1. 编译 DE1 内核

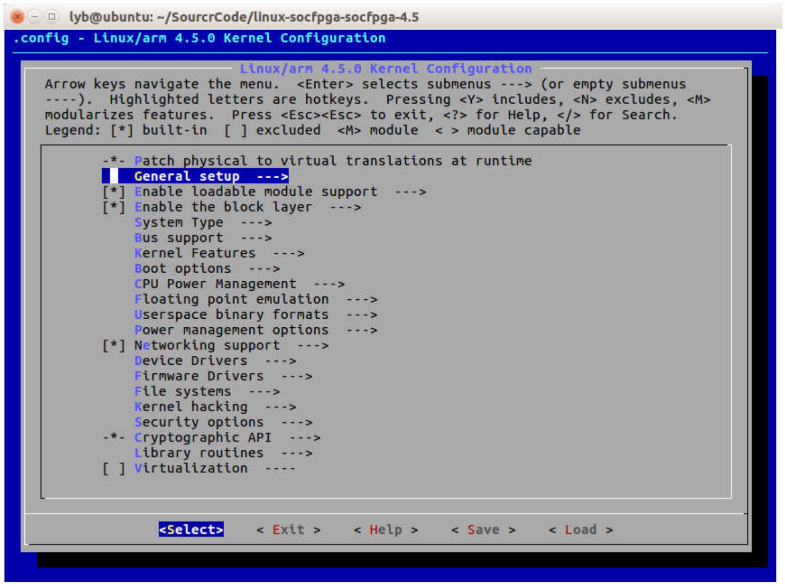
DE1-SoC 和 DE10-NANO 开发板在友晶网站有对应的几个镜像，这些镜像能够提供绝大部分的设备支持。一般情况下只需要下载官网的镜像烧录就能使用，不需要重新编译内核。编译内核一般是设备过旧或者需要实现一些自己定制的功能。

参考《编译 DE1-SOC 匹配的 linux 内核方法.doc》，补充几张图片及几点细节。

0.1 勾上可对 altera FPGA 支持的选项

进入 Linux 内核源文件第一个目录，打开终端执行：make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

出现图形化窗口：



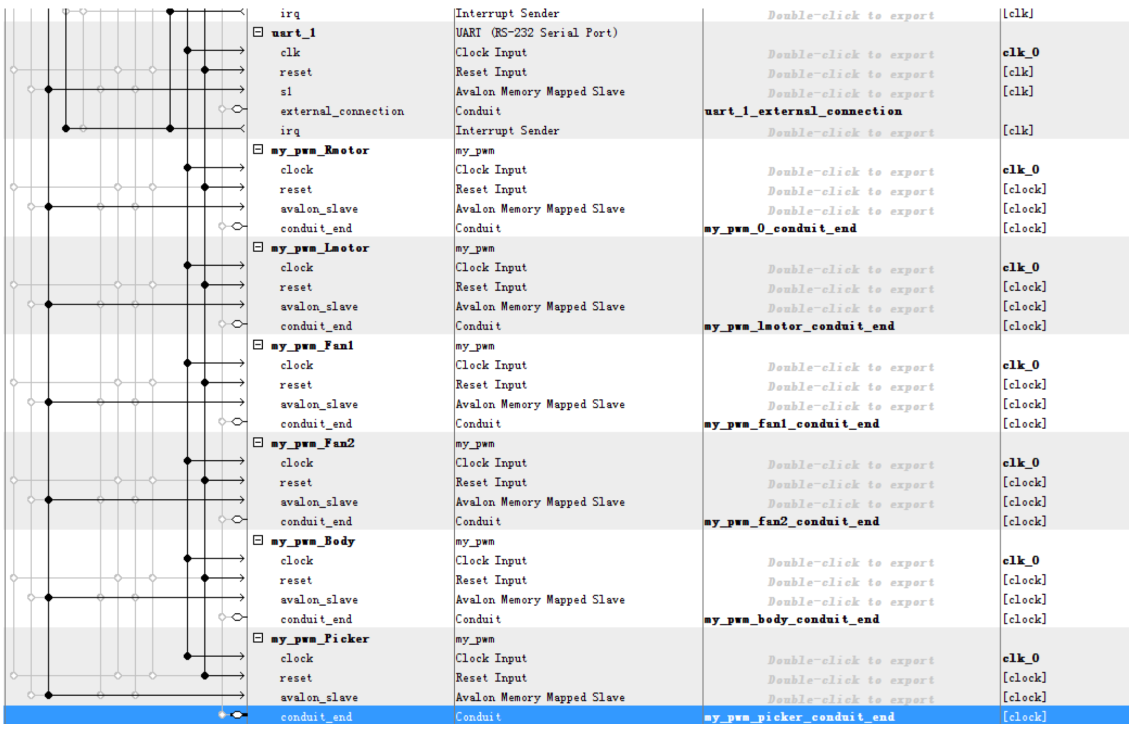
```
lyb@ubuntu: ~/SourcrCode/linux-socfpga-socfpga-4.5
.config - Linux/arm 4.5.0 Kernel Configuration
                    Linux/arm 4.5.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus
  ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.
  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        -*- Patch physical to virtual translations at runtime
            General setup  --->
        [*] Enable loadable module support  --->
        [*] Enable the block layer  --->
            System Type  --->
            Bus support  --->
            Kernel Features  --->
            Boot options  --->
            CPU Power Management  --->
            Floating point emulation  --->
            Userspace binary formats  --->
            Power management options  --->
        [*] Networking support  --->
            Device Drivers  --->
            Firmware Drivers  --->
            File systems  --->
            Kernel hacking  --->
            Security options  --->
        -*- Cryptographic API  --->
            Library routines  --->
        [ ] Virtualization  ----

            <Select>    < Exit >    < Help >    < Save >    < Load >
```

0.2 勾上对罗技 USB 摄像头的支持选项。参考《编译 DE1-SOC 匹配的 linux 内核方法.doc》

2. 在 quartus 中设计硬件

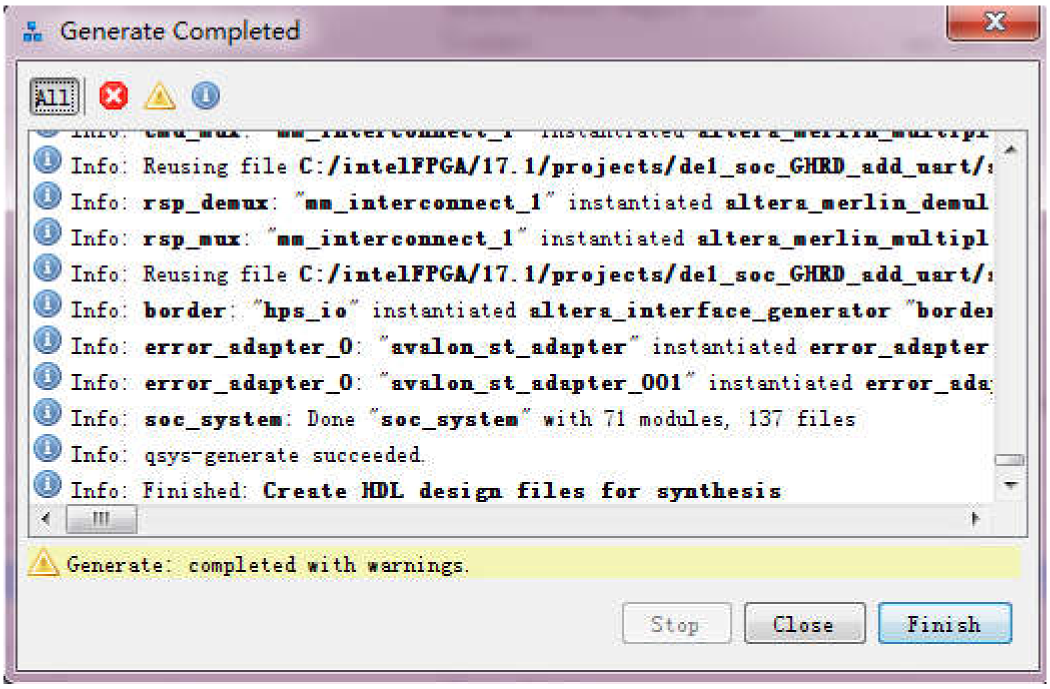
Quartus 版本 17.1 64 位

1.1 在 Qsys 中添加 ip、连接线，点击 generate

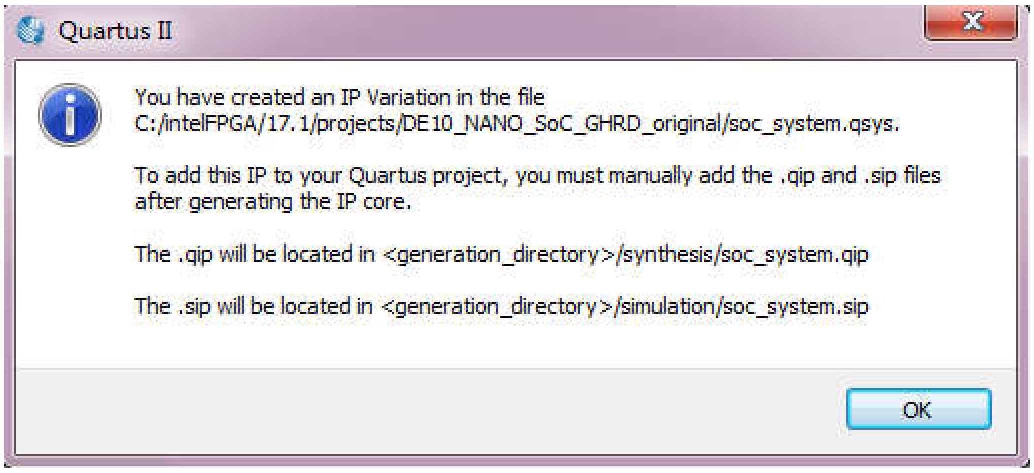


irq	Interrupt Sender	Double-click to export	[clk]
⊟ **uart_1**	UART (RS-232 Serial Port)		
clk	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clk]
s1	Avalon Memory Mapped Slave	Double-click to export	[clk]
external_connection	Conduit	**uart_1_external_connection**	
irq	Interrupt Sender	Double-click to export	[clk]
⊟ **my_pwm_Rmotor**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_0_conduit_end**	[clock]
⊟ **my_pwm_Lmotor**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_lmotor_conduit_end**	[clock]
⊟ **my_pwm_Fan1**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_fan1_conduit_end**	[clock]
⊟ **my_pwm_Fan2**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_fan2_conduit_end**	[clock]
⊟ **my_pwm_Body**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_body_conduit_end**	[clock]
⊟ **my_pwm_Picker**	my_pwm		
clock	Clock Input	Double-click to export	**clk_0**
reset	Reset Input	Double-click to export	[clock]
avalon_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]
conduit_end	Conduit	**my_pwm_picker_conduit_end**	[clock]

如图：



```
Info: Reusing file C:/intelFPGA/17.1/projects/de1_soc_GHRD_add_uart/
Info: rsp_demux: "mm_interconnect_1" instantiated altera_merlin_demul
Info: rsp_mux: "mm_interconnect_1" instantiated altera_merlin_multipl
Info: Reusing file C:/intelFPGA/17.1/projects/de1_soc_GHRD_add_uart/
Info: border: "hps_io" instantiated altera_interface_generator "border
Info: error_adapter_0: "avalon_st_adapter" instantiated error_adapter
Info: error_adapter_0: "avalon_st_adapter_001" instantiated error_ada
Info: soc_system: Done "soc_system" with 71 modules, 137 files
Info: qsys-generate succeeded.
Info: Finished: Create HDL design files for synthesis
Generate: completed with warnings.
```



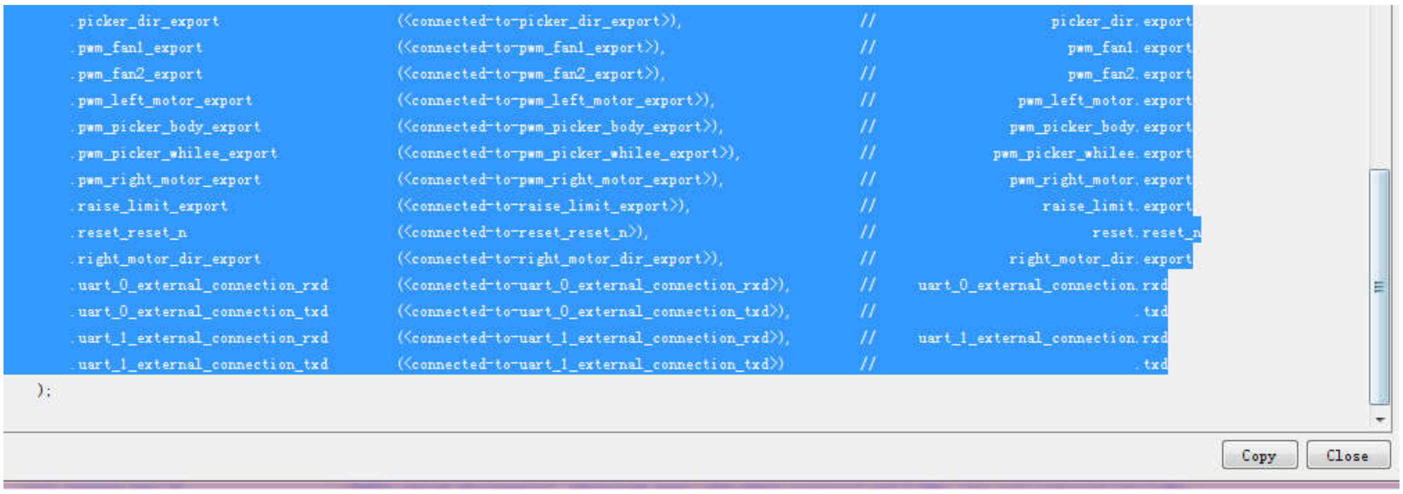
You have created an IP Variation in the file
C:/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD_original/soc_system.qsys.

To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.

The .qip will be located in <generation_directory>/synthesis/soc_system.qip

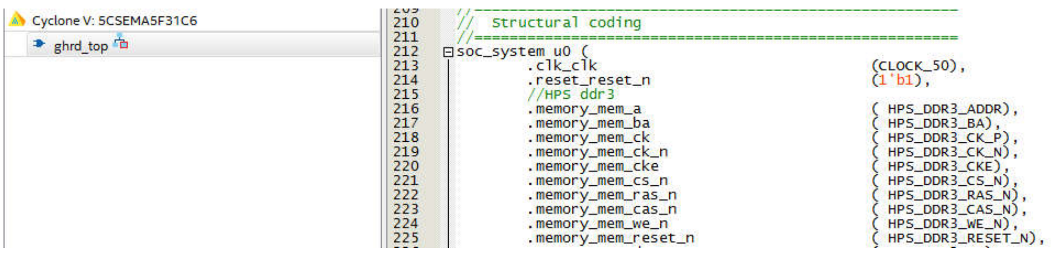
The .sip will be located in <generation_directory>/simulation/soc_system.sip

将模块接线情况复制到顶层设计文件中

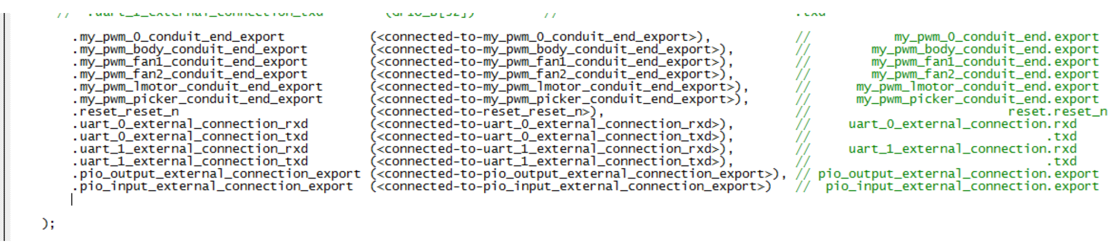


```
        .picker_dir_export              (<connected-to-picker_dir_export>),              //              picker_dir.export
        .pwm_fan1_export                (<connected-to-pwm_fan1_export>),                //                pwm_fan1.export
        .pwm_fan2_export                (<connected-to-pwm_fan2_export>),                //                pwm_fan2.export
        .pwm_left_motor_export          (<connected-to-pwm_left_motor_export>),          //          pwm_left_motor.export
        .pwm_picker_body_export         (<connected-to-pwm_picker_body_export>),         //         pwm_picker_body.export
        .pwm_picker_whilee_export       (<connected-to-pwm_picker_whilee_export>),       //       pwm_picker_whilee.export
        .pwm_right_motor_export         (<connected-to-pwm_right_motor_export>),         //         pwm_right_motor.export
        .raise_limit_export             (<connected-to-raise_limit_export>),             //             raise_limit.export
        .reset_reset_n                  (<connected-to-reset_reset_n>),                  //                   reset.reset_n
        .right_motor_dir_export         (<connected-to-right_motor_dir_export>),         //         right_motor_dir.export
        .uart_0_external_connection_rxd (<connected-to-uart_0_external_connection_rxd>), // uart_0_external_connection.rxd
        .uart_0_external_connection_txd (<connected-to-uart_0_external_connection_txd>), //                           .txd
        .uart_1_external_connection_rxd (<connected-to-uart_1_external_connection_rxd>), // uart_1_external_connection.rxd
        .uart_1_external_connection_txd (<connected-to-uart_1_external_connection_txd>)  //                           .txd
    );
```

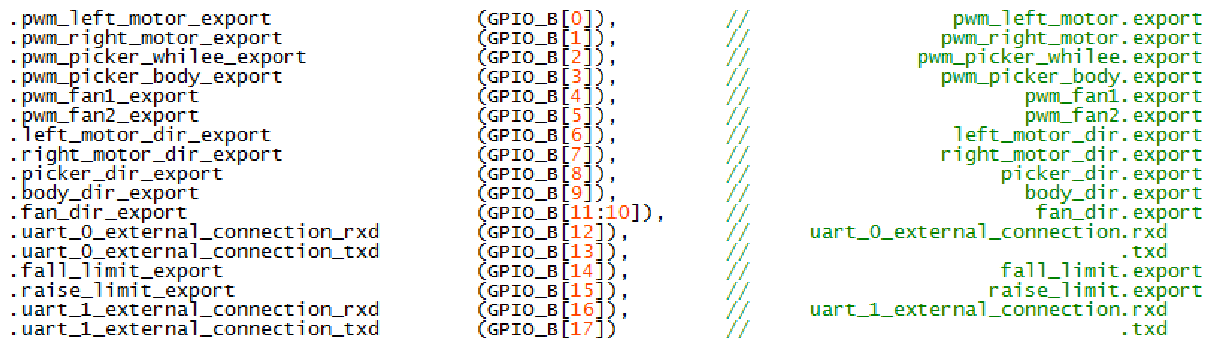
如图：



```
//  Structural coding
//=============================================================
soc_system u0 (
        .clk_clk                       (CLOCK_50),
        .reset_reset_n                 (1'b1),
        //HPS ddr3
        .memory_mem_a                  ( HPS_DDR3_ADDR),
        .memory_mem_ba                 ( HPS_DDR3_BA),
        .memory_mem_ck                 ( HPS_DDR3_CK_P),
        .memory_mem_ck_n               ( HPS_DDR3_CK_N),
        .memory_mem_cke                ( HPS_DDR3_CKE),
        .memory_mem_cs_n               ( HPS_DDR3_CS_N),
        .memory_mem_ras_n              ( HPS_DDR3_RAS_N),
        .memory_mem_cas_n              ( HPS_DDR3_CAS_N),
        .memory_mem_we_n               ( HPS_DDR3_WE_N),
        .memory_mem_reset_n            ( HPS_DDR3_RESET_N),
```



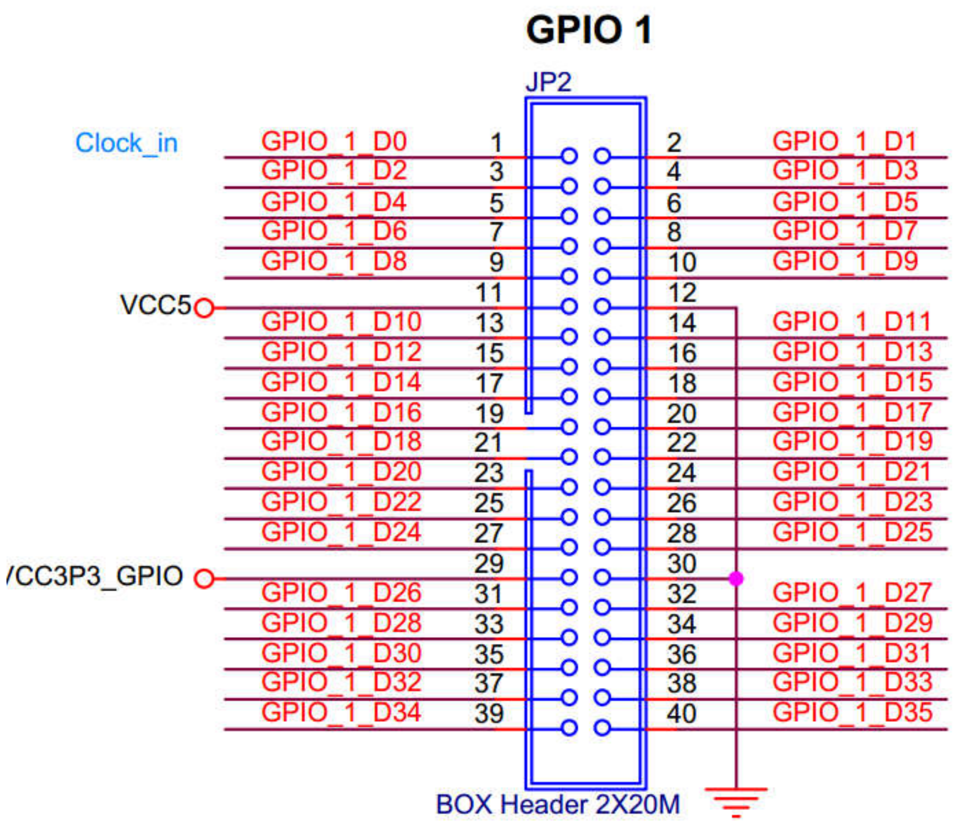
```
        .my_pwm_0_conduit_end_export          (<connected-to-my_pwm_0_conduit_end_export>),          //          my_pwm_0_conduit_end.export
        .my_pwm_body_conduit_end_export       (<connected-to-my_pwm_body_conduit_end_export>),       //       my_pwm_body_conduit_end.export
        .my_pwm_fan1_conduit_end_export       (<connected-to-my_pwm_fan1_conduit_end_export>),       //       my_pwm_fan1_conduit_end.export
        .my_pwm_fan2_conduit_end_export       (<connected-to-my_pwm_fan2_conduit_end_export>),       //       my_pwm_fan2_conduit_end.export
        .my_pwm_lmotor_conduit_end_export     (<connected-to-my_pwm_lmotor_conduit_end_export>),     //     my_pwm_lmotor_conduit_end.export
        .my_pwm_picker_conduit_end_export     (<connected-to-my_pwm_picker_conduit_end_export>),     //     my_pwm_picker_conduit_end.export
        .reset_reset_n                        (<connected-to-reset_reset_n>),                        //                         reset.reset_n
        .uart_0_external_connection_rxd       (<connected-to-uart_0_external_connection_rxd>),       //    uart_0_external_connection.rxd
        .uart_0_external_connection_txd       (<connected-to-uart_0_external_connection_txd>),       //                              .txd
        .uart_1_external_connection_rxd       (<connected-to-uart_1_external_connection_rxd>),       //    uart_1_external_connection.rxd
        .uart_1_external_connection_txd       (<connected-to-uart_1_external_connection_txd>),       //                              .txd
        .pio_output_external_connection_export (<connected-to-pio_output_external_connection_export>), // pio_output_external_connection.export
        .pio_input_external_connection_export  (<connected-to-pio_input_external_connection_export>)  //  pio_input_external_connection.export

    );
```



```
.pwm_left_motor_export                 (GPIO_B[0]),        //             pwm_left_motor.export
.pwm_right_motor_export                (GPIO_B[1]),        //            pwm_right_motor.export
.pwm_picker_whilee_export              (GPIO_B[2]),        //          pwm_picker_whilee.export
.pwm_picker_body_export                (GPIO_B[3]),        //            pwm_picker_body.export
.pwm_fan1_export                       (GPIO_B[4]),        //                   pwm_fan1.export
.pwm_fan2_export                       (GPIO_B[5]),        //                   pwm_fan2.export
.left_motor_dir_export                 (GPIO_B[6]),        //             left_motor_dir.export
.right_motor_dir_export                (GPIO_B[7]),        //            right_motor_dir.export
.picker_dir_export                     (GPIO_B[8]),        //                 picker_dir.export
.body_dir_export                       (GPIO_B[9]),        //                   body_dir.export
.fan_dir_export                        (GPIO_B[11:10]),    //                    fan_dir.export
.uart_0_external_connection_rxd        (GPIO_B[12]),       //     uart_0_external_connection.rxd
.uart_0_external_connection_txd        (GPIO_B[13]),       //                               .txd
.fall_limit_export                     (GPIO_B[14]),       //                 fall_limit.export
.raise_limit_export                    (GPIO_B[15]),       //                raise_limit.export
.uart_1_external_connection_rxd        (GPIO_B[16]),       //     uart_1_external_connection.rxd
.uart_1_external_connection_txd        (GPIO_B[17])        //                               .txd
```

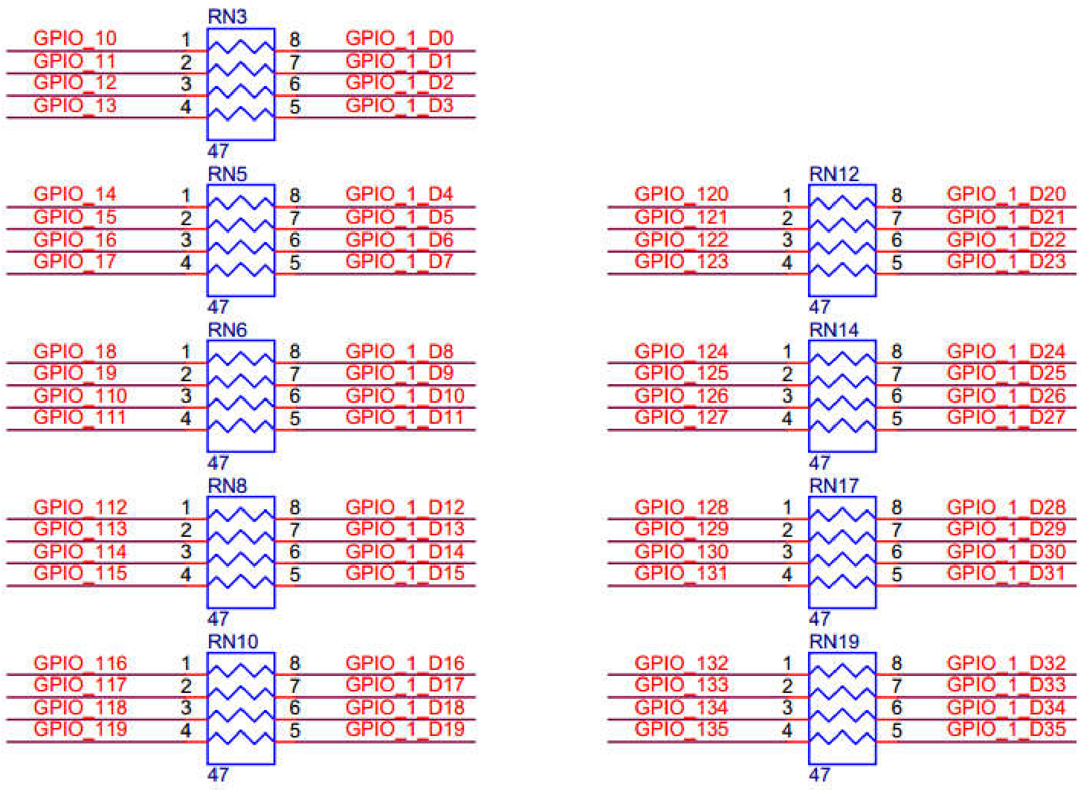
运行脚本文件修改管脚分配，如图：

DE1—SoC 开发板管脚分配

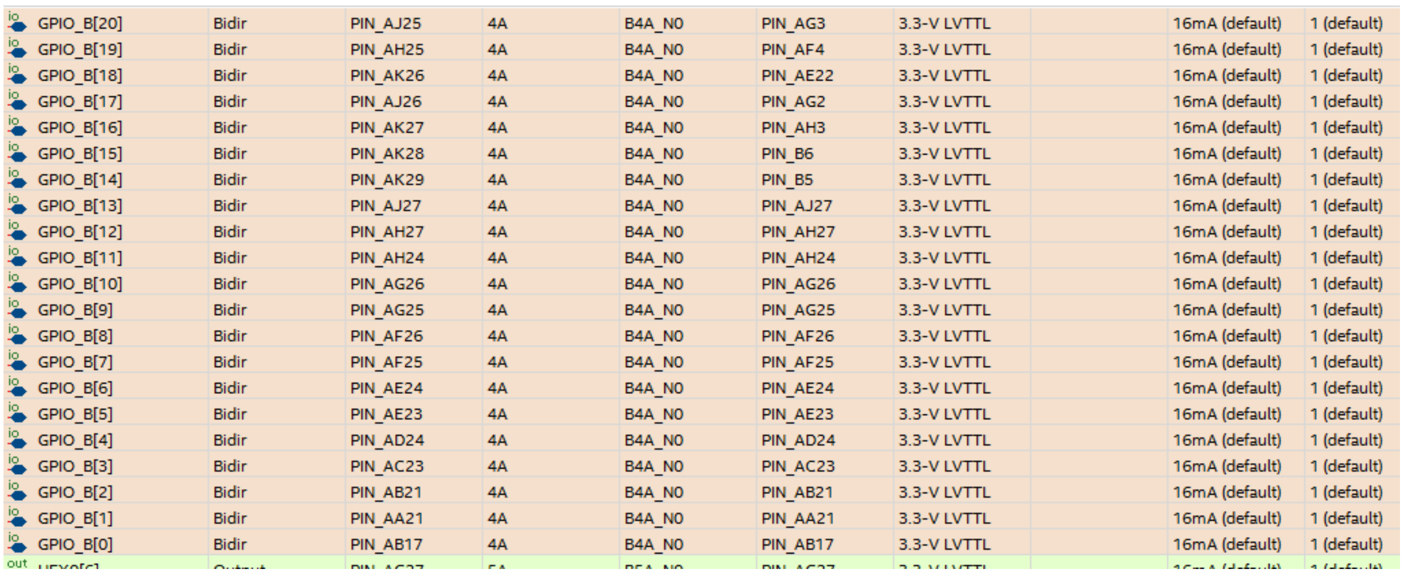


GPIO 1
JP2
Clock_in
VCC5
VCC3P3_GPIO
GPIO_1_D0 1 2 GPIO_1_D1
GPIO_1_D2 3 4 GPIO_1_D3
GPIO_1_D4 5 6 GPIO_1_D5
GPIO_1_D6 7 8 GPIO_1_D7
GPIO_1_D8 9 10 GPIO_1_D9
11 12
GPIO_1_D10 13 14 GPIO_1_D11
GPIO_1_D12 15 16 GPIO_1_D13
GPIO_1_D14 17 18 GPIO_1_D15
GPIO_1_D16 19 20 GPIO_1_D17
GPIO_1_D18 21 22 GPIO_1_D19
GPIO_1_D20 23 24 GPIO_1_D21
GPIO_1_D22 25 26 GPIO_1_D23
GPIO_1_D24 27 28 GPIO_1_D25
29 30
GPIO_1_D26 31 32 GPIO_1_D27
GPIO_1_D28 33 34 GPIO_1_D29
GPIO_1_D30 35 36 GPIO_1_D31
GPIO_1_D32 37 38 GPIO_1_D33
GPIO_1_D34 39 40 GPIO_1_D35
BOX Header 2X20M

开发板接口原理图



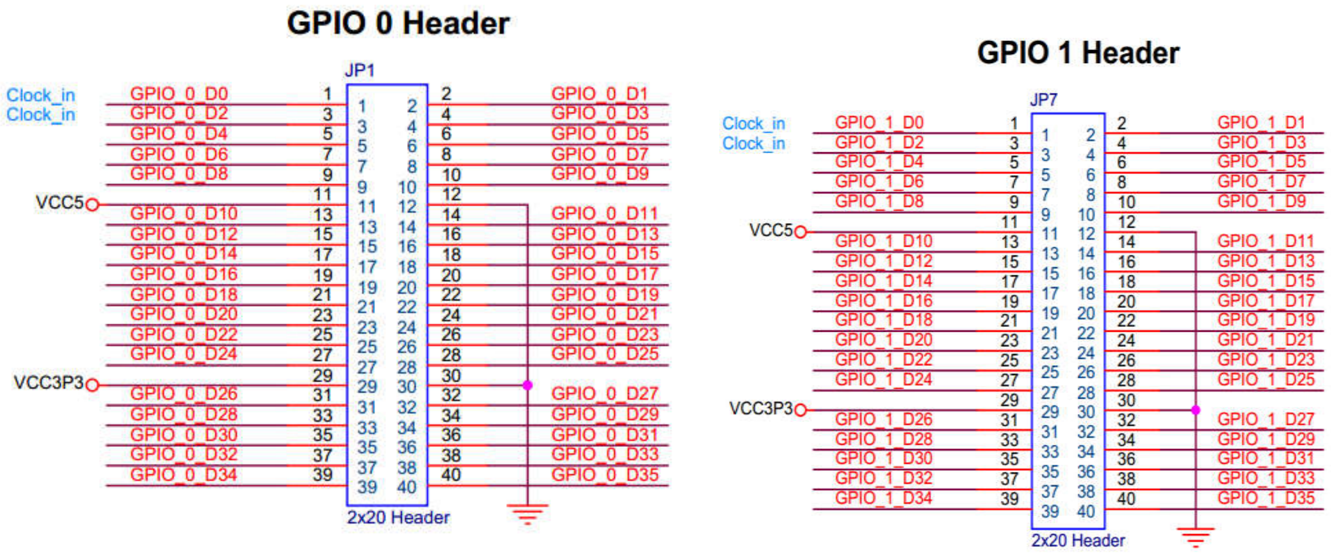
上下拉电阻后的管脚名称变化



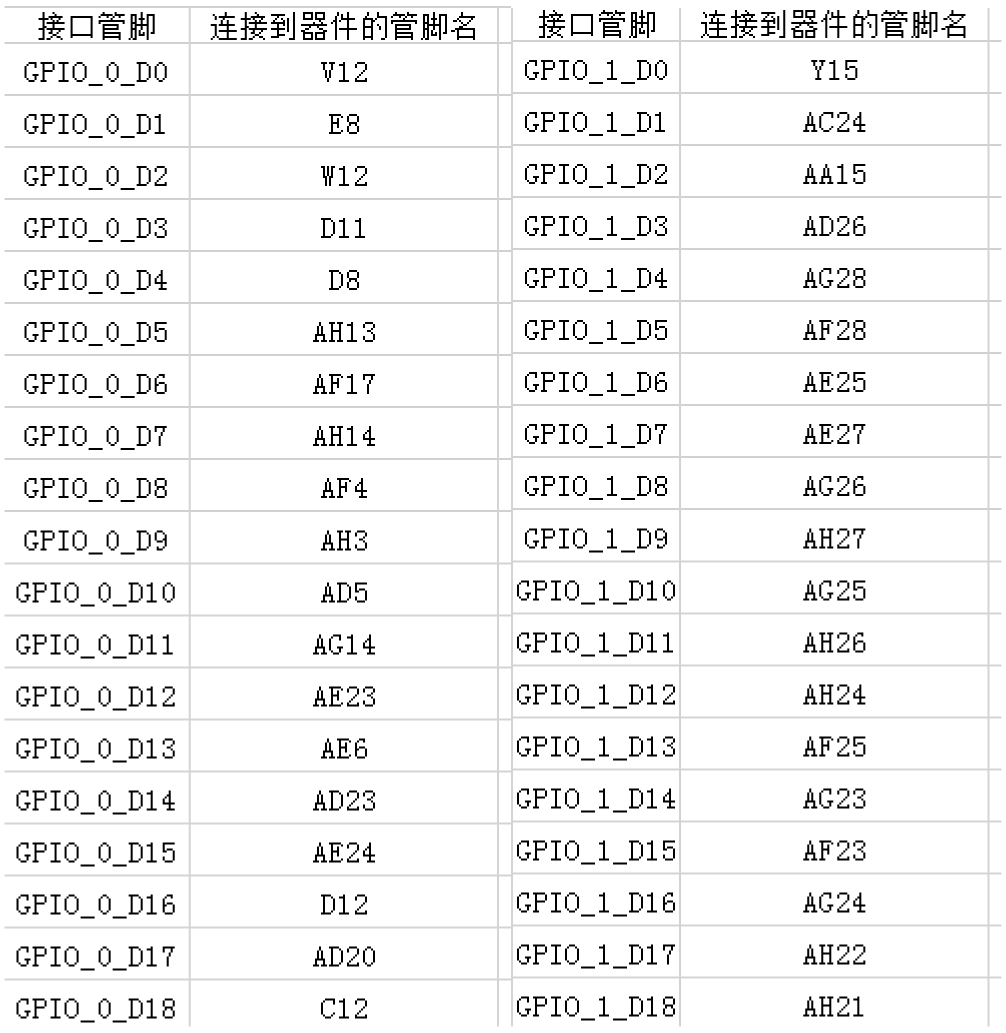
GPIO_B[20]	Bidir	PIN_AJ25	4A	B4A_N0	PIN_AG3	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[19]	Bidir	PIN_AH25	4A	B4A_N0	PIN_AF4	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[18]	Bidir	PIN_AK26	4A	B4A_N0	PIN_AE22	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[17]	Bidir	PIN_AJ26	4A	B4A_N0	PIN_AG2	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[16]	Bidir	PIN_AK27	4A	B4A_N0	PIN_AH3	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[15]	Bidir	PIN_AK28	4A	B4A_N0	PIN_B6	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[14]	Bidir	PIN_AK29	4A	B4A_N0	PIN_B5	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[13]	Bidir	PIN_AJ27	4A	B4A_N0	PIN_AJ27	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[12]	Bidir	PIN_AH27	4A	B4A_N0	PIN_AH27	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[11]	Bidir	PIN_AH24	4A	B4A_N0	PIN_AH24	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[10]	Bidir	PIN_AG26	4A	B4A_N0	PIN_AG26	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[9]	Bidir	PIN_AG25	4A	B4A_N0	PIN_AG25	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[8]	Bidir	PIN_AF26	4A	B4A_N0	PIN_AF26	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[7]	Bidir	PIN_AF25	4A	B4A_N0	PIN_AF25	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[6]	Bidir	PIN_AE24	4A	B4A_N0	PIN_AE24	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[5]	Bidir	PIN_AE23	4A	B4A_N0	PIN_AE23	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[4]	Bidir	PIN_AD24	4A	B4A_N0	PIN_AD24	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[3]	Bidir	PIN_AC23	4A	B4A_N0	PIN_AC23	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[2]	Bidir	PIN_AB21	4A	B4A_N0	PIN_AB21	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[1]	Bidir	PIN_AA21	4A	B4A_N0	PIN_AA21	3.3-V LVTTL		16mA (default)	1 (default)
GPIO_B[0]	Bidir	PIN_AB17	4A	B4A_N0	PIN_AB17	3.3-V LVTTL		16mA (default)	1 (default)

最终分配的管脚名称

DE10-nano 开发板管脚分配



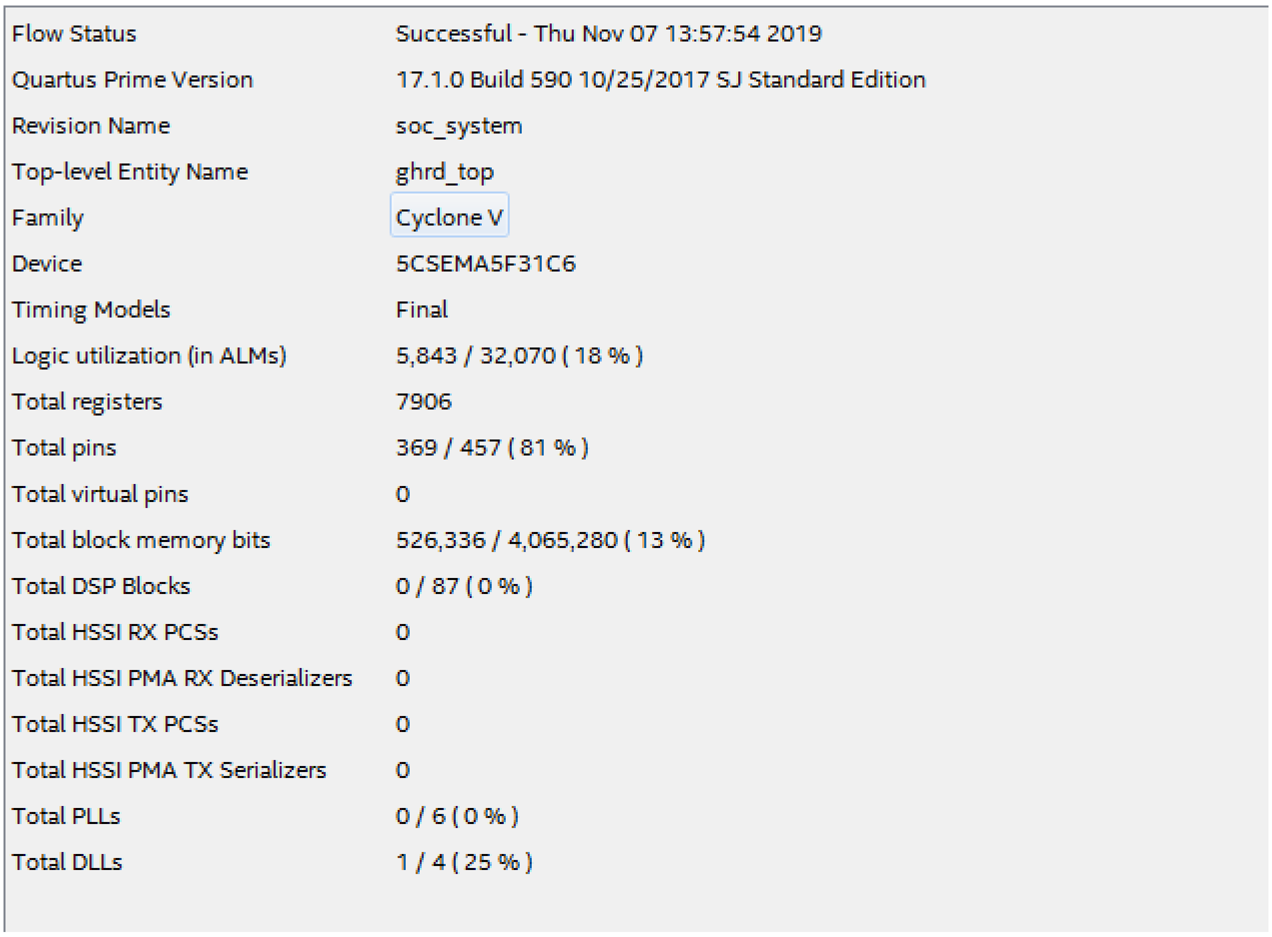
接口管脚名称



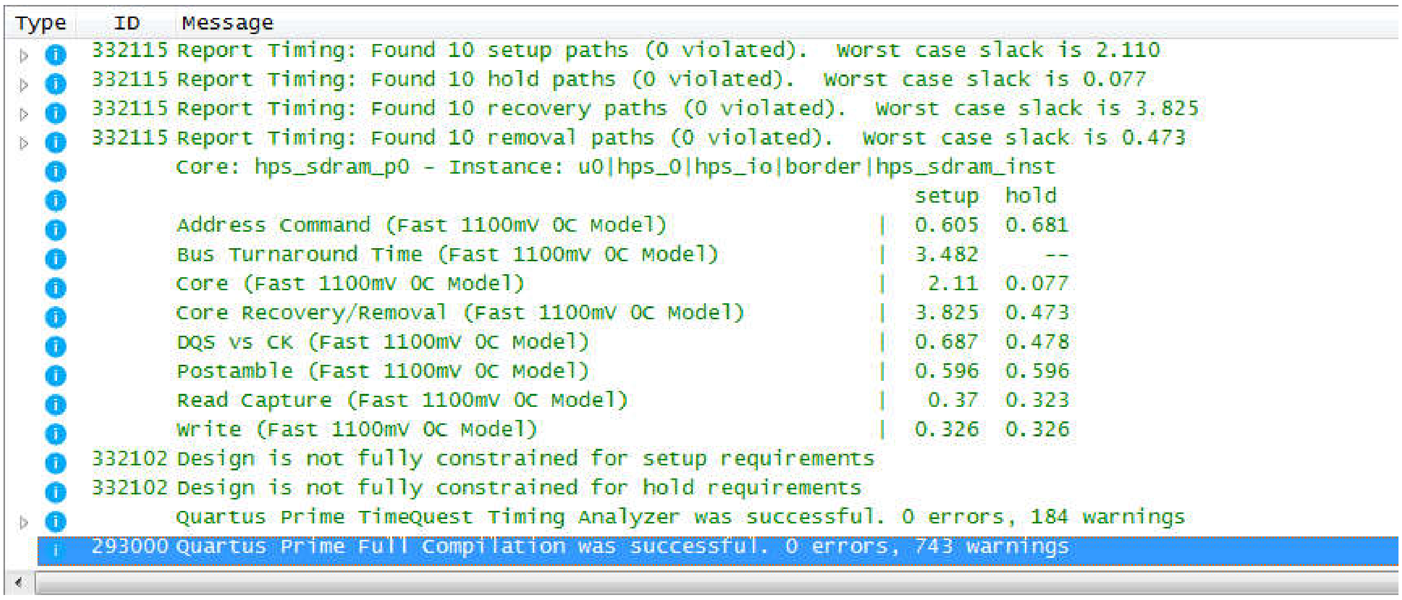
接口管脚	连接到器件的管脚名	接口管脚	连接到器件的管脚名
GPIO_0_D0	V12	GPIO_1_D0	Y15
GPIO_0_D1	E8	GPIO_1_D1	AC24
GPIO_0_D2	W12	GPIO_1_D2	AA15
GPIO_0_D3	D11	GPIO_1_D3	AD26
GPIO_0_D4	D8	GPIO_1_D4	AG28
GPIO_0_D5	AH13	GPIO_1_D5	AF28
GPIO_0_D6	AF17	GPIO_1_D6	AE25
GPIO_0_D7	AH14	GPIO_1_D7	AE27
GPIO_0_D8	AF4	GPIO_1_D8	AG26
GPIO_0_D9	AH3	GPIO_1_D9	AH27
GPIO_0_D10	AD5	GPIO_1_D10	AG25
GPIO_0_D11	AG14	GPIO_1_D11	AH26
GPIO_0_D12	AE23	GPIO_1_D12	AH24
GPIO_0_D13	AE6	GPIO_1_D13	AF25
GPIO_0_D14	AD23	GPIO_1_D14	AG23
GPIO_0_D15	AE24	GPIO_1_D15	AF23
GPIO_0_D16	D12	GPIO_1_D16	AG24
GPIO_0_D17	AD20	GPIO_1_D17	AH22
GPIO_0_D18	C12	GPIO_1_D18	AH21

最终连到 FPGA 的引脚

之后编译，编译时间大概为 17 分钟。



Flow Status	Successful - Thu Nov 07 13:57:54 2019
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Standard Edition
Revision Name	soc_system
Top-level Entity Name	ghrd_top
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	5,843 / 32,070 (18 %)
Total registers	7906
Total pins	369 / 457 (81 %)
Total virtual pins	0
Total block memory bits	526,336 / 4,065,280 (13 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	1 / 4 (25 %)



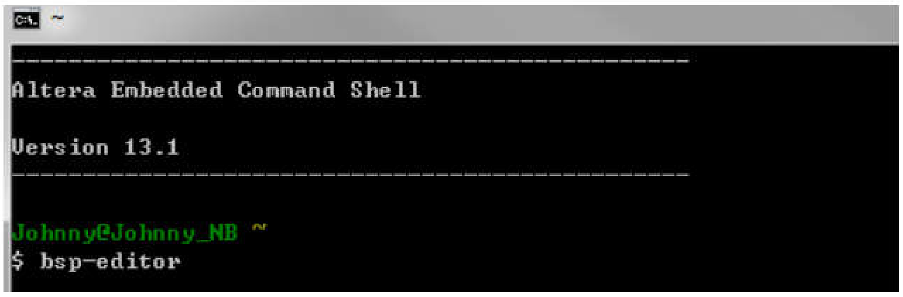
```
Type   ID     Message
       332115 Report Timing: Found 10 setup paths (0 violated).  Worst case slack is 2.110
       332115 Report Timing: Found 10 hold paths (0 violated).  Worst case slack is 0.077
       332115 Report Timing: Found 10 recovery paths (0 violated).  Worst case slack is 3.825
       332115 Report Timing: Found 10 removal paths (0 violated).  Worst case slack is 0.473
              Core: hps_sdram_p0 - Instance: u0|hps_0|hps_io|border|hps_sdram_inst
                                                              setup   hold
              Address Command (Fast 1100mV 0C Model)        |  0.605  0.681
              Bus Turnaround Time (Fast 1100mV 0C Model)    |  3.482     --
              Core (Fast 1100mV 0C Model)                   |   2.11  0.077
              Core Recovery/Removal (Fast 1100mV 0C Model)  |  3.825  0.473
              DQS vs CK (Fast 1100mV 0C Model)              |  0.687  0.478
              Postamble (Fast 1100mV 0C Model)              |  0.596  0.596
              Read Capture (Fast 1100mV 0C Model)           |   0.37  0.323
              Write (Fast 1100mV 0C Model)                  |  0.326  0.326
       332102 Design is not fully constrained for setup requirements
       332102 Design is not fully constrained for hold requirements
              Quartus Prime TimeQuest Timing Analyzer was successful. 0 errors, 184 warnings
       293000 Quartus Prime Full Compilation was successful. 0 errors, 743 warnings
```

3. eds 切换之 quartus 工程目录，生成以下文件

3.1. 准备工作

bsp-editor.exe 打开 bsp-editor 工具，选择 file->New BSP。把 Preloader settingdirectory 指定为 ghrd 中的 hps_isw_handoff/soc_system_hps_0。点击 OK 关闭。再点击 Geneate 生成 BSP 后 exit BSP Editor。此时应该可以在 GHRD 中看到 software 目录了。

工程目录。



```
-----------------------------------------------
Altera Embedded Command Shell

Version 13.1
-----------------------------------------------

Johnny@Johnny_NB ~
$ bsp-editor
```

3. 产生新的 BSP：打开 bsp-editor 后选择菜单 File --> New BSP 来创建新的 BSP 如图 2-81 所示：

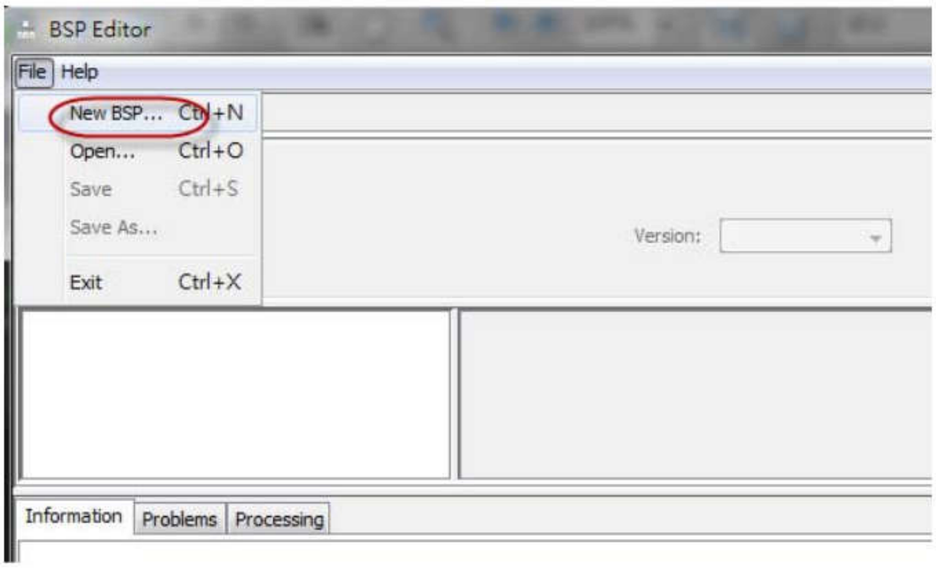
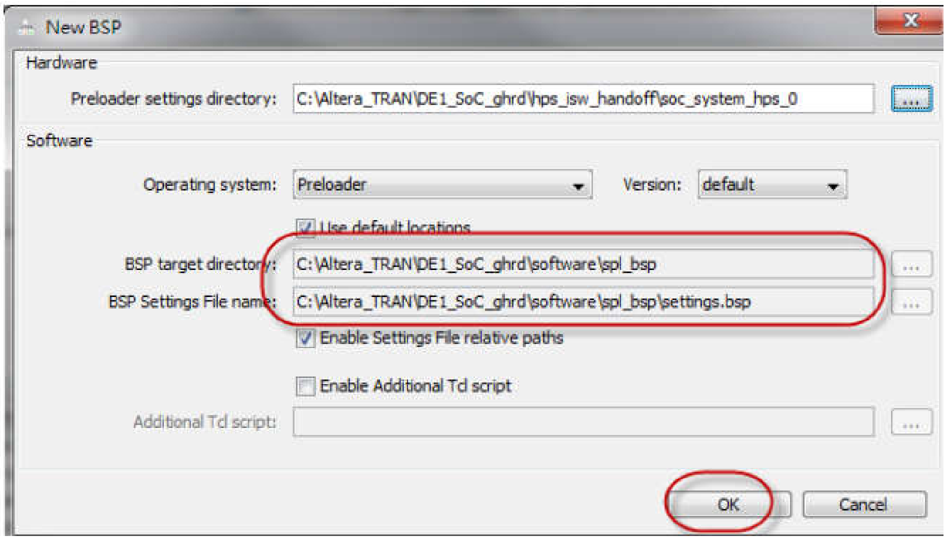


图 2-81 create new BSP

4. 然后设定 Preloader Setting Directory 的路径：

5. 在 New BSP 的窗口下选择 Preloader Setting Directory 的路径，把路径指向上节实验工程 DE1_SoC_ghrd\hps_isw_handoff\soc_system_hps_0\，如图 2-82 所示：

6. 选择 OK 来产生 BSP setting 文档以及文件夹，如图 2-83 所示：



New BSP

Hardware

Preloader settings directory: C:\Altera_TRAN\DE1_SoC_ghrd\hps_isw_handoff\soc_system_hps_0

Software

Operating system: Preloader   Version: default

☑ Use default locations

BSP target directory: C:\Altera_TRAN\DE1_SoC_ghrd\software\spl_bsp

BSP Settings File name: C:\Altera_TRAN\DE1_SoC_ghrd\software\spl_bsp\settings.bsp

☑ Enable Settings File relative paths

☐ Enable Additional Tcl script

Additional Tcl script:

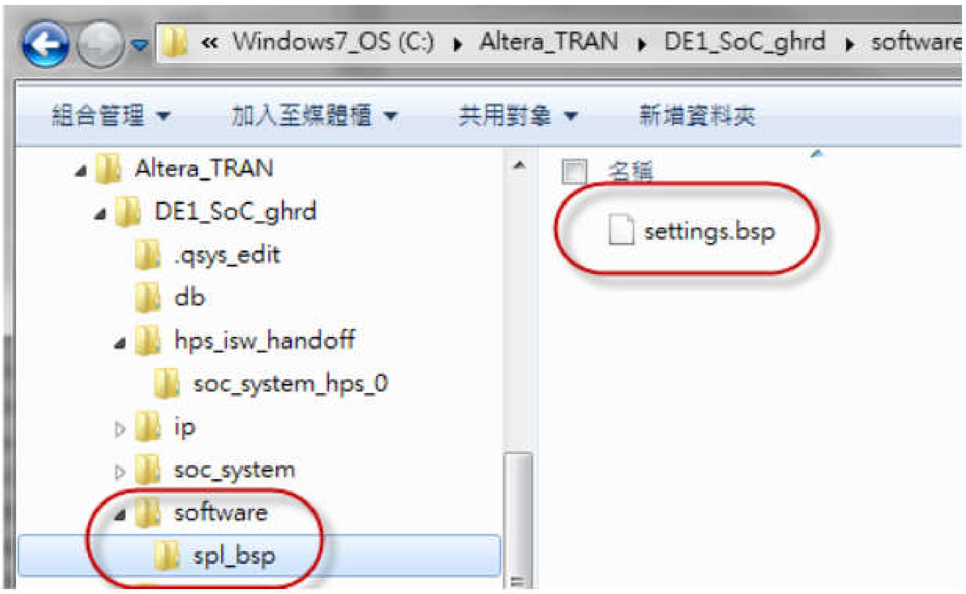
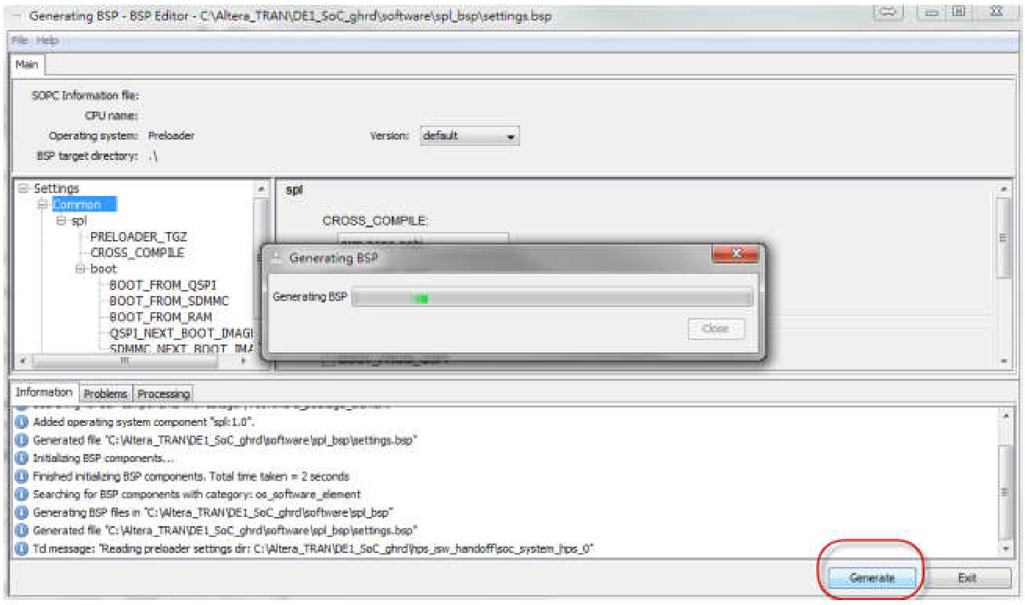


图 2-84 settings.bsp file

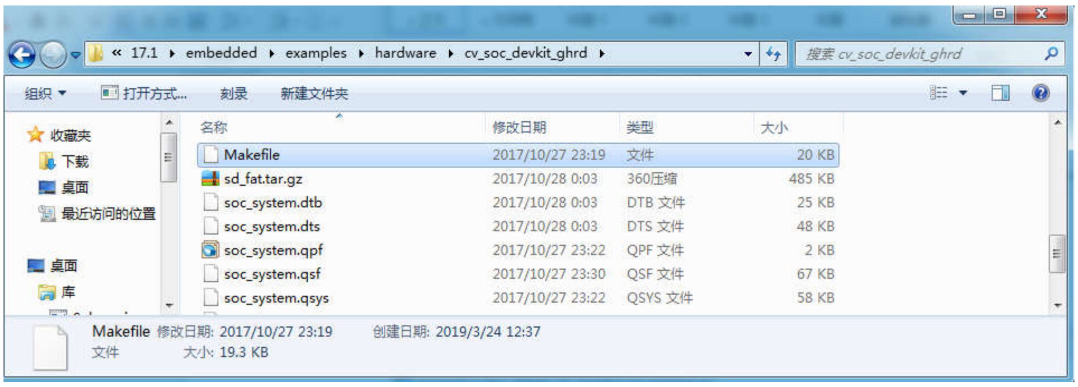
8. 接下来按下 Generate 生成 preloader 的原始档以及 Makefile。当生成档案完成后，按下 exit 完成任务离开窗口。如图 2-85 所示：



SOPC Information file:
CPU name:
Operating system: Preloader
Version: default
BSP target directory: .\

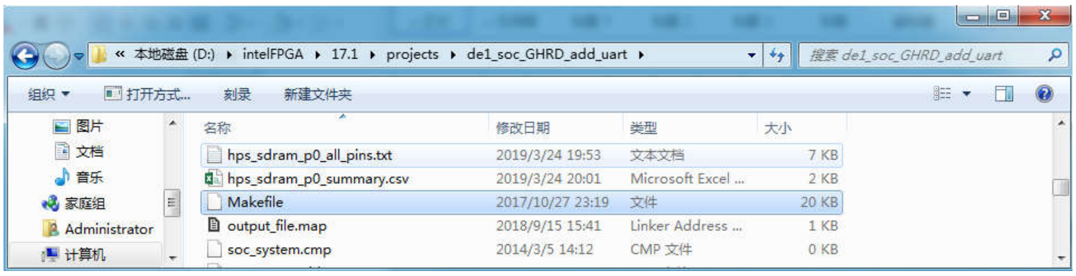
```
Added operating system component "spl:1.0".
Generated file "C:\Altera_TRAN\DE1_SoC_ghrd\software\spl_bsp\settings.bsp"
Initializing BSP components...
Finished initializing BSP components. Total time taken = 2 seconds
Searching for BSP components with category: os_software_element
Generating BSP files in "C:\Altera_TRAN\DE1_SoC_ghrd\software\spl_bsp"
Generated file "C:\Altera_TRAN\DE1_SoC_ghrd\software\spl_bsp\settings.bsp"
Tcl message: "Reading preloader settings dir: C:\Altera_TRAN\DE1_SoC_ghrd\hps_isw_handoff\soc_system_hps_0"
```

拷贝**\embedded\examples\hardware\cv_soc_devkit_ghrd** 中的 **makefile** 到 **quartus**



名称	修改日期	类型	大小
Makefile	2017/10/27 23:19	文件	20 KB
sd_fat.tar.gz	2017/10/28 0:03	360压缩	485 KB
soc_system.dtb	2017/10/28 0:03	DTB 文件	25 KB
soc_system.dts	2017/10/28 0:03	DTS 文件	48 KB
soc_system.qpf	2017/10/27 23:22	QPF 文件	2 KB
soc_system.qsf	2017/10/27 23:30	QSF 文件	67 KB
soc_system.qsys	2017/10/27 23:22	QSYS 文件	58 KB

Makefile 修改日期: 2017/10/27 23:19 创建日期: 2019/3/24 12:37
文件 大小: 19.3 KB

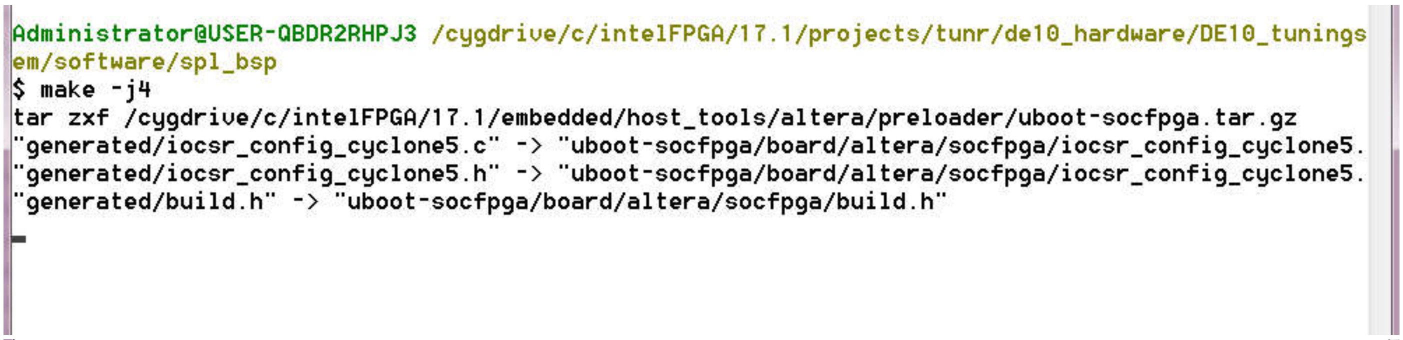

本地磁盘 (D:) › intelFPGA › 17.1 › projects › de1_soc_GHRD_add_uart ›
搜索 de1_soc_GHRD_add_uart
组织
打开方式...
刻录
新建文件夹
图片
文档
音乐
家庭组
Administrator
计算机

名称	修改日期	类型	大小
hps_sdram_p0_all_pins.txt	2019/3/24 19:53	文本文档	7 KB
hps_sdram_p0_summary.csv	2019/3/24 20:01	Microsoft Excel ...	2 KB
Makefile	2017/10/27 23:19	文件	20 KB
output_file.map	2018/9/15 15:41	Linker Address ...	1 KB
soc_system.cmp	2014/3/5 14:12	CMP 文件	0 KB

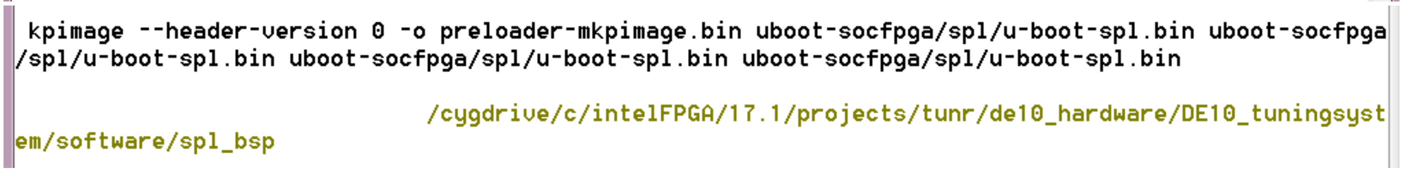
3.2. 生成 u-boot.img 和 u-boot- spi.bin

将 eds 的工作路径切换到工程目录\ softeare\spl_bsp，中输入 make (或者输入 make –j4):

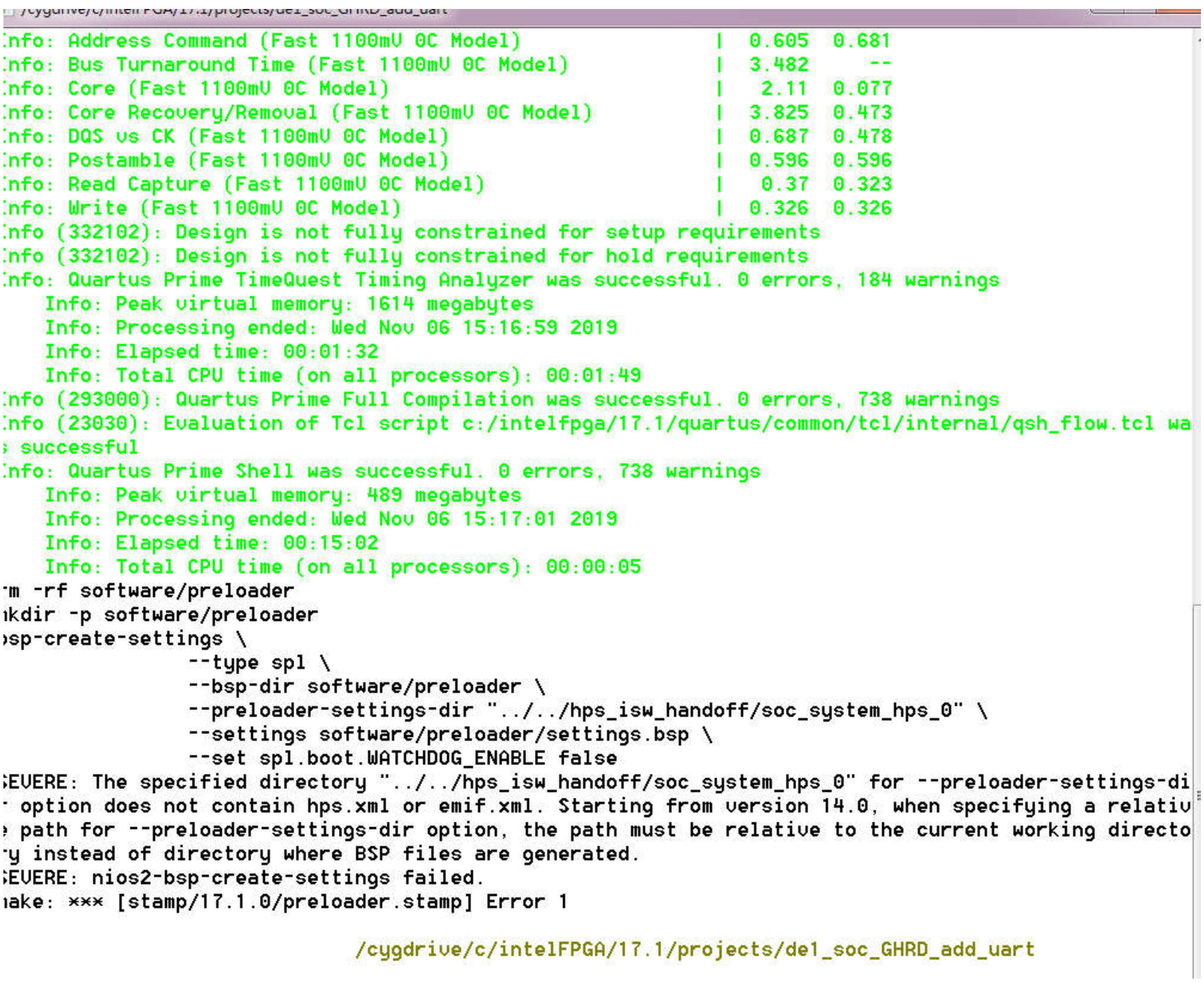
将 eds 的工作路径切换到工程目录,输入 make uboot,工程会进行重新编译（编译工程大概花费 20 分钟左右 22：50）



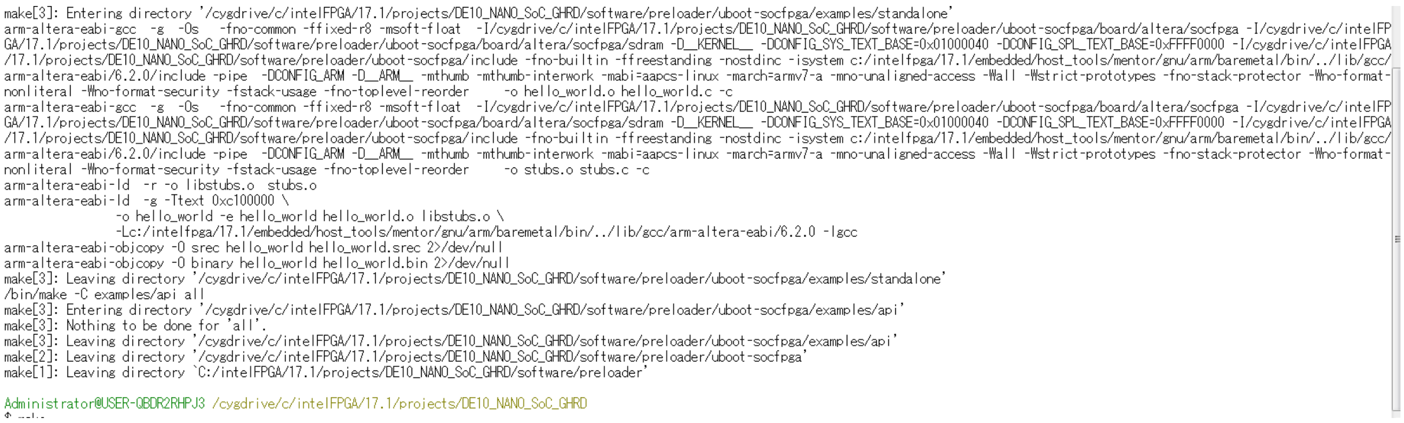
```
Administrator@USER-QBDR2RHPJ3 /cygdrive/c/intelFPGA/17.1/projects/tunr/de10_hardware/DE10_tunings
em/software/spl_bsp
$ make -j4
tar zxf /cygdrive/c/intelFPGA/17.1/embedded/host_tools/altera/preloader/uboot-socfpga.tar.gz
"generated/iocsr_config_cyclone5.c" -> "uboot-socfpga/board/altera/socfpga/iocsr_config_cyclone5.
"generated/iocsr_config_cyclone5.h" -> "uboot-socfpga/board/altera/socfpga/iocsr_config_cyclone5.
"generated/build.h" -> "uboot-socfpga/board/altera/socfpga/build.h"
```



```
 kpimage --header-version 0 -o preloader-mkpimage.bin uboot-socfpga/spl/u-boot-spl.bin uboot-socfpga
/spl/u-boot-spl.bin uboot-socfpga/spl/u-boot-spl.bin uboot-socfpga/spl/u-boot-spl.bin

                                /cygdrive/c/intelFPGA/17.1/projects/tunr/de10_hardware/DE10_tuningsyst
em/software/spl_bsp
```

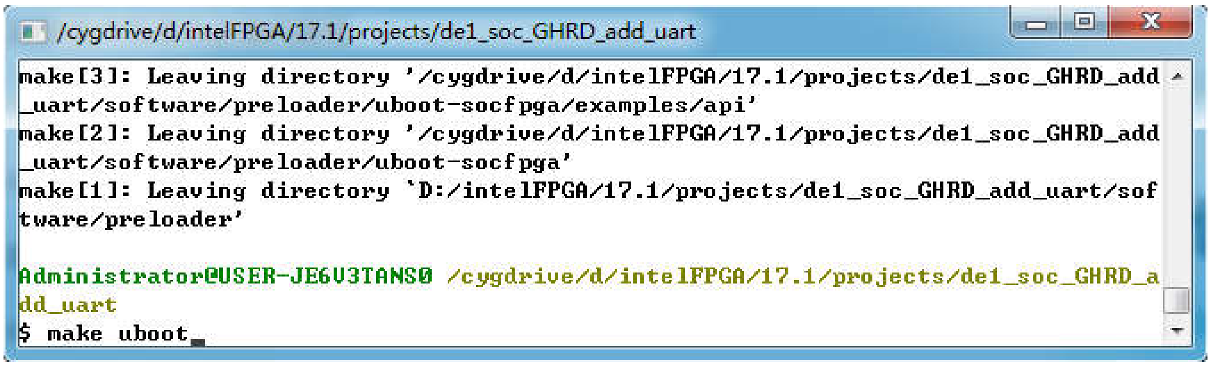


```
Info: Address Command (Fast 1100mV 0C Model)                    |  0.605  0.681
Info: Bus Turnaround Time (Fast 1100mV 0C Model)                |  3.482     --
Info: Core (Fast 1100mV 0C Model)                               |   2.11  0.077
Info: Core Recovery/Removal (Fast 1100mV 0C Model)              |  3.825  0.473
Info: DQS vs CK (Fast 1100mV 0C Model)                          |  0.687  0.478
Info: Postamble (Fast 1100mV 0C Model)                          |  0.596  0.596
Info: Read Capture (Fast 1100mV 0C Model)                       |   0.37  0.323
Info: Write (Fast 1100mV 0C Model)                              |  0.326  0.326
Info (332102): Design is not fully constrained for setup requirements
Info (332102): Design is not fully constrained for hold requirements
Info: Quartus Prime TimeQuest Timing Analyzer was successful. 0 errors, 184 warnings
    Info: Peak virtual memory: 1614 megabytes
    Info: Processing ended: Wed Nov 06 15:16:59 2019
    Info: Elapsed time: 00:01:32
    Info: Total CPU time (on all processors): 00:01:49
Info (293000): Quartus Prime Full Compilation was successful. 0 errors, 738 warnings
Info (23030): Evaluation of Tcl script c:/intelfpga/17.1/quartus/common/tcl/internal/qsh_flow.tcl wa
s successful
Info: Quartus Prime Shell was successful. 0 errors, 738 warnings
    Info: Peak virtual memory: 489 megabytes
    Info: Processing ended: Wed Nov 06 15:17:01 2019
    Info: Elapsed time: 00:15:02
    Info: Total CPU time (on all processors): 00:00:05
rm -rf software/preloader
mkdir -p software/preloader
bsp-create-settings \
                --type spl \
                --bsp-dir software/preloader \
                --preloader-settings-dir "../../hps_isw_handoff/soc_system_hps_0" \
                --settings software/preloader/settings.bsp \
                --set spl.boot.WATCHDOG_ENABLE false
SEVERE: The specified directory "../../hps_isw_handoff/soc_system_hps_0" for --preloader-settings-di
r option does not contain hps.xml or emif.xml. Starting from version 14.0, when specifying a relativ
e path for --preloader-settings-dir option, the path must be relative to the current working directo
ry instead of directory where BSP files are generated.
SEVERE: nios2-bsp-create-settings failed.
make: *** [stamp/17.1.0/preloader.stamp] Error 1

                              /cygdrive/c/intelFPGA/17.1/projects/de1_soc_GHRD_add_uart
```



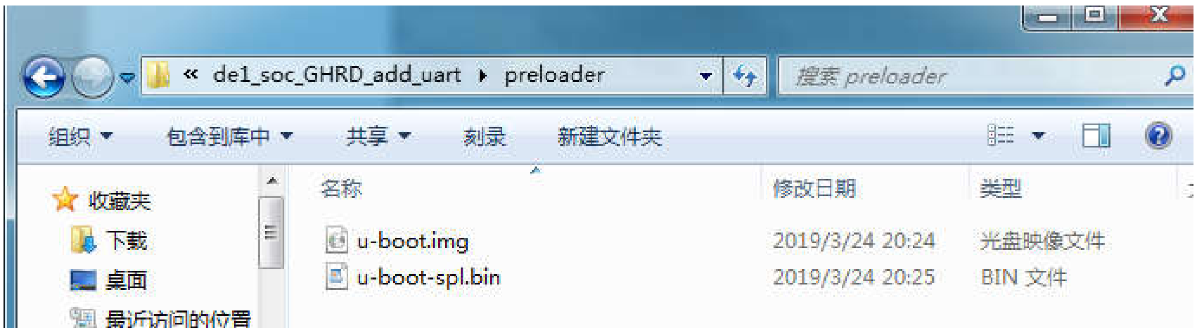
```
make[3]: Entering directory '/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/examples/standalone'
arm-altera-eabi-gcc  -g  -Os   -fno-common -ffixed-r8 -msoft-float  -I/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/board/altera/socfpga -I/cygdrive/c/intelFP
GA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/board/altera/socfpga/sdram -D__KERNEL__ -DCONFIG_SYS_TEXT_BASE=0x01000040 -DCONFIG_SPL_TEXT_BASE=0xFFFF0000 -I/cygdrive/c/intelFPGA
/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/include -fno-builtin -ffreestanding -nostdinc -isystem c:/intelfpga/17.1/embedded/host_tools/mentor/gnu/arm/baremetal/bin/../lib/gcc/
arm-altera-eabi/6.2.0/include -pipe  -DCONFIG_ARM -D__ARM__ -mthumb -mthumb-interwork -mabi=aapcs-linux -march=armv7-a -mno-unaligned-access -Wall -Wstrict-prototypes -fno-stack-protector -Wno-format-
nonliteral -Wno-format-security -fstack-usage -fno-toplevel-reorder     -o hello_world.o hello_world.c -c
arm-altera-eabi-gcc  -g  -Os   -fno-common -ffixed-r8 -msoft-float  -I/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/board/altera/socfpga -I/cygdrive/c/intelFP
GA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/board/altera/socfpga/sdram -D__KERNEL__ -DCONFIG_SYS_TEXT_BASE=0x01000040 -DCONFIG_SPL_TEXT_BASE=0xFFFF0000 -I/cygdrive/c/intelFPGA
/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/include -fno-builtin -ffreestanding -nostdinc -isystem c:/intelfpga/17.1/embedded/host_tools/mentor/gnu/arm/baremetal/bin/../lib/gcc/
arm-altera-eabi/6.2.0/include -pipe  -DCONFIG_ARM -D__ARM__ -mthumb -mthumb-interwork -mabi=aapcs-linux -march=armv7-a -mno-unaligned-access -Wall -Wstrict-prototypes -fno-stack-protector -Wno-format-
nonliteral -Wno-format-security -fstack-usage -fno-toplevel-reorder     -o stubs.o stubs.c -c
arm-altera-eabi-ld  -r -o libstubs.o  stubs.o
arm-altera-eabi-ld  -g -Ttext 0xc100000 \
                -o hello_world -e hello_world hello_world.o libstubs.o \
                -Lc:/intelfpga/17.1/embedded/host_tools/mentor/gnu/arm/baremetal/bin/../lib/gcc/arm-altera-eabi/6.2.0 -lgcc
arm-altera-eabi-objcopy -O srec hello_world hello_world.srec 2>/dev/null
arm-altera-eabi-objcopy -O binary hello_world hello_world.bin 2>/dev/null
make[3]: Leaving directory '/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/examples/standalone'
/bin/make -C examples/api all
make[3]: Entering directory '/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/examples/api'
make[3]: Nothing to be done for 'all'.
make[3]: Leaving directory '/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga/examples/api'
make[2]: Leaving directory '/cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader/uboot-socfpga'
make[1]: Leaving directory `C:/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD/software/preloader'

Administrator@USER-QBDR2RHPJ3 /cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD
```

编译完成后在 software\spl_bsp\uboot-socfpga 目录下会找到 u-boot.img 文件，在 software\preloader\uboot-socfpga\spl 下会找到 u-boot- spi.bin。



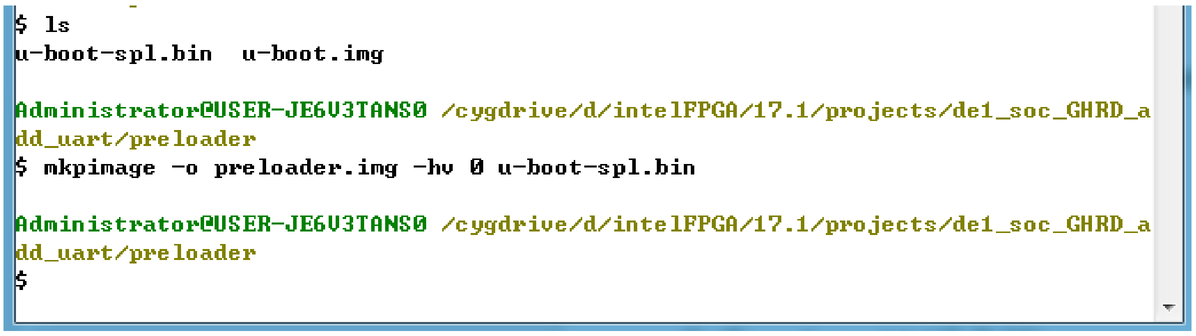
```
make[3]: Leaving directory '/cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_add
_uart/software/preloader/uboot-socfpga/examples/api'
make[2]: Leaving directory '/cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_add
_uart/software/preloader/uboot-socfpga'
make[1]: Leaving directory `D:/intelFPGA/17.1/projects/de1_soc_GHRD_add_uart/sof
tware/preloader'

Administrator@USER-JE6U3TANS0 /cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_a
dd_uart
$ make uboot
```

为了方便，把这两个文件复制到 D:\intelFPGA\17.1\projects\de1_soc_GHRD_add_uart\software\preloader 目录！



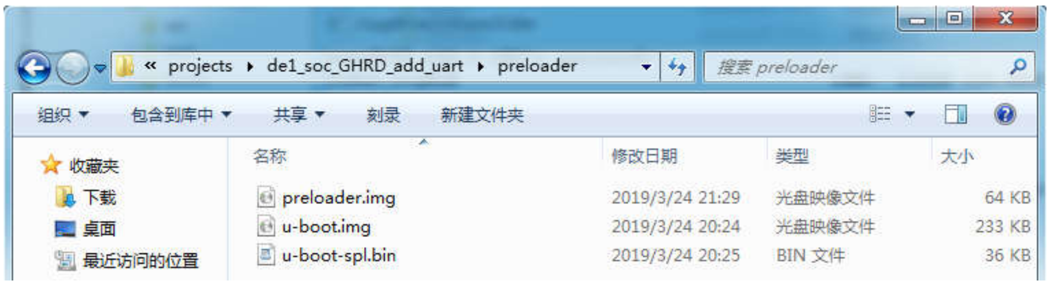
名称	修改日期	类型
u-boot.img	2019/3/24 20:24	光盘映像文件
u-boot-spl.bin	2019/3/24 20:25	BIN 文件

EDS 切换到~\de1_soc_GHRD_add_uart\software\preloader 目录，执行命令：mkpimage -o preloader.img -hv 0 u-boot-spl.bin



```
$ ls
u-boot-spl.bin  u-boot.img

Administrator@USER-JE6U3TANS0 /cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_a
dd_uart/preloader
$ mkpimage -o preloader.img -hv 0 u-boot-spl.bin

Administrator@USER-JE6U3TANS0 /cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_a
dd_uart/preloader
$
```

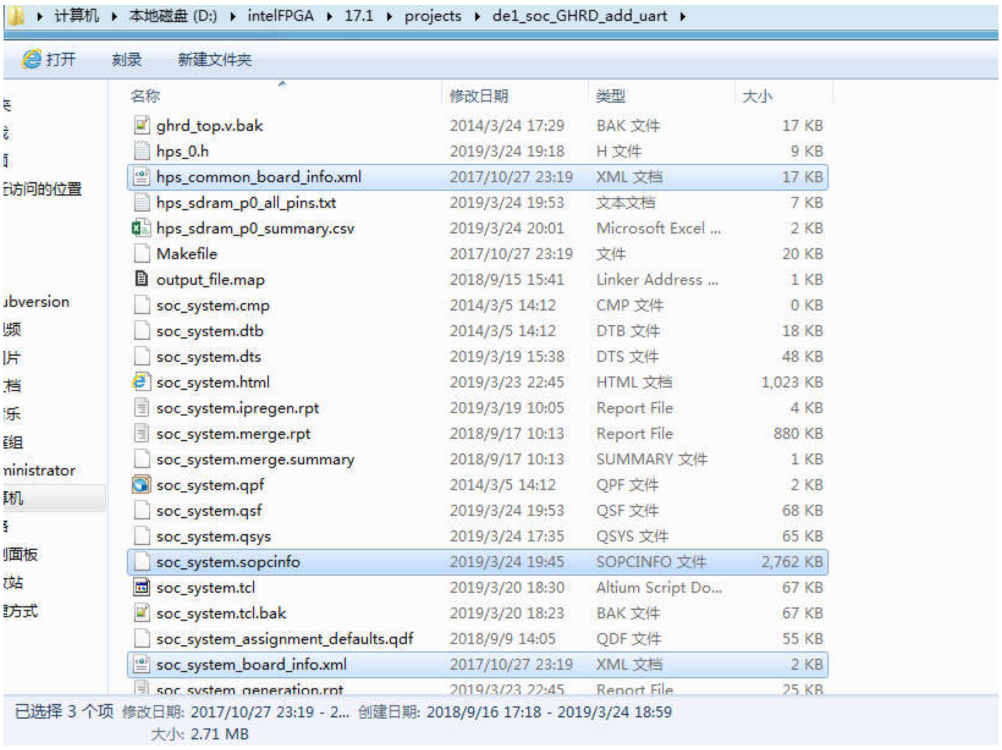
结果如下：



名称	修改日期	类型	大小
preloader.img	2019/3/24 21:29	光盘映像文件	64 KB
u-boot.img	2019/3/24 20:24	光盘映像文件	233 KB
u-boot-spl.bin	2019/3/24 20:25	BIN 文件	36 KB

3.3. 生成 socfpga.dtb(设备树)

确保工程目录下有三个文件

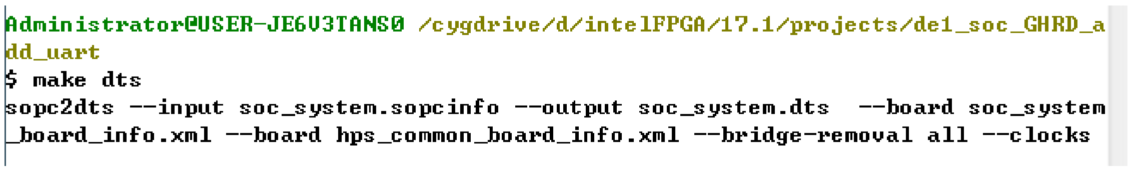

计算机 ▸ 本地磁盘 (D:) ▸ intelFPGA ▸ 17.1 ▸ projects ▸ de1_soc_GHRD_add_uart ▸

打开　刻录　新建文件夹

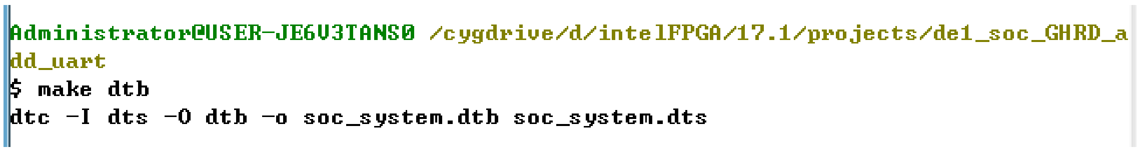
名称	修改日期	类型	大小
ghrd_top.v.bak	2014/3/24 17:29	BAK 文件	17 KB
hps_0.h	2019/3/24 19:18	H 文件	9 KB
hps_common_board_info.xml	2017/10/27 23:19	XML 文档	17 KB
hps_sdram_p0_all_pins.txt	2019/3/24 19:53	文本文档	7 KB
hps_sdram_p0_summary.csv	2019/3/24 20:01	Microsoft Excel ...	2 KB
Makefile	2017/10/27 23:19	文件	20 KB
output_file.map	2018/9/15 15:41	Linker Address ...	1 KB
soc_system.cmp	2014/3/5 14:12	CMP 文件	0 KB
soc_system.dtb	2014/3/5 14:12	DTB 文件	18 KB
soc_system.dts	2019/3/19 15:38	DTS 文件	48 KB
soc_system.html	2019/3/23 22:45	HTML 文档	1,023 KB
soc_system.ipregen.rpt	2019/3/19 10:05	Report File	4 KB
soc_system.merge.rpt	2018/9/17 10:13	Report File	880 KB
soc_system.merge.summary	2018/9/17 10:13	SUMMARY 文件	1 KB
soc_system.qpf	2014/3/5 14:12	QPF 文件	2 KB
soc_system.qsf	2019/3/24 19:53	QSF 文件	68 KB
soc_system.qsys	2019/3/24 17:35	QSYS 文件	65 KB
soc_system.sopcinfo	2019/3/24 19:45	SOPCINFO 文件	2,762 KB
soc_system.tcl	2019/3/20 18:30	Altium Script Do...	67 KB
soc_system.tcl.bak	2019/3/20 18:23	BAK 文件	67 KB
soc_system_assignment_defaults.qdf	2018/9/9 14:05	QDF 文件	55 KB
soc_system_board_info.xml	2017/10/27 23:19	XML 文档	2 KB
soc_system_generation.rpt	2019/3/23 22:45	Report File	25 KB

已选择 3 个项　修改日期: 2017/10/27 23:19 - 2...　创建日期: 2018/9/16 17:18 - 2019/3/24 18:59
大小: 2.71 MB

切换至工程目录，执行：make dts


```
Administrator@USER-JE6V3TANS0 /cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_a
dd_uart
$ make dts
sopc2dts --input soc_system.sopcinfo --output soc_system.dts  --board soc_system
_board_info.xml --board hps_common_board_info.xml --bridge-removal all --clocks
```

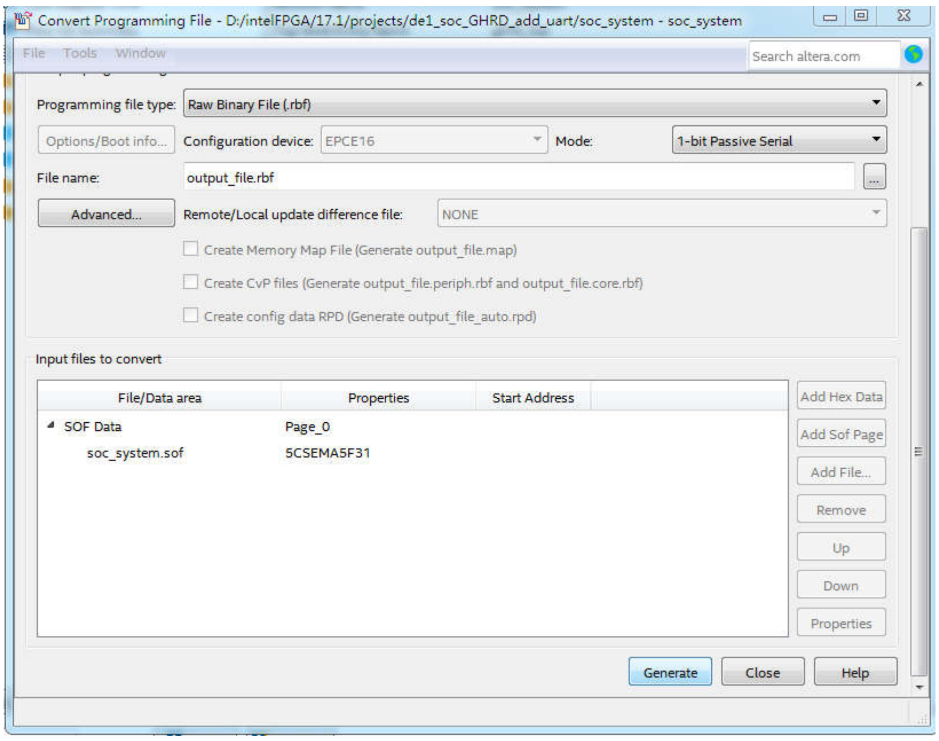
之后执行：make dtb


```
Administrator@USER-JE6V3TANS0 /cygdrive/d/intelFPGA/17.1/projects/de1_soc_GHRD_a
dd_uart
$ make dtb
dtc -I dts -O dtb -o soc_system.dtb soc_system.dts
```

将 soc_system.dtb 命名为 socfpga.dtb

3.4. 产生 FPGA 配置文件

生成 soc_system.rbf（FPGA 配置文件），既可以用 quartus 直接转换，也可以用脚本文件生成


Convert Programming File - D:/intelFPGA/17.1/projects/de1_soc_GHRD_add_uart/soc_system - soc_system

Programming file type: Raw Binary File (.rbf)

Configuration device: EPCE16 Mode: 1-bit Passive Serial

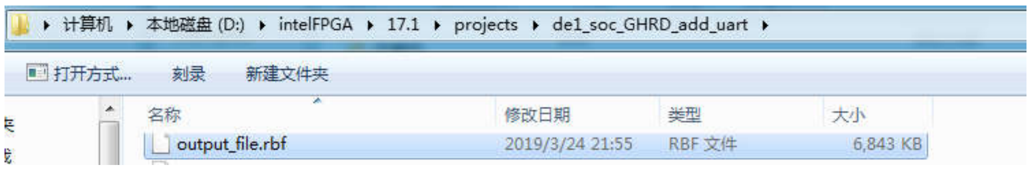
File name: output_file.rbf

Remote/Local update difference file: NONE

- [ ] Create Memory Map File (Generate output_file.map)
- [ ] Create CvP files (Generate output_file.periph.rbf and output_file.core.rbf)
- [ ] Create config data RPD (Generate output_file_auto.rpd)

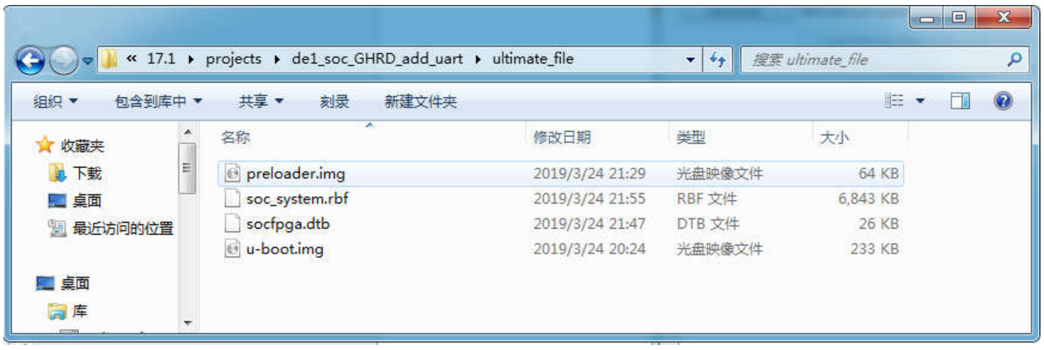
Input files to convert

File/Data area	Properties	Start Address
SOF Data	Page_0	
soc_system.sof	5CSEMA5F31	


计算机 ▸ 本地磁盘 (D:) ▸ intelFPGA ▸ 17.1 ▸ projects ▸ de1_soc_GHRD_add_uart ▸

名称	修改日期	类型	大小
output_file.rbf	2019/3/24 21:55	RBF 文件	6,843 KB

将生成的 rbf 文件改名为 soc_system.rbf 会同前面生成的三个文件放到一起。


« 17.1 ▸ projects ▸ de1_soc_GHRD_add_uart ▸ ultimate_file

名称	修改日期	类型	大小
preloader.img	2019/3/24 21:29	光盘映像文件	64 KB
soc_system.rbf	2019/3/24 21:55	RBF 文件	6,843 KB
socfpga.dtb	2019/3/24 21:47	DTB 文件	26 KB
u-boot.img	2019/3/24 20:24	光盘映像文件	233 KB

4. 更新 SD 卡

直接用 rbf 和 dts 文件替换原 SD 卡中 FAT 分区下的两个文件即可

4.1. 准备文件

将第 2 步中生成的 zImage\u-boot.img\preloader.img\socfpga.dtb\soc_system.rbf 四个文件拷贝 Linux 系统中

4.2. 将系统切换到 Linux

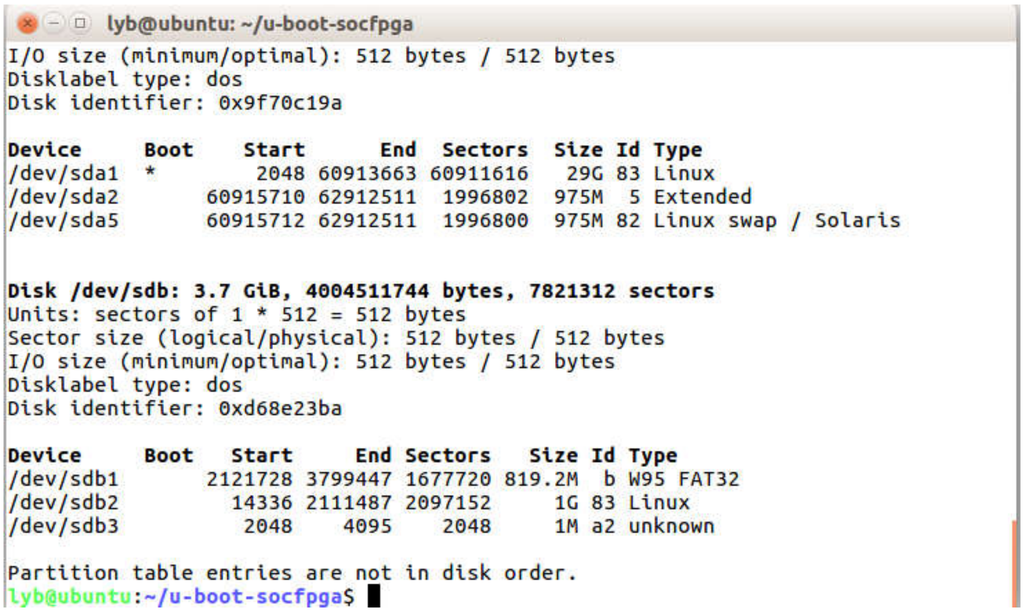
将 Linux 终端路径切换到有以上 4 个文件的目录

4.3. 更换 zImage

用编译好的 zImage 替换 SD 卡中的 zImage(第一次需要替换，后面不需要替换)
能用的 zImage 为

4.4. 查看设备

执行 sudo fdisk -l 查看 SD 的设备名称（我这里是/dev/sdb）

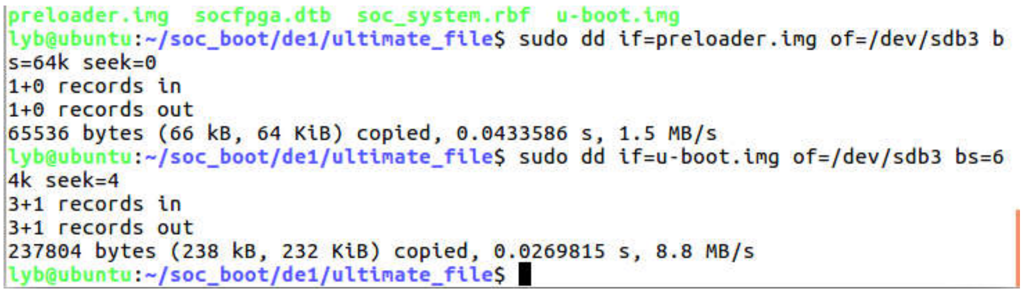


lyb@ubuntu: ~/u-boot-socfpga

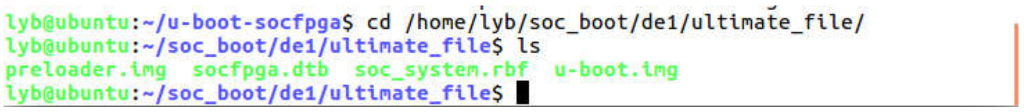
```
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x9f70c19a

Device     Boot     Start      End  Sectors  Size Id Type
/dev/sda1  *         2048 60913663 60911616   29G 83 Linux
/dev/sda2        60915710 62912511  1996802  975M  5 Extended
/dev/sda5        60915712 62912511  1996800  975M 82 Linux swap / Solaris


Disk /dev/sdb: 3.7 GiB, 4004511744 bytes, 7821312 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xd68e23ba

Device     Boot   Start     End Sectors   Size Id Type
/dev/sdb1       2121728 3799447 1677720 819.2M  b W95 FAT32
/dev/sdb2         14336 2111487 2097152     1G 83 Linux
/dev/sdb3          2048    4095    2048     1M a2 unknown

Partition table entries are not in disk order.
lyb@ubuntu:~/u-boot-socfpga$
```

4.5. 更新启动文件

分别执行 sudo dd if=preloader.img of=/dev/sdb3 bs=64k seek=0 和 sudo dd if=u-boot.img of=/dev/sdb3 bs=64k seek=4



```
preloader.img  socfpga.dtb  soc_system.rbf  u-boot.img
lyb@ubuntu:~/soc_boot/de1/ultimate_file$ sudo dd if=preloader.img of=/dev/sdb3 b
s=64k seek=0
1+0 records in
1+0 records out
65536 bytes (66 kB, 64 KiB) copied, 0.0433586 s, 1.5 MB/s
lyb@ubuntu:~/soc_boot/de1/ultimate_file$ sudo dd if=u-boot.img of=/dev/sdb3 bs=6
4k seek=4
3+1 records in
3+1 records out
237804 bytes (238 kB, 232 KiB) copied, 0.0269815 s, 8.8 MB/s
lyb@ubuntu:~/soc_boot/de1/ultimate_file$
```



```
lyb@ubuntu:~/u-boot-socfpga$ cd /home/lyb/soc_boot/de1/ultimate_file/
lyb@ubuntu:~/soc_boot/de1/ultimate_file$ ls
preloader.img  socfpga.dtb  soc_system.rbf  u-boot.img
lyb@ubuntu:~/soc_boot/de1/ultimate_file$
```

再使用 sudo sync 命令。

4.6. 替换启动文件

将 socfpga.dtb\soc_system.rbf 考进 SD 卡替换原有文件

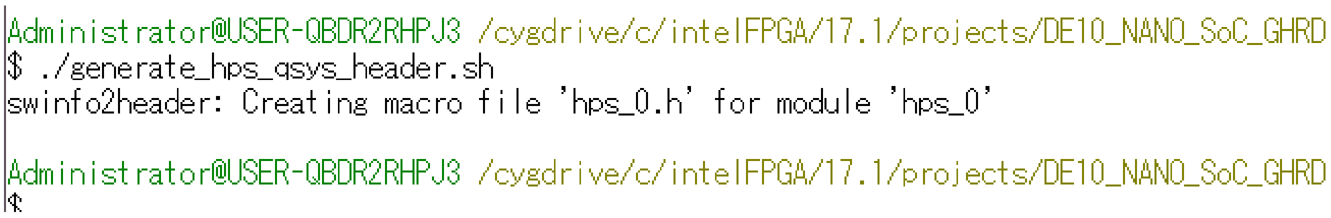
5. 嵌入式 Linux 端应用程序开发

经过前面步骤，FPGA 的硬件电路（设备）已经被挂载到 Linux 系统当中，设备的地址也都被映射到内存中。这一小结介绍如何进行编写设备应用程序。

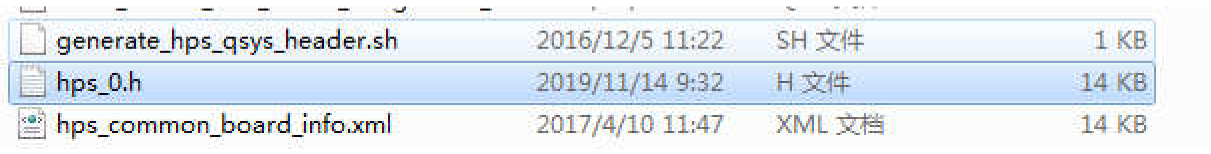
5.1. 生成头文件

头文件含有设备在内存中映射的偏移地址，因此编程之前需要生成头文件。打开 DES 切换至 Quartus 工程目录下，

（1）输入“./generate_hps_qsys_header.sh”.



```
Administrator@USER-QBDR2RHPJ3 /cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD
$ ./generate_hps_qsys_header.sh
swinfo2header: Creating macro file 'hps_0.h' for module 'hps_0'

Administrator@USER-QBDR2RHPJ3 /cygdrive/c/intelFPGA/17.1/projects/DE10_NANO_SoC_GHRD
$
```



generate_hps_qsys_header.sh	2016/12/5 11:22	SH 文件	1 KB
hps_0.h	2019/11/14 9:32	H 文件	14 KB
hps_common_board_info.xml	2017/4/10 11:47	XML 文档	14 KB

（2）将 hps_0.h 和/embedded/ip/altera/hps/altera_hps/hwlib/include/socal 中的 hps.h 复制到同一个文件夹，便于查找。我用的是 Quartus17.1，头文件目录在“C:\intelFPGA\17.1\embedded\ip\altera\hps\altera_hps\hwlib\include\soc_cv_av\socal”之中。

5.2. 映射设备地址

参考《02_DE-SOC 培训.pdf》117 页

（1）调用 open 函数打开 memory 设备驱动 /dev/mem

（2）调用 mmap 函数映射 HPS 的 L3 外设区域物理地址到虚拟地址并表示为一个空指针变量 virtual_base

（3）通过 virtual_base+偏移地址计算设备的虚拟地址

6. 在 Ubuntu 上安装 OpenCV

6.1. 先安装 ffmpeg

第一步：添加源。

```
sudo add-apt-repository ppa:djcj/hybrid
```

第二步：更新源。

```
sudo apt-get update
```

第三步：下载安装。

```
sudo apt-get install ffmpeg
```

6.2. 安装 cmake，

注意 opencv2.4.*系列最好安装 cmake2.8.*

1.下载解压 cmake-2.8.12.1.tar.gz

apt-getautoremove cmake

cd /usr

wget http://www.cmake.org/files/v2.8/cmake-2.8.12.1-Linux-i386.tar.gz

tar -zxvf cmake-2.8.12.1.tar.gz

2.链接命令

ln -s/usr/cmake-2.8.12.1-Linux-i386/bin/* /usr/bin/

3.检查是否成功

cmake --version

显示版本即成功。

cmake version2.8.12.1

6.3. 安装 opencv2.4.9

方法一：

1. 先下载 OpenCV 的源码　http://opencv.org/downloads.html

2. 解压到任意目录　unzip opencv-2.4.9.zip

3.进入源码目录　cd opencv-2.4.9

4. 事先安装下列软件

sudo apt-get install build-essential cmake libgtk2.0-dev pkg-config python-dev python-numpy libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install libavcodec-dev libavformat-dev libjpeg.dev libtiff4.dev libswscale-dev libjasper-dev

sudo apt-get install cmake

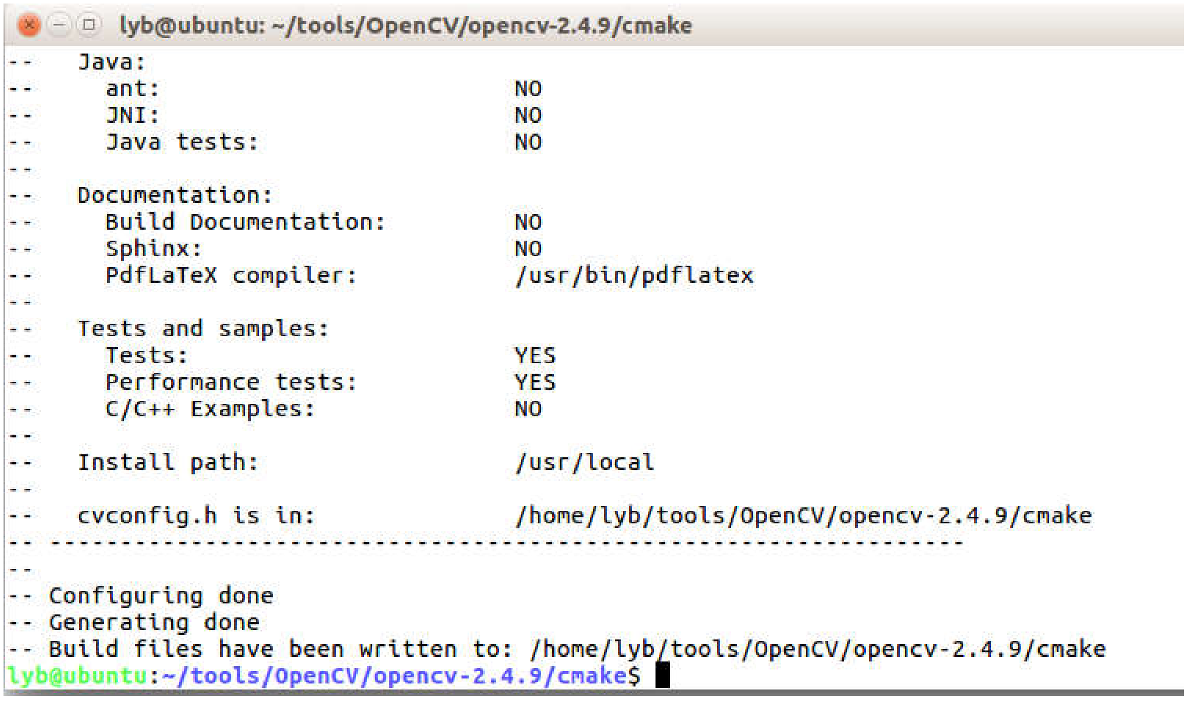
5. 进入 cmake

   cd cmake

6. cmake 编译生成 Makefile，

cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

安装所有的 lib 文件都会被安装到/usr/local 目录



```
lyb@ubuntu: ~/tools/OpenCV/opencv-2.4.9/cmake
--   Java:
--     ant:                          NO
--     JNI:                          NO
--     Java tests:                   NO
--
--   Documentation:
--     Build Documentation:          NO
--     Sphinx:                       NO
--     PdfLaTeX compiler:            /usr/bin/pdflatex
--
--   Tests and samples:
--     Tests:                        YES
--     Performance tests:            YES
--     C/C++ Examples:               NO
--
--   Install path:                   /usr/local
--
--   cvconfig.h is in:               /home/lyb/tools/OpenCV/opencv-2.4.9/cmake
-- -----------------------------------------------------------------
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/lyb/tools/OpenCV/opencv-2.4.9/cmake
lyb@ubuntu:~/tools/OpenCV/opencv-2.4.9/cmake$
```

7. 编译,并安装

sudo make install

方法 2：

sudo apt-get install build-essential cmake libgtk2.0-dev pkg-config python-dev python-numpy libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install libavcodec-dev libavformat-dev libjpeg.dev libtiff4.dev libswscale-dev libjasper-dev

$ unzip OpenCV-2.4.9.zip

$ cd opencv-2.4.9

$ mkdir build

$ cd build

$ cmake ..

$ sudo make -j4

尝试安装 ffmpeg:

进入 ffmpeg 目录，./configure --enable-gpl --enable-libx264 --enable-libmp3lame，然后就生成了新的 makefile 了。

执行 sudo make clean && make

sudo make instal

安装 x264：下载源代码、编译、安装

cd

git clone git://git.videolan.org/x264

cd x264,./configure --enable-shared　　//动态库,

make,

修改~x264-snapshot-20120214-2245-stable/input/lavf.c 里 248 行的

av_close_input_file(h->lavf);为：

avformat_close_input(h->lavf);

make install

安装 jpeg-8d

./configure ,make, make install

安装 libpng-1.5.14

./configure,make,make install

安装 lame

tar -zxvf lame-3.99.5.tar.gz

cd lame-3.99.5

./configure

make

sudo make install

安装完成后继续安装 opencv2.4.9

cd cmake,./configure,make –j2,

安装 jbigkit-bin_2.1-3.1+b2_i386.deb

一年后~~~~~~~~~~~

**********************折腾了许久还是不行**********************

**********************卸载 2.4.9，换 3.2*************************

1.安装所需库

sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev libgtk-3-dev

sudo apt-get install libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

sudo apt install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev

sudo apt-get -y install libavresample-dev libgphoto2-dev

2.下载 opencv3.2

＃若安装其它版本，将下面的 3.2.0.zip 代替成相应版本号即可

wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.2.0.zip

unzip opencv3.2.0.zip

cd ~/opencv-3.2.0

mkdir release

cd release

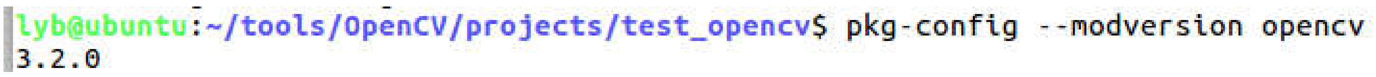
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..

make

sudo make install

查看：

pkg-config --modversion opencv



```
lyb@ubuntu:~/tools/OpenCV/projects/test_opencv$ pkg-config --modversion opencv
3.2.0
```

在/usr/local/lib 下面，在/etc/ld.so.conf.d/下面新建一个 opencv.conf，里面写入/usr/local/lib，最后执行下 sudo ldconfig -v 即可（每次安装 opencv 都需要执行这一步）

3.测试

参考文件夹 test_opencv

注意编译指令的使用：

1.g++ -o openpicture openpicture.cpp -I /usr/local/include -I /usr/local/include/opencv -I /usr/local/include/opencv2 -L /usr/local/lib /usr/local/lib/*.so

2. g++ `pkg-config opencv --cflags` opencv.cpp   -o opencv `pkg-config opencv --libs`

7. 在 Ubuntu 上安装交叉编译环境

参考网络。

8. 程序下载调试

通过网络(ssh)传送可执行文件到 Linux
1. root 账户登陆。
2. 输入 “udhcpc” 自动获取 ip。
3. 输入 “ifconfig” 查询获取的 IP 地址这个例子中 DE1-SOC 获得的 IP
4. “scp my_first_hps root@192.168.1.113:/home/root”，把文件拷贝到 “/home/root”路径下.

9. 设置程序开机自启动

****************** console_linux4.5.img******************

查看系统当前运行在什么级别：runlevel,显示 N 5,表示系统运行在级别为 5；

9.1. 直接在/etc/rc5.d/中创建脚本文件

接下来进入/etc/rc5.d/ 建立脚本文件 S89lidar.sh ， 文件里输入： /home/root/./lidar&，保存。

其中：/home/root/为我的应用程序 lidar 的存放路径，”./”为执行的意思，“&”为后台运行。

9.2. 标准做法

因为有时候不光要在 rc.5 里放，还可能放在 rc.2，因为不带界面启动时是 runlevel2。所以，合理的做法是将脚本文件和应用程序放在一起，在 rcN.d/目录中创建软连接。

这里有两点要注意：

第一，ln 命令会保持每一处链接文件的同步性，也就是说，不论你改动了哪一处，其它的文件都会发生相同的变化；

第二，ln 的链接又软链接 和硬链接两种，

软链接就是 ln -s src dst,它只会在你选定的位置上生成一个文件的镜像，不会占用磁盘空间，

硬链接 ln src dst,没有参数-s, 它会在你选定的位置上生成一个和源文件大小相同的文件，无论是软链接还是硬链接，文件都保持同步变化。

连接的删除：

直接 rm dst

例如：rm /usr/local/bin/hello

如果你用 ls 察看一个目录时，发现有的文件后面有一个@的符号，那就是一个用 ln 命令生成的文件，用 ls -l 命令去察看，就可以看到显示的 link 的路径了

****************** de10_nano_linux_console.img******************

对于这个系统，前面的方法会失效

```
systemctl daemon-reload
```

systemctl status lidar_de10nano.service

chmod 644 lidar.service

9.2.1. 编写脚本

编写一个 /opt/hello.sh 脚本

sudo vim /opt/hello.sh

/opt/hello.sh

```
#!/bin/bash
while true
do
    echo hello world >> /tmp/hello.log
    sleep 1
done
```

赋予执行权限。

```
sudo chmod 0755 /opt/hello.sh
```

9.2.2. 在/etc/systemd/system/ 下创建 Unit 定义文件

```
sudo vim /etc/systemd/system/hello.service
```

内容如下

```
/etc/systemd/system/hello.service
[Unit]
Description = hello daemon

[Service]
ExecStart = /opt/hello.sh
Restart = always
Type = simple
[Install]
WantedBy = multi-user.target
```

ExecStart 中填写想要执行的脚本。

Restart = always 是指进程或服务意外故障的时候可以自动重启的模式。

※Unit 文件的详细写法会另外给出。

(Type = simple 指默认的选项没有必要填写，或可理解成其余选项均为系统默认。)

9.2.3. 把 Unit 添加进 Service

使用 systemctl list-unit-files --type=service 命令，出现如下图所示即为正常。

```
$ sudo systemctl list-unit-files --type=service | grep hello
hello.service                                  disabled
OK!
```

9.2.4. enable 服务后使之 start

之后系统将以一般服务的形式对待它。

# 开机自启动 on

$ sudo systemctl enable hello

# 单次启动

$ sudo systemctl start hello

运行状态确认

$ sudo systemctl status hello

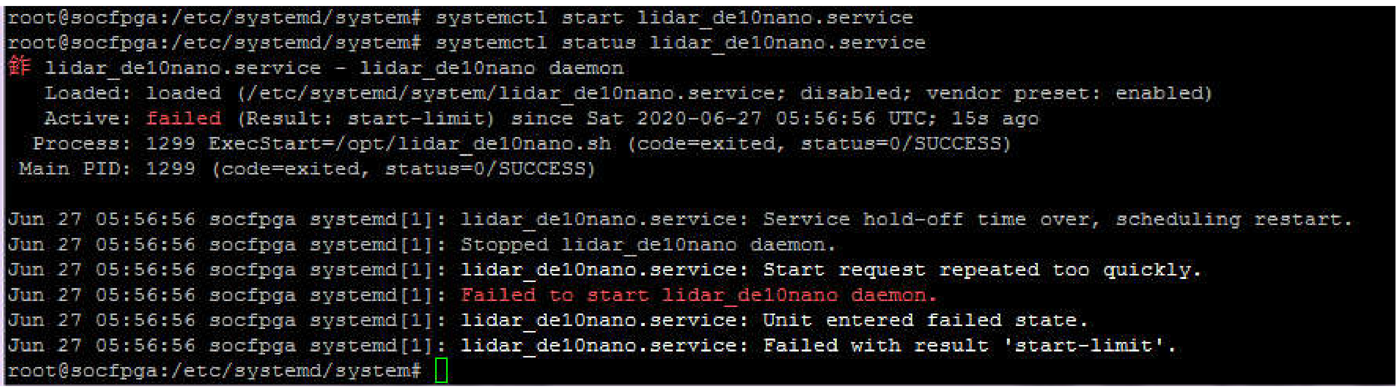
```
hello.service - hello daemon
Loaded: loaded (/etc/systemd/system/hello.service; enabled)
Active: active (running) since 2018-05-19 09:02:19 UTC; 2min 54s ago
Main PID: 551 (hello.sh)
    CGroup: /system.slice/hello.service
            ├── 551 /bin/bash /opt/hello.sh
            └──2062 sleep 1
```

9.2.5. 重启机器，查看服务是否正常自动启动

$ sudo reboot

重启后，如正常显示 hello 服务即为操作政工。

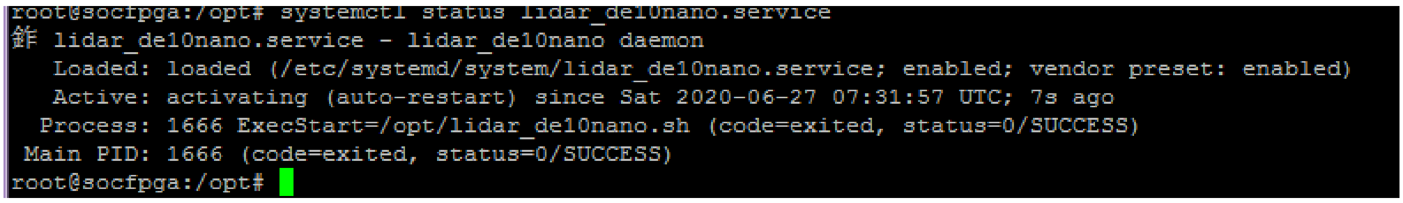
9.2.6. 报错解决



```
root@socfpga:/etc/systemd/system# systemctl start lidar_de10nano.service
root@socfpga:/etc/systemd/system# systemctl status lidar_de10nano.service
鈼 lidar_de10nano.service - lidar_de10nano daemon
   Loaded: loaded (/etc/systemd/system/lidar_de10nano.service; disabled; vendor preset: enabled)
   Active: failed (Result: start-limit) since Sat 2020-06-27 05:56:56 UTC; 15s ago
  Process: 1299 ExecStart=/opt/lidar_de10nano.sh (code=exited, status=0/SUCCESS)
 Main PID: 1299 (code=exited, status=0/SUCCESS)

Jun 27 05:56:56 socfpga systemd[1]: lidar_de10nano.service: Service hold-off time over, scheduling restart.
Jun 27 05:56:56 socfpga systemd[1]: Stopped lidar_de10nano daemon.
Jun 27 05:56:56 socfpga systemd[1]: lidar_de10nano.service: Start request repeated too quickly.
Jun 27 05:56:56 socfpga systemd[1]: Failed to start lidar_de10nano daemon.
Jun 27 05:56:56 socfpga systemd[1]: lidar_de10nano.service: Unit entered failed state.
Jun 27 05:56:56 socfpga systemd[1]: lidar_de10nano.service: Failed with result 'start-limit'.
root@socfpga:/etc/systemd/system#
```

/opt/lidar.sh

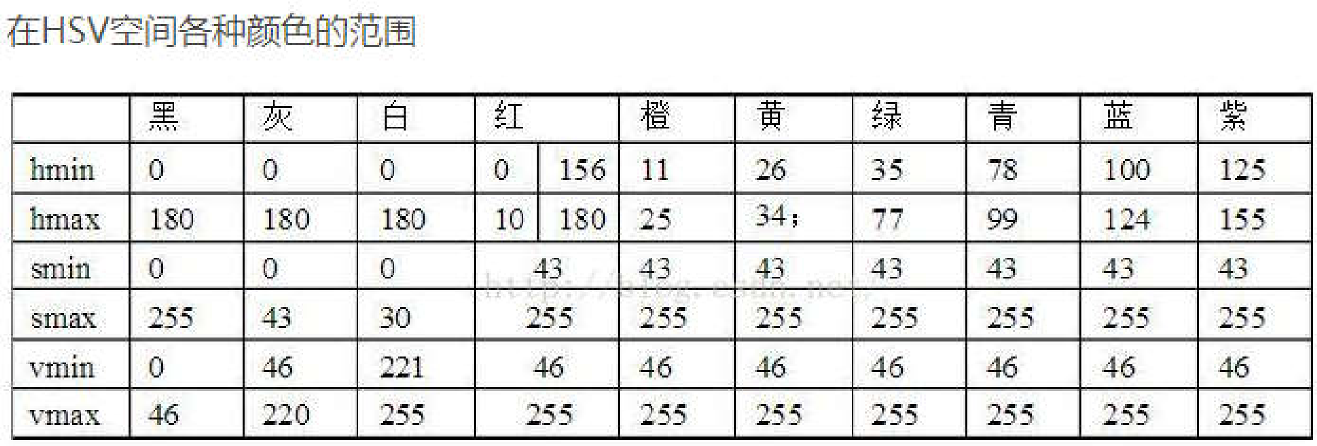
```
#!/bin/sh
cd /home/root/
./lidar
```



```
root@socfpga:/opt# systemctl status lidar_de10nano.service
鈼 lidar_de10nano.service - lidar_de10nano daemon
   Loaded: loaded (/etc/systemd/system/lidar_de10nano.service; enabled; vendor preset: enabled)
   Active: activating (auto-restart) since Sat 2020-06-27 07:31:57 UTC; 7s ago
  Process: 1666 ExecStart=/opt/lidar_de10nano.sh (code=exited, status=0/SUCCESS)
 Main PID: 1666 (code=exited, status=0/SUCCESS)
root@socfpga:/opt# 
```

10. OpenCV 中颜色识别

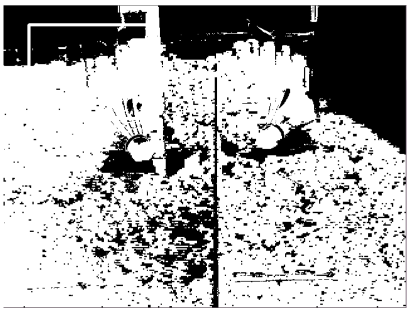
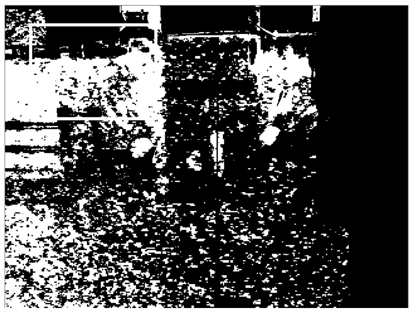
羽毛球参数：

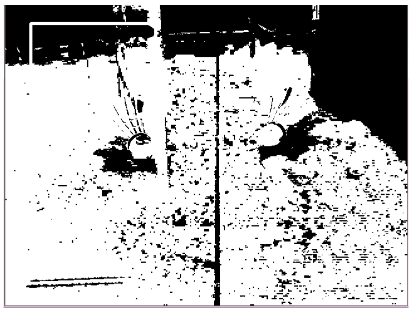


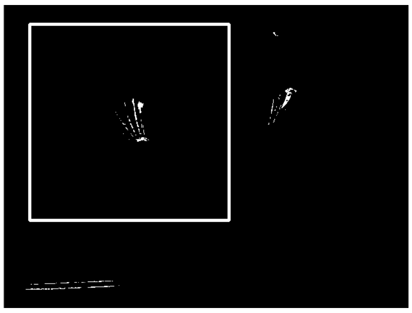
在HSV空间各种颜色的范围

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34；	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

HSV参数	开灯	关灯

int minh = 40;
int maxh = 120;
int mins = 3;
int maxs = 50;
int minv = 100;
int maxv = 210;
		

int minh = 40;
int maxh = 120;
int mins = 3;
int maxs = 30;
int minv = 100;
int maxv = 225;
		

int minh = 40;
int maxh = 120;
int mins = 3;
int maxs = 30;
int minv = 200;
int maxv = 255;
		

int minh = 40;
int maxh = 120;
int mins = 3;
int maxs = 49;
int minv = 164;
int maxv = 220;		

11. 摄像头—激光雷达标定（参考论文）

候选框大小标定		
顺序	参数	拟合曲线

1	底部：[(236@79),(238@116)], 羽毛球大小 197*280,191*282 ---237 290*290
	中部：[(1014@82),(1015@121],羽毛球大小 67*82，60*83 ---1014 90*90
	顶部：[(1692@82),(1727@122)],羽毛球大小 42*54,39*54 ---1708 60*60
	角度定为 82--120，即角度位置为
	像素中心位置：
	(角度值-82)*640/(120-82)
		候选框大小：
		拟合方程式：Y = (A - D) / [1 + (X/C)^B] + D
		参数：
		A = 290.087087091475
		B = 6.70452998764013
		C = 768.104581633397
		D = 58.916153673649

2	增加参数[560@101],大小 142*111，按每个点拟合	拟合方程式：Y = (A - D) / [1 + (X/C)^B] + D
		参数：
		A = 416.622303830546
		B = 1.87028747527089
		C = 315.966144666733
		D = 47.3964798918816

Position_Y 曲线拟合		

	R235@467,R379@320, R437@282,R530@243,R616@219	拟合方程式：y=a+b/x 参数：
		a = 66.4245701871308
		b = 94456.7457863983

	R235@470,R379@325, R437@285,R530@245,R616@225	

X: 236	Y: 280，282，188，142，82，83，54，60

238 341 560 1014 1015 1692 1727	

12. 问题与解决

12.1. Quartus 编译报错问题

地址不够用——》解锁后重新分配地址
编译显示内存不够用——》重新分配管脚

```
sudo rm -r /usr/local/include/opencv2 /usr/local/include/opencv /usr/include/opencv /usr/include/opencv2 /usr/local/share/opencv /usr/local/share/OpenCV /usr/share/opencv /usr/share/OpenCV /usr/local/bin/opencv* /usr/local/lib/libopencv*
```