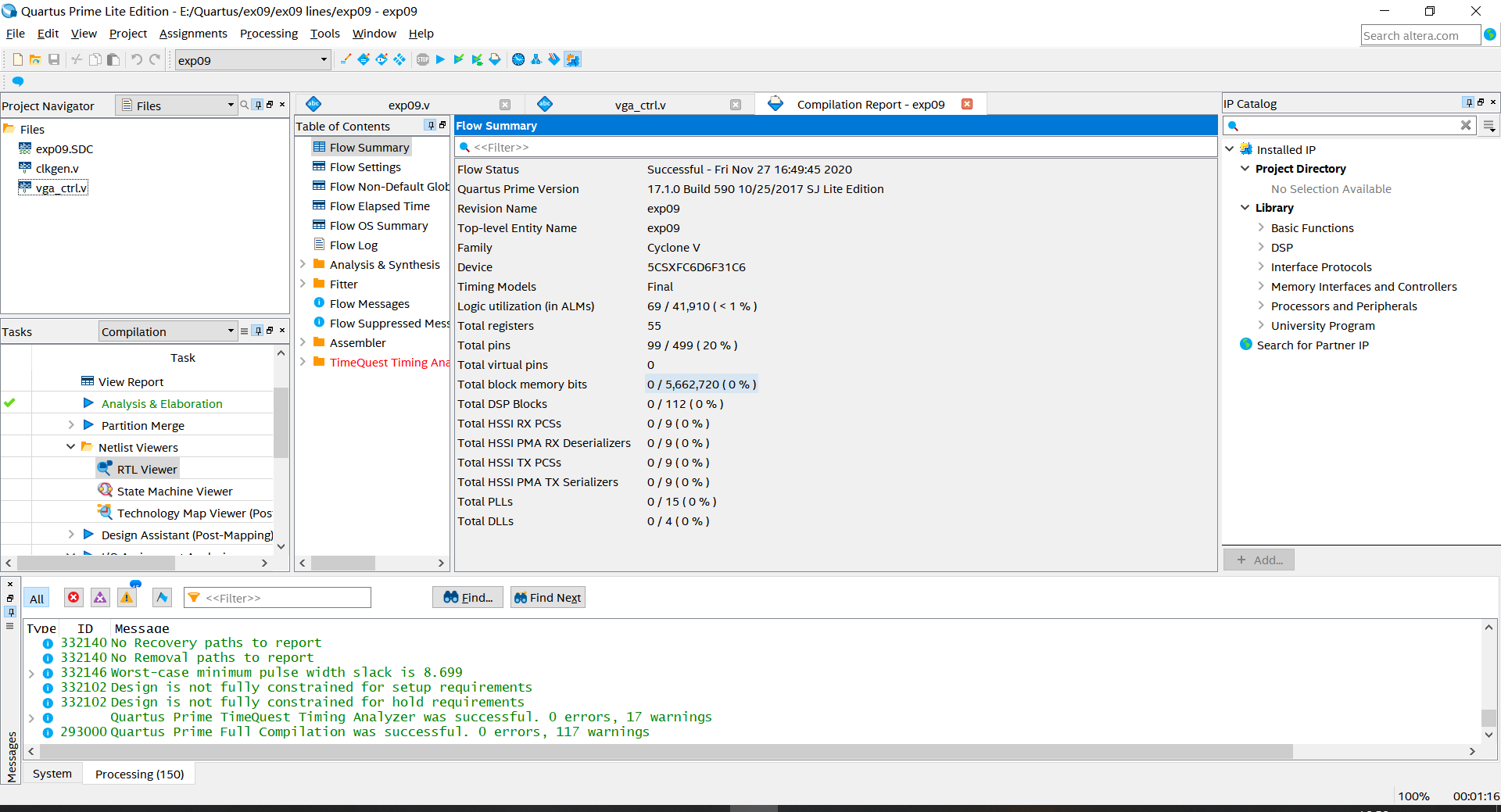


图2-1-4全编译成功

##### 2.2图片显示

###### 2.2.1模型概述

使用IP核生成的单口ROM储存要显示的图片，并根据VGA控制模块当前扫描的像素点的坐标从ROM中读取该位置的rgb数据. 考虑到开发板资源有限，采用低比特方式存储图片，显存分配大小为 640 × 512 word, 每个 word 为 12bit。用 h\_addr 的全部 10位和 v\_addr 的低 9 位合成 19 位地址来索引显存。为方便寻址，我们给行v\_addr 分配了 512 行的空间.这样，可以不用对地址进行复杂的转换。此处只需要分配 327680 个连续的存储单元，不需要考虑 h\_addr 大于 640 的情况。assign 红、绿、蓝颜色的时候，根据 12bit 显存数据中对应颜色的 4bit 值，设置输出 8bit 数据的高 4 位，低 4 位置零.

###### 2.2.2数字抽象

rgb数据的读取和对VGA控制模块的调用由picture模块实现.

|  |  |  |
| --- | --- | --- |
|  | 信号名称 | 含义 |
| 输入 | clk | VGA时钟信号(25MHz) |
| reset | 置位 |
| 输出 | VGA\_BLANK\_N | VGA消隐信号 |
| [7:0]VGA\_R | 当前像素点rgb颜色码字的输出 |
| [7:0]VGA\_B |
| [7:0]VGA\_G |
| VGA\_HS | 行同步信号 |
| VGA\_SYNC\_N | VGA同步信号 |
| VGA\_VS | 列同步信号 |

表 2-2-1:picture模块与顶层文件相连接的信号的解释

###### 2.2.3建立模型

根据2.2.1中对ROM内存储数据的分析，不难得出在任意时刻应从ROM中读取的rgb数据的地址应该满足如下关系：

addr = v\_addr + (h\_addr - 1) \* 18'd512 - 18'd1;

传递给VGA控制模块的RGB数据应分别对应从ROM中读出数据的0~3, 4~7, 8~11位.需要注意的是RGB数据的低4位需要置0.

下面是相关实现的代码：

module picture(

input clk,

input reset,

output VGA\_BLANK\_N,

output [7:0]VGA\_R,

output [7:0]VGA\_B,

output [7:0]VGA\_G,

output VGA\_HS,

output VGA\_SYNC\_N,

output VGA\_VS

);

wire [9:0]h\_addr;

wire [9:0]v\_addr;

wire hsync;

wire vsync;

wire valid;

wire [23:0]vga\_data;

assign VGA\_SYNC\_N = 1'b0;

assign VGA\_R = vga\_data[23:16];

assign VGA\_G = vga\_data[15:8];

assign VGA\_B = vga\_data[7:0];

assign VGA\_HS = hsync;

assign VGA\_VS = vsync;

assign VGA\_BLANK\_N = valid;

reg [23:0]data = 24'hffffff;

reg [18:0]addr;

vga\_ctrl ctrl1(clk,reset,data,h\_addr,v\_addr,hsync,vsync,valid,vga\_data[23:16],vga\_data[15:8],vga\_data[7:0]);

wire[11:0]curdata;

rimg r1(

.address(addr),

.clock(clk),

.q(curdata)

);

always @(clk)

begin

addr = v\_addr + (h\_addr - 1) \* 18'd512 - 18'd1;

data[3:0] <= 0;//低4位置零

data[7:4] <= curdata[3:0];

data[11:8] <= 0;//低4位置零

data[15:12] <= curdata[7:4];

data[19:16] <= 0;//低4位置零

data[23:20] <= curdata[11:8];

end

endmodule

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16  wire vsync;  wire valid;  wire [23:0]vga\_data;  assign VGA\_SYNC\_N = 1'b0;  assign VGA\_R = vga\_data[23:16];  assign VGA\_G = vga\_data[15:8];  assign VGA\_B = vga\_data[7:0];  assign VGA\_HS = hsync;  assign VGA\_VS = vsync;  assign VGA\_BLANK\_N = valid;  reg [23:0]data = 24'hffffff;  reg [18:0]addr;  vga\_ctrl ctrl1(clk,reset,data,h\_addr,v\_addr,hsync,vsync,valid,vga\_data[23:16],vga\_data[15:8],vga\_data[7:0]);  wire[11:0]curdata;  rimg r1(  .address(addr),  .clock(clk),  .q(curdata)  );  always @(clk)  begin  addr = v\_addr + (h\_addr - 1) \* 18'd512 - 18'd1;  data[3:0] <= 0;//低4位置零  data[7:4] <= curdata[3:0];  data[11:8] <= 0;//低4位置零  data[15:12] <= curdata[7:4];  data[19:16] <= 0;//低4位置零  data[23:20] <= curdata[11:8];  end  endmodule |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |
| 22 |  |
| 23 |  |
| 24 |  |
| 25 |  |
| 26 |  |
| 27 |  |
| 28 |  |
| 29 |  |
| 30 |  |
| 31 |  |
| 32 |  |
| 33 |  |
| 34 |  |
| 35 |  |
| 36 |  |
| 37 |  |
| 38 |  |
| 39 |  |
| 40 |  |
| 41 |  |
| 42 |  |
| 43 |  |
| 44 |  |
| 45 |  |
| 45 |  |
| 46 |  |
| 47 |  |
| 48 |  |
| 49 |  |
| 50 |  |
| 51 |  |

下图为ROM的IP核生成:

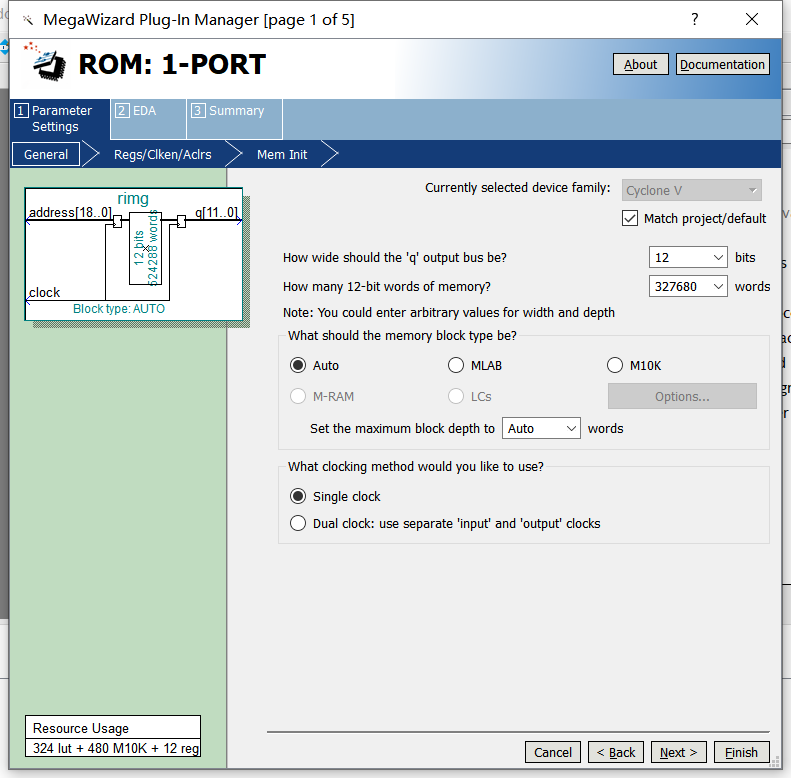


图2-2-1 IP核生成ROM

###### 2.2.4分析/综合

分析/综合实验成功，如下图所示：

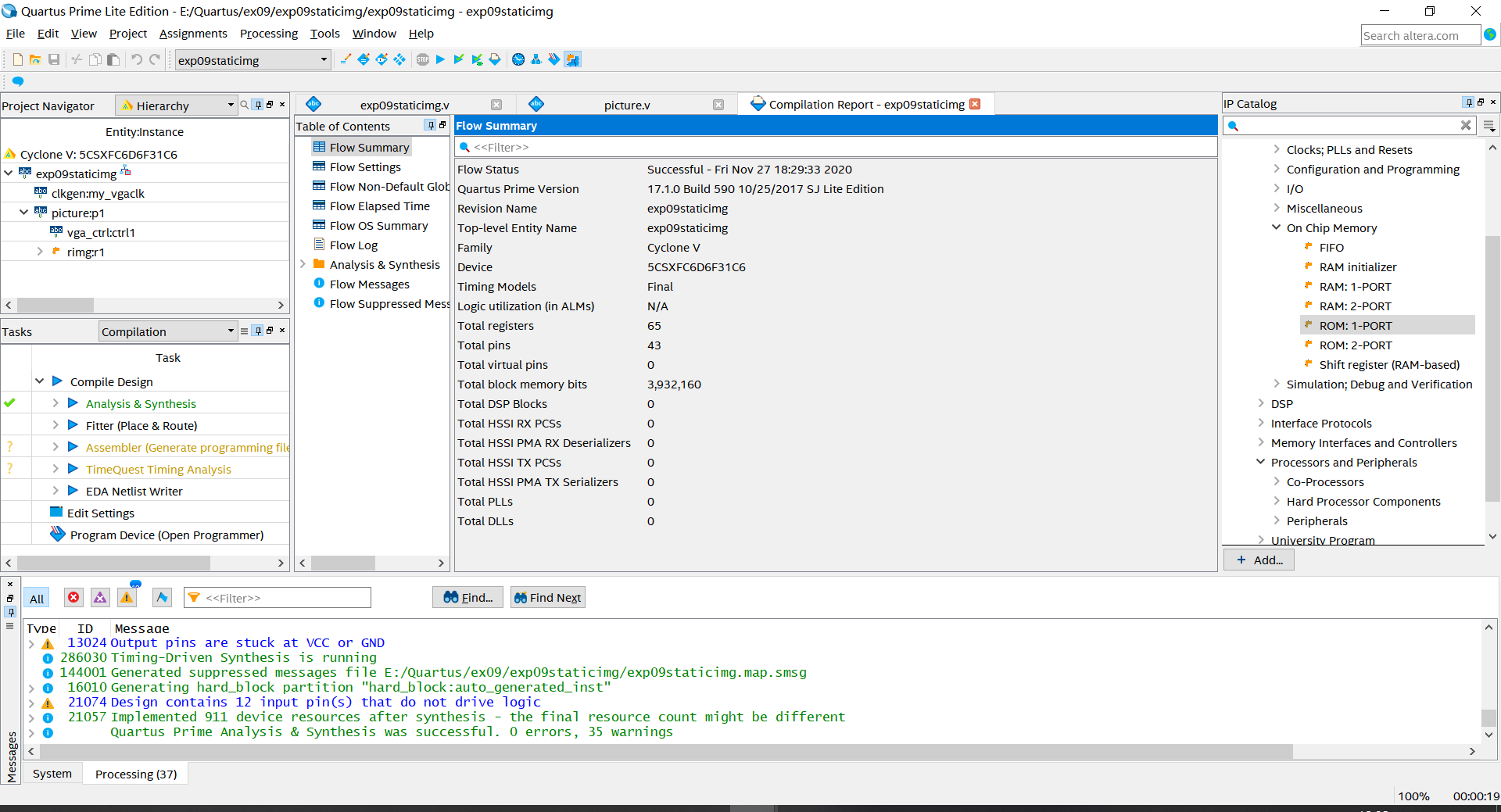
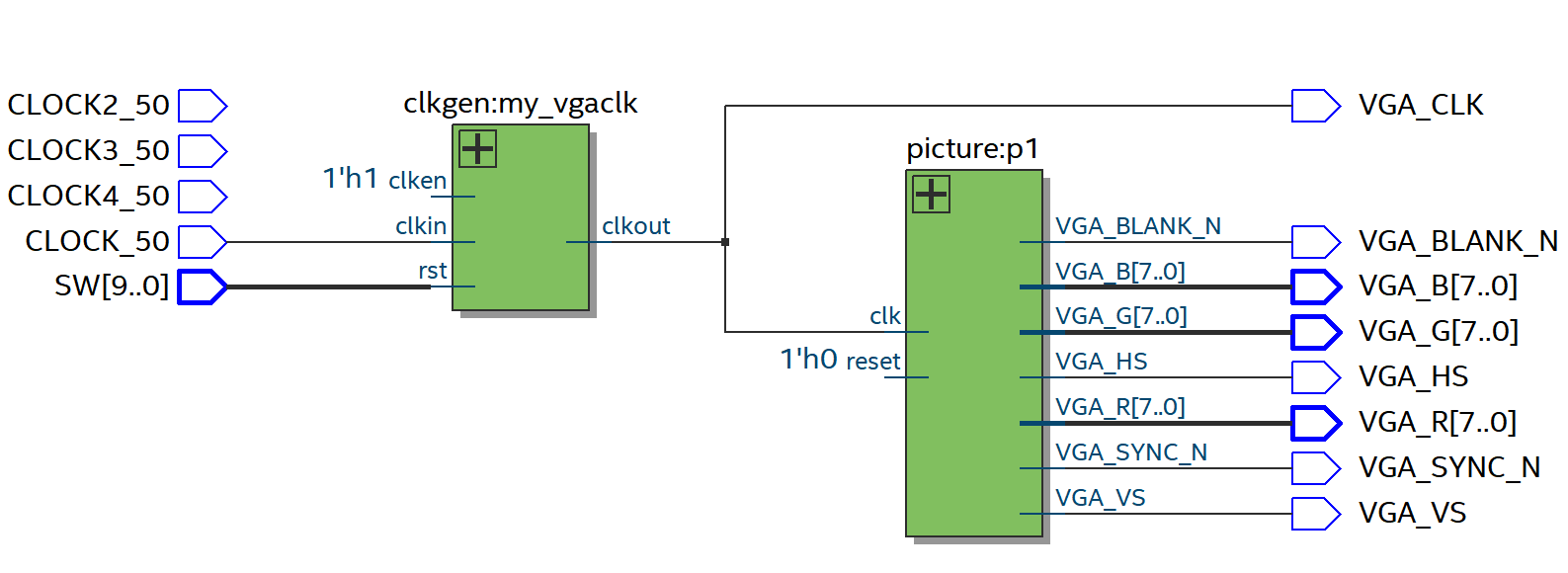


图2-2-2：分析/综合成功



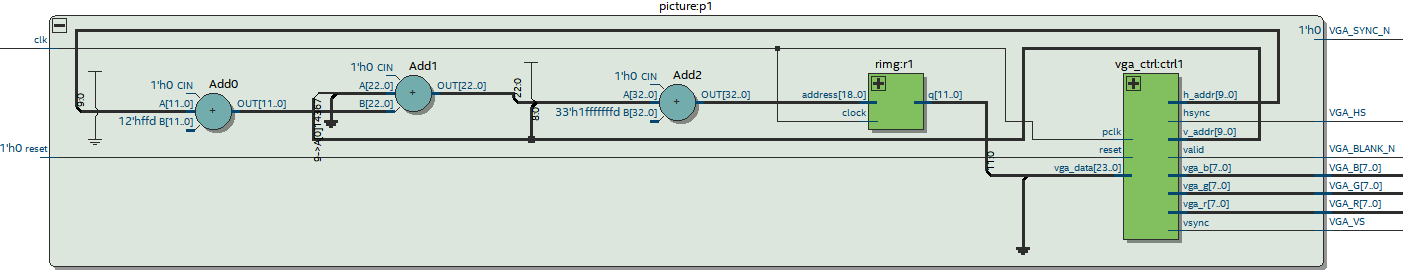
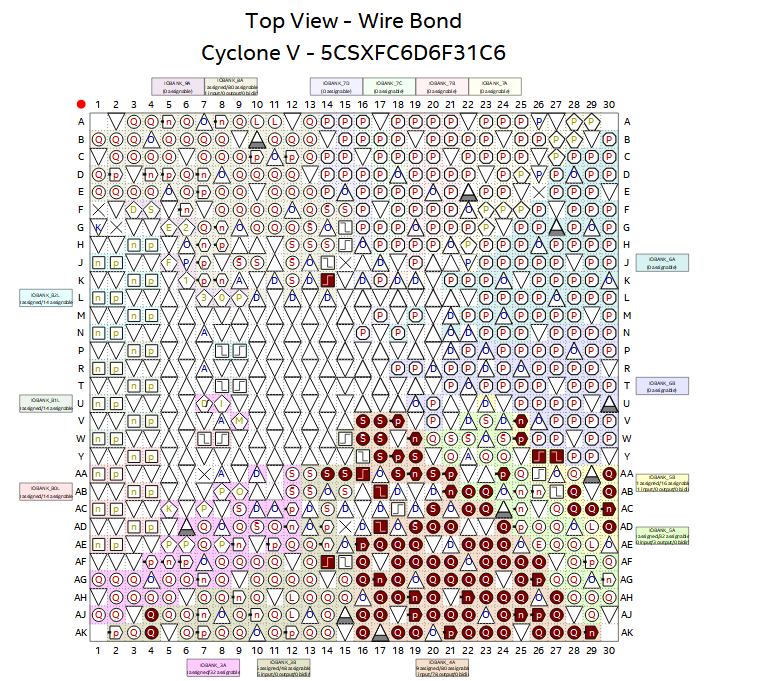
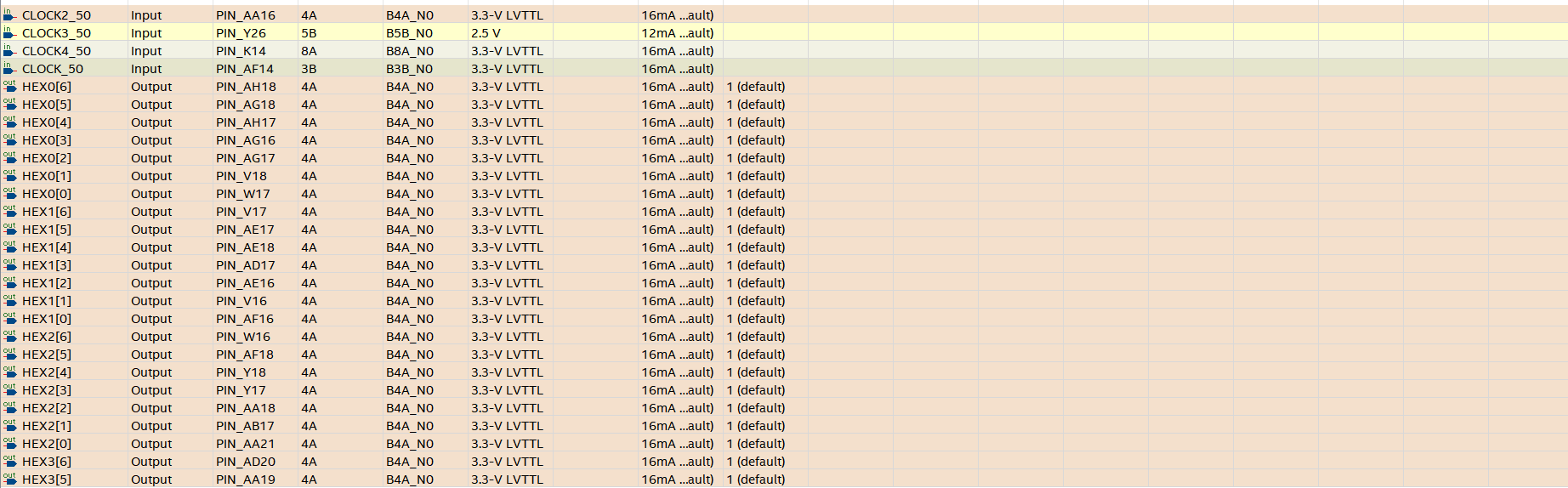


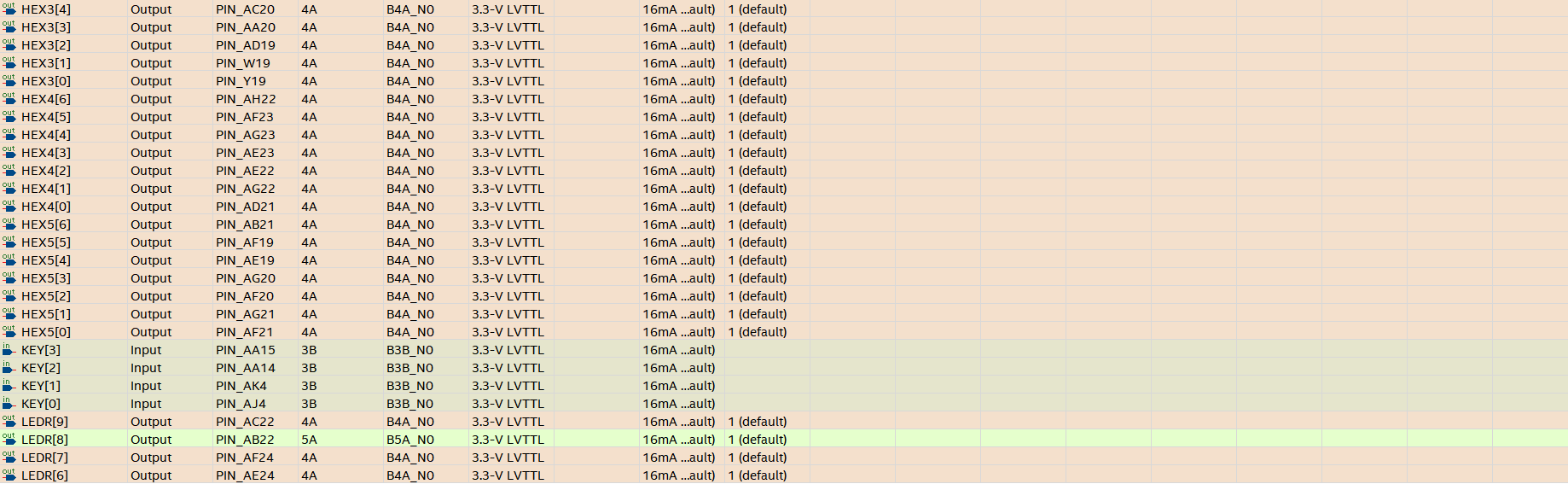
图 2-2-3:RTL视图

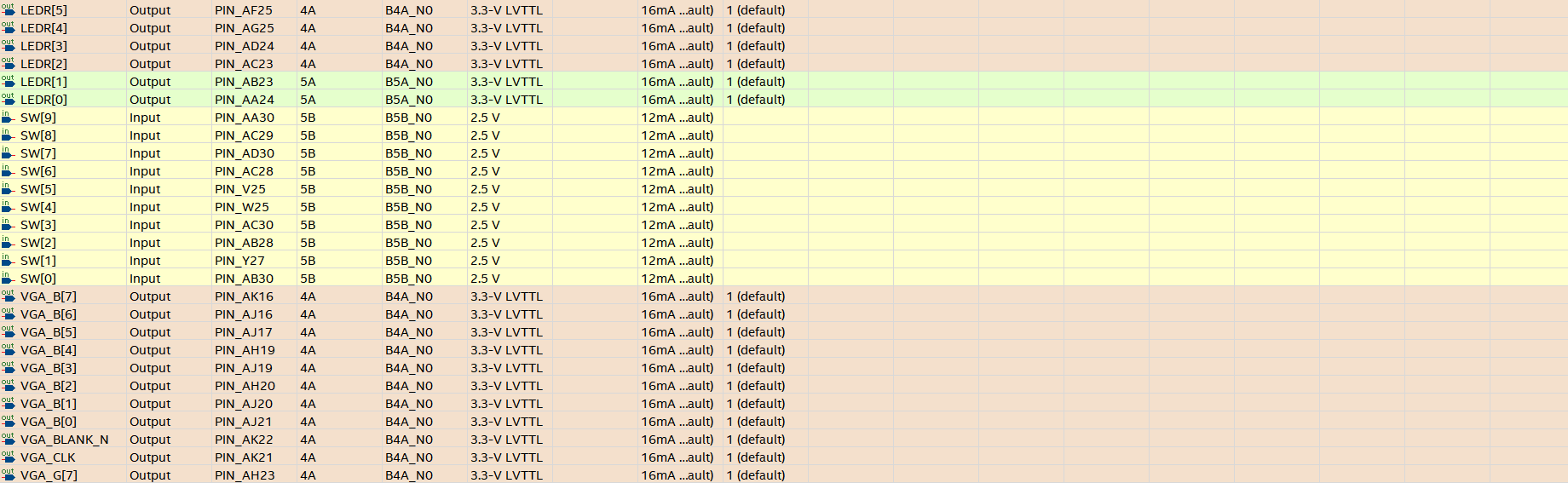
###### 2.2.5分配引脚

引脚分配使用DE10\_Standard\_SystemBuilder生成。









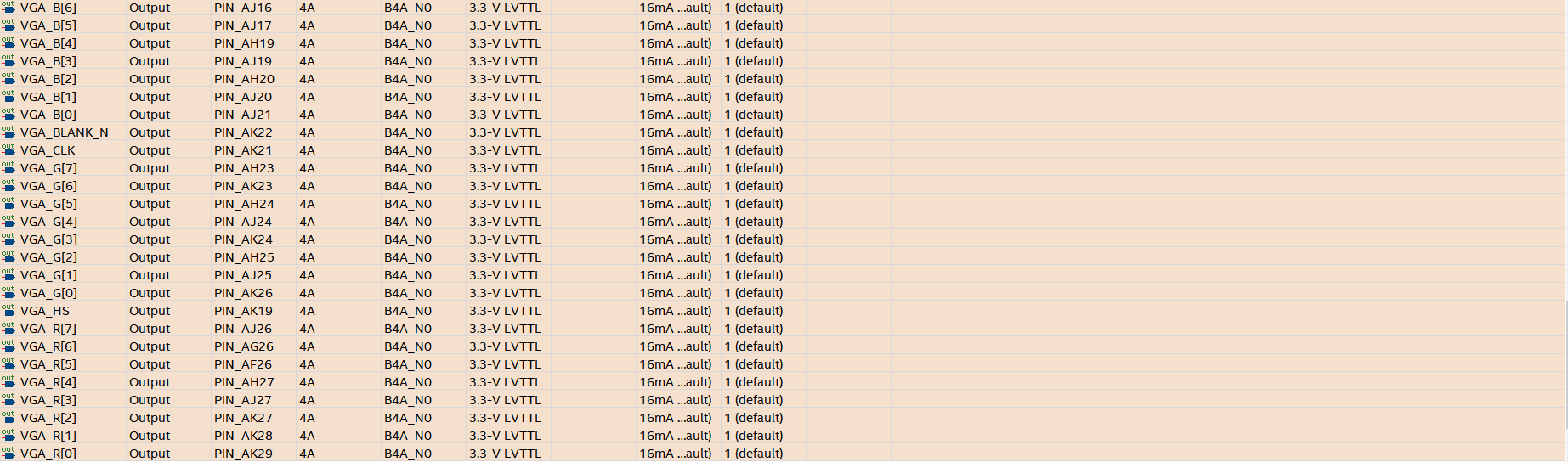


图2-2-4引脚分配图

###### 2.2.6全编译

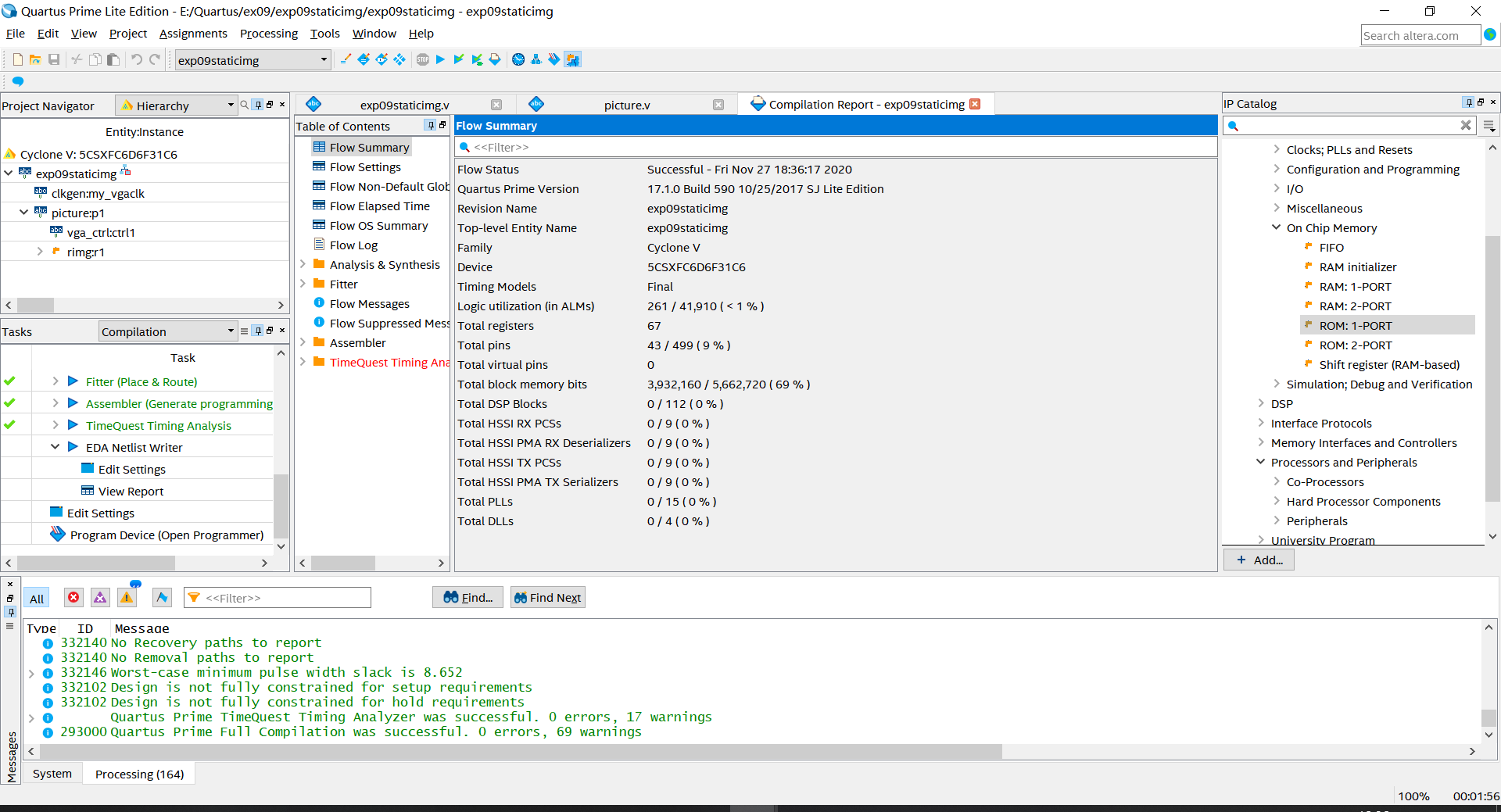


图2-2-5全编译成功

##### 2.3自定义移动图片

###### 2.3.1模型概述

参考课程网站上给出的top\_flyinglogo项目并结合100x100像素图片的使用需要对项目中的代码作出一定修改. 图片的rgb数据的存储与2.2节中的做法类似，用IP核生成的单口ROM存储.

###### 2.3.2建立模型

由于该部分的实现与参考的top\_flyinglogo项目相差无几，这里不再给出具体的实现，只指出在源代码上的修改之处：

①：

clkgen #(25000000) u0

(

.clkin(clk), // input clk\_in1

.clkout(pclk), // output clk\_out1

.rst(rst),

.clken(1'b1)

);

原来的实现中并没有给定clkgen模块参数, 需要添加参数, 如上所示.

②：

parameter [9:0] logo\_length = 10'd100;

parameter [9:0] logo\_hight = 10'd100;

原来的实现中logo的长宽并不是100x100, 需要修改成上面的形式.

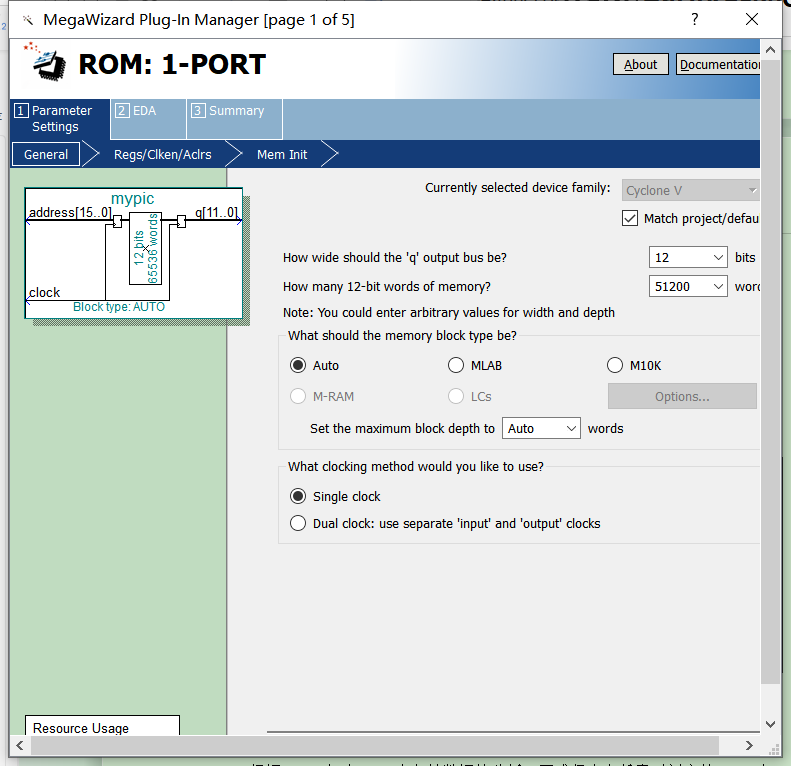


图2-3-1 IP核生成ROM

###### 2.3.3分析/综合

分析/综合实验成功，如下图所示：

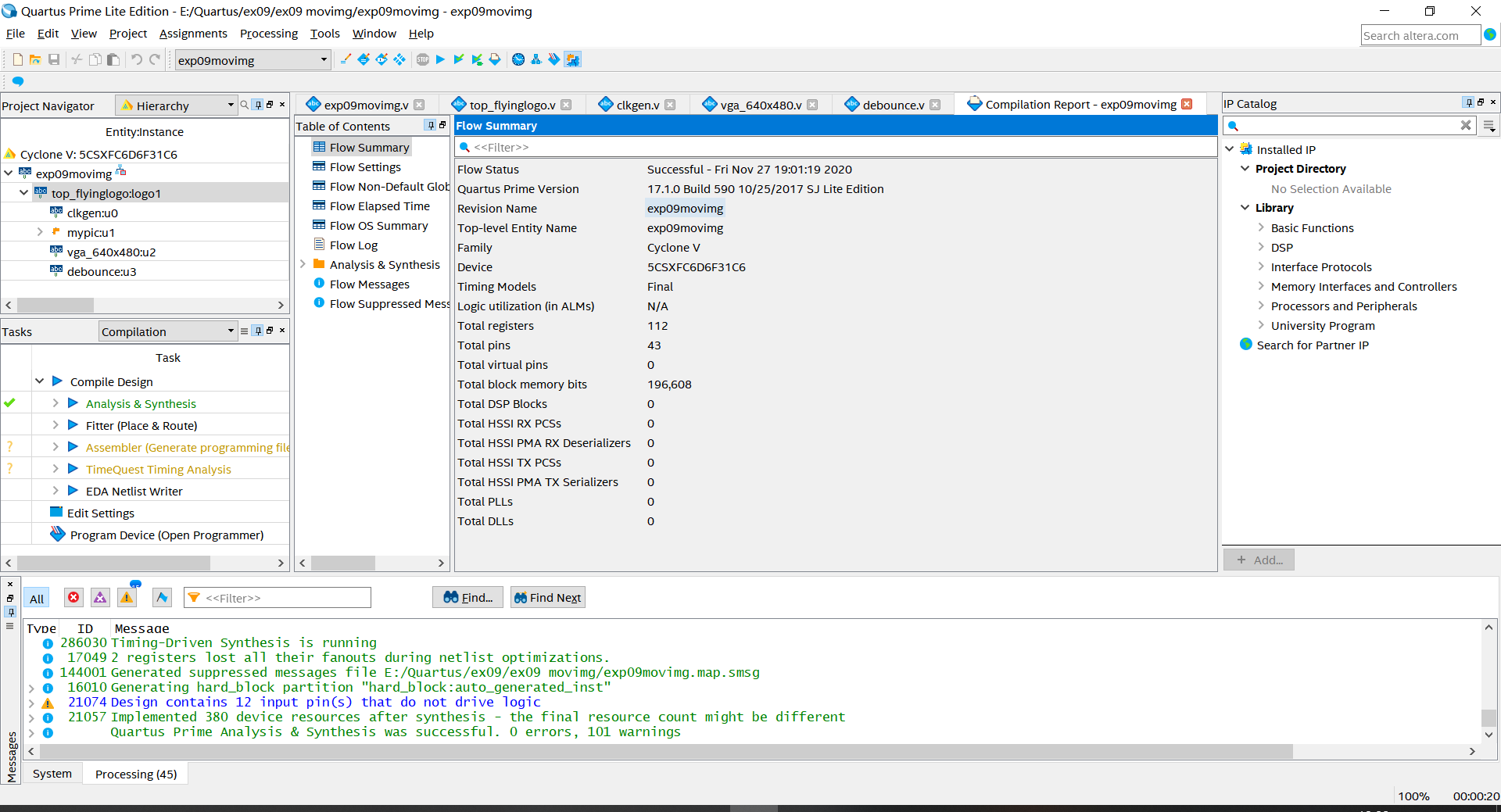


图2-3-2：分析/综合成功

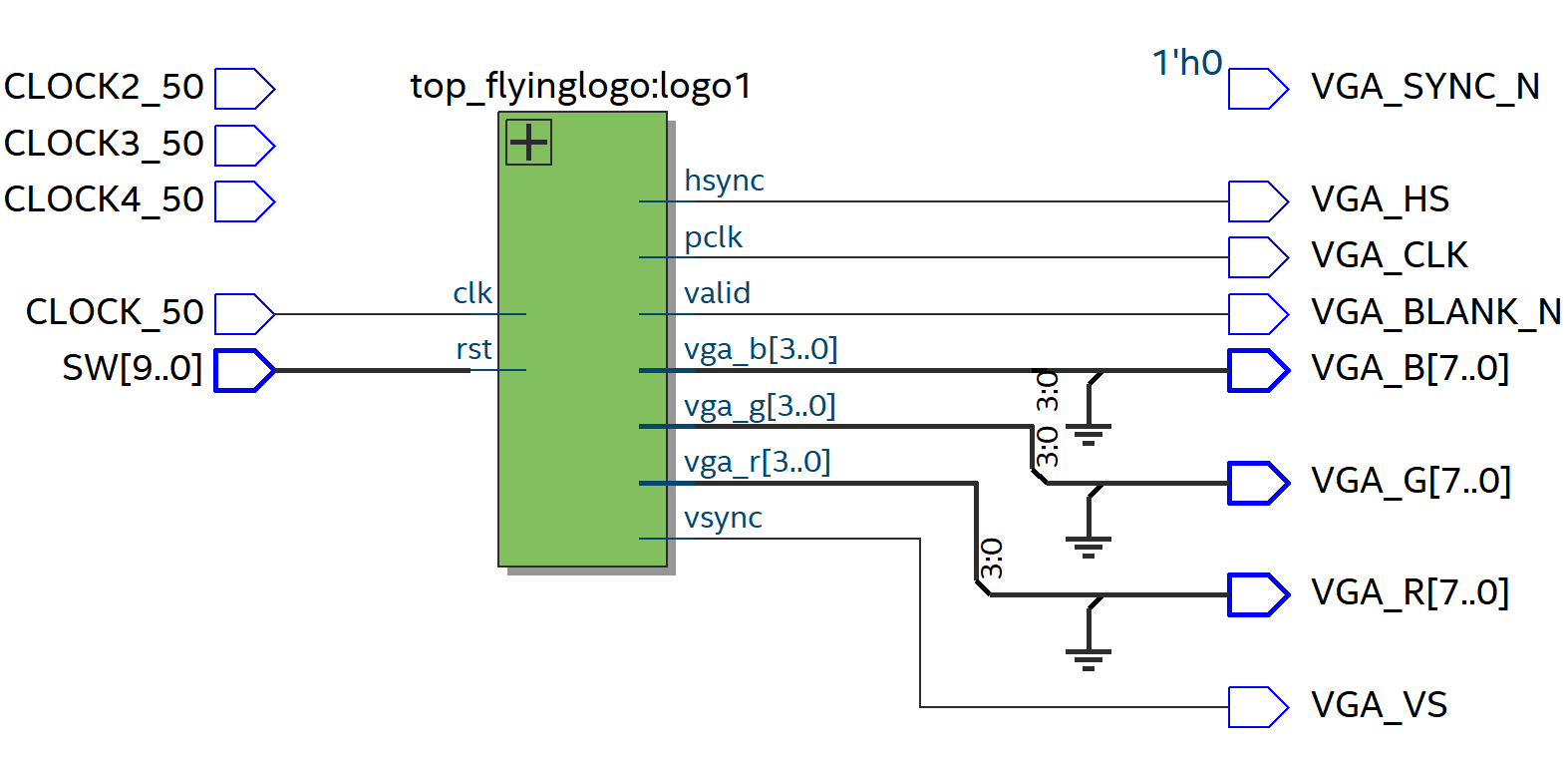
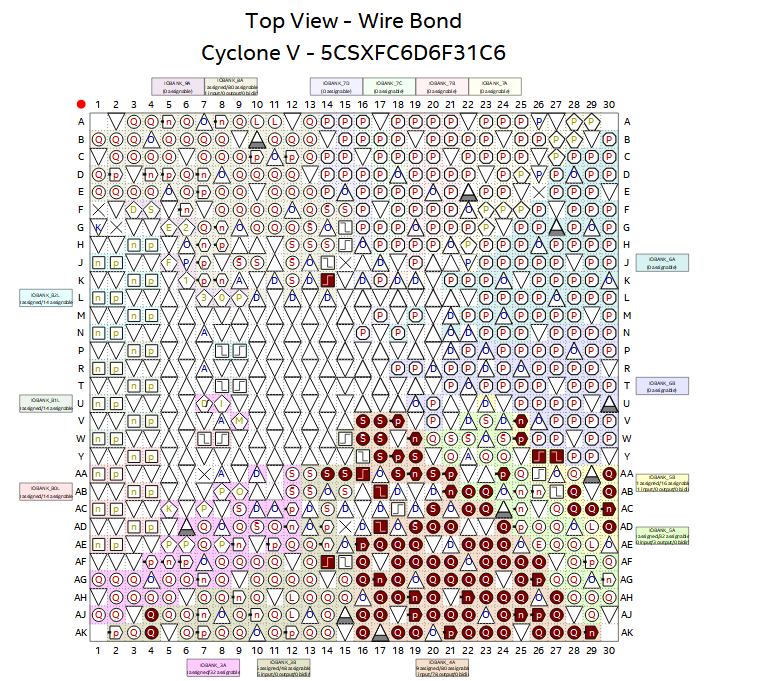
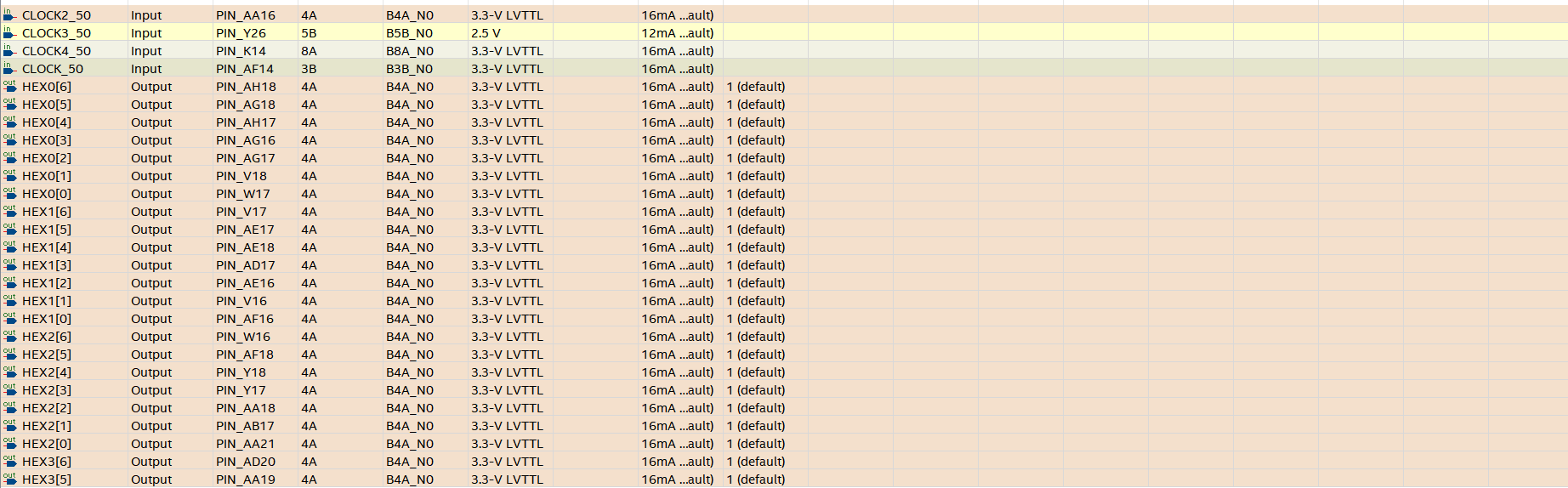


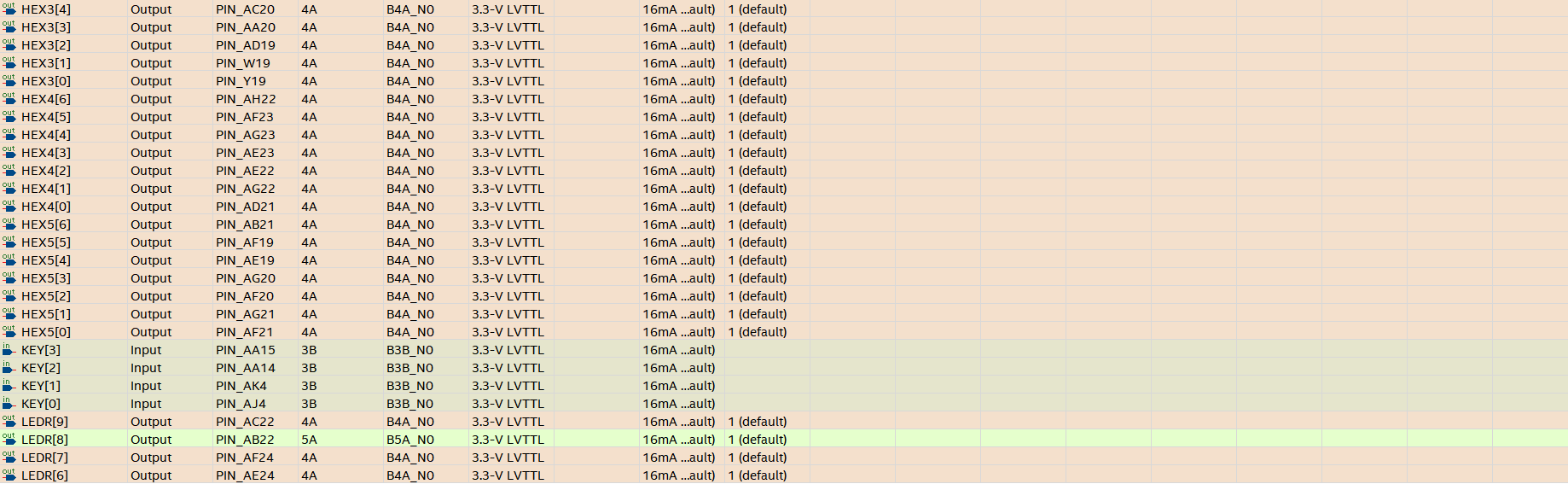
图 2-3-3:RTL视图

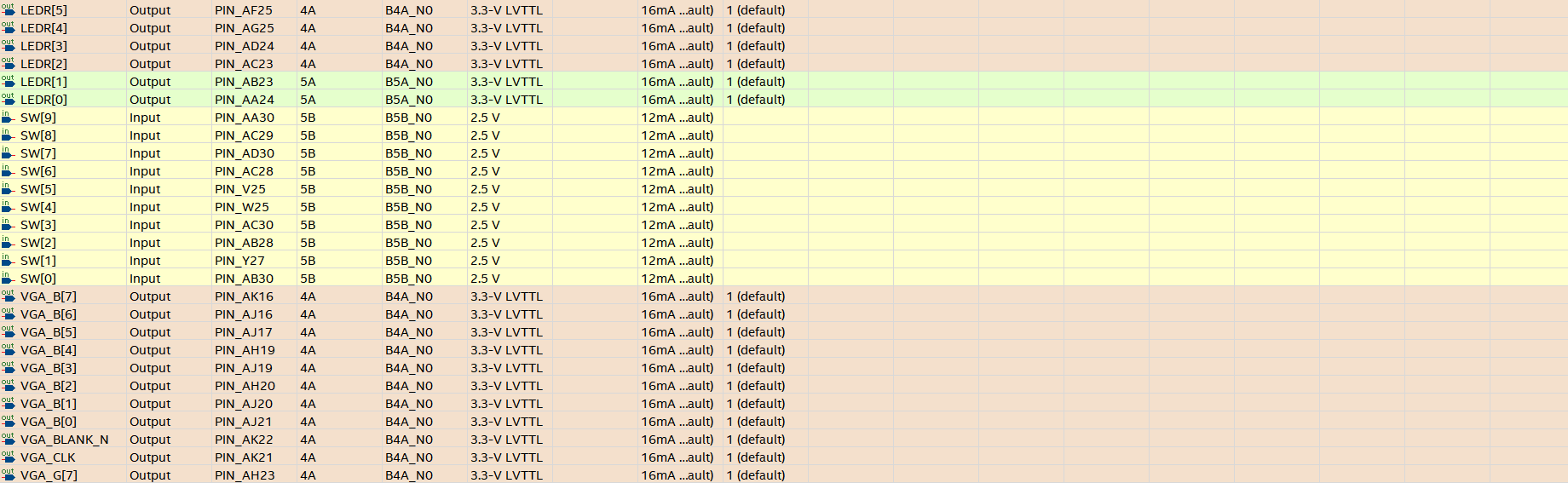
###### 2.3.4分配引脚

引脚分配使用DE10\_Standard\_SystemBuilder生成。









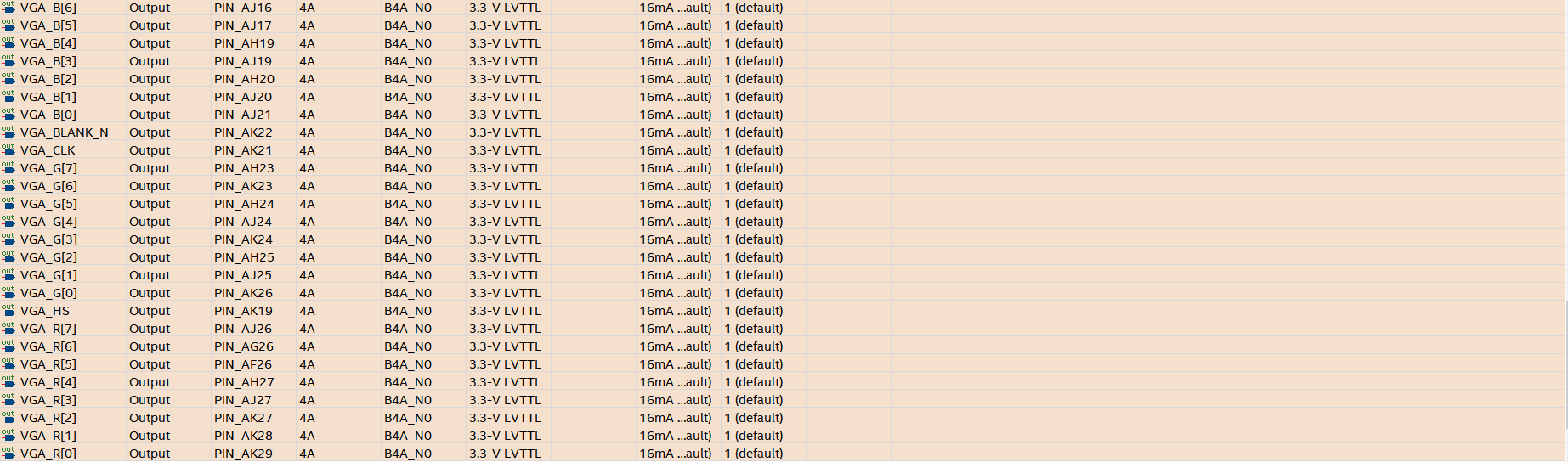


图2-3-4引脚分配图

###### 2.2.5全编译

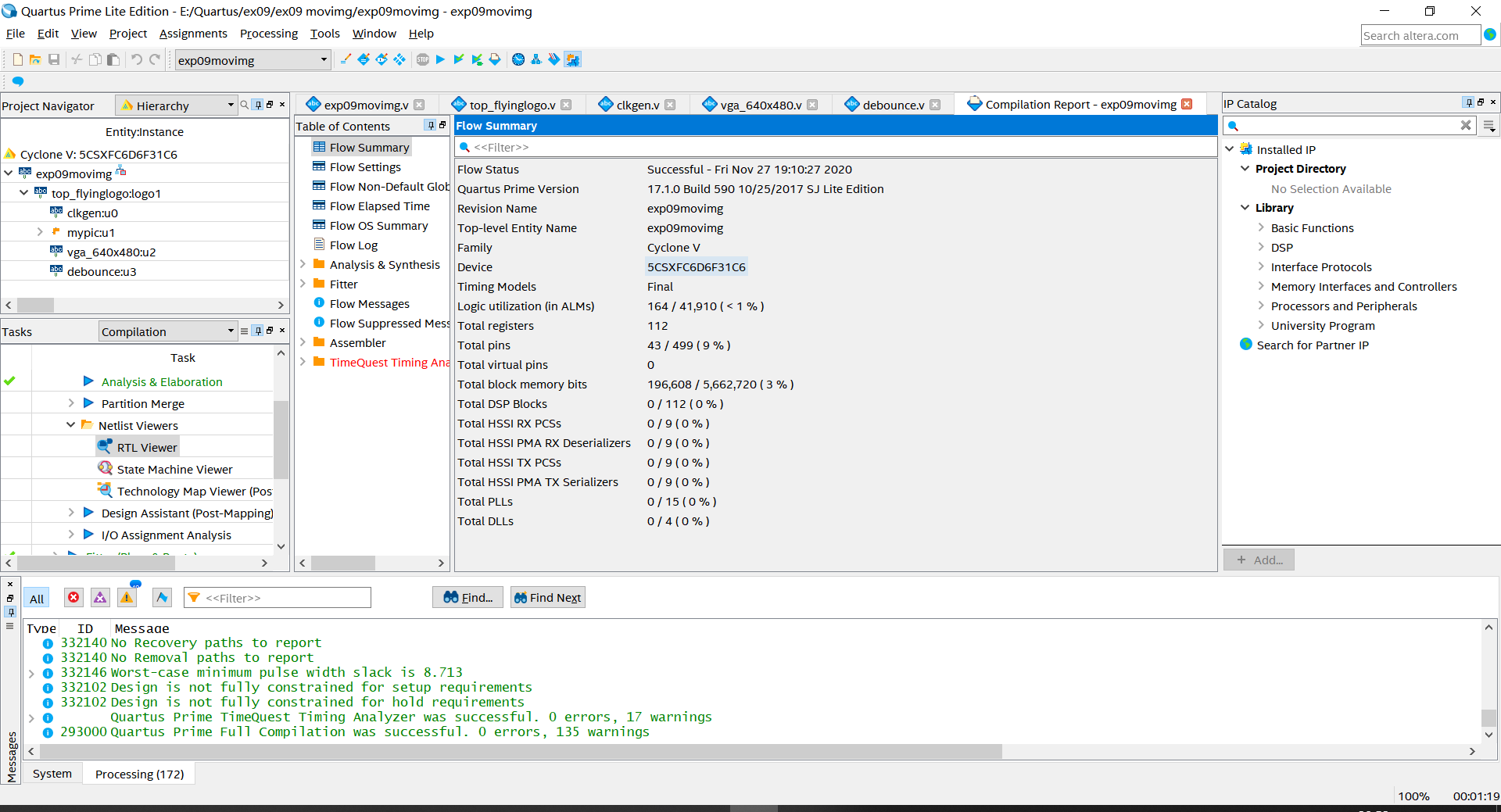


图2-3-5全编译成功

#### 实验总结

本次实验主要学习了VGA显示的原理，并在参考VGA控制模块实现的帮助下实现了简单图片的显示. 需要特别注意的是VGA在进行行扫描、场扫描时的时序关系. 另外，由于开发板的资源有限，对要显示的图片的像素颜色数量需要一定限制.