

# 软硬件协同设计·第九次作业

## 实验1. 自动旋转式栅门 (Vivado)

### 1.1 实验题目

设计一款自动旋转式栅门去，其有两个状态，分别是锁住和解锁。存在个输入变量C，P，C表示刷卡成功，P表示通过成功。输出变量为当前状态。

### 1.2 实验建模

之后以两种形式来对栅门进行IP建模，分别是图形化建模IP和Verilog建模IP。

#### 1.2.1 图形化建模IP

首先需要构建IP的源文件，即仿真模块的Verilog代码，在本节中将通过使用该代码来对IP进行构成。代码如下所示，并记为turnstile\_FSM.v文件：

```
*** 栅门的功能代码 ***
```

之后打开Vivado HLS来进行图形化IP构成，首先点击Create Project，如图1.1所示。



图1.1 构建图

点击后，构建项目，这里我将名字取名为project\_1，需要注意的是路径中不能出现中文，如图1.2所示。

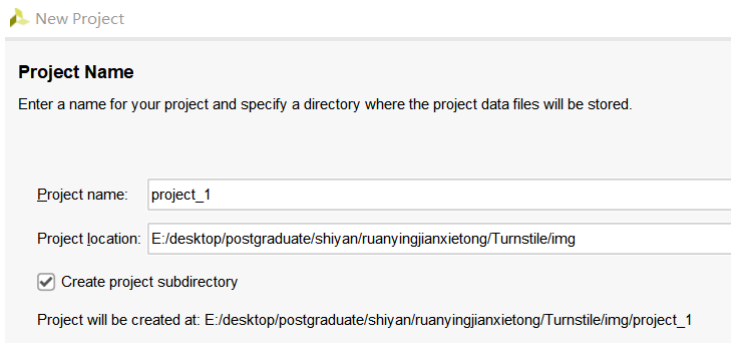


图1.2 项目创建图

之后一路默认即可，直到出现板子选取的页面，这里选取pynq-z2的款式，该板子对应的号码为xc7z020clg484-1，如图1.3所示。

Search: <input type="text" value="xc7z020clg484-1"/> (1 match)									
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Trans
xc7z020clg484-1	484	200	53200	106400	140	0	220	0	0

图1.3 板子选择图

完成上述步骤后，即可跳出项目汇总信息表，如图1.4所示。

### New Project Summary

- A new RTL project named 'project\_1' will be created.
- No source files or directories will be added. Use Add Sources to add them later.
- No constraints files will be added. Use Add Sources to add them later.
- The default part and product family for the new project:
  - Default Part: xc7z020clg484-1
  - Product: Zynq-7000
  - Family: Zynq-7000
  - Package: clg484
  - Speed Grade: -1

图1.4 项目汇总图

此时，已经进入到了Vivado HLS的主界面中，通过点击左侧Flow Navigator中的Add Sources来将最开始时的源文件添加进来，如图1.5所示。

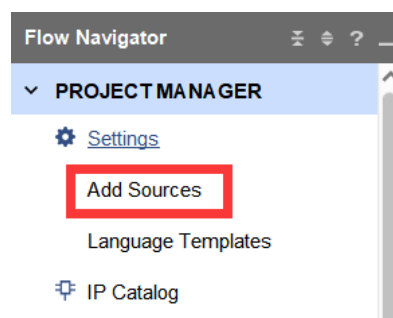


图1.5 源文件添加图

点击后可以跳出如图1.6所示的界面，选择默认的第二个选项即可。

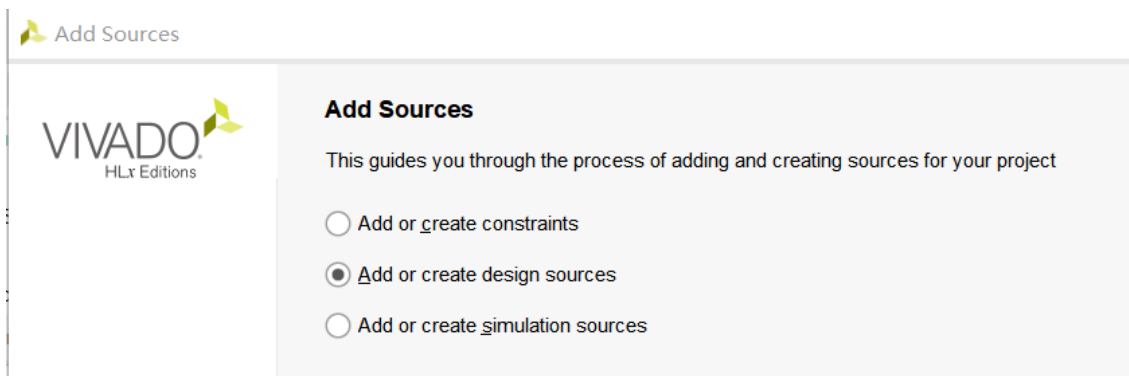


图1.6 构建图

将之前的verilog文件添加进来，需要注意的是，该文件的路径中同样不能出现中文，其效果如图1.7所示。



图1.7 源文件添加图

选择后，对应结果如图1.8所示。

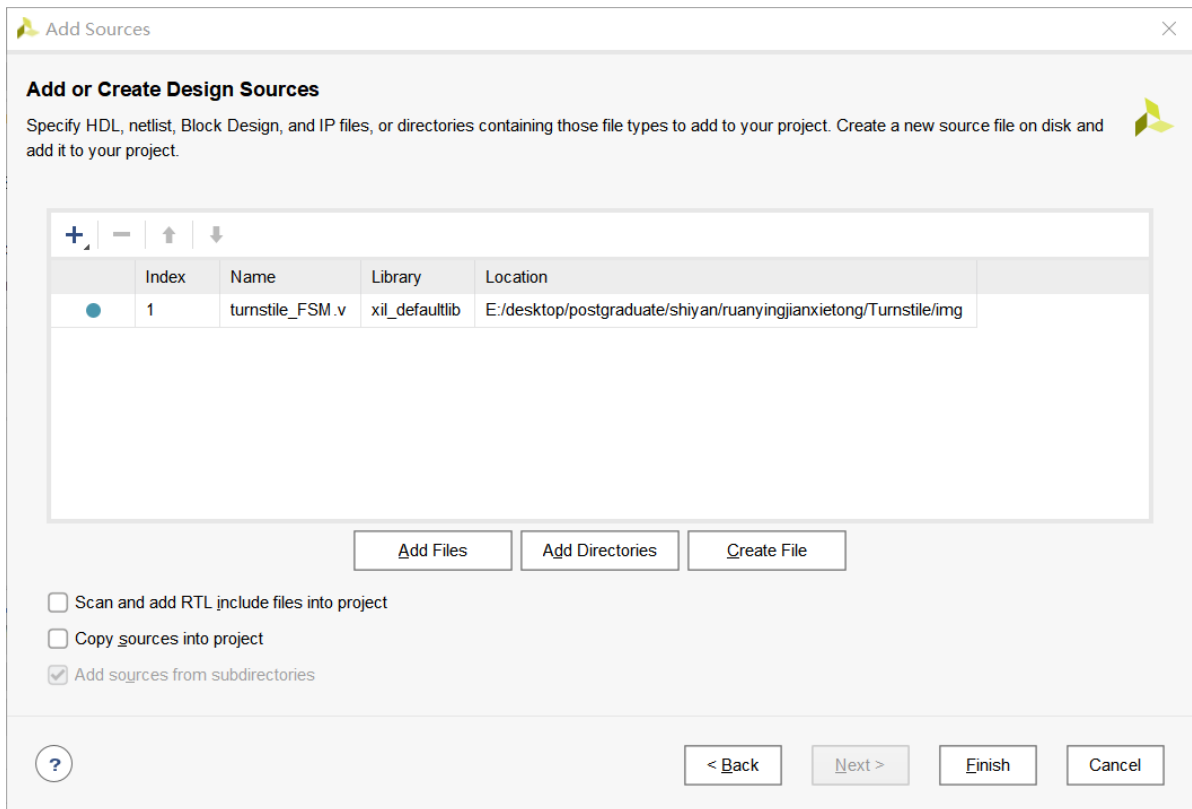


图1.8 添加结果图

关闭上述窗口，可以看到中间的Sources中已经有了该文件的有关信息，如图1.9所示。

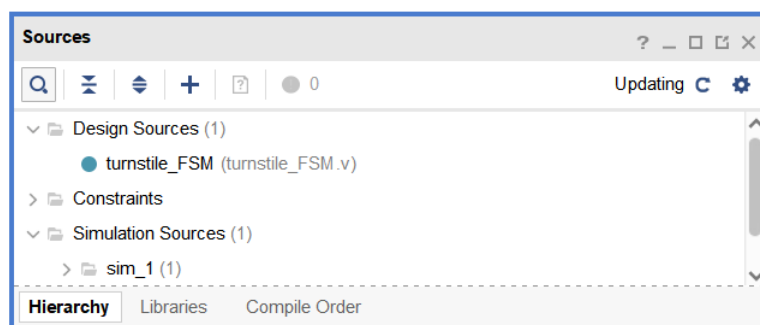


图1.9 sources显示图

此时，点击上侧栏中的Create and Package New IP，如图1.10所示。

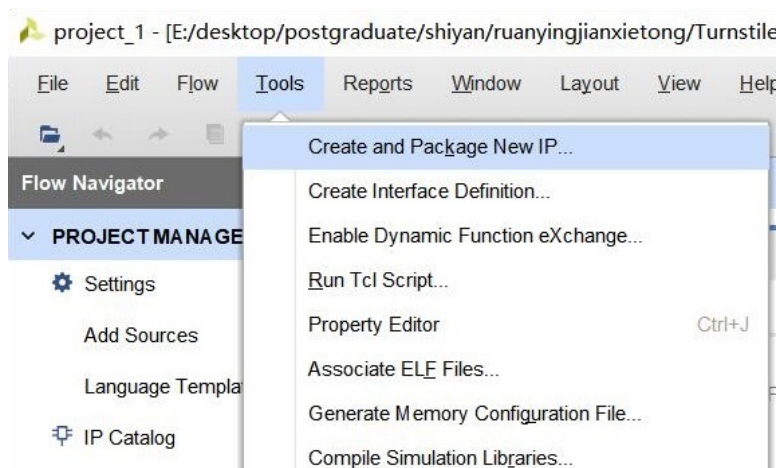


图1.10 位置图

点击后即可进入有关IP创建的界面，一路默认即可创建完成，点击Finish结束IP的创建，如图1.11所示。

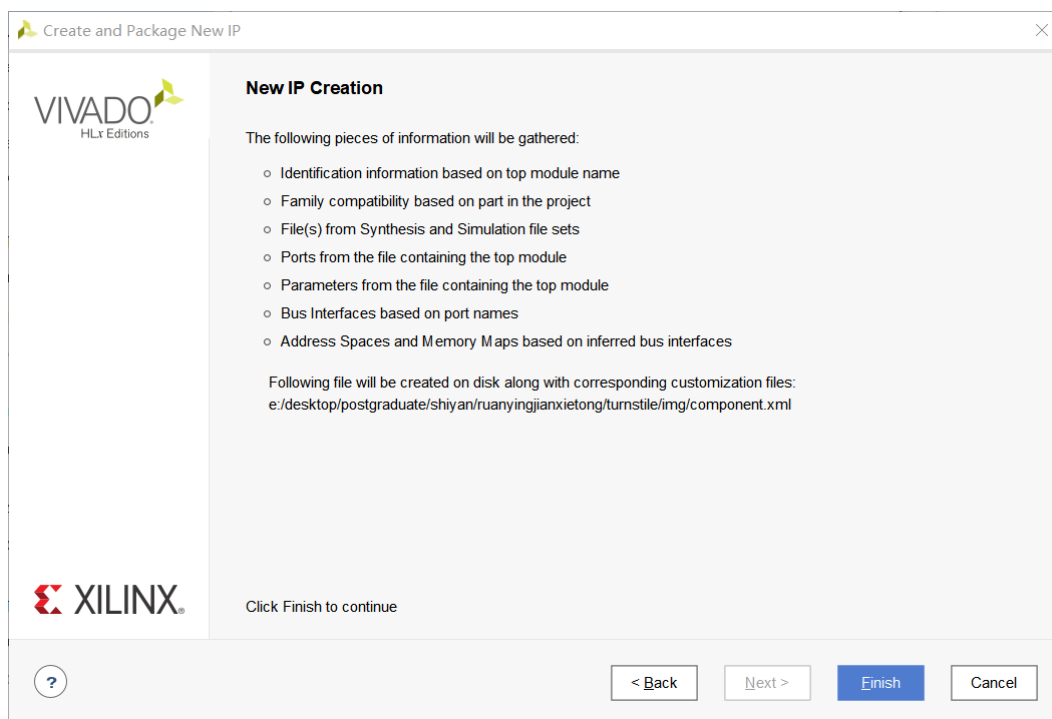


图1.11 IP创建图

此时右侧的工作区内已经有了相关的IP信息，点击Review and Package后，再点击Package IP即可将该IP进行打包，如图1.12所示。

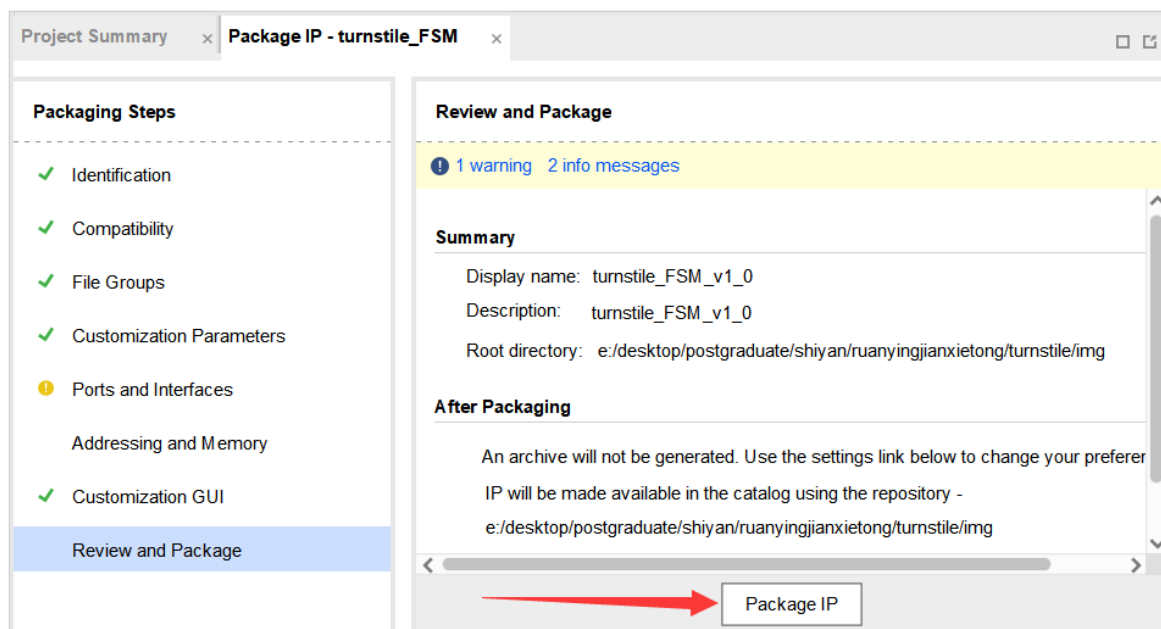


图1.12 IP打包图

点击后，若弹出如图1.13所示的打包成功界面，即创建成功。

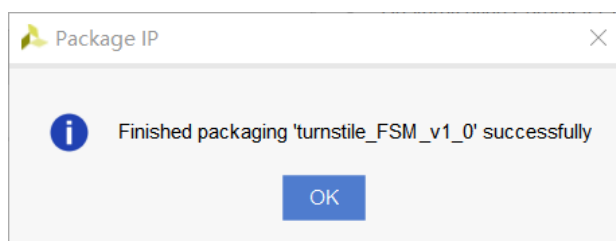


图1.13 IP打包结果图

此时点击左侧Flow Navigator下的Setting，进入IP下的Repository界面，在右侧将上面打包完成的IP包导入进来，路径为IP包对应verilog文件的路径，如图1.14所示。

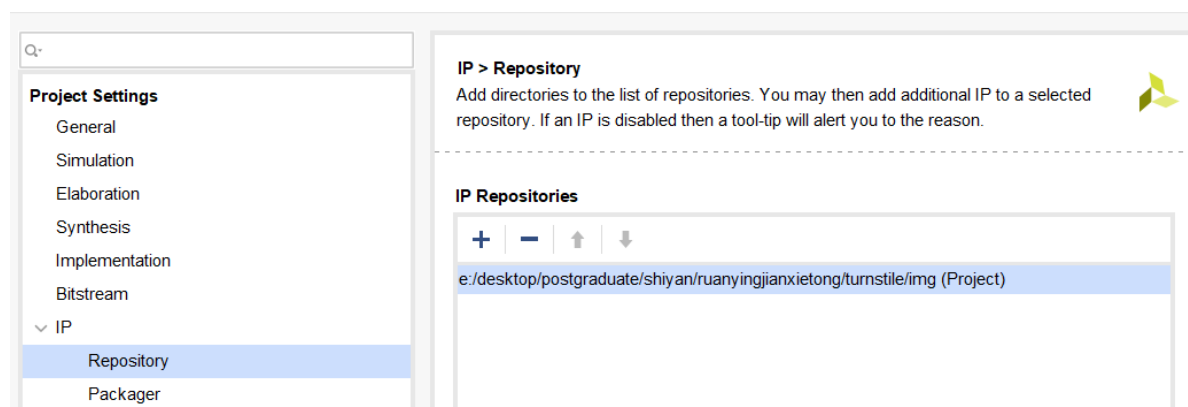


图1.14 IP包导入图

此时点击左侧Flow Navigator下的Create Block Design，如图1.15所示。

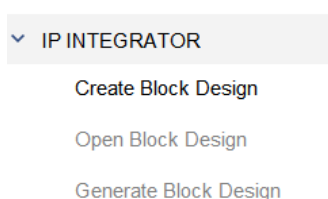


图1.15 Design位置图

点击后，可以跳出Block Design的设计区，此时还没有往里面加入任何的IP，如图1.16所示。

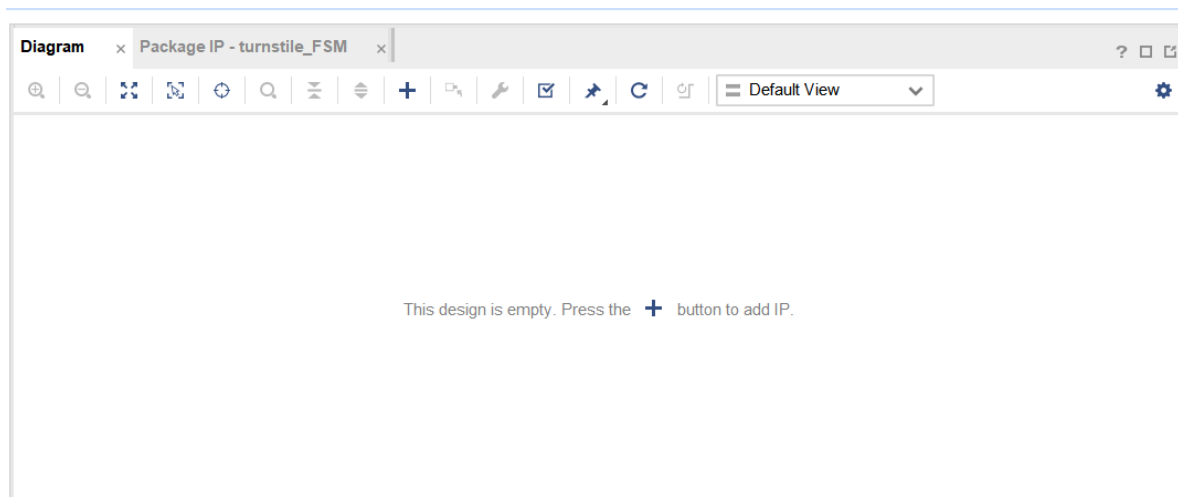


图1.16 设计区图

点击图中的“+”按钮，进行IP的添加，在搜索框中，搜索之前引入的IP包，如图1.17所示。

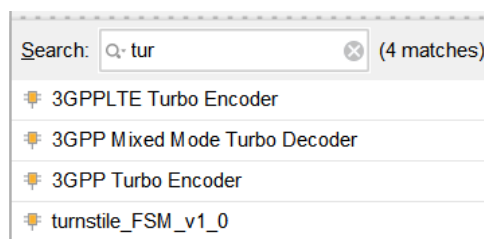


图1.17 IP打包结果图

选择后，即可导入成功，最终结果如图1.18所示。

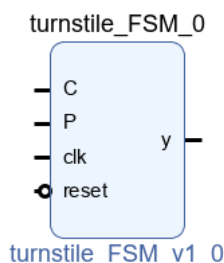


图1.18 图形化IP图

## 1.2.2 Verilog建模IP

之后通过Vitis hls来进行Verilog建模IP，首先要编写指定的C/C++功能代码。首先是turnstile.h文件：

```
*** 函数声明 ***
```

函数声明完成后，需要具体实现该函数，turnstile.cpp内容如下所示：

```
*** 函数功能实现 ***
```

最后还需要一个test bench文件，turnstile\_tb.cpp内容如下所示：

```
*** test bench文件 ***
```

上述代码完成后，进行RTL的生成、导出部分。首先打开Vitis hls，如图1.19所示。

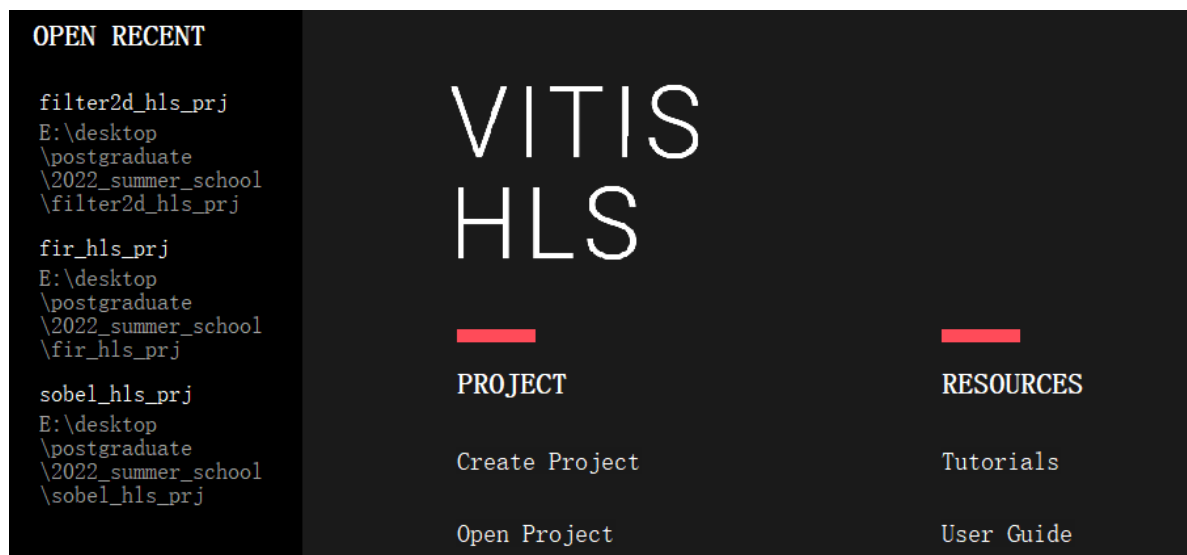


图1.19 Vitis图

点击Create Project后，将这个项目的名字取名为turnstile，存储在一个没有中文的路径上，如图1.20所示。

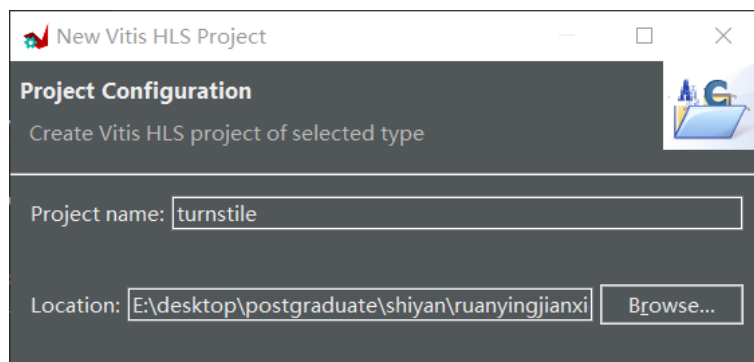


图1.20 Project设置图

之后将之前设计的.h文件和.cpp文件导入进去，并将顶层函数设置为之前写好的功能函数如图1.21所示。

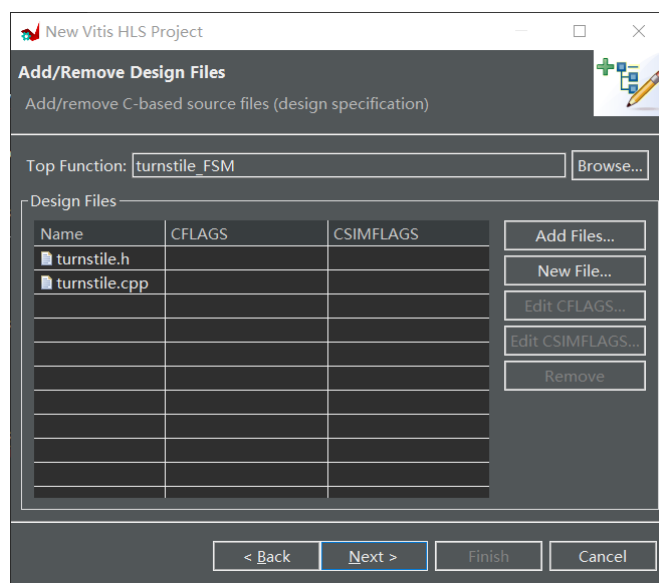


图1.21 C++文件导入图

之后将测试文件也导入进去，如图1.22所示。

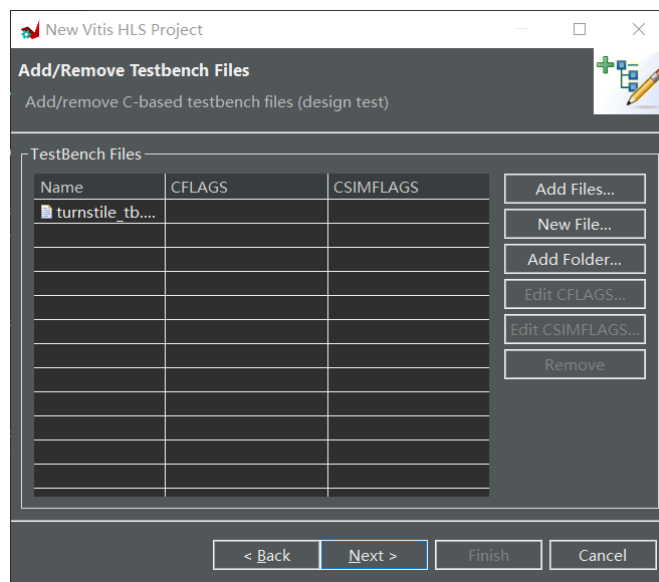


图1.22 tb文件导入图

之后一路默认确定，直到选择板卡界面，选择zynq-z2的板子作为结果，如图1.23所示。

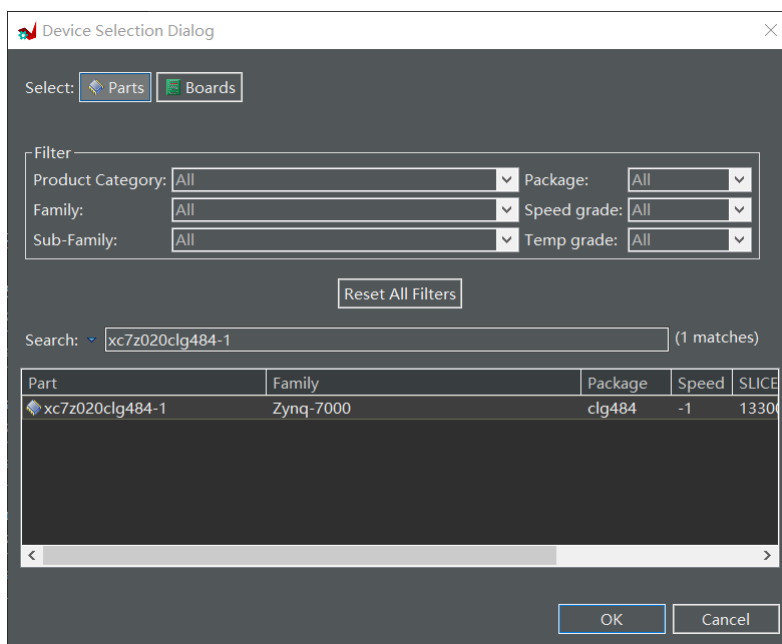


图1.23 板子选取图

选择完成后，板子选取界面上的结果如图1.24所示。

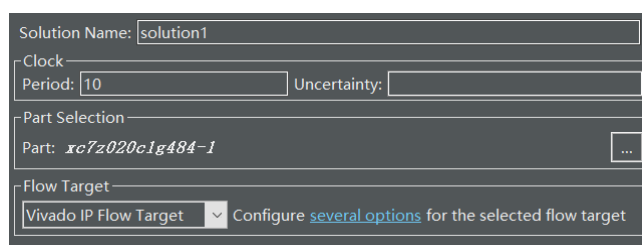


图1.24 板子结果图

之后进入Vitis HLS的主界面中，如图1.25所示。



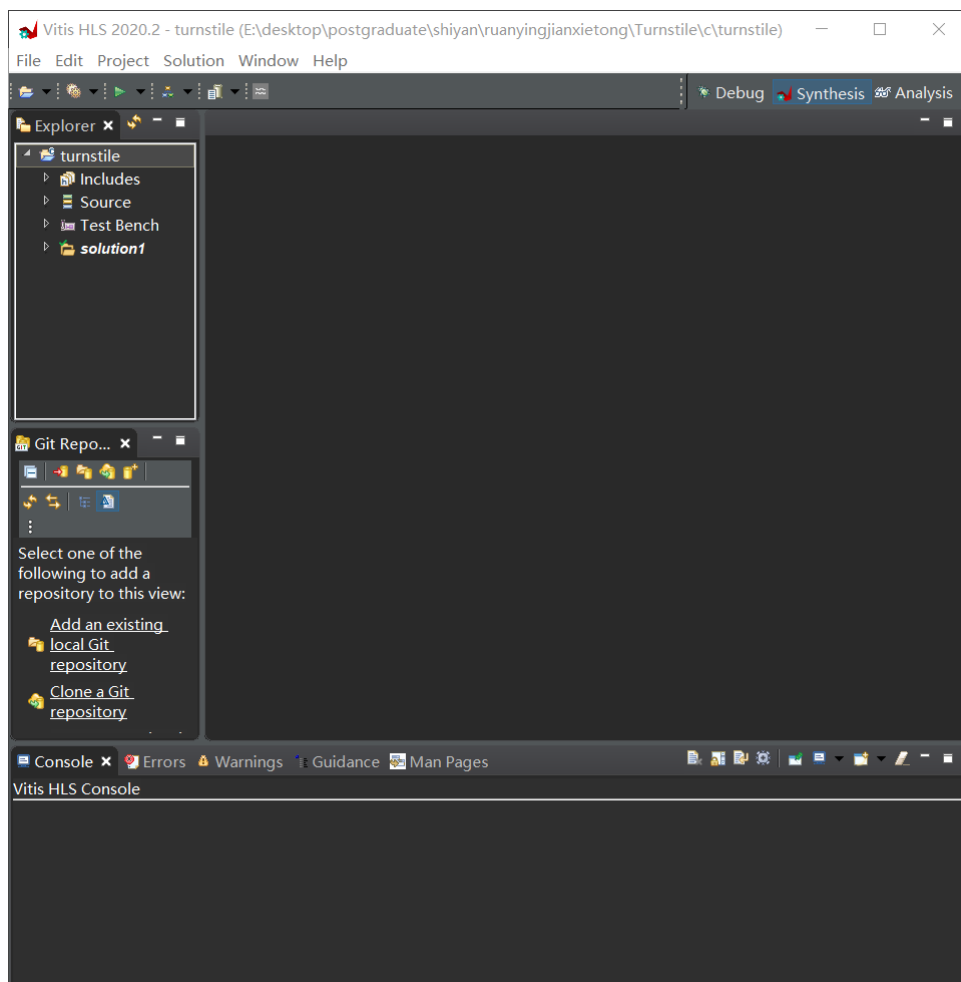


图1.25 Vitis HLS主界面图

之后点击上侧栏目Windows中的show view下的Flow Navigator选项，开启有关视图，如图1.26所示。

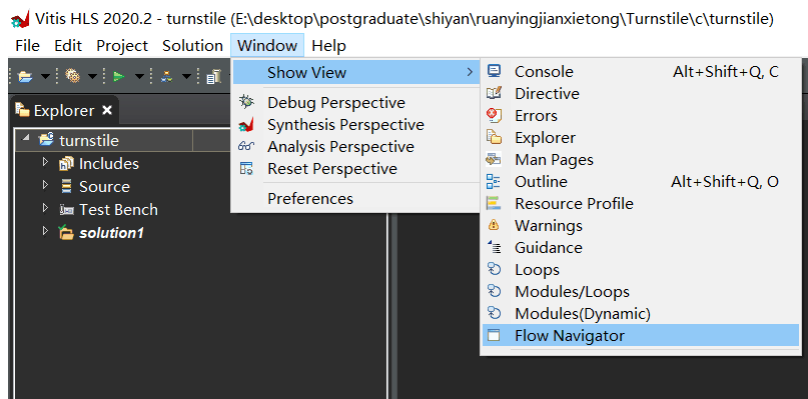


图1.26 Flow Navigator打开图

点开后，首先选择左侧栏目中的C Simulation，进行C仿真的测试，如图1.27所示。

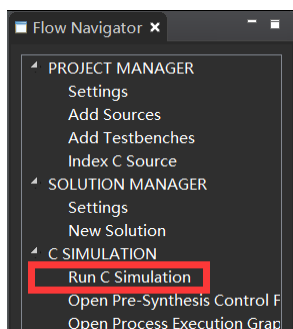


图1.27 C仿真位置图

点开，保持默认选项，选择OK后，其就会自动进行C仿真，当仿真结束后，其结果如图1.28所示。

```
INFO: [HLS 200-1510] Running: set_directive_top -name turnstile_FSM turnstile_FSM
INFO: [HLS 200-1510] Running: csim_design -quiet
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
Compiling ../../../../turnstile_tb.cpp in debug mode
Compiling ../../../../turnstile.cpp in debug mode
Generating csim.exe
0
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 1.57 seconds; current allocated memory: 162.593 M
Finished C simulation.
```

图1.28 C仿真结束图

C仿真结束后，还需要进行C综合，同样位于左侧栏目中，如图1.29所示。

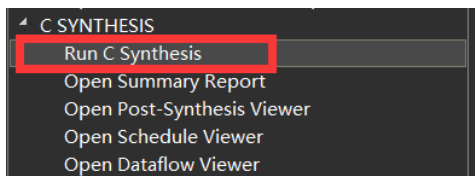


图1.29 C Synthesis位置图

点击后，保持默认选项，点击OK，执行结束后，其结果如图1.30所示。

```
INFO: [RTGEN 206-100] Finished creating RTL model for 'turnstile_FSM'.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 0.176 seconds; current allocated memory: 177.698
INFO: [HLS 200-111] Finished Generating all RTL models: CPU user time: 2 seconds. CPU system time: 0 seconds. Elapsed time: 1.648 seconds; current allocated memory: 1
INFO: [VHDL 208-304] Generating VHDL RTL for turnstile_FSM.
INFO: [VLOG 209-307] Generating Verilog RTL for turnstile_FSM.
INFO: [HLS 200-789] **** Estimated Fmax: 289.77 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 3 seconds. CPU system time: 0 seconds. Elapsed time: 8.603 seconds; current allocated memory: 183.91
INFO: [HLS 200-112] Total CPU user time: 7 seconds. Total CPU system time: 2 seconds. Total elapsed time: 11.999 seconds; peak allocated memory: 185.455 MB.
Finished C synthesis.
```

图1.30 C Synthesis结果图

同时可以在右侧的结果中看到有关的细节参数等内容，如图1.31所示。

Synthesis Summary Report of 'turnstile_FSM'			
General Information			
Date:	Thu Nov 17 15:39:32 2022	Solution:	solution1 (Vivado IP Flow Target)
Version:	2020.2 (Build 3064766 on Wed Nov 18 09:12:45 MST 2020)	Product family:	zynq
Project:	turnstile	Target device:	xc7z020-clg484-1
Timing Estimate			
Target	Estimated	Uncertainty	
10.00 ns	3.451 ns	2.70 ns	

图1.31 效率结果图

最后要进行的是C/RTL Cosimulation测试，其位置在上侧栏中的Solution下，如图1.32所示。

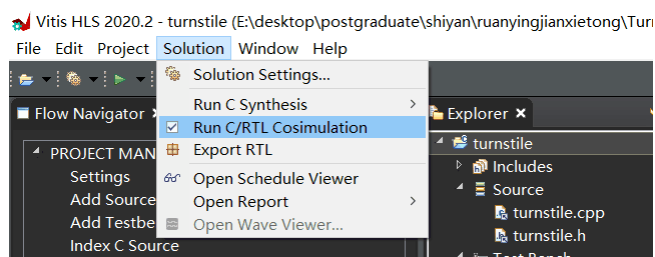


图1.31 Cosimulation位置图

点击后，保持默认的Verilog即可，如图1.32所示。

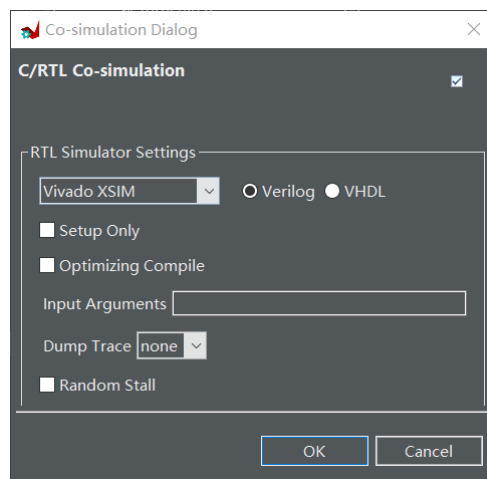


图1.32 Verilog类型图

点击OK后，等待执行结束，即可看到如图1.33所示的结果。

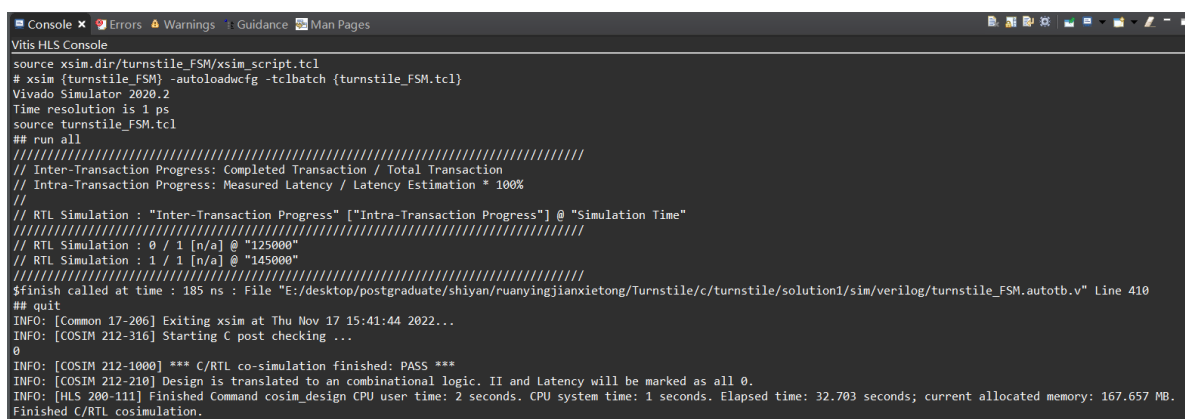


图1.33 Cosimulation结果图

完成上述所有测试后，即可进行RTL包的导出，位置还是在左侧的Flow Navigator中，如图1.34所示。

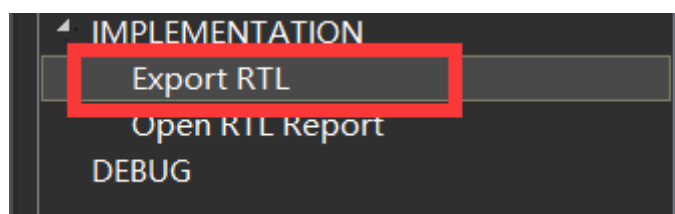


图1.34 导出位置图

点击后，在导出时保持默认的Verilog形式即可，如图1.35所示。

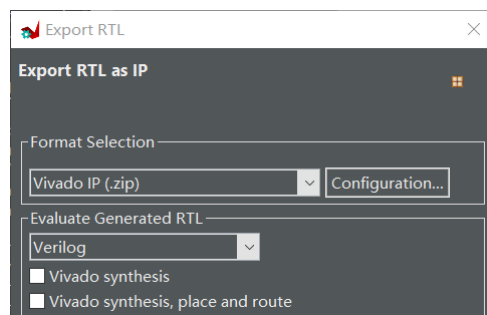


图1.35 导出类型图

点击OK后，其会自动进行导出，最后结果如图1.36所示。意味着Verilog版的IP包的生成。

```
***** Vivado v2020.2 (64-bit)
**** SW Build 3064766 on Wed Nov 18 09:12:45 MST 2020
**** IP Build 3064653 on Wed Nov 18 14:17:31 MST 2020
** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

source run_ippack.tcl -notrace
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'F:/Xilinx/Vivado/2020.2/data/ip'.
INFO: [Common 17-206] Exiting Vivado at Thu Nov 17 15:43:24 2022...
INFO: [HLS 200-802] Generated output file turnstile/solution1/impl/export.zip
INFO: [HLS 200-111] Finished Command export_design CPU user time: 1 seconds. CPU system time: 1 seconds. Elapsed time: 14.521 seconds;
Finished export RTL.
```

图1.36 RTL导出结果图

需要注意的是，如果是2020版本及之后的Vitis hls，都需要先解决软件中包含的千年虫问题，也就是需要打一个名为y2k22\_patch的补丁，否则在导出时会报错。有关问题的解决可以参考以下文章：[https://blog.csdn.net/weixin\\_42157664/article/details/122512603](https://blog.csdn.net/weixin_42157664/article/details/122512603)。

## 1.3 实验总结

# 实验2. 餐巾纸售货机 (Vivado)

## 2.1 实验题目

一款餐巾纸售货机，只接受 5 角和 1 元硬币，1 包餐巾纸价格为 1.5 元，没有找零功能。其中，Mealy型自动机如图2.1所示。

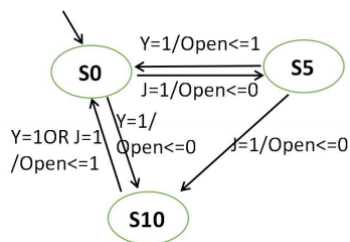


图2.1 图形化IP图

## 2.2 实验建模

之后以两种形式来对售货机进行IP建模，分别是图形化建模IP和Verilog建模IP。

### 2.2.1 图形化建模IP

与上面实验1中的步骤相似，首先先创建项目，这个实验中我创建的实验名为project\_2，如图2.2所示。

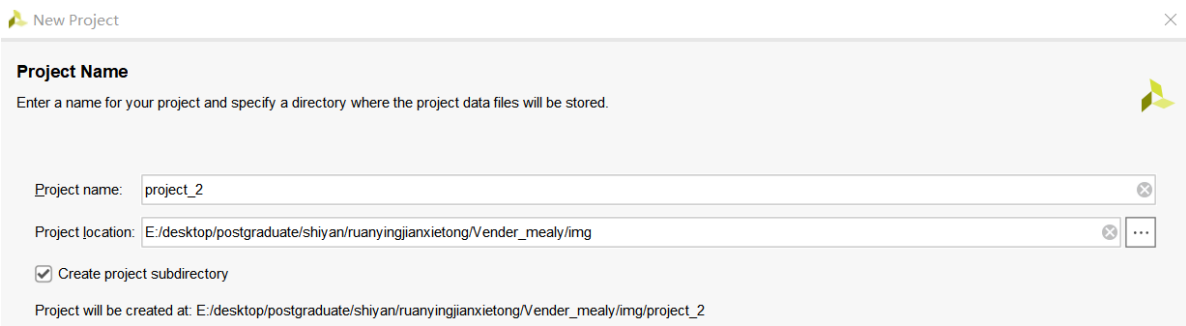


图2.2 新项目创建图

之后引入Mealy型售货机的verilog文件，如图2.3所示。

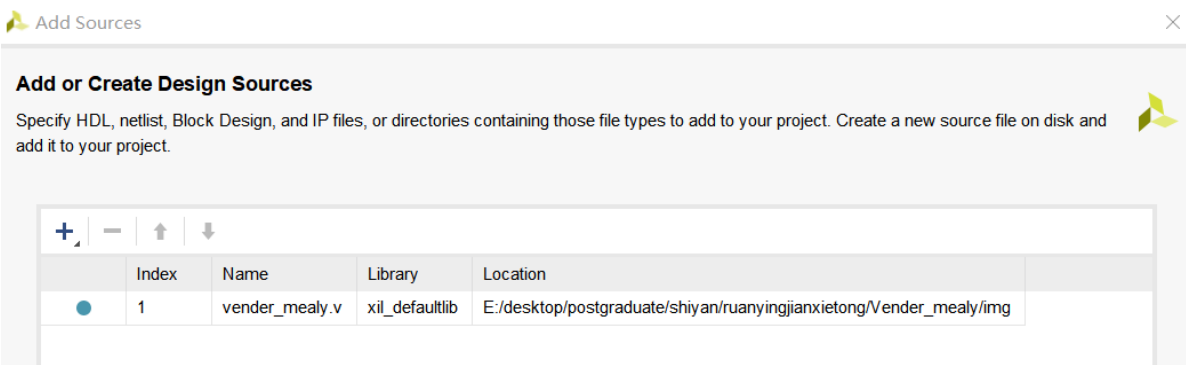


图2.3 verilog文件导入图

verilog文件的代码如下所示：

```
*** mealy型售货机功能代码 ***
```

之后按照之前实验1的步骤进行操作即可，打包售货机对应的IP如图2.4所示。

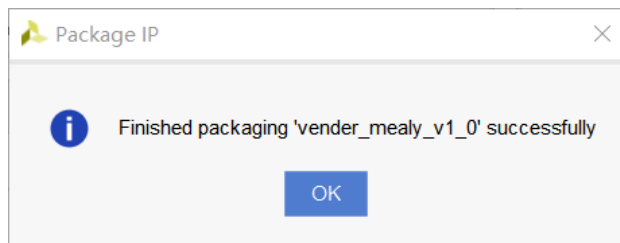


图2.4 IP包打包图

接着在Block Design中根据名称进行寻找，如图2.5所示。

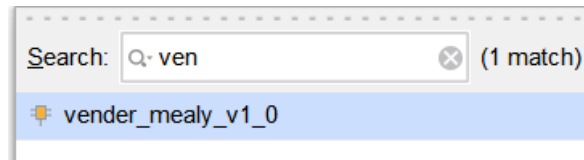


图2.5 IP包寻找图

最后的图形化建模IP结果如图2.6所示。

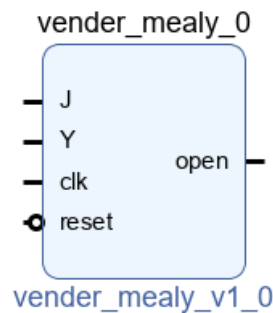


图2.6 图形化IP建模图

## 2.2.2 Verilog建模IP

之后通过Vitis hls来进行Verilog建模IP，首先要编写指定的C/C++功能代码。首先是vender.h文件：

```
*** 函数声明 ***
```

函数声明完成后，需要具体实现该函数，vender.cpp内容如下所示：

```
*** 函数功能实现 ***
```

最后还需要一个test bench文件，vender\_tb.cpp内容如下所示：

```
*** test bench文件 ***
```

之后与1.2.2中的步骤相似，首先创建本体项目，我将其取名为vender\_mealy，如图2.7所示。

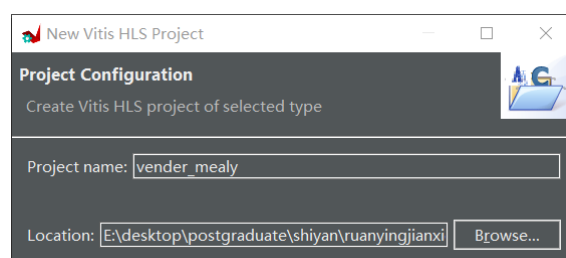


图2.7 Project创建图

之后将.h文件和.cpp文件导入，并选定好有关的顶层函数，如图2.8所示。

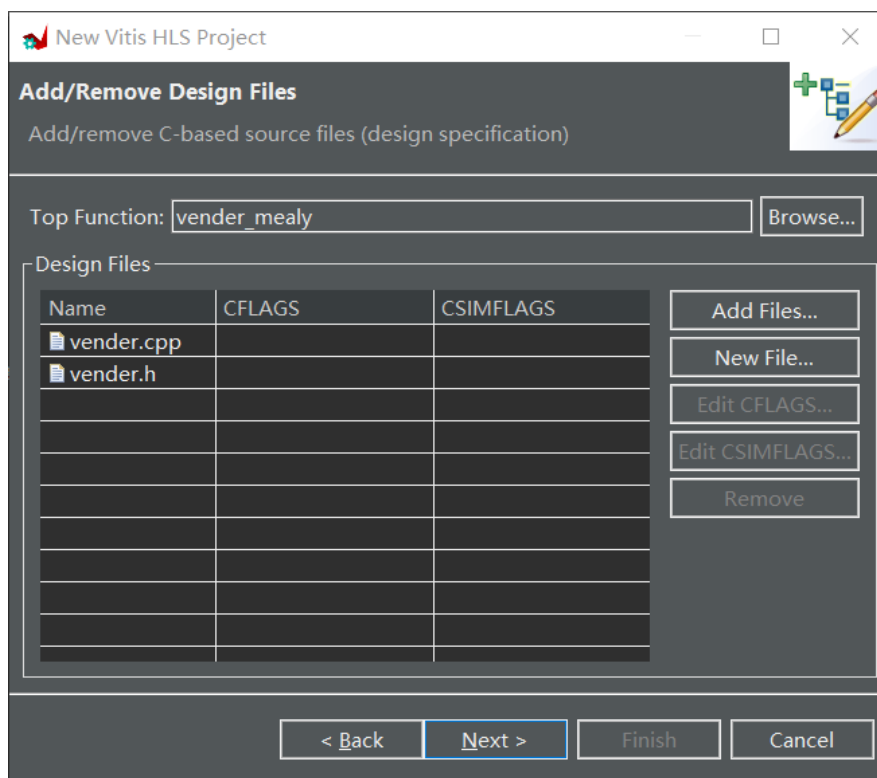


图2.8 文件导入图

此外还需要将测试文件导入，如图2.9所示。

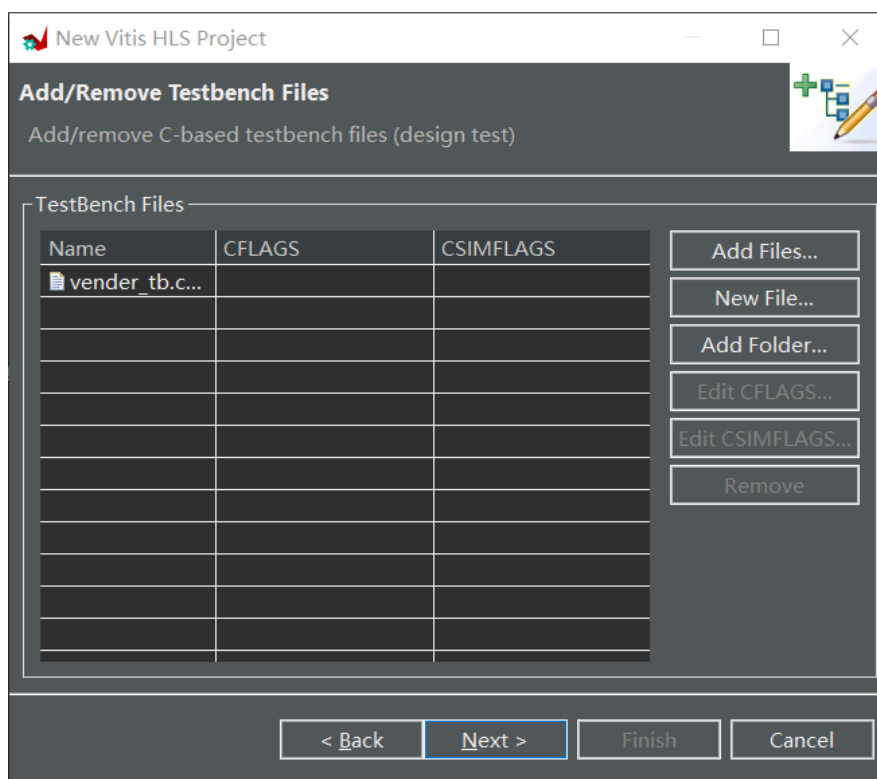


图2.9 tb文件导入图

项目创建后，首先进行C Simulation，其结果如图2.10所示。

```
Console x Errors Warnings Guidance Man Pages
Vitis HLS Console
INFO: [HLS 200-1510] Running: csim_design -quiet
INFO: [SIM 211-2] ***** CSIM start *****
INFO: [SIM 211-4] CSIM will launch GCC as the compiler.
Compiling ../../../../vender_tb.cpp in debug mode
Compiling ../../../../vender.cpp in debug mode
Generating csim.exe
0
INFO: [SIM 211-1] CSim done with 0 errors.
INFO: [SIM 211-3] ***** CSIM finish *****
INFO: [HLS 200-111] Finished Command csim_design CPU user time: 0 seconds. CPU system time: 1 seconds. Elapsed time: 1.736 seconds; current
Finished C simulation.
```

图2.10 C Simulation图

之后，进行C Synthesis，其结果如图2.11所示。

```
INFO: [RTGEN 206-100] Finished creating RTL model for 'vender_mealy'.
INFO: [HLS 200-111] Finished Creating RTL model: CPU user time: 0 seconds. CPU system time: 0 seconds. Elapsed time: 0.168 seconds; current
INFO: [HLS 200-111] Finished Generating all RTL models: CPU user time: 1 seconds. CPU system time: 0 seconds. Elapsed time: 1.452 seconds; c
INFO: [VHDL 208-304] Generating VHDL RTL for vender_mealy.
INFO: [VLOG 209-307] Generating Verilog RTL for vender_mealy.
INFO: [HLS 200-789] **** Estimated Fmax: 246.24 MHz
INFO: [HLS 200-111] Finished Command csynth_design CPU user time: 2 seconds. CPU system time: 1 seconds. Elapsed time: 7.923 seconds; curren
INFO: [HLS 200-112] Total CPU user time: 7 seconds. Total CPU system time: 2 seconds. Total elapsed time: 11.116 seconds; peak allocated mem
Finished C synthesis.
```

图2.11 C Synthesis图

接下来，进行C/RTL Cosimulation，其结果如图2.12所示。

```
Console x Errors Warnings Guidance Man Pages
Vitis HLS Console
// RTL Simulation : 3 / 3 [n/a] @ "165000"
$finish called at time : 205 ns : File "E:/desktop/postgraduate/shiyan/ruanyingjianxietong/Vender_mealy/c/vender_mealy/solution1/sim/verilog
## quit
INFO: [Common 17-206] Exiting xsim at Thu Nov 17 17:24:11 2022...
INFO: [COSIM 212-316] Starting C post checking ...
0
INFO: [COSIM 212-1000] *** C/RTL co-simulation finished: PASS ***
INFO: [COSIM 212-210] Design is translated to an combinational logic. II and Latency will be marked as all 0.
INFO: [HLS 200-111] Finished Command cosim_design CPU user time: 1 seconds. CPU system time: 1 seconds. Elapsed time: 27.641 seconds; curren
Finished C/RTL cosimulation.
```

图2.12 C/RTL Cosimulation图

其对应的功耗结果，如图2.13所示。


Synthesis Summary Report of 'vender_mealy'			
General Information			
Date:	Thu Nov 17 17:23:09 2022	Solution:	solution1 (Vivado IP Flow Target)
Version:	2020.2 (Build 3064766 on Wed Nov 18 09:12:45 MST 2020)	Product family:	zynq
Project:	vender_mealy	Target device:	xc7z020-clg484-1
Timing Estimate			
			
Target	Estimated	Uncertainty	
10.00 ns	4.061 ns	2.70 ns	

图2.13 Cosimulation对应功耗图

最后，对其进行RTL导出，对应结果如图2.14所示。意味着Verilog版的IP包的生成。

```
Console x Errors Warnings Guidance Man Pages
Vitis HLS Console
**** IP Build 3064653 on Wed Nov 18 14:17:31 MST 2020
** Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.

source run_ippack.tcl -notrace
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository 'F:/Xilinx/Vivado/2020.2/data/ip'.
INFO: [Common 17-206] Exiting Vivado at Thu Nov 17 17:24:49 2022...
INFO: [HLS 200-802] Generated output file vender_mealy/solution1/impl/export.zip
INFO: [HLS 200-111] Finished Command export_design CPU user time: 2 seconds. CPU system time: 1 seconds. Elapsed time: 14.012 seconds; curren
Finished export RTL.
```

图2.14 RTL导出图



### 2.3 实验总结