

EDA 开发综合实例 2：Libero SoC 完整设计流程

如果需要将设计烧录到 FPGA 中，则需要综合使用多个不同的 EDA 工具。本综合实例以开发 2-4 译码器为例子，使用 Libero 集成开发环境（本书配套实验环境为 Libero SoC），让读者了解简单的 Verilog HDL 编程和 FPGA 设计的完整流程。

5.1.1 真值表

2-4 译码器真值表如表 5-1 所示。理解时应注意 y 作了反相处理。

表 5-1 2-4 译码器真值表

输入			输出			
en	a	b	y[0]	y[1]	y[2]	y[3]
0	x	x	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

5.1.2 逻辑表达式

根据真值表可得出以下逻辑表达式（为避免混淆，表 5-1 中的 en，在逻辑表达式中记为“e_n”）：

$$y[0] = \overline{a} \overline{b} e_n$$

$$y[1] = \overline{a} b e_n$$

$$y[2] = a \overline{b} e_n$$

$$y[3] = a b e_n$$

5.1.3 用 Verilog 描述 2-4 译码器

以下用三种不同风格的 Verilog 语句来描述 2-4 译码器，在本例中可任选其中一种来实现。

1. 行为风格的描述

```
module decoder2x4(a,b,en,y);
  input a,b,en;
  output reg [3:0] y;
  reg af,bf;

  always
    @(a or b or en)           // 当 a、b、en 发生变化时执行后面的代码
  begin
```

```
af=~a;           // a 取反后赋值给 af
bf=~b;
    // 非阻塞赋值，以下 4 句同时执行。
y[0]<=~(af&bf&en);    // 根据逻辑表达式写出
y[1]<=~(af&b&en);
y[2]<=~(a&bf&en);
y[3]<=~(a&b&en);
end
endmodule
```

2. 数据流风格的描述

```
module decoder2x4(a,b,en,y);
    input a,b,en;
    output [3:0] y;
    wire af,bf;
    // assign 连续赋值，以下各语句并发执行
    assign af=~a;
    assign bf=~b;
    assign y[0]=~(af&bf&en);
    assign y[1]=~(af&b&en);
    assign y[2]=~(a&bf&en);
    assign y[3]=~(a&b&en);
endmodule
```

3. 门级风格的描述

```
module decoder2x4(a,b,en,y);
    input a,b,en;
    output [3:0] y;
    wire af,bf;

    not
        u0not(af,a),
        u1not(bf,b);

    nand
        u0nand(y[0],en,af,bf),
        u1nand(y[1],en,af,b),
        u2nand(y[2],en,a,bf),
        u3nand(y[3],en,a,b);
endmodule
```

行为风格和数据流风格均可根据逻辑表达式写出,不需自行画出逻辑结构图;门级风格建模直接描述门结构,故应先画出逻辑图。

5.1.4 编写测试平台

测试平台编写如下:

<pre> `timescale 1ns/1ns module testdecoder2x4; reg pa,pb,pen; wire [3:0] py; decoder2x4 u1(pa,pb,pen,py); initial begin pa=0;pb=0;pen=0; #5 pen=1; end endmodule </pre>	<pre> // 单位时间为 1 纳秒 // 调用 decoder2x4 模块，按端口顺序对应方式连接 // 赋予初值 // 5 个单位时间延迟后进行赋值 </pre>
--	--

```

#10 pa=1;
#5 pb=1;
#5 pa=0;
#10 pb=0;
end

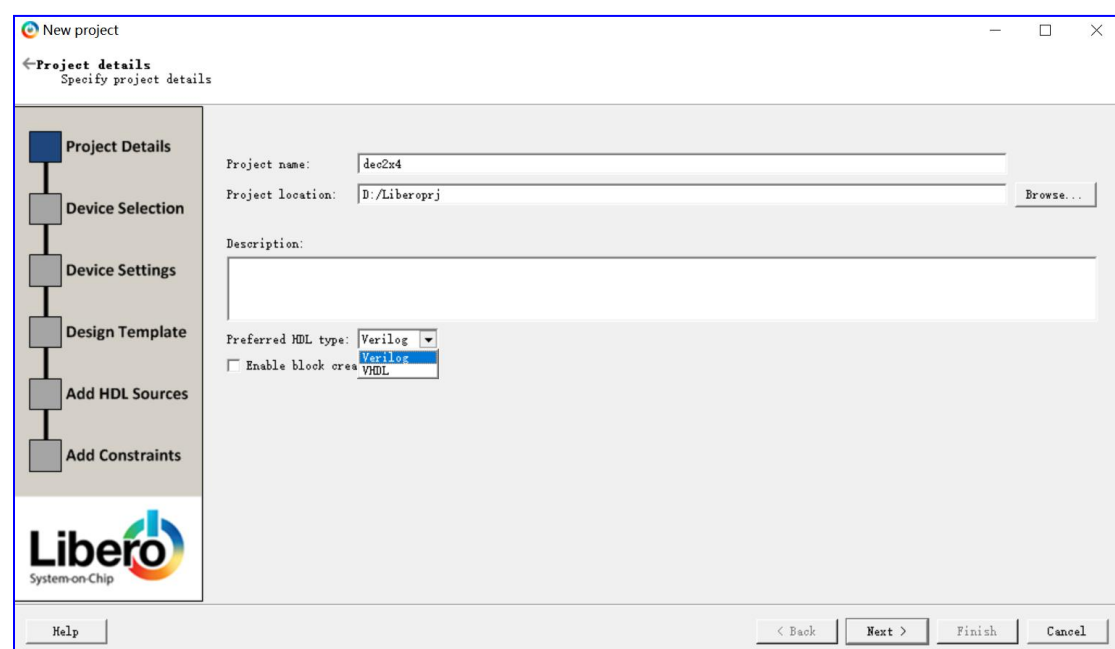
initial
$monitor("time=%t,a=%b,b=%b,en=%b,y=%b",$time,pa,pb,pen,py);
// 调用系统任务 monitor，使得 pa、pb、pen、py 当中任一发生变化时就输出显示
endmodule

```

5.1.5 FPGA 开发完整流程

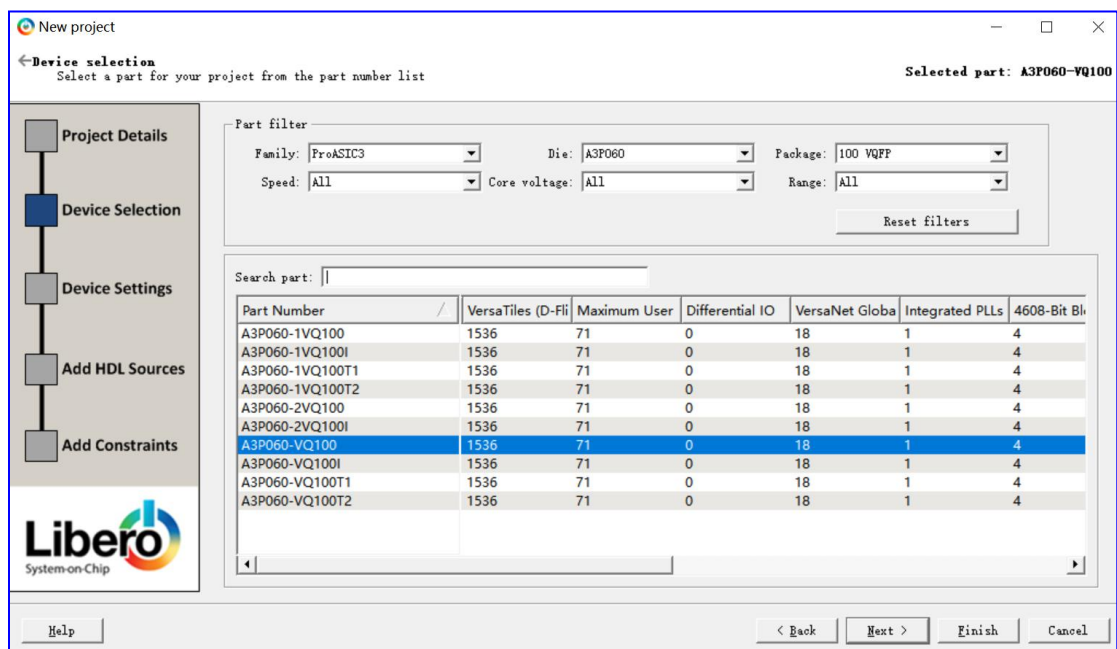
1. 新建工程

打开 Libero SoC，选择“Project”菜单的“New Project”命令，输入项目名称、选择项目存放路径，选择语言 Verilog（如图所示，默认）。每一个项目使用一个目录进行存放，项目中各操作的结果将存放于项目文件夹下不同子目录下。



新建工程

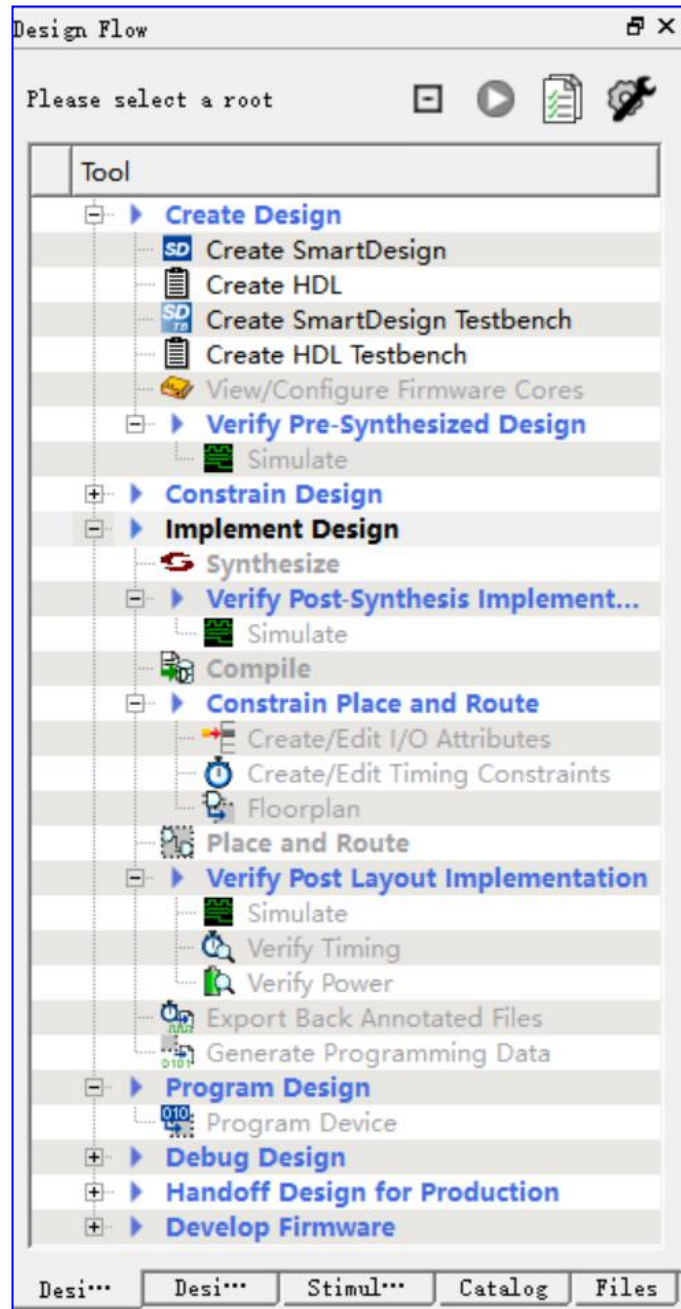
选择实际使用的设备型号、系列和封装，点击“finish”。本书配套实验环境为 ProASIC3 型号 A3P060 系列 100 引脚的 FPGA 芯片，即“A3P060-VQ100”（如图所示）。如果不需要实际烧录到 FPGA，随意选择一个也没问题。



选择设备

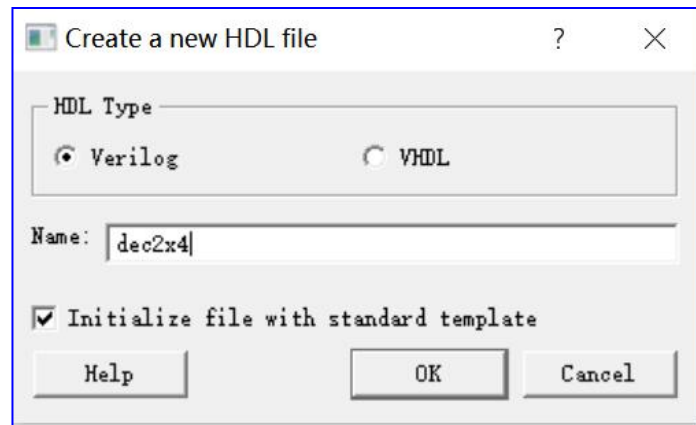
2. 输入设计代码

项目成功新建后，“Design Flow”窗口中显示了基本项目流程，可以看到许多操作，包括综合及仿真等，都是未可用状态。如图所示：



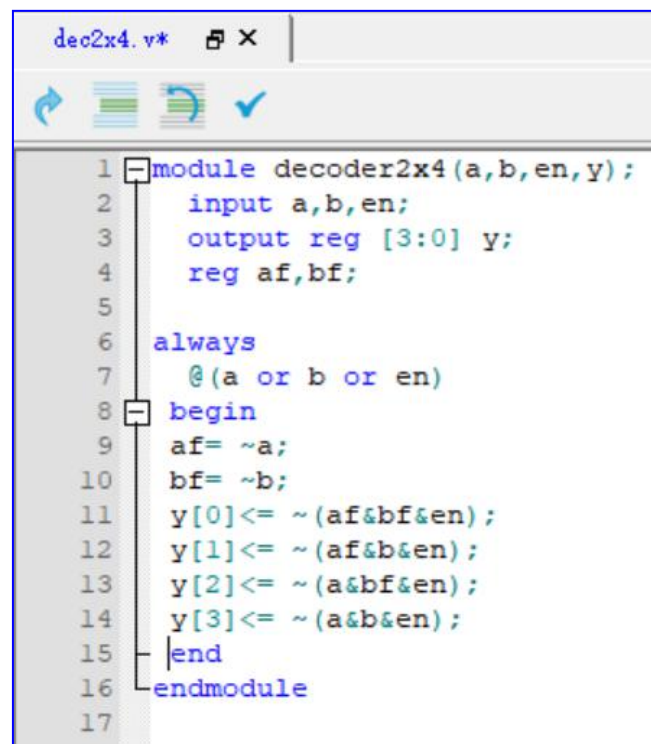
Design Flow 窗口

双击“Create HDL”，在弹出的对话框中选择“Verilog”，输入源程序文件名（扩展名为 .v）。如图所示。



添加源程序文件

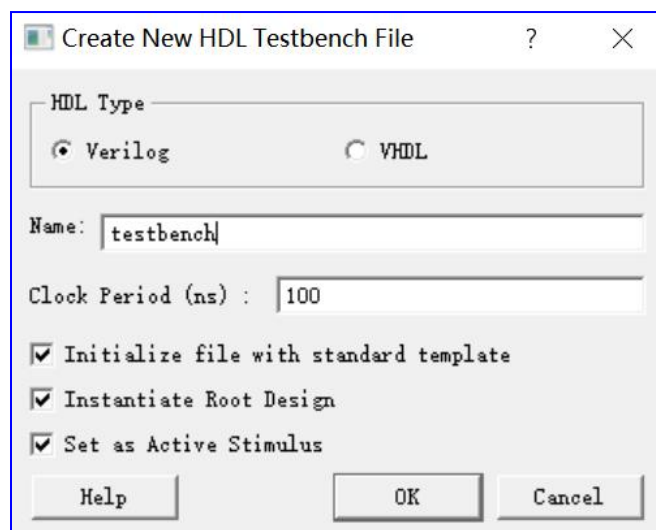
输入 2-4 译码器的功能描述代码（在此采用前述的行为风格代码），保存（如下图所示，有*表示未保存）。该文件将保存于项目文件夹的“\hdl”子目录下。



输入程序代码

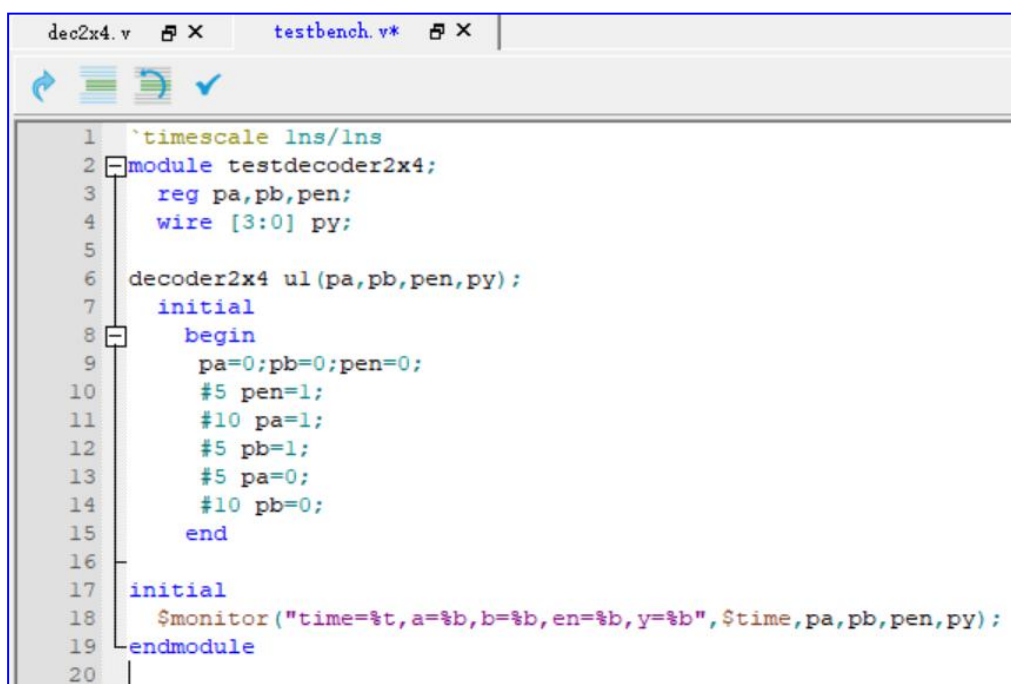
3. 输入测试平台代码

在“Design Flow”窗口双击“Create HDL Testbench”，在弹出的对话框中选择“Verilog”，输入测试平台程序文件名（扩展名为 .v）。如图所示：



新建测试平台程序文件

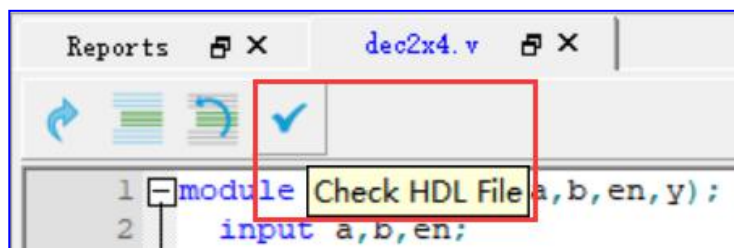
在打开的编辑器中输入测试平台程序代码，保存（如图所示），该文件将保存于项目文件夹的“\stimulus”子目录下。



输入测试平台程序代码

4. 代码检查及修改

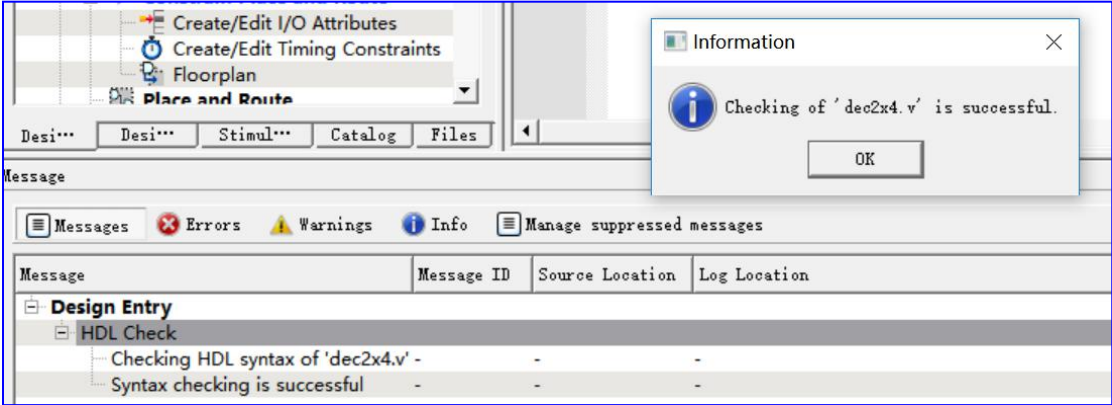
点击“Check HDL file”按钮，可检查程序是否有语法错误。如图所示。



语法检查

注意两个文件（dec2x4.v 和 testbench.v）都应该进行代码检查操作。在 Message 窗口中

可看到是否有错误信息（如图 5-26 所示），如有错误，则程序将无法通过编译，在代码编辑窗口进行修改。

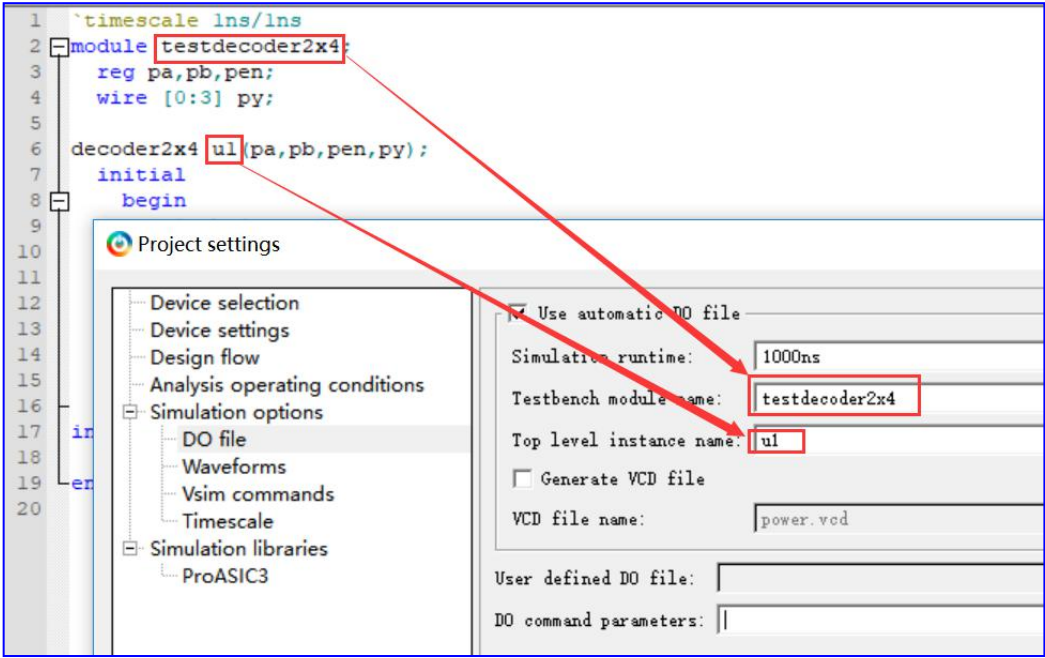


代码检查结果

以上操作可检查语法错误，但非语法错误就需要自行检查发现，如在此应检查功能模块的名称（“decoder2x4”）与测试平台中调用的名称是否相同。

5. 对应名称设置

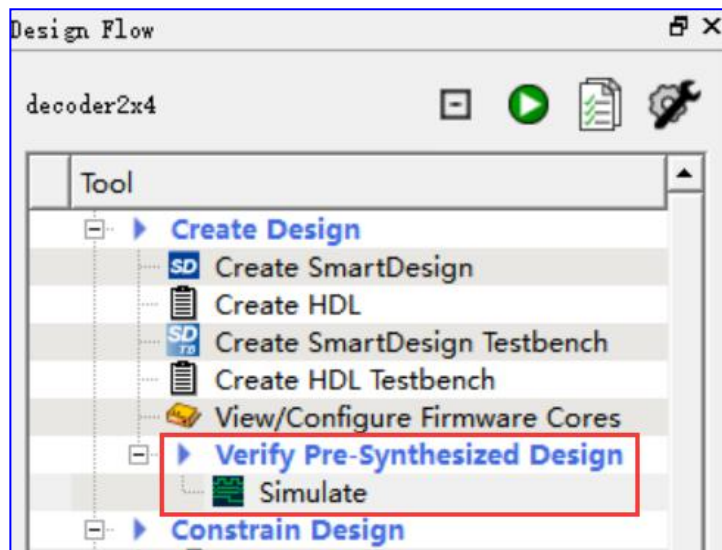
有一个重要参数应首先设置：“测试平台模块名称”和“顶层实例名”须与代码中的一致。在“Project”菜单选择“Project Settings”功能，将“Testbench module name”值修改为测试平台模块的名称，将“Top level instance name”值修改为所使用的实例名，如图所示：



测试平台模块名称与实例名须与代码一致

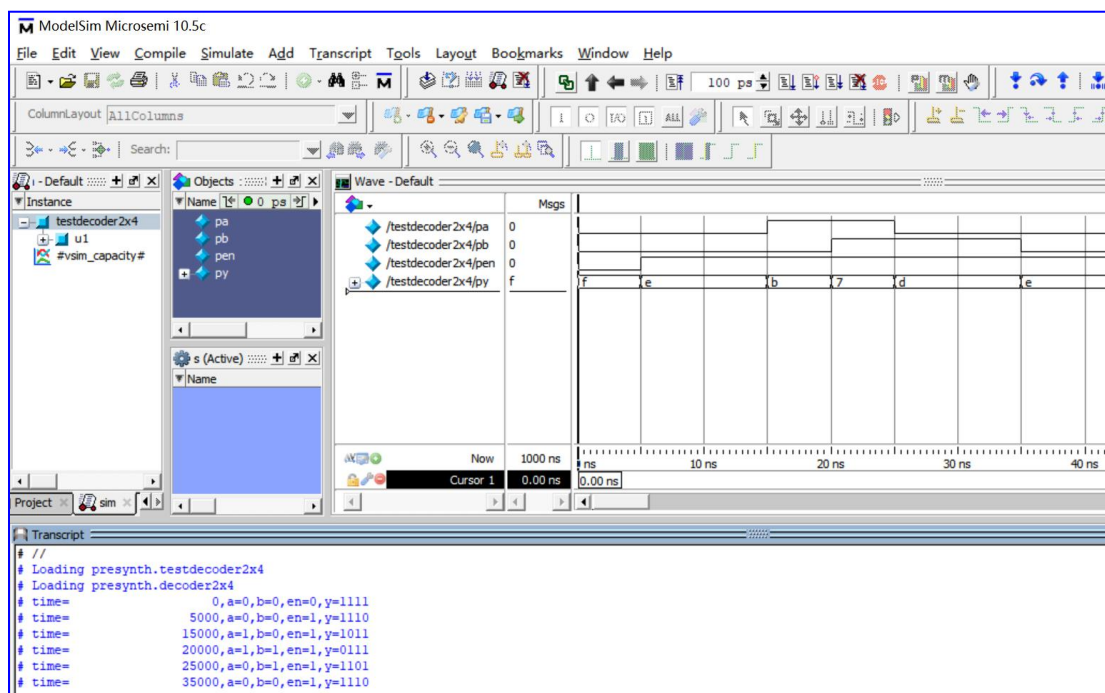
6. 仿真（综合前）

在“Design Flow”窗口中，运行“Simulate”，进行综合前的仿真。



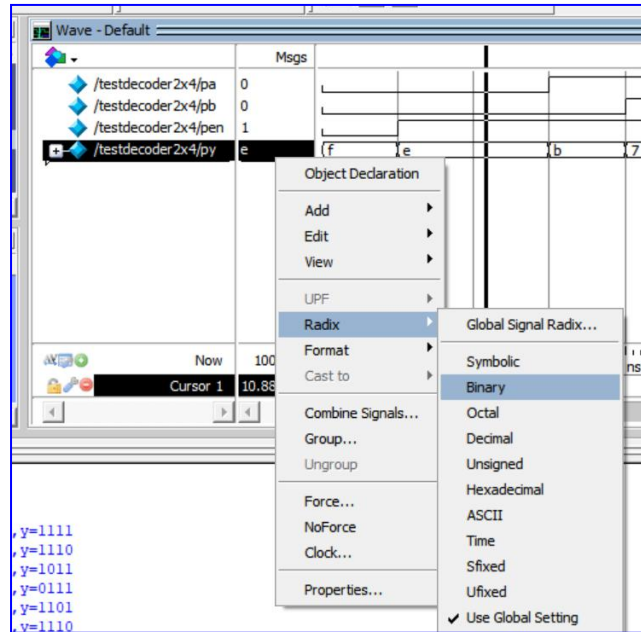
综合前仿真

ModelSim 软件被打开，正常情况下（无程序错误，没有打开多个 ModelSim 程序等），ModelSim 会自动执行刚才关联的测试平台程序，通过测试平台程序调用 2x4 译码器程序，并显示仿真结果。如图所示。



仿真结果

信号“py”的输出默认为 16 进制显示，在此并不直观，可以设置为二进制方式显示，并点击“+”号展开，操作如图所示：

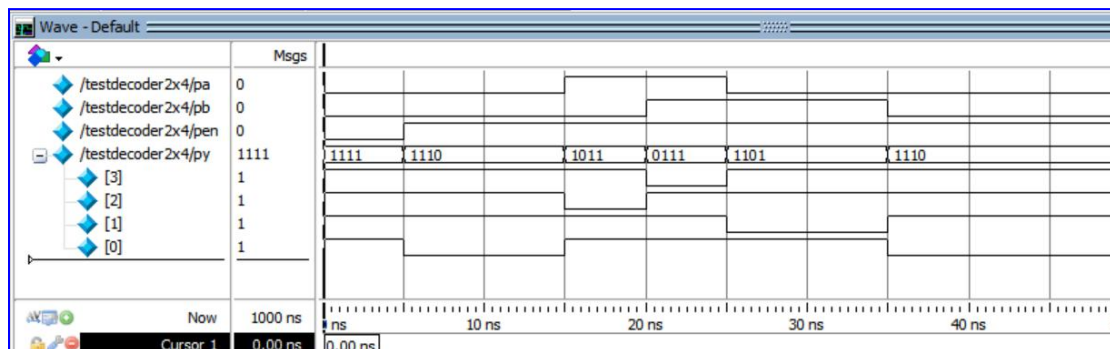


设置二进制显示

“Wave”窗口显示的波形，波形显示可能会太宽或太窄影响查看，可通过工具栏



进行缩放显示，调整效果如图所示。



波形显示

在 Transcript 小窗口显示执行情况及执行结果：


```
# Loading presynth.testdecoder2x4
# Loading presynth.decoder2x4
# time= 0,a=0,b=0,en=0,y=1111
# time= 5000,a=0,b=0,en=1,y=1110
# time= 15000,a=1,b=0,en=1,y=1011
# time= 20000,a=1,b=1,en=1,y=0111
# time= 25000,a=0,b=1,en=1,y=1101
# time= 35000,a=0,b=0,en=1,y=1110
```


验证输出结果和波形是否正确，验证后关闭 ModelSim。

其它说明：

- (1) 测试平台程序（testbench.v）中的延迟设置为 1 纳秒（ns），而当前显示的单位时间为 1 皮秒（1ns=1000ps），所以在结果中显示的时间为“0、5000、

15000……35000”，而不是测试平台程序中的“0、5……”。

- (2) 某些情况下进入 ModelSim 后未必能正常显示仿真结果，如程序错误、参数设置错误等，可以点击工具栏  图标执行“Restart”操作，清除之前的显示

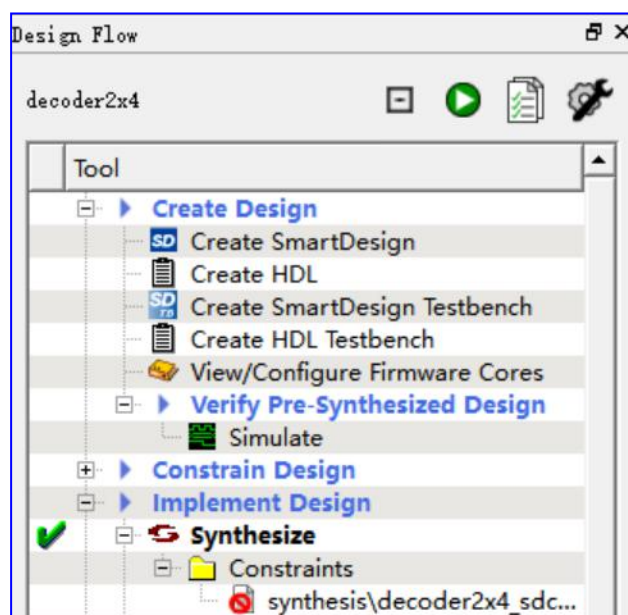
(不清除的话可能影响后面的显示)，然后点击  重新执行 (Run -All)。

- (3) 如运行不正常又觉得代码没问题，可尝试对程序进行重编译。代码如有变动，也需对程序进行重编译。

7. 综合

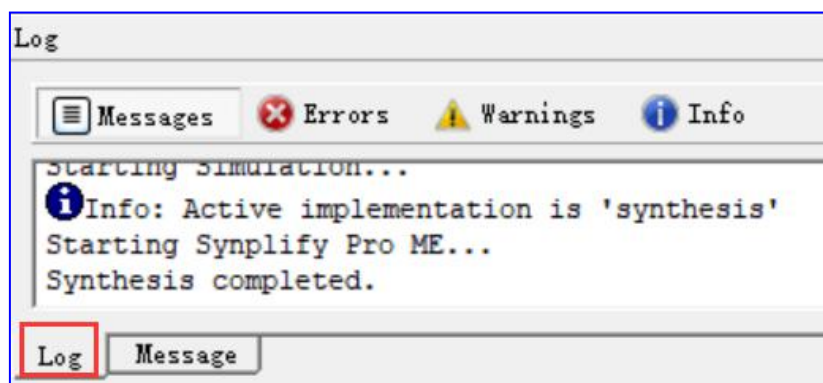
- (1) 直接综合

在“Design Flow”中，双击“Synthesize”（如图所示），进行“综合”操作。



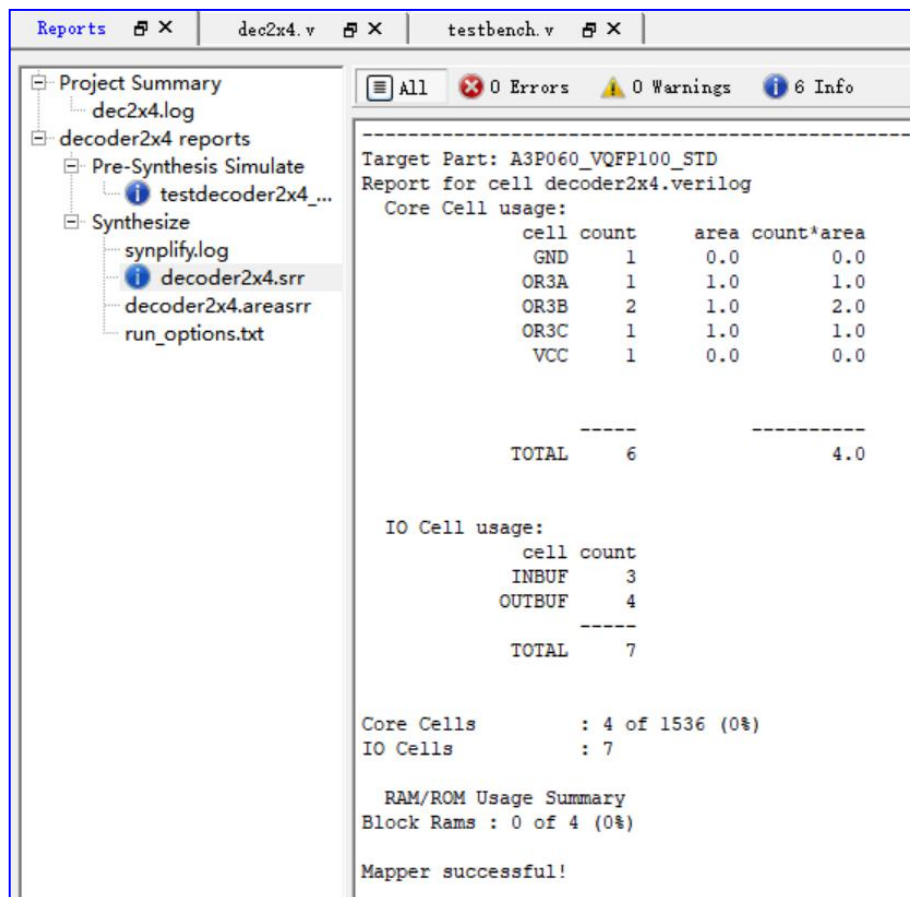
“综合”操作

综合操作需要一些时间，完成后在 Log 窗口中可查看完成结果，如图所示。



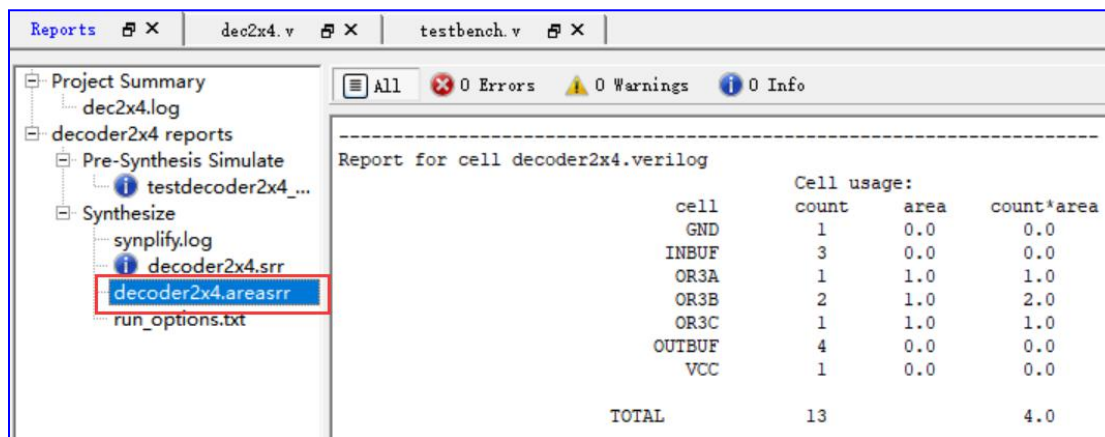
Log 日志

在“Reports”窗口中显示了该操作的所有日志信息（包含之前的综合前仿真等）。如图所示。



综合结果

可点击“decoder2x4.areasrr”，查看资源使用情况统计。如图所示。

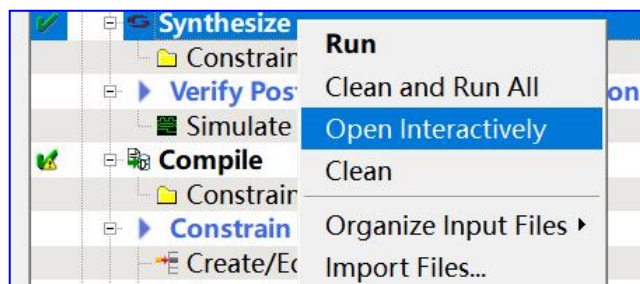


资源使用情况

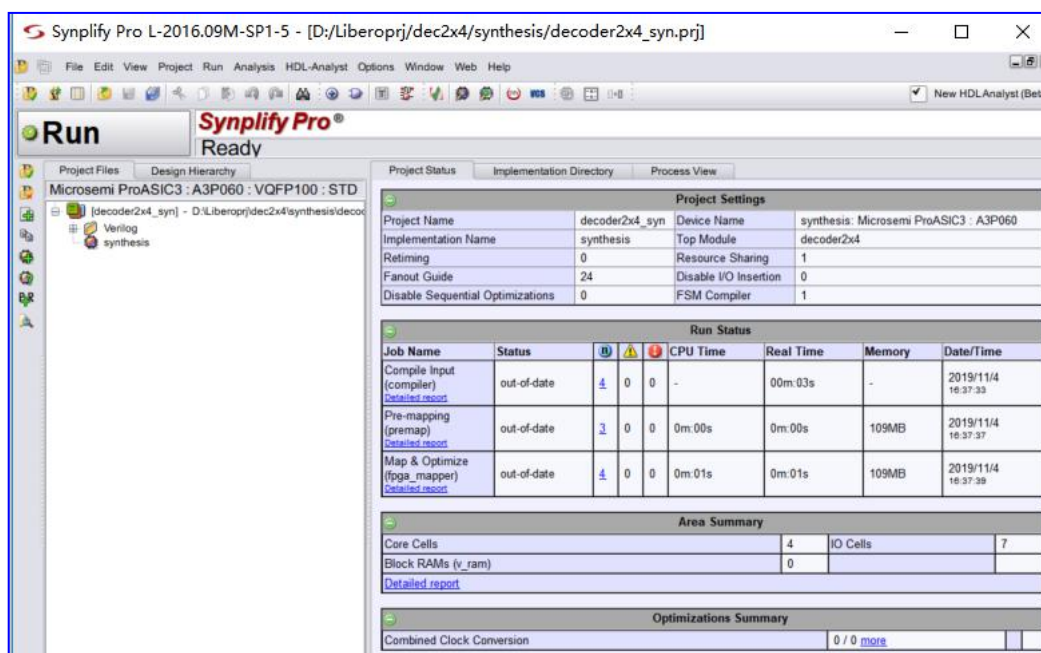
(2) 打开 Synplify Pro ME 软件进行综合操作

双击“Synthesize”后，会后台调用 Synplify Pro ME 进行编译、制图等操作，但这些后台操作用户看不到，只看软件报告也不太清晰。可以显式地打开 Synplify Pro ME 软件，通过可视化界面操作和看到综合的结果，会更为直观。

对着“Synthesize”点击右键，选择“Open Interactively”，如下图所示。

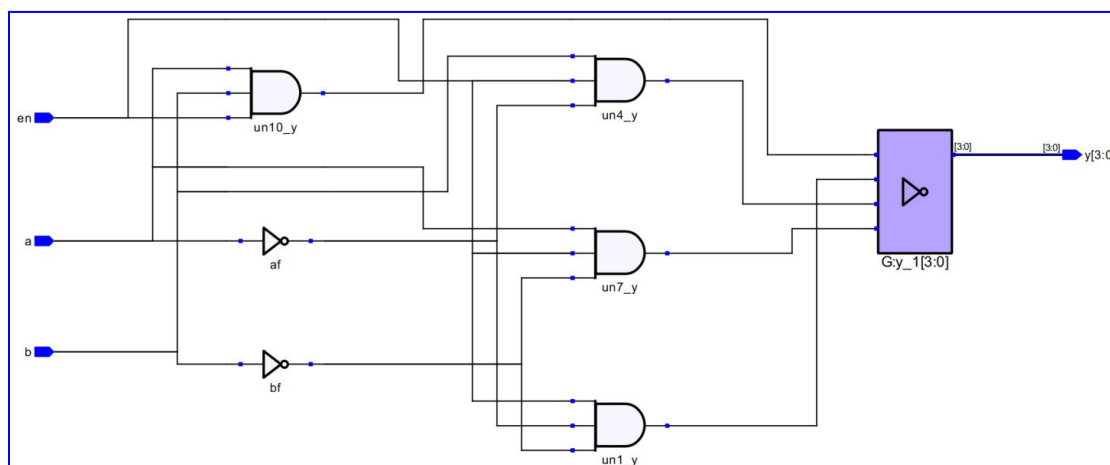


打开业界有名的综合软件 Synplify Pro，界面如下图所示。




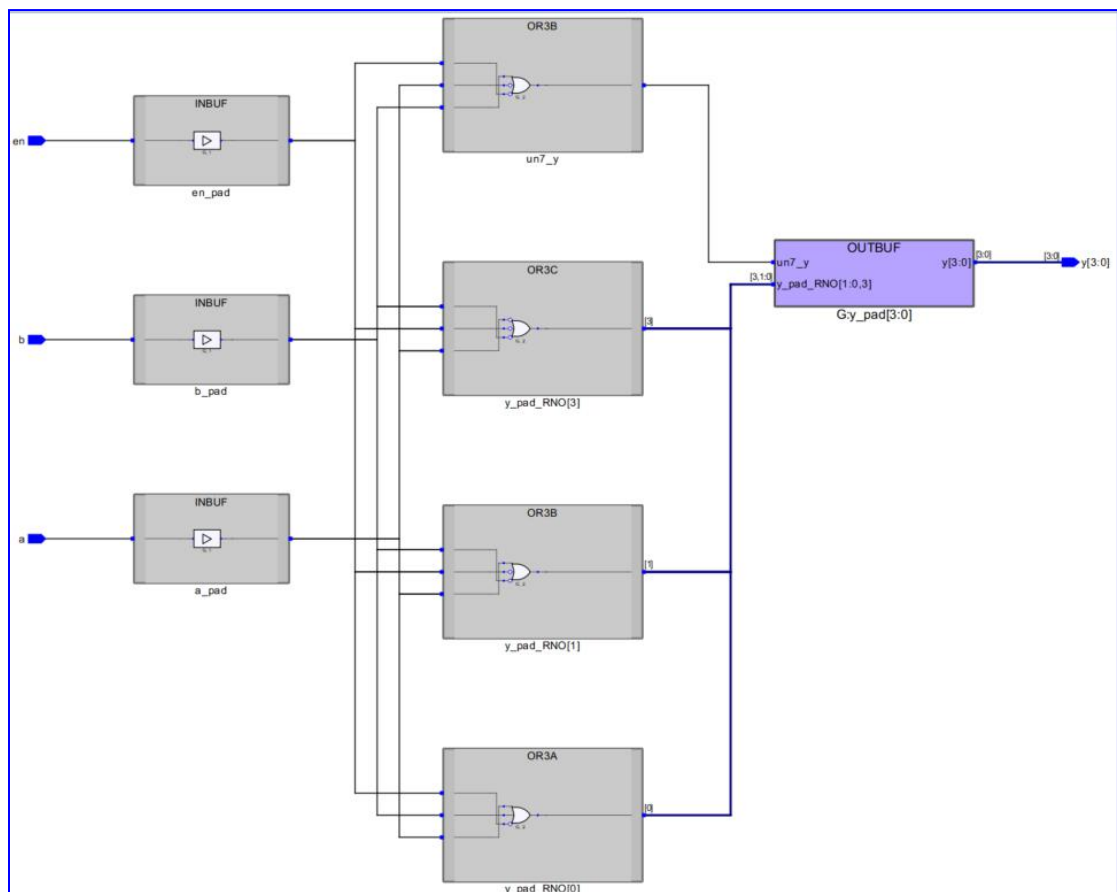
在此点击“Run”按钮，可实现综合操作，其功能与之前直接综合是一致的。

点击工具栏  按钮，可查看 RTL 视图，该图由 Synplify Pro 自动生成。如图所示。



RTL 视图

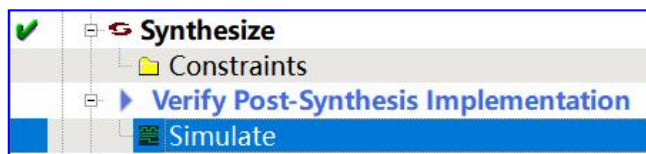
点击工具栏  按钮，可查看 Technonogy View（工艺视图），这是针对 Microsemi FPGA 芯片实现的工艺电路图。如图所示。



Technology View

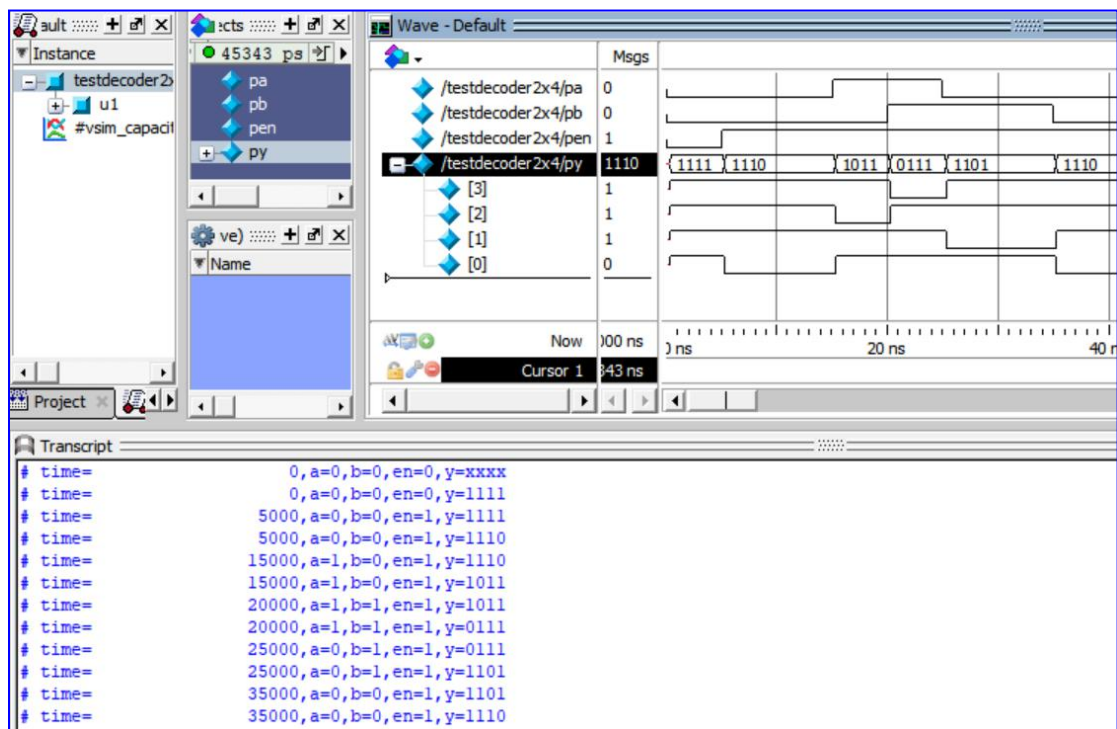
8. 仿真（综合后）

综合后，命令的提示图标会发生变化（如下图所示），此时可再一次进行仿真，验证综合后是否也能得到正确的结果。点击“Verify Post-Synthesis Implementation”下的“Simulate”，如图所示：



综合后仿真

正常情况下，系统自动进行仿真，并显示仿真结果。如图所示。



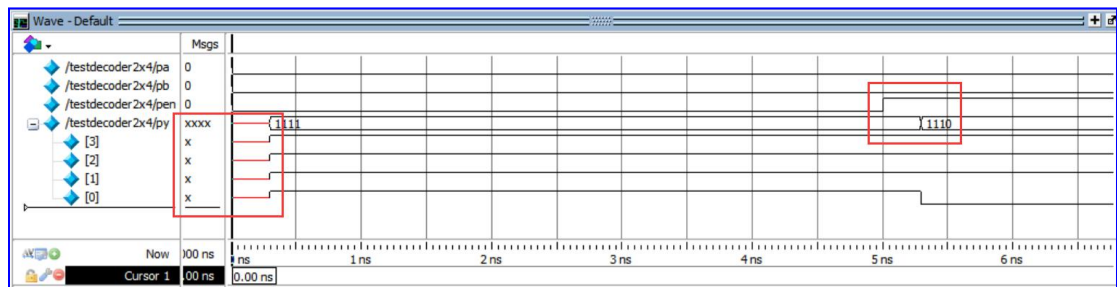
仿真结果

从逻辑上描述各种设计时，很多时候是假设程序运行是零延迟的，但在实际电路中，电流通过任何一个器件和线路都是有延迟的。可看到“Transcript”窗口中得到如下输出，发现比综合前仿真多了一些输出结果，这是加上器件延迟等因素后输出的结果。

```
# time= 0,a=0,b=0,en=0,y=xxxx
# time= 0,a=0,b=0,en=0,y=1111
# time= 5000,a=0,b=0,en=1,y=1111
# time= 5000,a=0,b=0,en=1,y=1110
# time= 15000,a=1,b=0,en=1,y=1110
# time= 15000,a=1,b=0,en=1,y=1011
# time= 20000,a=1,b=1,en=1,y=1011
# time= 20000,a=1,b=1,en=1,y=0111
# time= 25000,a=0,b=1,en=1,y=0111
# time= 25000,a=0,b=1,en=1,y=1101
# time= 35000,a=0,b=0,en=1,y=1101
# time= 35000,a=0,b=0,en=1,y=1110
```

读者在实际的验证过程中，运行结果不一定跟上述的完全一致，有少量差异是正常的，因为延迟的情况，跟实际操作时选择的器件、综合结果有关。

在“Wave”窗口可看到如图 5-41 所示的波形，波形达到预期运行结果。



延迟效果

说明：

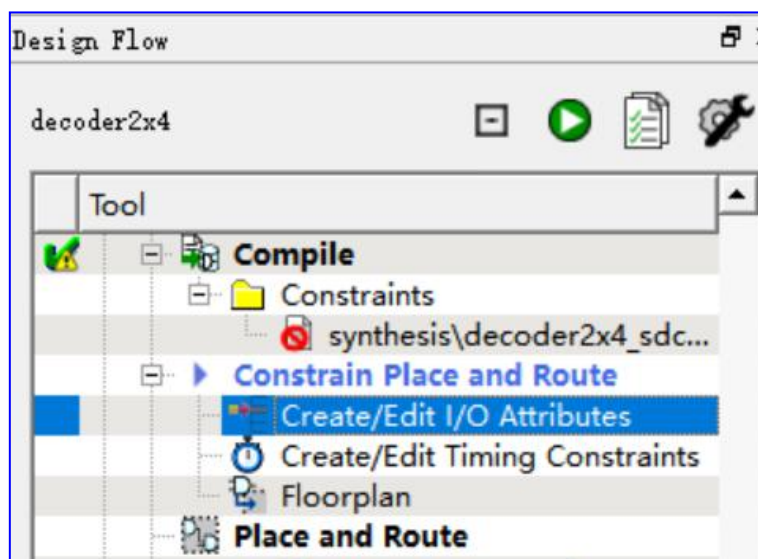
(1) 注意这里说法是“波形达到预期结果”，而非“与前面的仿真结果一致”：对波形进行局部放大仔细查看（如图 5-42 所示），可看到综合后仿真与综合前仿真稍有不同，多出了 0.3ns（300ps）的输出延迟，py 信号在运行开始到有效输出期间，会出现红色的 x 态（如图所示），这是由于综合后考虑了元器件本身工作的延迟（电路越复杂，延迟越明显）。

(2) 如果把测试平台程序中的单位时间设置语句“`timescale 1ns/1ns”去掉，在综合前仿真中也能得到同样的运行效果，但到了综合后仿真中，会发现得不到正确的波形，其结果及其原因请读者自行验证和思考。

9. 布局布线约束

(1) 打开软件

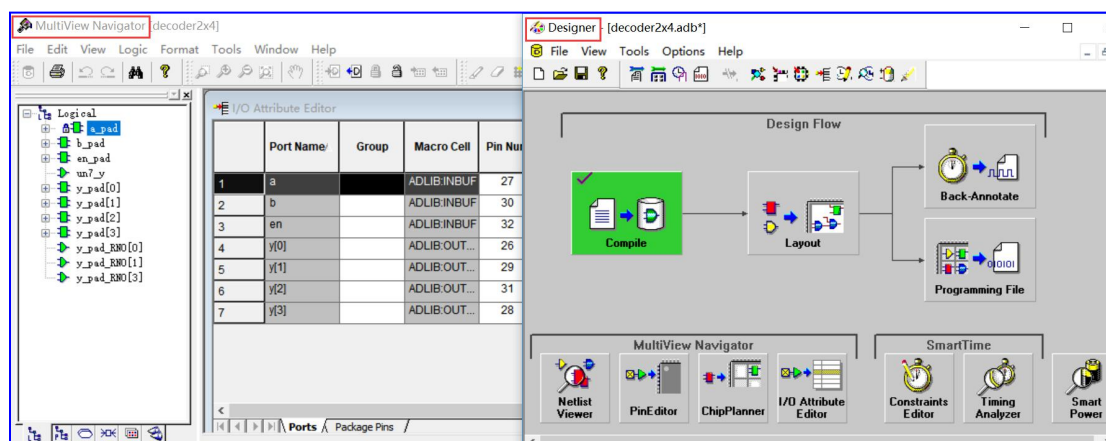
在“Design Flow”中，点击“Constrain Place and Route”下的“Create/Edit I/O Attributes”（如图所示）。



选择“Create/Edit I/O Attributes”

自动调用软件“Designer”和“MultiView Navigator”，同时打开两个软件。“Designer”中可调用“MultiView Navigator”修改引脚设置，“MultiView Navigator”中修改的参数会同步到“Designer”中影响布局布线的操作。

“Compile”操作在软件打开的时候默认进行，所以看到该按钮为绿色（已完成）。

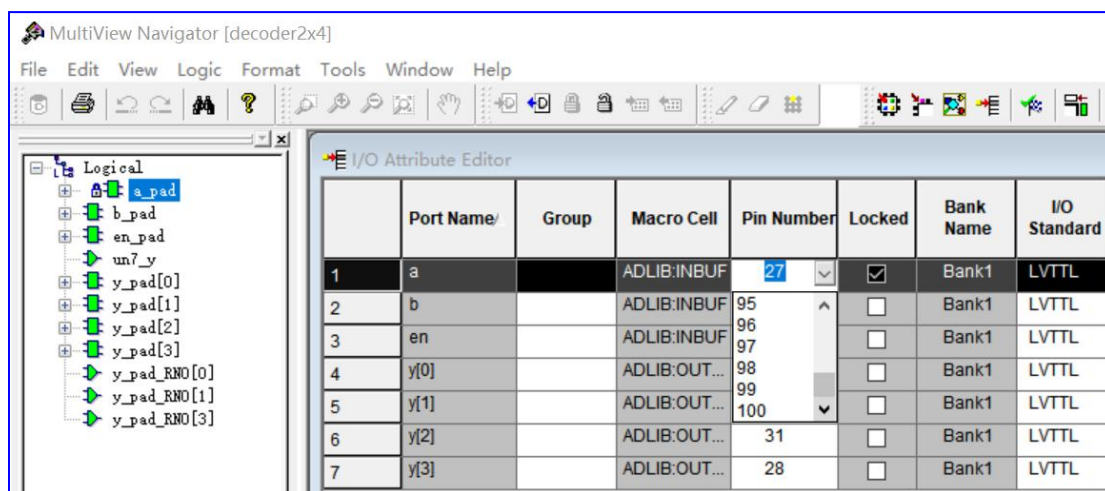


同时打开两个软件


（2）引脚设定

引脚设定并非必须进行的操作步骤，Designer 软件会自动分配好引脚和端口的对应关系。

一块 FPGA 芯片上有多个引脚（本例使用的“A3P060”芯片有 100 个引脚），而本例设计的 2-4 译码器只需要 7 个引脚（a, b, en, y[0], y[1], y[2], y[3]），可以在“Pin Number”列修改端口所使用的引脚（如图所示）。

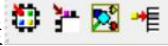


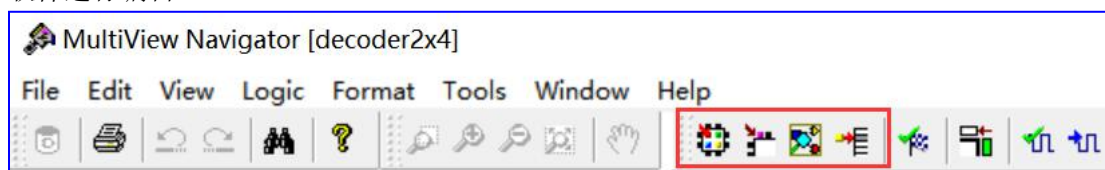
修改引脚配置

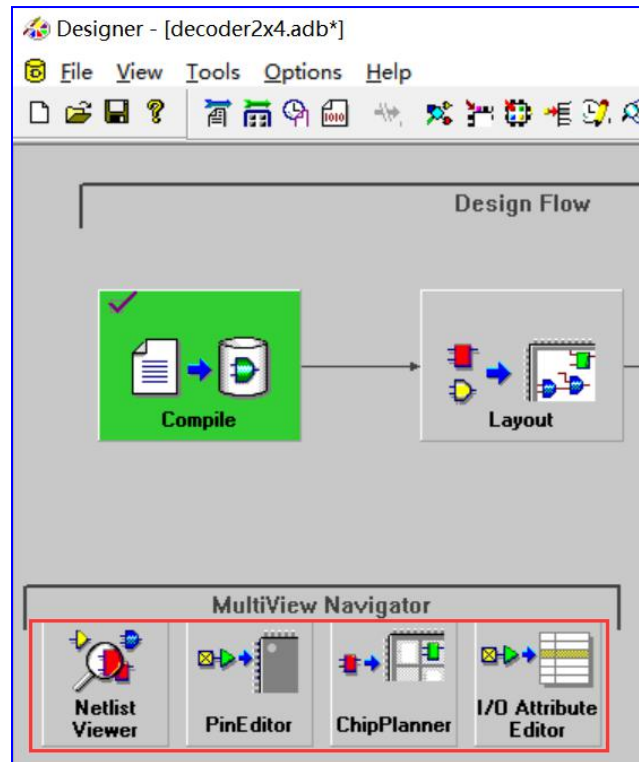
如果设计过程中修改了引脚配置，可点击  按钮，或选择“File”菜单的“Commit and Check”按钮，来检查引脚分配是否有问题。如果检查没有错误，则引脚重新分配成功。此后，需要通过点击“Layout”按钮重新进行布局布线操作。

注：如真正考虑写入 FPGA 芯片，应找到对应硬件的引脚使用说明，并按相应指引分配引脚，否则烧录后可能得不到正确的运行结果。

（3）布局规划

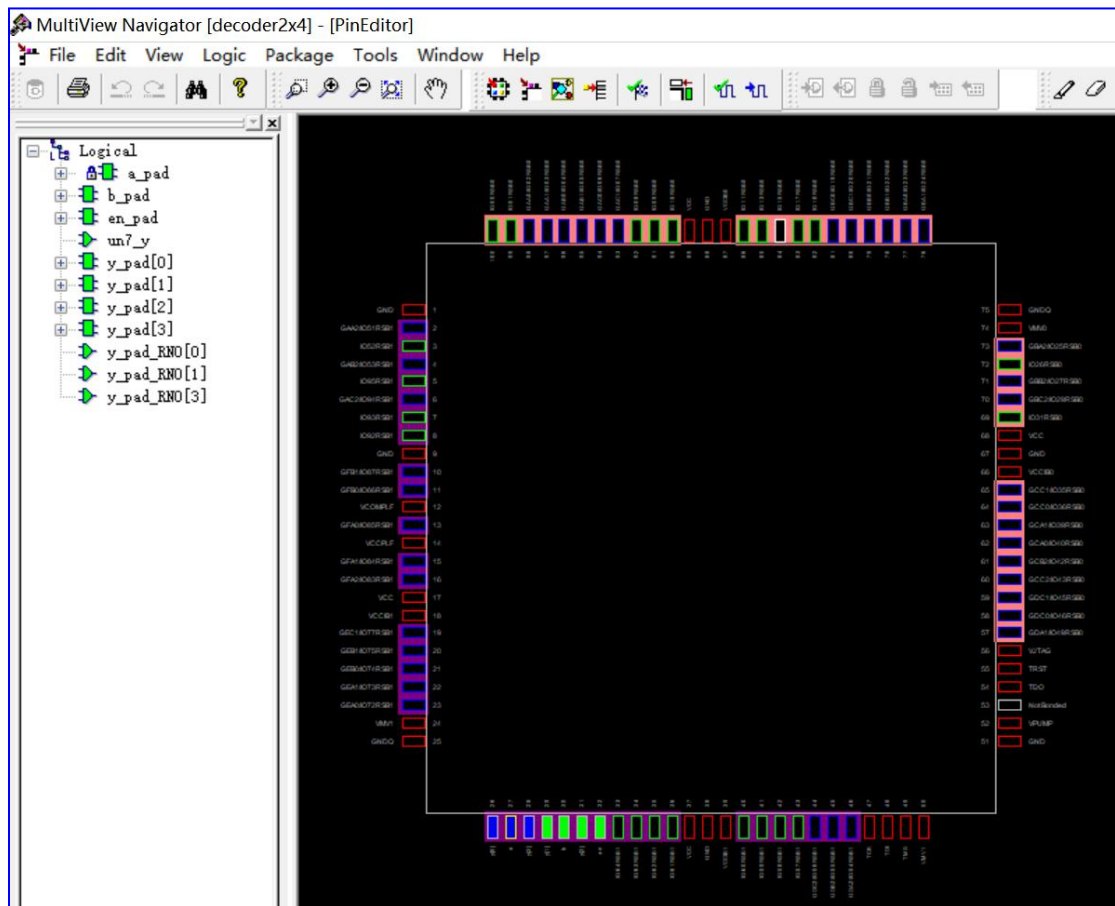
如果想手工设定引脚配置，“MultiView Navigator”工具栏  中选择对应功能，也可在 Designer 软件点击“I/O Attribute Editor”按钮（如图所示）调用“MultiView Navigator”软件进行编辑。



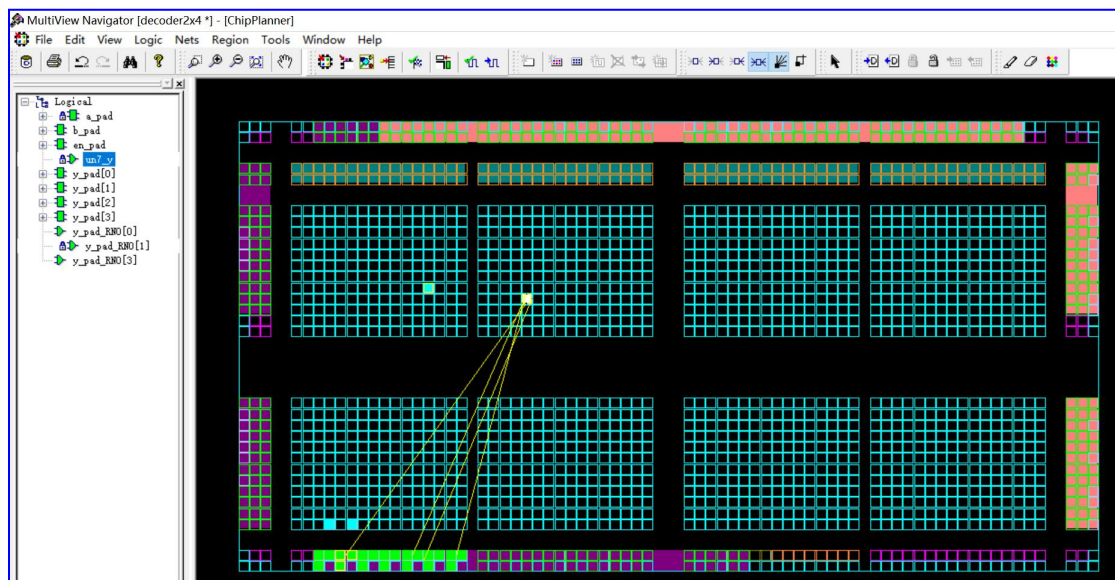


“MultiView Navigator”

可使用“PinEditor”和“ChipPlanner”功能，通过可视化方式配置端口、引脚和元胞分配的对应关系，在这些功能下，更加直观的呈现引脚分配、元胞分配，更为清晰的看到硬件细节。

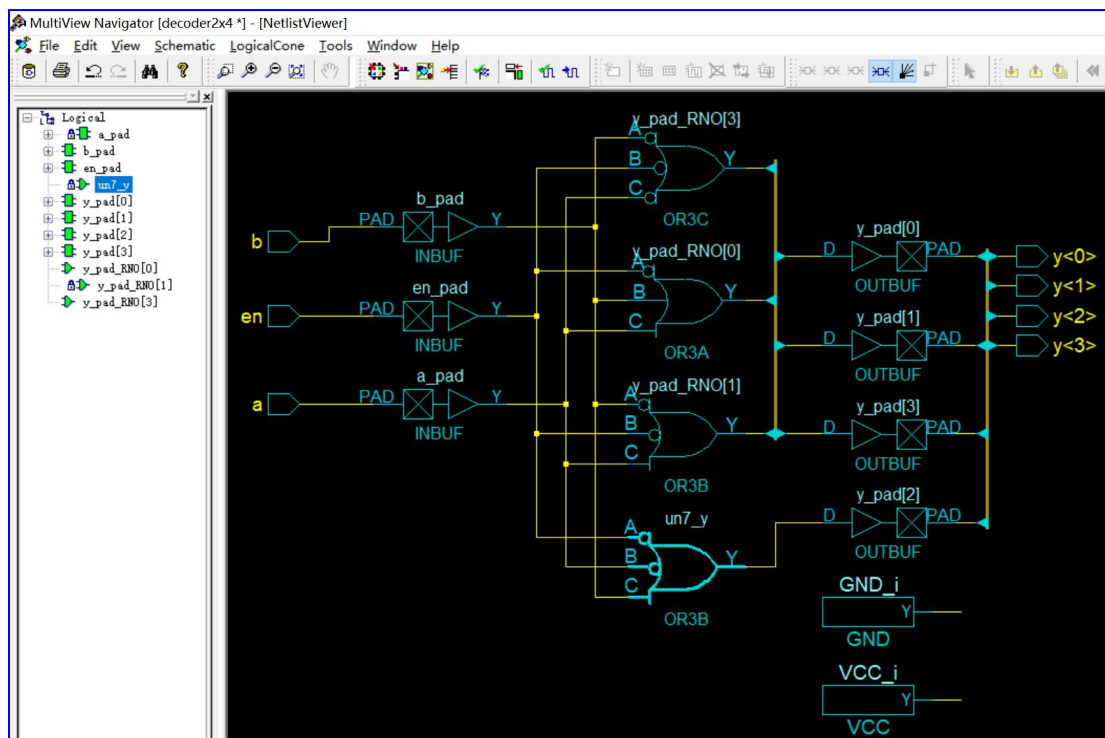


“PinEditor”



“ChipPlanner”

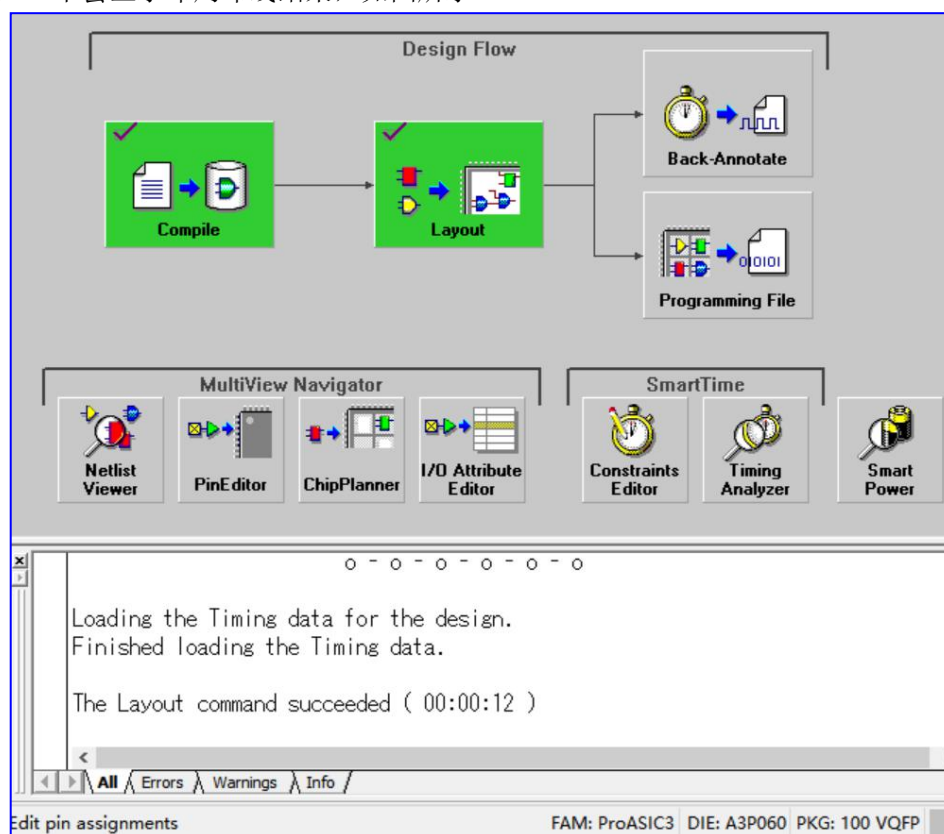
可通过 Netlist Viewer 看到网表视图。



“网表视图”

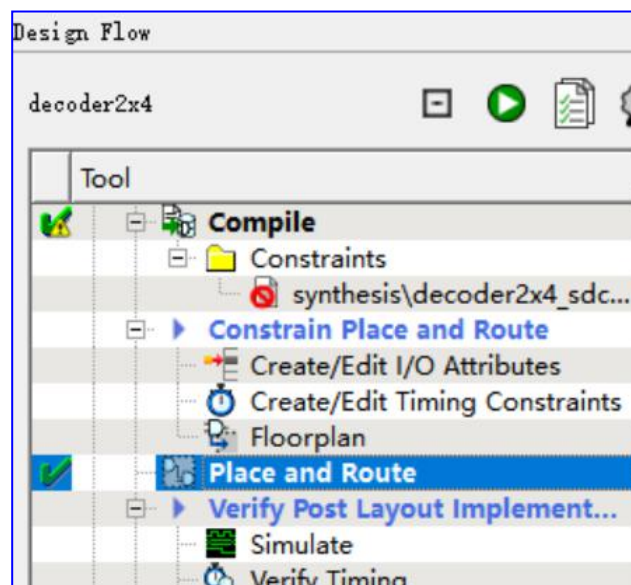
10. 布局布线 (Layout)

在 Designer 软件里，点击“Layout”按钮，布局布线成功后，按钮会变为绿色。“Log Window”中会显示布局布线结果，如图所示。



布局布线操作

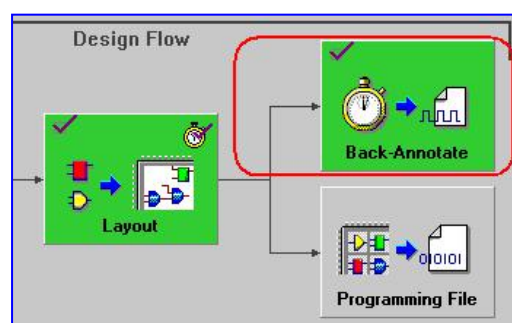
也可以在 Libero 软件中直接双击“Place and Route”，布局布线操作将自动完成而不打开 Designer 软件。



“Place and Route”

11. 反标 (Back-Annotate)

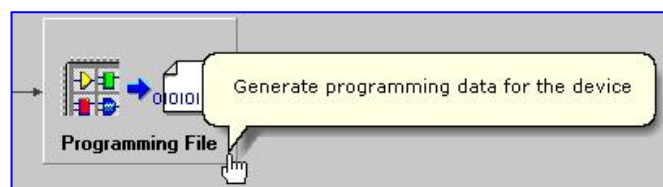
在 Designer 中点击“Back-Annotate”按钮进行“反标”操作，该操作将生成延时参数文件。在弹出的窗口中选择默认配置，完成后按钮变成绿色（如图所示），“Log Window”中会显示结果“The Back-Annotate command succeeded”。



反标

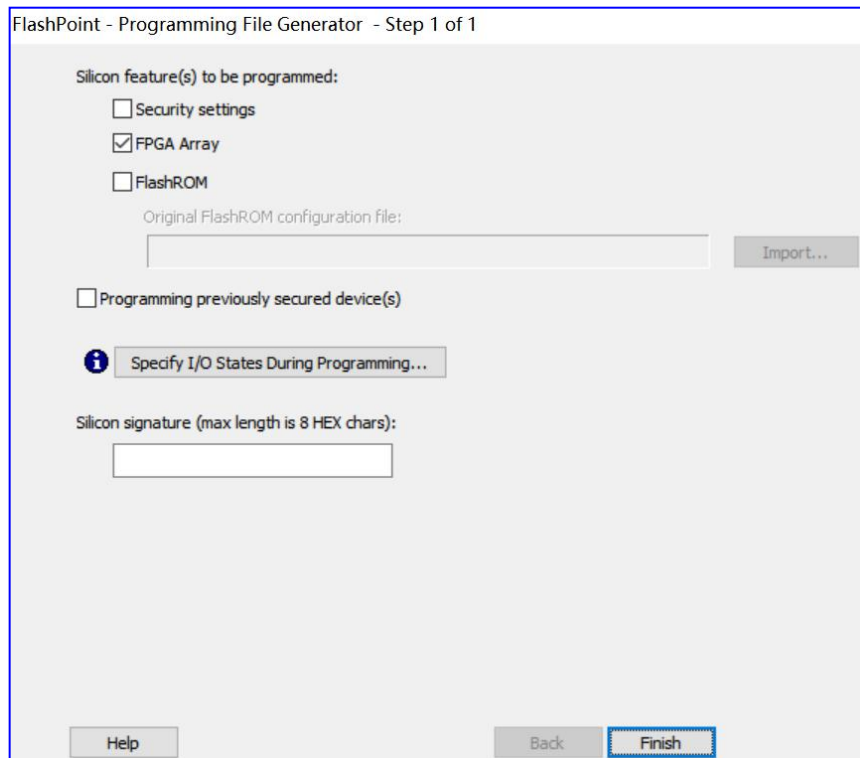
12. 生成编程文件

如果需要生成编程文件并烧录到 FPGA 里，则可进行以下操作，如果只是仿真，则可不进行该操作。在 Designer 中点击“Programming File”按钮，如图所示：



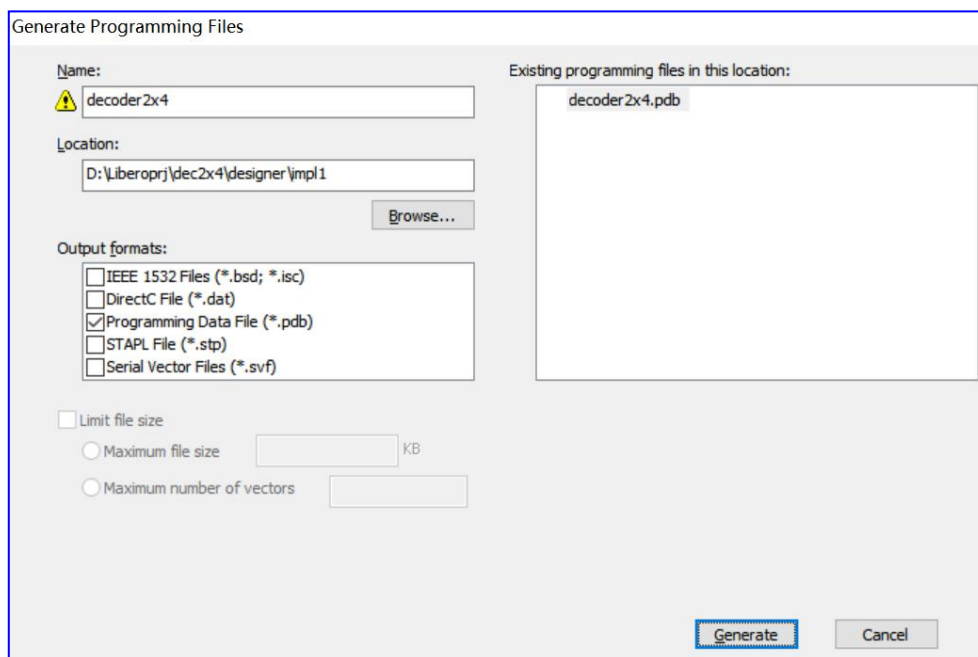
“Programming File”按钮

弹出如图所示的对话框，在此直接点击“finish”按钮：



生成编程文件

在弹出的对话框中选择输出格式，点击“Generate”按钮，如图所示。

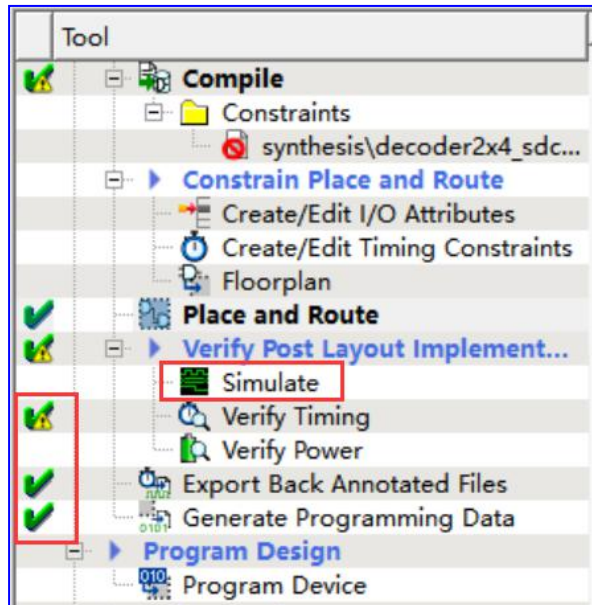


选择输出格式

程序文件生成成功，则可在项目文件夹的“\designer\impl1”子文件夹下找到刚生成的“decoder2x4.pdb”文件。

13. 仿真（布局布线后）

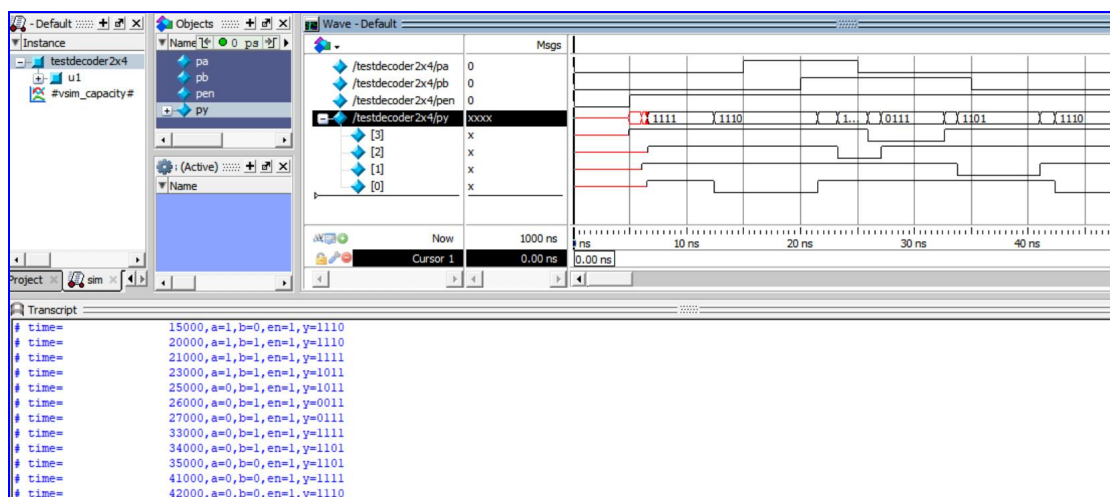
在 Designer 软件中进行保存操作，回到 Libero 软件，在“Design Flow”中，可看到完成布局布线后，各流程的提示信息有更新，如图所示：



多项操作已完成

选择“Simulate”进行布局布线后仿真。

此时将再一次进入“ModelSim”软件，仿真结果如图所示。



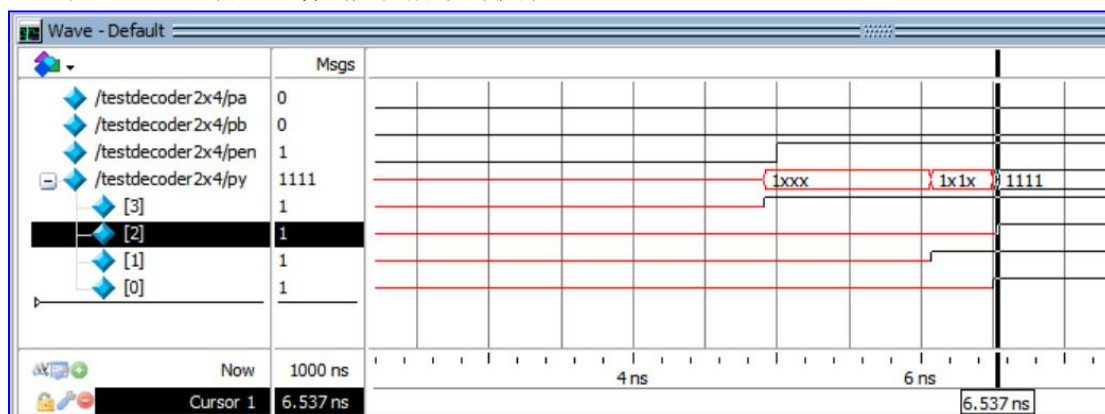
仿真结果

在“Transcript”窗口中得出如下输出，可发现该结果与综合后仿真的输出结果又有了不同，这是加上线路延迟等具体因素后输出的结果。

```
# time= 0,a=0,b=0,en=0,y=xxxx
# time= 5000,a=0,b=0,en=0,y=1xxx
# time= 5000,a=0,b=0,en=1,y=1xxx
# time= 6000,a=0,b=0,en=1,y=1x1x
# time= 7000,a=0,b=0,en=1,y=1x11
# time= 7000,a=0,b=0,en=1,y=1111
# time= 12000,a=0,b=0,en=1,y=1110
# time= 15000,a=1,b=0,en=1,y=1110
# time= 20000,a=1,b=1,en=1,y=1110
# time= 21000,a=1,b=1,en=1,y=1111
# time= 23000,a=1,b=1,en=1,y=1011
```

```
# time=      25000,a=0,b=1,en=1,y=1011
# time=      26000,a=0,b=1,en=1,y=0011
# time=      27000,a=0,b=1,en=1,y=0111
# time=      33000,a=0,b=1,en=1,y=1111
# time=      34000,a=0,b=1,en=1,y=1101
# time=      35000,a=0,b=0,en=1,y=1101
# time=      41000,a=0,b=0,en=1,y=1111
# time=      42000,a=0,b=0,en=1,y=1110
```

在“Wave”窗口可看到如图所示的波形。

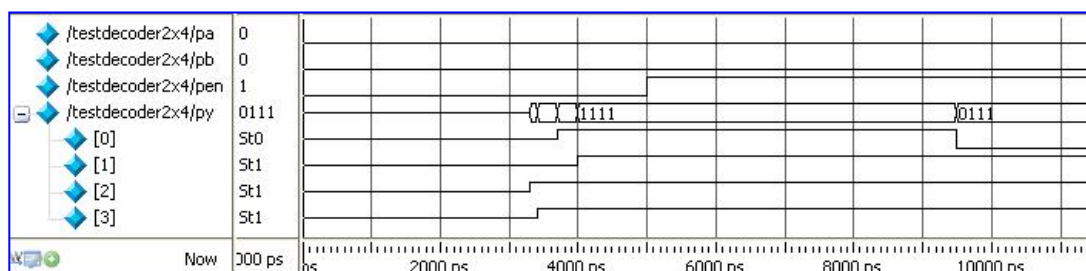


仿真结果波形

可看到波形与综合前、综合后的仿真结果差别很大，延迟非常明显（如 5000ps 时 pen 输入的变化，到了 6537ps 时输出 py 才发生改变），并且结果并不正确，伴随而来的是有毛刺的产生（竞争冒险）。

对波形进行局部放大仔细查看（如图所示），可看到 py 的各位先后发生变化（py[2]、py[3]、py[0]、py[1]），导致了 py 的值发生多次变化，形成了毛刺。

延迟和毛刺的产生和加剧，都是因为考虑了布局布线后，具体器件的延迟和线路的延迟而产生，属于正常和需要关注的现象。

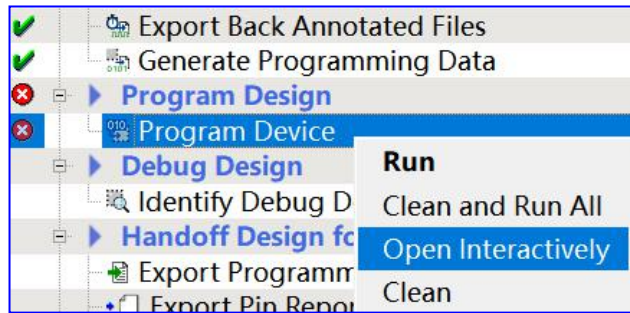


仿真延迟与毛刺

14. 烧录程序

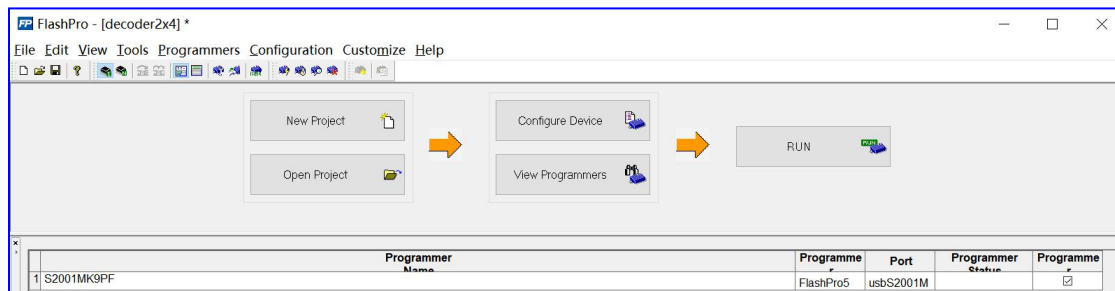
如需往 FPGA 芯片中写入程序文件，需要烧录器设备（通过 USB 接口连接计算机）和 FPGA 芯片，使用 FlashPro 软件进行烧录操作。

在“Program Design”中，右键点击“Program Device”，选择“Open Interactively”（如图所示），打开 FlashPro 软件。



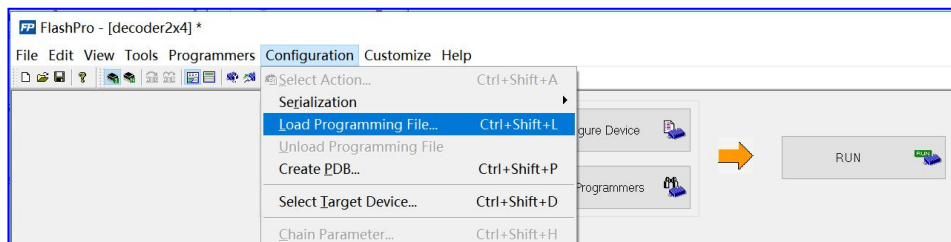
打开 FlashPro

FlashPro 软件运行界面如图所示，如果烧录设备正常连接，则会显示检测到的烧录设备及其参数。



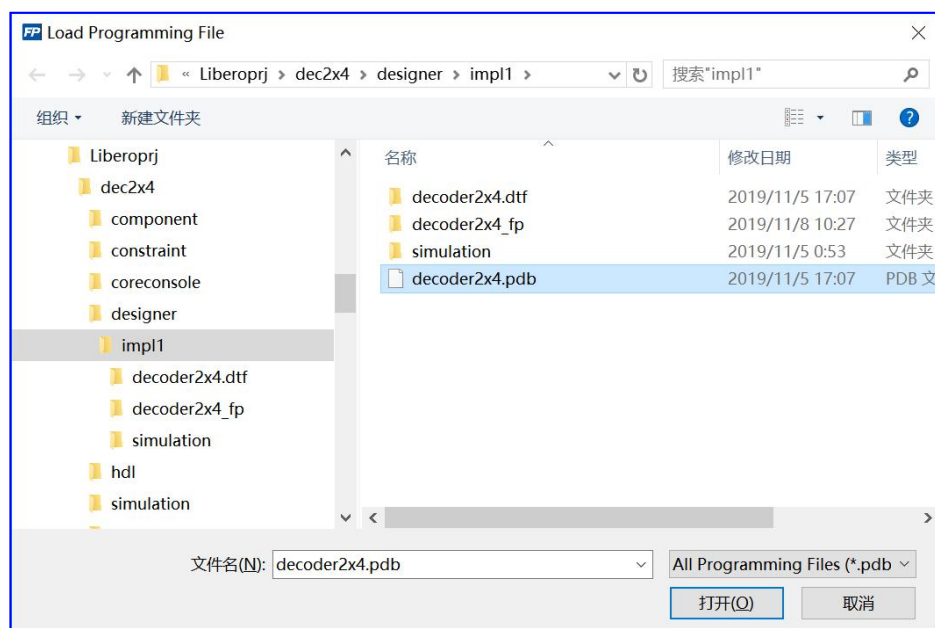
FlashPro 运行界面

此时如果“Run”按钮不可用，则需手动加载编程文件。选择“Configuration”菜单下的“Load Programming File”，如图所示。



加载编程文件

在项目文件夹下，找到之前生成的“decoder2x4.pdb”文件。



找到对应编程文件

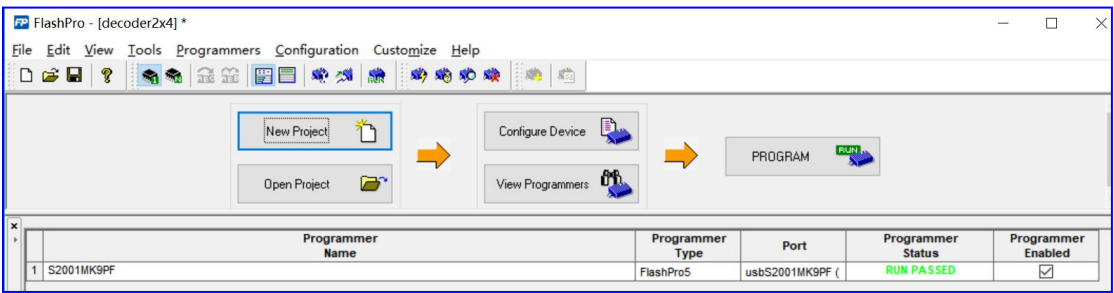
“Run”按钮变成“PROGRAM”按钮。检查烧录器硬件和FPGA芯片烧录接口连接好后，点击“PROGRAM”按钮，可开始进行烧录操作，在“Programmer Status”项和“Log Window”窗口中可看到烧录进展情况，如图所示。

Programmer Type	Port	Programmer Status	Programmer Enabled
FlashPro5	usbS2001MK9PF (ERASING.	<input checked="" type="checkbox"/>

擦除进行中

Programmer Type	Port	Programmer Status	
FlashPro5	usbS2001MK9PF (50 %	

烧录进行中



烧录完成

15. 实际测试

程序烧录到FPGA芯片上后，就可按照定义好的引脚编号及对应功能，连接输入输出设备并进行现场测试了。