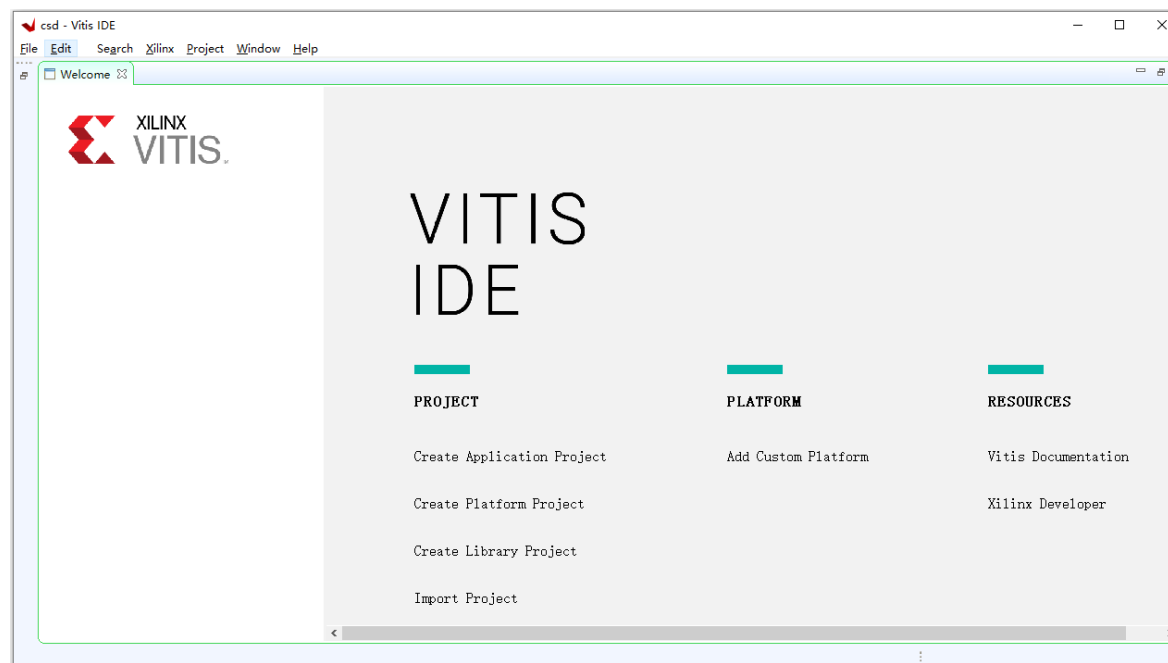
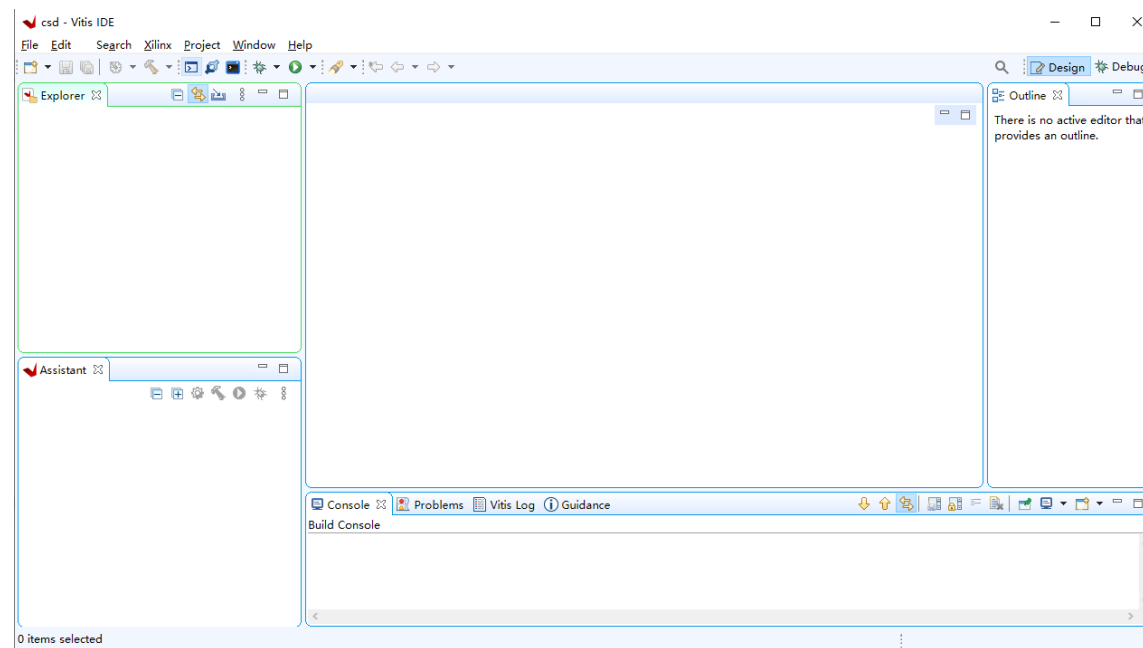


1. 新建Vitis空白工程

①打开Vitis，自行选择工作目录

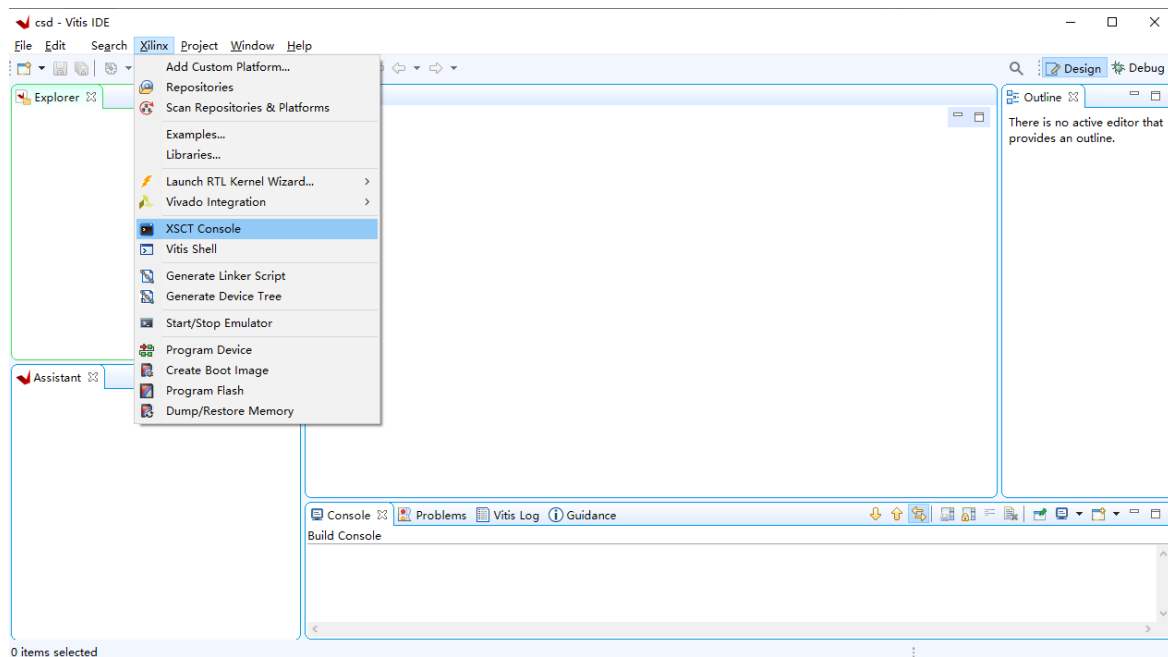


②关闭Welcome页面，即会出现一个空白项目，如下图

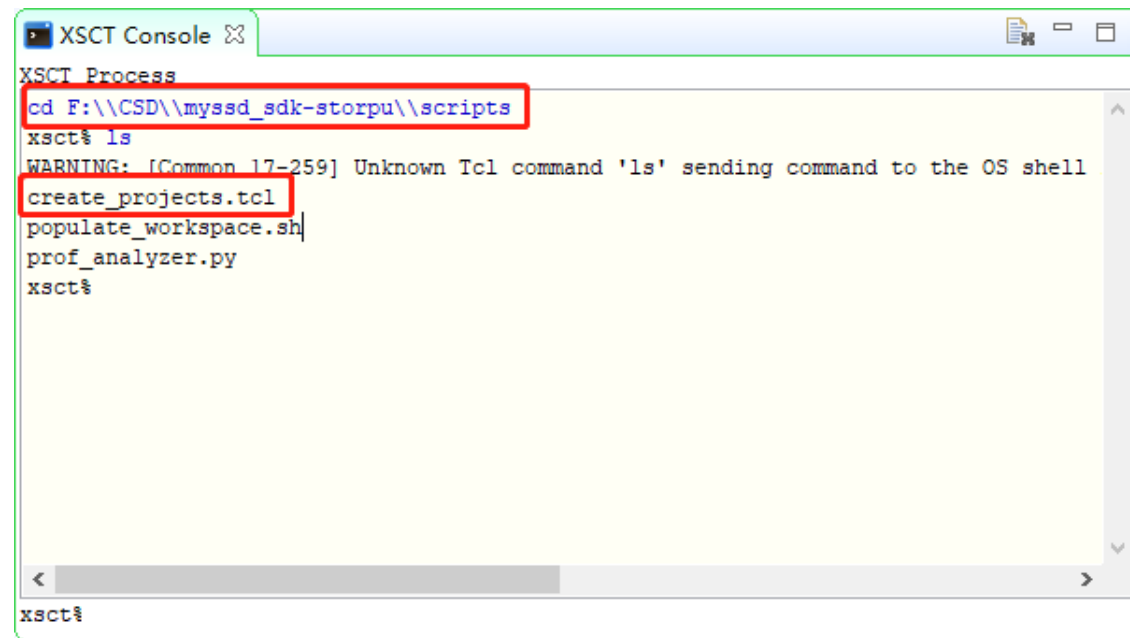


2. 打开XSCT Console

①打开Vitis中的XSCT Console

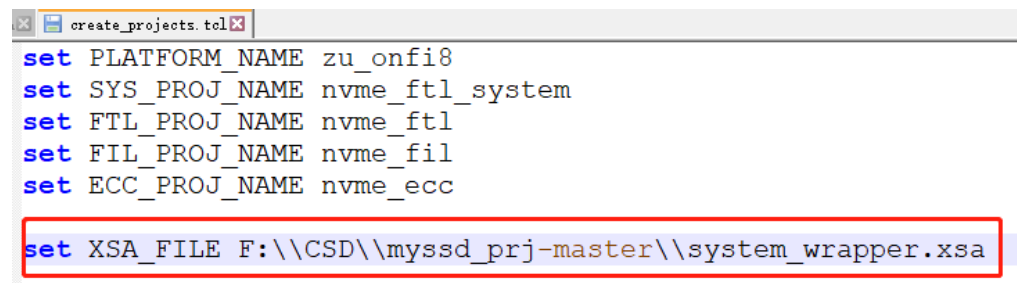
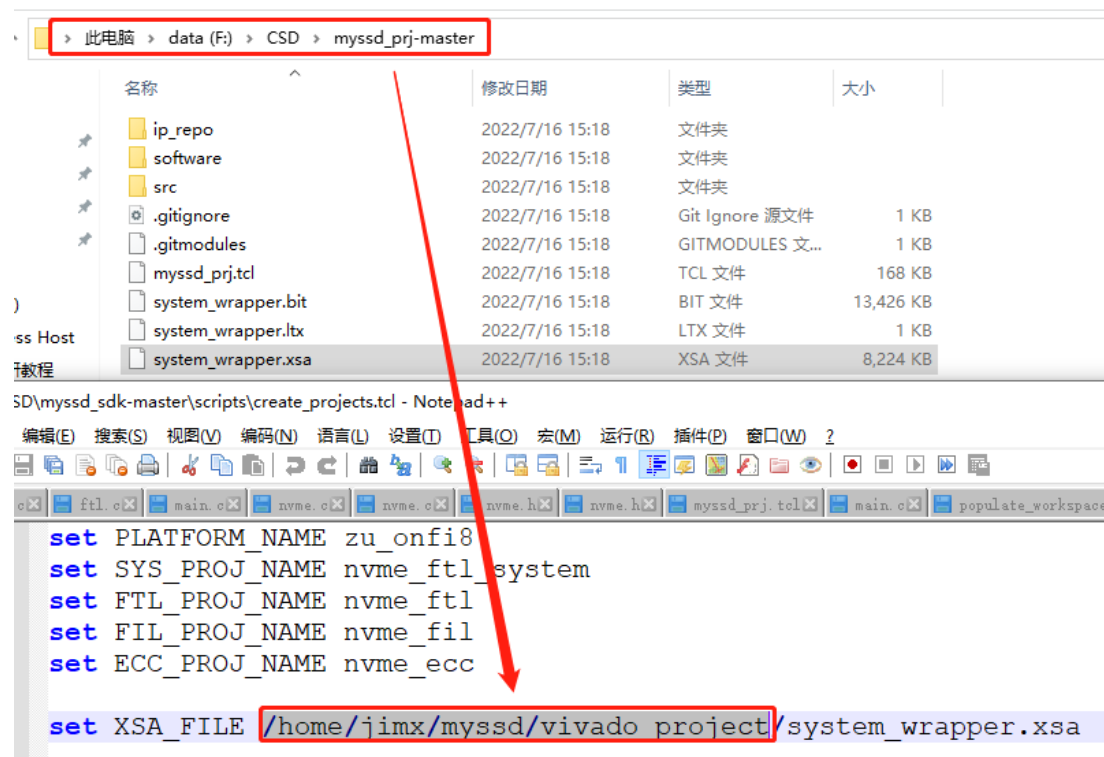


②cd进项目中的scripts文件夹中, 找到create_projects.tcl



3. 修改create_projects.tcl

将XSA_FILE的路径改成自己电脑vivado工程下的路径



3. 生成项目

①在XSCT Console里运行source create_projects.tcl
等待项目的生成

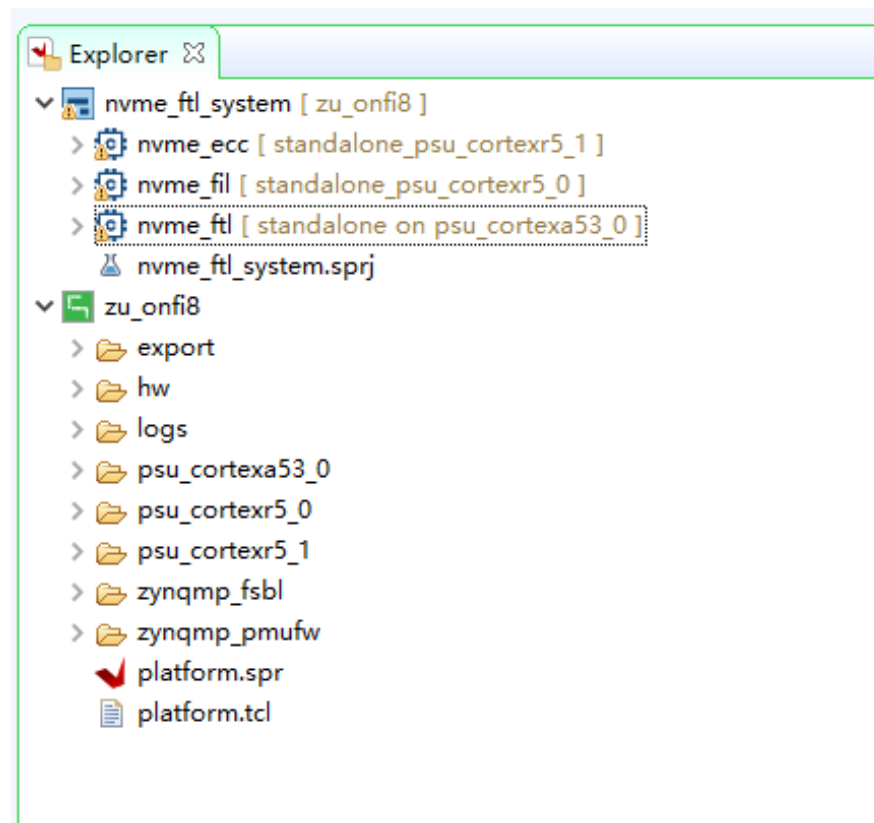
```
psu_cortexr5_1/lib/xaxidma_bdring.o psu_cortexr5_1/lib/mpu
_stats.o psu_cortexr5_1/lib/xaxidma_bdring.o psu_cortexr5_1/lib/mpu

armr5-none-eabi-ar: creating psu_cortexr5_1/lib/libxil.a

'Finished building libraries'

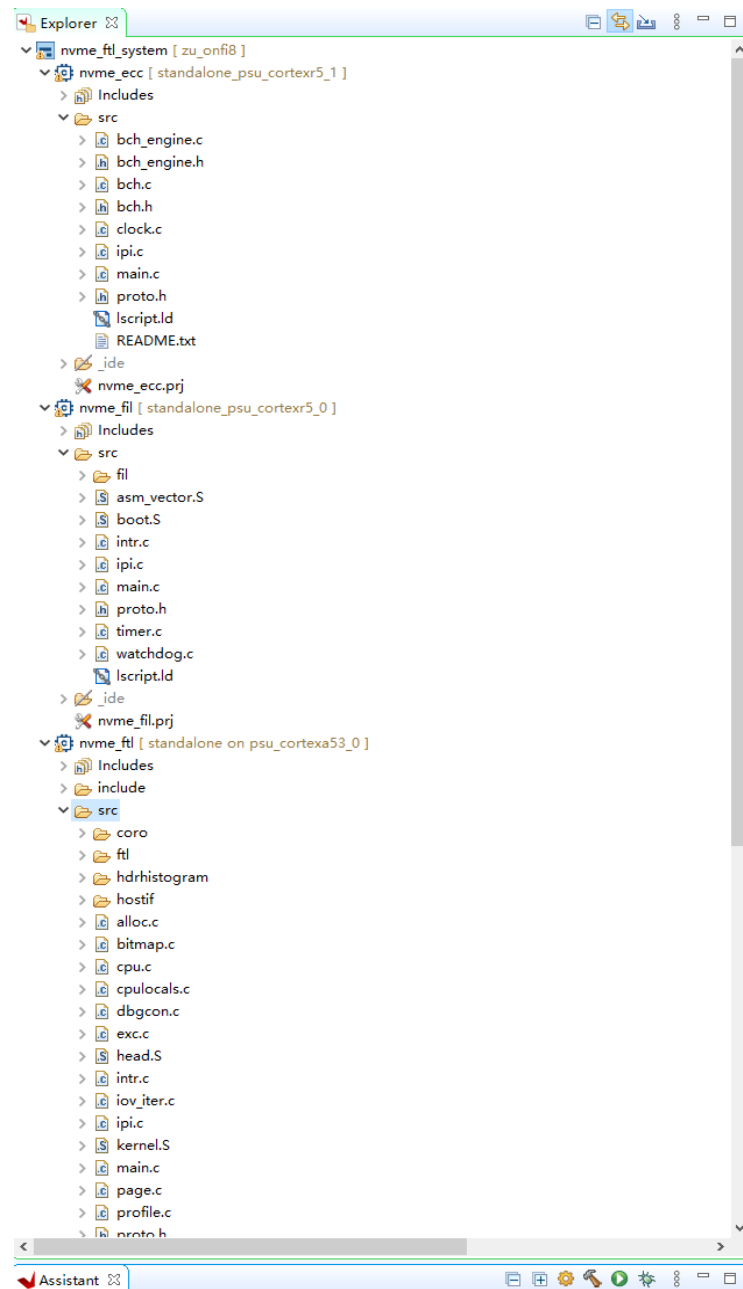
No system project exists, creating new system project.
xsct%
<
```

②生成成功则可在左侧看到三个项目，
如下图



4. 添加源文件

把myssd_sdk-storpu里面的r5poll覆盖到fil/src里，eccengine里面全部覆盖到ecc/src里，然后将include和src覆盖到ftl/里。总共会替换三个Iscrip.ld文件。结果如右图：



5.修改ftl/include/config.h

```
1 #ifndef _CONFIG_H
2 #define _CONFIG_H
3
4 /* Number of APUs */
5 #define NR_CPUS 2
6 #define FTL_CPU_ID (NR_CPUS - 1)
7 #define STORPU_CPU_ID 0
8
9 /* Uncomment this line to format EMMC on the first boot. */
10 /* #define FORMAT_EMMC */
11
12 /* Uncomment this line to wipe the entire SSD. */
13 /* #define WIPE_SSD */
14
15 /* Uncomment this line to perform a full bad block scan. */
16 /* #define FULL_BAD_BLOCK_SCAN */
17
18 /* Uncomment this line to reset FTL metadata. */
19 /* #define WIPE_MANIFEST */
20
21 /* Uncomment this line to wipe the mapping table. */
22 #define WIPE_MAPPING_TABLE
23
24 /* Uncomment this line to perform self-test on the NAND controller. */
25 /* #define NFC_SELFTEST */
26
27 /* Whether to enable volatile write cache. */
28 #define USE_WRITE_CACHE 1
29
30 /* Use histograms to track request statistics. */
31 #define USE_HISTOGRAM 1
32
33 /* Use hardware ECC engine. */
34 #define USE_HARDWARE_ECC 0
35
36 /* Defer DMA initialization until PCIe link is up. */
37 #define LAZY_DMA_INIT 1
38
39 /* System clock frequency. */
40 #define SYSTEM_HZ 25
41
42 #define CONFIG_NVME_IO_QUEUE_MAX 16
43
44 #define NR_WORKER_THREADS 16
45 #define NR_FLUSHERS 8
46 #define NR_FTL_THREADS (NR_WORKER_THREADS + NR_FLUSHERS)
47
48 #define CONFIG_STORAGE_CAPACITY_BYTES (512ULL << 30) /* 512 GiB */
49
50 #define SECTOR_SHIFT 12
51 #define SECTOR_SIZE (1UL << SECTOR_SHIFT)
52
53 #define MAX_DATA_TRANSFER_SIZE 8 /* 256 pages */
54
55 #define CONFIG_DATA_CACHE_CAPACITY (512UL << 20) /* 512MB */
56
57 #define CONFIG_MAPPING_TABLE_CAPACITY (1UL << 30) /* 1GB */
58
59 #define DEFAULT_PLANE_ALLOCATE_SCHEME PAS_CNDP
60
61 #define NAMESPACE_MAX 32
62
63 #define FILE_MAX 1
64
65 #endif
66
```

①将NR_CPUS 从2改成1

②打开第10、13、16、19、22、25行的注释

③将LAZY_DMA_INIT 1改成0

③保持 LAZY_DMA_INIT 为1

改完结果如右图

```
1 #ifndef _CONFIG_H
2 #define _CONFIG_H
3
4 /* Number of APUs */
5 #define NR_CPUS 1
6 #define FTL_CPU_ID (NR_CPUS - 1)
7 #define STORPU_CPU_ID 0
8
9 /* Uncomment this line to format EMMC on the first boot. */
10 #define FORMAT_EMMC
11
12 /* Uncomment this line to wipe the entire SSD. */
13 #define WIPE_SSD
14
15 /* Uncomment this line to perform a full bad block scan. */
16 #define FULL_BAD_BLOCK_SCAN
17
18 /* Uncomment this line to reset FTL metadata. */
19 #define WIPE_MANIFEST
20
21 /* Uncomment this line to wipe the mapping table. */
22 #define WIPE_MAPPING_TABLE
23
24 /* Uncomment this line to perform self-test on the NAND controller. */
25 #define NFC_SELFTEST
26
27 /* Whether to enable volatile write cache. */
28 #define USE_WRITE_CACHE 1
29
30 /* Use histograms to track request statistics. */
31 #define USE_HISTOGRAM 1
32
33 /* Use hardware ECC engine. */
34 #define USE_HARDWARE_ECC 0
35
36 /* Defer DMA initialization until PCIe link is up. */
37 #define LAZY_DMA_INIT 1
38
39 /* System clock frequency. */
40 #define SYSTEM_HZ 25
41
42 #define CONFIG_NVME_IO_QUEUE_MAX 16
43
44 #define NR_WORKER_THREADS 16
45 #define NR_FLUSHERS 8
46 #define NR_FTL_THREADS (NR_WORKER_THREADS + NR_FLUSHERS)
47
48 #define CONFIG_STORAGE_CAPACITY_BYTES (512ULL << 30) /* 512 GiB */
49
50 #define SECTOR_SHIFT 12
51 #define SECTOR_SIZE (1UL << SECTOR_SHIFT)
52
53 #define MAX_DATA_TRANSFER_SIZE 8 /* 256 pages */
54
55 #define CONFIG_DATA_CACHE_CAPACITY (512UL << 20) /* 512MB */
56
57 #define CONFIG_MAPPING_TABLE_CAPACITY (1UL << 30) /* 1GB */
58
59 #define DEFAULT_PLANE_ALLOCATE_SCHEME PAS_CNDP
60
61 #define NAMESPACE_MAX 32
62
63 #define FILE_MAX 1
64
65 #endif
66
```

6.修改ftl/include/flash_config.h

```
nvme_ftl nvme_fil nvme_ecc config.h flash_config.h ec
1 #ifndef _FIL_FLASH_CONFIG_H_
2 #define _FIL_FLASH_CONFIG_H_
3
4 #include <config.h>
5
6 #define FLASH_PG_SHIFT 14U
7 #define FLASH_PG_SIZE (1U << FLASH_PG_SHIFT)
8 #define FLASH_PG_OOB_SIZE (1872)
9 /* Full page size (user data + spare), rounded up to memory page size. */
10 #define FLASH_PG_BUFFER_SIZE \
11     (roundup(FLASH_PG_SIZE + FLASH_PG_OOB_SIZE, 0x1000))
12
13 #define SECTORS_PER_FLASH_PG (FLASH_PG_SIZE >> SECTOR_SHIFT)
14
15 #ifdef __UM__
16
17 #define NR_CHANNELS 1
18 #define CHIPS_PER_CHANNEL 1
19 #define DIES_PER_CHIP 2
20 #define PLANES_PER_DIE 2
21 #define BLOCKS_PER_PLANE 1048
22 #define PAGES_PER_BLOCK 512
23
24 #else
25
26 #define NR_CHANNELS 8
27 #define CHIPS_PER_CHANNEL 4
28 #define DIES_PER_CHIP 2
29 #define PLANES_PER_DIE 2
30 #define BLOCKS_PER_PLANE 1048
31 #define PAGES_PER_BLOCK 512
32
33 #endif
34
35 #define CHIPS_ENABLED_PER_CHANNEL CHIPS_PER_CHANNEL
36
37 /* #define ENABLE_MULTIPLANE */
38
39 /* clang-format off */
40
41 #if CHIPS_PER_CHANNEL == 4
42
43 #define CE_PINS \
44     36, 20, 30, 27, \
45     22, 28, 37, 43, \
46     63, 58, 31, 21, \
47     57, 65, 38, 54, \
48     67, 64, 72, 61, \
49     60, 66, 76, 73, \
50     74, 71, 56, 59, \
51     70, 69, 55, 75,
52
53 #elif CHIPS_PER_CHANNEL == 2
```

根据板上实际的flash片的数量来
对CHIPS_PER_CHANNEL进行修改

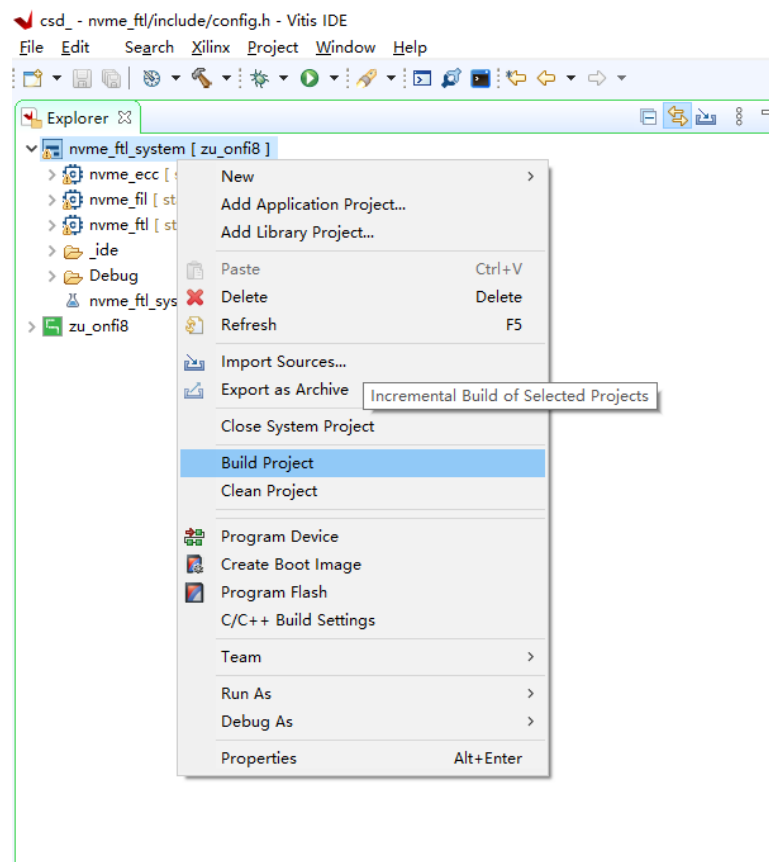
如果板子上有四个flash片，就改为2；
如果有八个flash片就改成4

改完结果如右图

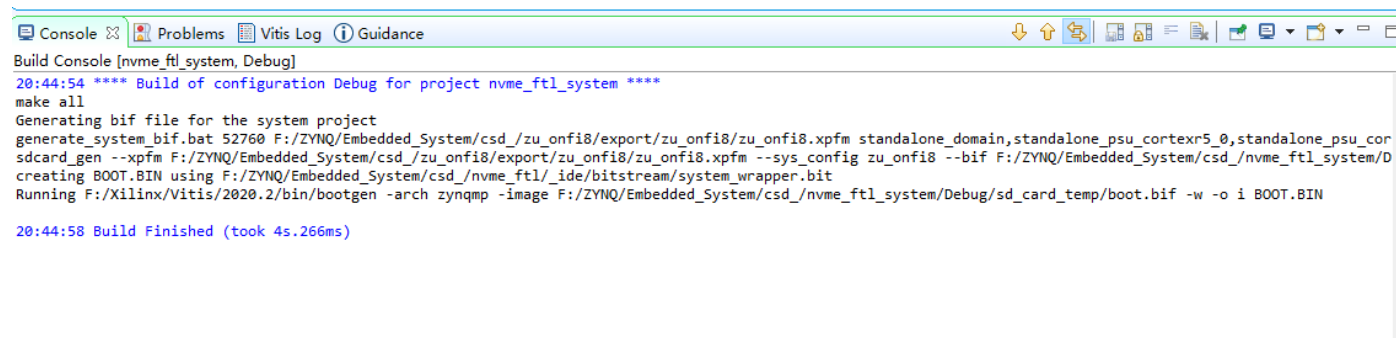
```
nvme_ftl nvme_fil nvme_ecc config.h flash_config.h ec
1 #ifndef _FIL_FLASH_CONFIG_H_
2 #define _FIL_FLASH_CONFIG_H_
3
4 #include <config.h>
5
6 #define FLASH_PG_SHIFT 14U
7 #define FLASH_PG_SIZE (1U << FLASH_PG_SHIFT)
8 #define FLASH_PG_OOB_SIZE (1872)
9 /* Full page size (user data + spare), rounded up to memory page size. */
10 #define FLASH_PG_BUFFER_SIZE \
11     (roundup(FLASH_PG_SIZE + FLASH_PG_OOB_SIZE, 0x1000))
12
13 #define SECTORS_PER_FLASH_PG (FLASH_PG_SIZE >> SECTOR_SHIFT)
14
15 #ifdef __UM__
16
17 #define NR_CHANNELS 1
18 #define CHIPS_PER_CHANNEL 1
19 #define DIES_PER_CHIP 2
20 #define PLANES_PER_DIE 2
21 #define BLOCKS_PER_PLANE 1048
22 #define PAGES_PER_BLOCK 512
23
24 #else
25
26 #define NR_CHANNELS 8
27 #define CHIPS_PER_CHANNEL 2
28 #define DIES_PER_CHIP 2
29 #define PLANES_PER_DIE 2
30 #define BLOCKS_PER_PLANE 1048
31 #define PAGES_PER_BLOCK 512
32
33 #endif
34
35 #define CHIPS_ENABLED_PER_CHANNEL CHIPS_PER_CHANNEL
36
37 /* #define ENABLE_MULTIPLANE */
38
39 /* clang-format off */
40
41 #if CHIPS_PER_CHANNEL == 4
42
43 #define CE_PINS \
44     36, 20, 30, 27, \
45     22, 28, 37, 43, \
46     63, 58, 31, 21, \
47     57, 65, 38, 54, \
48     67, 64, 72, 61, \
49     60, 66, 76, 73, \
50     74, 71, 56, 59, \
51     70, 69, 55, 75,
52
53 #elif CHIPS_PER_CHANNEL == 2
```

7.编译构建工程

①右键nvme_ftl_system， 点击Build Project



②构建成功如下图



8.安装串口驱动

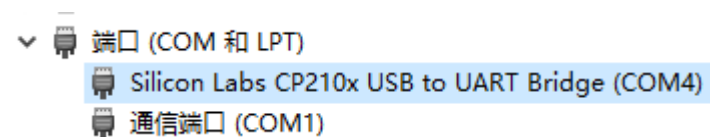
①将板子的串口连接到PC上，同时接好下载器，并接上电源（注意只能往电源那方向拨动电源开关），如下图



②打开设备管理器，如果CP2104驱动出现感叹号，说明没装好驱动，需要手动安装，如下图

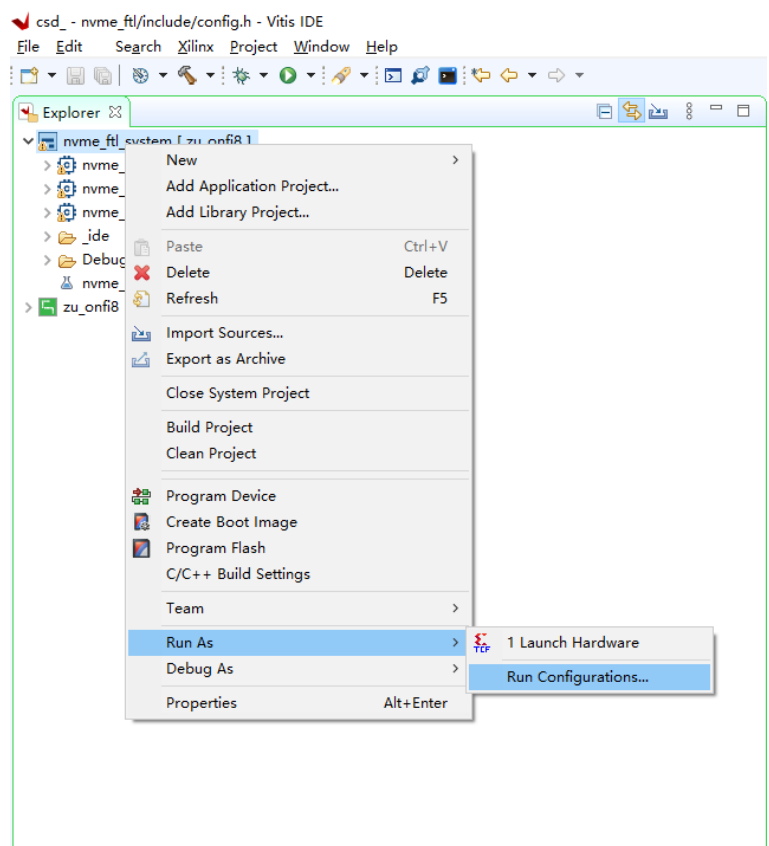


③成功安装完驱动后，可在端口(COM和LPT)里找到，如下图

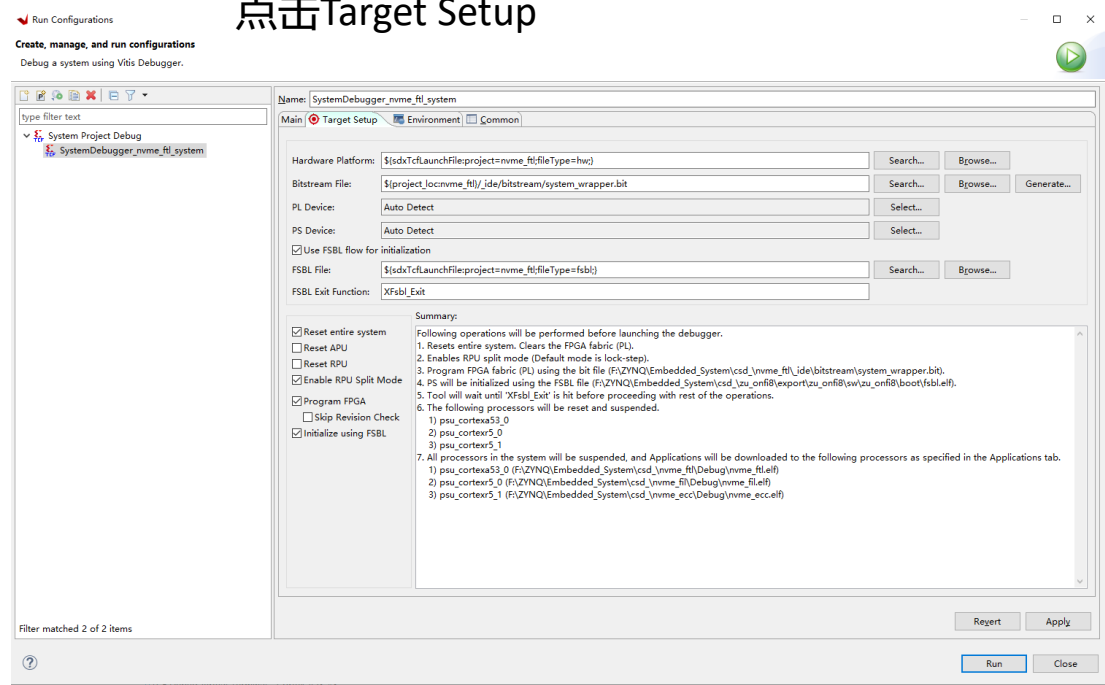


9.配置Run Configuration

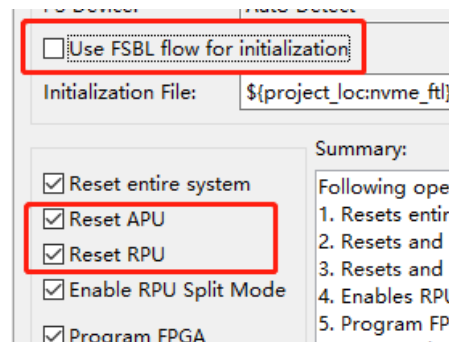
①右键工程，点击Run Configuarions



②点击new launch configuration, 在
点击Target Setup

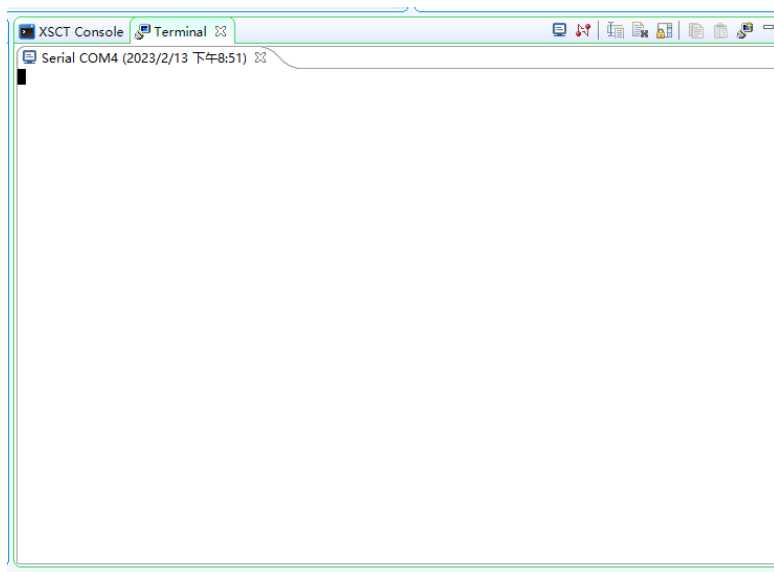


③将Use FSBL flow for initialization 取消勾选
将Reset APU 勾选
将Reset RPU 勾选，并点击Apply

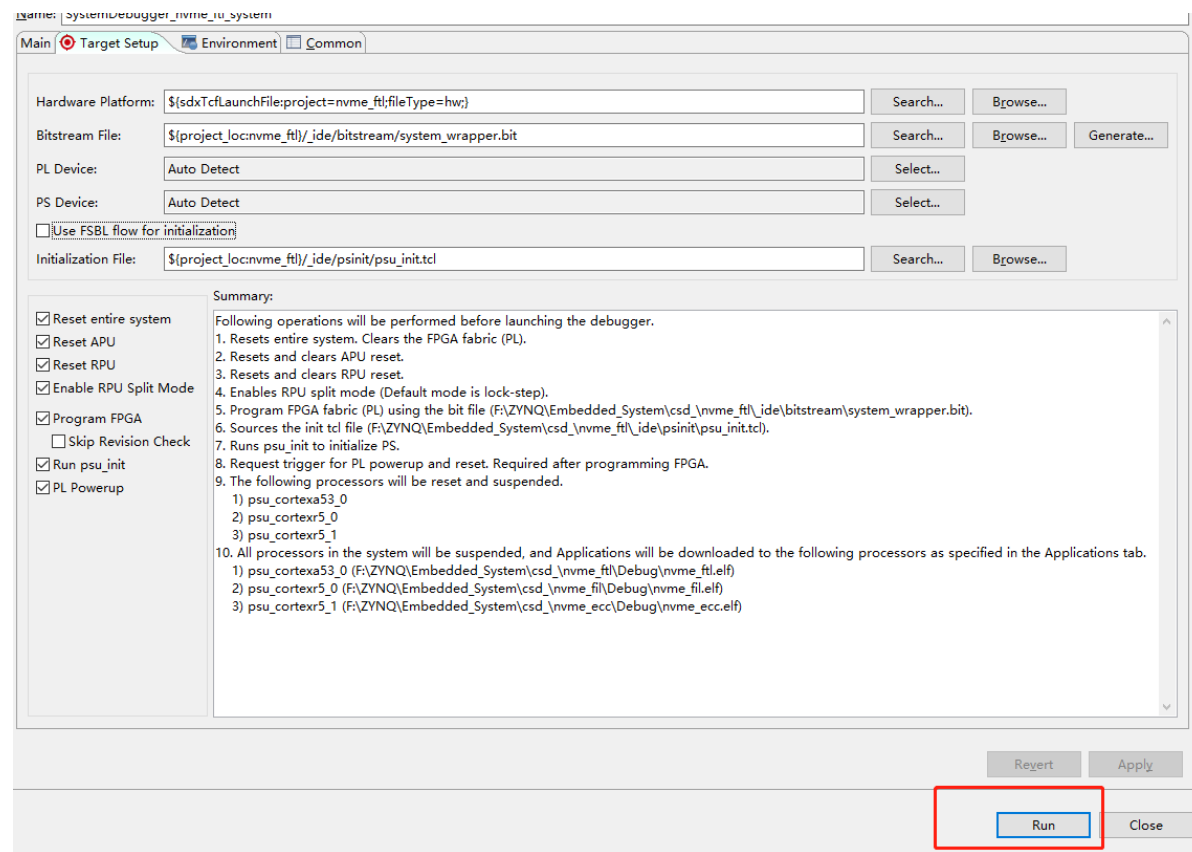


10.打开串口并烧录程序

①打开vitis自带串口，设置保持115200/8/0/1



②回到上一步的界面，点击Run，开始烧录程序
烧录完成后，如果板子正常，则串口会有数据输出



11.进行坏块检测

程序烧录成功后会自动进行坏块检测，等待一段时间，出现下图则表示检测完成

```
!!!! Error t2 s0 ch7 w1 d1 pl1 b1029 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1030 p
0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1031 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1032 p
0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1033 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1034 p
0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1035 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1036 p
0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1037 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1038 p
0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1046 p0
!!!! Error t2 s0 ch7 w1 d1 pl1 b1047 p
0
OK
[AMU] Initializing new global translation directory for namespace 1 ...OK
] Initialized namespace 1 with 33554432 logical pages
Started thread 1 0x78a38
Started thread 2 0x78ac0
Started thread 3 0x78b48
Started thread 4 0x78bd0
Started thread 5 0x78c58
Started thread 6 0x78ce0
Started thread 7 0x78d68
Started thread 8 0x78df0
Started thread 9 0x78e78
Started thread 10 0x78f00
Started thread 11 0x78f88
Started thread 12 0x79010
Started thread 13 0x79098
Started thread 14 0x79120
Started thread 15 0x791a8
Started thread 0 0x789b0
```

12.查看坏块检测结果

①将config.h中这四行注释，重新编译和下载程序

```
1  #ifndef _CONFIG_H_
2  #define _CONFIG_H_
3
4  /* Number of APUs */
5  #define NR_CPUS 1
6  #define FTL_CPU_ID (NR_CPUS - 1)
7  #define STORPU_CPU_ID 0
8
9  /* Uncomment this line to format EMMC on the first boot. */
10 // #define FORMAT_EMMC
11
12 /* Uncomment this line to wipe the entire SSD. */
13 // #define WIPE_SSD
14
15 /* Uncomment this line to perform a full bad block scan. */
16 // #define FULL_BAD_BLOCK_SCAN
17
18 /* Uncomment this line to reset FTL metadata. */
19 // #define WIPE_MANIFEST
20
21 /* Uncomment this line to wipe the mapping table. */
22 #define WIPE_MAPPING_TABLE
23
24 /* Uncomment this line to perform self-test on the NAND controller. */
25 #define NFC_SELFTEST
26
27 /* Whether to enable volatile write cache. */
28 #define USE_WRITE_CACHE 1
29
30 /* Use histograms to track request statistics. */
31 #define USE_HISTOGRAM 1
32
33 /* ... */
```

②检查串口打印结果，如果8个channel都是0或者1 error，则说明selftest通过

```
[FTL] Restoring manifest (800 bytes) ...OK[BM] Restoring planes (12288 bytes) ...OK[BM] Restoring bad blocks (12288 bytes) ...OK[AMU] Initializing new global translation directory for namespace 1 ...OK[AMU] Initialized namespace 1 with 33554432 logical pagesStarted thread 0 0x789b0
Started thread 1 0x78a38
Started thread 2 0x78ac0
Started thread 3 0x78b48
Started thread 4 0x78bd0
Started thread 5 0x78c58
Started thread 6 0x78ce0
Started thread 7 0x78d68
Started thread 8 0x78df0
Started thread 9 0x78e78
Started thread 10 0x78f00
Started thread 11 0x78f88
Started thread 12 0x79010
Started thread 13 0x79098
Started thread 14 0x79120
Started thread 15 0x791a8

— Entering nvme main() plddr —
Channel 0: 0 err (0 0)1 err (0 0)2 err (0 0)3 err (0 0)4 err (0 0)5 err (0 0)6 err (0 0)7 err (0 0)8 err (0 0)9 err (0 0)
Channel 1: 0 err (1 0)1 err (1 0)2 err (1 0)3 err (1 0)4 err (1 0)5 err (1 0)6 err (1 0)7 err (1 0)8 err (1 0)9 err (1 0)
Channel 2: 0 err (0 0)1 err (0 0)2 err (0 0)3 err (0 0)4 err (0 0)5 err (0 0)6 err (0 0)7 err (0 0)8 err (0 0)9 err (0 0)
Channel 3: 0 err (1 0)1 err (0 0)2 err (1 0)3 err (0 0)4 err (0 0)5 err (1 0)6 err (0 0)7 err (1 0)8 err (0 0)9 err (1 0)
Channel 4: 0 err (0 0)1 err (0 0)2 err (1 0)3 err (1 0)4 err (1 0)5 err (1 0)6 err (1 0)7 err (1 0)8 err (1 0)9 err (1 0)
Channel 5: 0 err (0 0)1 err (0 0)2 err (0 0)3 err (0 0)4 err (0 0)5 err (0 0)6 err (1 0)7 err (0 0)8 err (1 0)9 err (0 0)
Channel 6: 0 err (0 0)1 err (0 0)2 err (0 0)3 err (0 0)4 err (0 0)5 err (0 0)6 err (0 0)7 err (0 0)8 err (0 0)9 err (0 0)
Channel 7: 0 err (1 0)1 err (1 0)2 err (1 0)3 err (1 0)4 err (1 0)5 err (1 0)6 err (1 0)7 err (1 0)8 err (1 0)9 err (1 0)
```

13.进入BIOS

之后进入bios，可以看到一块约550GB的盘，无名称

14.进入Ubuntu

之后进入系统，在disk工具里可以看到512GiB的SSD，可以对其进行benchmark，格式化文件系统，读写文件等