

《数字逻辑》

(FPGA开发板测试)

厦门大学信息学院软件工程系 曾文华

2024年9月9日

目录

- 第一部分：FPGA开发板介绍
- 第二部分：安装Vivado软件
- 第三部分：FPGA开发板测试
- 第四部分：Verilog HDL硬件描述语言（自学）

第一部分：FPGA开发板介绍

FPGA芯片

- **FPGA**（Field Programmable Gate Array，**现场可编程逻辑门阵列**）是在PAL（Programmable Array Logic，可编程阵列逻辑）、GAL（Generic Array Logic，通用阵列逻辑）等可编程器件的基础上进一步发展的产物。
- **FPGA芯片**是一种可重新编程的集成电路，能够实现各种数字逻辑功能。
- 它是作为专用集成电路（ASIC：Application Specific Integrated Circuit）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。



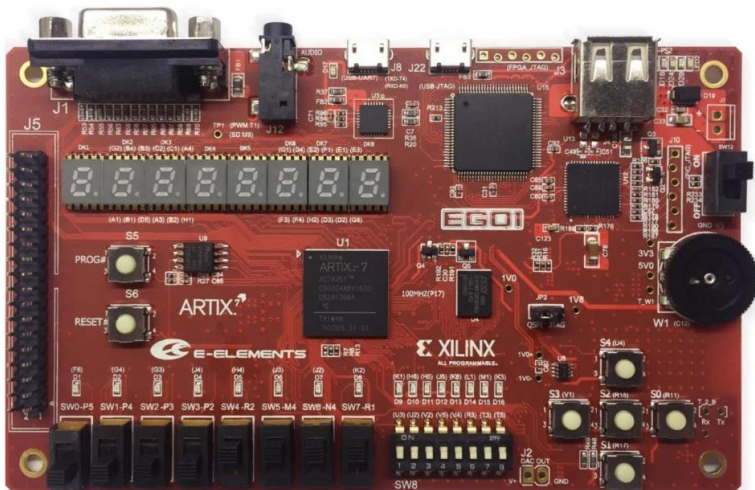
FPGA开发板

- **FPGA开发板**是一种基于FPGA芯片的硬件平台，FPGA开发板是用于学习、原型设计和产品开发的重要工具。
- FPGA开发板通常由FPGA芯片、外部存储器、输入/输出接口和其他辅助电路组成。它们提供了一个灵活且可定制的平台，使得开发人员能够快速构建和测试各种数字电路。通过使用专门的开发软件（例如**Vivado**），开发人员可以对FPGA芯片进行编程，实现所需的逻辑功能。



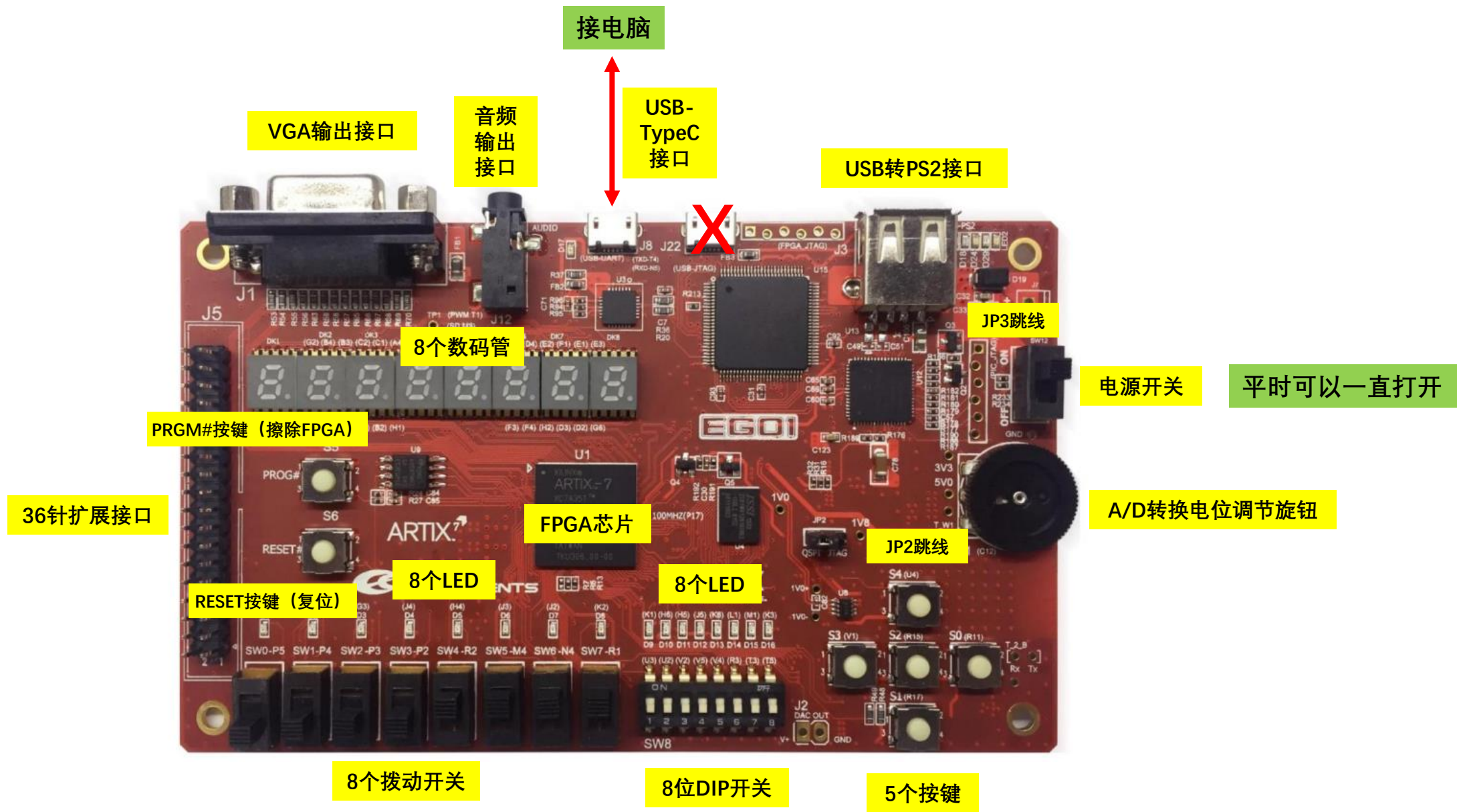
FPGA开发板： EGO1

- EGO1 是依元素科技公司基于 **Xilinx Artix-7 FPGA** 研发的便携式数模混合基础教学平台。
- EGO1 配备的 FPGA (**XC7A35T-1CSG324C**)具有大容量高性能等特点， 能实现较复杂的数字逻辑设计。EGO1 拥有丰富的外设， 以及灵活的通用扩展接口。



FPGA : Xilinx Artix-7 XC7A35T
时钟 : 100MHz
配置方式 : USB-JTAG/SPI Flash
存储器 :
 SRAM : 2Mbit
 SPI Flash : N25Q032A
通用IO :
 Switch : x8
 LED : x16
 Button : x5
 DIP : x8

通用扩展IO : 32pin
音视频/显示 :
 7段数码管 : x8
 VGA视频输出接口
 Audio音频接口
通信接口 :
 UART : USB转UART
 Bluetooth : 蓝牙模块
模拟接口 :
 DAC: 8-bit分辨率
 XADC: 2路12bit 1Msps ADC



EGO1开发板FPGA芯片的管脚定义 (1)

EGO1.xdc - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

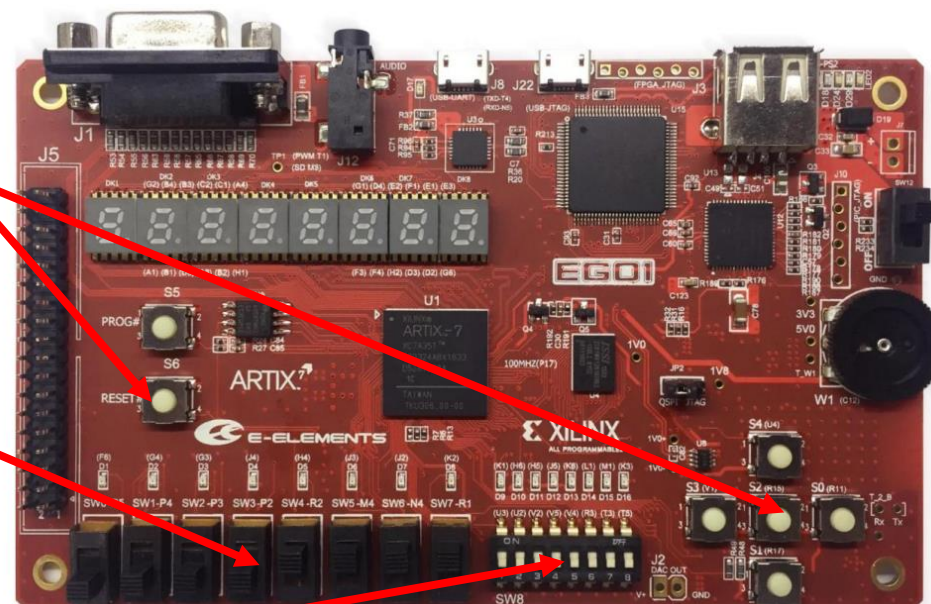
EGO1.xdc

```
//-----系统时钟和复位-----  
set_property -dict {PACKAGE_PIN P17 IOSTANDARD LVCMOS33} [get_ports sys_clk_in]  
set_property -dict {PACKAGE_PIN P15 IOSTANDARD LVCMOS33} [get_ports sys_rst_n]
```

```
//-----5个按键-----  
set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports {btn_0}]  
set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {btn_1}]  
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {btn_2}]  
set_property -dict {PACKAGE_PIN V1 IOSTANDARD LVCMOS33} [get_ports {btn_3}]  
set_property -dict {PACKAGE_PIN U4 IOSTANDARD LVCMOS33} [get_ports {btn_4}]
```

```
//-----拨码开关sw0~sw7-----  
set_property -dict {PACKAGE_PIN P5 IOSTANDARD LVCMOS33} [get_ports {sw_pin[0]}]  
set_property -dict {PACKAGE_PIN P4 IOSTANDARD LVCMOS33} [get_ports {sw_pin[1]}]  
set_property -dict {PACKAGE_PIN P3 IOSTANDARD LVCMOS33} [get_ports {sw_pin[2]}]  
set_property -dict {PACKAGE_PIN P2 IOSTANDARD LVCMOS33} [get_ports {sw_pin[3]}]  
set_property -dict {PACKAGE_PIN R2 IOSTANDARD LVCMOS33} [get_ports {sw_pin[4]}]  
set_property -dict {PACKAGE_PIN M4 IOSTANDARD LVCMOS33} [get_ports {sw_pin[5]}]  
set_property -dict {PACKAGE_PIN N4 IOSTANDARD LVCMOS33} [get_ports {sw_pin[6]}]  
set_property -dict {PACKAGE_PIN R1 IOSTANDARD LVCMOS33} [get_ports {sw_pin[7]}]
```

```
//-----拨码开关 (DIP开关) sw8~sw15-----  
set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[0]}]  
set_property -dict {PACKAGE_PIN U2 IOSTANDARD LVCMOS33} [get_ports {dip_pin[1]}]  
set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {dip_pin[2]}]  
set_property -dict {PACKAGE_PIN V5 IOSTANDARD LVCMOS33} [get_ports {dip_pin[3]}]  
set_property -dict {PACKAGE_PIN V4 IOSTANDARD LVCMOS33} [get_ports {dip_pin[4]}]  
set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[5]}]  
set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[6]}]  
set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVCMOS33} [get_ports {dip_pin[7]}]
```



EGO1开发板FPGA芯片的管脚定义 (2)

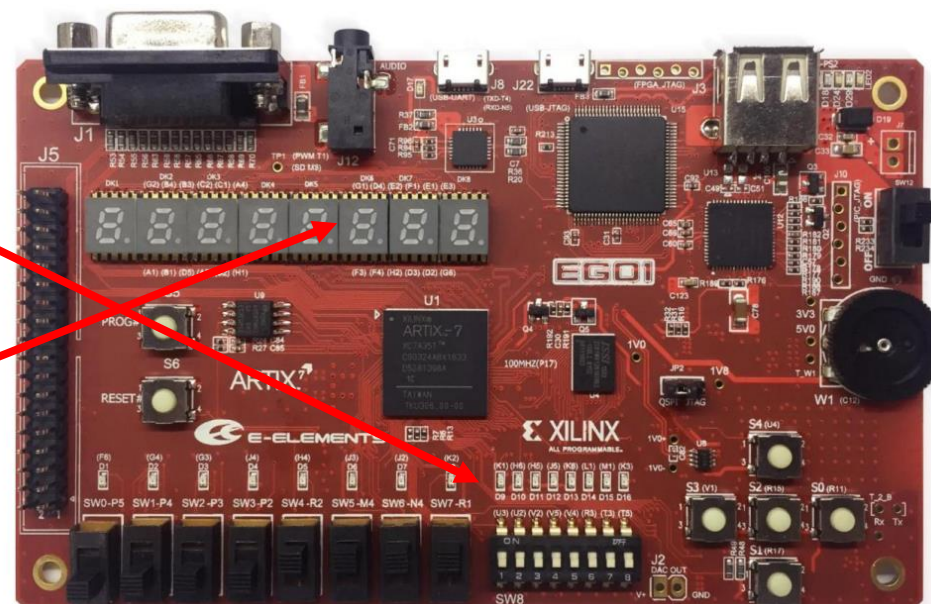
```
//-----LED0~LED15-----
set_property -dict {PACKAGE_PIN F6 IOSTANDARD LVCMOS33} [get_ports {led_pin[0]]}
set_property -dict {PACKAGE_PIN G4 IOSTANDARD LVCMOS33} [get_ports {led_pin[1]]}
set_property -dict {PACKAGE_PIN G3 IOSTANDARD LVCMOS33} [get_ports {led_pin[2]]}
set_property -dict {PACKAGE_PIN J4 IOSTANDARD LVCMOS33} [get_ports {led_pin[3]]}
set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {led_pin[4]]}
set_property -dict {PACKAGE_PIN J3 IOSTANDARD LVCMOS33} [get_ports {led_pin[5]]}
set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {led_pin[6]]}
set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {led_pin[7]]}
```

```
set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {led_pin[8]]}
set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {led_pin[9]]}
set_property -dict {PACKAGE_PIN H5 IOSTANDARD LVCMOS33} [get_ports {led_pin[10]]}
set_property -dict {PACKAGE_PIN J5 IOSTANDARD LVCMOS33} [get_ports {led_pin[11]]}
set_property -dict {PACKAGE_PIN K6 IOSTANDARD LVCMOS33} [get_ports {led_pin[12]]}
set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {led_pin[13]]}
set_property -dict {PACKAGE_PIN M1 IOSTANDARD LVCMOS33} [get_ports {led_pin[14]]}
set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVCMOS33} [get_ports {led_pin[15]]}
```

```
//-----8个数码管位选信号-----
set_property -dict {PACKAGE_PIN G2 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[0]]}
set_property -dict {PACKAGE_PIN C2 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[1]]}
set_property -dict {PACKAGE_PIN C1 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[2]]}
set_property -dict {PACKAGE_PIN H1 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[3]]}
set_property -dict {PACKAGE_PIN G1 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[4]]}
set_property -dict {PACKAGE_PIN F1 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[5]]}
set_property -dict {PACKAGE_PIN E1 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[6]]}
set_property -dict {PACKAGE_PIN G6 IOSTANDARD LVCMOS33} [get_ports {seg_cs_pin[7]]}
```

```
//-----数码管段选信号-----
set_property -dict {PACKAGE_PIN B4 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[0]]}
set_property -dict {PACKAGE_PIN A4 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[1]]}
set_property -dict {PACKAGE_PIN A3 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[2]]}
set_property -dict {PACKAGE_PIN B1 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[3]]}
set_property -dict {PACKAGE_PIN A1 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[4]]}
set_property -dict {PACKAGE_PIN B3 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[5]]}
set_property -dict {PACKAGE_PIN B2 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[6]]}
set_property -dict {PACKAGE_PIN D5 IOSTANDARD LVCMOS33} [get_ports {seg_data_0_pin[7]]}
```

```
set_property -dict {PACKAGE_PIN D4 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[0]]}
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[1]]}
set_property -dict {PACKAGE_PIN D3 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[2]]}
set_property -dict {PACKAGE_PIN F4 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[3]]}
set_property -dict {PACKAGE_PIN F3 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[4]]}
set_property -dict {PACKAGE_PIN E2 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[5]]}
set_property -dict {PACKAGE_PIN D2 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[6]]}
set_property -dict {PACKAGE_PIN H2 IOSTANDARD LVCMOS33} [get_ports {seg_data_1_pin[7]]}
```



EGO1开发板FPGA芯片的管脚定义 (3)

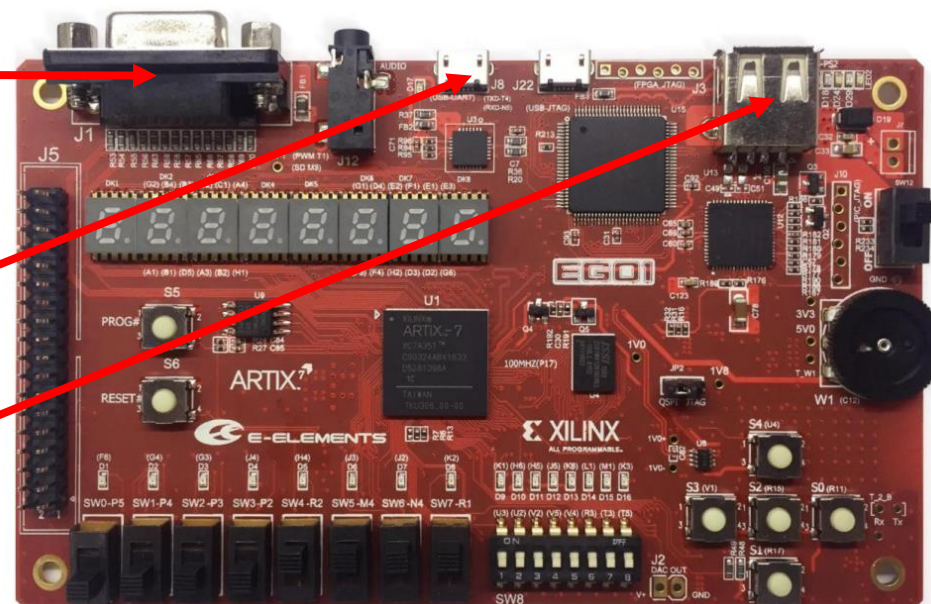
```
//-----VGA行同步场同步信号-----  
set_property -dict {PACKAGE_PIN D7 IOSTANDARD LVCMOS33} [get_ports vga_hs_pin]  
set_property -dict {PACKAGE_PIN C4 IOSTANDARD LVCMOS33} [get_ports vga_vs_pin]
```

```
//-----VGA红绿蓝信号-----  
set_property -dict {PACKAGE_PIN F5 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[0]}]  
set_property -dict {PACKAGE_PIN C6 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[1]}]  
set_property -dict {PACKAGE_PIN C5 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[2]}]  
set_property -dict {PACKAGE_PIN B7 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[3]}]  
set_property -dict {PACKAGE_PIN B6 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[4]}]  
set_property -dict {PACKAGE_PIN A6 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[5]}]  
set_property -dict {PACKAGE_PIN A5 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[6]}]  
set_property -dict {PACKAGE_PIN D8 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[7]}]  
set_property -dict {PACKAGE_PIN C7 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[8]}]  
set_property -dict {PACKAGE_PIN E6 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[9]}]  
set_property -dict {PACKAGE_PIN E5 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[10]}]  
set_property -dict {PACKAGE_PIN E7 IOSTANDARD LVCMOS33} [get_ports {vga_data_pin[11]}]
```

```
//-----串口-----  
set_property -dict {PACKAGE_PIN N5 IOSTANDARD LVCMOS33} [get_ports PC_Uart_rxd]  
set_property -dict {PACKAGE_PIN T4 IOSTANDARD LVCMOS33} [get_ports PC_Uart_txd]
```

```
//-----PS2接口-----  
set_property -dict {PACKAGE_PIN K5 IOSTANDARD LVCMOS33} [get_ports ps2_clk]  
set_property -dict {PACKAGE_PIN L4 IOSTANDARD LVCMOS33} [get_ports ps2_data]
```

```
//-----IIC接口-----  
set_property -dict {PACKAGE_PIN F18 IOSTANDARD LVCMOS33} [get_ports pw_iic_scl_io]  
set_property -dict {PACKAGE_PIN G18 IOSTANDARD LVCMOS33} [get_ports pw_iic_sda_io]
```



EGO1开发板FPGA芯片的管脚定义 (4)

```
//-----蓝牙-----  
set_property -dict {PACKAGE_PIN L3 IOSTANDARD LVCMOS33} [get_ports BT_Uart_rxd]  
set_property -dict {PACKAGE_PIN N2 IOSTANDARD LVCMOS33} [get_ports BT_Uart_txd]
```

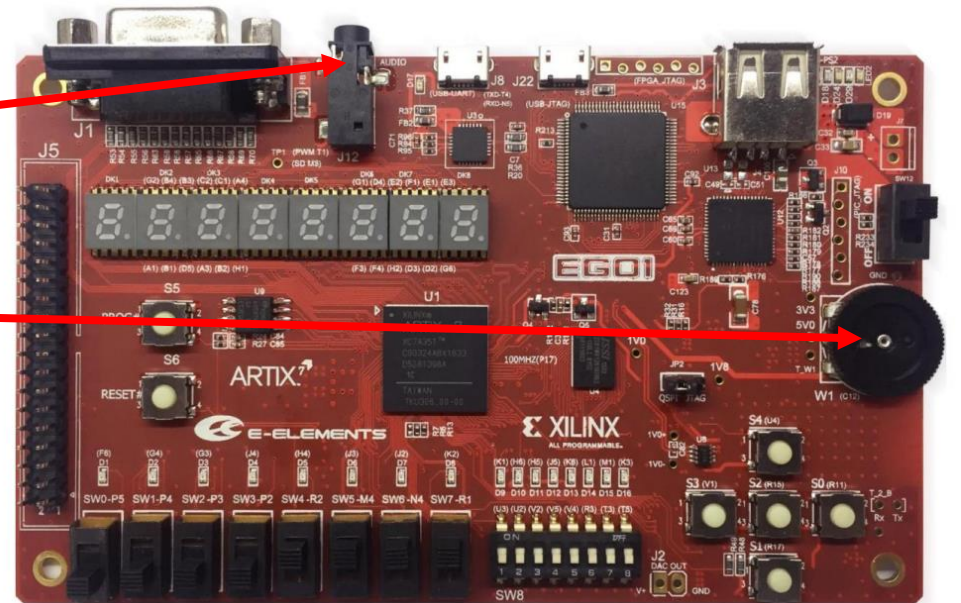
```
set_property -dict {PACKAGE_PIN D18 IOSTANDARD LVCMOS33} [get_ports {bt_ctrl_o[0]}]  
set_property -dict {PACKAGE_PIN M2 IOSTANDARD LVCMOS33} [get_ports {bt_ctrl_o[1]}]  
set_property -dict {PACKAGE_PIN H15 IOSTANDARD LVCMOS33} [get_ports {bt_ctrl_o[2]}]  
set_property -dict {PACKAGE_PIN C16 IOSTANDARD LVCMOS33} [get_ports {bt_ctrl_o[3]}]  
set_property -dict {PACKAGE_PIN E18 IOSTANDARD LVCMOS33} [get_ports {bt_ctrl_o[4]}]  
  
set_property -dict {PACKAGE_PIN C17 IOSTANDARD LVCMOS33} [get_ports bt_mcu_int_i]
```

```
//-----音频接口-----  
set_property -dict {PACKAGE_PIN T1 IOSTANDARD LVCMOS33} [get_ports audio_pwm_o]  
set_property -dict {PACKAGE_PIN M6 IOSTANDARD LVCMOS33} [get_ports audio_sd_o]
```

```
//-----XADC模数转换-----  
set_property -dict {PACKAGE_PIN B12 IOSTANDARD LVCMOS33} [get_ports XADC_AUX_v_n]  
set_property -dict {PACKAGE_PIN C12 IOSTANDARD LVCMOS33} [get_ports XADC_AUX_v_p]  
set_property -dict {PACKAGE_PIN K9 IOSTANDARD LVCMOS33} [get_ports XADC_VP_VN_v_n]  
set_property -dict {PACKAGE_PIN J10 IOSTANDARD LVCMOS33} [get_ports XADC_VP_VN_v_p]
```

```
//-----DAC数模转换-----  
set_property -dict {PACKAGE_PIN R5 IOSTANDARD LVCMOS33} [get_ports dac_ile]  
set_property -dict {PACKAGE_PIN N6 IOSTANDARD LVCMOS33} [get_ports dac_cs_n]  
set_property -dict {PACKAGE_PIN V6 IOSTANDARD LVCMOS33} [get_ports dac_wr1_n]  
set_property -dict {PACKAGE_PIN R6 IOSTANDARD LVCMOS33} [get_ports dac_wr2_n]  
set_property -dict {PACKAGE_PIN V7 IOSTANDARD LVCMOS33} [get_ports dac_xfer_n]
```

```
set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS33} [get_ports {dac_data[0]}]  
set_property -dict {PACKAGE_PIN R8 IOSTANDARD LVCMOS33} [get_ports {dac_data[1]}]  
set_property -dict {PACKAGE_PIN T6 IOSTANDARD LVCMOS33} [get_ports {dac_data[2]}]  
set_property -dict {PACKAGE_PIN R7 IOSTANDARD LVCMOS33} [get_ports {dac_data[3]}]  
set_property -dict {PACKAGE_PIN U6 IOSTANDARD LVCMOS33} [get_ports {dac_data[4]}]  
set_property -dict {PACKAGE_PIN U7 IOSTANDARD LVCMOS33} [get_ports {dac_data[5]}]  
set_property -dict {PACKAGE_PIN V9 IOSTANDARD LVCMOS33} [get_ports {dac_data[6]}]  
set_property -dict {PACKAGE_PIN U9 IOSTANDARD LVCMOS33} [get_ports {dac_data[7]}]
```

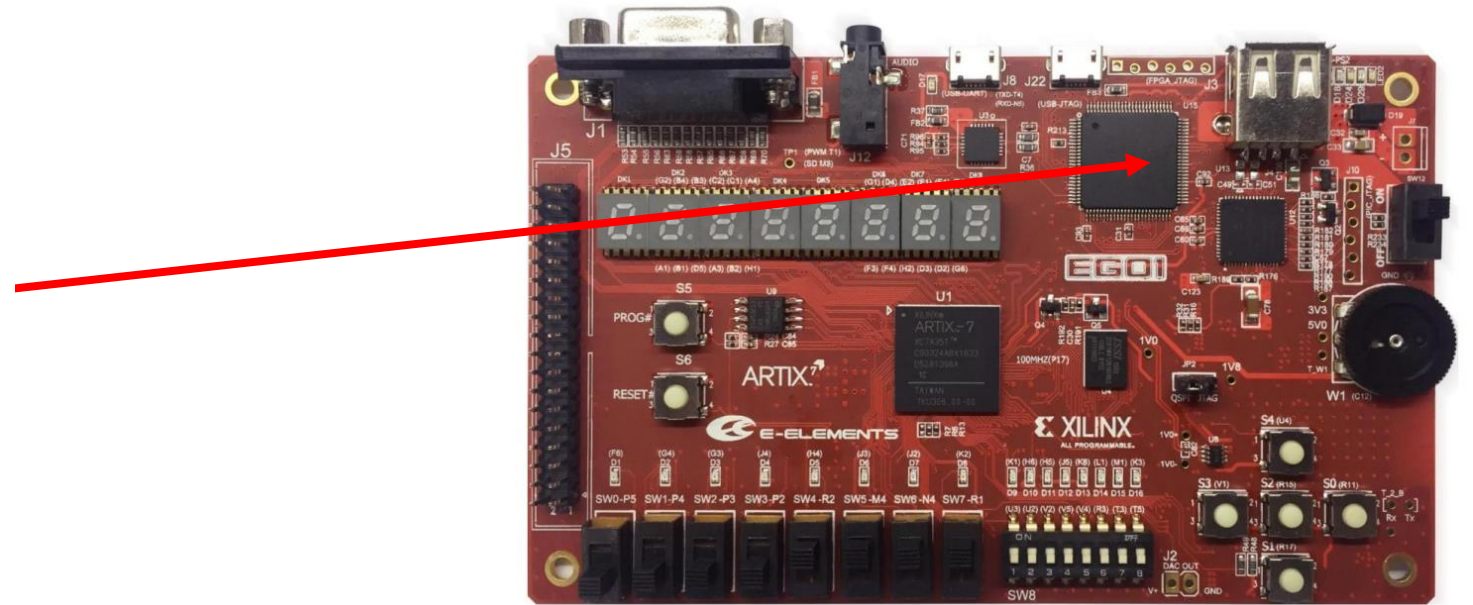


EGO1开发板FPGA芯片的管脚定义 (5)

```
//-----SDRAM芯片接口-----
set_property -dict {PACKAGE_PIN L15 IOSTANDARD LVCMOS33} [get_ports {sram_addr[18]]}
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {sram_addr[17]]}
set_property -dict {PACKAGE_PIN L18 IOSTANDARD LVCMOS33} [get_ports {sram_addr[16]]}
set_property -dict {PACKAGE_PIN M18 IOSTANDARD LVCMOS33} [get_ports {sram_addr[15]]}
set_property -dict {PACKAGE_PIN R12 IOSTANDARD LVCMOS33} [get_ports {sram_addr[14]]}
set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {sram_addr[13]]}
set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {sram_addr[12]]}
set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports {sram_addr[11]]}
set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {sram_addr[10]]}
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {sram_addr[9]]}
set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33} [get_ports {sram_addr[8]]}
set_property -dict {PACKAGE_PIN N17 IOSTANDARD LVCMOS33} [get_ports {sram_addr[7]]}
set_property -dict {PACKAGE_PIN P18 IOSTANDARD LVCMOS33} [get_ports {sram_addr[6]]}
set_property -dict {PACKAGE_PIN M16 IOSTANDARD LVCMOS33} [get_ports {sram_addr[5]]}
set_property -dict {PACKAGE_PIN M17 IOSTANDARD LVCMOS33} [get_ports {sram_addr[4]]}
set_property -dict {PACKAGE_PIN N15 IOSTANDARD LVCMOS33} [get_ports {sram_addr[3]]}
set_property -dict {PACKAGE_PIN N16 IOSTANDARD LVCMOS33} [get_ports {sram_addr[2]]}
set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33} [get_ports {sram_addr[1]]}
set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports {sram_addr[0]]}

set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports sram_ce_n]
set_property -dict {PACKAGE_PIN R10 IOSTANDARD LVCMOS33} [get_ports sram_lb_n]
set_property -dict {PACKAGE_PIN T16 IOSTANDARD LVCMOS33} [get_ports sram_oe_n]
set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports sram_ub_n]
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports sram_we_n]

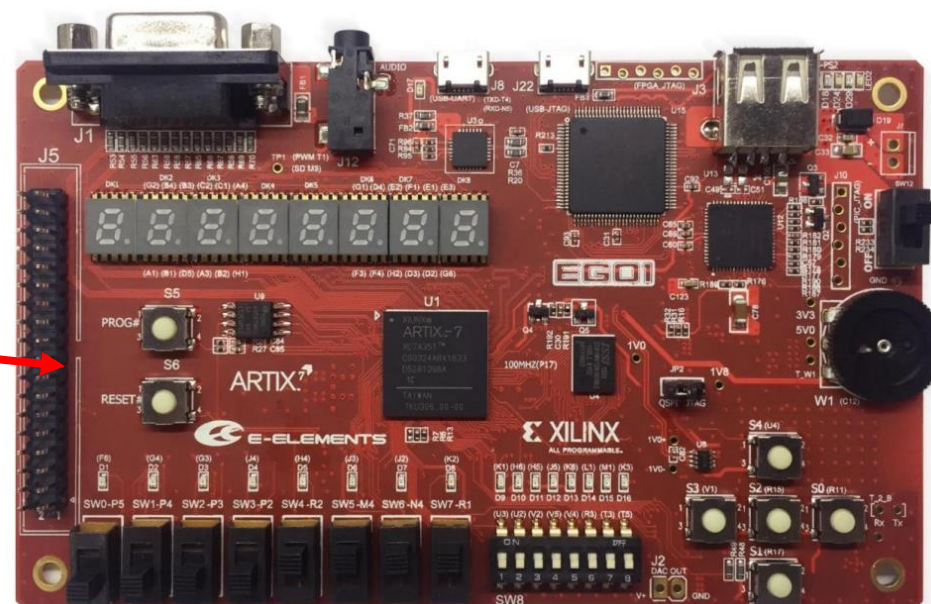
set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33} [get_ports {sram_data[15]]}
set_property -dict {PACKAGE_PIN T9 IOSTANDARD LVCMOS33} [get_ports {sram_data[14]]}
set_property -dict {PACKAGE_PIN U13 IOSTANDARD LVCMOS33} [get_ports {sram_data[13]]}
set_property -dict {PACKAGE_PIN T13 IOSTANDARD LVCMOS33} [get_ports {sram_data[12]]}
set_property -dict {PACKAGE_PIN V14 IOSTANDARD LVCMOS33} [get_ports {sram_data[11]]}
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33} [get_ports {sram_data[10]]}
set_property -dict {PACKAGE_PIN V11 IOSTANDARD LVCMOS33} [get_ports {sram_data[9]]}
set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {sram_data[8]]}
set_property -dict {PACKAGE_PIN V12 IOSTANDARD LVCMOS33} [get_ports {sram_data[7]]}
set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {sram_data[6]]}
set_property -dict {PACKAGE_PIN U11 IOSTANDARD LVCMOS33} [get_ports {sram_data[5]]}
set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33} [get_ports {sram_data[4]]}
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {sram_data[3]]}
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {sram_data[2]]}
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {sram_data[1]]}
set_property -dict {PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports {sram_data[0]]}
```



EGO1开发板FPGA芯片的管脚定义 (6)

```
//-----32个pmod接口 (扩展接口) -----  
set_property -dict {PACKAGE_PIN B16 IOSTANDARD LVCMOS33} [get_ports {exp_io[0]]}  
set_property -dict {PACKAGE_PIN A15 IOSTANDARD LVCMOS33} [get_ports {exp_io[1]]}  
set_property -dict {PACKAGE_PIN A13 IOSTANDARD LVCMOS33} [get_ports {exp_io[2]]}  
set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33} [get_ports {exp_io[3]]}  
set_property -dict {PACKAGE_PIN F13 IOSTANDARD LVCMOS33} [get_ports {exp_io[4]]}  
set_property -dict {PACKAGE_PIN B13 IOSTANDARD LVCMOS33} [get_ports {exp_io[5]]}  
set_property -dict {PACKAGE_PIN D14 IOSTANDARD LVCMOS33} [get_ports {exp_io[6]]}  
set_property -dict {PACKAGE_PIN B11 IOSTANDARD LVCMOS33} [get_ports {exp_io[7]]}  
set_property -dict {PACKAGE_PIN E15 IOSTANDARD LVCMOS33} [get_ports {exp_io[8]]}  
set_property -dict {PACKAGE_PIN D15 IOSTANDARD LVCMOS33} [get_ports {exp_io[9]]}  
set_property -dict {PACKAGE_PIN H16 IOSTANDARD LVCMOS33} [get_ports {exp_io[10]]}  
set_property -dict {PACKAGE_PIN F15 IOSTANDARD LVCMOS33} [get_ports {exp_io[11]]}  
set_property -dict {PACKAGE_PIN H14 IOSTANDARD LVCMOS33} [get_ports {exp_io[12]]}  
set_property -dict {PACKAGE_PIN E17 IOSTANDARD LVCMOS33} [get_ports {exp_io[13]]}  
set_property -dict {PACKAGE_PIN K13 IOSTANDARD LVCMOS33} [get_ports {exp_io[14]]}  
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {exp_io[15]]}
```

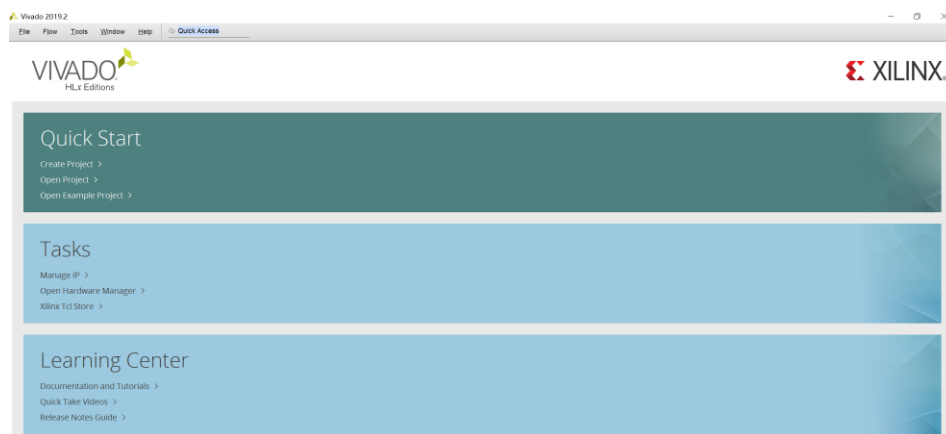
```
set_property -dict {PACKAGE_PIN B17 IOSTANDARD LVCMOS33} [get_ports {exp_io[16]]}  
set_property -dict {PACKAGE_PIN A16 IOSTANDARD LVCMOS33} [get_ports {exp_io[17]]}  
set_property -dict {PACKAGE_PIN A14 IOSTANDARD LVCMOS33} [get_ports {exp_io[18]]}  
set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33} [get_ports {exp_io[19]]}  
set_property -dict {PACKAGE_PIN F14 IOSTANDARD LVCMOS33} [get_ports {exp_io[20]]}  
set_property -dict {PACKAGE_PIN B14 IOSTANDARD LVCMOS33} [get_ports {exp_io[21]]}  
set_property -dict {PACKAGE_PIN C14 IOSTANDARD LVCMOS33} [get_ports {exp_io[22]]}  
set_property -dict {PACKAGE_PIN A11 IOSTANDARD LVCMOS33} [get_ports {exp_io[23]]}  
set_property -dict {PACKAGE_PIN E16 IOSTANDARD LVCMOS33} [get_ports {exp_io[24]]}  
set_property -dict {PACKAGE_PIN C15 IOSTANDARD LVCMOS33} [get_ports {exp_io[25]]}  
set_property -dict {PACKAGE_PIN G16 IOSTANDARD LVCMOS33} [get_ports {exp_io[26]]}  
set_property -dict {PACKAGE_PIN F16 IOSTANDARD LVCMOS33} [get_ports {exp_io[27]]}  
set_property -dict {PACKAGE_PIN G14 IOSTANDARD LVCMOS33} [get_ports {exp_io[28]]}  
set_property -dict {PACKAGE_PIN D17 IOSTANDARD LVCMOS33} [get_ports {exp_io[29]]}  
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {exp_io[30]]}  
set_property -dict {PACKAGE_PIN G17 IOSTANDARD LVCMOS33} [get_ports {exp_io[31]]}
```



第二部分：安装Vivado软件

Vivado简介

- Vivado是一款由赛灵思（Xilinx）公司开发的集成电路设计工具，主要用于FPGA的设计、仿真和综合。
- Vivado具有界面友好、操作简单的特点。由于Xilinx公司的FPGA芯片占有很大的市场，这也使得Vivado成为非常通用的FPGA工具软件。
- Vivado的主要功能包括设计输入、综合、仿真、实现和装载，涵盖了FPGA开发的全过程。



Vivado安装

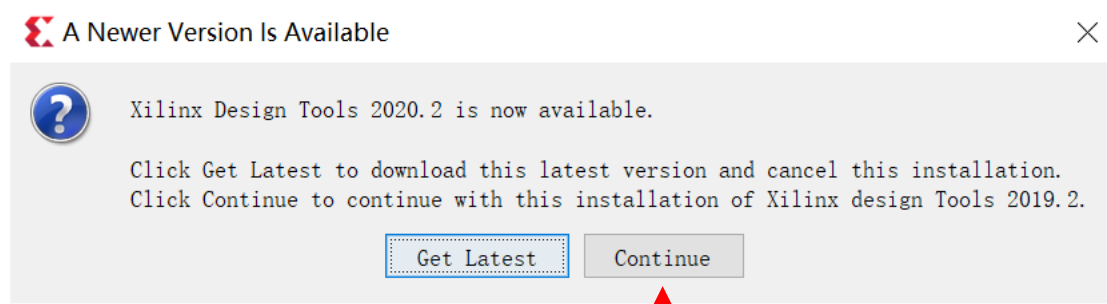
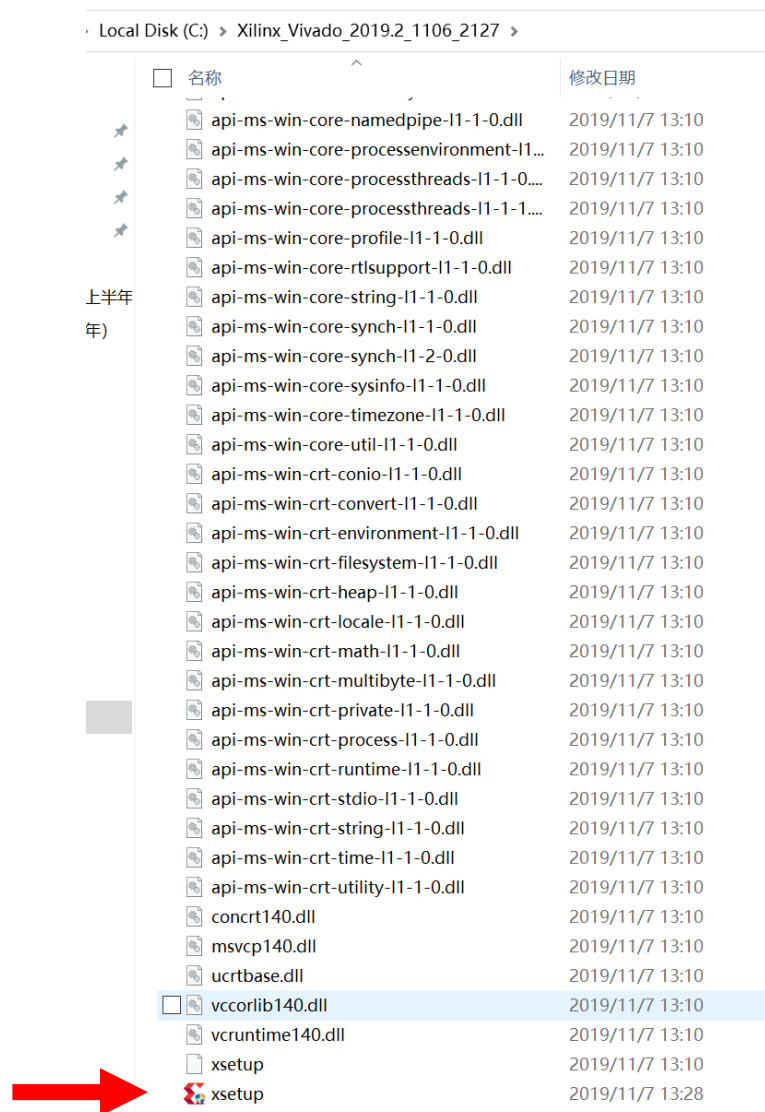
- 下载Xilinx_Vivado_2019.2_1106_2127.tar.zip文件，解压到电脑硬盘的根目录（例如，C:\Xilinx_Vivado_2019.2_1106_2127，也可以是其它硬盘）。

(C:) > Xilinx_Vivado_2019.2_1106_2127 >

名称	修改日期	类型	大小
api-ms-win-core-heap-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-interlocked-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-libraryloader-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-localization-l1-2-0.dll	2019/11/7 13:10	应用程序扩展	21 KB
api-ms-win-core-memory-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-namedpipe-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-processenvironment-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	20 KB
api-ms-win-core-processthreads-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	21 KB
api-ms-win-core-processutils-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-profile-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	18 KB
api-ms-win-core-rtlsupport-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-string-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-synch-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	21 KB
api-ms-win-core-synch-l1-2-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-sysinfo-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	20 KB
api-ms-win-core-timezone-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-core-utility-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-crt-conio-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	20 KB
api-ms-win-crt-convert-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	23 KB
api-ms-win-crt-environment-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-crt-filestream-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	21 KB
api-ms-win-crt-heap-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	20 KB
api-ms-win-crt-locale-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
api-ms-win-crt-math-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	28 KB
api-ms-win-crt-multibyte-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	27 KB
api-ms-win-crt-private-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	70 KB
api-ms-win-crt-process-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	20 KB
api-ms-win-crt-runtime-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	23 KB
api-ms-win-crt-stdio-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	25 KB
api-ms-win-crt-string-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	25 KB
api-ms-win-crt-time-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	21 KB
api-ms-win-crt-utility-l1-1-0.dll	2019/11/7 13:10	应用程序扩展	19 KB
concr140.dll	2019/11/7 13:10	应用程序扩展	328 KB
msvc140.dll	2019/11/7 13:10	应用程序扩展	625 KB
ucrtbase.dll	2019/11/7 13:10	应用程序扩展	960 KB
vccorlib140.dll	2019/11/7 13:10	应用程序扩展	387 KB
vcruntime140.dll	2019/11/7 13:10	应用程序扩展	88 KB
xsetup	2019/11/7 13:10	文件	3 KB
xsetup	2019/11/7 13:28	应用程序	439 KB



- 点击执行“**xsetup.exe**”。



Accept License Agreements



Please read the following terms and conditions and indicate that you agree by checking the I Agree checkboxes.

Xilinx Inc. End User License Agreement

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

☒ I Agree 

WebTalk Terms And Conditions

By checking "I Agree" below, I also confirm that I have read [Section 13 of the terms and conditions](https://www.xilinx.com/products/design-tools/webtalk.html) above concerning WebTalk and have been afforded the opportunity to read the WebTalk FAQ posted at <https://www.xilinx.com/products/design-tools/webtalk.html>. I understand that I am able to disable WebTalk later if certain criteria described in Section 13(c) apply. If they don't apply, I can disable WebTalk by uninstalling the Software or using the Software on a machine not connected to the internet. If I fail to satisfy the applicable criteria or if I fail to take the applicable steps to prevent such transmission of information, I agree to allow Xilinx to collect the information described in Section 13(a) for the purposes described in Section 13(b).

☒ I Agree 

Third Party Software End User License Agreement

By checking "I Agree" below, or OTHERWISE ACCESSING, DOWNLOADING, INSTALLING or USING THE SOFTWARE, I AGREE on behalf of licensee to be bound by the agreement, which can be viewed by [clicking here](#).

☒ I Agree 

Select Edition to Install



Select an edition to continue installation. You will be able to customize the content in the next page.

☒ **Vivado HL WebPACK**



Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

☐ Vivado HL Design Edition

Vivado HL Design Edition includes the full complement of Vivado Design Suite tools for design, including C-based design with Vivado High-Level Synthesis, implementation, verification and device programming. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

☐ Vivado HL System Edition

Vivado HL System Edition is a superset of Vivado HL Design Edition with the addition of System Generator for DSP. Complete device support, cable drivers and Documentation Navigator are included. Users can optionally add Model Composer to this installation.

Vivado HL WebPACK



Customize your installation by (de)selecting items in the tree below. Moving cursor over selections below provide additional information.

Vivado HL WebPACK is the no cost, device limited version of Vivado HL Design Edition. Users can optionally add Model Composer and System Generator for DSP to this installation.

- ☒ Design Tools
 - ☒ Vivado Design Suite
 - ☒ DocNav
- ☒ Devices
 - ☒ Production Devices
 - ☒ SoCs
 - ☒ 7 Series (limited support)
 - ☒ UltraScale (limited support)
 - ☒ UltraScale+ (limited support)
 - ☐ Engineering Sample Devices
- ☒ Installation Options
 - ☒ Install Cable Drivers (You MUST disconnect all Xilinx Platform Cable USB II cables before proceeding)
 - ☒ Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license)
 - ☐ Install WinPCap for Ethernet Hardware Co-simulation
 - ☐ Launch configuration manager to associate System Generator for DSP with MATLAB

Select Destination Directory



Choose installation options such as location and shortcuts.

Installation Options

Select the installation directory

C:\Xilinx

...

Installation location(s)

C:\Xilinx\Vivado\2019.2

C:\Xilinx\DocNav

Disk Space Required

Download Size:	NA
Disk Space Required:	30.35 GB
Final Disk Usage:	25.57 GB
Disk Space Available:	120.03 GB

Select shortcut and file association options

☒ Create program group entries

Xilinx Design Tools

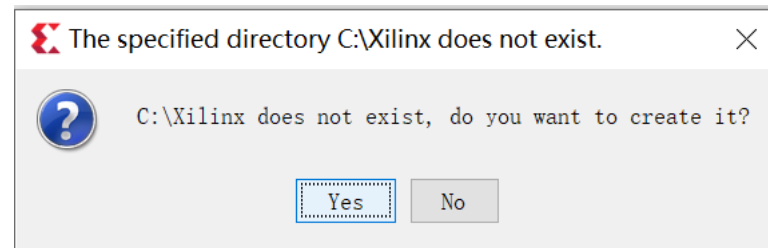
☒ Create desktop shortcuts

☒ Create file associations

Apply shortcut & file association selections to

☒ Current user

☐ All users





Installation Summary

Edition: Vivado HL WebPACK

Devices

- Production Devices (SoCs, 7 Series, UltraScale, UltraScale+)

Design Tools

- Vivado Design Suite (Vivado)
- DocNav

Installation Options


- Enable WebTalk for Vivado to send usage statistics to Xilinx (Always enabled for WebPACK license)
- Install Cable Drivers (You MUST disconnect all Xilinx Platform Cable USB II cables before proceeding)

Installation location

- C:\Xilinx\Vivado\2019.2
- C:\Xilinx\DocNav

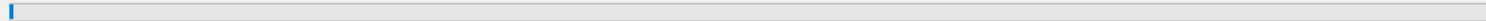
Disk Space Required

- Download Size: NA
- Disk Space Required: 30.35 GB
- Final Disk Usage: 25.57 GB

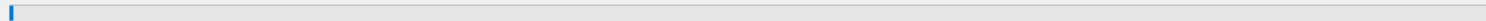
 Vivado 2019.2 Installer - Installation Progress

Installation Progress

Installing files, 0% completed.



Final Processing...



Windows 安全中心



你想安装这个设备软件吗？

名称: Xilinx, Inc.

发布者: Xilinx Inc



始终信任来自 "Xilinx Inc" 的软件(A)。

安装(I)

不安装(N)



你应仅从可信的发布者安装驱动程序软件。[我如何确定哪些设备软件可以安全安装？](#)



Installation Progress

✓ It took 10 minutes to install files.

✓ Done Final Processing.

Xilinx Software Install

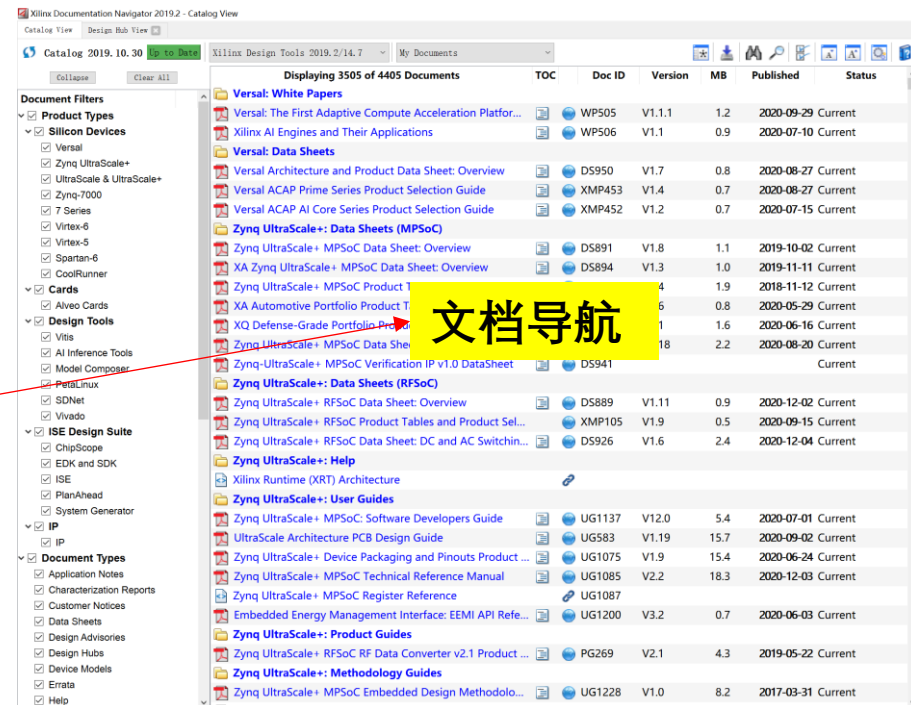
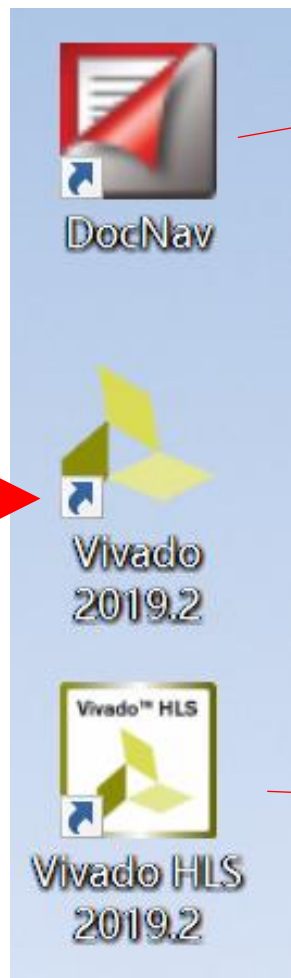


Installation completed successfully.

确定

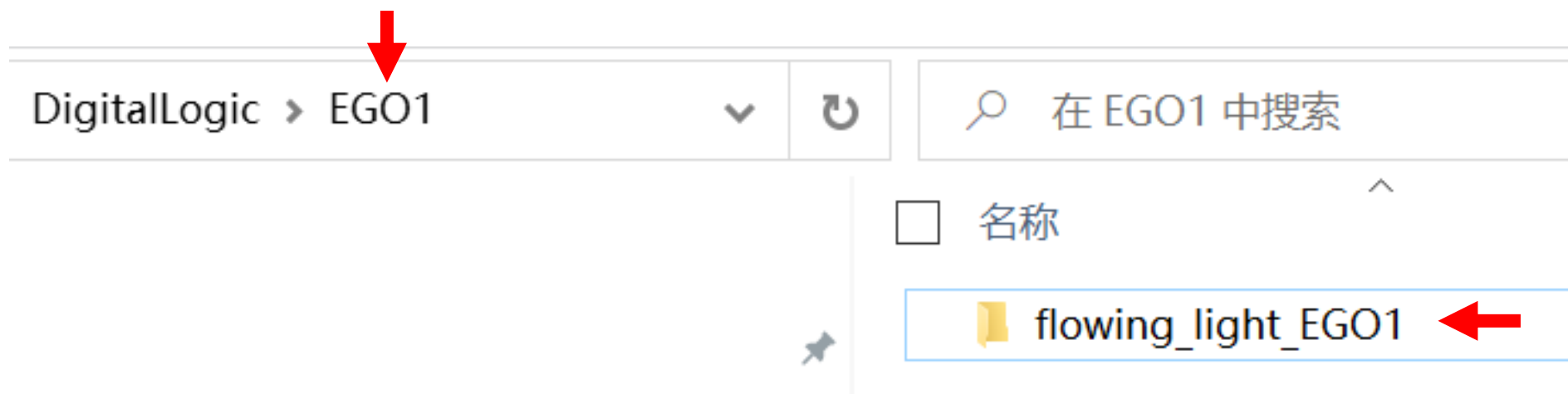
- 安装完成。

普通版本



第三部分：FPGA开发板测试

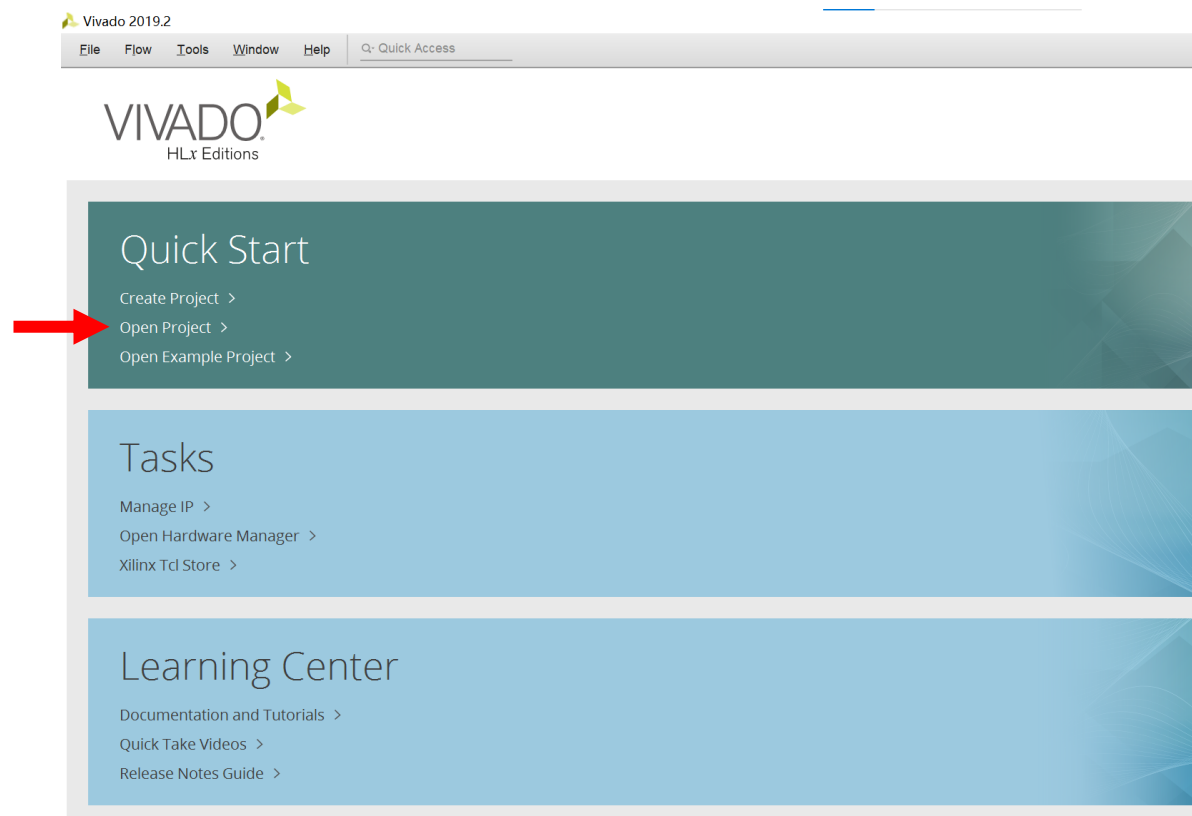
- 第1步：在电脑的硬盘根目录上新建一个目录“\DigitalLogic\EGO1”；将压缩文件“**flowing_light_EGO1.zip**”解压后，拷贝到该目录中：



- 第2步：打开Vivado 2019.2（普通版本），点击“Open Project”，打开一个工程。



普通版本



Open Project

Look in:



- flowing_light_EGO1.cache
- flowing_light_EGO1.hw
- flowing_light_EGO1.ip_user_files
- flowing_light_EGO1.runs
- flowing_light_EGO1.sim
- flowing_light_EGO1.srds
- flowing_light_EGO1.xpr**

Recent Directories

File Preview

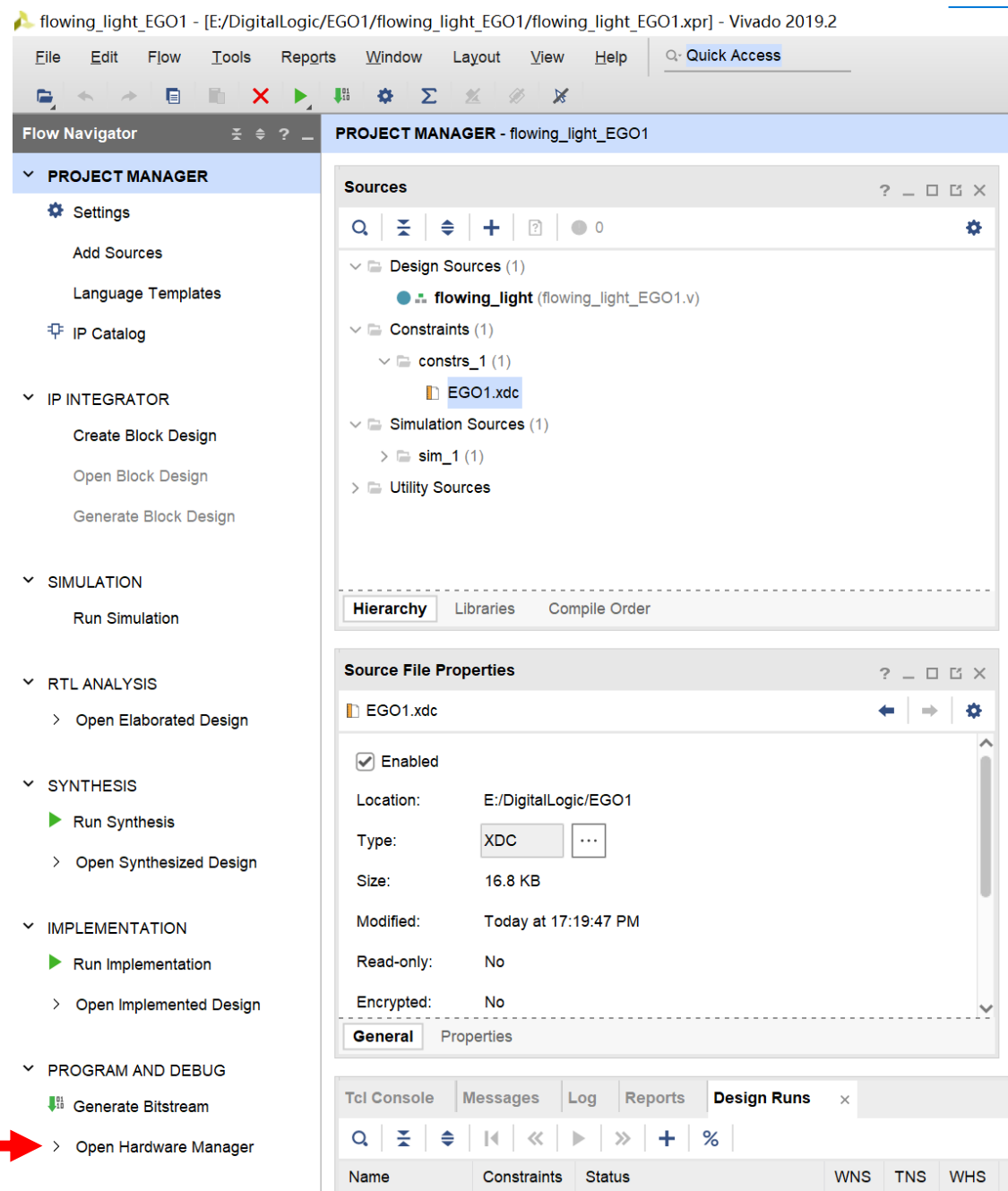
C:/DigitalLogic/ACE1/flowing_light_EGO1
Created: Today at 10:15 AM
Accessed: Today at 17:28 PM
Modified: Wednesday 23/09/27 05:23 PM
Size: 8.9 KB
Type: Vivado project
Version: Vivado v2019.2
Owner: DESKTOP-CSQVPR3\lily2002

File name:

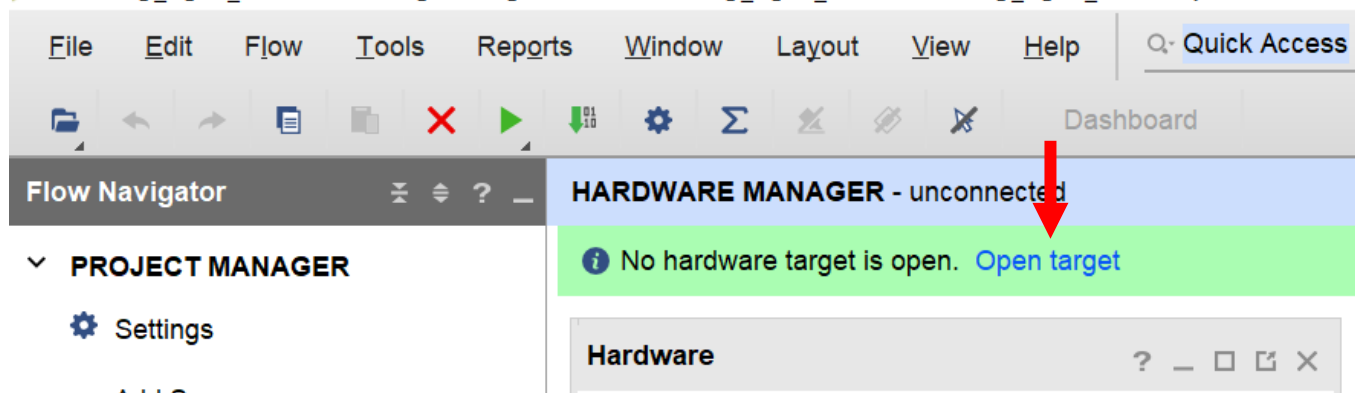
Files of type:

- 第3步：下载bit文件到FPGA开发板

打开硬件管理器

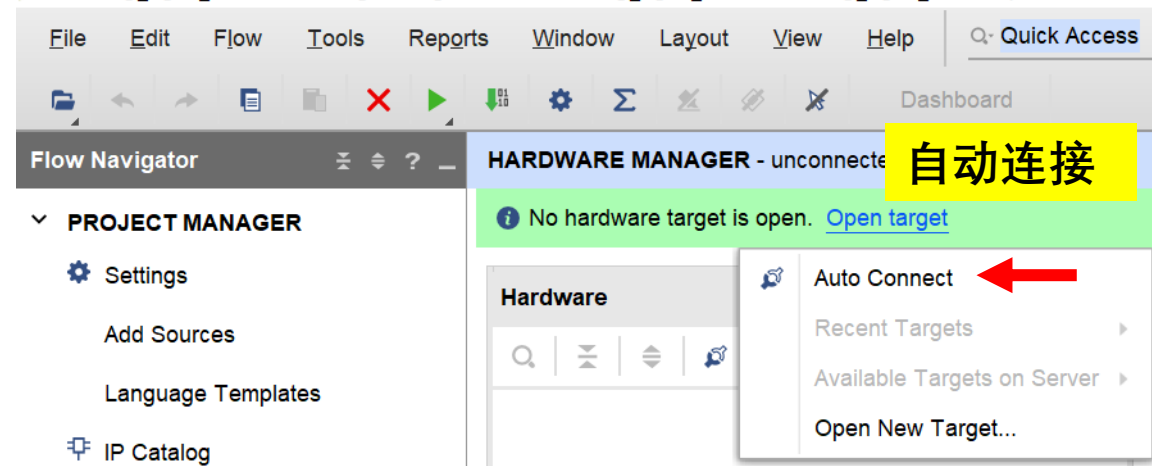


flowing_light_EGO1 - [E:/DigitalLogic/EGO1/flowing_light_EGO1/flowing_light_EGO1.xpr] - Vivado 2019.2

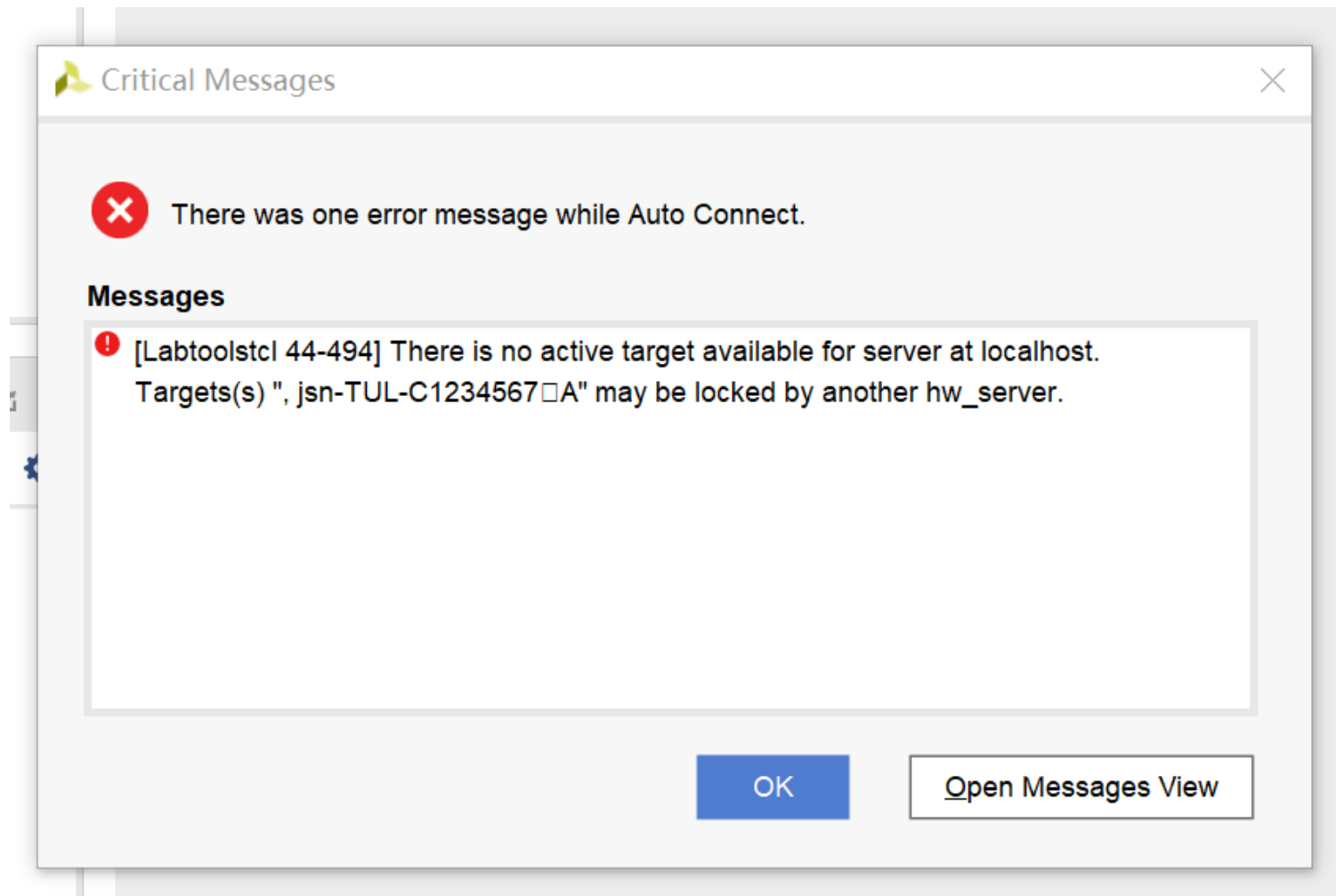


打开目标板 (FPGA开发板)

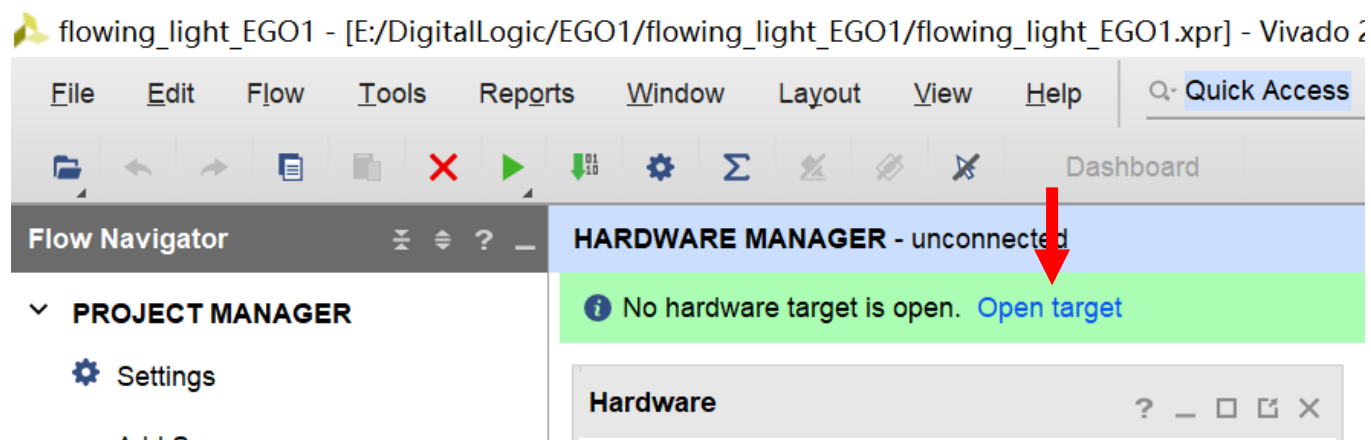
flowing_light_EGO1 - [E:/DigitalLogic/EGO1/flowing_light_EGO1/flowing_light_EGO1.xpr] - Vivado 2019.2



- 如果出现这个界面，请退出Vivado；再运行Vivado，重新打开工程。

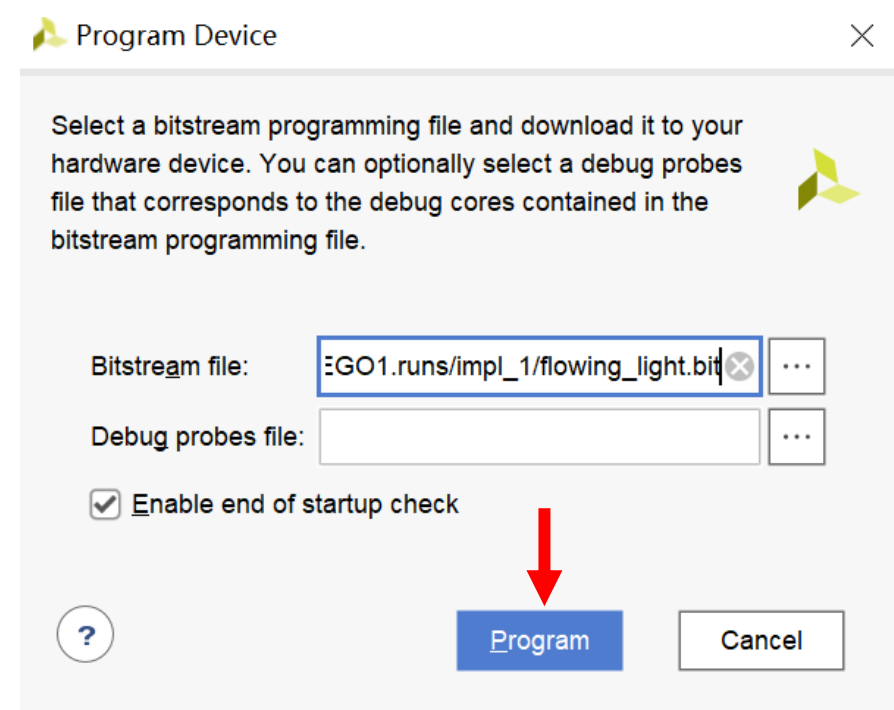
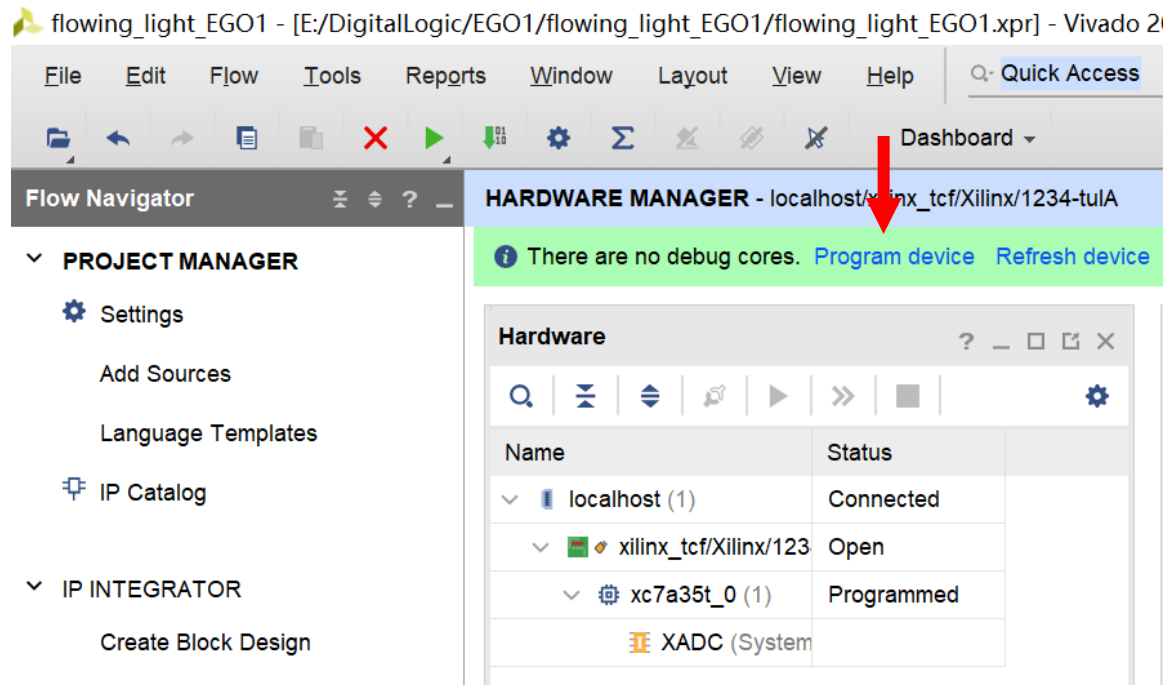


- 如果打开不了目标板（FPGA开发板），请将USB接口从电脑端拨下，再重新插上，然后再打开目标板。



打开目标板（FPGA开发板）

下载bit文件到FPGA开发板



flowing_light_ACE1 - [F:/DigitalLogic/ACE1/flowing_light_ACE1.xpr] - Vivado 2019.2

File Edit Flow Tools Reports Window Layout View Help Q Quick Access

Dashboard

Flow Navigator

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/C1234567 A

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager
 - Open Target
 - Program Device
 - Add Configuration Memory Device

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Xilinx/C12	Open
xc7a75t_0 (2)	Programmed
XADC (System)	
hw_ila_1 (ILA)	Idle

hw_ila_1

Waveform - hw_ila_1

Properties

Select an object to see properties

Settings - hw_ila_1 **Status - hw_ila_1**

Core status: ● ○ ○ ○ ○ Idle

Capture status - Window 1 of 1

Window sample 0 of 2048

Idle

Tcl Console

```
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Xilinx/C1234567 A
set_property PROGRAM_FILE {F:/DigitalLogic/ACE1/flowing_light_ACE1/flowing_light_ACE1.runs/impl_1/flowing_light.bit} [get_hw_device current_hw_device]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a75t_0] 0]
INFO: [Labtools 27-2302] Device xc7a75t (JTAG device index = 0) is programmed with a design that has 1 ILA core(s).
```

也有可能是这个界面

下载bit文件到FPGA开发板

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager
 - Open Target
 - Program Device
 - Add Configuration Memory Device

Add Configuration Memory Device

xc7a75t_0

Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: it_ACE1.runs/impl_1/flowing_light. (X) ...

Debug probes file: ...

☒ Enable end of startup check

Program **Cancel**

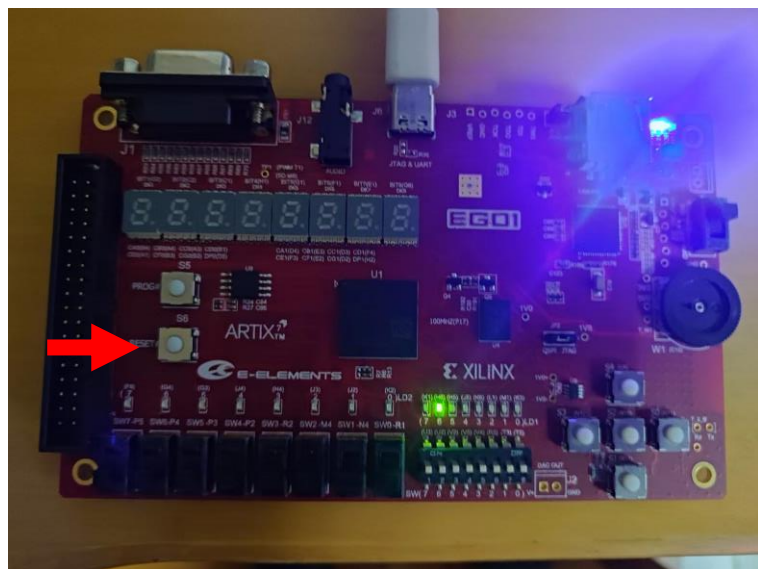
Tcl Console x Messages Serial I/O Links Serial I/O Scans

Q [] [] [] [] [] [] []

```
set_property PROGRAM.FILE {E:/DigitalLogic/EG01/flowing_light_EG01/flowing_light_EG01.runs/impl_1/flowing_light.bit} [get_hw_devices xc7a35t_0]
program_hw_devices [get_hw_devices xc7a35t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
```

下载成功后，Vivado界面显示

此时，EGO1开发板上16个LED灯**从左到右**轮流显示，按复位键后，从最左边开始显示



复位键

flowing_light_EG01.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

timescale 1ns / 1ps

```
module flowing_light(
    input sys_clk_in,
    input sys_rst_n,
    output [15:0] led_pin //拥有16个led_pin灯
);
    reg [23:0] cnt_reg;
    reg [15:0] light_reg;

    always @ (posedge sys_clk_in)
    begin
        if (!sys_rst_n)
            cnt_reg <= 0;
        else
            cnt_reg <= cnt_reg + 1;
    end

    always @ (posedge sys_clk_in)
    begin
        if (!sys_rst_n)
            light_reg <= 16'h0001;
        else if (cnt_reg == 24'hffffff)
            begin
                if (light_reg == 16'h8000)
                    light_reg <= 16'h0001;
                else
                    light_reg <= light_reg << 1; //左移
            end
    end

    assign led_pin = light_reg;
endmodule
```

实现16个LED从左到右循环显示的设计文件（flowing_light_EGO1.v）

```
`timescale 1ns / 1ps           //定义仿真时间单位（1ns）
                                //和时间精度（1ps）

module flowing_light(           //定义模块flowing_light
    input sys_clk_in,           //时钟输入
    input sys_rst_n,            //复位按键
    output [15:0] led_pin       //16个LED灯
);
    reg [23 : 0] cnt_reg;        //定义寄存器变量（reg）
    reg [15 : 0] light_reg;

    always @ (posedge sys_clk_in) //时钟上升沿触发
    begin
        if (!sys_rst_n)
            cnt_reg <= 0;        //按复位键，计数清0
        else
            cnt_reg <= cnt_reg + 1; //计数加1，实现延时
    end
```

```
always @ (posedge sys_clk_in)
begin
    if (!sys_rst_n)                //按复位键，则显示0001H，
        light_reg <= 16'h0001;     //即最左边的LED亮
    else if (cnt_reg == 24'hffffff) //如果延时到，则改变显示值
    begin
        if (light_reg == 16'h8000) //如果显示值=8000H（最右边
            light_reg <= 16'h0001; //的LED灯），则置为0001H
                                   //（最左边的LED灯）
        else
            light_reg <= light_reg << 1; //左移
    end
end

    assign led_pin = light_reg;    //将显示值送LED灯

endmodule                          //模块结束
```


第四部分： Verilog HDL硬件描述语言（自学）

(一) Verilog HDL硬件描述语言概述

- 硬件描述语言（HDL）是用来描述硬件各个组成模块以及它们之间关系的语言。当前广泛使用的硬件描述语言包括VHDL以及Verilog HDL。相对来说，Verilog HDL在实际的生产实践中使用得更多。在本实验教材中，就使用Verilog HDL语言来描述各个实验的设计。
- 由于Verilog HDL语言有很多语法借鉴了非常广泛使用的C编程语言，因此相对来说学习Verilog HDL硬件描述语言是非常容易的。
- **HDL**: Hardware Description Language，硬件描述语言。
- **VHDL**: **V**ery-High-Speed Integrated Circuit **H**ardware **D**escription **L**anguage，超高速集成电路硬件描述语言。
- **Verilog HDL**: 由美国Gateway Design Automation公司（该公司于1989年被美国Cadence公司收购）的工程师于1983年末创立的硬件描述语言。

- 在没有**硬件描述语言**（HDL）之前，设计硬件的时候往往使用绘制电路图的方式。实际上，电路图方式仍然是非常重要的设计硬件电路的方法，特别是在设计硬件电路板，将多个芯片进行组合的时候。
- 随着电子技术的发展以及大规模集成电路的规模与复杂程度的不断扩大，图形化的描述受到了极大的限制，已经不太可能跟得上所需要设计硬件的复杂性。在这种情况下，硬件描述语言（HDL）应运而生，用以描述硬件的各个组成模块以及它们之间的连接关系。硬件描述语言（HDL）可以被认为是与电路的图形化描述等价的描述，在当前的硬件设计领域占到了主导地位。相对于图形化的电路设计方法，**硬件描述语言**（HDL）具有以下**特点与优势**：
 - ① 硬件描述语言的抽象程度高，方便进行精确的描述。
 - ② 可以应对不同层次的设计，方便进行自底向上和自顶向下进行硬件设计。
 - ③ 具有易于修改的特点，这一点尤为对图形描述有优势。
 - ④ 可以适合更大规模硬件的设计，模块化的方法使得各个部分可以协同工作。
 - ⑤ 匹配硬件描述语言的设计工具以及文档现在都非常丰富。
 - ⑥ 硬件描述语言本身就是对于硬件最好的说明文档。
- 有一定的编程基础，特别是有了C语言的经验的话，学习Verilog HDL会比较容易。相对来说，**Verilog HDL**编程语言的有以下**特点**：
 - ① 语法简洁，使用和学习都比较方便。
 - ② 功能强大，适应于各个层面的电路设计。
 - ③ 支持丰富，兼容性强。
 - ④ 易于复用，使用内建的元件方便。
 - ⑤ 机制灵活，易于扩展。

(二) Verilog HDL的模块程序基本结构

- 例：一个2输入的与门的逻辑描述

```
module and2x(a,b,r);  
    input a,b;  
    output r;  
    wire a,b,r;  
    assign r=a & b;  
endmodule
```

//wire表示（连线）信号类型，表示1位数据
//assign表示持续赋值语句

- 总结来看，以下几个元素是在Verilog HDL语言中最基本的组成部分：
 - ① 模块的声明和结束，格式如下：**module**模块名称（端口 1，端口 2，端口 3，……）;模块以**endmodule**作为结束。这里的模块名称可以任取为符合要求的标识符。
 - ② 端口的定义：对于模块的每一个端口都需要定义出是输入端口（**input**），还是输出端口（**output**），还是双向端口（**inout**）。不仅要定义端口的输入输出类型，还需要定义端口的信号类型。输入和双向端口不能被定义为寄存器（reg，见下）类型。
 - ③ 信号类型的声明：在本实验教程中，信号类型声明只使用了两类，即寄存器类型**reg**以及连线类型**wire**。在Verilog HDL语言中定义的各种信号的类型实际上都对应着具体电路中的各种电路连接以及电路实体。如果没有任何声明的话，默认类型就是连线wire类型。为了书写方便，Verilog HDL还可以将定义和类型说明放在一起。并且可以直接放置在模块声明的端口列表中。例如，上述例子的类型声明可以使用下面的形式：`input wire a,b; output wire r;`或者直接放到端口列表中：`module and2x(input wire a,b,output wire r);`因为wire是默认的，这里的所有的wire都可以省略。
 - ④ 模块的功能定义：这部分是编写模块最为主要的部分，这将定义当前模块所能够完成的功能。可以有多种方法来完成任务。上述的例子给出了一个最为基本的方法，即通过**assign**语句来进行持续的赋值。这个语句通常被用来进行组合逻辑的设计，在描述上也很明显，即持续对输入进行响应。当然，在实际的工作过程中，上述的逻辑还需要考虑一定的延迟，所有的电路都会有一定的延迟。延迟在编写程序的时候体现不出来，但是在综合出电路的时候会有很大的影响，相同的功能低延迟的电路会获得更高的性能。如何编写一个更加低延迟而且高效的电路需要长期的经验。

(三) Verilog HDL的语言元素

- 1、标识符

- 例：合法的标识符举例

a
total
_delay
_d1_c2
D\$
SOURCE

- 例：非法的标识符举例

1total	//以数字开头
number#	//使用了非法的字符
\$5	//以\$开头

- 2、参数

- parameter 参数名 1=表达式 1, 参数名 2=表达式 2……;
- 例如: **parameter** WIDTH=16

• 3、编译指导语句

- 常用的编译指导语句包括`define 宏替换语句，`include 文件包含语句，`ifdef, `else, `elsif, `endif 条件编译语句。
- ``define WIDTH 16`
`reg[WIDTH:1] r;`
这就与 `reg[16:1]` 相当
- ``define sum a+b`
定义之后，可以使用:
`assign res=sum;`
来获得将两个 a 和 b 信号相加的效果，因为最终会被字符串替换，获得语句:
`assign res=a+b;`
- ``include "adder.v"`
- ``include "../common/adder.v"`
- 例：条件编译语句
``define sum a+b`
``ifdef sum`
`assign res=sum;`
``else`
`assign res=a+b;`
``endif`

(四) Verilog HDL中的数据

- 整数的常量是按照一定的格式写出的，下面是格式规范：

+/- <位宽> ' <进制><数字> //4部分

通常也表示为：

+/- <size> ' <base><value> //4部分

- +/-：表示数的符号（正数、负数）
 - <size>：表示数的位数
 - <base>：为进制的情况，包括了二进制（b 或者 B），十进制（d 或者 D），八进制（o或者 O）以及十六进制（h 或者 H）
 - <value>：为数值
- 例：整数数值举例

8'b01001010

16'H45EF

-8'D123

-16'o3333

- 数据取值：

- 除了0和1的这样两个常用的信号取值之外，有两个比较重要的即为x（或者X）和z（或者Z）。
- x或者X的取值一般表明为**不确定**，或者未知的逻辑状态，用于不关心对应信号值的情况，不影响整个逻辑电路的功能。
- z或者Z代表**高阻态**。在处理器设计中，高阻态一个非常典型的应用就是用于对内存的输入时序（从内存中读数据到处理器内部）。在进行数据输入的时候，先将处理器引脚的赋值状态置于z即高阻态，之后经过一定的时间延迟，就可以从对应的引脚处获得外部内存的输入值。

- 数据类型：

- **wire**类型代表了在硬件电路中的**连线**，其特征就是输出的值紧随着输入值的变化而变化。**reg**数据类型会放到过程语句中（例如always），通过过程赋值语句进行赋值。reg类型不一定必然会对应到硬件的寄存器，综合的时候会依据实际的情况使用连线或者寄存器来完成reg数据类型的功能。
- 数据类型还有**向量**和**标量**的区别：
 - wire[7:0] data; //data是一个8位的连线；向量。
 - reg[31:0] res; //res是一个32位的数据变量；向量。
 - l = data[7]; //获取data的最高位；标量。
 - lob = res[7:0]; //获取数据res中的最低8位，即最低一个字节；向量。

(五) Verilog HDL中的运算

• 1、位运算符

- 位运算符是最基本的运算符，从位运算符上开发人员直接可以想象得出最终的电路是如何通过逻辑门来实现的。与所有的编程语言一样，位运算符表达了两个操作数对应的位进行位运算的结果。在Verilog HDL中的运算符包括以下的操作：（在下面的例子中，假设 $a=8'b00001001$ ， $b=8'b01010101$ 。）

- ① \sim 按位进行取反操作。 $\sim a=8'b11110110$ 。
- ② $\&$ 按位进行与操作。 $a\&b=8'b00000001$ 。
- ③ $|$ 按位进行或操作。 $a|b=8'b01011101$ 。
- ④ \wedge 按位进行异或操作。 $a\wedge b=8'b01011100$ 。
- ⑤ $\wedge\sim$ 或者 $\sim\wedge$ 这两个运算符都是按位同或操作。 $a\wedge\sim b=8'b10100011$ 。
- ⑥ $>>$ 右移运算符。 $a>>2=8'b00000010$ ；右移2位。
- ⑦ $<<$ 左移运算符。 $a<<2=8'b00100100$ ；左移2位。

• 2、**缩位运算符**

- 在位运算符中还有一类特殊的运算符需要注意一下，即缩位运算符，值得单独提出来作为一类描述。
缩位运算符可以将一个向量按照一定的运算“缩”成1位，因此被称为是缩位运算符。缩位运算符有下面的几个：

- ① & **与**
- ② ~& **与非**
- ③ | **或**
- ④ ~| **或非**
- ⑤ ^ **异或**
- ⑥ ^~, ~^ **同或**

- 由于缩位运算符能够将多个位缩成一位，其表达式即可以写为操作符后面跟着一个向量的操作数。
- 例如：reg [7:0] value;
- 如果 value = 7'b01010101。 则最终：
- &value 结果为 0， |value 结果为 1， ~^value 结果为 1。

不好理解？

value怎么是7位？

&value 结果怎么为 0， |value 结果怎么为 1， ~^value 结果怎么为 1

• 3、关系和逻辑运算符

- 与位运算符直接相关的操作就是关系和逻辑运算符。这些运算符的直接应用就是应用在条件判断上（例如使用在if语句中）。下面是常用的关系和逻辑运算符，这些运算符的含义与其它语言的相同。这些运算符的取值结果可以被认为是true（1位的逻辑值1）或者false（1位的逻辑值0）。

- ① && 逻辑与运算符。
- ② || 逻辑或运算符。
- ③ ! 逻辑非运算符。
- ④ < 关系运算符小于。
- ⑤ <= 关系运算符小于或者等于。
- ⑥ > 关系运算符大于。
- ⑦ >= 关系运算符大于或者等于。
- ⑧ == 关系运算符等于。
- ⑨ != 关系运算符不等于。
- ⑩ === 关系运算符全等。
- ⑪ !== 关系运算符不全等。

- 相等运算符（==）在进行比较的时候，需要按每位进行比较，只有所有的位都相等的时候，最后的结果值才会是true。但是如果其中的某一位是高阻态（用Z表示）或者是不定值（用X表示），那么最终的结果是不定值，这一点是需要值得注意的。对于全等（===）来说，对于这些高阻态或者不定值也需要进行比较，只有完全一致才会获得true的结果。因此，在进行真正比较的时候，需要注意这一点来选择合适的运算符。

A	B	A==B	A===B
4b1101	4b1101	1	1
4b1100	4b1101	0	0
4b110Z	4b110Z	X	1
4b11XX	4b11XX	X	1

- 4、**算术运算符**

- 算术运算符是常用的运算符，包括了加（+）、减（-）、乘（×）、除（÷）运算符，这些运算符可以用在整数的运算中。还有一个算术运算符与 C 语言中的运算符一致，即**求余运算符**（%）。

- 5、**条件运算符**

- 在C语言中有一个需要三个参数的运算符，即条件运算符“condition? result1 : result2”。这个运算符在Verilog HDL中也存在，也完成了相同的功能，即 **signal = condition ? expression1 : expression2**。如果condition为true的话，signal取expression1的值，否则取expression2的值。

- 6、**位拼接运算符**

- 还有一个重要的运算符即位**拼接运算符**“{ }”。这个运算符能够很方便把多个信号拼接为向量的形式。位拼接运算符的使用也非常方便，只需要把信号排列在大括号中间即可。**{a[3:0], b[7:6], c}**代表了将a的第3至第0位，b的第7位和第6位，以及信号c拼接在一起，构成一个新的信号向量。

(六) Verilog HDL的行为语句

- Verilog HDL的行为语句与其它语言的语法功能（可以将前面的运算功能当成其它语言中的表达式）相当，包括了赋值语句、过程语句、条件语句、编译指导语句等。这些语句构成了Verilog HDL中对于功能描述的基本模式。
- 在Verilog HDL中的可综合的行为语句主要包括以下几个部分：
 - ① always过程语句；
 - ② 使用begin-end组合起来的语句块；
 - ③ 可以进行持续赋值的语句assign；
 - ④ 阻塞的过程赋值语句=，非阻塞的过程赋值语句<=；
 - ⑤ for循环语句。

• 1、always过程语句

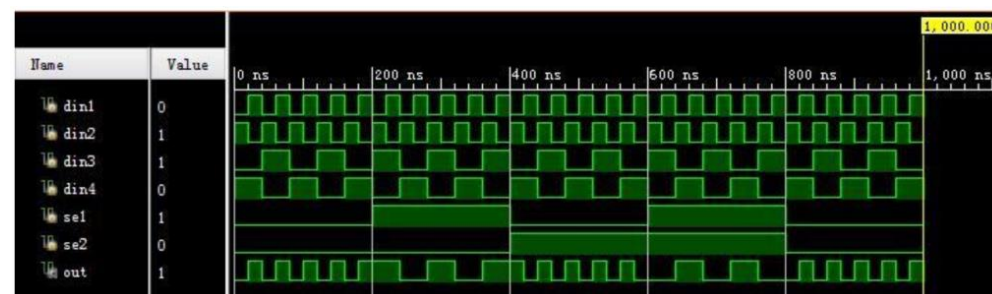
- always@(敏感信号列表)
语句（可以是一条语句，或者是语句块）

- always@(敏感信号列表)
begin
 //本过程的功能描述
end

- 例：使用always过程语句描述四选一数据选择器

```
module mux4_1(din1, din2, din3, din4, se1, se2, dout);  
input din1, din2, din3, din4, se1, se2;  
output reg dout;  
always @(din1 or din2 or din3 or din4 or se1 or se2)  
    case({se1,se2})  
        2'b00 : dout=din1;  
        2'b01 : dout=din2;  
        2'b10 : dout=din3;  
        2'b11 : dout=din4;  
    endcase  
endmodule
```

//输出一般是reg数据类型



可以用：always @(din1, din2, din3, din4, se1, se2) 或 always @(*) 或always @* 代替：always @(din1 or din2 or din3 or din4 or se1 or se2)

- 敏感信号的类型：
 - 敏感信号可以分为两个类型，一个是电平敏感型，一个是边沿敏感型。
 - 电平敏感型在发生电平变化，从0变成1或者从1变成0的时候，always过程语句会依据变化完成的值执行一次。
 - 而边沿敏感型则可以进一步分为上升沿触发或者下降沿触发。在发生了一次上升沿事件，或者下降沿事件的时候，触发always过程语句的执行。在Verilog HDL中，使用posedge指定上升沿，使用negedge指定下降沿。
 - 可以将边沿敏感类型的信号放置到always过程块的敏感信号列表中，下面是一个常用的always过程的敏感信号列表。
 - always @(posedge clk) //时钟上升沿
 - 这里响应的是一个时钟clk的上升沿信号，一旦一个时钟的上升沿发生，那么下面的always过程语句就发生执行。



上升沿

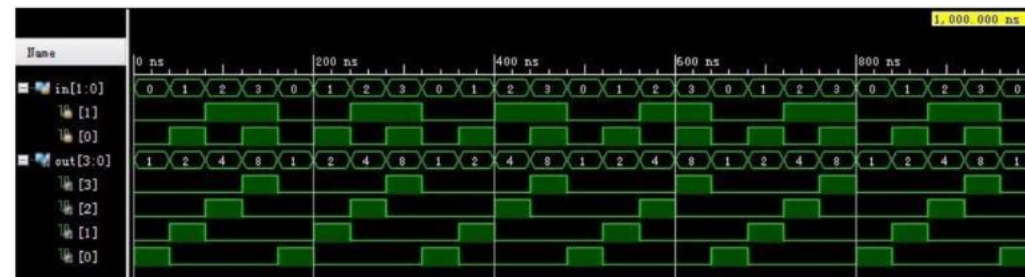


下降沿

- 2、begin/end块语句

- 例：使用begin/end构造语句块（2-4译码器）

```
module decoder2_4(din,dout);  
input [1:0] din;  
output reg [3:0] dout;          //输出一般是reg数据类型  
always @(din)  
begin  
    dout = 4'b0000;  
    case(din)  
        2'b00 : dout = 4'b0001;  
        2'b01 : dout = 4'b0010;  
        2'b10 : dout = 4'b0100;  
        2'b11 : dout = 4'b1000;  
    endcase  
end  
endmodule
```



• 3、赋值语句

- 赋值语句包括了**持续赋值语句**和**过程赋值语句**。持续赋值语句在过程外使用，与过程语句并行执行。过程赋值语句在过程内使用，串行执行，用于描述过程的功能。
- 在Verilog HDL中使用**assign**作为**持续赋值语句**使用，用于对wire类型的变量进行赋值。其对应的硬件也非常好理解，即通过对输出进行赋值，当输入变化的时候，经过一定的延迟，输出就会按照assign所描述的那样发生变化。
- 例如：**assign res = input_a & input_b;** //持续赋值语句
- 在这个例子中，输入input_a和input_b，输出res都是wire类型的变量。当两个输入的任意一个发生变化的时候，输出res都会发生变化。
- 在过程（always）里面的赋值语句被称为是**过程赋值语句**，一般用以对reg类型的变量进行赋值。过程赋值语句分为两种类型，一个是**非阻塞赋值语句**（<=），一个是**阻塞赋值语句**（=）。它们之间的区别是：
 - ① **非阻塞non-blocking赋值语句**（<=）在赋值语句出现的地方**不是立即发生的**，而是等到整个过程块结束的时候才发生。由于不是立即发生的，在过程内的描述中，仿佛这条语句不存在一样，因此被称为是非阻塞的。只是在过程的最后会执行所有的非阻塞赋值语句，在这个执行的过程中，所有的右值会维持原来的值不变。
 - ② **阻塞blocking赋值语句**（=）在赋值语句出现的地方就**立即完成赋值操作**，左值立刻发生变化。一个块语句中存在多条阻塞赋值语句的话，这些阻塞赋值语句会按照先后顺序关系，一条一条的执行，前面的赋值语句没有执行完，后面的赋值语句不会执行。这样的一种行为模式，就跟网络IO编程中的阻塞函数调用方式一样，一定要完成函数执行之后，这个函数调用才会退出。

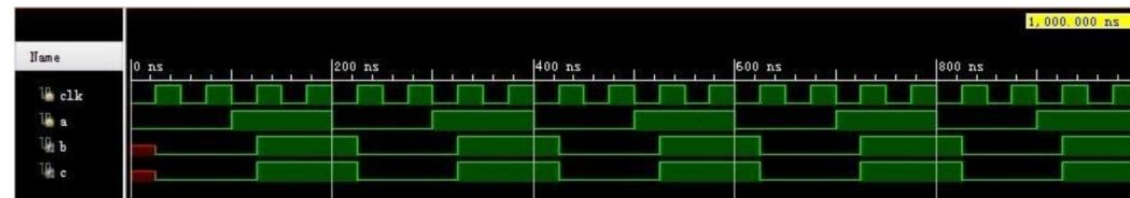
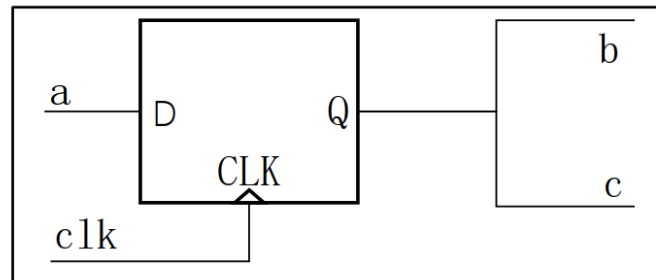
- 例：阻塞赋值语句的使用

```

module blocking(clk, a, c);
    input clk,a;
    output reg c;      //输出一般是reg数据类型
    reg b;             //b是中间变量
    always @(posedge clk)
    begin
        b = a;
        c = b;
    end
endmodule

```

立即完成赋值操作



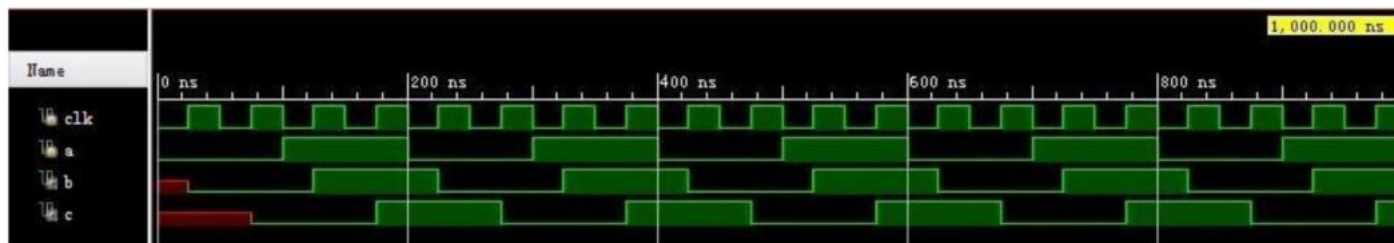
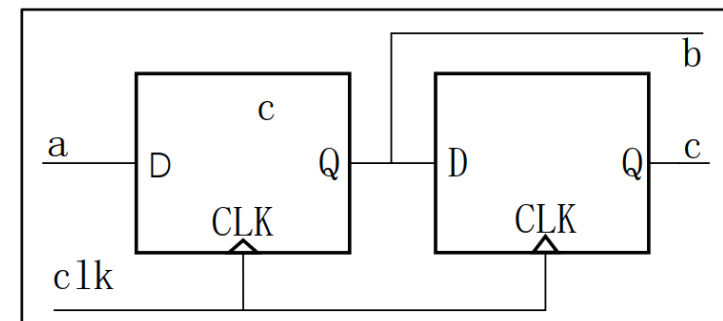
- 例：非阻塞赋值语句的使用

```

module nonblocking(clk, a, c);
    input clk,a;
    output reg c;
    reg b;
    always @(posedge clk)
    begin
        b <= a;
        c <= b;
    end
endmodule

```

赋值操作不是立即发生的，而是等到整个过程块结束的时候才发生



• 4、条件语句

- 例：使用if-else语句实现2-4译码器

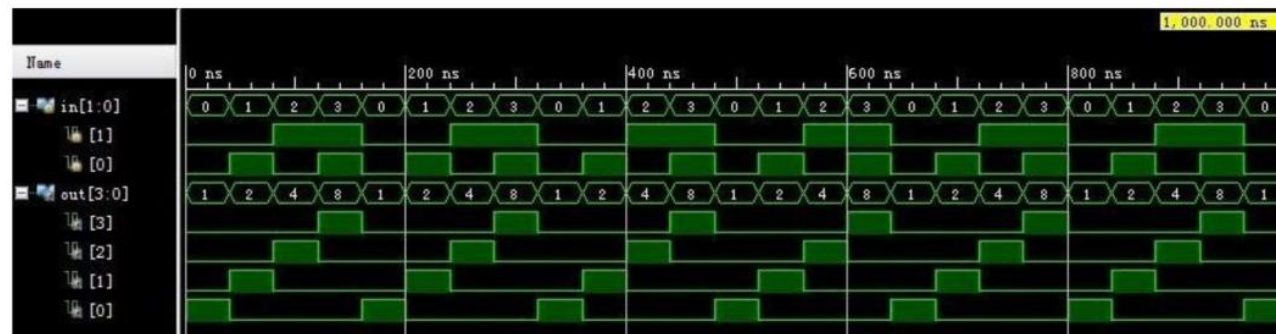
```
module decoder2_4(din,dout);  
input [1:0] din;  
output reg [3:0] dout;  
always @(din)  
begin
```

```
    dout = 4'b0000;  
    if (din == 2'b00)  
        dout = 4'b0001;  
    else if (din == 2'b01)  
        dout = 4'b0010;  
    else if (din == 2'b10)  
        dout = 4'b0100;  
    else if (din == 2'b11)  
        dout = 4'b1000;
```

```
    end  
endmodule
```

- **case** (敏感表达式)
 条件判断1: 语句1;
 条件判断2: 语句2;

 条件判断n: 语句n;
 default: 语句n+1
endcase



(1) if

(2) if
else

(3) if
else if
else if
.....
else if

(4) if
else if
else if
.....
else

• 5、循环语句

- **for** (循环变量赋初值; 循环结束条件; 循环变量增值) 执行语句;

- **repeat**(循环次数的表达式)

begin

语句或者语句块

end

//单个语句不需要begin和end

- **while**(循环执行的条件表达式)

begin

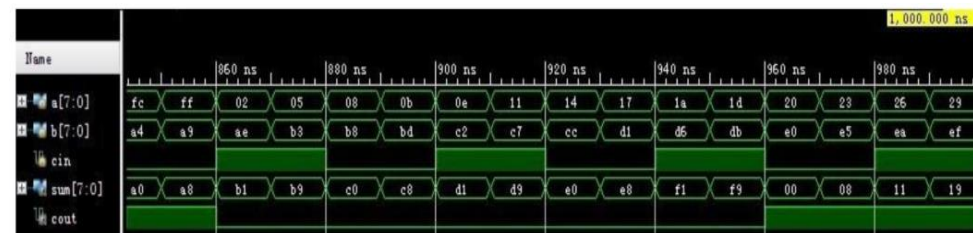
语句或者语句块

end

//单个语句不需要begin和end

- 例子: for 循环语句 (全加器)

```
module for_adder(a,b,cin,sum,cout);  
    input [7:0] a,b; input cin;  
    output reg [7:0] sum; output reg cout;  
    reg c; integer i; //中间变量  
    always @*  
    begin  
        c = cin;  
        for (i = 0; i < 8; i++)  
            begin  
                {c,sum[i]} = a[i] + b[i] + c;  
            end  
        cout = c;  
    end  
endmodule
```



(七) Verilog HDL的设计层次与风格

- 例：1位全加器的门级结构描述

```
module full_adder1(a,b,cin,sum,cout);
```

```
    input a,b,cin;
```

```
    output sum,cout;
```

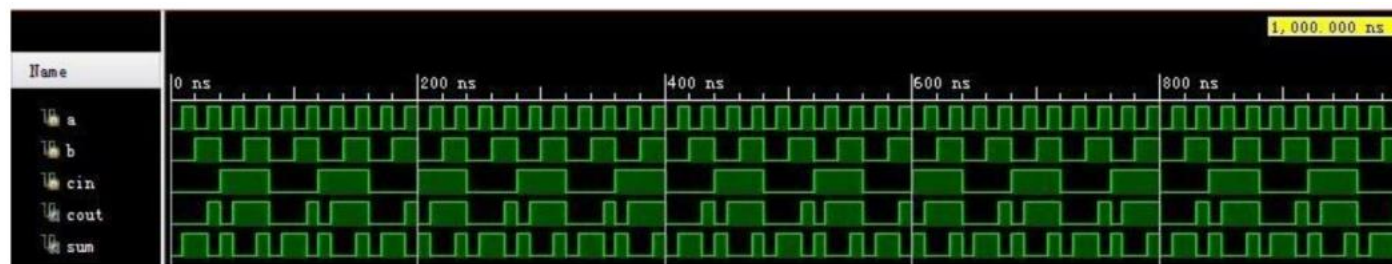
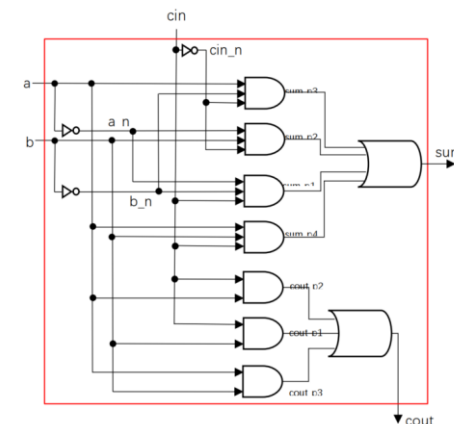
```
    wire a_n, b_n, cin_n, sum_p1,sum_p2,sum_p3,sum_p4, cout_p1, cout_p2, cout_p3;
```

```
    not(a_n,a),(b_n,b),(cin_n,cin);           //3个非门
```

```
    and(sum_p1,a_n,b_n,cin),(sum_p2,a_n,b,cin_n),(sum_p3,a,b_n,cin_n),(sum_p4,a,b,cin),(cout_p1,b,cin),(cout_p2,a,cin),(cout_p3,a,b);           //7个与门
```

```
    or(sum,sum_p1,sum_p2,sum_p3,sum_p4),(cout,cout_p1,cout_p2,cout_p3);           //2个或门
```

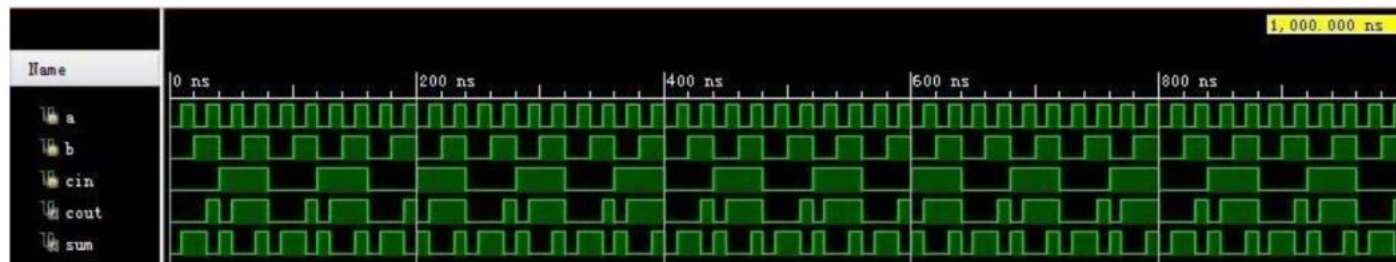
```
endmodule
```



- 例：1位全加器的数据流级描述

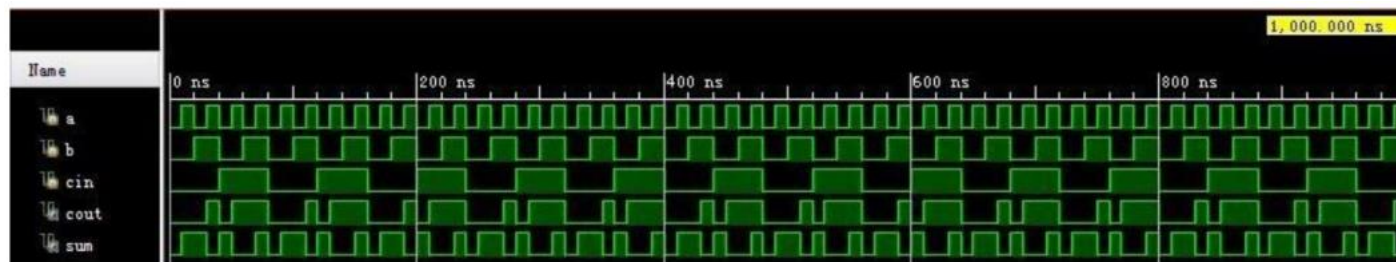
```
module full_adder1(a,b,cin,sum,cout);
    input a,b,cin;
    output sum,cout;
    assign sum=(~a&~b&cin)|(~a&b&~cin)|(a&~b&~cin)|(a&b&cin);
    assign cout=(b&cin)|(a&cin)|(a&b);
endmodule
```

$$\begin{aligned} \text{CarryOut} &= (\neg A * B * \text{CarryIn}) + (A * \neg B * \text{CarryIn}) + (A * B * \neg \text{CarryIn}) + (A * B * \text{CarryIn}) \\ &= (B * \text{CarryIn}) + (A * \text{CarryIn}) + (A * B) \\ \text{Sum} &= (\neg A * \neg B * \text{CarryIn}) + (\neg A * B * \neg \text{CarryIn}) + (A * \neg B * \neg \text{CarryIn}) + (A * B * \text{CarryIn}) \end{aligned}$$



- 例：1位全加器的行为级描述

```
module full_adder1(a,b,cin,sum,cout);
    input wire a, b,cin;
    output reg sum,cout;
    always @*
    begin
        {cout,sum}=a+b+cin;
    end
endmodule
```

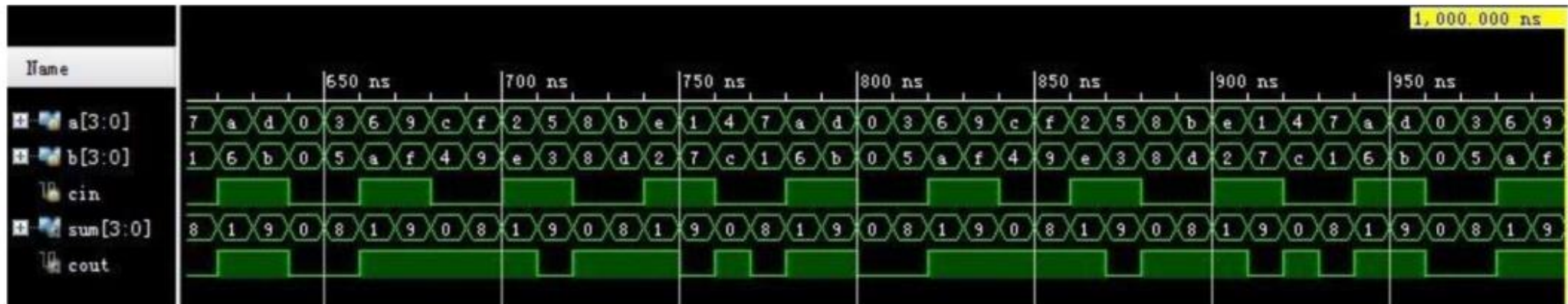


- 例：4位全加器的行为级描述

```

module full_adder4(a,b,cin,sum,cout);
    input cin;
    input [3:0]a,b;
    output [3:0]sum;
    output cout;
    full_adder1 a0(a[0],b[0],cin,sum[0],cin1);
    full_adder1 a1(a[1],b[1],cin1,sum[1],cin2);
    full_adder1 a2(a[2],b[2],cin2,sum[2],cin3);
    full_adder1 a3(a[3],b[3],cin3,sum[3],cout);
endmodule

```



Thanks