FPGA 入门课程 4-走马灯

第四节走马灯

今天的课程将引入开发板的使用，本节课使用 ZX-2 开发板，此开发板是至芯科技公司推出的低价位高性价比的初级开发板。本系列的视频教程也是围绕 ZX-2 开发板展开的。

后续大家可以通过淘宝购买此开发板，还有另外一种方式可以得到 ZX-2 开发板，去论坛申请免费评测开发板。

淘宝地址:

http://item.taobao.com/item.htm?spm=a1z10.1.w4004-6568874930.3.KDvq2a&id=38226425231

论坛免费申请地址:

http://www.fpgaw.com/thread-68086-1-1.html

开发板资料请在至芯科技论坛下载，地址如下

http://www.fpgaw.com/forum.php?mod=viewthread&tid=67978&extra=page%3D1

1. 走马灯例子内容:

走马灯的一个含义就是控制一列 LED 灯循环点亮，这种循环点亮 LED 灯的过程就是跑马灯效果。

图 1 是开发板 4 个 LED 灯原理图，可以看出四只发光二极管正极接 VDD3.3 负极接如果为 0 的话，二极管正向导通使得发光。所以通过用低电平来扫描这四只管脚就可以实现咱们的走马灯效果。

那么咱们实现走马灯以 500ms 为间隔跳转。

设计思想，我们定义一个计数器，这个计数器最大计数到 500ms，然后去触发一个 4bit 的移位寄存器，使得 4bit 以为寄存器每 500ms 移位一次，把 4bit 移位寄存器绑定到 FPGA 对应 LED 灯的管教即可实现跑马灯。

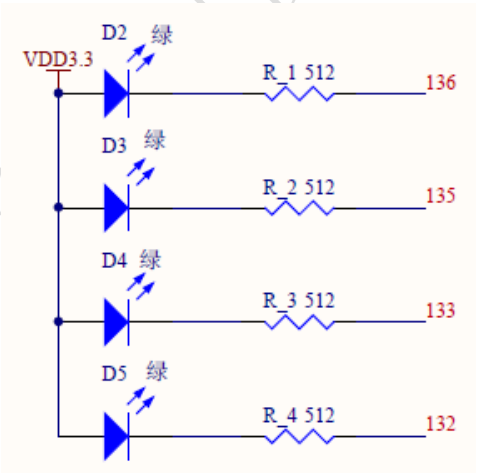


图 1 LED 灯原理图

✧ **带有触发条件的移位寄存器组**

这个跑马灯的关键就是讲移位寄存器，那么我们先把条件触发的移位寄存器组 RTL 图

画出来如图 2，这样我们就能参照 RTL 图很好的去写代码了。确切的说用代码描述出 RTL 图。

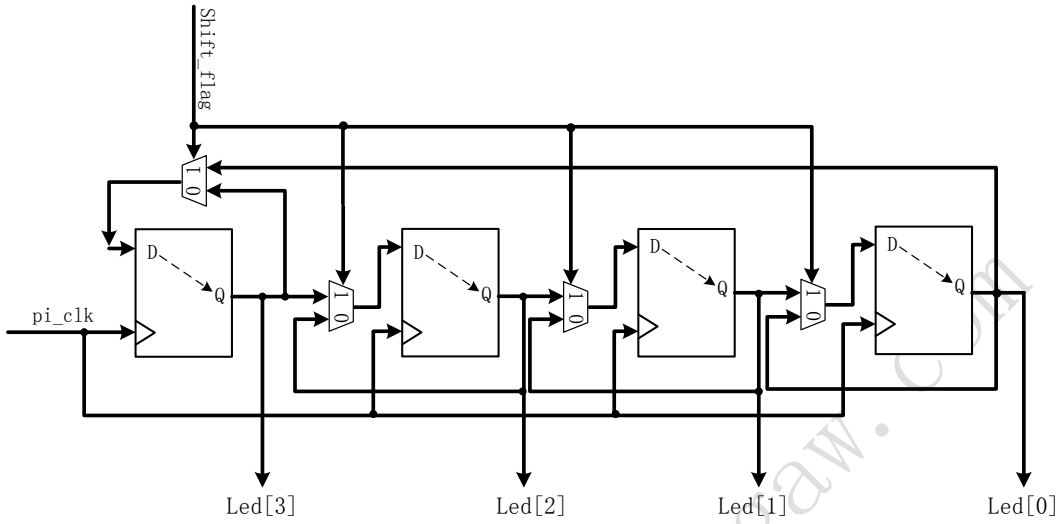


图 2 条件出发 4 位移位寄存器

- Verilog 代码实现

```
//完整代码保存到 design/shift_led.v
//shift_led.v
module   shift_led(
      input     wire          pi_clk,
      input     wire          rst_n,
      output    wire     [3:0]led
);
parameter          div_end=25'd25000000-1;
reg       [24:0]   div_cnt;
reg                div_end_flag;
reg       [3:0]    shift_led=4'b1110;
wire               shift_flag;

always @(posedge pi_clk or negedge rst_n)
      if(rst_n == 1'b0)
            div_cnt<='d0;
      else if(div_end_flag == 1'b1)
            div_cnt<= 'd0;
      else
            div_cnt<=div_cnt + 1'b1;

always @(posedge pi_clk or negedge rst_n)
      if(rst_n == 1'b0)
            div_end_flag <= 'd0;
      else if(div_cnt == (div_end-1))
```

```
            div_end_flag<='d1;
        else
            div_end_flag<='d0;

    assign      shift_flag=div_end_flag;
    always @(posedge pi_clk or negedge rst_n)
        if(rst_n == 1'b0)
            shift_led<=4'b1110;
        else if(shift_flag == 1'd1)
            shift_led<={shift_led[2:0],shift_led[3]};
    assign      led=shift_led;
    endmodule
```

- 带有详细注释代码

```
//带有注释的代码
//shift_led.v
// 这里有一个低电平复位输入信号，此信号会引入到寄存器中控制寄存器的复位；
module   shift_led(
    input     wire          pi_clk,
    input     wire          rst_n,
    output    wire     [3:0] led
);
//参数声明,这里触发时钟 pi_clk 是 50Mh 周期是 20ns 那么跑马灯闪灯切换周期为 500ms
需要 500_000_000ns/20ns=25_000_000 个 pi_clk 周期，所以声明一个参数来作为计数器
终止值的标志。
//语法，关键字 parameter 用来声明参数变量，参数变量只在本模块内有效，在关键字
parameter 之后跟的事参数名称，参数名称是用户自定义的即本例的 div_end 就是参数
名，赋值方式是阻塞赋值；在模块外可以通过关键字 defparam 关键字去更改目标模块
的参数变量。在以后的章节将会讲到，这里先记住有这么个关键字即可。
parameter          div_end=25'd25000000-1;
//寄存器线网的声明，其中 shift_led 通过声明变量时用阻塞赋值赋初始值 4’b1110；
reg       [24:0]   div_cnt;
reg                div_end_flag;
reg       [3:0]    shift_led=4'b1110;
wire               shift_flag;
//分频计数器编程
always @(posedge pi_clk or negedge rst_n)
    if(rst_n == 1'b0)
        div_cnt<='d0;
    else if(div_end_flag == 1'b1)
        div_cnt<= 'd0;
    else
        div_cnt<=div_cnt + 1'b1;
//分频计数器结束标志产生
```

```
always @(posedge pi_clk or negedge rst_n)
    if(rst_n == 1'b0)
        div_end_flag <= 'd0;
    else if(div_cnt == (div_end-1))
        div_end_flag<='d1;
    else
        div_end_flag<='d0;
```

//条件触发的移位寄存器；

//这里 rst_n 异步复位触发移位寄存器复位，当 shift_flag 标志来临时触发寄存器移位一次，这里采用一个位拼接符来描写一个移位寄存器，{}是位拼接符，语法是{变量 A,变量 B,变量 C}其中变量用逗号隔开，最后一个变量后不需要逗号，这样就把变量 A 和变量 B 和变量 C 拼接到一起，拼接后的位宽为 A,B,C 位宽的综合，其中变量 A,B,C 都是 MSB to LSB，拼接后的变量也是 MSB to LSB。

//帮助理解，因为寄存器可以延时数据一个周期，那么当移位标志信号来临时，当前周期寄存器的第 3bit 的值赋值给当前寄存器的第 0bit，第 0bit 赋值给第 1bit，第 1bit 赋值给第 2bit，第 2 比特赋值给第 3bit。那么寄存器值被赋值成功是在标志信号有效的下一个 pi_clk 周期。大家可以结合图 2 来理解此结构

```
assign    shift_flag=div_end_flag;
always @(posedge pi_clk or negedge rst_n)
    if(rst_n == 1'b0)
        shift_led<=4'b1110;
    else if(shift_flag == 1'd1)
        shift_led<={shift_led[2:0],shift_led[3]};
assign    led=shift_led;
endmodule
```

2. Quartus II 软件管教绑定脚本编写

//此文件保存在 quartus_pro/pin/shift_led.txt

//to 指的的是模块的输入输出管脚，location 指的是 FPGA 的输入输出管脚

```
to,location
pi_clk,pin_23
rst_n,pin_69
led[0],pin_136
led[1],pin_135
led[2],pin_133
led[3],pin_132
```

3. 建立 Quartus 工程

建立 quartus 工程的详细过程在第三讲里详细的讲了，如果还不清楚如何建立工程的话请回顾第三讲分频器讲义的第五页。

- 建立工程如图 3，工程名和顶层入口名都是 shift_led

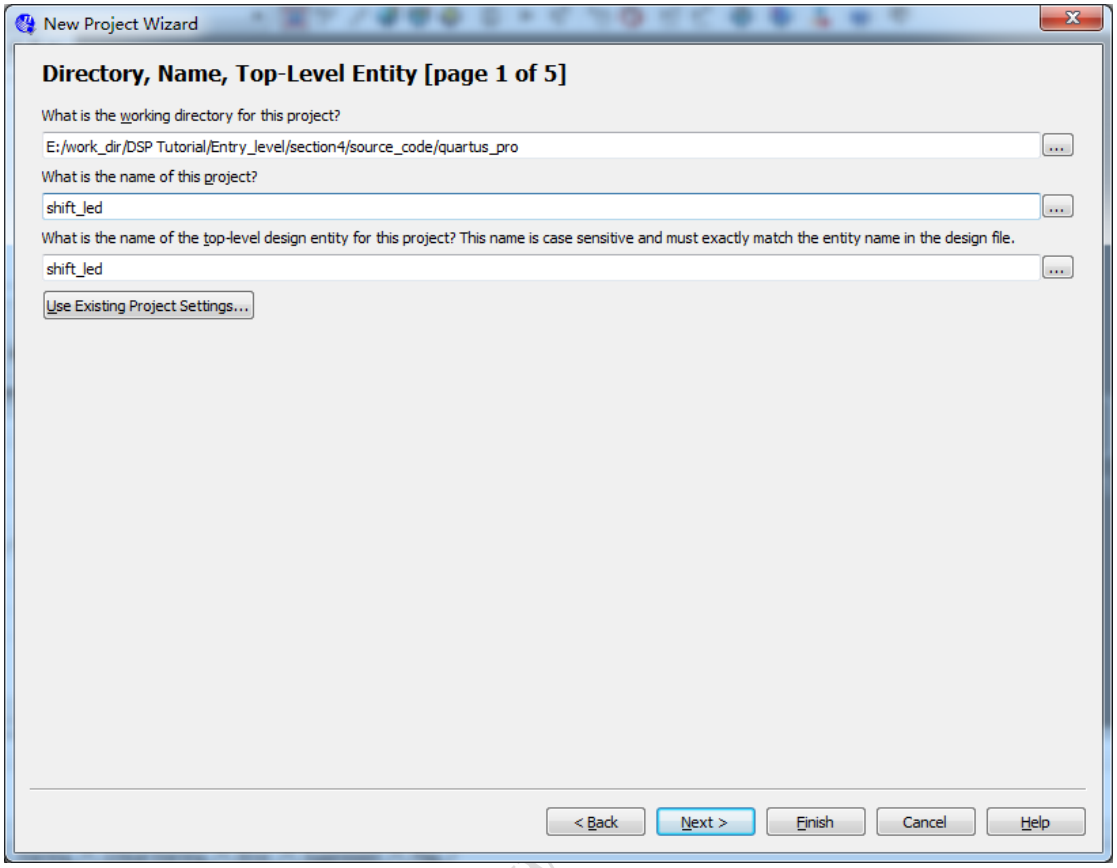


图 3 工程设置

- 加入工程所需的源码文件

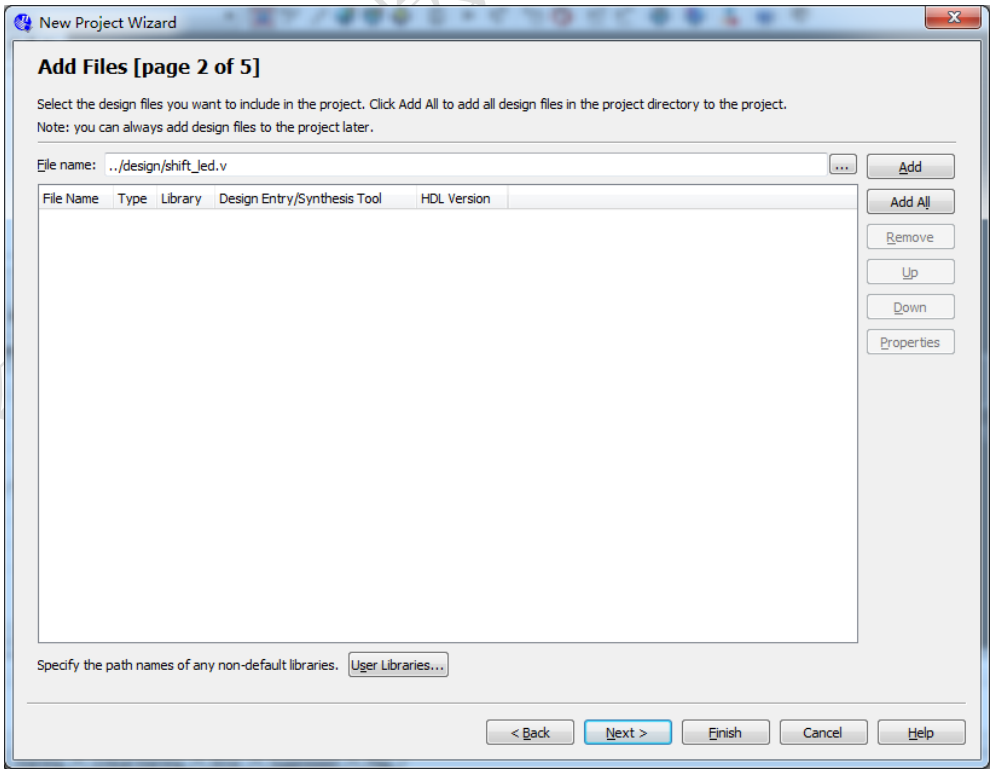


图 4

- 选择器件 cyclone IV E ep4ce6e22c8

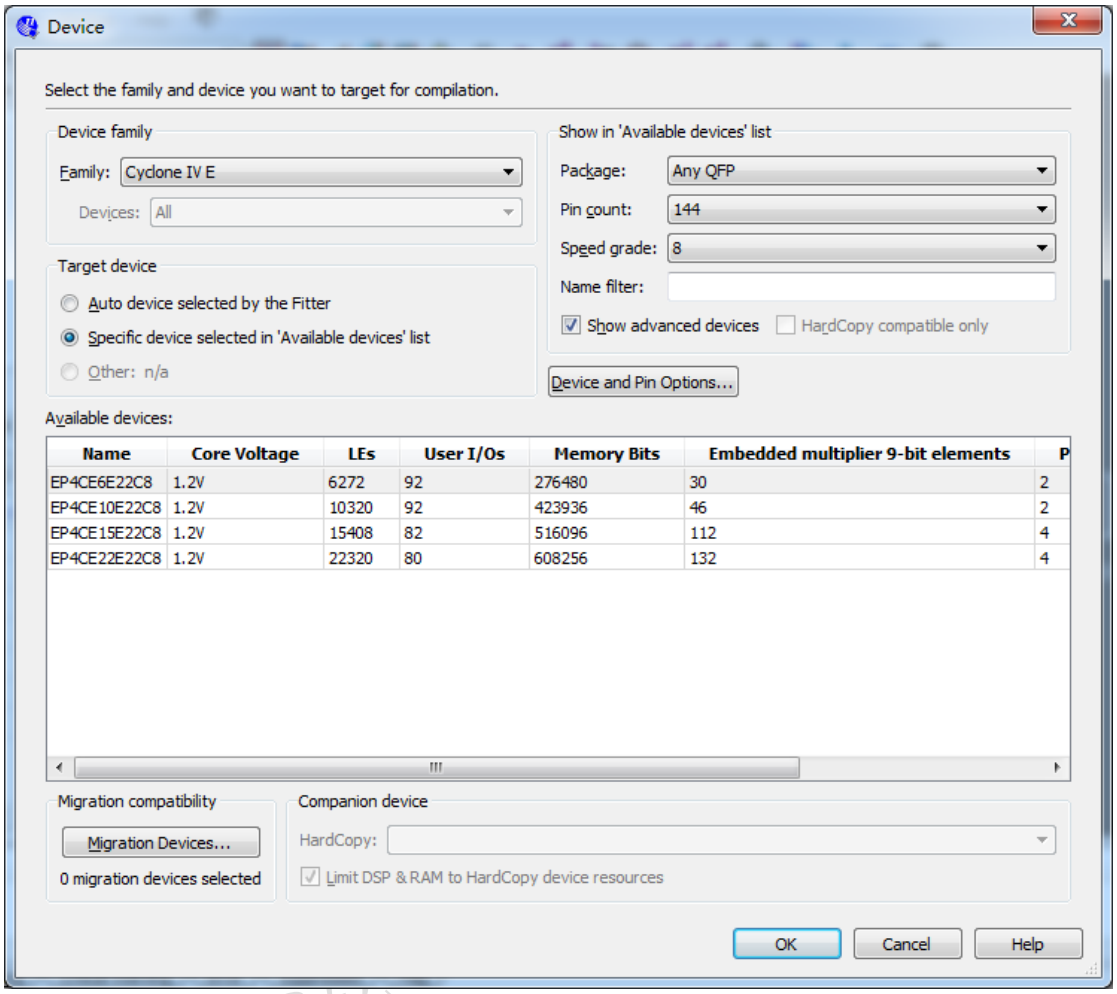


图 5

- 完成工程建立后，导入管脚绑定脚本，assignments->import assignments

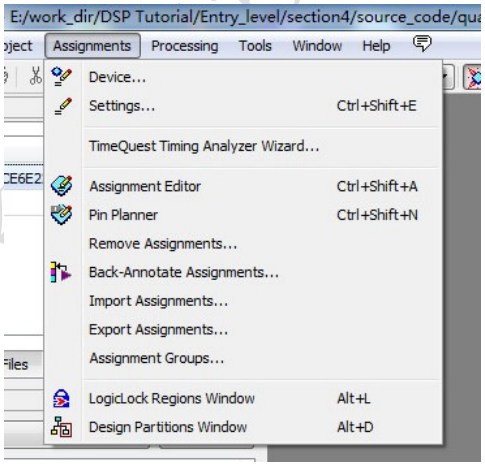


图 6

- 选中 quartus/pin/shift_led.txt 并点击 OK

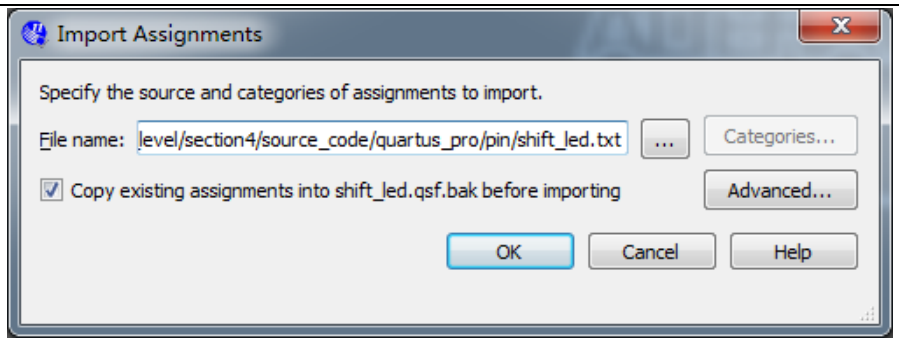


图 7

- 综合工程

在窗口中左下角有一个 task 窗口，如图 8，如果没有的话在 view 菜单中调出即可。然后双击 analysis & synthesis 进行综合程序

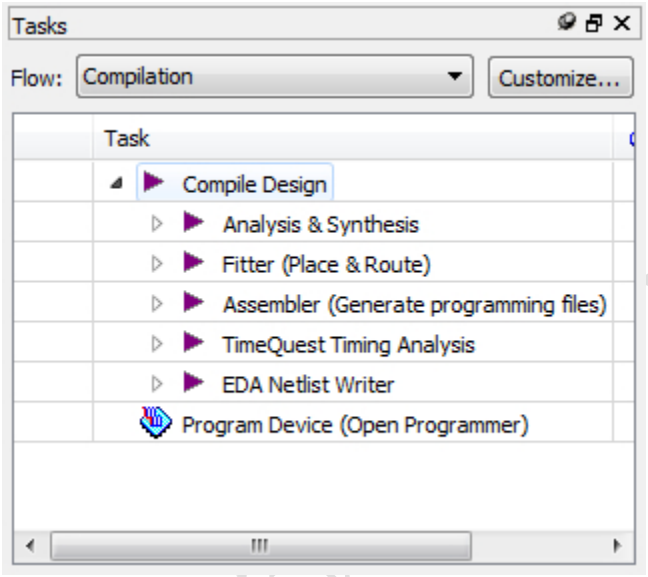


图 8 task 窗口

- 检查管脚绑定是否正确，点击 assignments->assignment editor，出来的结果和图 10 一致即表示绑定正确。

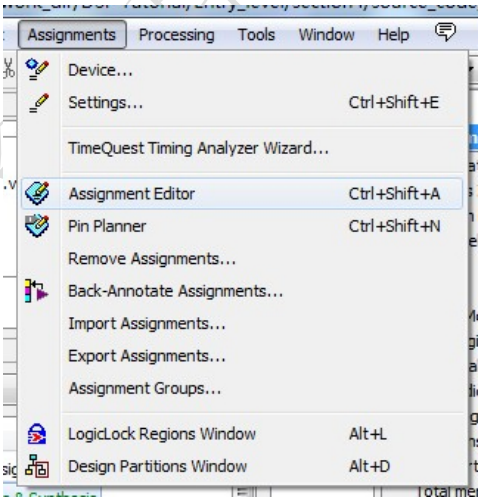


图 9

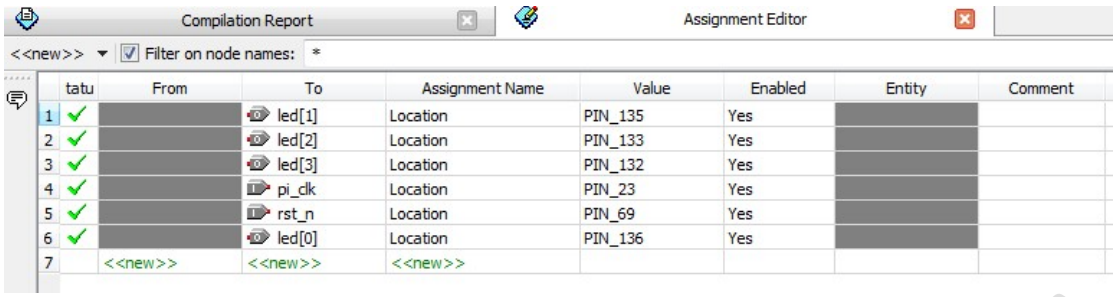


图 10

- 生成编程文件

在 task 窗口 双击 Assembler (Generate programming files)选项，编译结束后如图 11 结果即表示编译成功

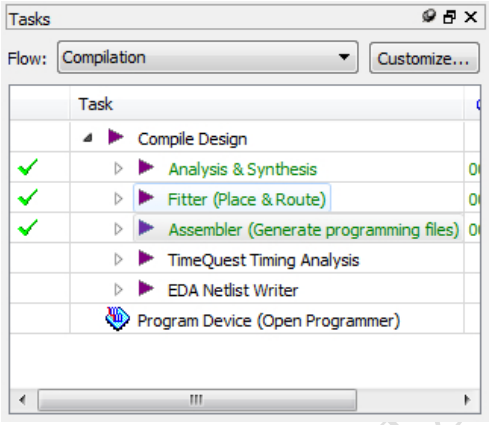


图 11

- 通过 JTAG 下载口下载 sof 文件，sof 文件只是烧写到 FPGA 内部的 sram 中掉电后数据丢失，如果想永久烧写到 flash 中请选择烧写 JIC 或者 POF 文件。

点击菜单 tools->programmer 弹出 programmer 窗口如图 12 所示，此时已经默认了 sof

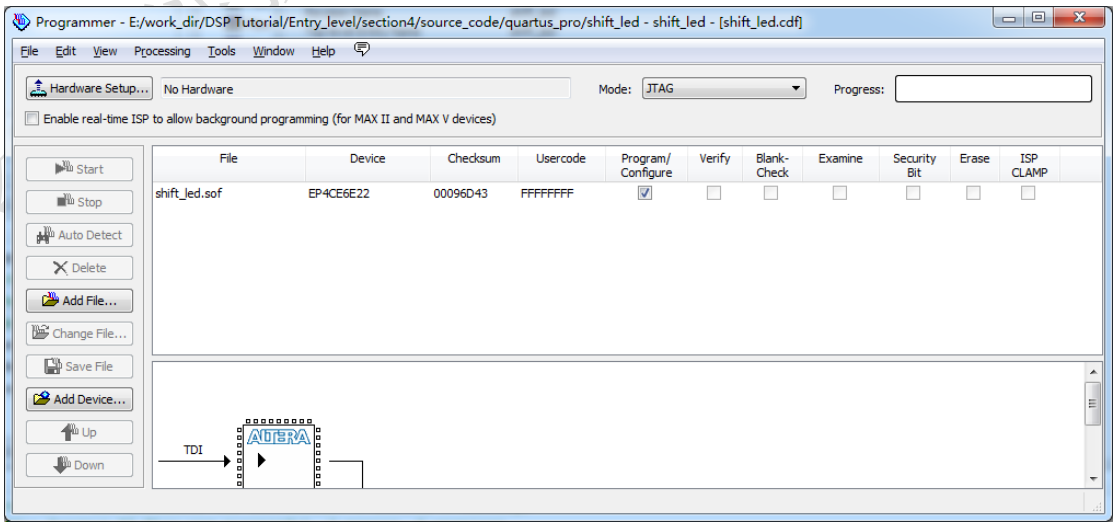


图 12 programmer 窗口

如果没有默认的 sof 可以自行选择，双击 file 列表下的空白处即可选择(不是菜单栏的

file 哦)，如图 13 所示，添加完成 sof 后一定要选中 program/configure 的选择框。

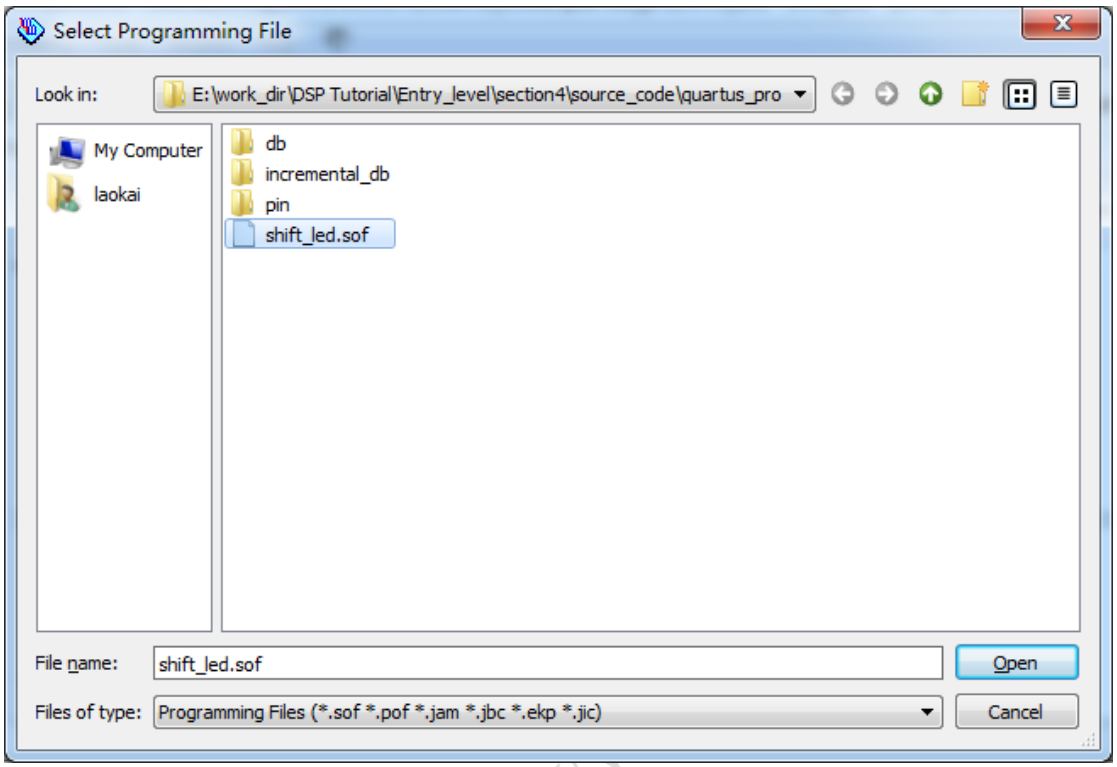


图 13 添加 sof 窗口

- 把 altera USB blaster 下载线插入计算机 USB 插口中，并安装 altera 下载线驱动
  - 右键点击我的计算机，选择属性，在弹出窗口中选择设备管理器，打开设备管理器窗口如图 14 所示，存在一个带有叹号的 usb_blaster 的设备
  - 右键点击它选择更新驱动程序，选择浏览计算机以查找驱动程序如图 15 所示
  - 打开您安装 quartus 的路径，找到 Quartus11.1\quartus\drivers\usb-blaster 这个路径(选择这里就可以，不需要再选择 x64 和 x32，系统会自动选择)如图 16。
  - 点击下一步进行安装如果提示警告选择始终安装此驱动程序如图 17
  - 安装成功叹号消失如图 18 所示，然后回到上一步的 program 窗口

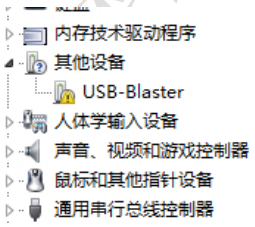


图 14

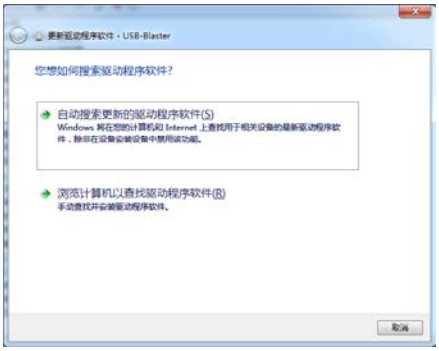


图 15

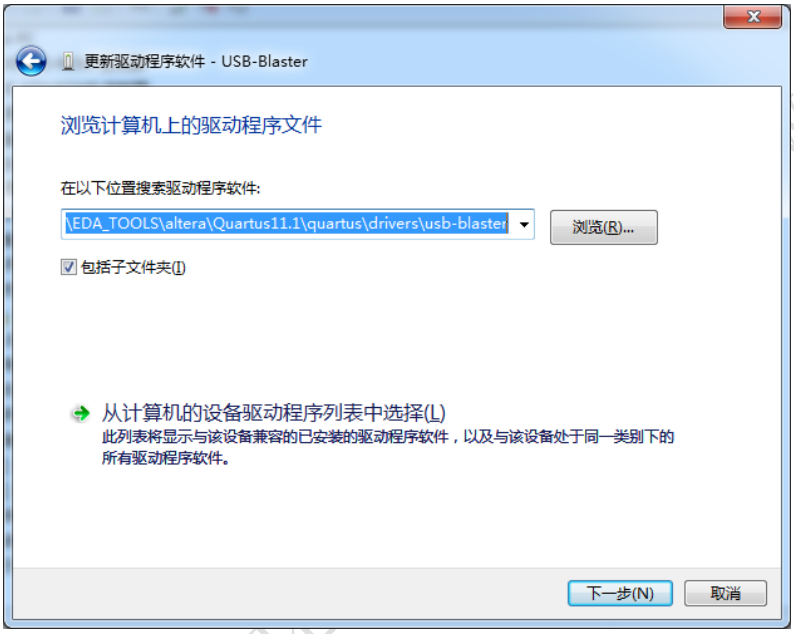


图 16

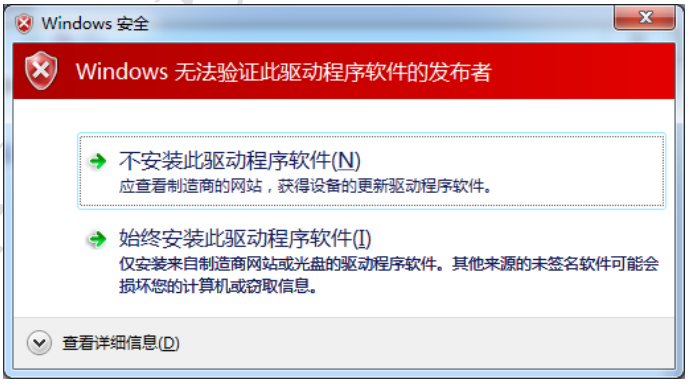


图 17

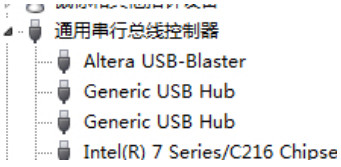
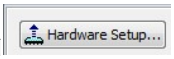
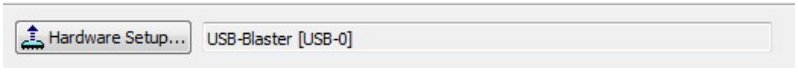


图 18

- 在 programmer 窗口中左上角找到 hardware setup 按钮如 

打开后如图 19 所示(此时一定要连接好 usb blaster 下载线)，
选择 currently selecte hardware 这里有一个下拉选项，选择如图所示 usb-blaster 即可，选择好后 close 即可。此时 programmer 窗口的 hardware setup 栏里就有 usb-blaster 显示。 

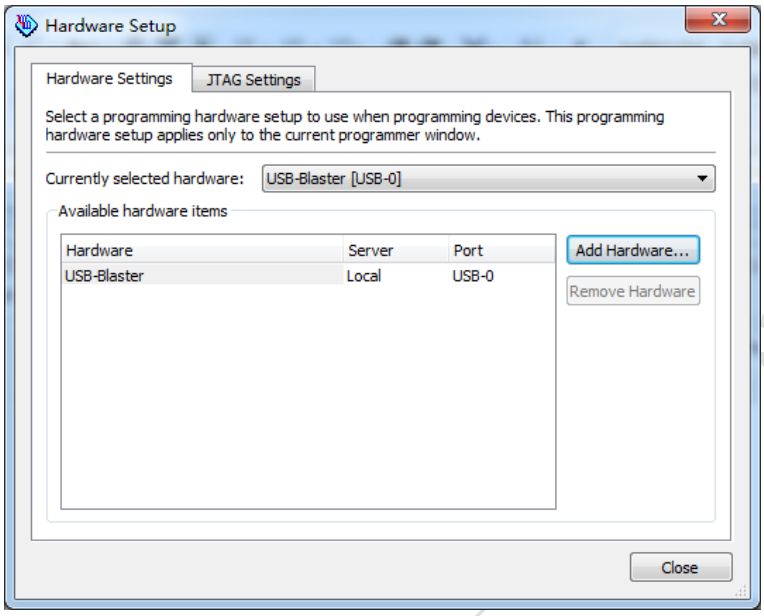


图 19 usb blaster 选择

- 把下载线连接到 ZX-2 开发板的 jtag 编程口
在图 20 中指示标号为 10 的位置是 JTAG 编程口

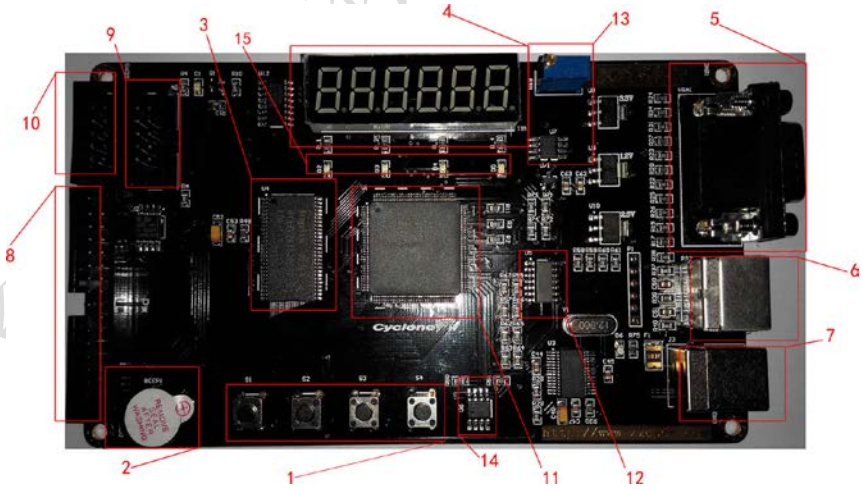
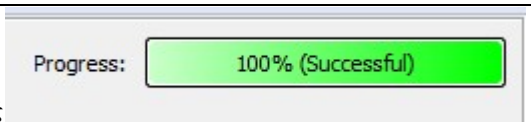


图 20 ZX-2 开发板

- 下载 sof 文件

点击左侧的 start 按钮 



下载成功后右侧的进度条会显示
此时观察 led 灯是不是跑起来了。

- 下载固化 JIC 文件

回到 quartus II 工程主窗口点击 FILE->convert programming file,打开后如图 21 所示

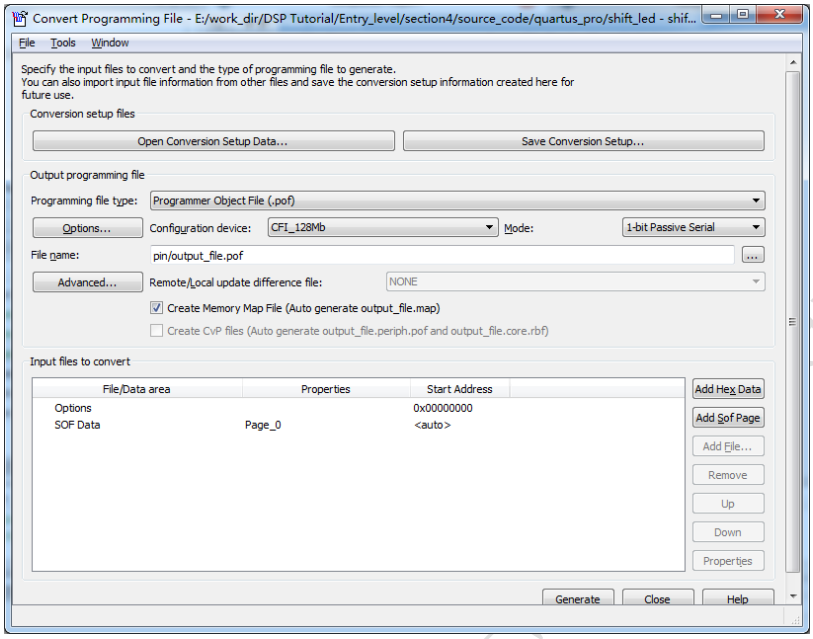
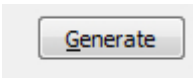


图 21

- ✧ 设置 programming file type: JTAG Indirect Configuration File (.jic)
- ✧ 选择 flash 容量 configuration device:EPCS16
- ✧ Mode：Active Serial
- ✧ File_name：shift_led.jic
- ✧ 选中 flash loader，点击右侧的 add device
- ✧ 选择 cyclone IV E 和 EP4CE6 如图 22 所示，选择 OK 回到 convert 窗口
- ✧ 选中 sof data 点击右侧 add file 添加 SOF 文件



- ✧ 选中之后如图 23 所示，这时选择右下角 Generate 按钮
- ✧ 如果生成成功会有提示窗口如图 24
- ✧ 切换到 programmer 窗口，把之前的 sof 文件选中，右键选择 delete 删除之。之后在 flie 列表空白处双击添加入刚才生成的 shift_led.jic 文件。注意选中 program/configure 的选项框才能进行下载如图 25 所示
- ✧ 点击 start 按钮进行下载 jic 到 flash 中，下载完成后把开发板断电后再上电看看是不是 led 灯还在跑，说明程序已经固化到 flash 中了。

课后练习:大家可以参考 JIC 下载方法，把 pof 生成，并通过 AS 口下载，提示在 programmer 里需要选择 AS 模式。ZX-2 的开发板 AS 口是 JTAG 口旁边那个。

这次的课程到这里就结束了，大家有什么问题请到至芯科技论坛提交问题，哪里会有专门的老师给大家回答问题。www.fpga.com

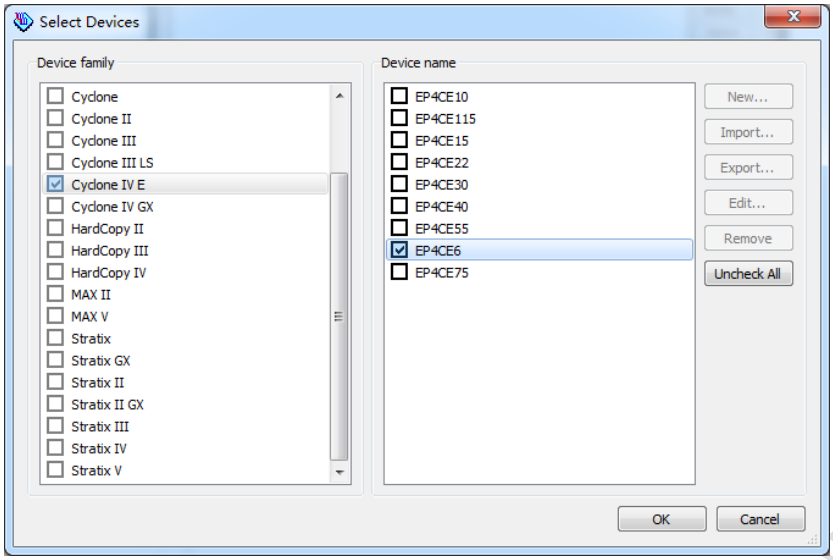

Select Devices
Device family
Cyclone
Cyclone II
Cyclone III
Cyclone III LS
Cyclone IV E
Cyclone IV GX
HardCopy II
HardCopy III
HardCopy IV
MAX II
MAX V
Stratix
Stratix GX
Stratix II
Stratix II GX
Stratix III
Stratix IV
Stratix V
Device name
EP4CE10
EP4CE115
EP4CE15
EP4CE22
EP4CE30
EP4CE40
EP4CE55
EP4CE6
EP4CE75
New...
Import...
Export...
Edit...
Remove
Uncheck All
OK
Cancel

图 22

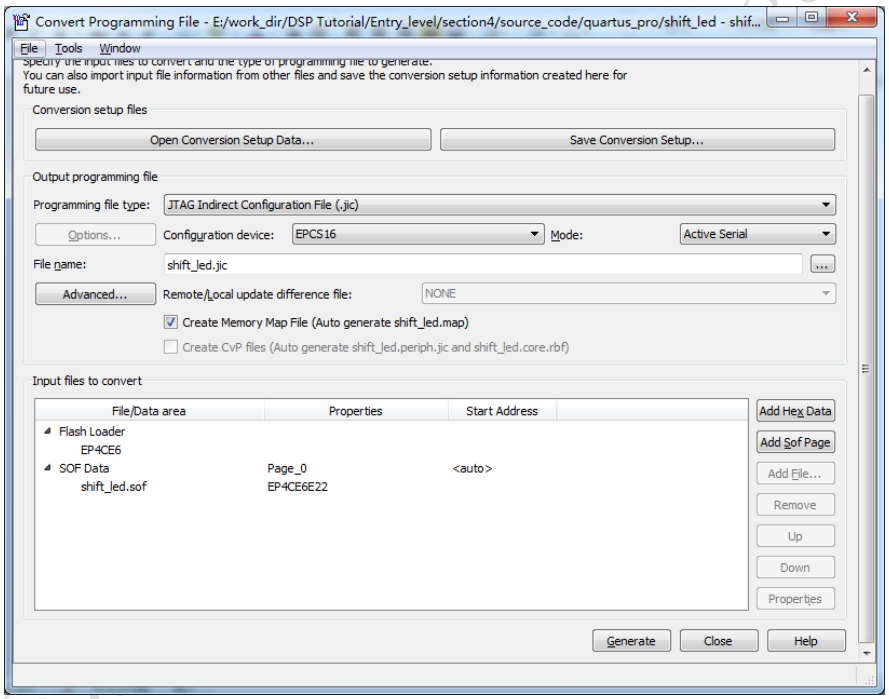

Convert Programming File - E:/work_dir/DSP Tutorial/Entry_level/section4/source_code/quartus_pro/shift_led - shif...
File
Tools
Window
You can also import input file information from other files and save the conversion setup information created here for future use.
Conversion setup files
Open Conversion Setup Data...
Save Conversion Setup...
Output programming file
Programming file type:
JTAG Indirect Configuration File (.jic)
Options...
Configuration device:
EPCS16
Mode:
Active Serial
File name:
shift_led.jic
Advanced...
Remote/Local update difference file:
NONE
Create Memory Map File (Auto generate shift_led.map)
Create CvP files (Auto generate shift_led.periph.jic and shift_led.core.rbf)
Input files to convert
File/Data area
Properties
Start Address
Flash Loader
EP4CE6
SOF Data
shift_led.sof
Page_0
EP4CE6E22
<auto>
Add Hex Data
Add Sof Page
Add File...
Remove
Up
Down
Properties
Generate
Close
Help

图 23

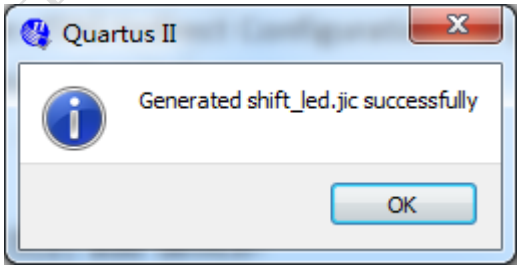

Quartus II
Generated shift_led.jic successfully
OK

图 24