

Vivado SoC 软硬件开发流程示例

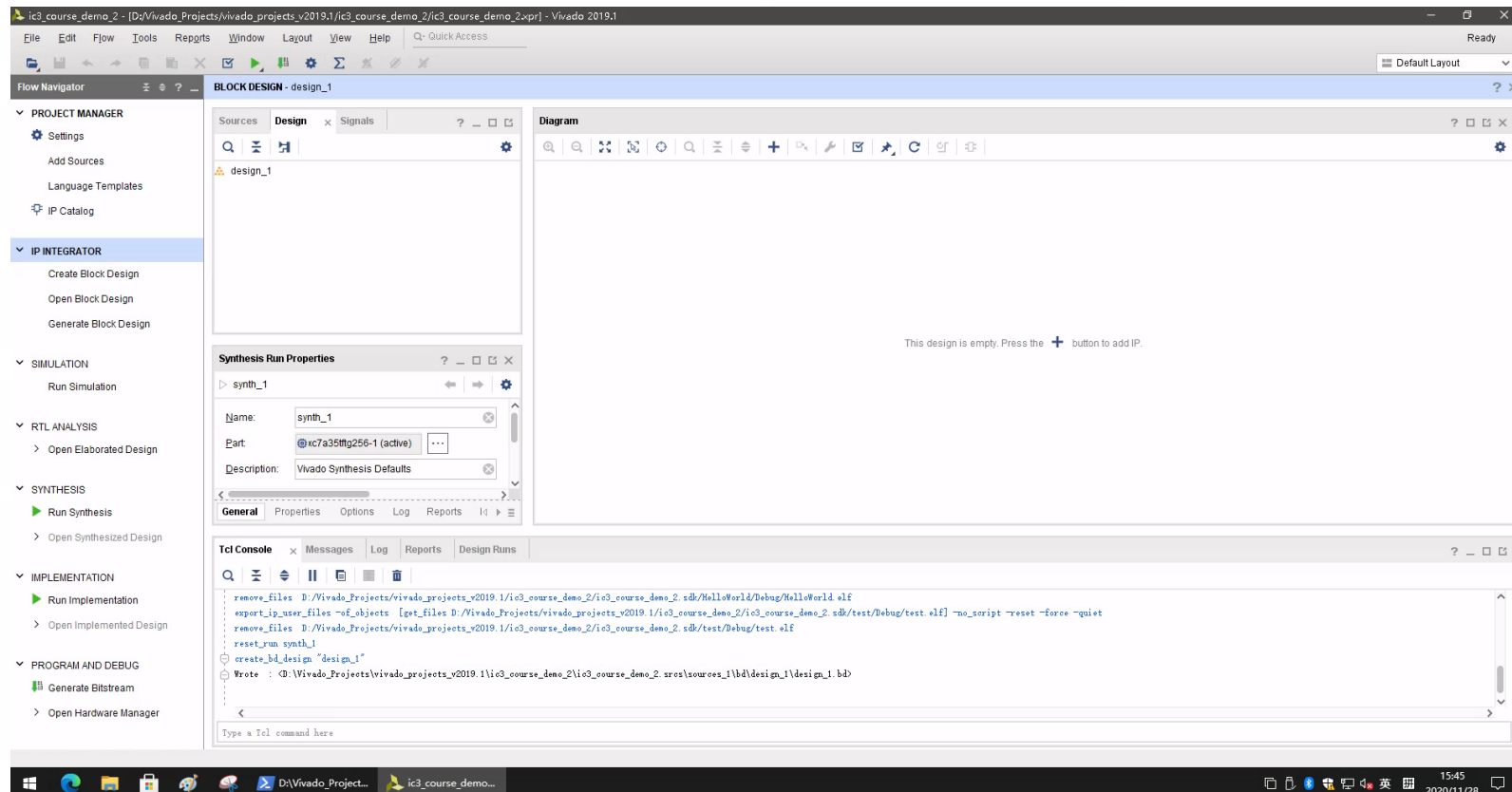
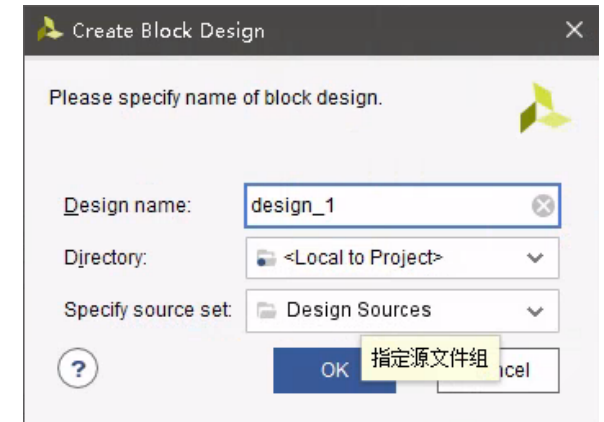
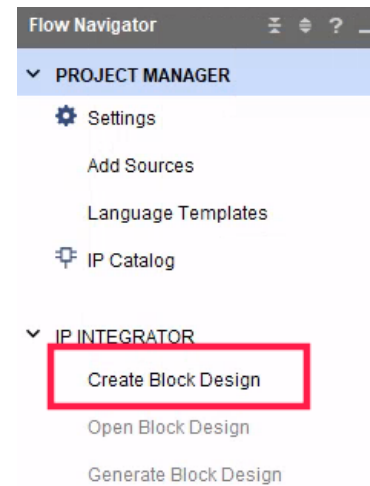
以及：IC 综合实验3 SoC 实验部分的实验要求

pt27@live.cn

- 使用 Vivado 2019.1
- 本文内容：
 1. 创建 Block Design ➡
 2. 生成 HDL、bitstream 以及导出硬件 ➡
 3. Xilinx SDK 的使用 ➡
 4. SoC 实验部分的实验要求 ➡

创建 Block Design

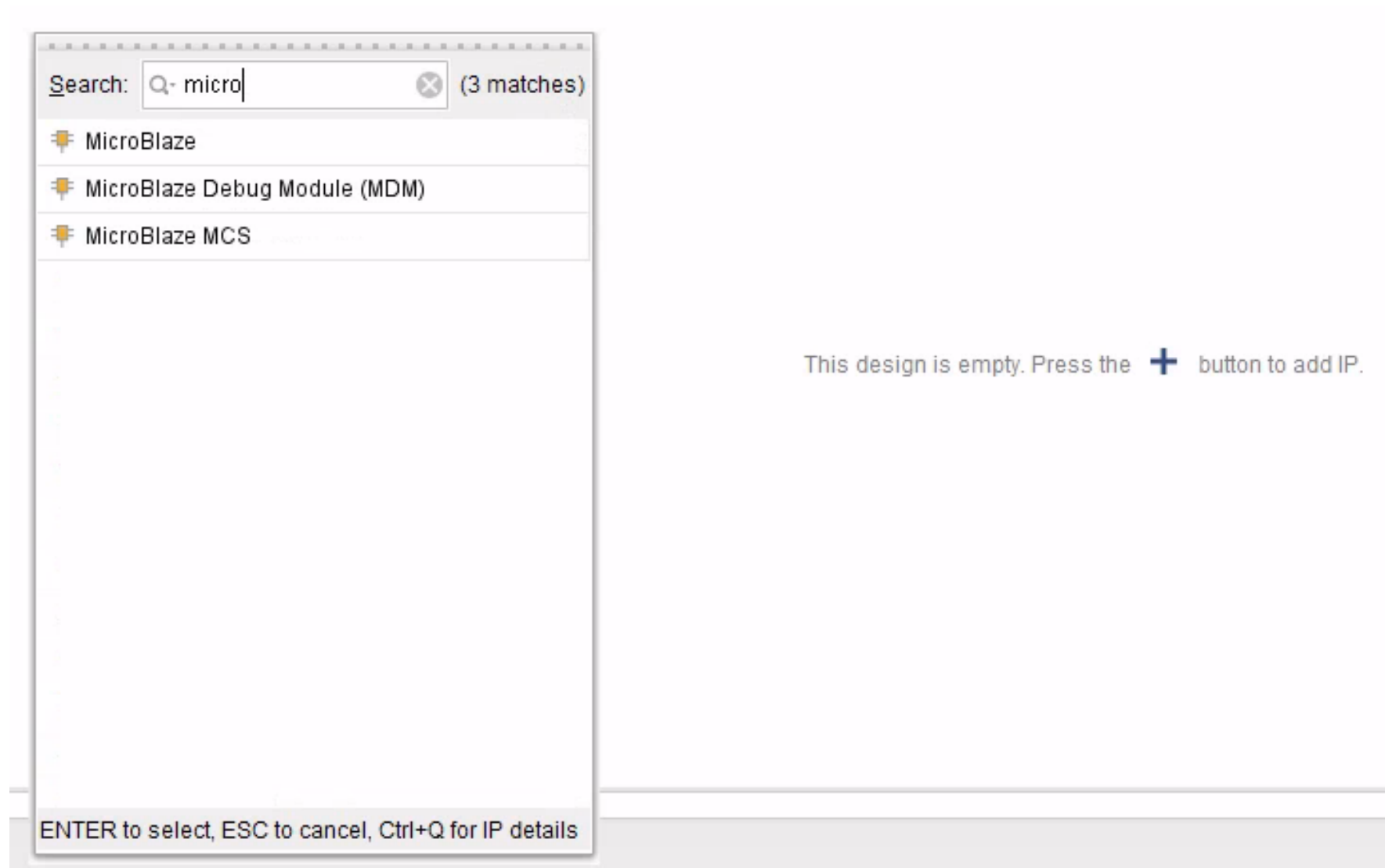
1. 创建 Vivado 项目 (略)
2. 创建 Block Design



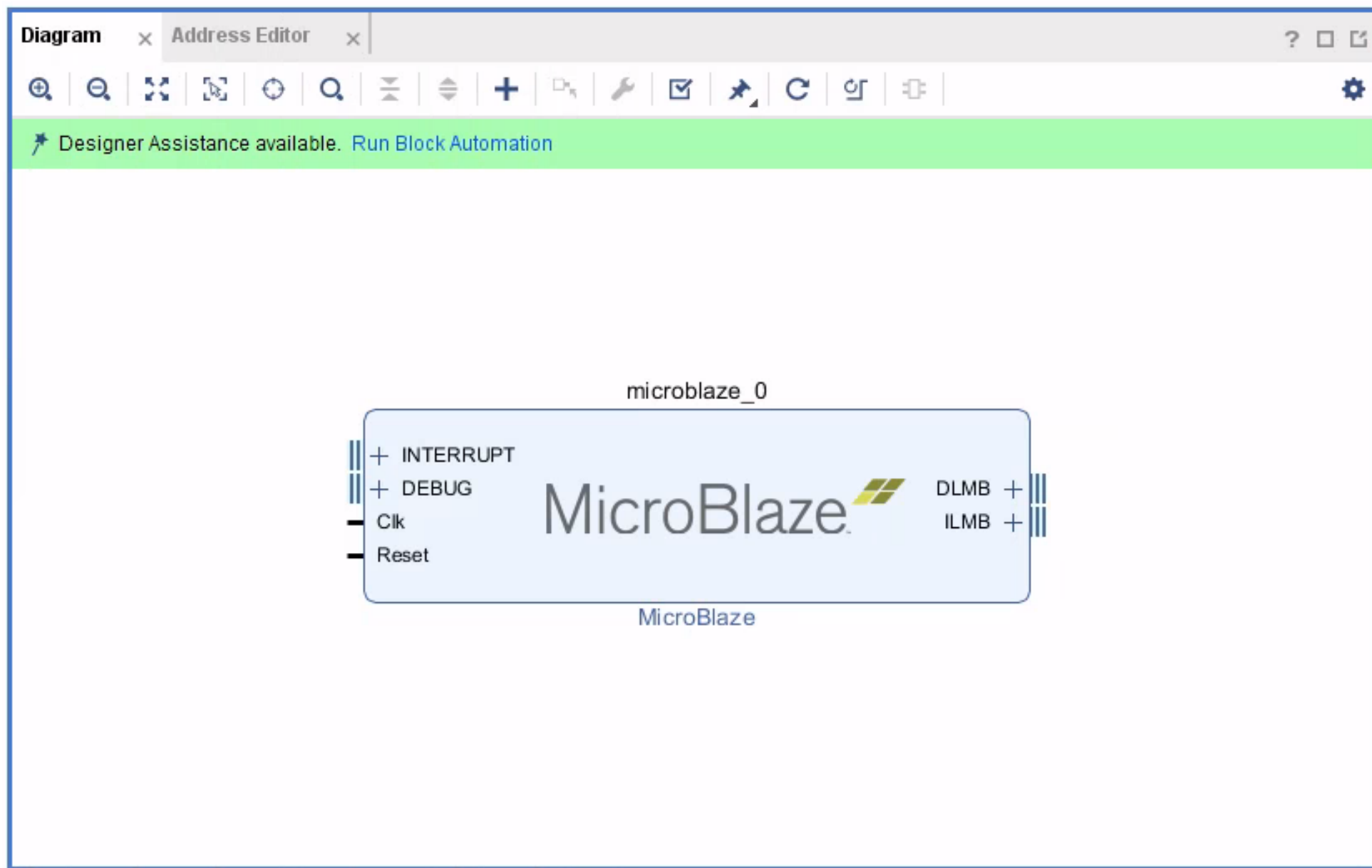
创建 Block Design

添加电路模块的方法：

单击 “+”，搜索模块的名字，添加模块

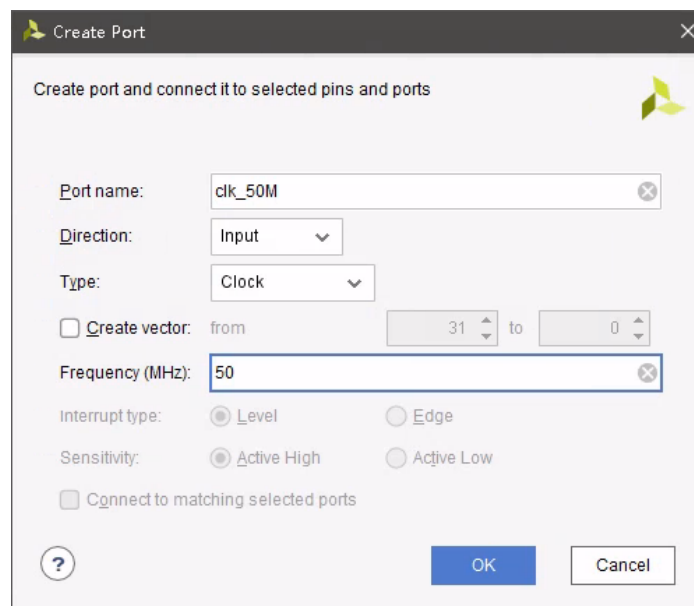
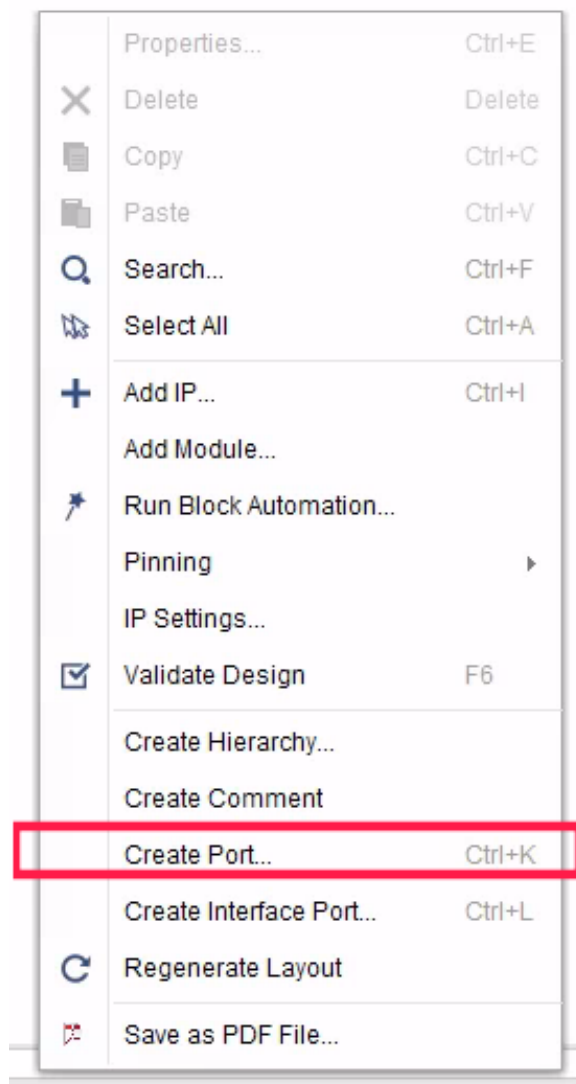


创建 Block Design



如图，添加 MicroBlaze 核

创建 Block Design

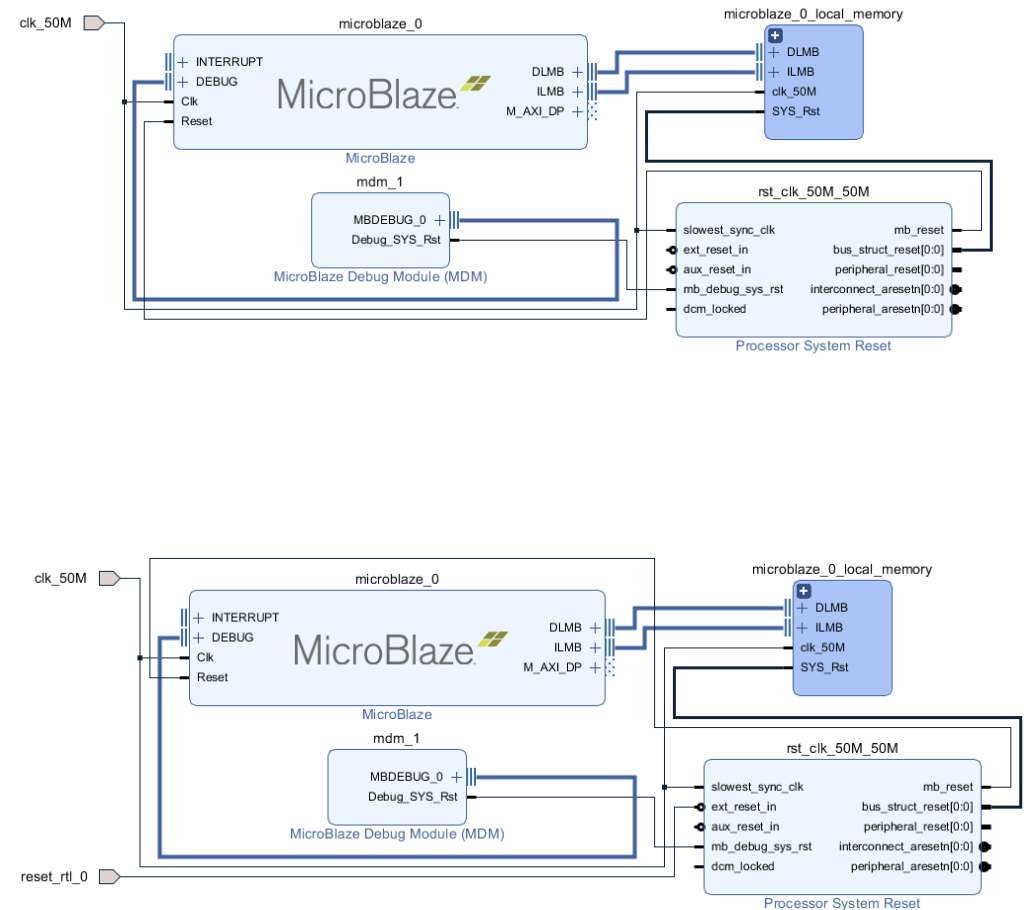
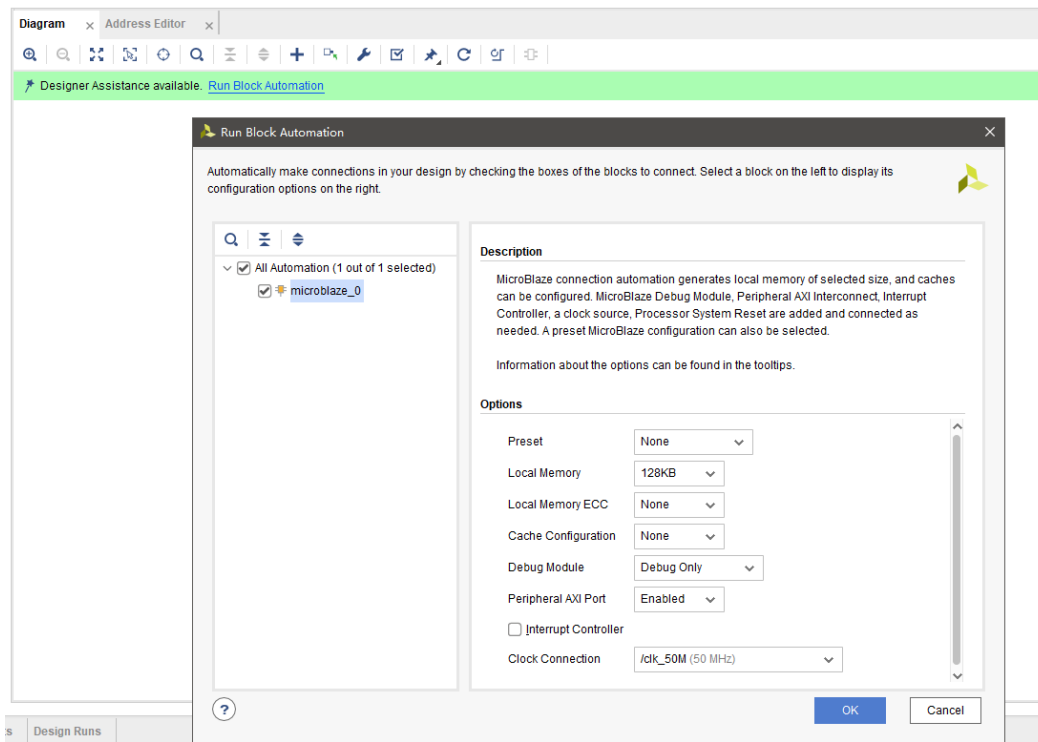


clk_50M



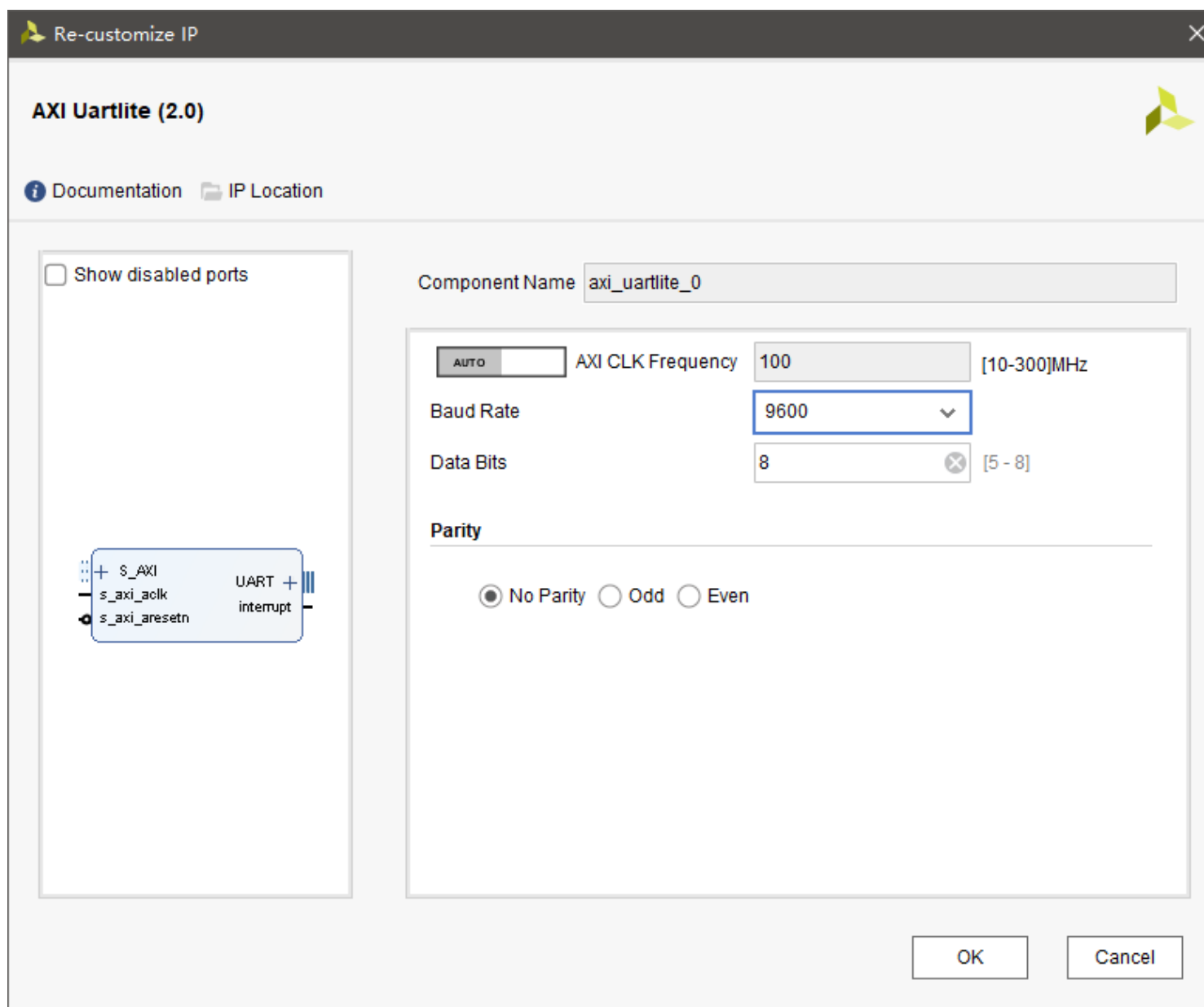
右键单击空白处（或者 Ctrl + K）添加端口，如图，添加时钟端口，设置频率为 50 MHz

创建 Block Design



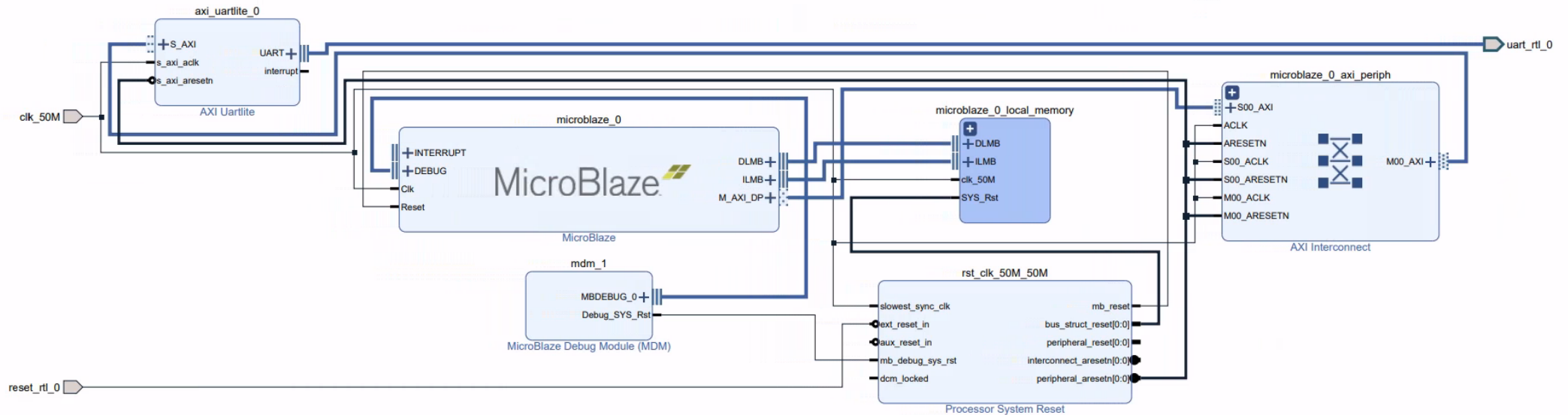
- 点击 “Run Block Automation”，配置 MicroBlaze Core
- 点击 “Run Connection Automation”，自动生成复位引脚 “reset_rtl_0”

创建 Block Design



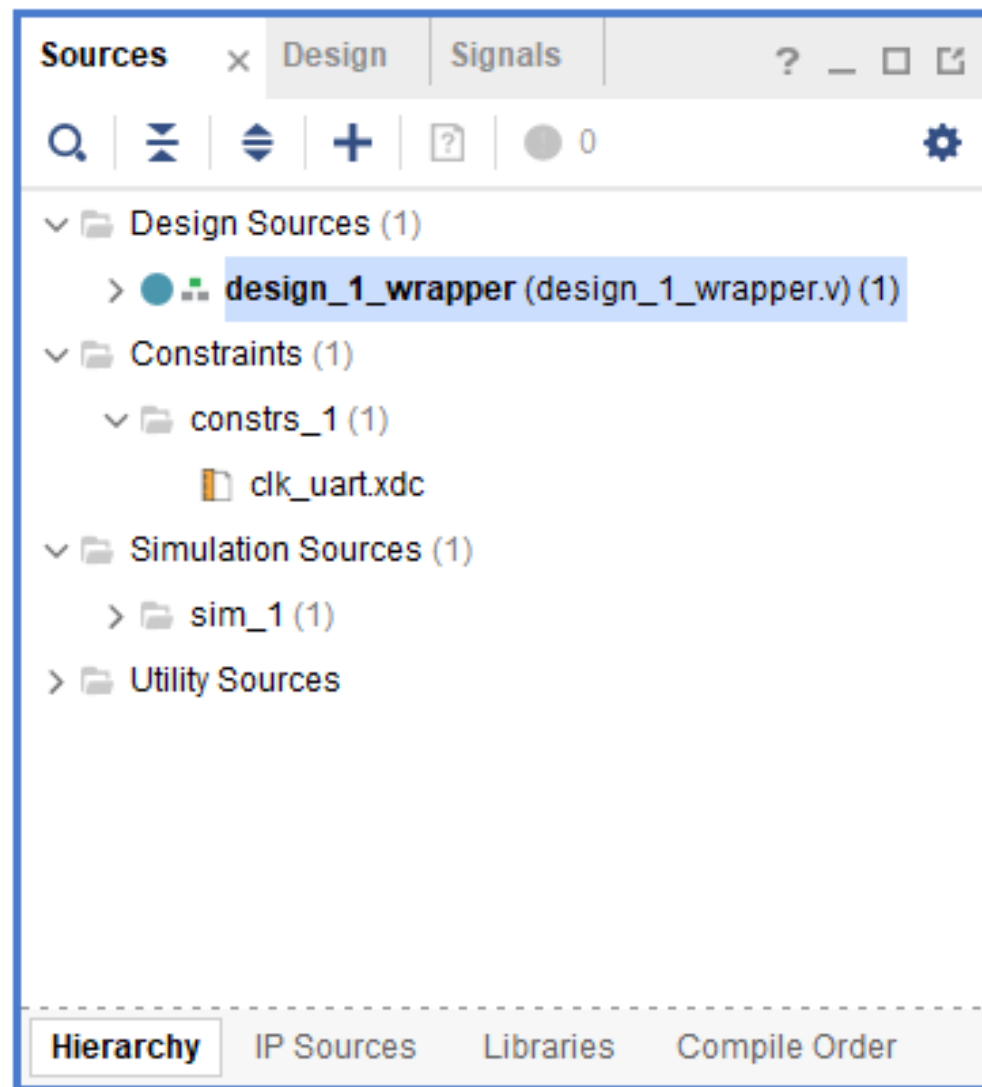
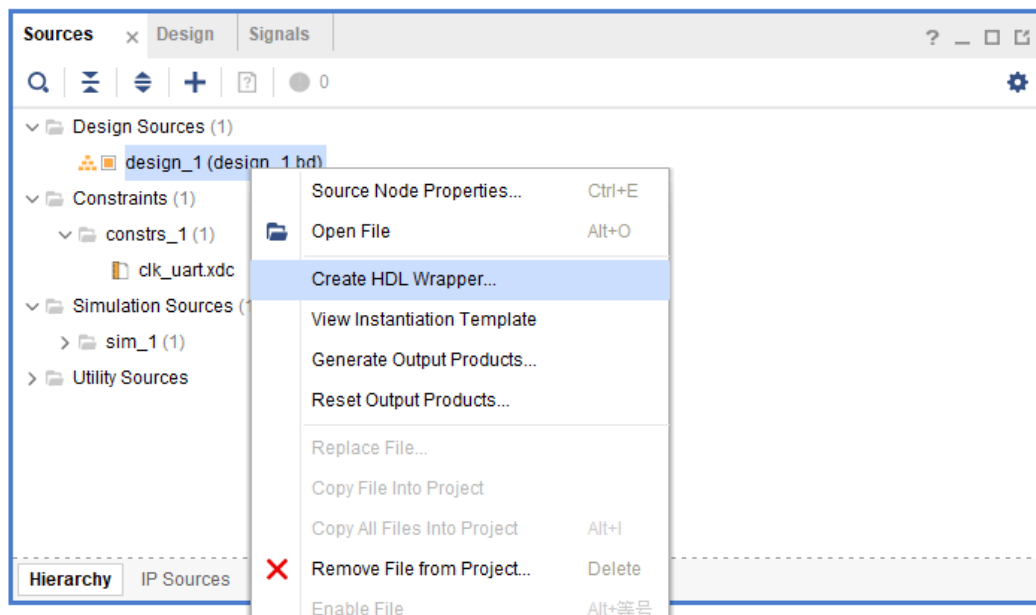
添加 AXI Uartlite 模块，使用默认配置

创建 Block Design



Block Design 总体框图

导出可综合 HDL



生成可以综合的 HDL，生成的是 Verilog 代码，这个代码用于综合生成电路。

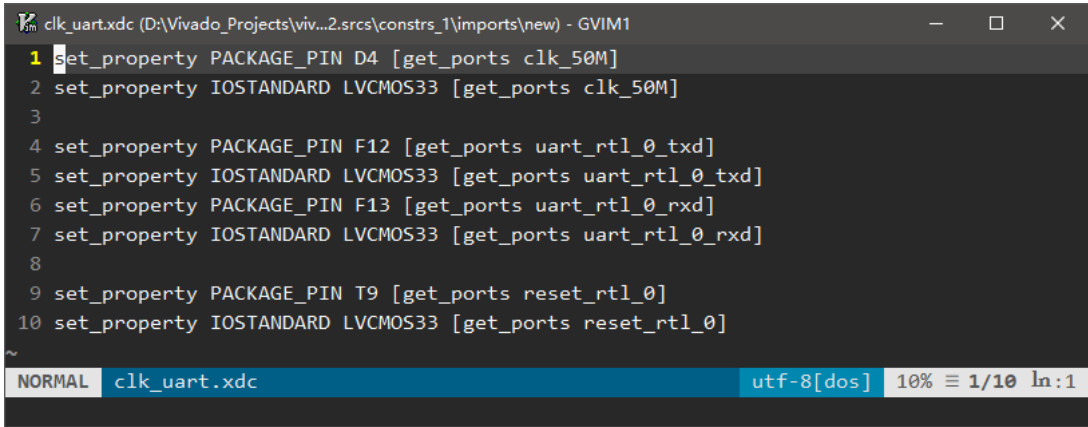
编写约束文件

生成的文件：

design_1_wrapper.v

对应的约束文件内容：

```
10 `timescale 1 ps / 1 ps
11
12 module design_1_wrapper
13     (clk_50M,
14      reset_rtl_0,
15      uart_rtl_0_rxd,
16      uart_rtl_0_txd);
17     input clk_50M;
18     input reset_rtl_0;
19     input uart_rtl_0_rxd;
20     output uart_rtl_0_txd;
21
22     wire clk_50M;
23     wire reset_rtl_0;
24     wire uart_rtl_0_rxd;
25     wire uart_rtl_0_txd;
26
27     design_1 design_1_i
28         (.clk_50M(clk_50M),
29          .reset_rtl_0(reset_rtl_0),
30          .uart_rtl_0_rxd(uart_rtl_0_rxd),
31          .uart_rtl_0_txd(uart_rtl_0_txd));
32 endmodule
```



```
clk_uart.xdc (D:\Vivado_Projects\viv...2.srcs\constrs_1\imports\new) - GVIM1
1 set_property PACKAGE_PIN D4 [get_ports clk_50M]
2 set_property IOSTANDARD LVCMOS33 [get_ports clk_50M]
3
4 set_property PACKAGE_PIN F12 [get_ports uart_rtl_0_txd]
5 set_property IOSTANDARD LVCMOS33 [get_ports uart_rtl_0_txd]
6 set_property PACKAGE_PIN F13 [get_ports uart_rtl_0_rxd]
7 set_property IOSTANDARD LVCMOS33 [get_ports uart_rtl_0_rxd]
8
9 set_property PACKAGE_PIN T9 [get_ports reset_rtl_0]
10 set_property IOSTANDARD LVCMOS33 [get_ports reset_rtl_0]
~
NORMAL clk_uart.xdc utf-8[dos] 10% 1/10 ln:1
```

约束文件编写好后，就可以生成 bitstream（XSDK 会用到）。

导出文件、启动 XSDK

导出硬件

菜单栏:

File

> Export

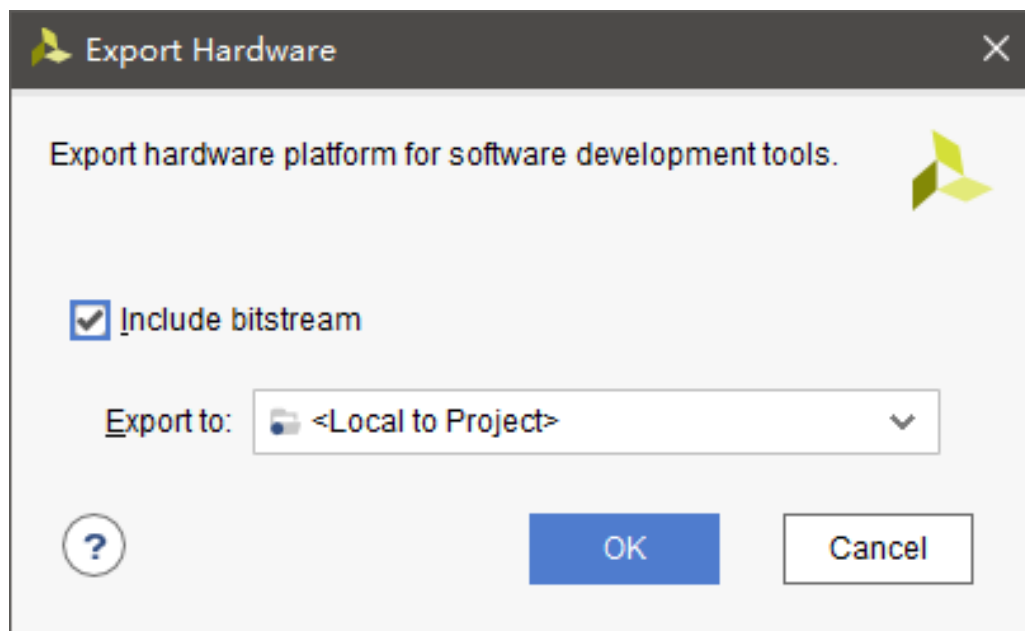
> Export Hardware

启动 XSDK

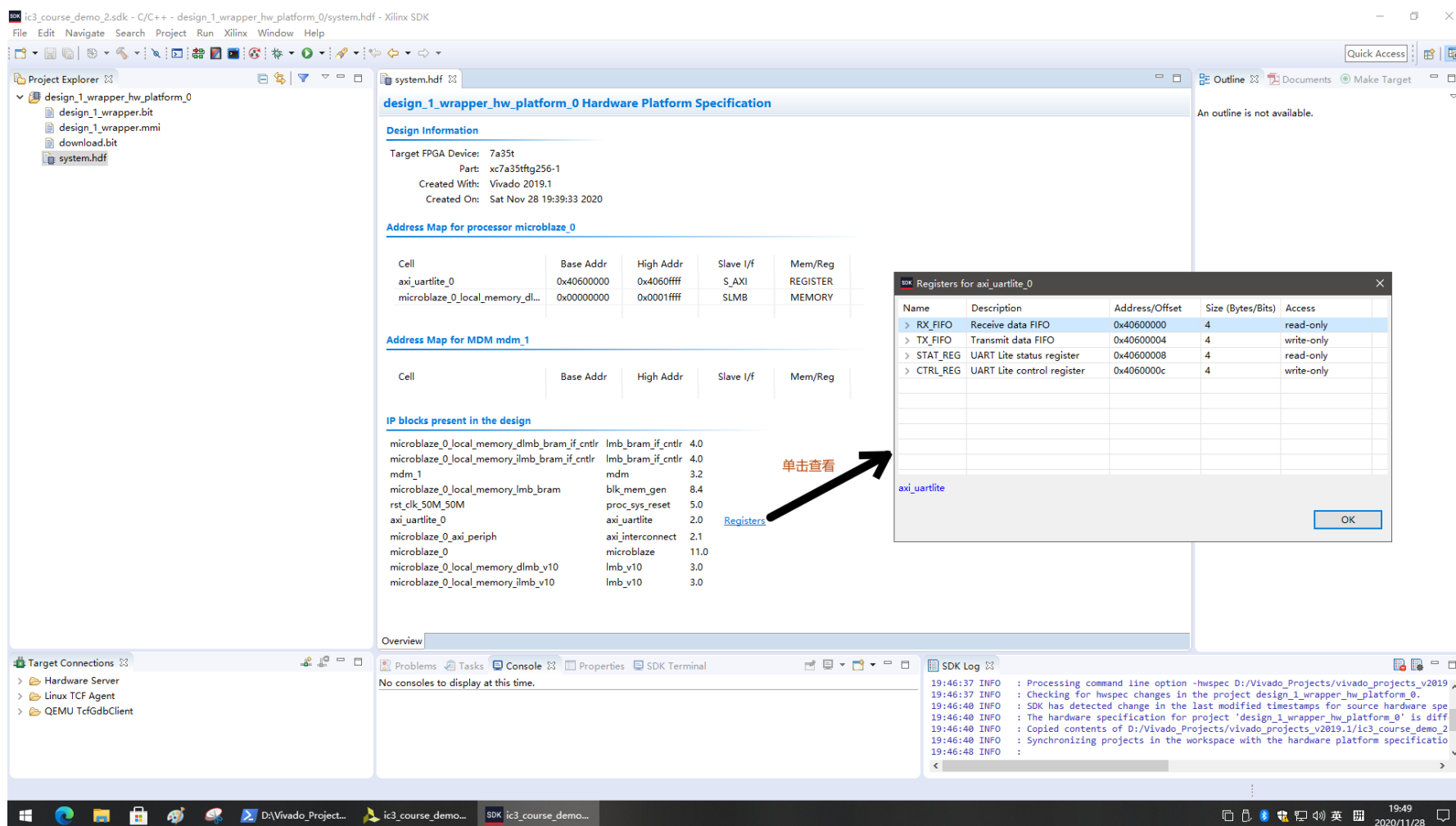
菜单栏：

File

> Launch SDK



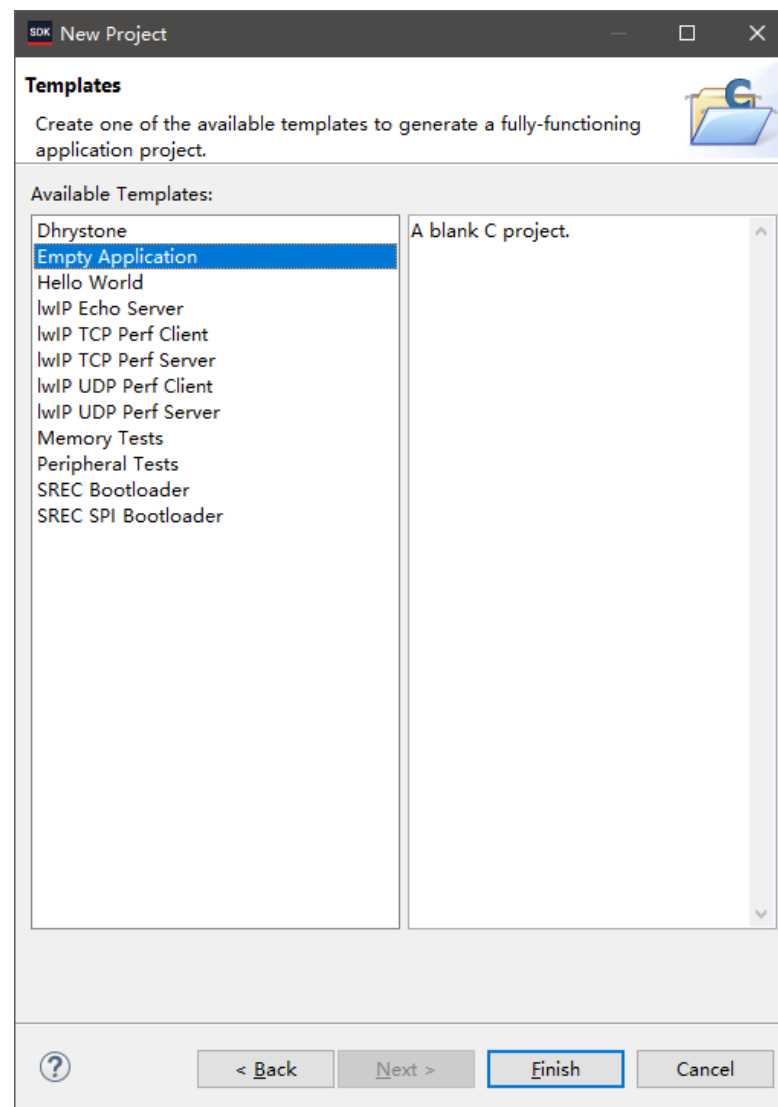
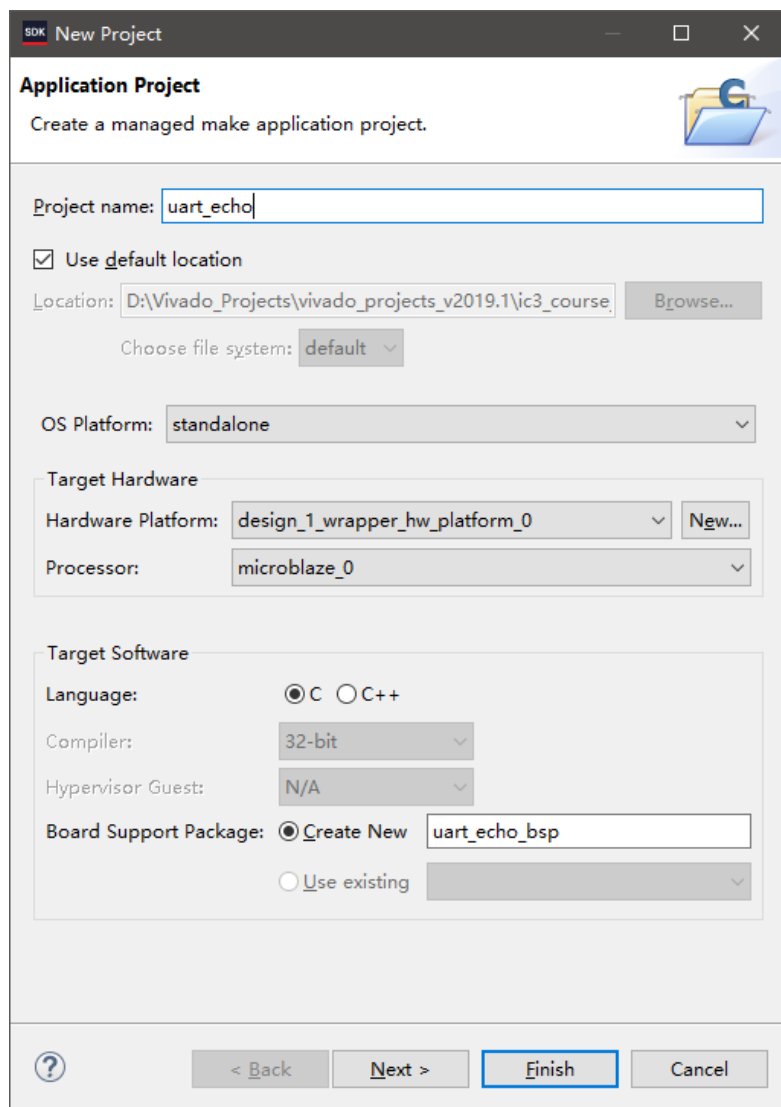
Xilinx SDK UI 概览



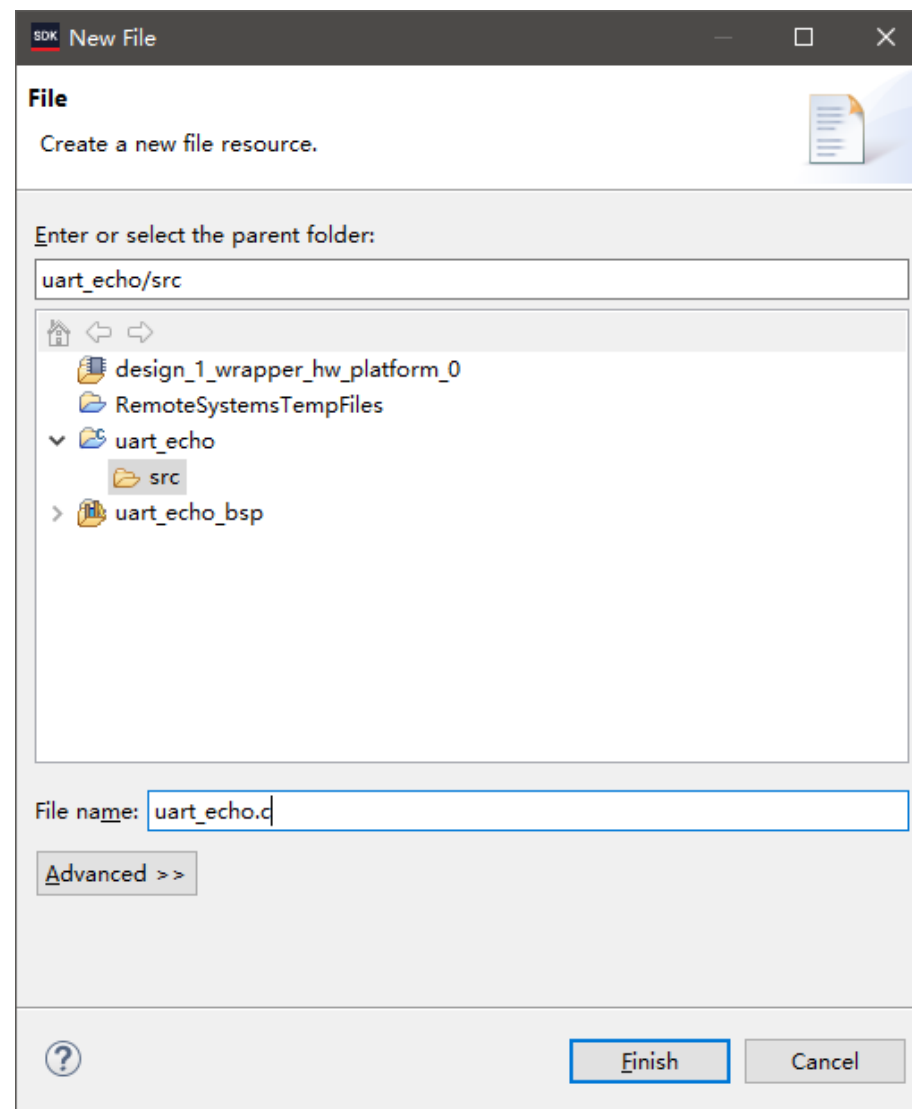
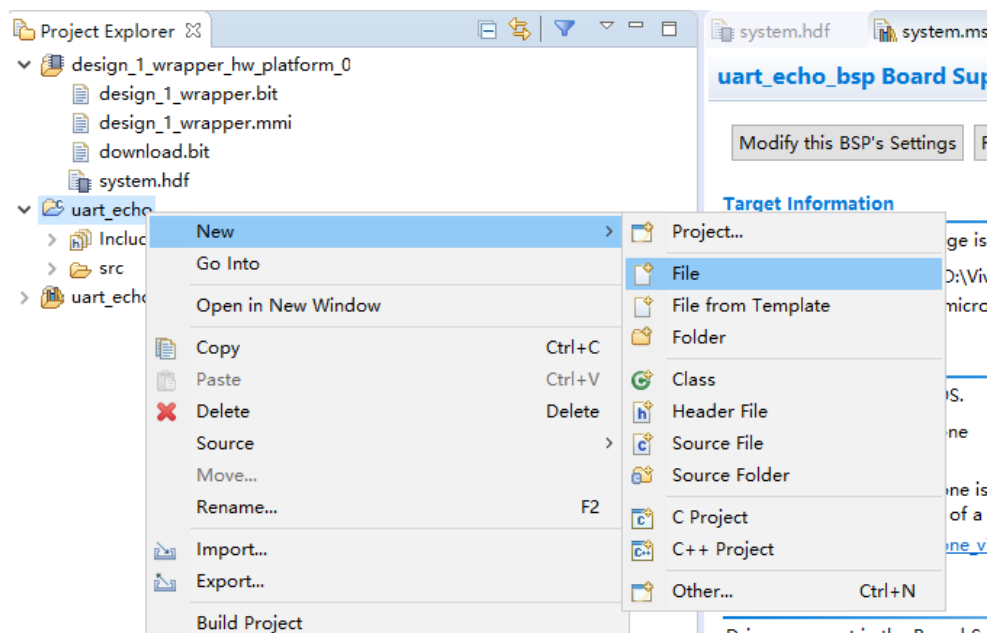
导出的文件实际上是 “system.hdf”，XSDK 打开这个文件，可以得到各个电路模块的地址分配情况（这与 Vivado Block Design 的 Address Editor 一致，前文没有叙述）

创建 APP 项目

菜单栏: File > New > Application Project



为 APP 项目添加源文件

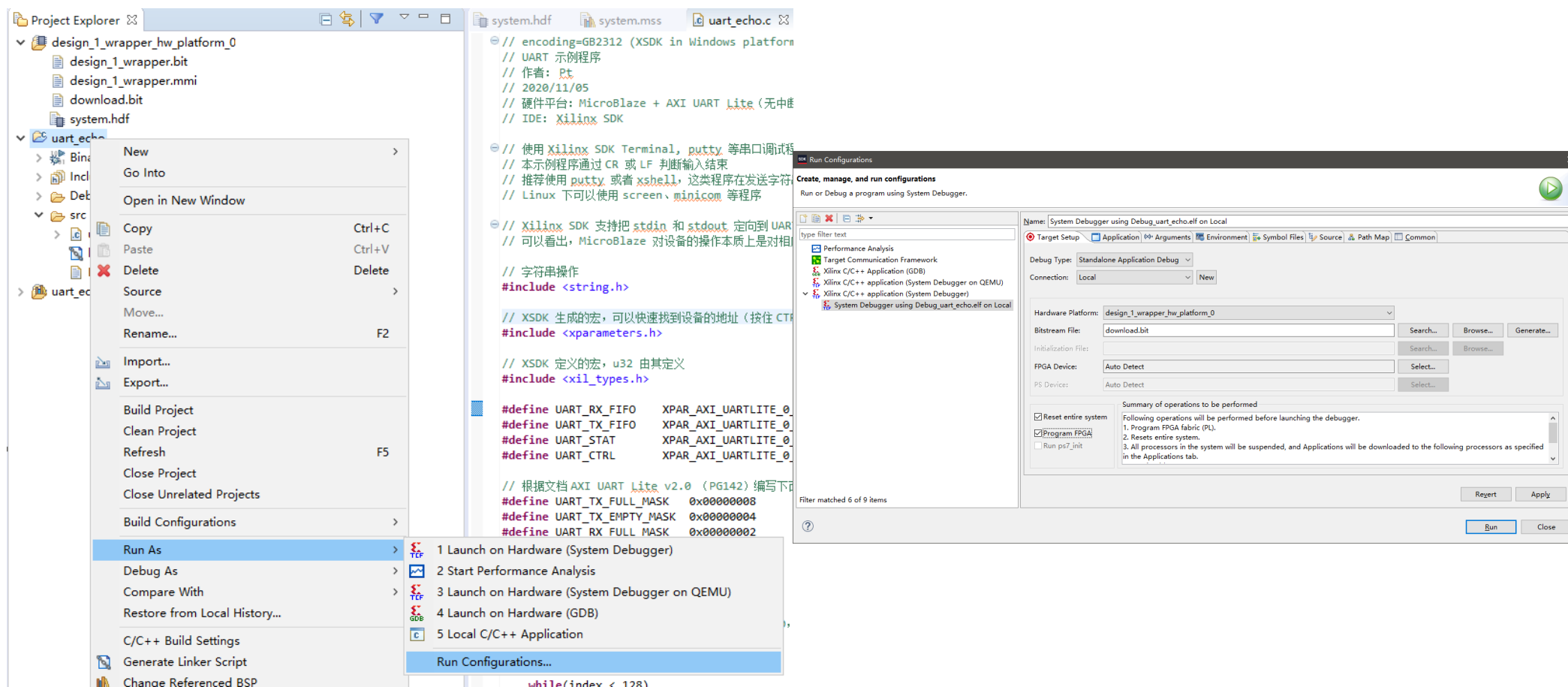


添加源文件

uart_echo.c

```
1 // encoding=GB2312 (XSDK in Windows platform)
2 // UART 示例程序
3 // 作者: Pt
4 // 2020/11/05
5 // 硬件平台: MicroBlaze + AXI UART Lite (无中断)
6 // IDE: Xilinx SDK
7
8 // 使用 Xilinx SDK Terminal, putty 等串口调试程序, 返回输入的字符串
9
10 // 本示例程序通过 CR 或 LF 判断输入结束
11
12 // 推荐使用 putty 或者 xshell, 这类程序在发送字符串后会在其后加上CRLF,
13 // 可以实现类似命令行的效果
14
15 // Linux 下可以使用 screen、minicom 等程序
16
17 // Xilinx SDK 支持把 stdin 和 stdout 定向到 UART, 但是此程序旨在演示
18 // 如何使用 C 语言操作设备, 因此此程序不使用 stdio.h 可以看出,
19 // MicroBlaze 对设备的操作本质上是对相应地址的读和写
20
21 // 字符串操作
22 #include <string.h>
23
24 // XSDK 生成的宏, 可以快速找到设备的地址 (按住 CTRL 可打开)
25 #include <xparameters.h>
26
27 // XSDK 定义的宏, u32 由其定义
28 #include <xil_types.h>
29
30 #define UART_RX_FIFO    XPAR_AXI_UARTLITE_0_BASEADDR
31 #define UART_TX_FIFO    XPAR_AXI_UARTLITE_0_BASEADDR + 0x00000004
32 #define UART_STAT       XPAR_AXI_UARTLITE_0_BASEADDR + 0x00000008
33 #define UART_CTRL       XPAR_AXI_UARTLITE_0_BASEADDR + 0x0000000c
34
35 // 根据文档 AXI UART Lite v2.0 (PG142) 编写下面的掩码,
36 // 通过掩码可以操作对应的寄存器的位
37 #define UART_TX_FULL_MASK    0x00000008
38 #define UART_TX_EMPTY_MASK   0x00000004
39 #define UART_RX_FULL_MASK    0x00000002
40 #define UART_RX_VALID_MASK   0x00000001
41 #define UART_RX_CLEAR_MASK   0x00000002
42 #define UART_TX_CLEAR_MASK   0x00000001
43
44 // 通过 UART 打印字符串
45 // 注意, 粗略地限制了 128 个字符, 因此如果没有 \0, 字符数组的打印可能出现异常。
46 void uart_puts(char * ch)
47 {
48     int index = 0;
49     while(index < 128)
50     {
51         if (((*(u32 *) (UART_STAT)) & UART_TX_FULL_MASK) == 0)
52
53         {
54             if(ch[index] == '\0') // End of a string
55             {
56                 break;
57             }
58             * (u32 *) (UART_TX_FIFO) = ch[index];
59             index++;
60         }
61         else
62         {
63             // 等待发送 FIFO 不为满, 避免数据被覆盖
64         }
65     }
66 }
67 // 通过 UART 获取字符串
68 // 需要给定字符数组的地址和获取的字节数
69 void uart_gets(char * buff, int numBytes)
70 {
71     int index = 0;
72     while(index < numBytes)
73     {
74         if (((*(u32 *) (UART_STAT)) & UART_RX_VALID_MASK)
75         {
76             buff[index] = *(u32 *) (UART_RX_FIFO);
77             if(buff[index] == '\n' || buff[index] == '\r')
78             {
79                 buff[index] = '\0';
80                 break;
81             }
82             index ++;
83         }
84         else
85         {
86             // 等待输入
87         }
88     }
89 }
90
91 int main()
92 {
93     uart_puts("\n*** UART demo: echo server ***\n");
94     char rx_buff[50] = {0};
95     char tx_buff[50] = {0};
96     while(1)
97     {
98         uart_puts("\nIn: ");
99         uart_gets(rx_buff, 50);
100        strcpy(tx_buff, "\nOut: ");
101        strcat(tx_buff, rx_buff);
102        uart_puts(tx_buff);
103        uart_puts("\n");
104    }
105    return 0;
106 }
```

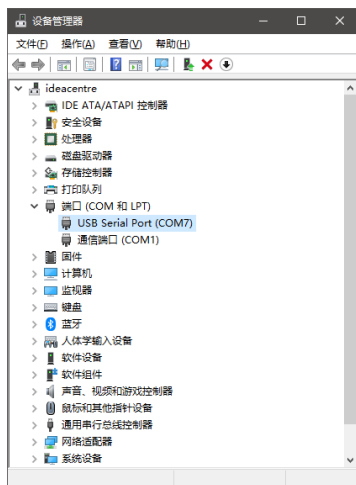
运行 APP



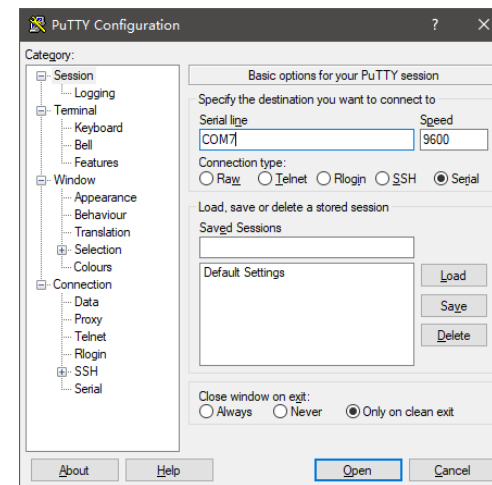
当源文件被保存后，程序被自动编译。然后，如图，打开“Run Configurations”，双击“Xilinx C/C++ application (System Debugger)”，新建一个配置。因为之前导出硬件时包含了 bitstream，因此可以在配置中设置中勾选“Program FPGA”。

配置 putty

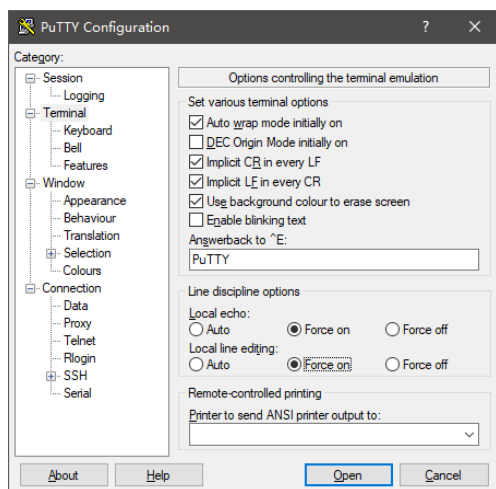
PuTTY 是一个Telnet、SSH、rlogin、纯TCP以及串行接口连接软件。（百度百科）为了使 putty 作为带有交互能力的串口终端，需要以下配置：



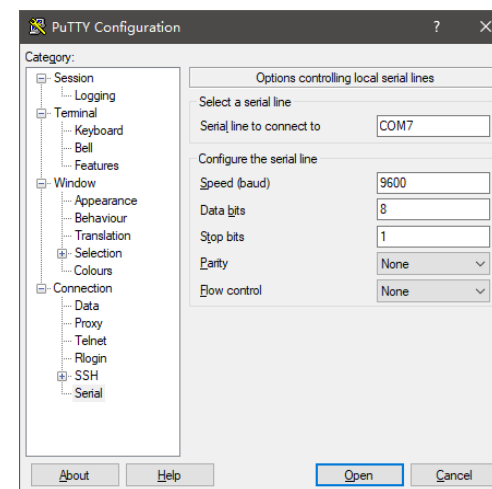
1. 打开 Windows 的设备管理器，找到被使用的串口



2. 选择Serial，波特率符合 Block Design 时 Uartlite 模块的配置

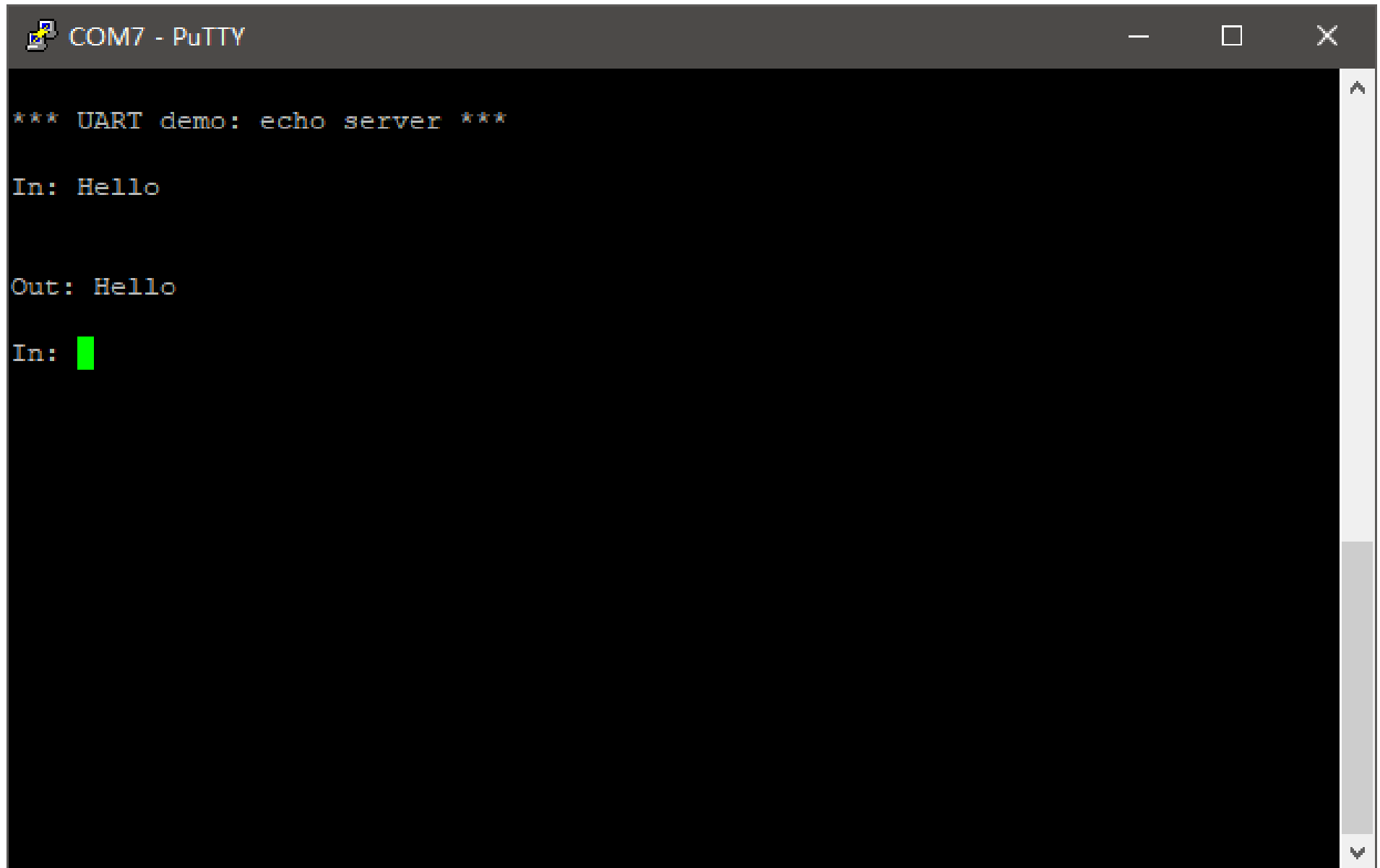


3. 使 putty 具有交互能力



4. 其它参数也符合 Block Design 时 Uartlite 模块的配置

APP 运行结果



```
COM7 - PuTTY

*** UART demo: echo server ***

In: Hello

Out: Hello

In: 
```

如图所示，电路通过 UART 接收数据并把数据传回。

IC 综合实验3 - SoC 实验 - 实验要求

1. 学习 Xilinx Vivado 平台 SoC 开发的流程;
2. 使用 MicroBlaze 核搭建 SoC 平台, 并使用 UART 实现输入输出 (即此教程之内容);
3. 结合之前的实验, 实现软件方式控制之前的实验设计的电路 (举个例子: 通过数码管显示UART的输入)