FPGA 入门课程 3-分频器

第三节分频器

➢ 8 分频器;

例:输入为 50Mhz 占空比为(高低电平持续时间的比值)50%的时钟,将其 8 分频后输出

分析:将 50Mhz8 分频频率为 50/8=6.25Mhz 周期为 20ns*8=160ns，高电平持续时间是 80ns，低电平持续时间是 80ns，因此可以用 50Mhz 作为计数器的触发时钟，当从 0 计数到 3 时，保持高电平即 20ns*4=80ns，计数从 4 到 7 时是低电平即可实现 8 分频。

注释:由于初学者对波形理解比较模糊，建议大家先画出波形在进行代码编写，这是很好的设计习惯。

绘制波形如图 1

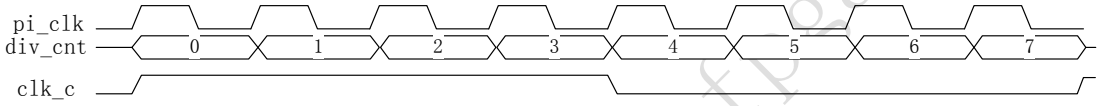

图 1 分频器波形设计

当把分频器的波形图绘制出来之后,是不是大家已经觉得分频器的代码实现已经很简单了，只需要实现一个 divider counter 和 一个根据 divider counter 计数器数值产生的高低电平信号即可。

➢ Verilog 代码实现

```
//完整的代码
//frequence_divider_8.v
module   frequence_divider_8(
      input     wire          pi_clk,
      input     wire          rst_n,
      output    wire          po_clk_r,
      output    wire          po_clk_c
);
reg       [2:0]     div_cnt;
reg                 clk_r,clk_c;

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

always@(div_cnt)
      if(div_cnt<3'd4)
            clk_c = 1'b1;
```

```
    else
        clk_c=1'b0;
always @(posedge pi_clk or negedge rst_n)
    if(rst_n==1'b0)
        clk_r<=1'b0;
    else if(div_cnt<3'd4)
        clk_r<=1'b1;
    else
        clk_r<=1'b0;
assign  po_clk_c = clk_c;
assign  po_clk_r = clk_r;
endmodule
```

详细注释代码

```
module  frequence_divider_8(
    input   wire        pi_clk,
    input   wire        rst_n,
    output  wire        po_clk_r,
    output  wire        po_clk_c
);
reg     [2:0]   div_cnt;//分频计数器
reg             clk_r,clk_c;//clk_r 是综合为寄存器，clk_c 综合为组合逻辑；r 指的是 registe
c 指的是 combination；

always @(posedge pi_clk or negedge rst_n)
    if(rst_n==1'b0)//这里是异步复位，由于敏感列表里有 rst_n 的下降沿会触发复位，此复
位脉冲与 pi_clk 并不同步。
        div_cnt<='d0;//这里使用'd0 表示此数值的位宽自动等于 div_cnt 的位宽
    else
        div_cnt <= div_cnt + 1'b1;//这里因为只有 3 位所以计数到 7 后自动归零，因为
3'b111+1'b1=4'b1000 那么又因为寄存器只有 3 位所以高位被丢弃,只剩低三位即 3'b000;

//下面这个 always 块包含的是一个组合逻辑，是由敏感列表里 div_cnt 的各个数据线的电平
变化触发的，电平触发的逻辑电路是立刻执行的。这里使用的是阻塞赋值即等号(=),阻塞的
意思就是当触发条件满足的时刻等号右边的值如果没有计算完成或者叫未准备好，是不会发
生赋值操作的。映射到电路的理解就是当某信号满足触发条件开始计算，计算的延迟是 T1
从计算电路到目标被赋值电路的路径延迟是 T2，那么也就是说目标被赋值电路在满足条件
的时刻到被赋值时刻需要经历 T1+T2 这么长时间，在 T1+T2 这段时间内目标被赋值电路是被
阻塞状态。由于此特性阻塞赋值基本都是用到组合逻辑中的。
always@(div_cnt)
    if(div_cnt<3'd4)
        clk_c = 1'b1;
```

```
        else
            clk_c=1'b0;
```

//下边的这个 always 块包含的是一个时序逻辑，是由时钟 pi_clk 上升沿触发的；这里用的是非阻塞赋值(<=)；非阻塞赋值配合沿触发会被综合成 D 触发器(针对 FPGA 的综合工具)，D 触发器如图 2，大家可以回去查阅，当时钟沿到来时，无论非阻塞赋值符号右边是否计算完成都会把此时刻的值打入到寄存器 D 端，此时如果右端没有计算完成那么就称之为时序违例,D 端是 D 触发器的输入端，Q 端是 D 触发器的输出端。时序逻辑基本都是使用非阻塞赋值。

```
always @(posedge pi_clk or negedge rst_n)
    if(rst_n==1'b0)
        clk_r<=1'b0;
    else if(div_cnt<3'd4)
        clk_r<=1'b1;
    else
        clk_r<=1'b0;
assign    po_clk_c = clk_c;
assign    po_clk_r = clk_r;
endmodule
```

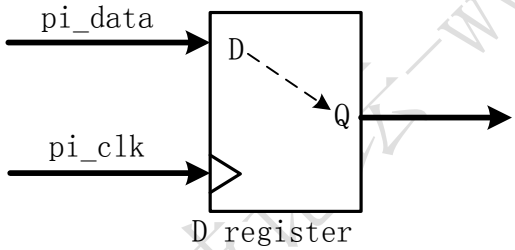

图 2 D 触发器表示图

测试时钟和复位信号的产生

完整代码

```
//tb_frequence_divider_8.v
`timescale    1ns/1ns
module   tb_frequence_divider_8();
reg              clk;
reg              rst_n;
wire             clk_r,clk_c;
initial
begin
    clk=0;
    rst_n=0;
    #100
rst_n=1;
end
```

```
always #10 clk <= ~clk;
frequence_divider_8 fd8_inst(
    .pi_clk(clk),
    .rst_n(rst_n),
    .po_clk_r(clk_r),
    .po_clk_c(clk_c)
);
endmodule
```

详细注释的代码

```
//tb_frequence_divider_8.v
`timescale    1ns/1ns
module   tb_frequence_divider_8();
reg          clk;
reg          rst_n;
wire         clk_r,clk_c;
initial
begin
    clk=0;
    rst_n=0;
    #100 //在 begin end 标签内是顺序执行语句,仅仅用于仿真不可综合，#是延迟信号的关键字后边跟的数字是延迟时间，单位是在 timescale 中定义。这里指延迟 100ns 后触发下一条语句，即延迟 100ns 后 rst_n 信号被拉高。
    rst_n=1;
end
always #10 clk <= ~clk;
//被测模块的例化
frequence_divider_8 fd8_inst(
    .pi_clk(clk),
    .rst_n(rst_n),
    .po_clk_r(clk_r),
    .po_clk_c(clk_c)
);
endmodule
```

- Modelsim 功能仿真结果波形图

在第二节中给大家详细的讲了最基本的通过创建 Project 来仿真功能模块。本节的讲义里就不在一一的截图 modelsim 操作流程，在视频中可以看到我的详细操作流程。

大家仔细看波形 clk_r 和 clk_c 为什么会相差一个时钟周期呢？

答案:clk_r 是寄存器，当满足条件时，触发时钟把数据线上的数据打入到 D 端，也就是当前满足条件后寄存器 D 端被写入 1'b1，而此时刻 Q 端存储的上一拍的数据被打出也就是 1'b0 被打出，在下一个时钟上升沿到来时，Q 端会把 1'b1 打出。因此会延迟 1 个 pi_clk 周期。这是 D 触发器特性。

clk_c 由于是组合逻辑，当满足条件时，数据被赋值只跟路径延迟和组合逻辑处理延迟有关,

由于是功能仿真，没有加入延迟文件，所以组合逻辑产生的 clk_c 满足条件的时刻即被赋值。在实际电路中路径延迟和组合逻辑处理延迟是很大的。

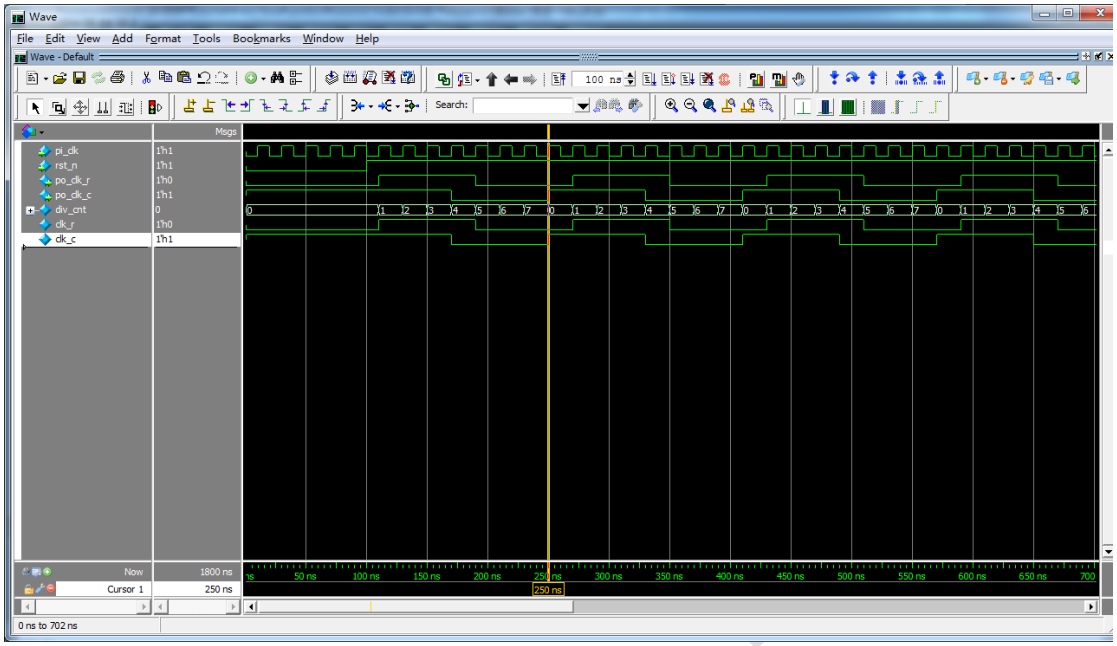

图 2 Modelsim 功能仿真波形

- **应用 Quartus II 软件进行电路综合布局布线生成带有延迟文件的后仿真文件**

1. 在 design 和 sim 目录的同级目录建立一个文件夹命名为 Quartus_pro，打开 quartus II 软件如下界面如图 3

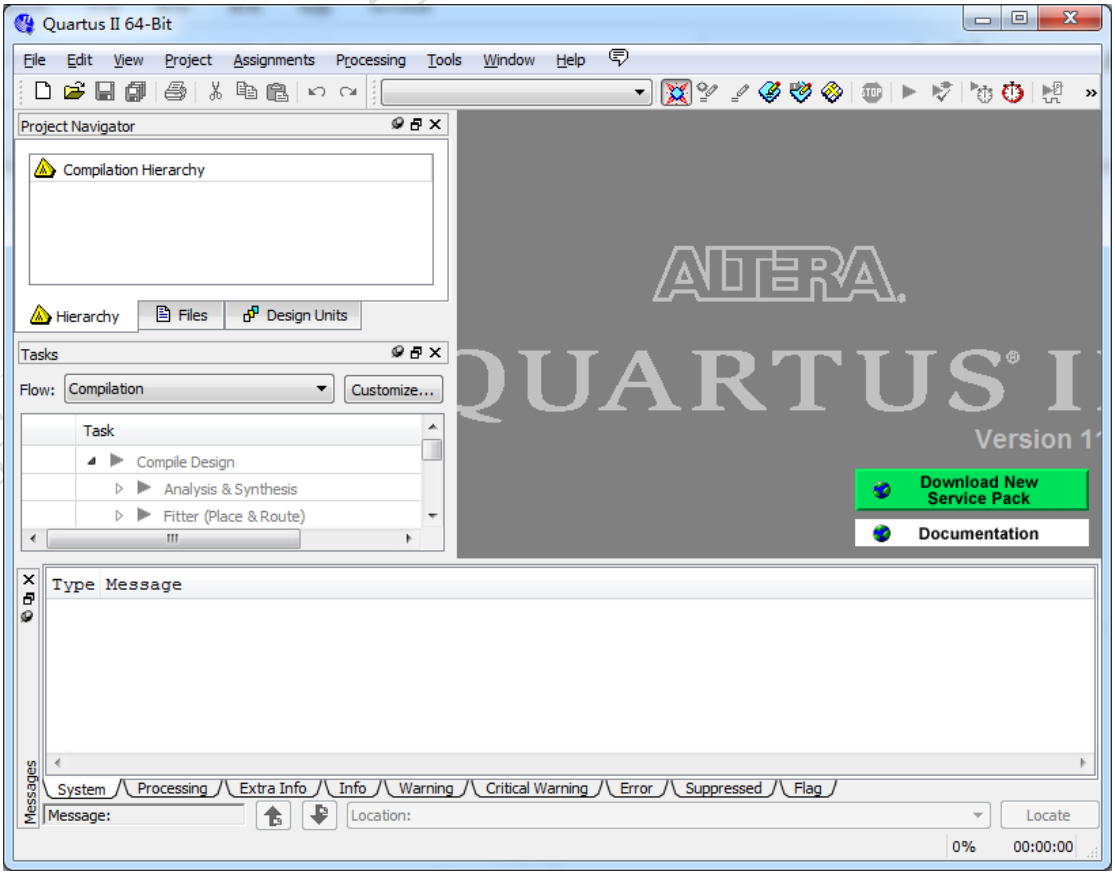

图 3 Quartus II 打开界面

2. 新建 quartus 工程

✧ 选择 FILE->New Project Wizard,如图 4

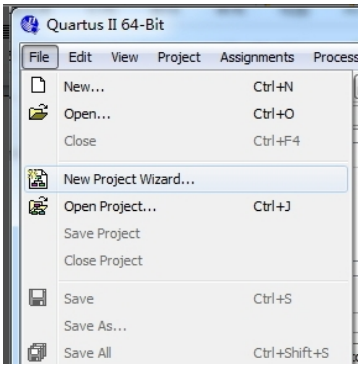

图 4 新建工程

✧ 工程设置介绍页选择 NEXT 即可

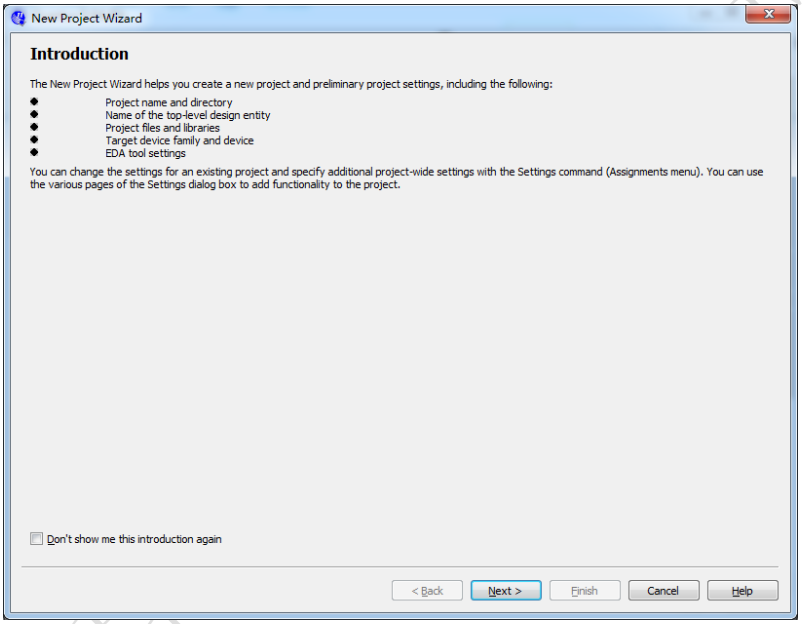

图 5 工程设置介绍页

✧ 工程名称,路径,顶层入口设置

工程路径选择之前建的 quartus_pro 文件夹

工程名称填写 frequence_divider_8

顶层入口填写 frequence_divider_8

注释:如果顶层入口与顶层模块的模块名不一致的话会出现编译错误。

如果图 6

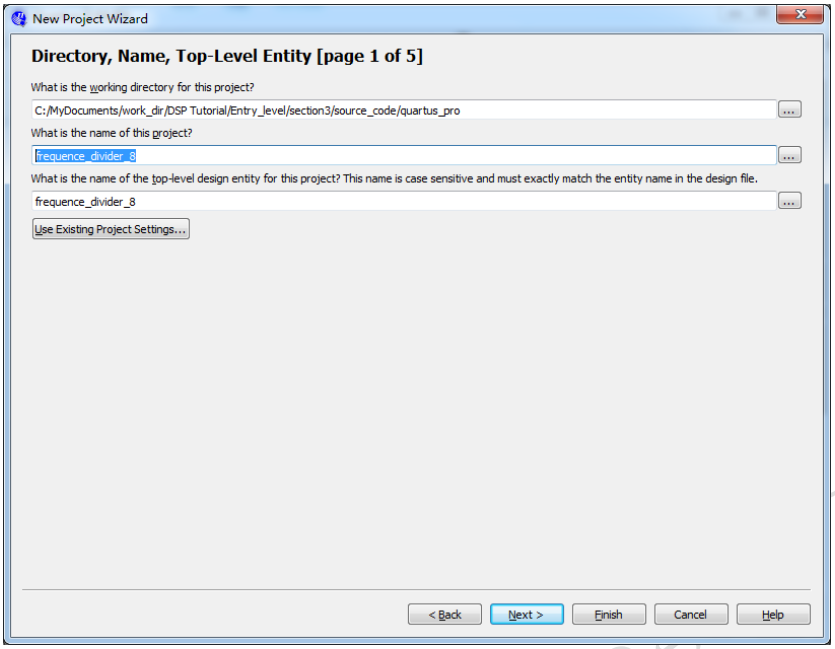

图 6 工程设置界面

✧ 添加程序文件

选择 design 目录下的 frequence_divider_8.v

选择完成后一定要记住点击 ADD 按钮把文件加入到工程。

如图 7

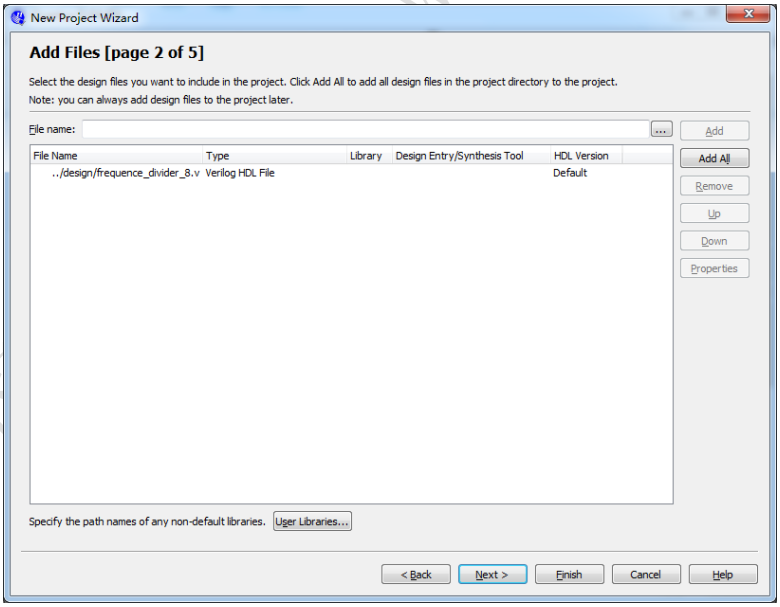

图 7 程序文件添加

✧ 选择相应 FPGA 器件

至芯科技推出一款超级开发板，此开发板价格极低(100 元左右)就是满足初学者学习的需求，而又不需要高额投资。这里选择此款开发板的目标器件，cyclone IV EP4CE6E22C8

封装选择 QFP

Pin out 选择 144

Speed grade 选择 8

如图 8

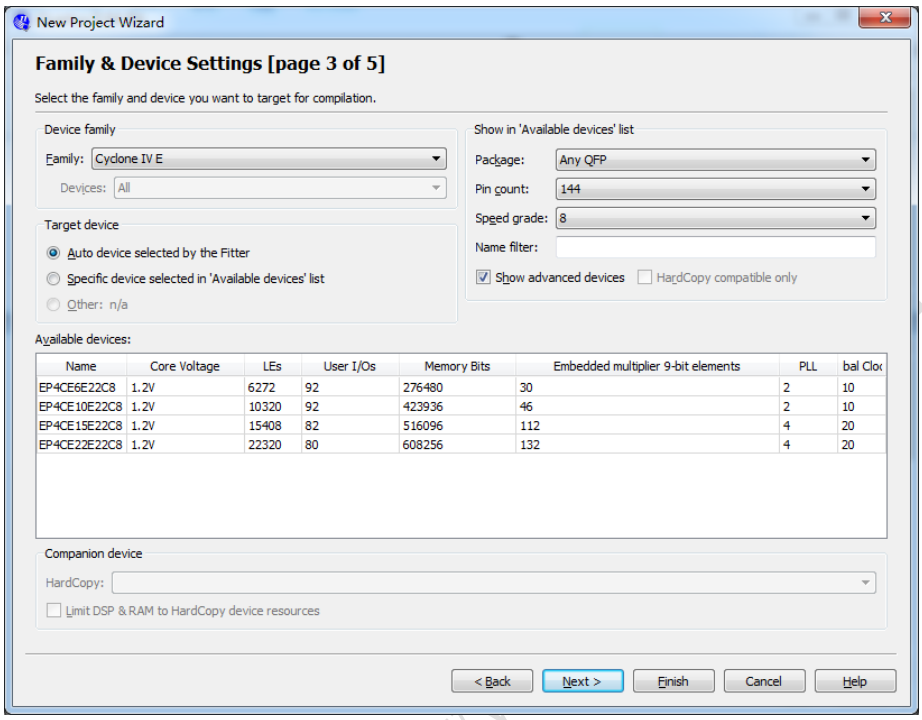

图 8 器件的选择

✧ 这一步是设计工具设置，此步骤完全默认选择 Finish 即可如图 9

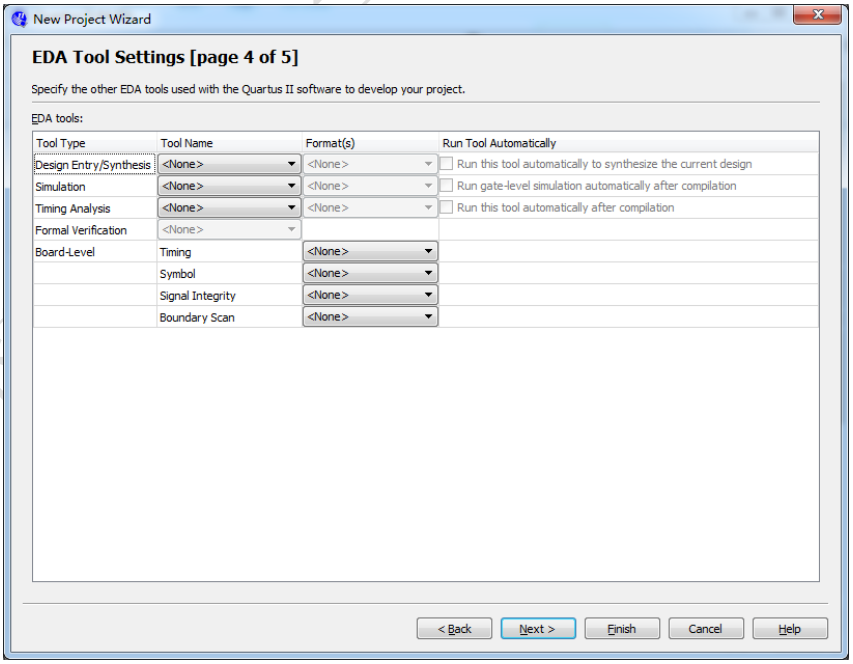

图 9 工具设置界面

✧ 工程设置完成界面，左侧有两个子窗口一个是 project Navigator 和 task，其中 project navigator 会把工程内的所有程序模块列出来，本程序只有一个模块。Task 窗口有 Compile Design 选项，此选项是编译工程选项，

■ 子选项 Analysis & Synthesis 综合和语法分析，对模块进行语法分析然

后综合处 RTL 电路

- Fitter (Place & Route)是布局布线，把 RTL 电路布局到实际的门电路和寄存器中，实现 FPGA 器件的映射
- Assembler (Generate programming files)这个选项可以产生符合 altera 标准的相应 fpga 编程程序。
- TimeQuest Timing Analysis 这个选项会生成时序分析相应的报表。
- EDA Netlist Writer 此选项可以生成网表文件

注释:如果没有此两个窗口可以点击 view->utility windows->然后选择 project Navigator 和 task 即可调出此窗口。

如图 9

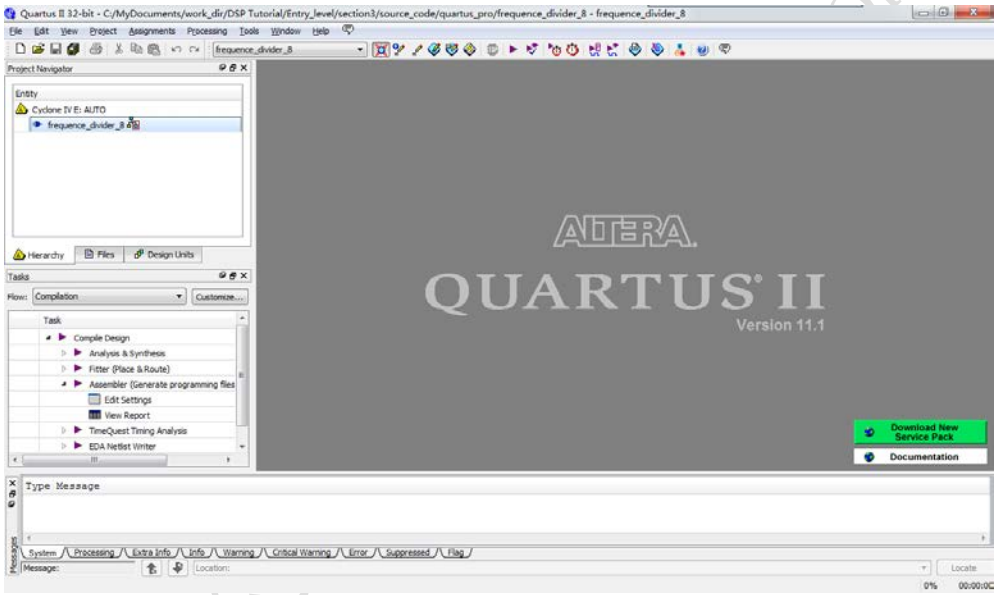

图 9 工程界面

✧ 设置生成 modelsim 后仿真的网表文件

1. 点击 assignments 菜单->setting->simulation
2. Tool name 选择 modelsim
3. Format for output netlist 选择 Verilog HDL Time scale 选择 1ps
4. 其他默认选择 OK

如图 10 设置仿真网表文件

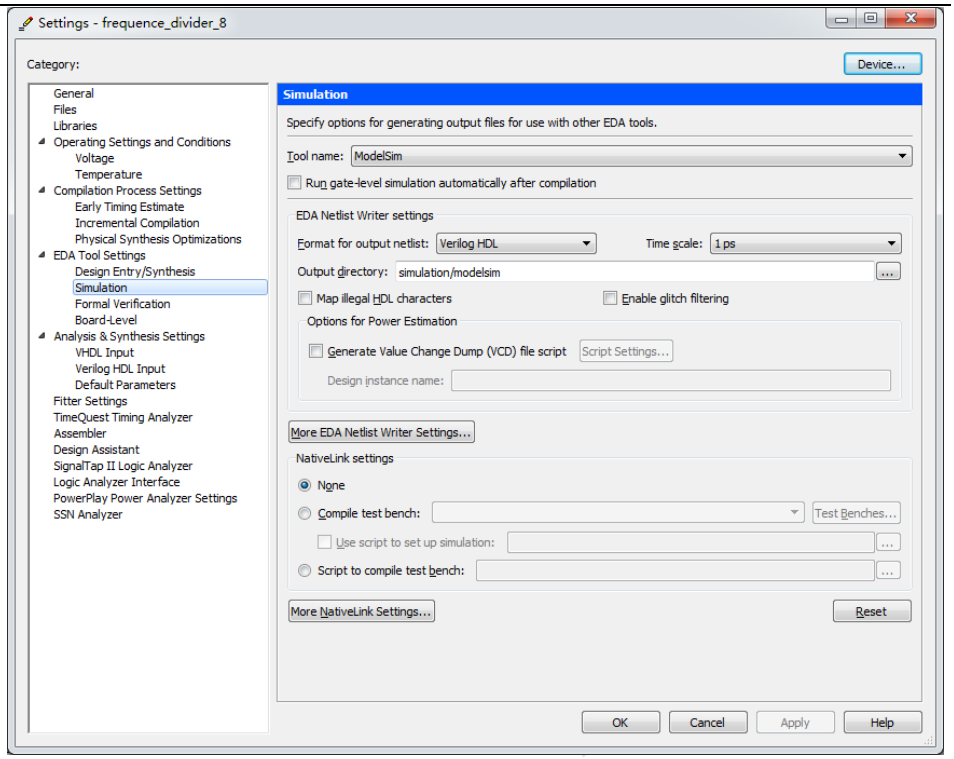

图 10 设置仿真网表文件

✧ 生成网表

双击任务栏的 EDA Netlist Writer，此时这里会显示编译进度

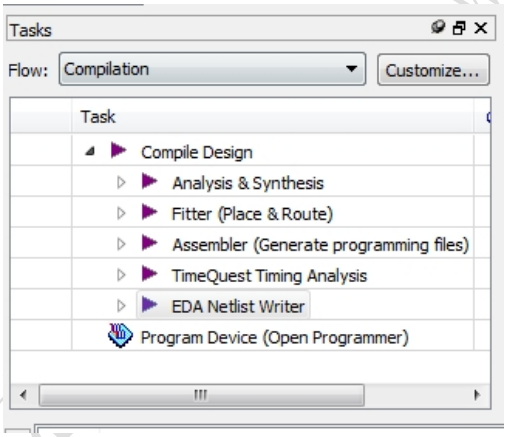

图 11 启动编译过程

编译完成后任务栏状态如图 12

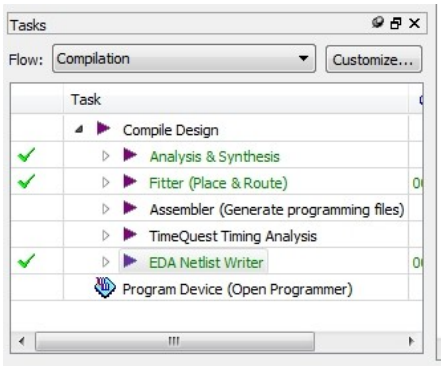

图 12 状态栏

✧ 将网表文件和延时文件拷贝到 SIM 目录

打开 your path/quartus_pro/simulation/modelsim

找到 frequence_divider_8.vo 文件,网表文件

和 frequence_divider_8_v.sdo 文件,延时文件

复制这两个文件到 your path/sim/ 目录

➢ **应用 modelsim 和上一步产生的网表和延时文件进行后仿真**

1. 在 sim 目录内建立一个新的文件夹命名为 altera_lib
2. 打开 quartus II 安装工具的安装目录，进入如下目录 your path\Quartus II 11.1\quartus\eda\sim_lib

   在此目录中找到如下三个文件

   altera_mf.v

   altera_primitives.v

   cycloneive_atoms.v

   并把他们复制到刚刚创建的 altera_lib 目录中。
3. 打开 modelsim 在 sim 目录创建一个工程，命名不要和之前的功能仿真名称重复。创建工程方法在第 2 讲中有详细讲解，这里就不在赘述，不懂得请参看视频。
4. 把如下文件加入到工程中

   Sim/Altera_lib 目录下的：

   altera_mf.v

   altera_primitives.v

   cycloneive_atoms.v

   sim 目录下的:

   frequence_divider_8.vo//此文件就是网表文件

   tb_frequence_divider_8.v

   如图 13

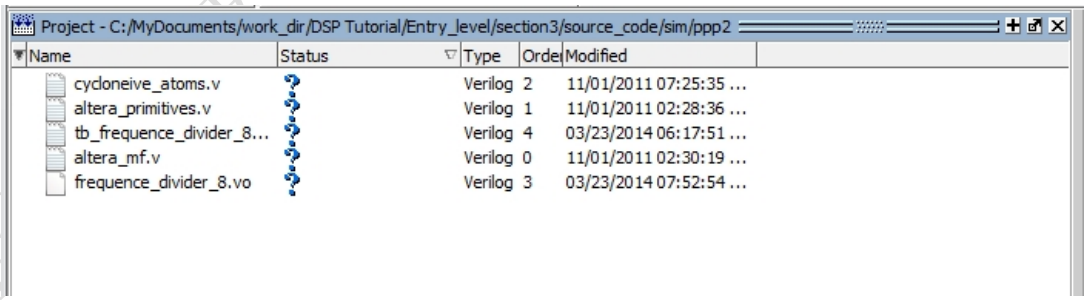

图 13 添加的编译文件列表

5. 编译以上列表文件

   编译后如下图 14

   选择 compile->compile all

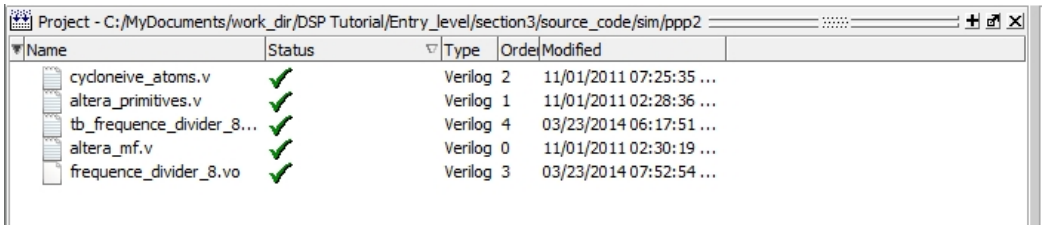

图 14 编译完成后

6. 设置延时文件

- 第一步点击 simulate->start simulation 如图 15

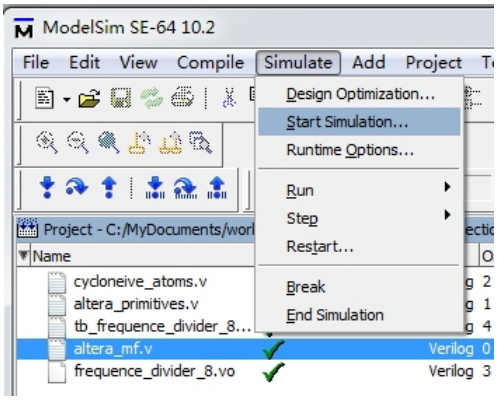

图 15 开始仿真设置选项

- 进入仿真选项列表，选择 SDF 选项卡如图 16

点击 add，选中 frequence_divider_8_v.sdo//延时文件

并把下边的 Disable SDF WARNINGS 和 Reduc SDF errors to warning，选项选中

然后切换到 Design 选项卡

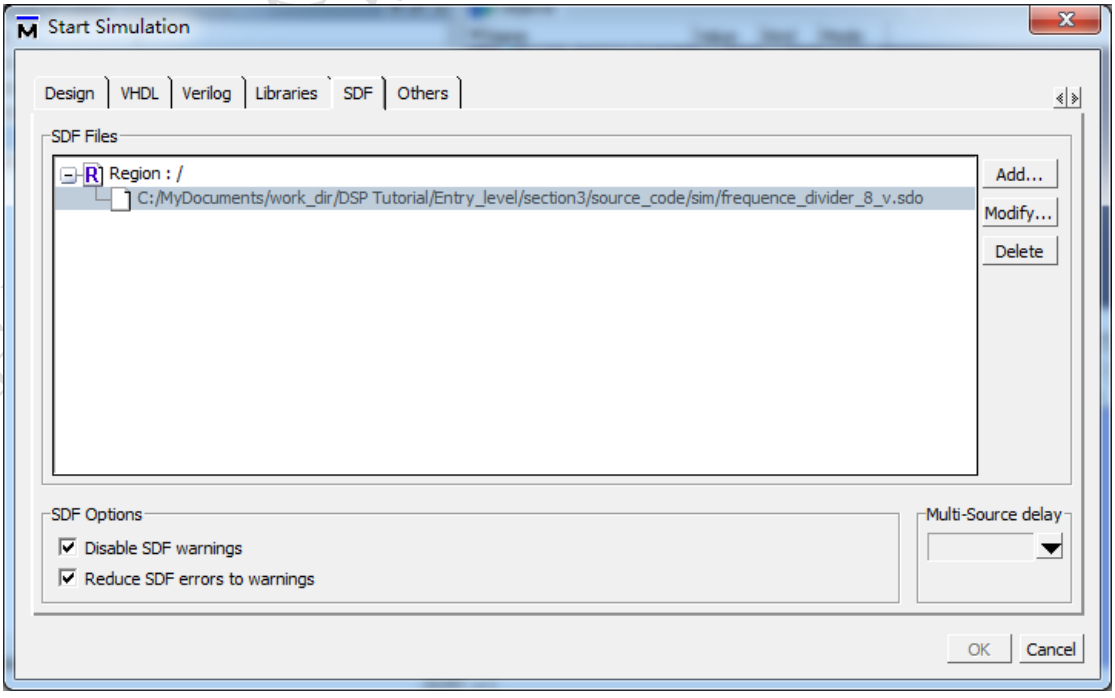

图 16 添加延时文件

在 design 选项卡中展开 work 库的目录找到 tb_frequence_divider_8 模块，并选中

tb_frequence_divider_8，这时候在 design unit 编辑窗会显示 work.tb_frequence_divider_8，之后点击 OK 启动仿真如图 17

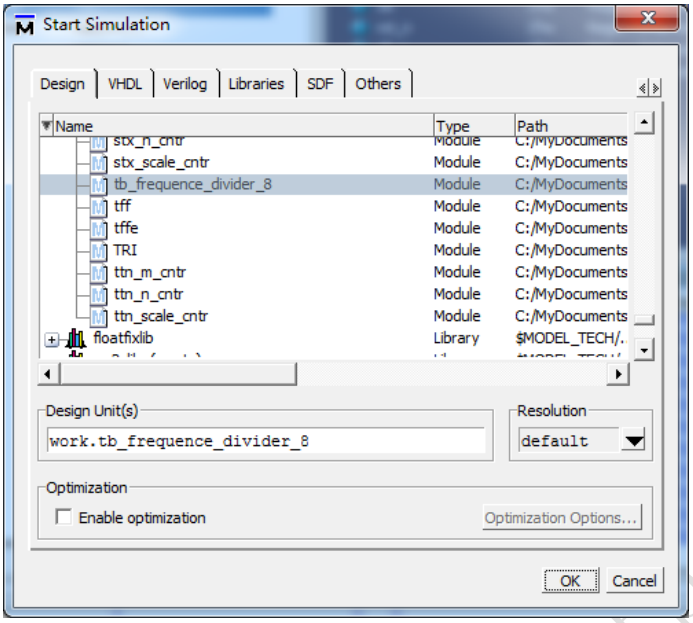

图 17 启动仿真界面

7. 后仿真波形输出

在主窗口中打开 sim 选项卡,选中 tb_frequence_divider_8 并把所有波形加入到 wave 窗口，右键点击这个模块的名称并选中 Add to->wave->all items in region

此时可以在波形窗口会有如下波形如图 18，会发现模块中的变量名字已经变了，这就是综合后的名字，有些名字已经被优化了，或者消失了。

把所有想看的信号加入到 wave 窗口后，可以再 Transcript 窗口输入如下命令，run 500ns 此时观察 wave 窗口是不是已经有波形出现了。

从波形图中可以发现 po_clk_c 与 pi_clk 上升沿有了 6.426ns(两个黄色时标之间的延迟)的延迟，这个延迟在功能仿真时是 0。Po_clk_r 与 pi_clk 的上升沿有 6.559ns 的延迟(红色时标之间的延迟)，这个延迟在功能仿真是 0。但是在后仿真时就把所有实际器件特性和走线延迟特性加入到仿真延时文件中。

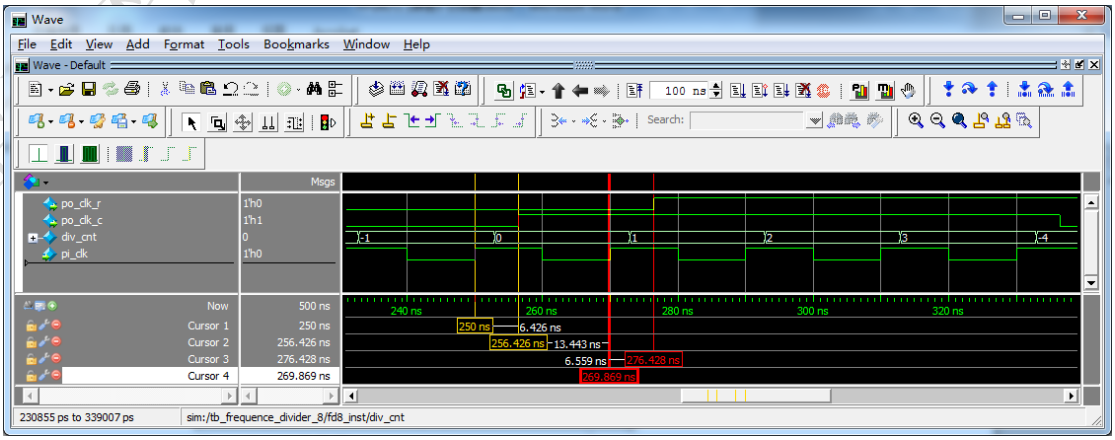

图 18 最终后仿真波形

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.fpgaw.com