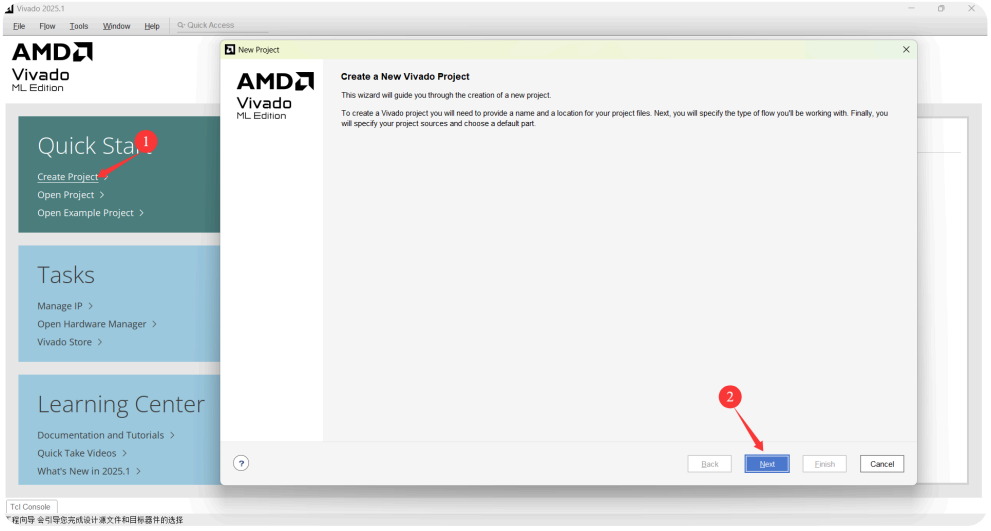
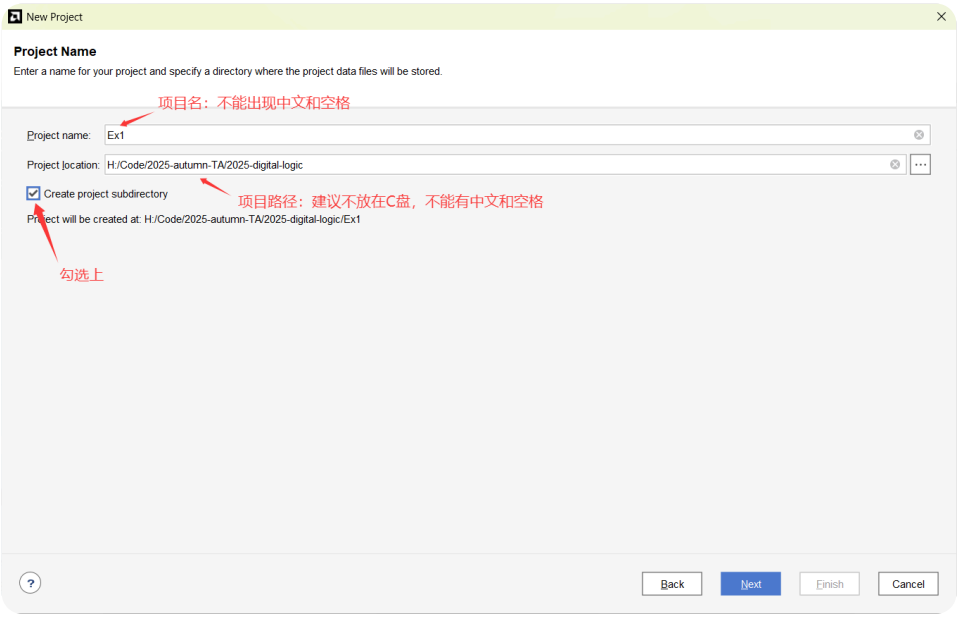
Vivado 2025.1 设计流程

1. 创建新项目

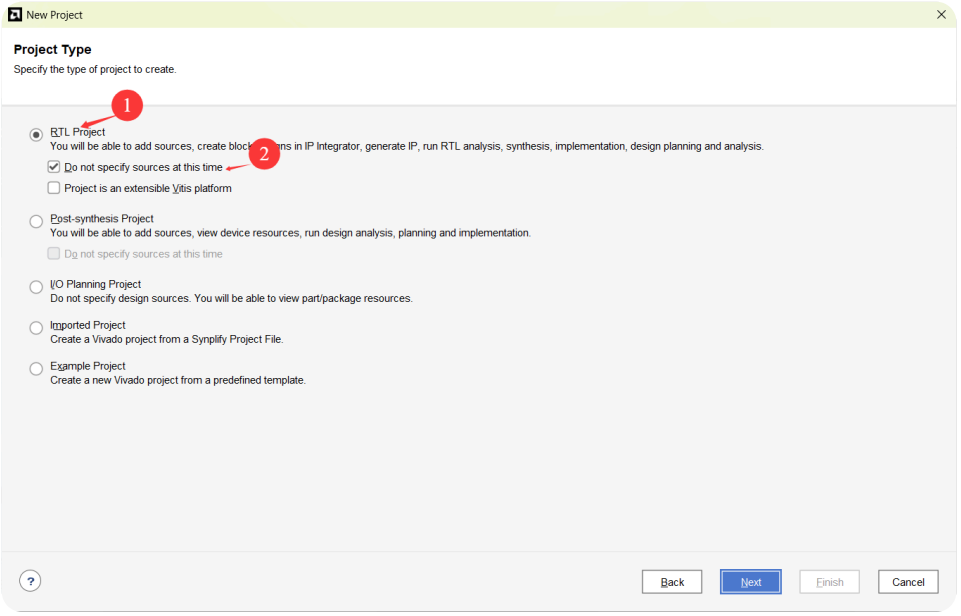
1. 打开Vivado，选择"Create Project"，点击"Next"



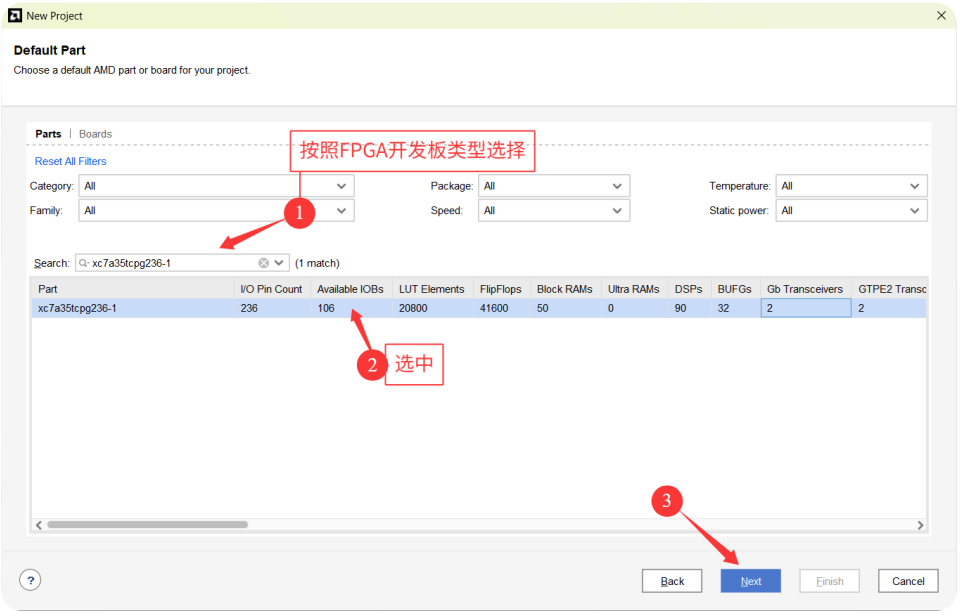
2. 按照以下方式进行配置，勾选上"Create project subdirectory"，点击"Next".



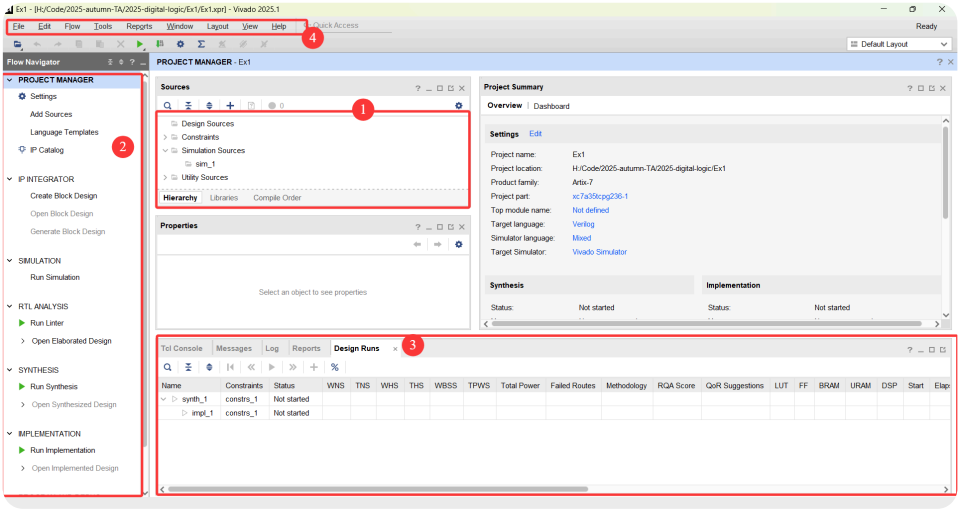
3. 选择"RTL Project", 并勾选"Do not specify sources this time", 目的是不在此处添加设计文件.



4. 选择开发板，如果是**Artix-7**则输入 `XC7A100TCSG324-1`；如果是**Basys3**，则输入 `xc7a35tcpg236-1`， **选中**后"Next". 没问题后点击"Finish".



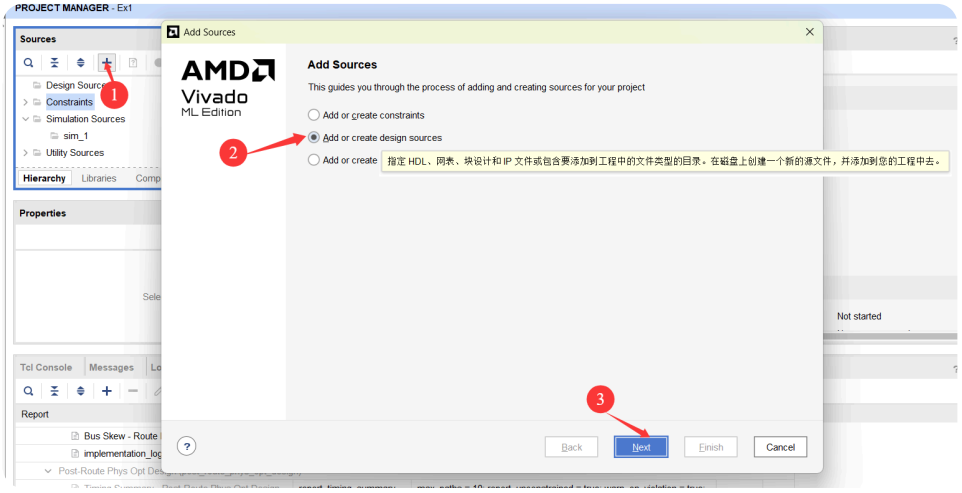
5. 界面介绍：
   - ① 代码区：Design Sources存放RTL设计代码；Simulation Sources存放仿真代码；Constraits存放约束文件【上板】
   - ② 功能区：IP INTEGRATOR(IP核)、SIMULATION(仿真)、SYNTHESIS(综合)、IMPLEMENTATION(布局布线)、PROGRAM AND DEBUG(上板流程)
   - ③ 调试区：Tcl Console(控制命令台)、Messages(警告、错误等信息)、Log(运行过程产生的日志)、Report(由功能区各个功能产生的报告)、Design Runs(设计的综合分析情况)
   - ④ 菜单区



> 【选做】将Vivado的默认代码编辑器设置为VS Code
> 如果不习惯Vivado提供的编辑器，可以将其更改，详见最后<附录0 更改Vivado的代码编辑器为VS Code>

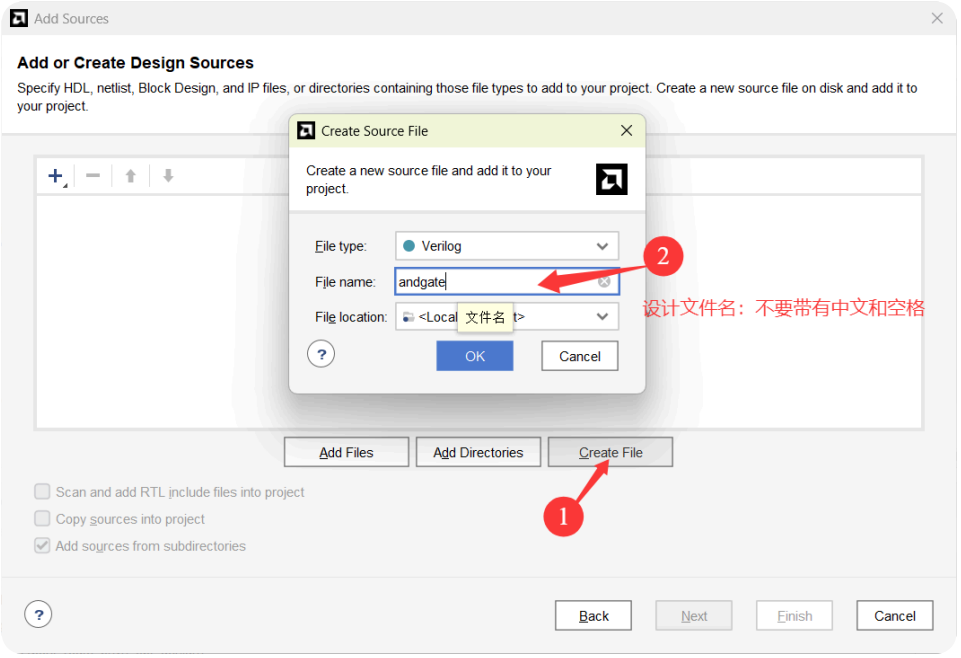
2. 创建设计文件

1. 选择"Add or create design Sources", 点击"Next".

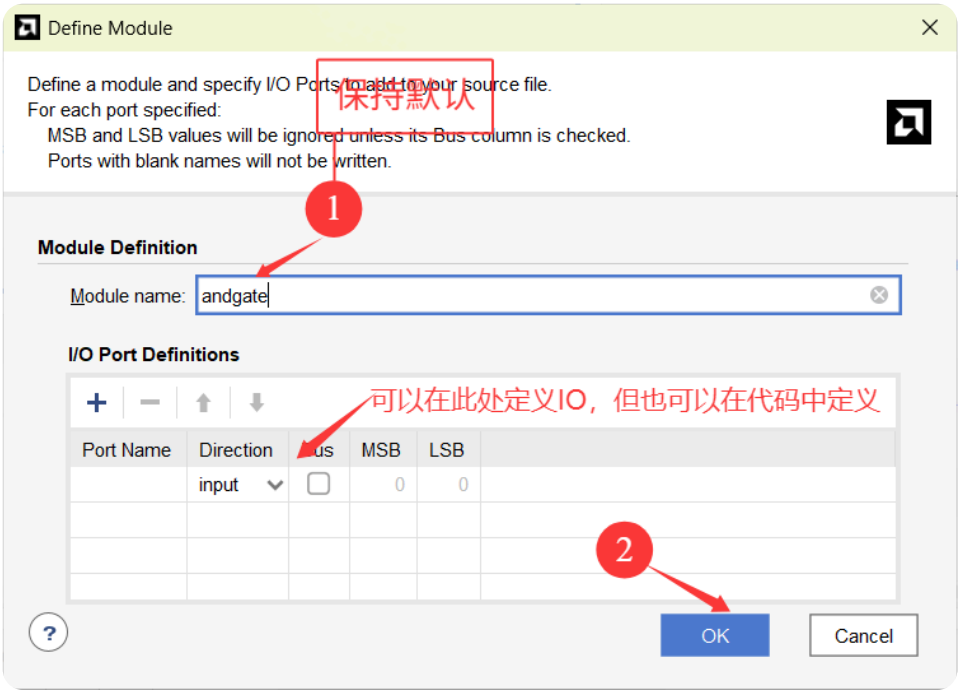


2. 新建设计文件：选择"Create File", 并输入文件名【不要带有中文和空格】，点击"OK".

> Add Files: 从已经有的.v文件直接导入
> Add Directories: 导入整个文件夹中的.v文件
> Create File: 从头新建一个.v文件



3. 确认创建：保持默认的模块名，点击"OK"；然后选择弹窗的"YES".



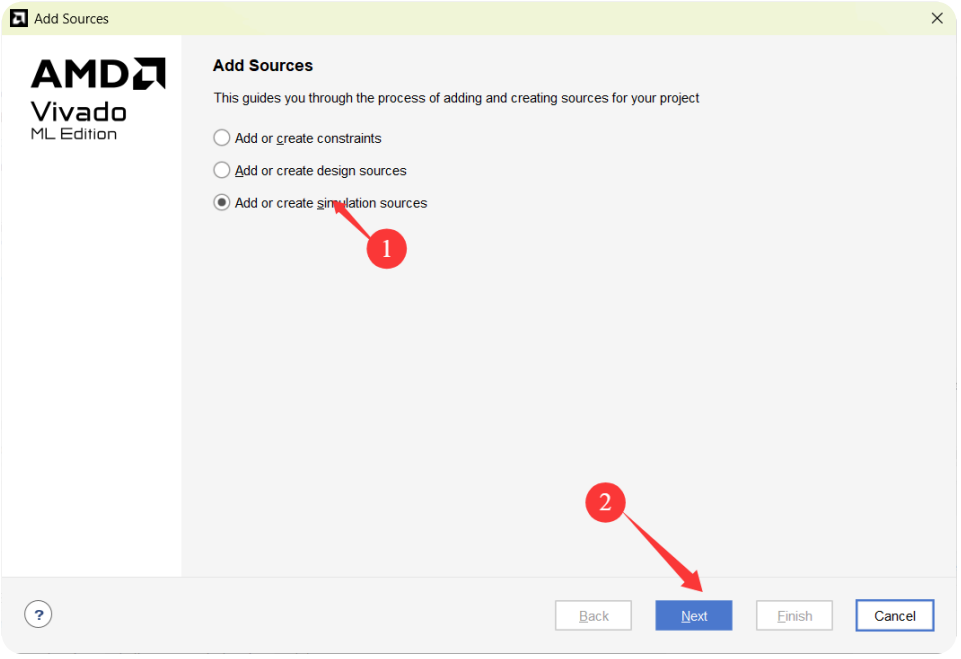
4. 双击文件进入代码编写，以与门(and)模块为例.

```
`timescale 1ns / 1ps    //1ns代表基本时间刻度为1ns, 1ps代表精度精确到1ps, 可以自己修改
module andgate          //定义模块名
#(parameter WIDTH = 1) (    //定义模块参数
    input [WIDTH-1:0] a,    //输入端口使用input
    input [WIDTH-1:0] b,    //使用英文","隔开
    output [WIDTH-1:0] c    //输出端口使用output, 最后一个端口不要有","
);
assign c = a & b;   //与门, 每条语句后使用英文";"结束
endmodule           //不要忘了endmodule

```

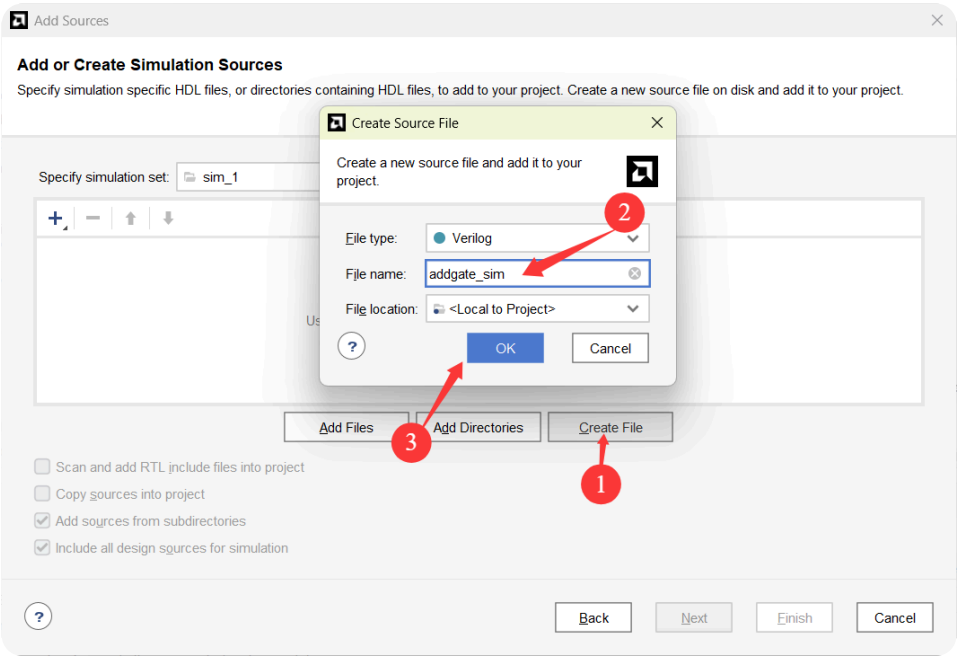
3. 创建仿真文件

1. 点击”+“，选择"Add or create simulation sources"

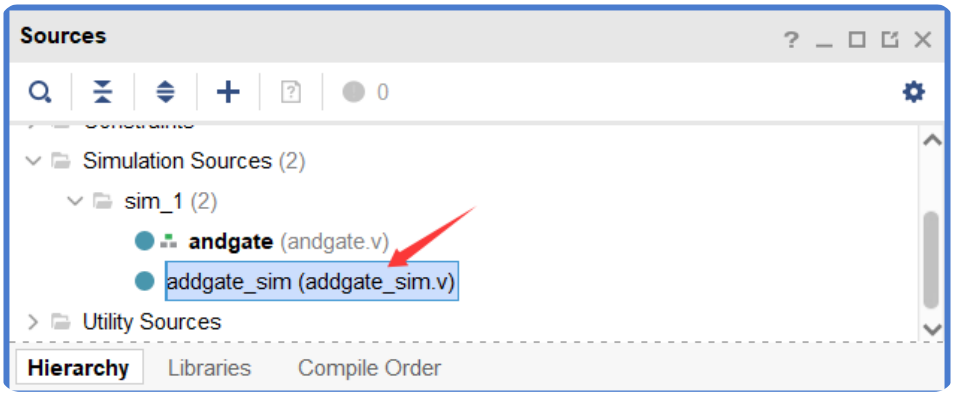


2. 接下来步骤同创建设计文件, 文件名可以添加上"_sim"后缀, 点击"OK".

> 图片有误, 改成"andgate_sim"



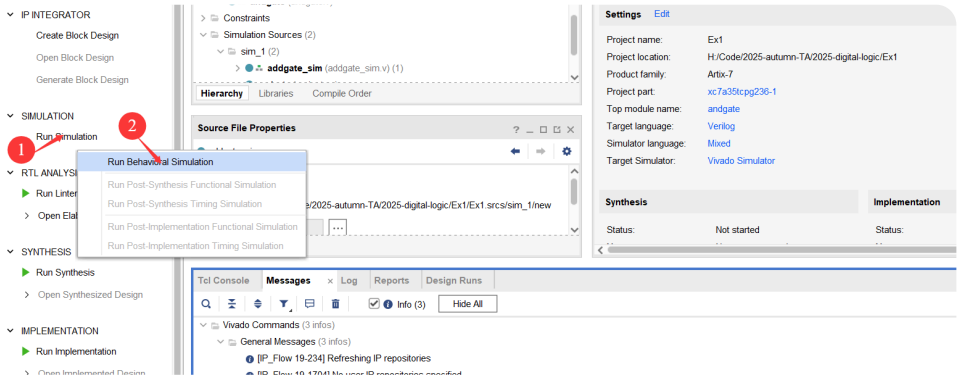
3. 在Simulation Sources文件夹下双击addgate_sim编写仿真代码.




```
`timescale 1ns / 1ps
module andgate_sim();
    //创建andgate模块的输入输出端口
    reg [31:0] a;
    reg [31:0] b;  //input
    wire [31:0] c; //output
    //实例化模块
    andgate #(32) andgate1(
        .a(a),
        .b(b),
        .c(c)
    );
    initial begin
        a = 32'h0000_0000;
        b = 32'h0000_0000;          //初始化输入端口
        #100 a = 32'hffff_ffff;    //#100代表经过100ns后进行的操作
        #50 begin
          a = 32'haabb_89ff;
          b = 32'hffff_0000;
        end                         //多个操作需要使用begin end包括起来
```

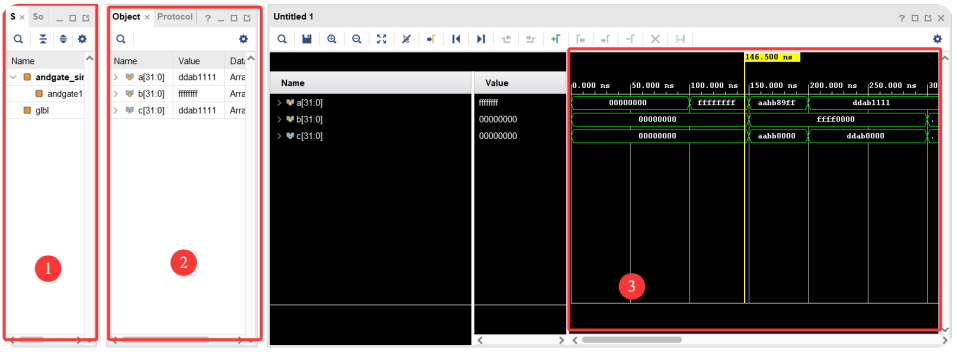
```
21          #50 a = 32'hddab_1111;
22          #100 b = 32'hffff_ffff;
23      end
24  endmodule
25
```

4. 进行仿真："Run Simulation" -> "Run Behaviral Simulation"



5. 观察仿真波形图

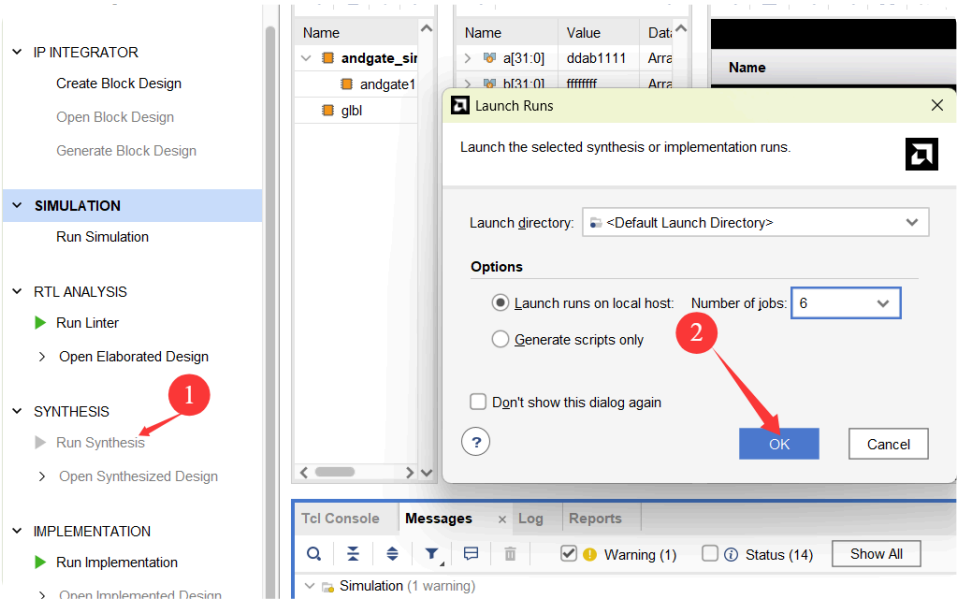
> ① 查看参与仿真的各个模块
> ② 查看该模块下对应的变量名和值
> ③ 查看各个时刻的波形图



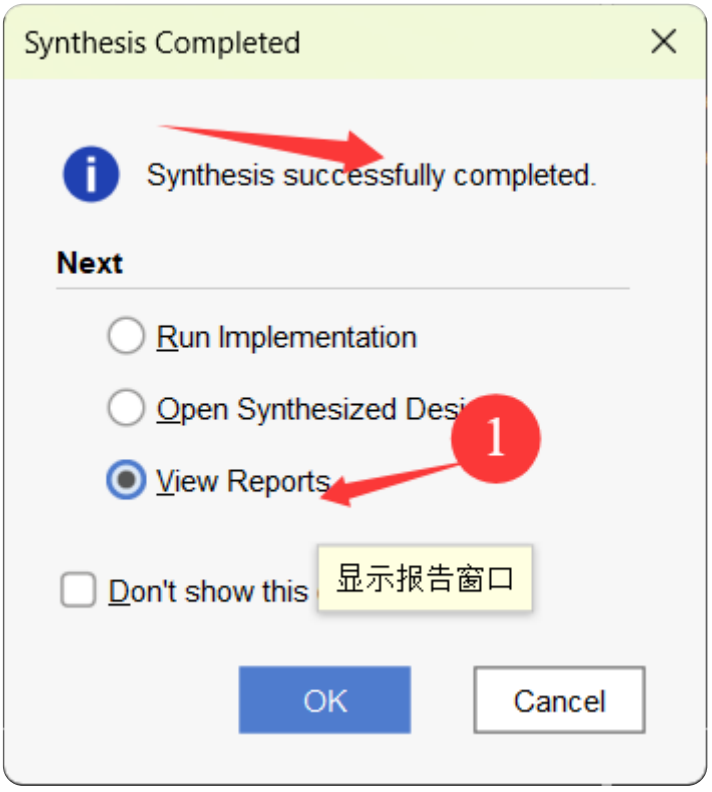
4. 综合流程

1. 实现综合：选择"Run Synthesis" -> "OK"

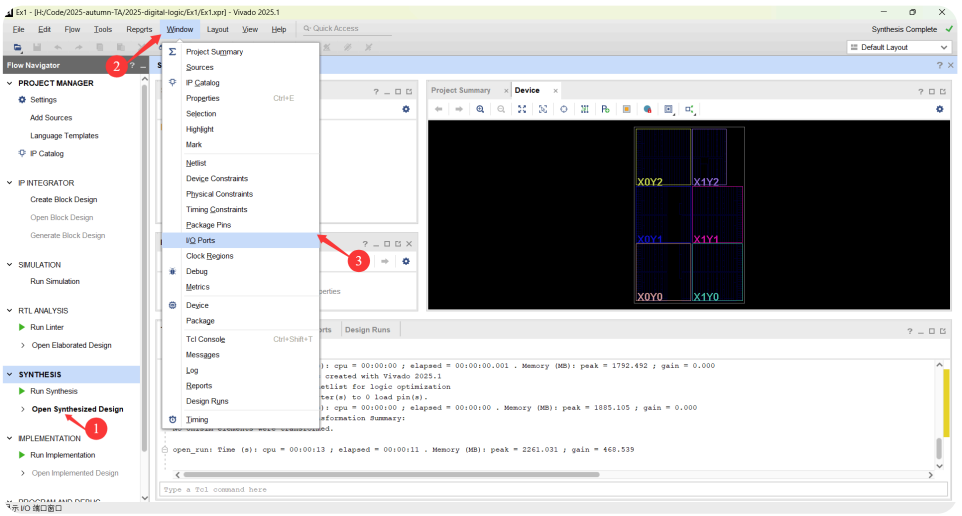
> 不同的电脑对同一个项目的综合时间有所不同, 可以更改number of jobs(数字越大理论上速度越快，但是这也取决于你的CPU配置哦)



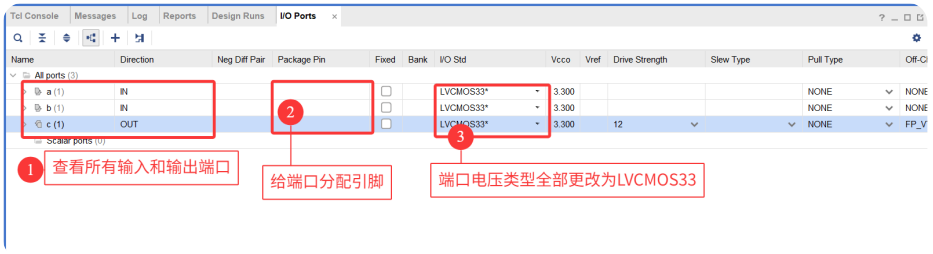
2. 查看报告：若显示"Synthesis successfully completed"则代表综合成功；可以查看报告，也可以直接选择"Cancel"



3. 分配引脚：这里介绍使用vivado可视化界面进行分配
   - 综合完成后，点击"Open Synthesized Design", 等待加载完成
   - 选择菜单栏Windows --> I/O Ports



- 查看IO Ports



① 查看顶层模块的所有端口，需要点左侧的展开图标

② Package Pin：需要将该端口分配给FPGA上的哪一个引脚

- Basys3引脚分配：

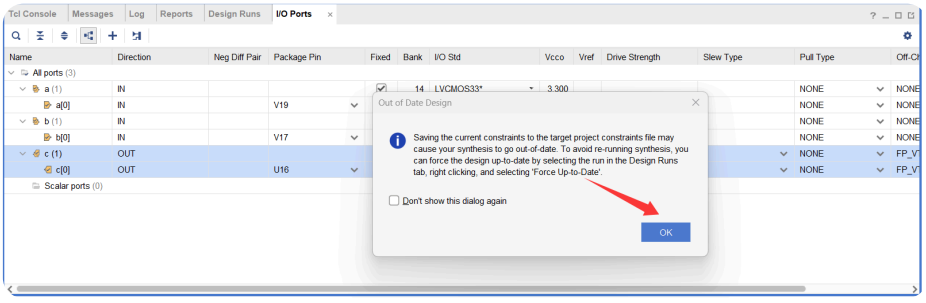
LED	PIN	CLOCK	PIN	SWITCH	PIN	BUTTON	PIN	Seven-segment digital tube	PIN
LD0	U16	MRCC	W5	SW0	V17	BTNU	T18	AN0	U2
LD1	E19			SW1	V16	BTNR	T17	AN1	U4
LD2	U19			SW2	W16	BTND	U17	AN2	V4
LD3	V19			SW3	W17	BTNL	W19	AN3	W4
LD4	W18			SW4	W15	BTNC	U18	CA	W7
LD5	U15			SW5	V15			CB	W6
LD6	U14			SW6	W14			CC	U8
LD7	V14			SW7	W13			CD	V8
LD8	V13	USB(J2)	PIN	SW8	V2			CE	U5
LD9	V3	PS2_CLK	C17	SW9	T3			CF	V5
LD10	W3	PS2_DAT	B17	SW10	T2			CG	U7
LD11	U3			SW11	R3			DP	V7
LD12	P3			SW12	W2				
LD13	N3			SW13	U1				
LD14	P1			SW14	T1				
LD15	L1			SW15	R2				

VGA	PIN	JA	PIN	JB	PIN	JC	PIN	JXADC	PIN
RED0	G19	JA0	J1	JB0	A14	JC0	K17	JXADC0	J3
RED1	H19	JA1	L2	JB1	A16	JC1	M18	JXADC1	L3
RED2	J19	JA2	J2	JB2	B15	JC2	N17	JXADC2	M2
RED3	N19	JA3	G2	JB3	B16	JC3	P18	JXADC3	N2
GRN0	J17	JA4	H1	JB4	A15	JC4	L17	JXADC4	K3
GRN1	H17	JA5	K2	JB5	A17	JC5	M19	JXADC5	M3
GRN2	G17	JA6	H2	JB6	C15	JC6	P17	JXADC6	M1
GRN3	D17	JA7	G3	JB7	C16	JC7	R18	JXADC7	N1
BLU0	N18								
BLU1	L18								
BLU2	K18								
BLU3	J18								
HSYNC	P19								
YSYNC	R19								

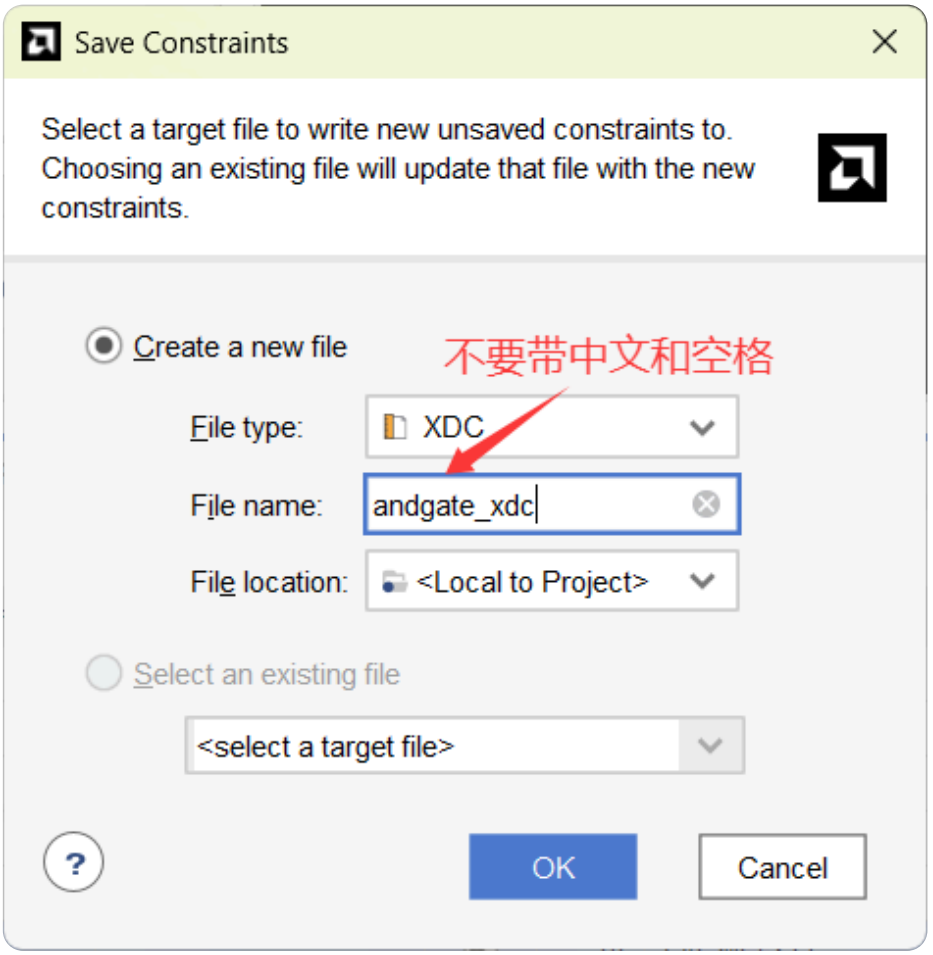
- NEXYS 4 DDR引脚分配：

LED 信号	FPGA 引脚	数码管信号	FPGA 引脚	SW 信号	FPGA 引脚	其他 I/O 信号	FPGA 引脚
LD0	H17	CA	T10	SW0	J15	BTNC	N17
LD1	K15	CB	R10	SW1	L16	BTNU	M18
LD2	J13	CC	K16	SW2	M13	BTNL	P17
LD3	N14	CD	K13	SW3	R15	BTNR	M17
LD4	R18	CE	P15	SW4	R17	BTND	P18
LD5	V17	CF	T11	SW5	T18	CLK100MHz	E3
LD6	U17	CG	L18	SW6	U18		
LD7	U16	DP	H15	SW7	R13		
LD8	V16	AN[0]	J17	SW8	T8		
LD9	T15	AN[1]	J18	SW9	U8		
LD10	U14	AN[2]	T9	SW10	R16		
LD11	T16	AN[3]	J14	SW11	T13		
LD12	V15	AN[4]	P14	SW12	H6		
LD13	V14	AN[5]	T14	SW13	U12		
LD14	V12	AN[6]	K2	SW14	U11		
LD15	V11	AN[7]	U13	SW15	V10		

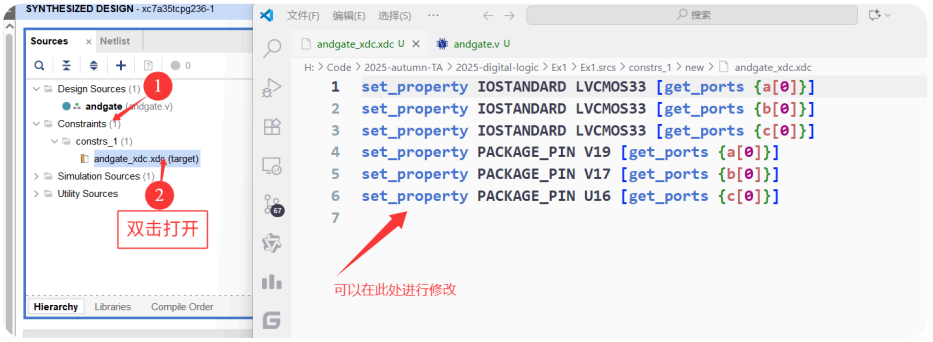
  - 保存约束文件：ctrl + S 保存约束文件，点击"OK"



  - 输入约束文件名，点击"OK"

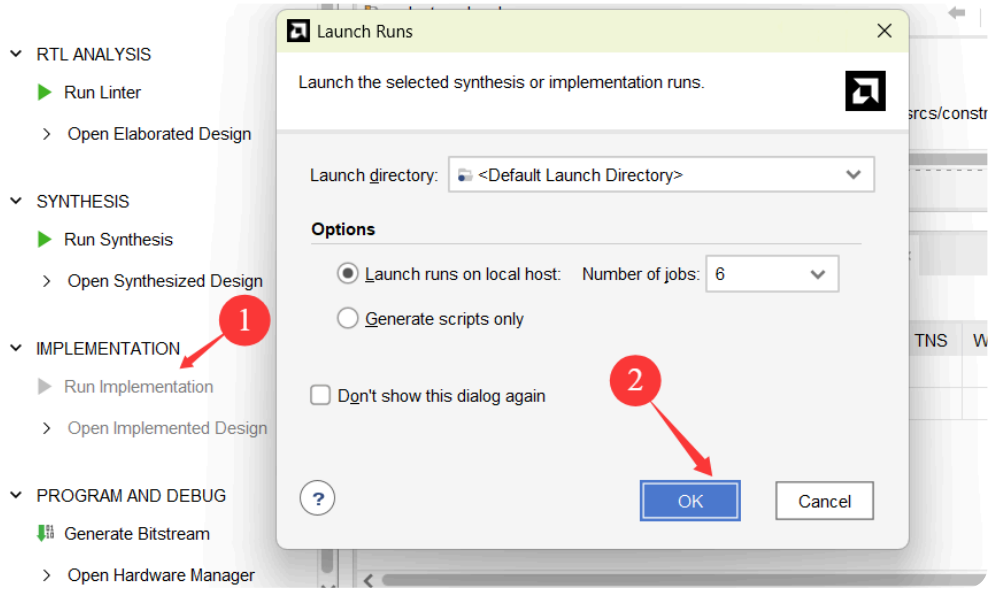


- 查看/更改约束文件：

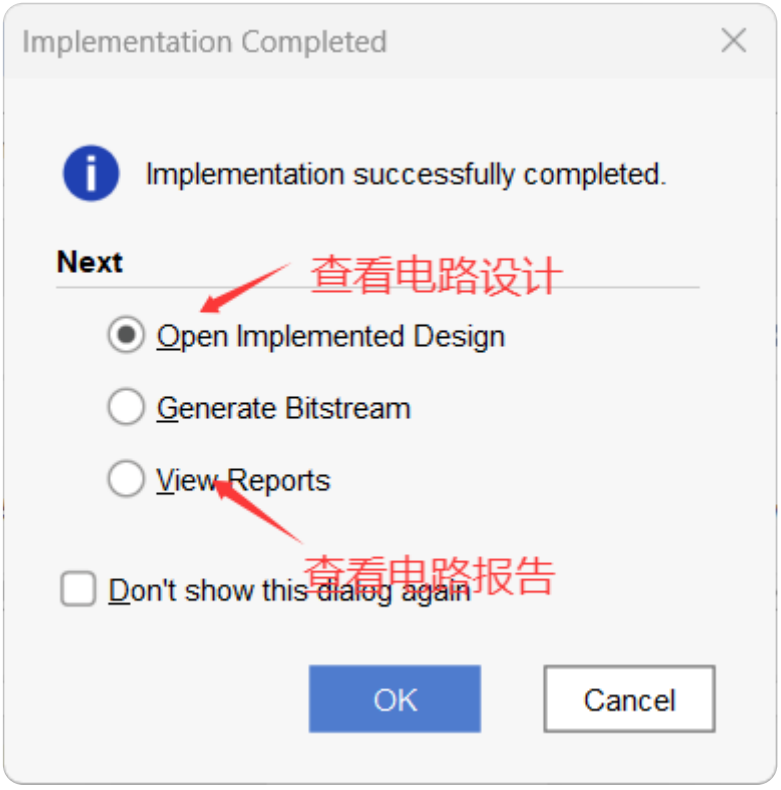


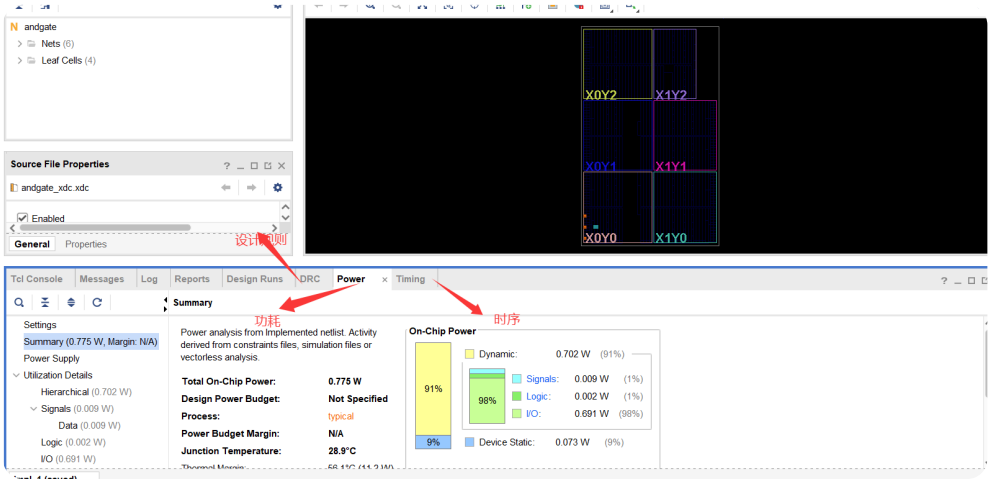
5. 布局布线(Implementation)

1. 完成IMPLEMENTATION



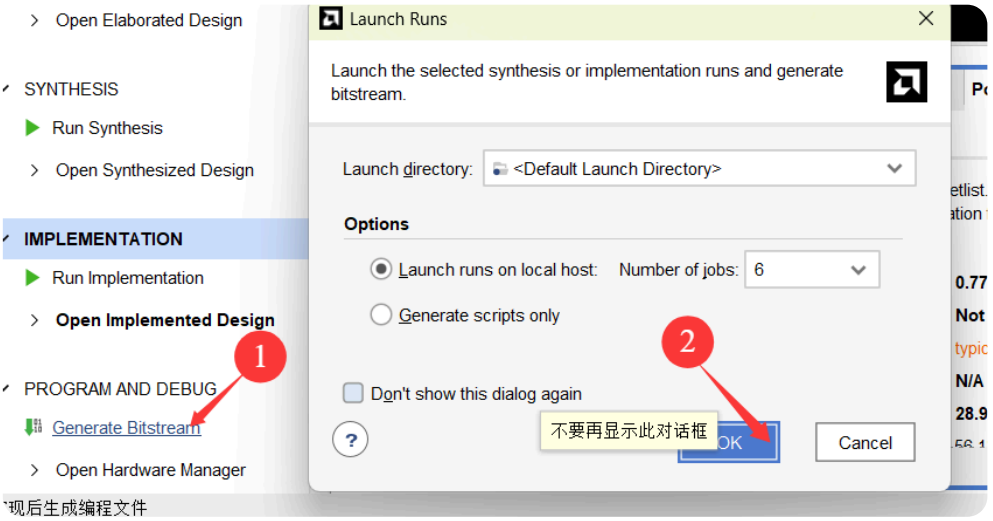
- 完成后可以查看电路设计的性能分析报告：



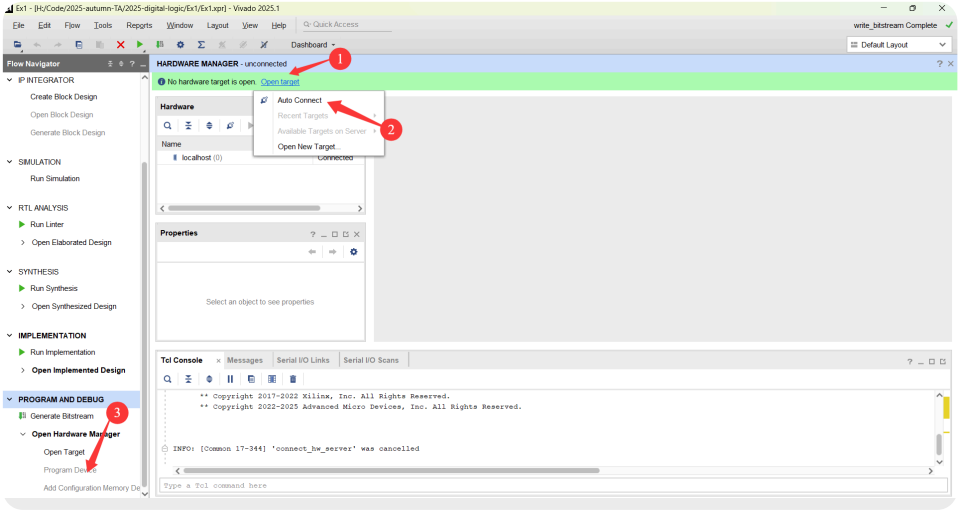


6. 生成bitstream

1. 上板前，需要生成bit流，硬件电路上本质是二进制运算



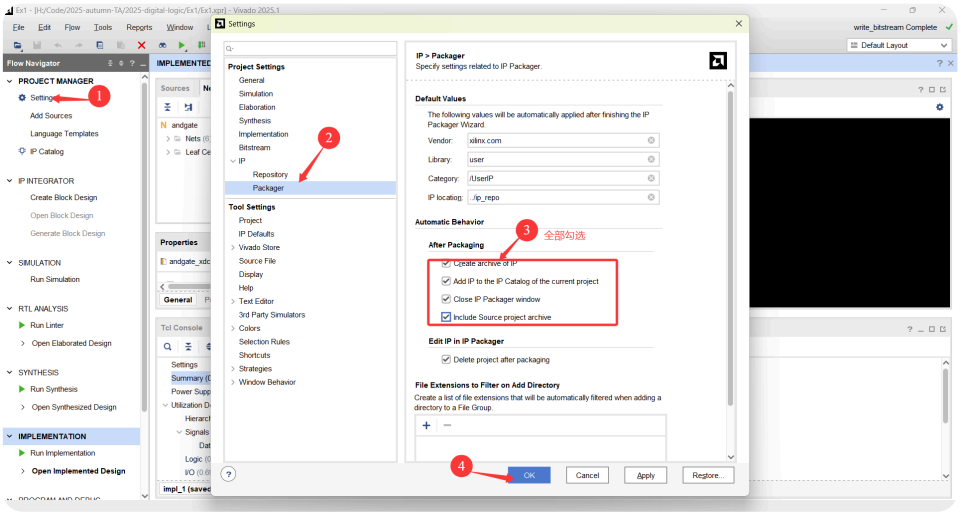
2. 此时，需要使用USB连接线将FPGA开发板和电脑连接，然后点击"Open Hardware Manager":



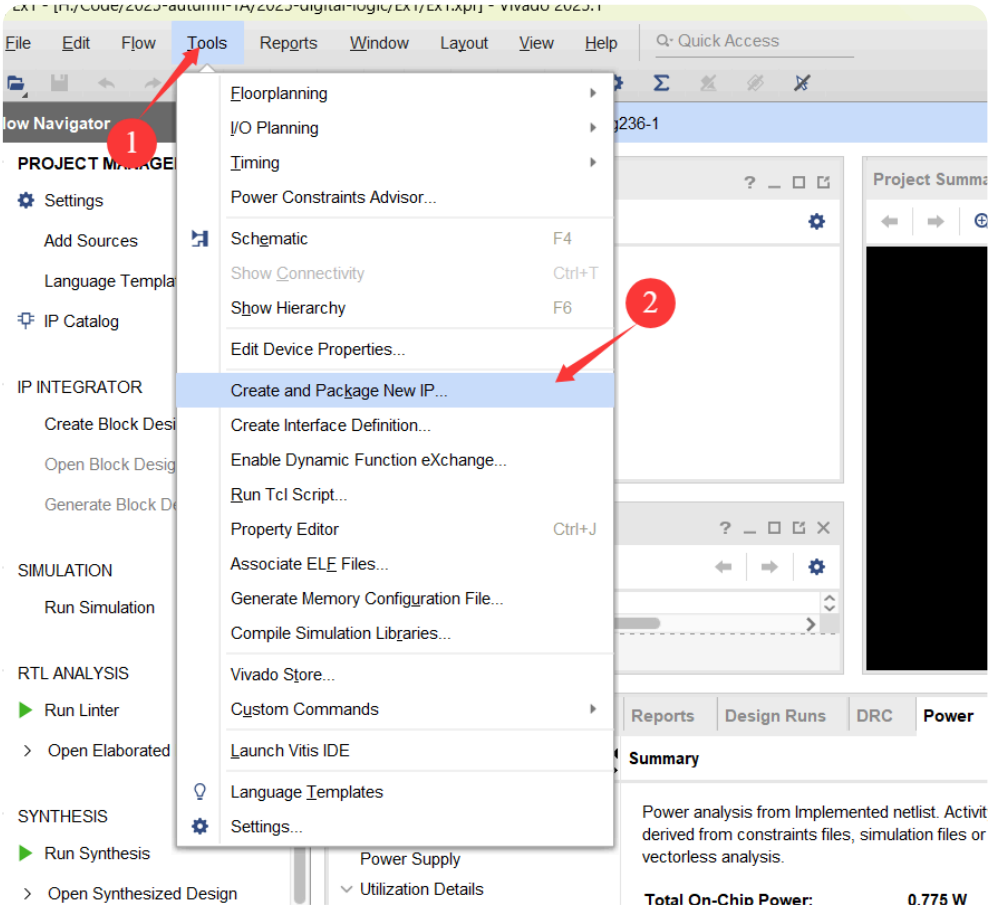
话题1 封装IP核

在仿真结果正确后，需要**完成综合流程**，才能封装IP核

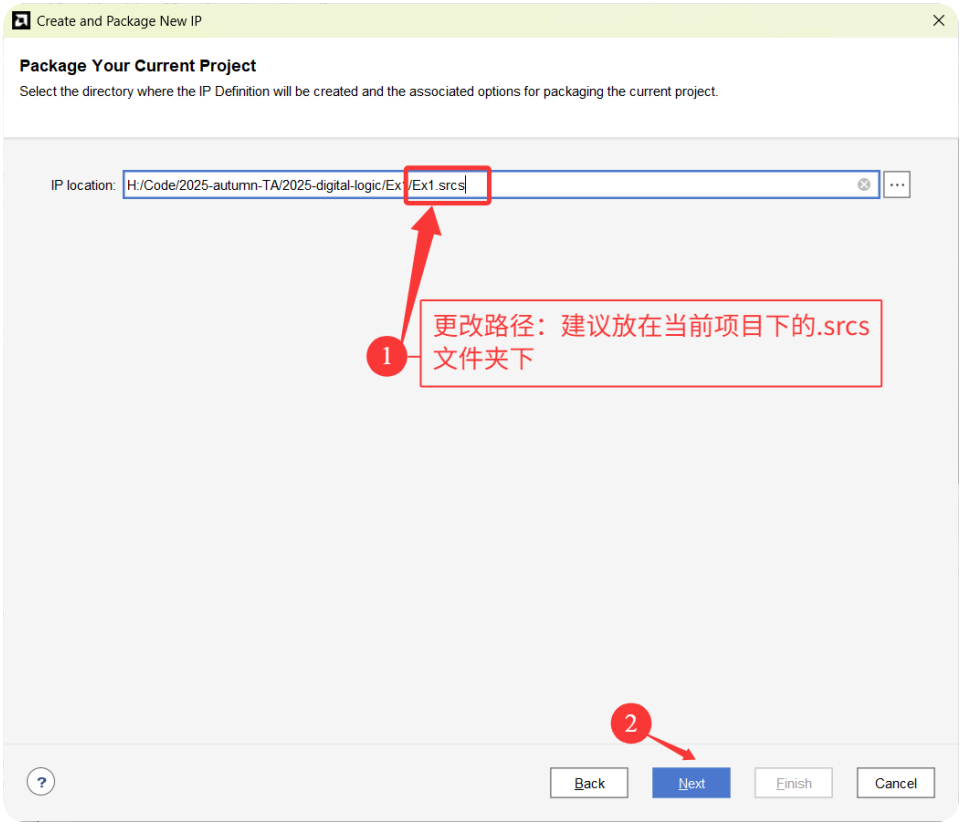
1. 封装IP核之前的设置：Setting -> Packager -> 勾选配置 -> OK



2. 设置封装路径：选择后，连续点击两次"Next"

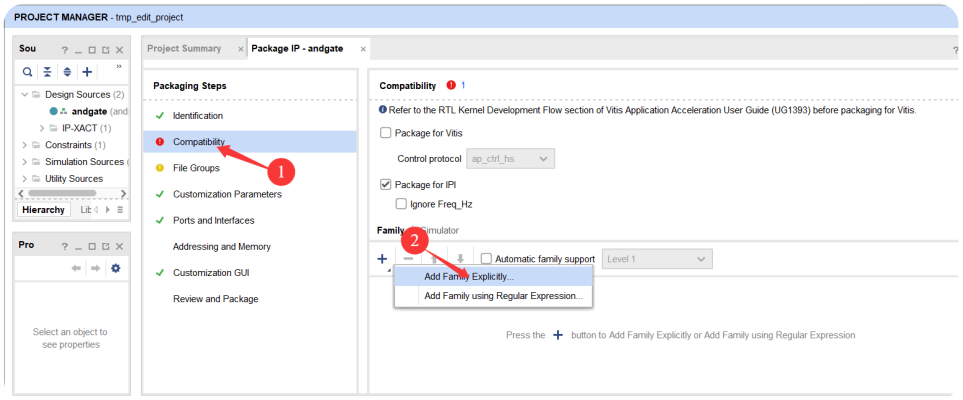


- 建议将IP路径更改为当前项目下的xxx.srcs文件夹下，点击"Next". 然后依次点击"OK" -> Finish

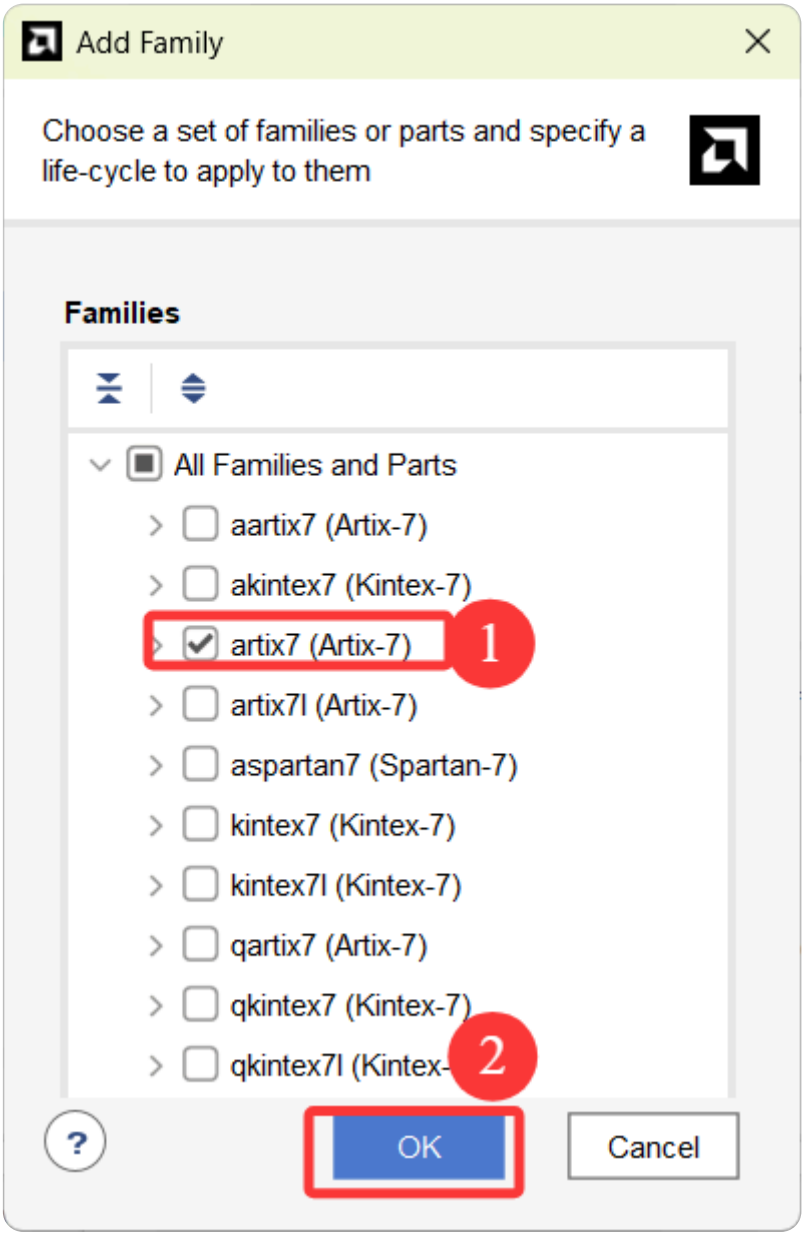


3. IP配置

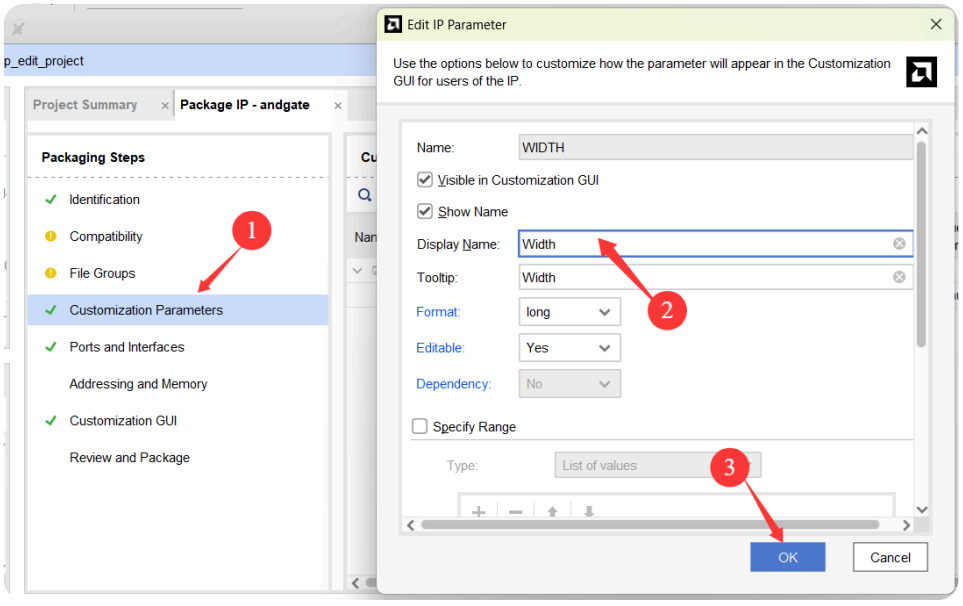
- 兼容性：Add Family



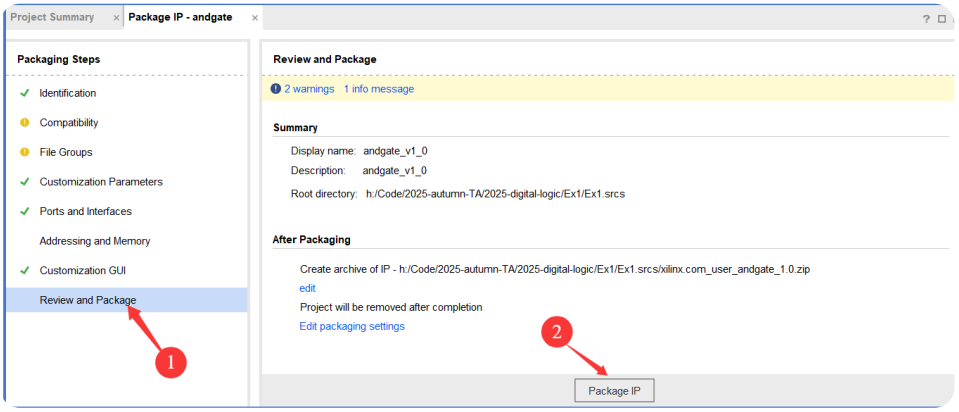
- 只需要"artix7"系列被选中即可



- 双击"Customization Parameters" -> 编辑参数名 -> “OK”



- 点击"Review and Package" --> "Package IP"，完成后将保存在...../xxx.srcs/xilinx.com_user_andgate_1.0.zip



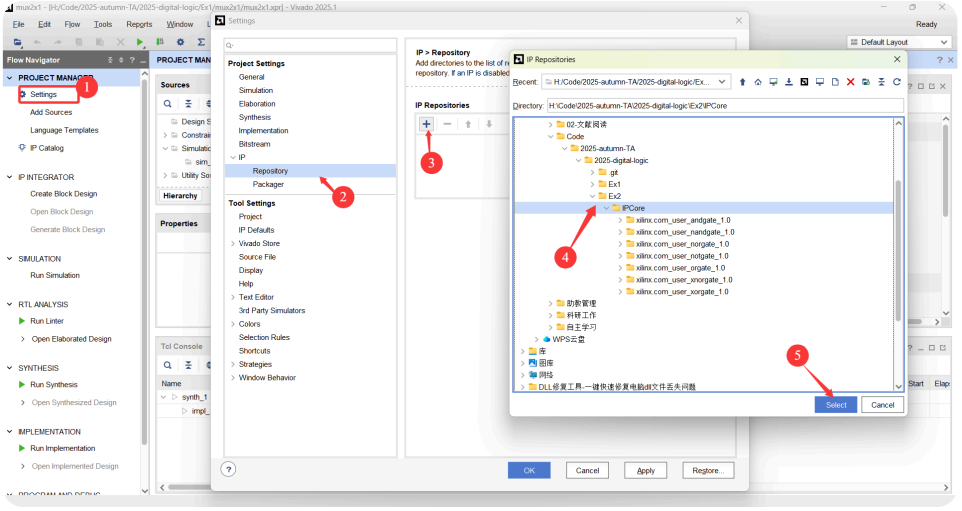
4. 封装更多门电路：按照以上方式分别设计 或门(or)、非门(not)、与非门(NAND)、或非门(NOR)、异或门(XOR)、同或门(XNOR).

话题2 使用IP核进行电路设计

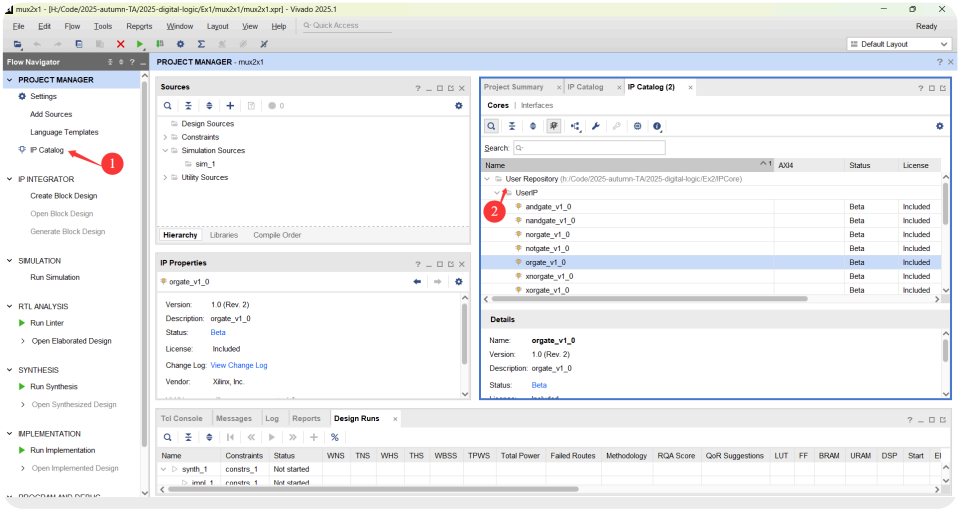
1. 新建一个项目，将封装的基本门IP核压缩包拷贝到一个空文件夹，如IPCore，然后解压：

名称	修改日期	类型	大小
xilinx.com_user_andgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_nandgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_norgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_notgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_orgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_xnorgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_xorgate_1.0	2025/9/18 16:29	文件夹	
xilinx.com_user_andgate_1.0	2025/9/18 12:51	WinRAR ZIP 压缩...	20 KB
xilinx.com_user_nandgate_1.0	2025/9/18 15:03	WinRAR ZIP 压缩...	30 KB
xilinx.com_user_norgate_1.0	2025/9/18 15:08	WinRAR ZIP 压缩...	34 KB
xilinx.com_user_notgate_1.0	2025/9/18 14:56	WinRAR ZIP 压缩...	25 KB
xilinx.com_user_orgate_1.0	2025/9/18 14:41	WinRAR ZIP 压缩...	20 KB

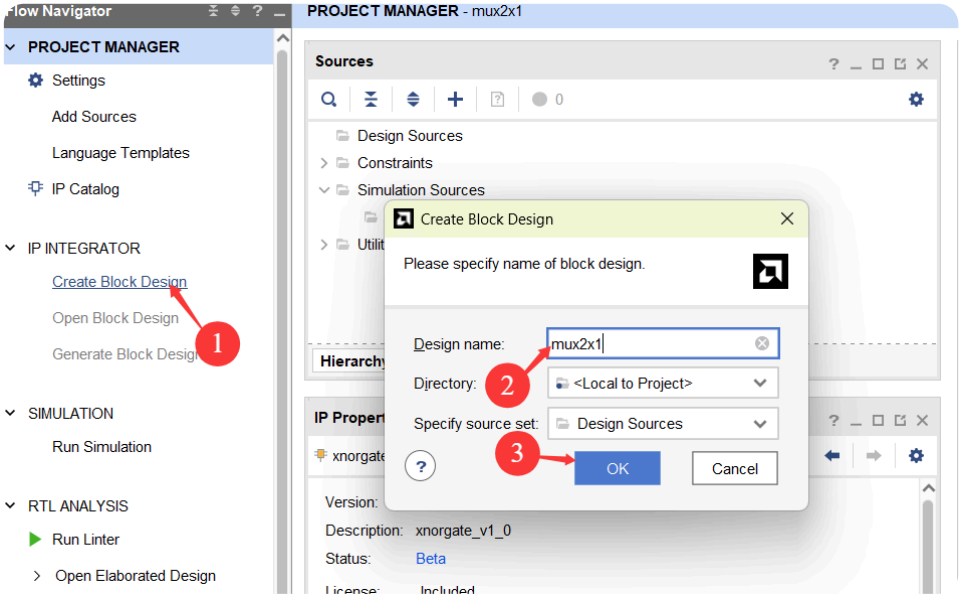
2. 导入IP核：按照以下方式将1中的文件夹添加到项目中，然后选择"OK".



3. 查看IP核：点击"IP Catalog" --> UserIP查看导入的IP核，双击可以查看对应的门电路

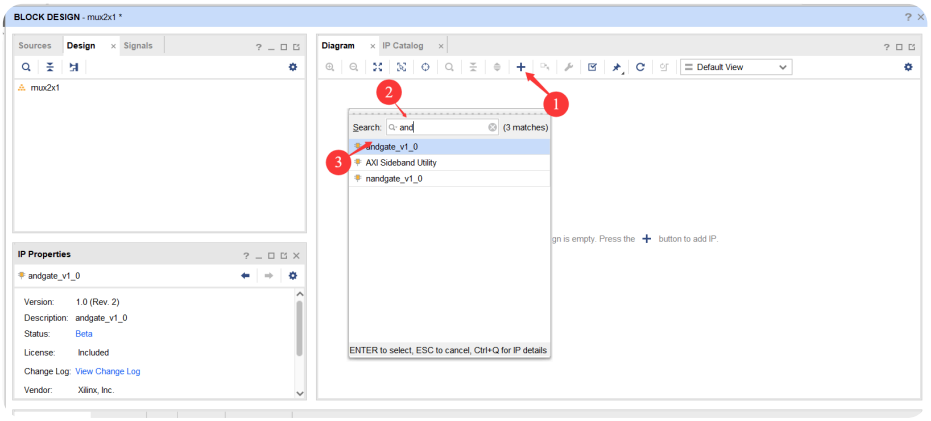


4. 创建Block Design：点击"IP INTEGRATOR"下的Create Block Design --> 输入name --> "OK".

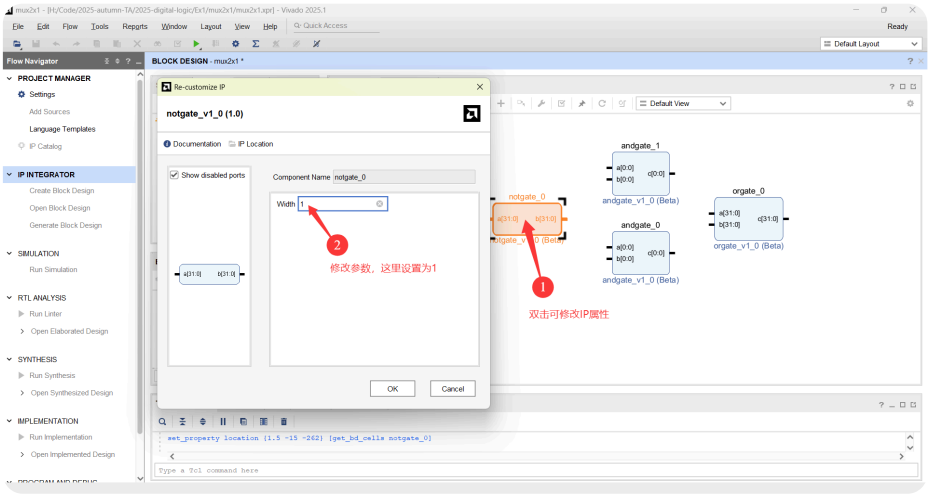


5. 实现mux2x1：二选一多路选择器表达式为$F = \bar{s}a + sb$，需要两个与门、一个非门和一个或门.

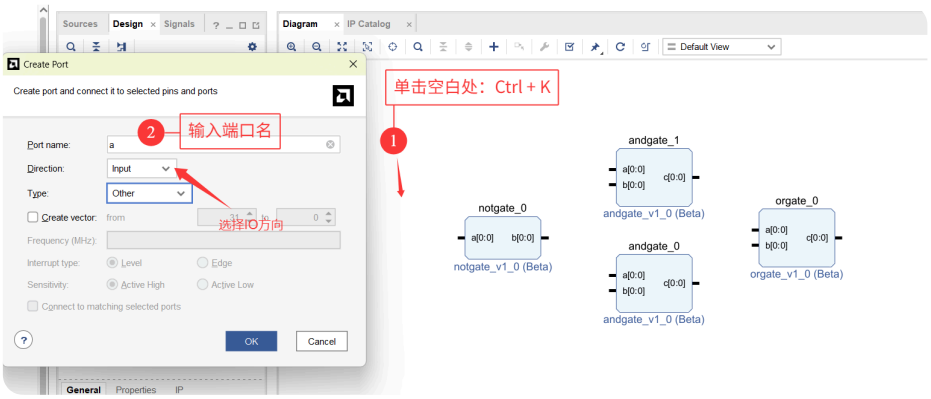
   - 点击"+"，依次搜索andgate、notgate、orgate, 双击添加.



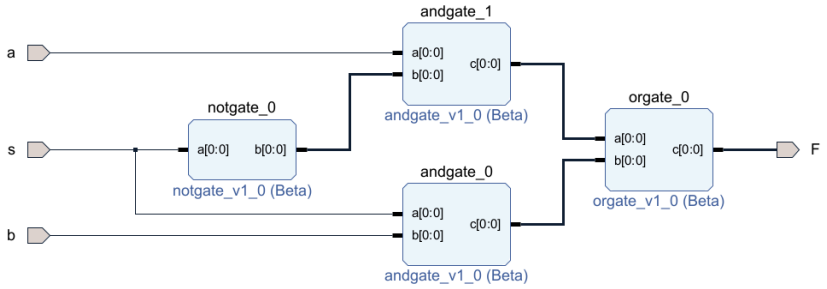
   - 修改IP参数：这里将所有端口位宽设置为1



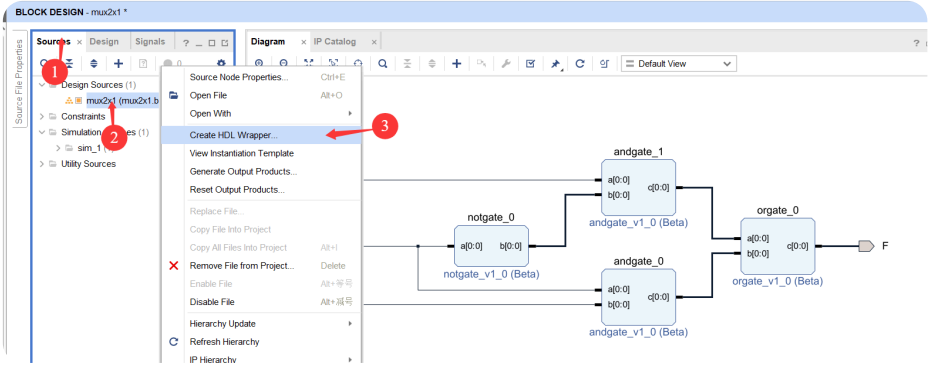
- 创建端口：按照以下方式创建端口，注意选择端口方向



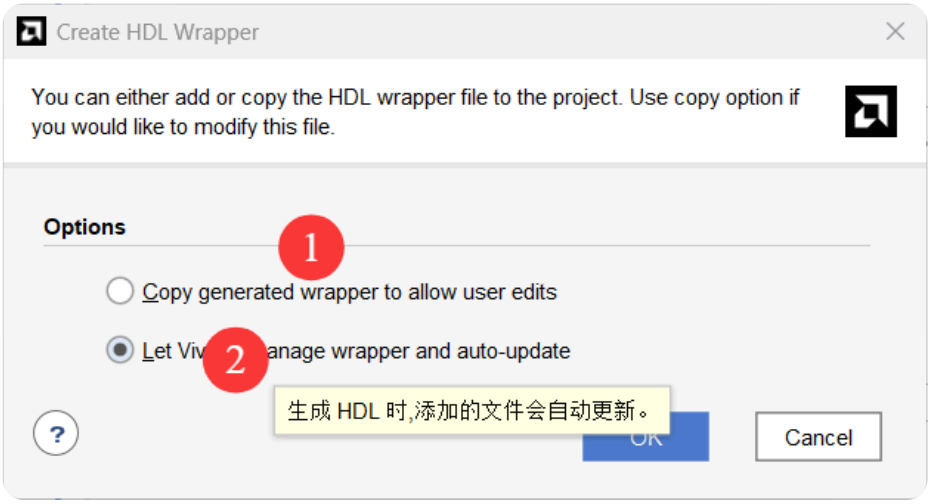
- 添加连接线：按照如下方式添加连接线



- 创建HDL Wrapper：



- 选择Wrapper方式：后续需要修改文件时选择①，让Vivado自动管理选择②；然后点击"OK".



6. 对mux2x1进行仿真，提供参考的示例仿真代码为：

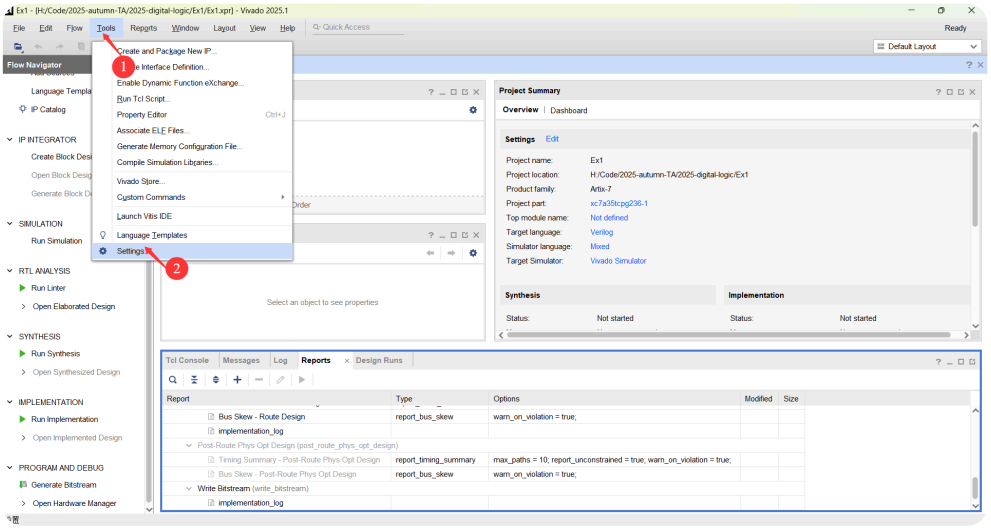
```
`timescale 1ns / 1ps
module tb_mux2x1;
    // mux2x1_wrapper Inputs
    reg   a = 0;
    reg   b = 0;
    reg   s = 0;
    // mux2x1_wrapper Outputs
    wire      F;
    mux2x1_wrapper  u_mux2x1_wrapper (
        .a(a),
        .b(b),
        .s(s),

        .F(F)
    );
    initial
    begin
        #50 begin
          a = 0; b = 1; s = 0;
        end
        #50 begin
          a = 1; b = 0; s = 0;
        end
        #50 begin
          a = 1; b = 1; s = 0;
        end
        #50 begin
          a = 0; b = 0; s = 1;
        end
        #50 begin
          a = 0; b = 1; s = 1;
        end
        #50 begin
          a = 1; b = 0; s = 1;
        end
```

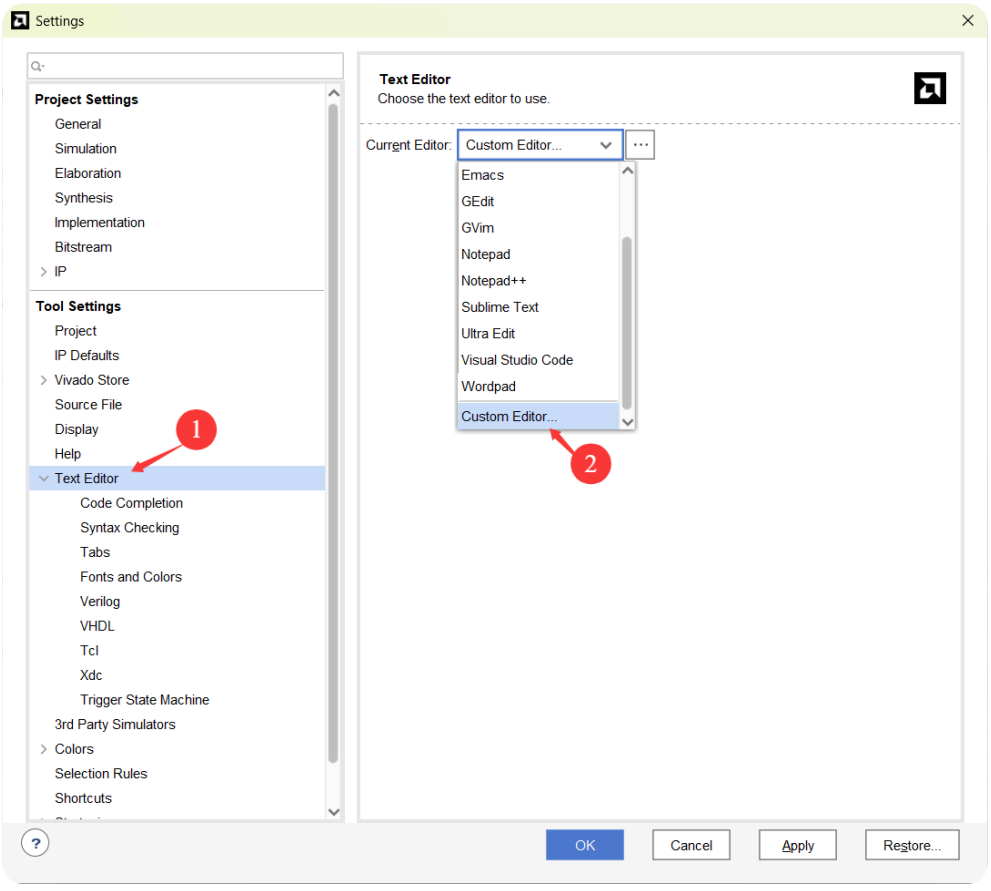
```
36          #50 begin
37            a = 1; b = 1; s = 1;
38          end
39          $finish;
40      end
41  endmodule
```

附录1 更改Vivado的代码编辑器为VS Code

1. 选择菜单栏"Tools" -> Settings



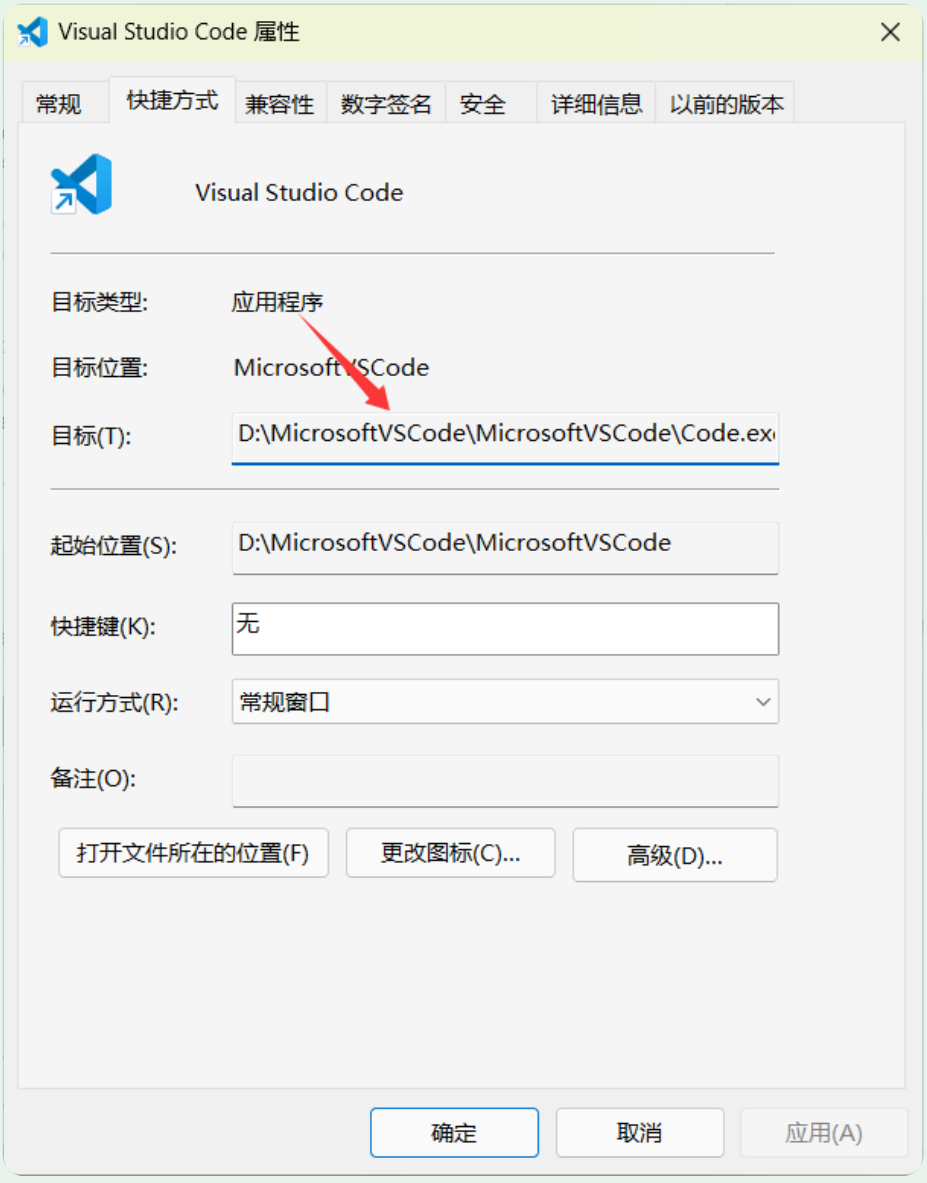
2. 选择"Text Editor" -> 单击"Custom Editor"



3. 按照以下流程操作，在②中输入以下内容；然后点击"OK", 最后选择"Apply"即可.

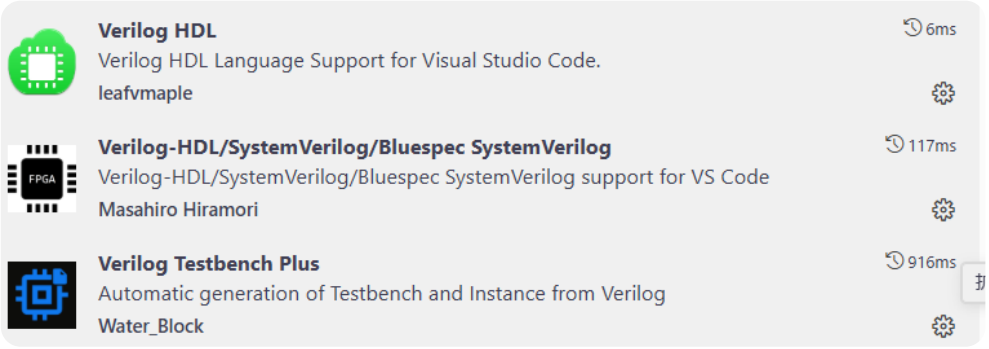
```
cmd /S /k ""替换成你自己的VSCODE绝对路径" -g [file name]:[line number]"
```

路径查看：在桌面VS CODE查看属性，将目标处的内容复制到替换处即可
温馨提示：**该绝对路径最好不要带任何空格**，比如"Microsoft VS CODE"建议更改为"MicrosoftVSCODE". 更改软件路径后记得此处的目标也要跟着变化



附录2 如何让你的VS Code编写Verilog代码更加高效

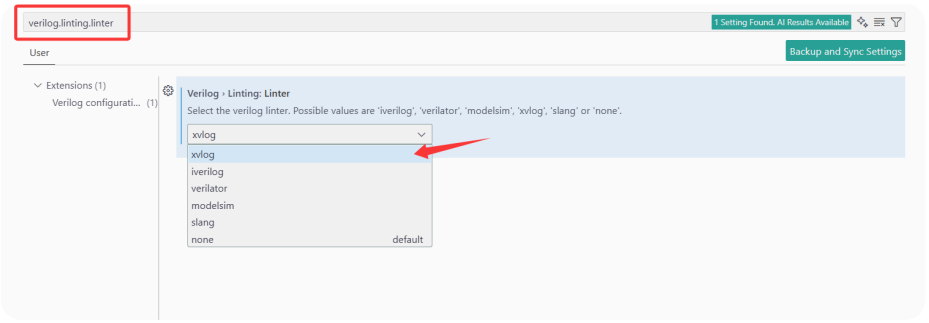
1. VS Code 中安装插件： Verilog HDL 、 Verilog-HDL/SystemVerilog/Bluespec SystemVerilog、Verilog Testbench Plus.



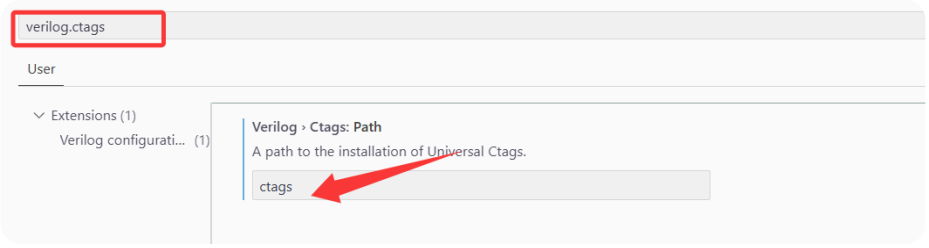
2. 添加环境变量：找到你安装vivado的路径，比如我的是`D:\ToolsCode\Xilinx\2025.1\Vivado\bin`，将其添加到系统PATH环境变量中。

> 如果你不知道怎么添加环境变量，百度一下吧~

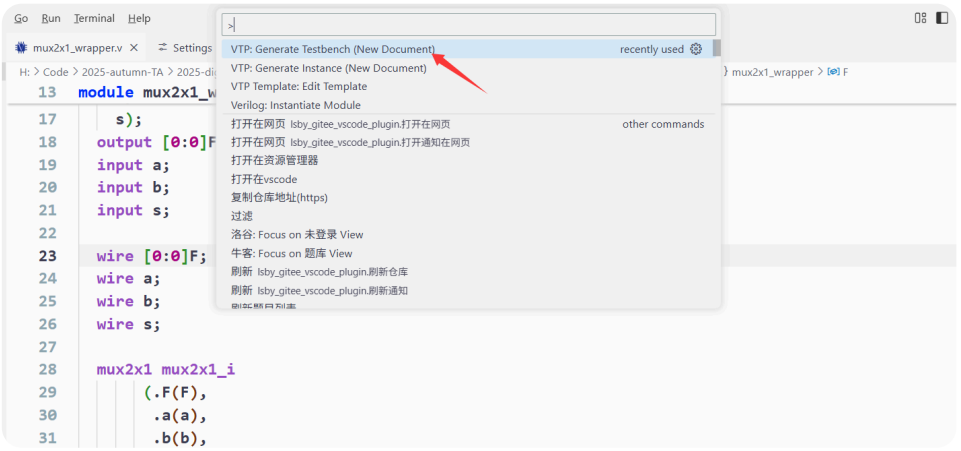
3. 下载ctags(附件中已经提供)：将ctags压缩包解压，放入某个文件夹，比如我放入`D:\ToolsCode\Xilinx\ctags`中，然后添加到环境变量。
4. 完善VSCODE设置：
   - 打开VSCODE设置，搜索框输入`verilog.linting.linter`，勾选`xvlog`，目的是使用vivado的语法检查工具在VSCODE进行语法检查



   - 搜索框中搜索`verilog.ctags`，然后输入`ctags`.【请确保ctags已经添加到环境变量】



5. 演示：随便编写一个模块，在VSCODE中打开，比如以`mux2x1_wrapper.v`为例，使用快捷键`Ctrl + Shift + P`，搜索框输入`VTPGT`，选择:



6. 这就为你自动生成了testbench仿真代码框架啦，当然如果想生成你自己的模板，自行百度一下吧~

> 已知bug：该方式生成的testbench的输出端口永远只有一位宽，需要你手动修改一下

```
//~ `New testbench
`timescale  1ns / 1ps
module tb_mux2x1_wrapper;
// mux2x1_wrapper Parameters
parameter PERIOD = 10;      //周期，涉及时钟信号时需要
// mux2x1_wrapper Inputs
reg   a = 0;
reg   b = 0;
reg   s = 0;

// mux2x1_wrapper Outputs
wire      F;

// mux2x1_wrapper Bidirs
                          //模拟时钟信号，没有的话就注释掉
//initial
//begin
//    forever #(PERIOD/2)  clk=~clk;
//end
                         //模拟复位信号，没有的话就注释掉
//initial
//begin
//    #(PERIOD*2) rst_n  =  1;
//end

mux2x1_wrapper  u_mux2x1_wrapper (
    .a(a),
    .b(b),
    .s(s),

    .F(F)
```

```
);

initial
begin
    //在这里生成你的测试语句
    $finish;
end

endmodule
```