[L28]双核AMP

一般来说,Linux在多核系统是运行在SMP状态,多余Linux来说,不管是异构,同构,多处理器都能很好的支持,但是两个核心可以一个运行Linux,一个运行FreeRTOS,这个在Xilinx还有描述,建议阅读XAPP1078和XAPP1079.

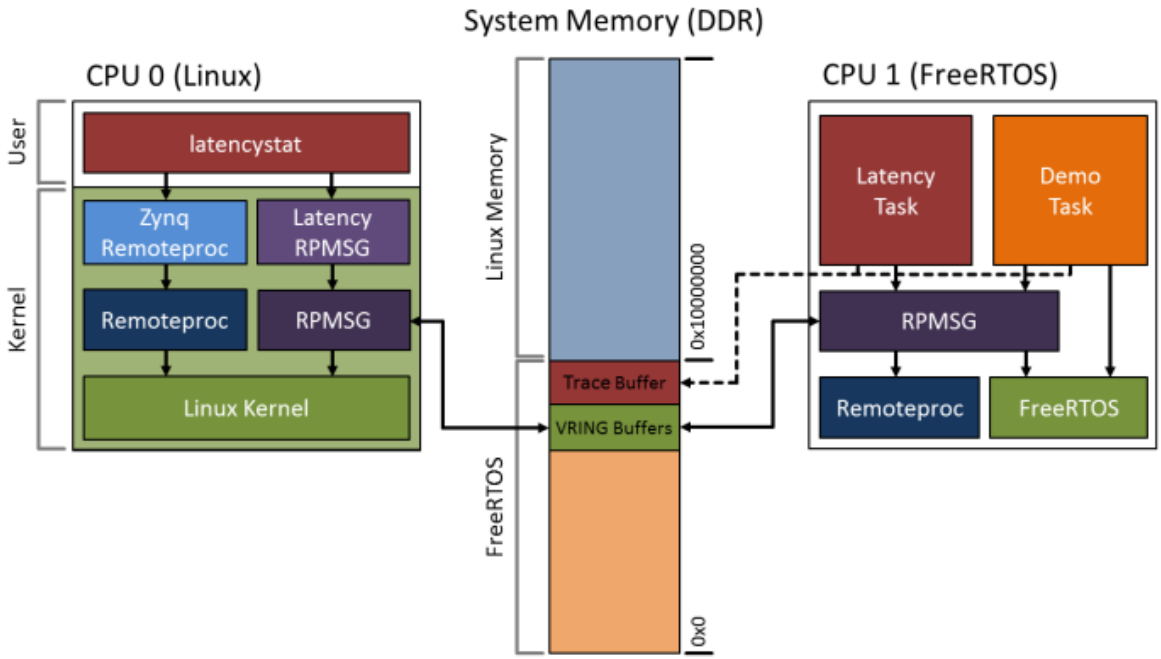
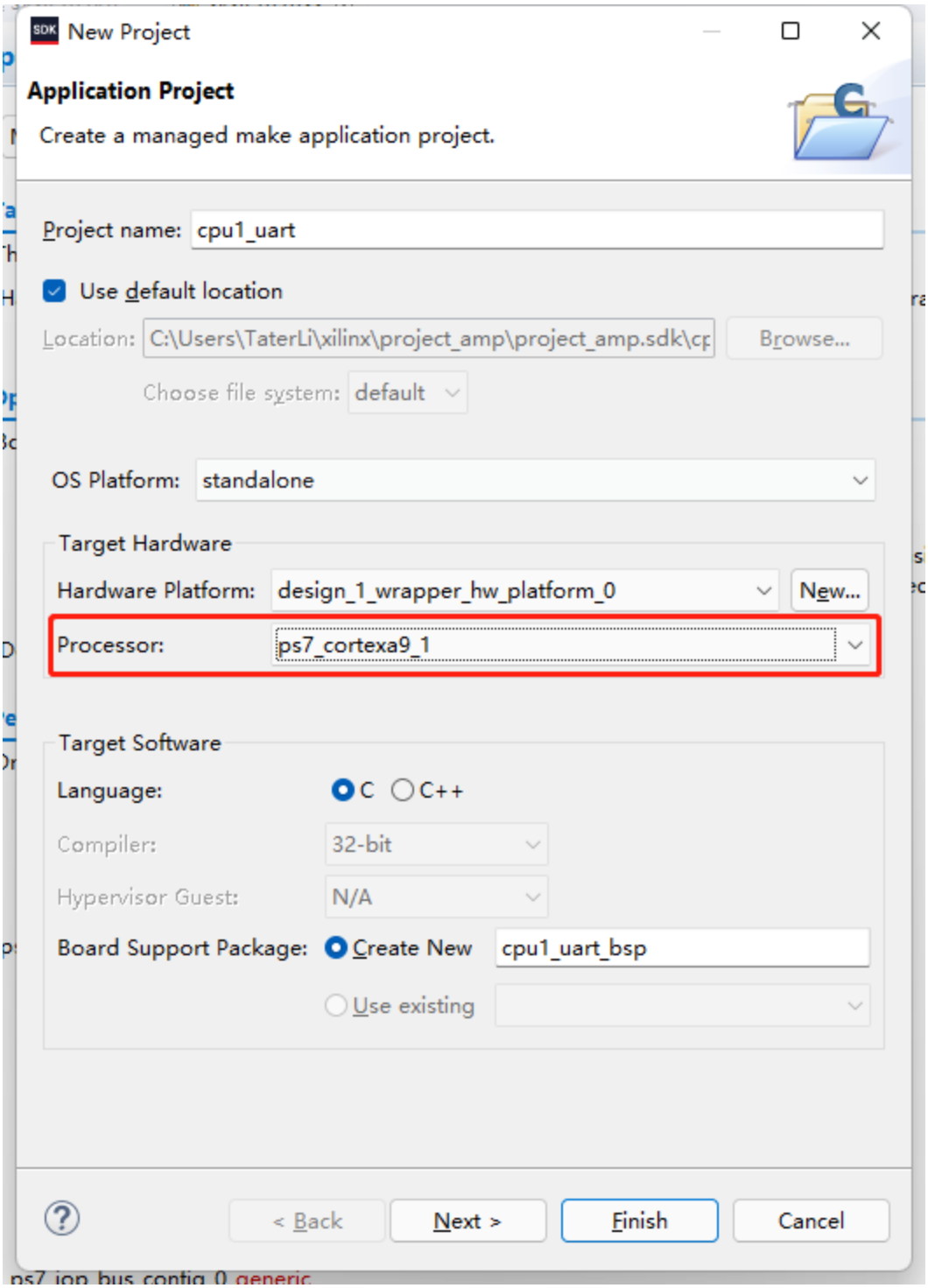


Figure 1: Linux-FreeRTOS AMP Reference Design

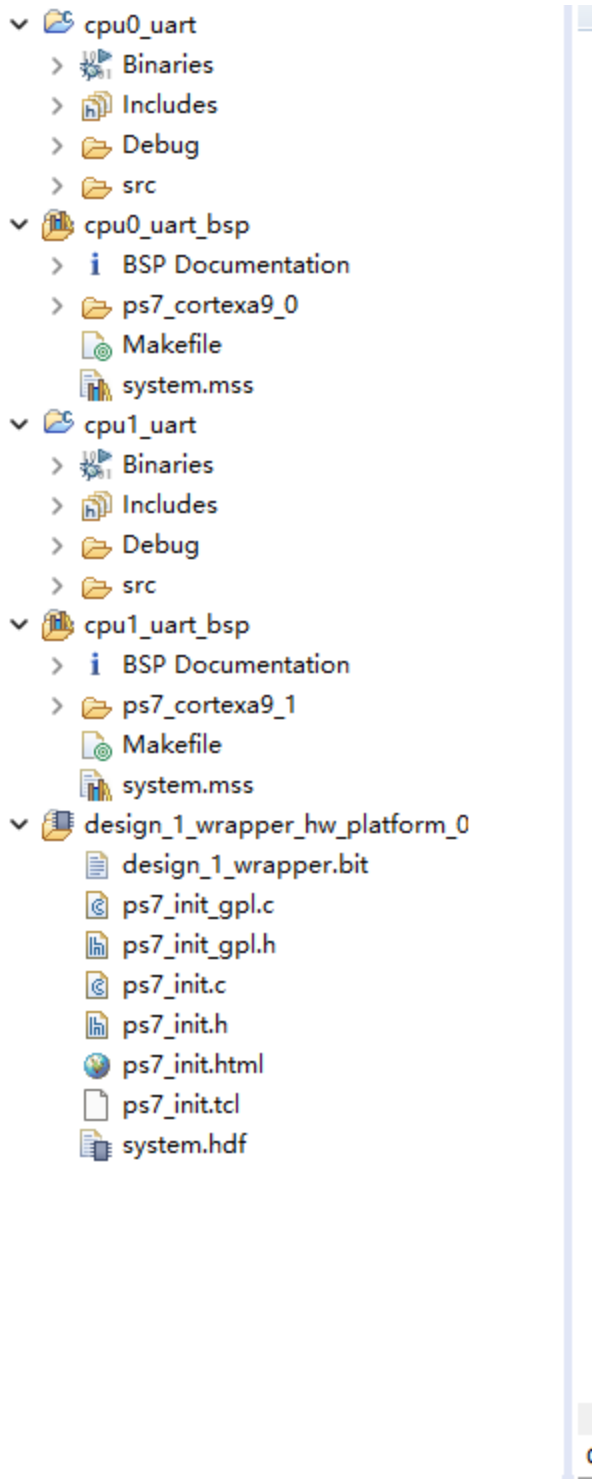
其实现在都是OpenAMP,所以其实其他系统也是这样来操作的,不过我们既然说Zynq,就拿Zynq说事,为了简单,先学习裸机AMP,再学习带Linux+FreeRTOS的AMP,他们差不多.

每个CPU都有自己的核上RAM,整个系统有OCRAM和DDR,所以我们对待相同的内存要有策略,否则会读写冲突,或者读写缓存导致不一致性,双核之间通信靠软件SGI中断,有16个中断号,CPU0先启动,然后启动CPU1,基础知识先交代到这里.

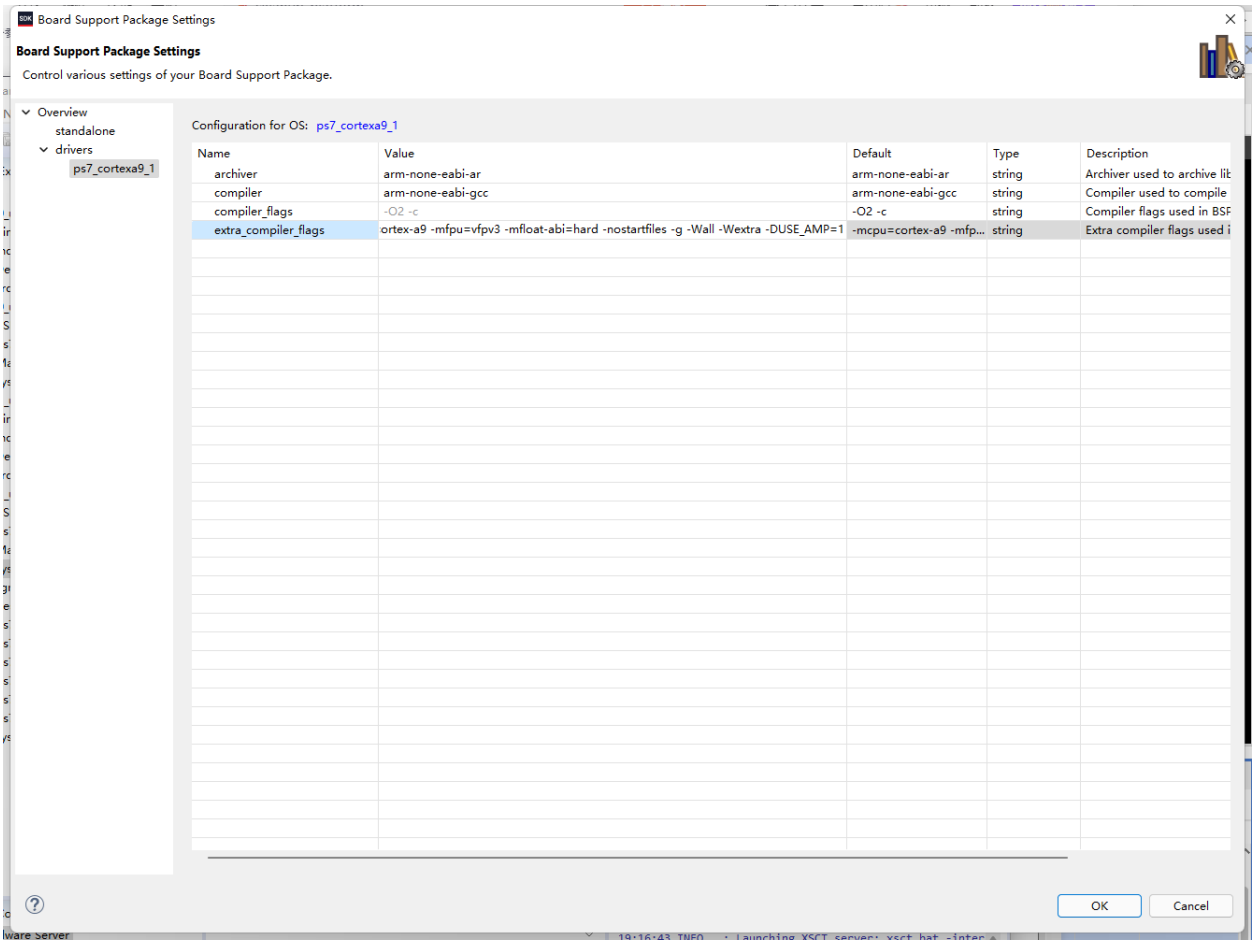
为了最简测试,我开启2个LED和2个串口,创建两个工程,唯一区别就是选择不同的CPU,暂时先不用OpenAMP,创建默认Hello World.


New Project
Application Project
Create a managed make application project.
Project name: cpu1_uart
Use default location
Location: C:\Users\TaterLi\xilinx\project_amp\project_amp.sdk\cp
Browse...
Choose file system: default
OS Platform: standalone
Target Hardware
Hardware Platform: design_1_wrapper_hw_platform_0
New...
Processor: ps7_cortexa9_1
Target Software
Language: C C++
Compiler: 32-bit
Hypervisor Guest: N/A
Board Support Package: Create New cpu1_uart_bsp
Use existing
< Back
Next >
Finish
Cancel

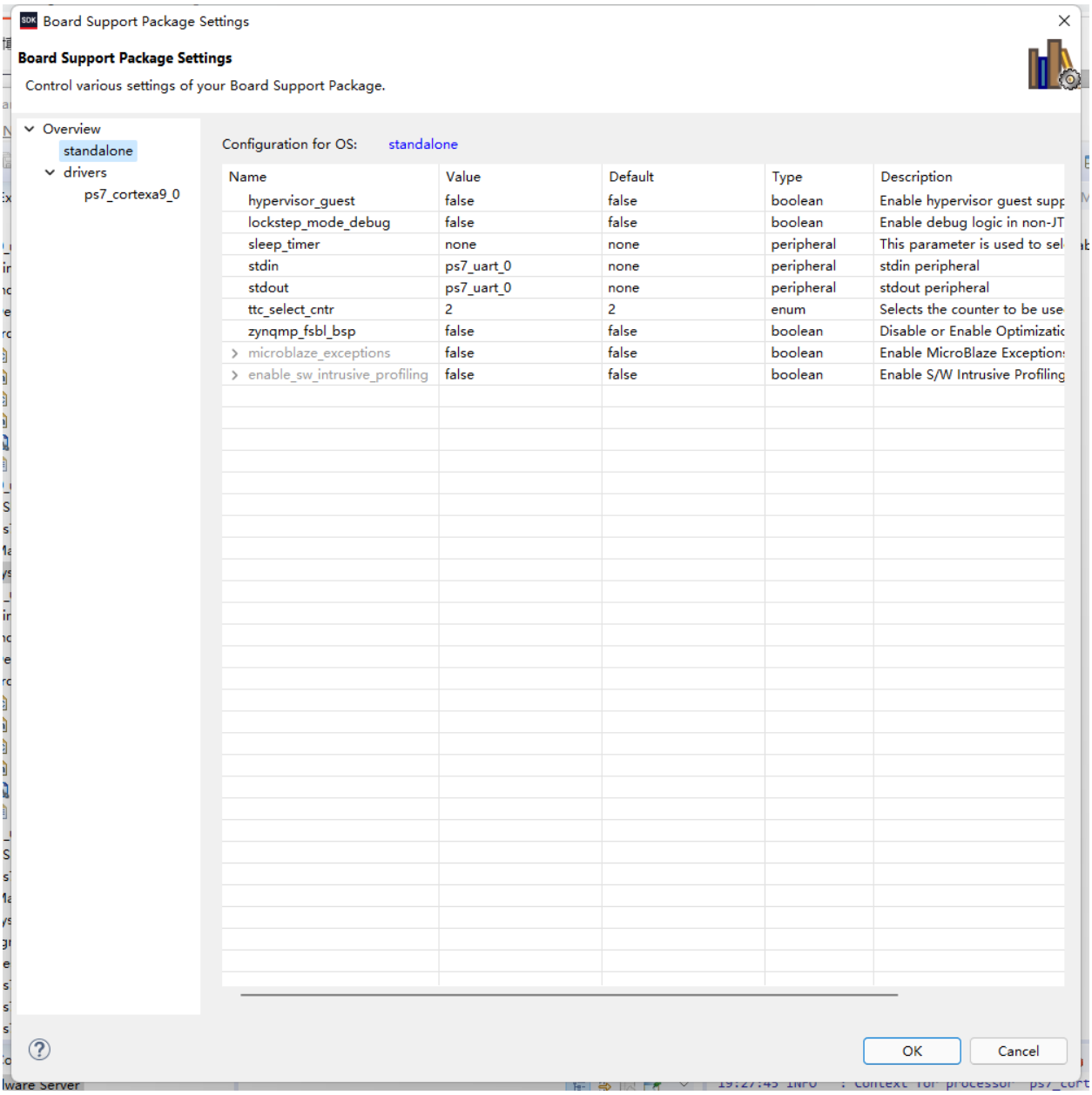
结构如下:



给cpu1_uart工程定义**-DUSE_AMP=1**,目的是为了不用L2.

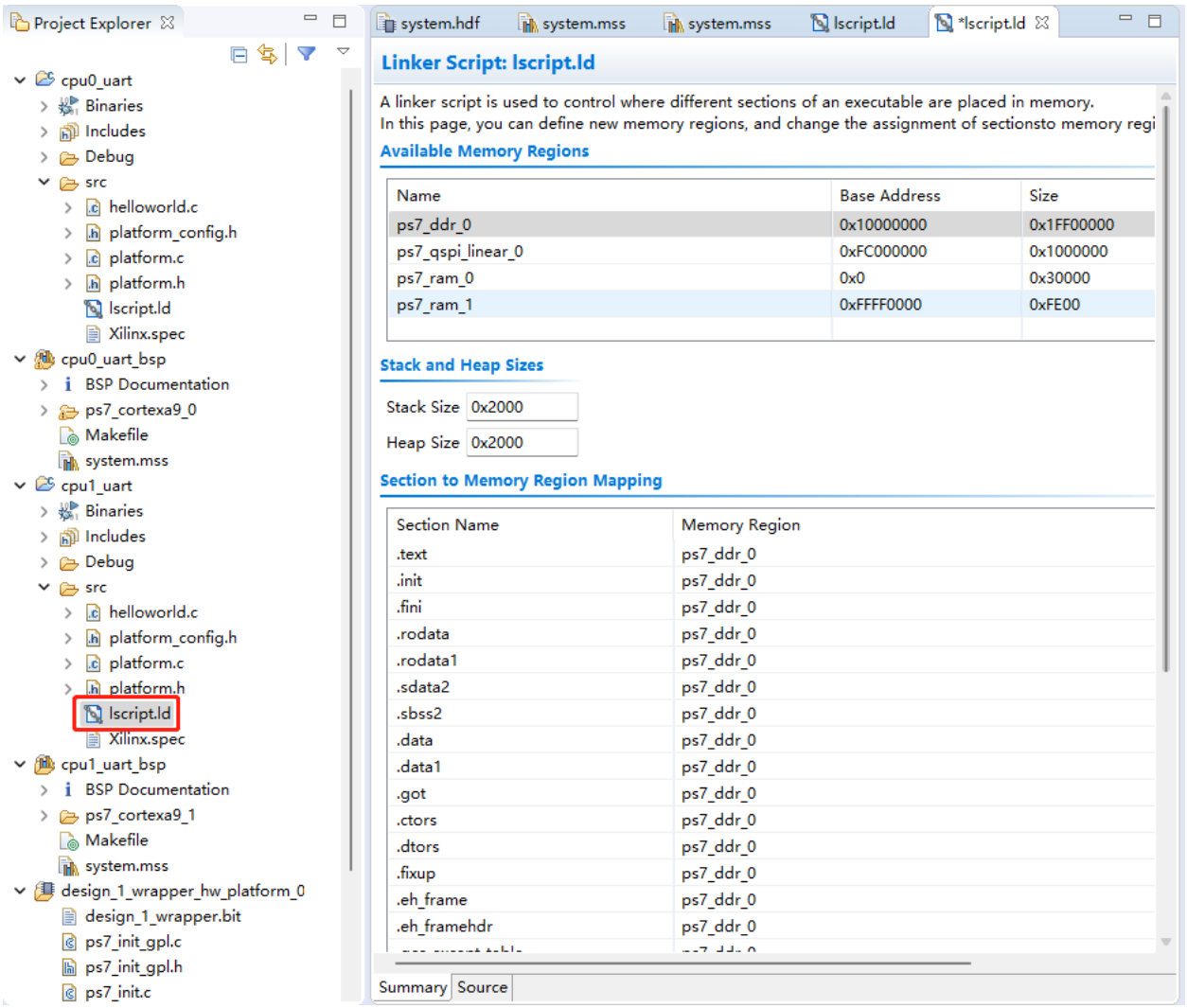


修改串口使他们用不同的串口.

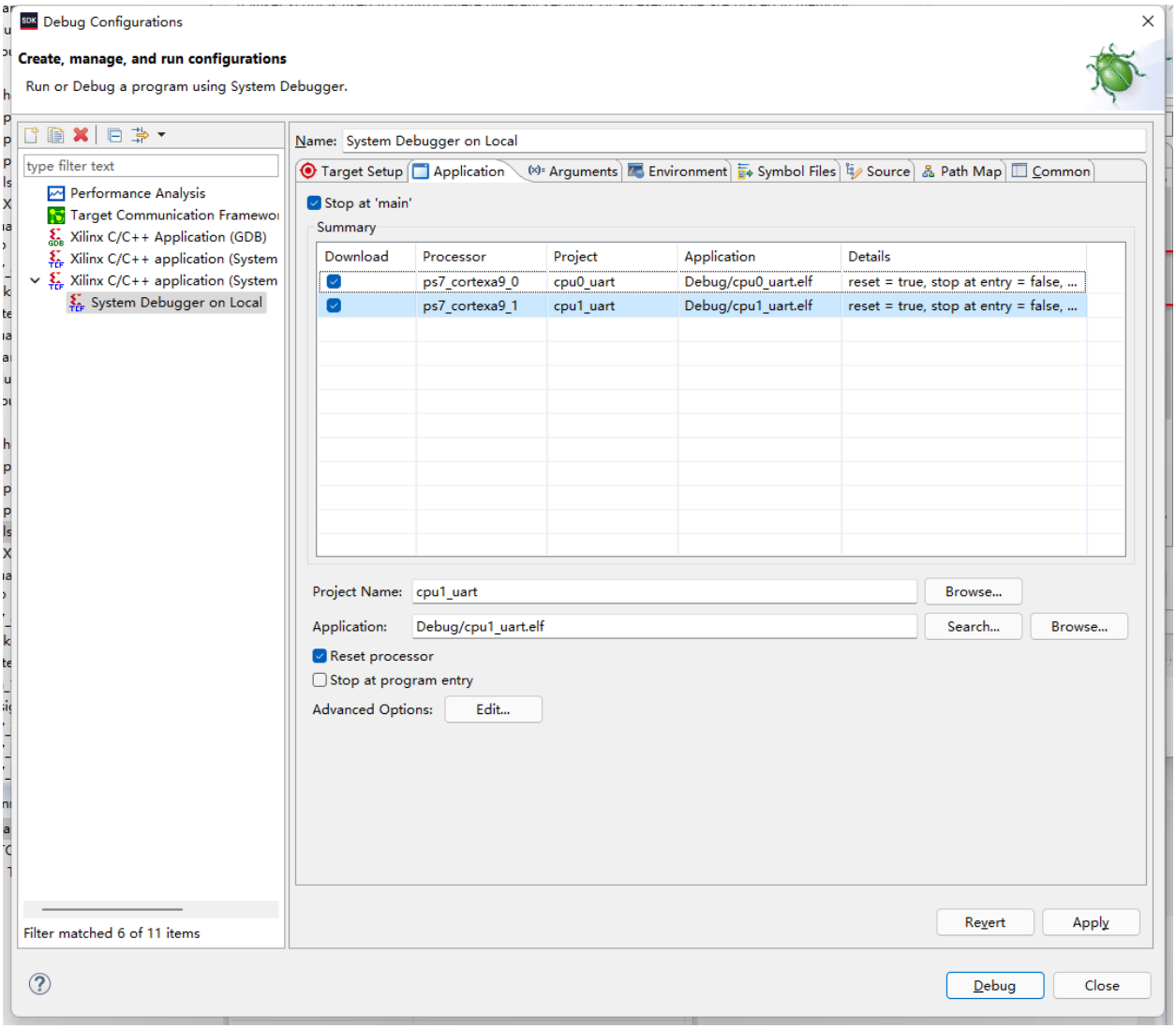


修改ld文件,使得两个任务使用不同的内存.

1. CPU0 ⇒ Base:0x100000 Size:0x0FF00000
2. CPU1 ⇒ Base:0x10000000 Size:0x0FF00000



创建一个调试配置并把两个CPU都选上.



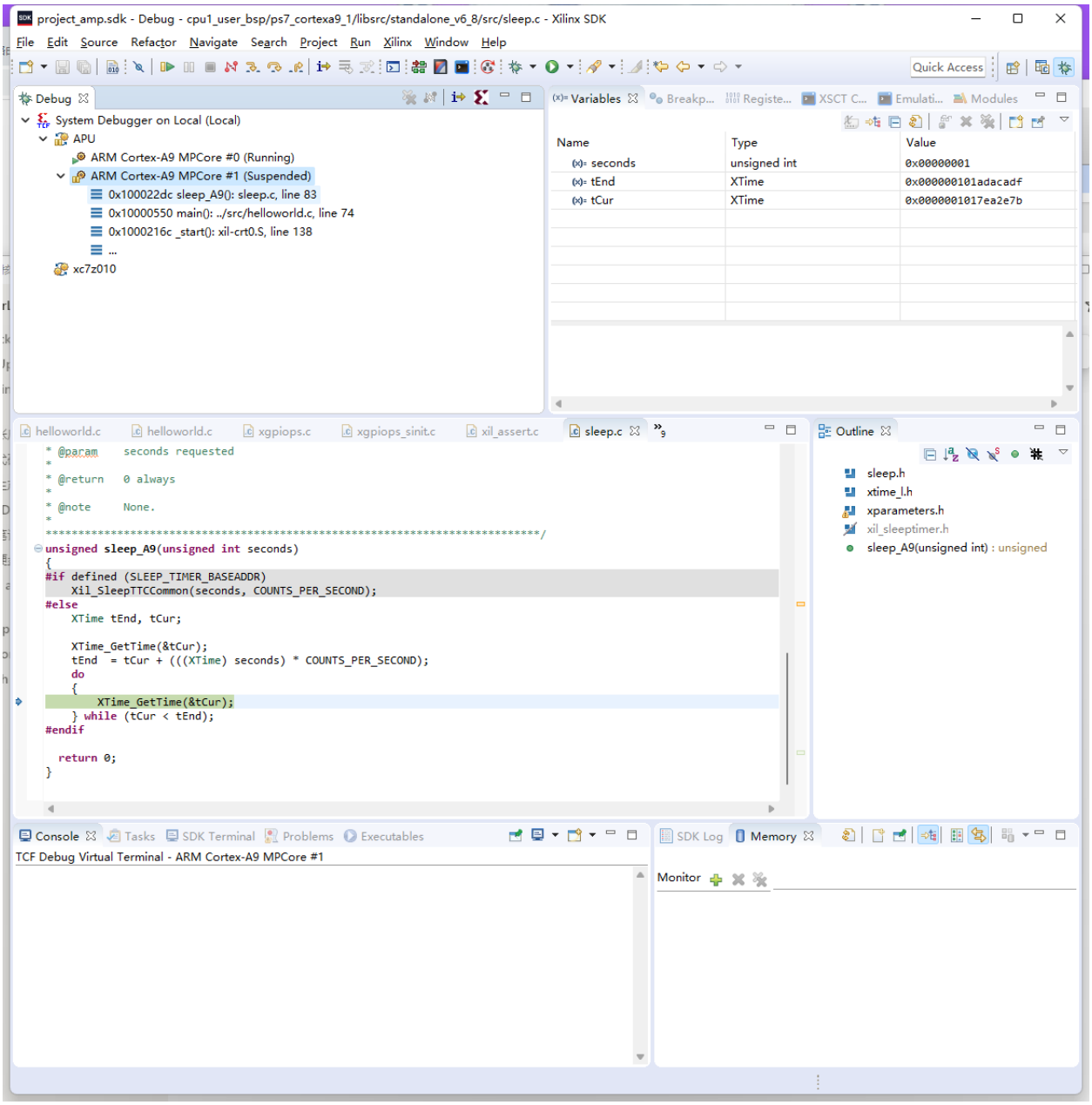
然后我把程序修改了一下,两个核心程序有些差距.

```
/*****************************************************************************
*
* Copyright (C) 2009 - 2014 Xilinx, Inc.  All rights reserved.
*
* Permission is hereby granted, free of charge, to any person obtaining a copy
* of this software and associated documentation files (the "Software"), to deal
* in the Software without restriction, including without limitation the rights
* to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
* copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
*
```

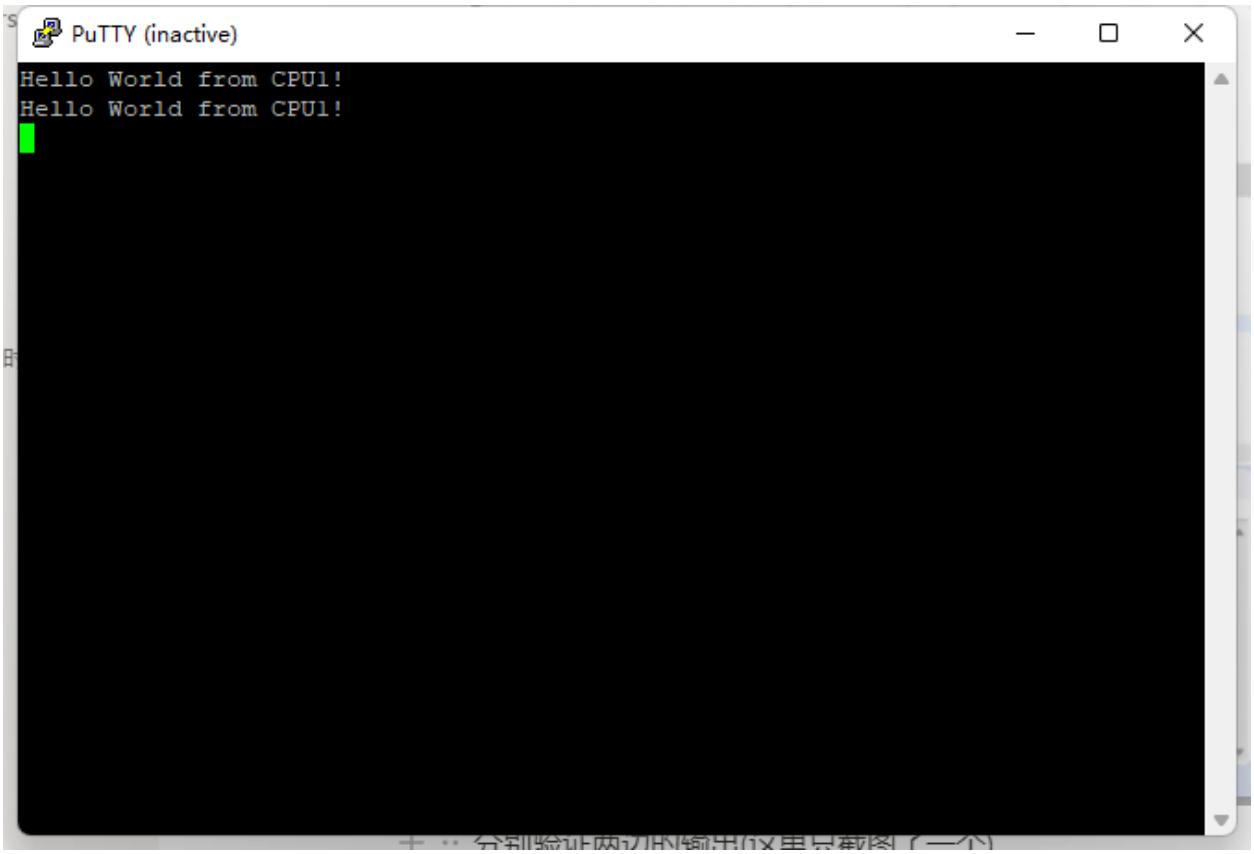
```
* Use of the Software is limited solely to applications:
* (a) running on a Xilinx device, or
* (b) that interact with a Xilinx device through a bus or interconnect.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* XILINX  BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
* OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*
* Except as contained in this notice, the name of the Xilinx shall not be used
* in advertising or otherwise to promote the sale, use or other dealings in
* this Software without prior written authorization from Xilinx.
*
******************************************************************************/

/*
 * helloworld.c: simple test application
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ------------------------------------------------
 * | UART TYPE   BAUD RATE                        |
 * ------------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xparameters.h"
#include "xgpiops.h"
#include "sleep.h"

int main()
{
  XGpioPs GPIOInst;
  XGpioPs_Config *GPIOCfg;

    init_platform();

    print("Hello World from CPU0!\n\r");
    GPIOCfg = XGpioPs_LookupConfig(XPAR_XGPIOPS_0_DEVICE_ID);
    XGpioPs_CfgInitialize(&GPIOInst,GPIOCfg,GPIOCfg->BaseAddr);

    // EMIO - GPIO0
    XGpioPs_SetDirectionPin(&GPIOInst,54,1);
    XGpioPs_SetOutputEnablePin(&GPIOInst,54,1);
```

```
    for(;;){
      XGpioPs_WritePin(&GPIOInst,54,1);
      sleep(1);
      XGpioPs_WritePin(&GPIOInst,54,0);
      sleep(1);
    }

    cleanup_platform();
    return 0;
}
```

另一个核心程序稍微区别,大致IO配置,串口打印内容稍微改变.然后看到他们各自跑在自己的main上面.

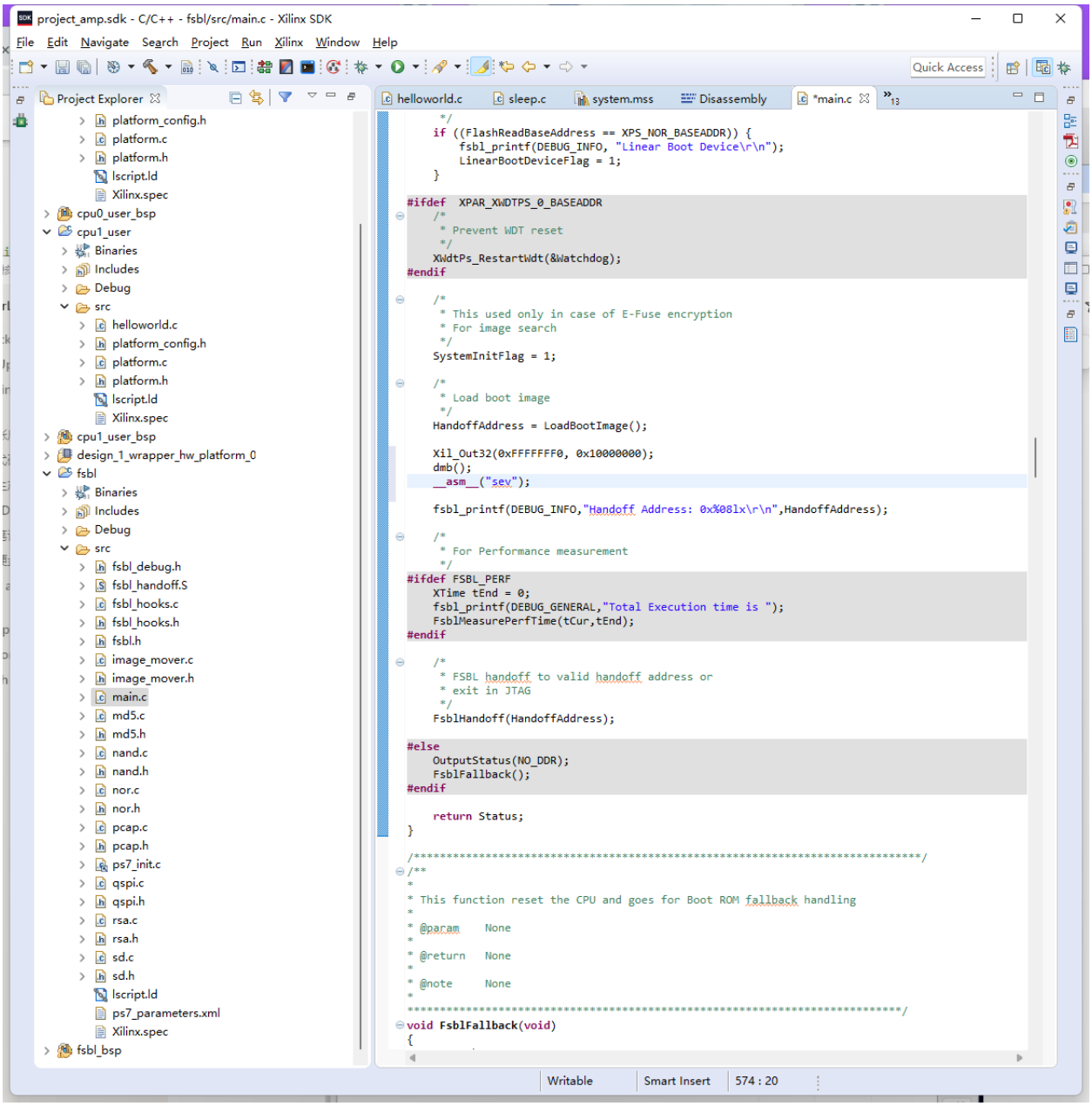


分别验证两边的输出(这里只截图了一个).

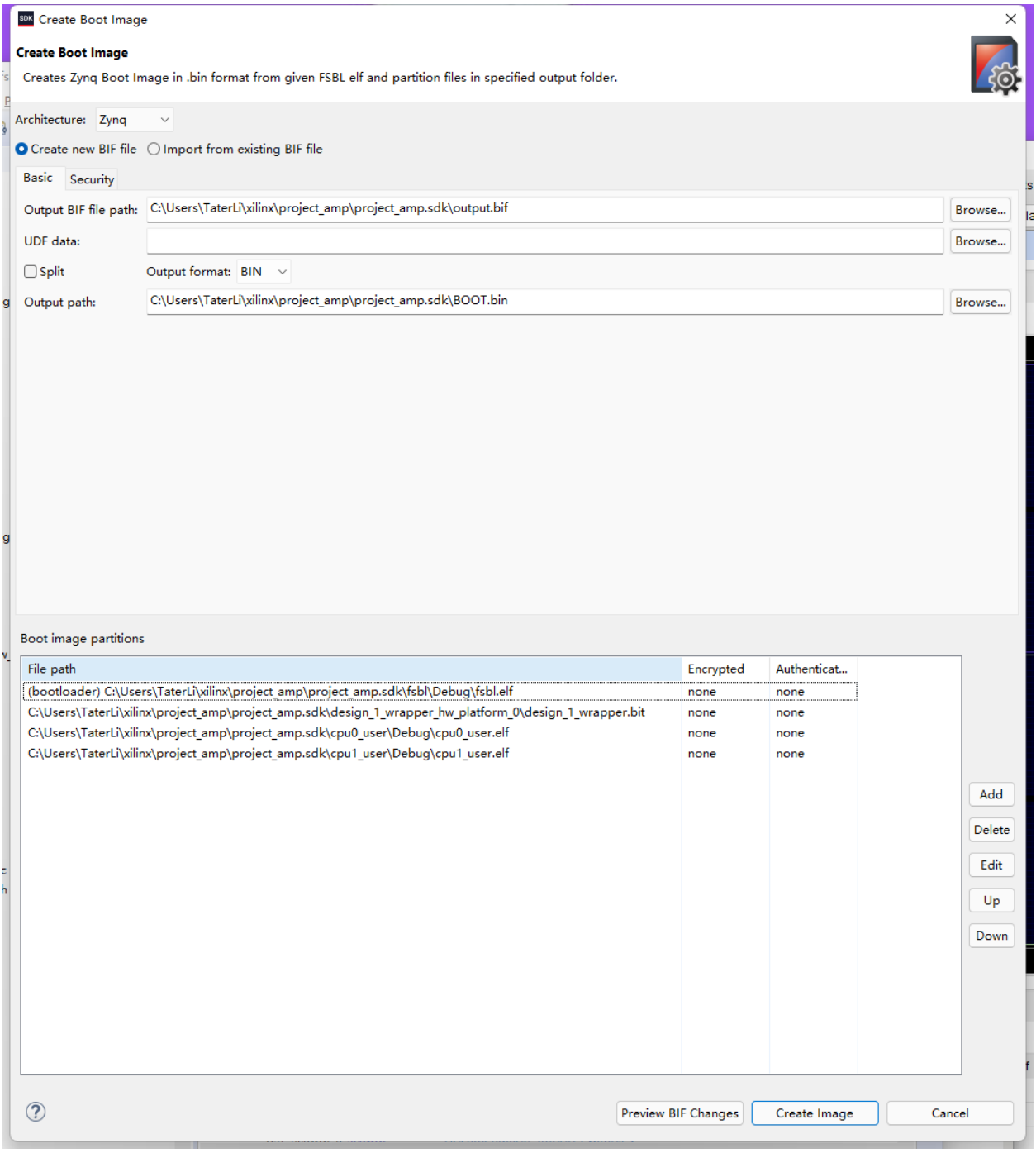


但是现在如果固化成BOOT.bin依然是没法引导的,因为在调试时候,CPU1是由调试器启动的,实际上要由CPU0启动,启动过程在UG585的Starting Code on CPU1有讲解,大致就是向0xFFFFFFF0写入CPU1启动RAM地址并等待执行完成,添加到fsbl工程的LoadBootImage之后.

```
Xil_Out32(0xFFFFFFF0, 0x10000000);
dmb();
__asm__("sev");
```


project_amp.sdk - C/C++ - fsbl/src/main.c - Xilinx SDK
File Edit Navigate Search Project Run Xilinx Window Help
Quick Access
Project Explorer
platform_config.h
platform.c
platform.h
lscript.ld
Xilinx.spec
cpu0_user_bsp
cpu1_user
Binaries
Includes
Debug
src
helloworld.c
platform_config.h
platform.c
platform.h
lscript.ld
Xilinx.spec
cpu1_user_bsp
design_1_wrapper_hw_platform_0
fsbl
Binaries
Includes
Debug
src
fsbl_debug.h
fsbl_handoff.S
fsbl_hooks.c
fsbl_hooks.h
fsbl.h
image_mover.c
image_mover.h
main.c
md5.c
md5.h
nand.c
nand.h
nor.c
nor.h
pcap.c
pcap.h
ps7_init.c
qspi.c
qspi.h
rsa.c
rsa.h
sd.c
sd.h
lscript.ld
ps7_parameters.xml
Xilinx.spec
fsbl_bsp
helloworld.c
sleep.c
system.mss
Disassembly
*main.c
*/
if ((FlashReadBaseAddress == XPS_NOR_BASEADDR)) {
fsbl_printf(DEBUG_INFO, "Linear Boot Device\r\n");
LinearBootDeviceFlag = 1;
}
#ifdef XPAR_XWDTPS_0_BASEADDR
/*
* Prevent WDT reset
*/
XWdtPs_RestartWdt(&Watchdog);
#endif
/*
* This used only in case of E-Fuse encryption
* For image search
*/
SystemInitFlag = 1;
/*
* Load boot image
*/
HandoffAddress = LoadBootImage();
Xil_Out32(0xFFFFFFF0, 0x10000000);
dmb();
__asm__("sev");
fsbl_printf(DEBUG_INFO,"Handoff Address: 0x%08lx\r\n",HandoffAddress);
/*
* For Performance measurement
*/
#ifdef FSBL_PERF
XTime tEnd = 0;
fsbl_printf(DEBUG_GENERAL,"Total Execution time is ");
FsblMeasurePerfTime(tCur,tEnd);
#endif
/*
* FSBL handoff to valid handoff address or
* exit in JTAG
*/
FsblHandoff(HandoffAddress);
#else
OutputStatus(NO_DDR);
FsblFallback();
#endif
return Status;
}
/**
* This function reset the CPU and goes for Boot ROM fallback handling
* @param None
* @return None
* @note None
void FsblFallback(void)
{
Writable
Smart Insert
574 : 20

固化测试,注意顺序:



现在对AMP有一定的理解了,其实Linux上AMP更简单,都有现成的例子,固件是从Linux加载,只需要提前保留内存就行,这里放一个我的实验记录.

链接:https://www.taterli.com/8718/ **UG1186 OpenAMP实验 (Vivado 2018.3)**

COM7 - PuTTY

```
echo test: sent : 472
 received payload number 463 of size 472

 sending payload number 464 of size 473
echo test: sent : 473
 received payload number 464 of size 473

 sending payload number 465 of size 474
echo test: sent : 474
 received payload number 465 of size 474

 sending payload number 466 of size 475
echo test: sent : 475
 received payload number 466 of size 475

 sending payload number 467 of size 476
echo test: sent : 476
 received payload number 467 of size 476

 sending payload number 468 of size 477
echo test: sent : 477
 received payload number 468 of size 477

 sending payload number 469 of size 478
echo test: sent : 478
 received payload number 469 of size 478

 sending payload number 470 of size 479
echo test: sent : 479
 received payload number 470 of size 479

 sending payload number 471 of size 480
echo test: sent : 480
 received payload number 471 of size 480

 ****************************************

 Echo Test Round 0 Test Results: Error count = 0

 ****************************************
root@openamp:~#
```