场景功耗调试指导手册

WWW.UNISOC.COM

紫 光 展 锐 科 技

修改历史

版本号	日期	注释
V1.0	2019/5/24	初稿
V1.1	2020/8/11	1. 文档名从《Unisoc_Scenario_Power_Debug_Handbook》修改为《Scenario Power Debug指导手册》 2. 更新样式，优化结构，完善内容。 3. 适用Android 9.0、Android10.0、Android 11.0。
V1.2	2020/9/9	1. 文档从《Scenario Power Debug指导手册》修改为《场景功耗调试指导手册》。 2. 更新样式。

关键字

关键字： 功耗、调试、指导手册。

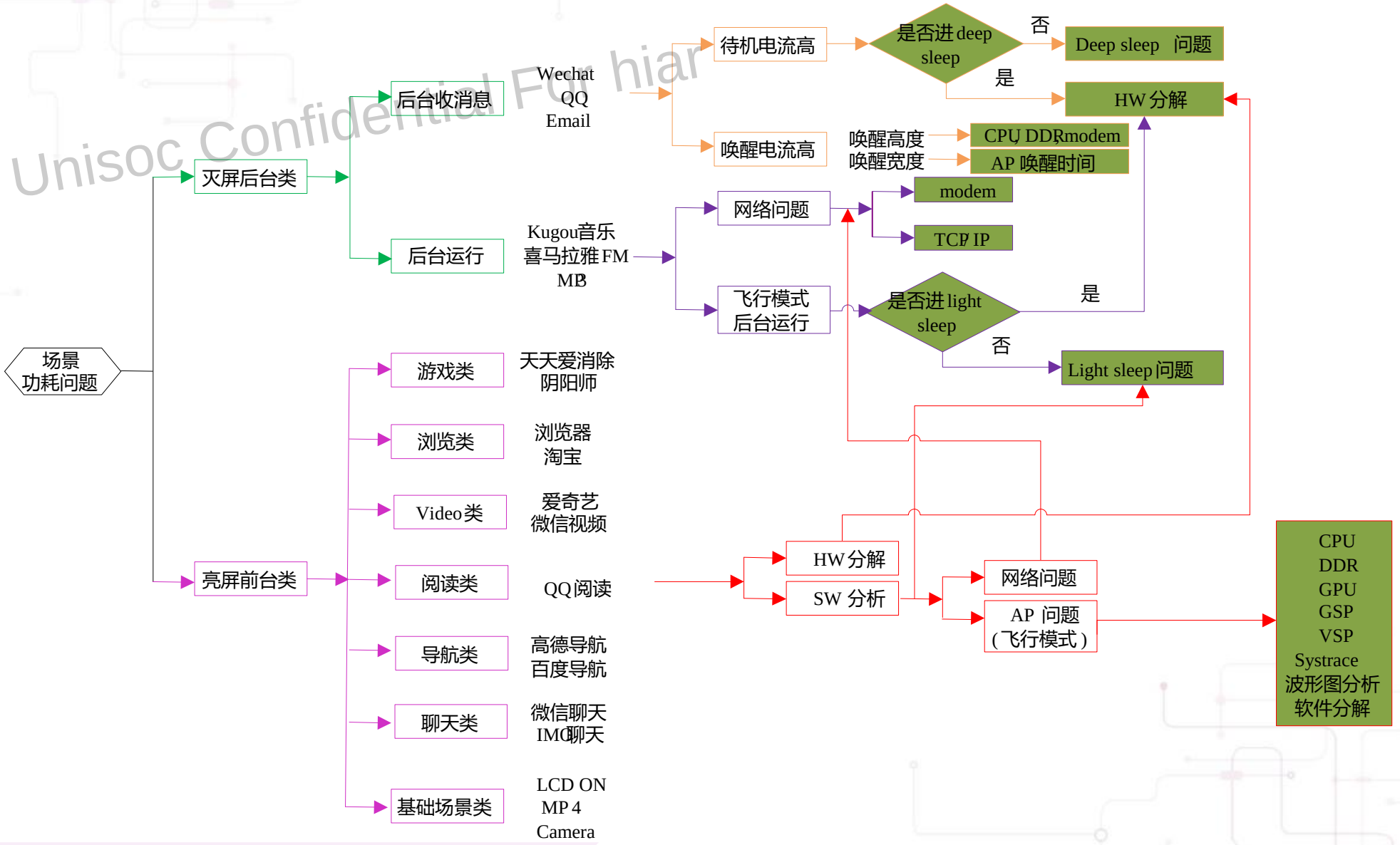
目录

01

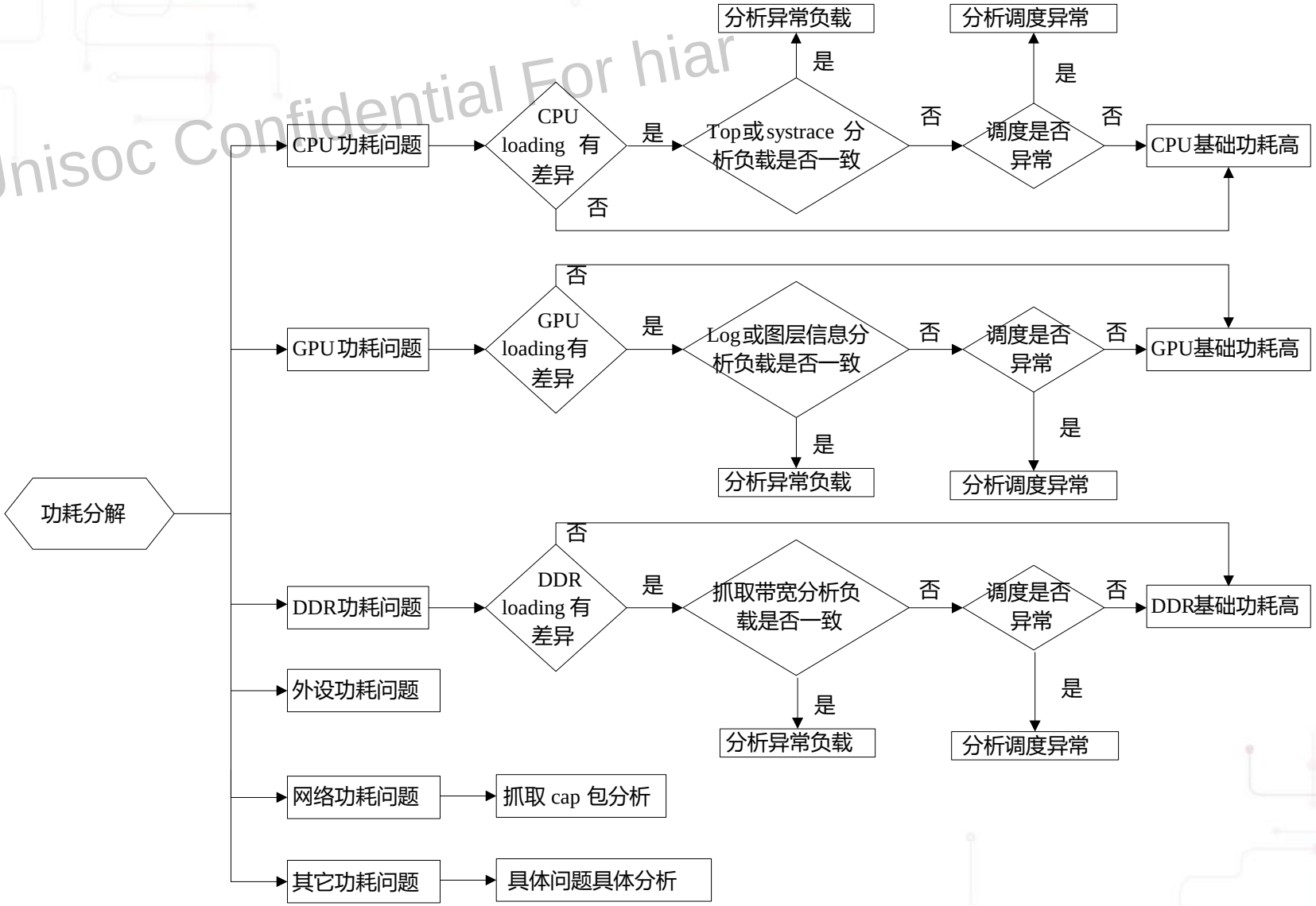
场景功耗分析思路

场景功耗分析思路—功耗优化场景

场景功耗分析思路—场景功耗问题分类



场景功耗分析思路—场景功耗分解

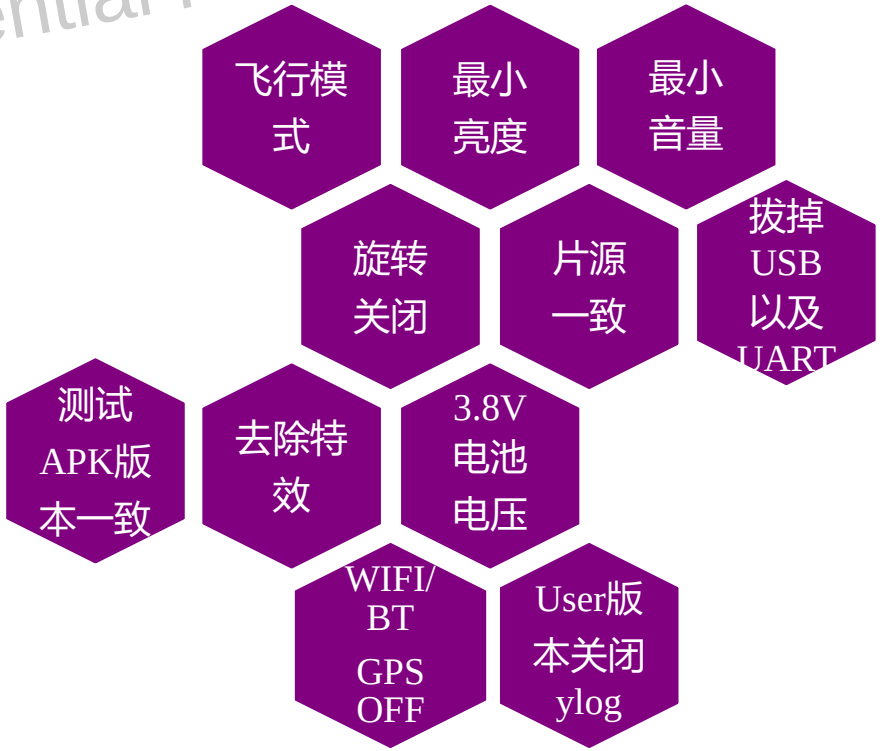


02

场景功耗常用分析方法

场景功耗常用分析方法—测试前置条件

统一测试前置条件



场景功耗常用分析方法—CPU状态分析方法

CPU状态的四种方式介绍
(检查CPU的核数以及CPU的频率是否合理)

1	top 脚本检查每个线程的CPU loading
2	通过执行脚本查看CPU在每个频点下的loading以及CPU的各个状态
3	通过固定CPU的核数和频率，排除CPU上的功耗差异
4	通过抓取systrace/ftrace或DS5来查看每个核上任务的运行情况

场景功耗常用分析方法—GPU分析方法

GPU 分析方法介绍
(频率，核数)

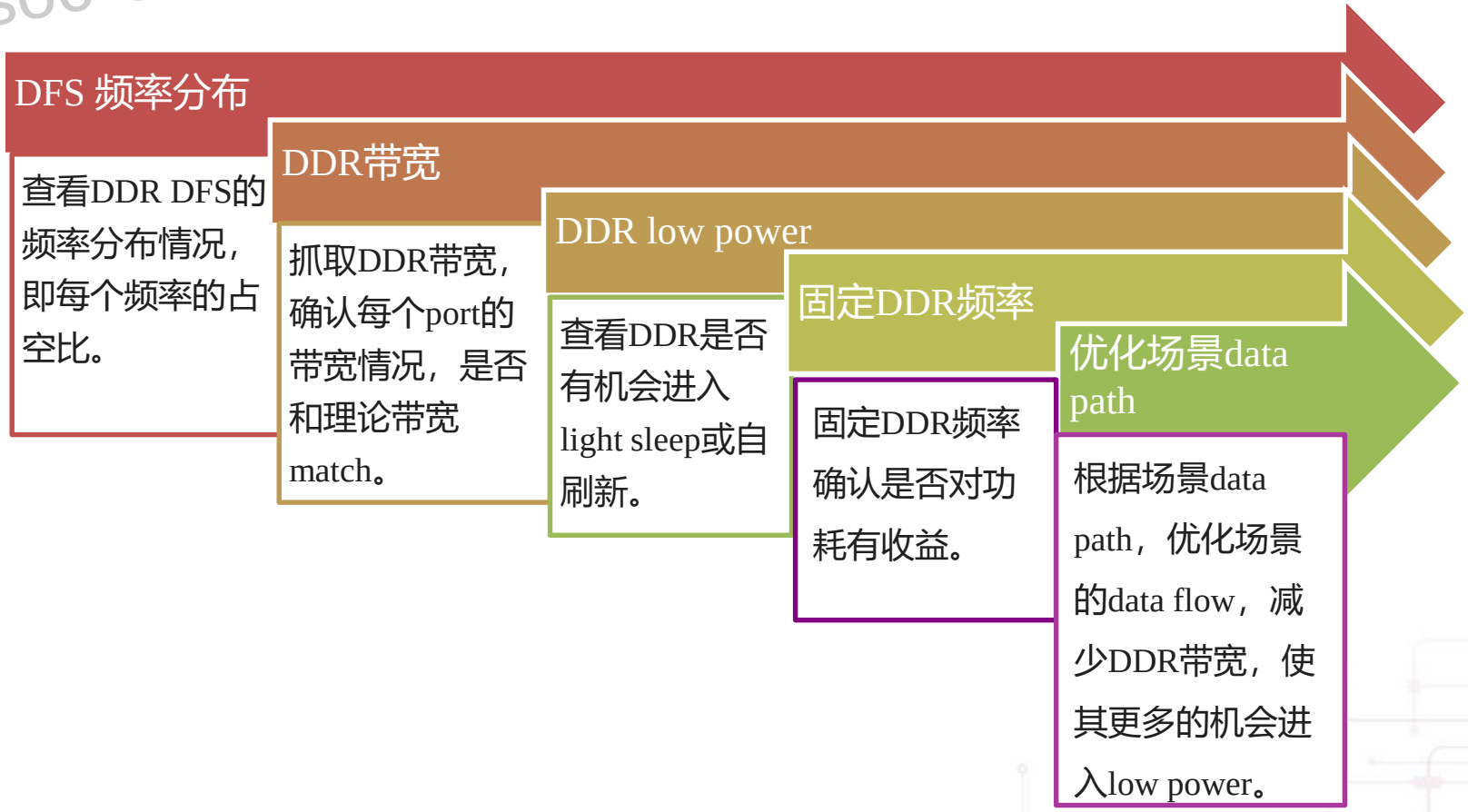
通过脚本统计GPU的loading

通过脚本限制GPU的频率，排除GPU的功耗影响。

Check GPU支持数据的压缩率以及支持的数据压缩格式

场景功耗常用分析方法—DDR状态分析方法

DDR状态查看方式介绍 (频率，核数)



场景功耗常用分析方法—图层信息分析方法

图层信息分析方法介绍(dumpsys SurfaceFlinger)

HWC代表是GSP/DPU合成，GLES代表是GPU合成

Device代表是GSP/DPU合成，Client代表是GPU合成

Android 7.0

```
Hardware Composer state (version 01040000):
 mDebugForceFakeVSync=0
 Display[0] configurations (* current):
   * 0: 1080x1920, xdpi=403.411011, ydpi=403.040985, refresh=16666000, colorTransform=-22
 numHwLayers=4, flags=00000000
    type    |  handle  | hint | flag | tr | blnd |    format    |      source crop (l,t,r,b)      |          frame          | name
------------+----------+------+------+----+------+--------------+---------------------------------+-------------------------+-----
-
        HWC | 726d2318d200 | 0000 | 0000 | 00 | 0100 | RGBx_8888    |     0.0,     0.0, 1080.0, 1920.0 |     0,     0, 1080, 1920 |
SurfaceView - com.android.gallery3d/com.android.gallery3d.app.GalleryActivity
        HWC | 726d24e26980 | 0002 | 0000 | 00 | 0105 | RGBA_8888    |     0.0,     0.0, 1080.0,    72.0 |     0,     0, 1080,    72 |
com.android.gallery3d/com.android.gallery3d.app.GalleryActivity
        HWC | 726d23178c80 | 0002 | 0000 | 00 | 0105 | RGBA_8888    |     0.0,     0.0, 1080.0,    72.0 |     0,     0, 1080,    72 |
StatusBar
 FB TARGET | 726d2555ef80 | 0000 | 0000 | 00 | 0105 | RGBA_8888    |     0.0,     0.0, 1080.0, 1920.0 |     0,     0, 1080, 1920 |
```

Android 8.0以后

Unisoc定制

基于Android 8.0以后Unisoc定制

```
Display 0 HWC layers:
------------------------------------------------------------------------------------------
 Layer name
           Z |  Comp Type |  Disp Frame (LTRB) |          Source Crop (LTRB)
------------------------------------------------------------------------------------------
 SurfaceView - com.tencent.tmgp.sgame/com.tencent.tmgp.sgame.SGameActivity#0
  4294967294 |      Device |    0    0  720 1440 |     0.0     0.0  640.0 1280.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 RoundCorner#0
      311000 |      Device |  690    0  720 1440 |     0.0     0.0   30.0 1440.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 RoundCorner#1
      311005 |      Device |    0    0   30 1440 |     0.0     0.0   30.0 1440.0
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
h/w composer state:
 h/w composer enabled
-------------------------------------------------------------------------------------------
comp type |  format | fbc | pitch | height | transform |  blend | alpha | zorder |  dl  dt  dr  db |
-------------------------------------------------------------------------------------------
    GSP | YCbCr420 |  N | 1280 |  720 |     90 |  NONE |  255 |    0 |
-------------------------------------------------------------------------------------------
    DPU | RGBA8888 |  N |  720 | 1280 |      0 | PREMULT |  255 |    1 |  0  0  0  0 |
-------------------------------------------------------------------------------------------
    DPU | RGBA8888 |  N |   96 |   96 |      0 | PREMULT |  255 |    2 |  0  0  0  0 |
```

上面的表格中，comp type这一列就是合成类型。可以看到每一层是谁处理的。

场景功耗常用分析方法—其他IP分析方法

其它IP分析方法介绍(以VSP为例)

- ✓ VSP是否正常工作可以根据 sprd.h264.decoder的CPU占用率来确定。如下图是使用top命令查看的结果，thread sprd.h264.decoder（top里会显示为rd.h264.decoder） CPU占用率在2%代表目前是VSP硬解。

```
shell@scx35l64_sp9838aea_5mod:/ # top -t -m 10 | grep h264
 3285  3655  1   2% S 204592K  20768K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% S 204592K  20384K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% R 204592K  20996K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% S 204592K  21136K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% R 204592K  21120K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% R 204592K  20468K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% S 204592K  20376K  fg media    rd.h264.decoder /system/bin/mediaserver
 3285  3655  0   2% S 204592K  20652K  fg media    rd.h264.decoder /system/bin/mediaserver
```

- ✓ 软解的话使用MXPlayer_2015.01.30.apk设置为软解模式播放1080P MP4，通过top无法看到rd.h264.decoder。
- ✓ 也可以通过如下命令，查看中断的方式看VSP是否工作。

  cat /proc/interrupts
- ✓ 对于中断这种方式，可以查看是否有其它异常中断导致功耗偏高。

场景功耗常用分析方法—软硬件分解分析方法

软硬件分解方法介绍

硬件分解

- 测量各路power rail的实际电压电流值。
- 根据分解的电流值和对比机器进行对比分析，看电流高在哪个power rail上。
- 如果没有对比机的分解数据，直接从功耗最高的power rail开始优化，或者从异常的power rail进行优化。

正常播放	249mA	差值	结论
声音关闭，其它都打开	223mA	26mA	声音消耗：26mA
显示关闭，其它都打开	226mA	23mA	显示消耗：23mA
显示和视频频解码两个关闭，声音打开	186mA	63mA	解码消耗：40mA
声音、显示和解码全都关闭	162mA	87mA	视频播放消耗：87mA

软件分解

- 将场景分解成独立的模块，通过测量某个模块开关前后的功耗差值来确定该模块的功耗，从而找到优化的方向。
- 涉及到场景的流程架构，需要相应owner积极配合。
- 将场景先分解成简单场景进行分析，比如关闭网络，本地运行，静止场景，暂停场景，关闭弹幕，停留在输入界面等。
- 找波形周期和规律，先分析一个周期的电流是否正常。

场景功耗常用分析方法—其他状态分析方法

其它状态查看方式介绍

波形分析

- 通过分析波形检查是否有异常突起，包括DDR是否可以进light sleep等。
- 抓取各个power domain的波形分析，包括vbat波形。

HW方法

- 通过HW分解查看是否存在异常的power domain。
- 通过把外设拆除的方法检查是否有外设漏电。
- 排出单体问题，至少2台机器测试。

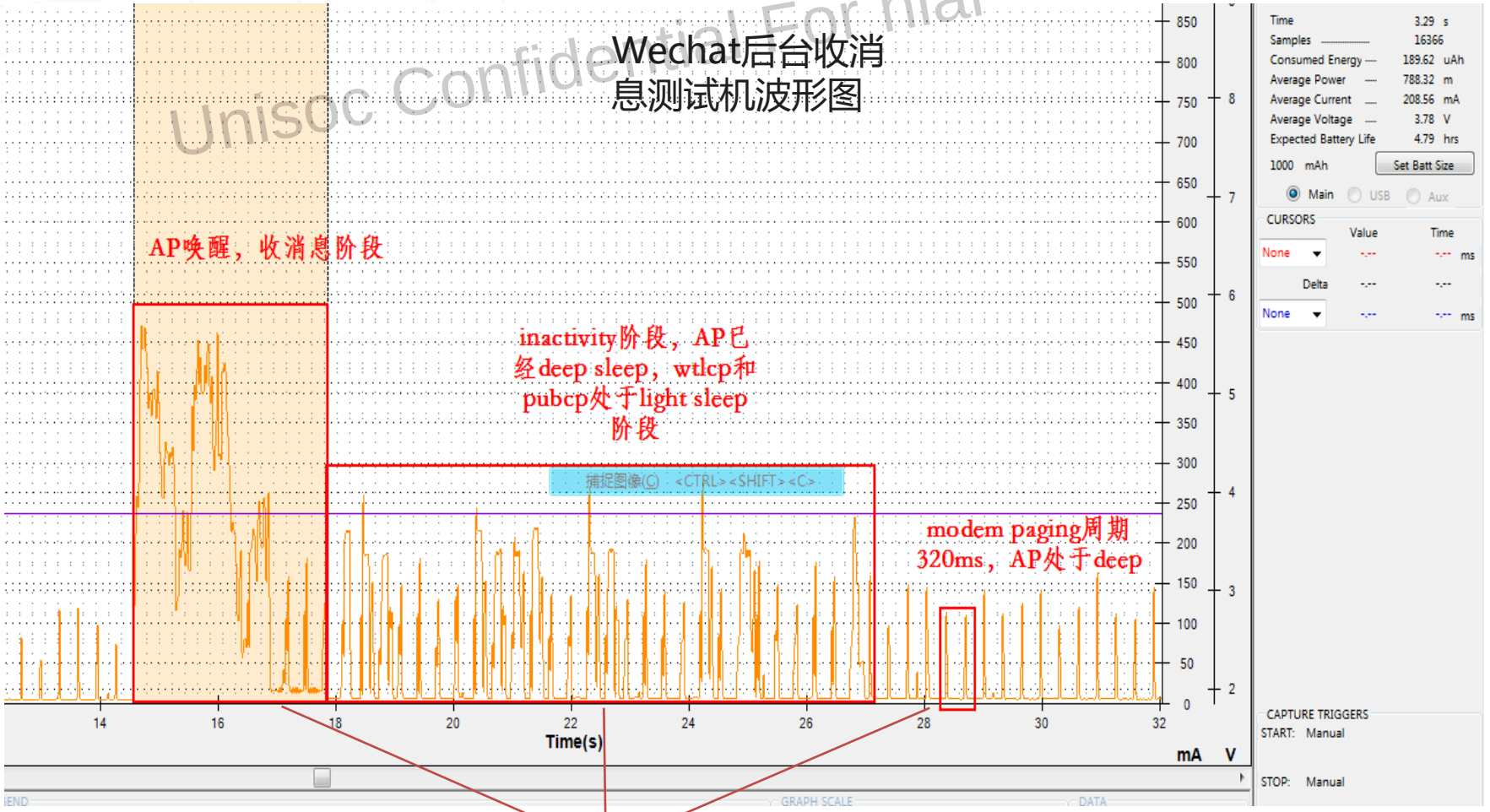
SW方法

- 软件把复杂场景分解成各个独立场景进行分析。

03

场景功耗实例分析

场景功耗实例分析—后台收消息类 1/2



收消息阶段	时间	此阶段的功耗
测试机	2.24s	36.4mA
参考机	2.12s	38.6mA

inactivity阶段	时间	此阶段的功耗
测试机	8.21s	59.7mA
参考机	8.37s	27.9mA

结论

测试机的DDR和CPU电压比参考机高，所以平均电流比参考机高。

分别对比分析这三个阶段的功耗，哪个阶段功耗高，就重点分析该阶段即可。

场景功耗实例分析—后台收消息类 2/2

- **后台收消息类—唤醒时间长问题**

从上层android log分析acquire和release wakelock的时间间隔：

搜索关键字：WakeLockInternal

Android log：

A111-22 17:59:27.201 3413 3805 D PowerManagerService: acquireWakeLockInternal: lock=35655253, flags=0x1, tag="AudioMix", ws=null, uid=1013, pid=0 packageName=media

A111-22 17:59:31.953 3413 6751 D PowerManagerService: releaseWakeLockInternal: lock=35655253 [AudioMix],

从上述时间点可以看到，audiomix 持锁时间达到4s，而wechat 收消息阶段总时间才3s不到，因此需要优化audiomix的持锁时间。

- **结论：声音的大小会影响audiomix的持锁时间，静音之后，发现持锁时间很短。**

场景功耗实例分析—BBAT模式功耗类

●灭屏后台类——BBAT模式功耗问题


```
User 0%, System 0%, IOW 1%, IRQ 0%
User 7 + Nice 0 + Sys 8 + Idle 999 + IOW 17 + IRQ 0 + SIRQ 0 = 1031
  PID   TID USER     PR  NI CPU% S     VSS     RSS PCY Thread          Proc
  512   512 root     20   0   1% R   4524K   2604K  fg top             top
  294   294 root     20   0   0% S      0K      0K  fg kworker/0:2
  190   190 root     20   0   0% S   2992K   1312K  fg ueventd         /sbin/ueventd
  214   621 logd     30  10   0% S  17860K   3112K  bg logd.reader.per /system/bin/logd
  214   622 logd     30  10   0% S  17860K   3112K  bg logd.reader.per /system/bin/logd
  233   613 system   20   0   0% S 137256K   2932K  fg ylog            /system/bin/ylog
    7     7 root     20   0   0% S      0K      0K  fg rcu_preempt
    8     8 root     20   0   0% S      0K      0K  fg rcu_sched
    9     9 root     20   0   0% S      0K      0K  fg rcu_bh
   10    10 root     RT   0   0% S      0K      0K  fg migration/0
```

正常功耗情况下的top命令结果

```
User 96%, System 2%, IOW 0%, IRQ 0%
User 2520 + Nice 1 + Sys 56 + Idle 21 + IOW 1 + IRQ 0 + SIRQ 0 = 2599
  PID   TID USER     PR  NI CPU% S     VSS     RSS PCY Thread          Proc
 3053  3321 root     20   0  8% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3053  3318 root     20   0  8% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3139  3139 root     20   0  7% R 240600K 132812K  fg main            /system/bin/dex2oat
 3053  3317 root     20   0  7% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3053  3316 root     20   0  7% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3053  3315 root     20   0  7% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3053  3053 root     20   0  6% R 245720K 120456K  fg main            /system/bin/dex2oat
 3053  3319 root     20   0  6% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3053  3320 root     20   0  6% R 245720K 120456K  fg Compiler driver /system/bin/dex2oat
 3139  3342 root     20   0  5% R 240600K 132812K  fg Compiler driver /system/bin/dex2oat
```

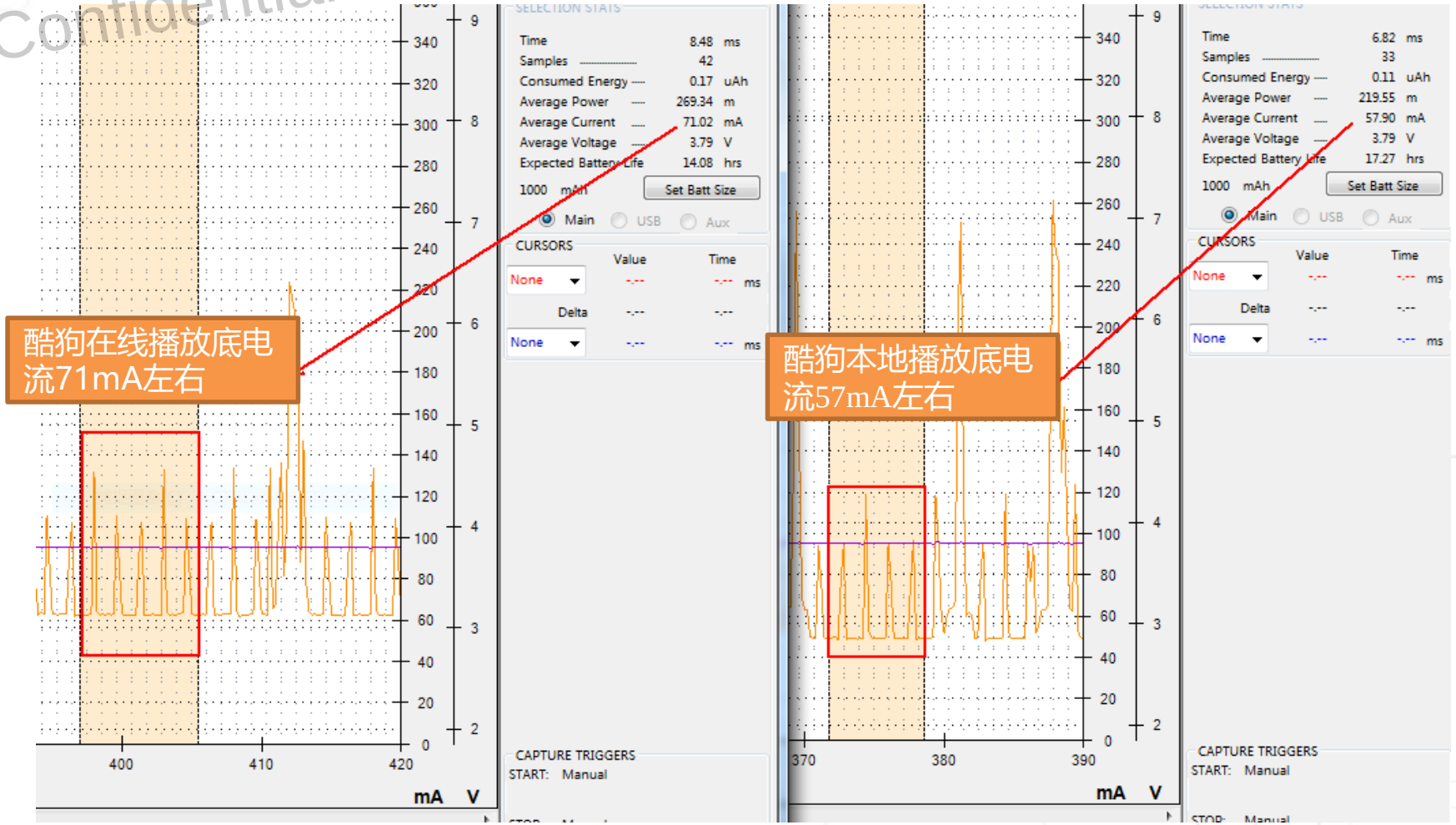
异常功耗情况下的top命令结果

从对比的top命令结果来看，功耗异常的top结果，dex2oat进程占用较多CPU loading，导致CPU功耗明显高出很多。

场景功耗实例分析—后台运行类 1/2

- 后台运行——底电流功耗高

根据右侧2个底电流的差异，通过做实验可以发现只要插sim卡，纯light sleep下底电流也会差14mA，因此直接在light sleep下调查这个问题即可。



场景功耗实例分析—后台运行类 2/2

- **后台运行——底电流功耗高**

  根据前面的对比分析，底电流高14mA左右，问题肯定和modem相关，因此做HW分解数据如下：

VDDRF0/DCDCMEM(transceiver VDD1V1DIG)功耗有增大，所以怀疑 AVDD1V8的增大部分可能与RF LVDS控制有关，可以直接对比两种状态的transceiver状态或者直接对比analog部分的模块开启状态看差异点。

SW Version	W17.24.5	W17.24.5
Test Case	Lightsleep 飞行模式	Lightsleep 4G实网
VBAT Total	37.7	49
DCDCCORE	14.8	15.71
DCDCGPU	4.9	5.1
DCDCMEM	2.1	5.4
DCDCGEN	8.4	15
AVDD1V8	/5.2	/9.2
VDDRF0	/0.2	/2.6

场景功耗实例分析—LCD ON

- **亮屏前台类——LCD ON**
  - ✓**问题描述**: 飞行模式下，LCD ON的功耗是160mA，过了1分钟后它的功耗降到150mA。
  - ✓**分析方法**：通过log和寄存器发现1分钟前后系统的状态没有变化，然后请硬件帮忙分解，发现主要问题是在VDD2V8；该domain给TP和sensor供电，排除sensor的影响发现是TP的原因，TP本身的中间件有个下电时间，设置成了60s，后面版本更新TP的下电时间为15s。

此问题也可通过拆除外设的方法定位问题

LCD_ON（mA@3.8V）	DDR400M bottom +CA driver 48ohm 1204 version
Total	160
VDDCORE	59
VDDARM	3
VDDMEM	17
VDD1V85	30
AVDD1V8	8.6
VDDRF0	1.6
LCD	15.7
VDDDCXO	5.5
VDD2V8	13.5
LCD Driver	19.3
LCD BL	7.7
LCD ON-LCD remove	42

LCD_ON（mA@3.8V）	DDR400M bottom +CA driver 48ohm 1204 version
Total	150
VDDCORE	59
VDDARM	3
VDDMEM	17
VDD1V85	30
AVDD1V8	8.6
VDDRF0	1.6
LCD	15.7
VDDDCXO	5.5
VDD2V8	3.5
LCD Driver	19.3
LCD BL	7.7
LCD ON-LCD remove	42

场景功耗实例分析—BT收发文件类 1/2

●CPU功耗高问题分析—BT收发文件

HW分解数据

BT TX 配对华为荣耀4A	测试机	参考机	Delta
VBAT	117.7	96.5	21.2
VDDCORE	29.3	29.3	0
VDDARM	44.7	11.9	32.8
VDDMEM	5.6	3.6	2
VDDWRF	1.3	1.2	0.1
VDDWCN	22.7	32.7	-10
AVDD1V8	4.81	4.9	-0.09
VDDRF0	0.55	1.2	-0.65
VDD1V8	1.92	2.3	-0.38
AVDD2V8	0.343	0.15	0.193
VDD2V8	0.076	0.025	0.051
VDDDCXO	3.28	2.39	0.89
VDDWIFIPA	4.69	5.9	-1.21

功耗高在CPU上

```
|freq(HZ)-> |1300000    |900000     |768000     |ONLINE
|OFFLINE    |
|0(cpuid)   |27.842%    |0.627%     |16.955%    |45.424%
|54.576%    |
|1          |47.643%    |1.073%     |29.012%    |77.728%    |22.272%
|
```

测试机CPU loading

```
|freq(HZ)-> |1300000    |900000     |768000     |ONLINE
|OFFLINE    |
|0(cpuid)   |0.454%     |0.836%     |15.173%    |16.463%
|83.537%    |
|1          |2.727%     |5.026%     |91.195%    |98.948%
|1.052%     |
```

参考机CPU loading

CPU功耗高的主要原因是CPU频率run在了高频1.3G

场景功耗实例分析—BT收发文件类 2/2

- CPU功耗高问题分析—BT收发文件

CPU功耗高的主要原因，分析如下：

Name	Wall Duration
BTIF	1,071.450 ms
BTU	1,013.252 ms
BtOpp ClientThr	362.650 ms
krtccd	292.159 ms
Binder 5	225.344 ms
Notification Up	223.628 ms
bt hc worker	188.586 ms
userial read	167.774 ms
Bluetooth Share	141.593 ms
ndroid.systemui	130.139 ms
kworker/0:2	113.577 ms
ksmd	106.751 ms
GC	93.584 ms
Binder 1	35.972 ms
system server	33.415 ms
FinalizerDaemon	29.878 ms
Binder 3	25.052 ms
Binder B	24.235 ms
gki timer	21.417 ms

从systrace上看，测试机上BTU，BTIF比参考机要搞出近1S的使用率。

测试机

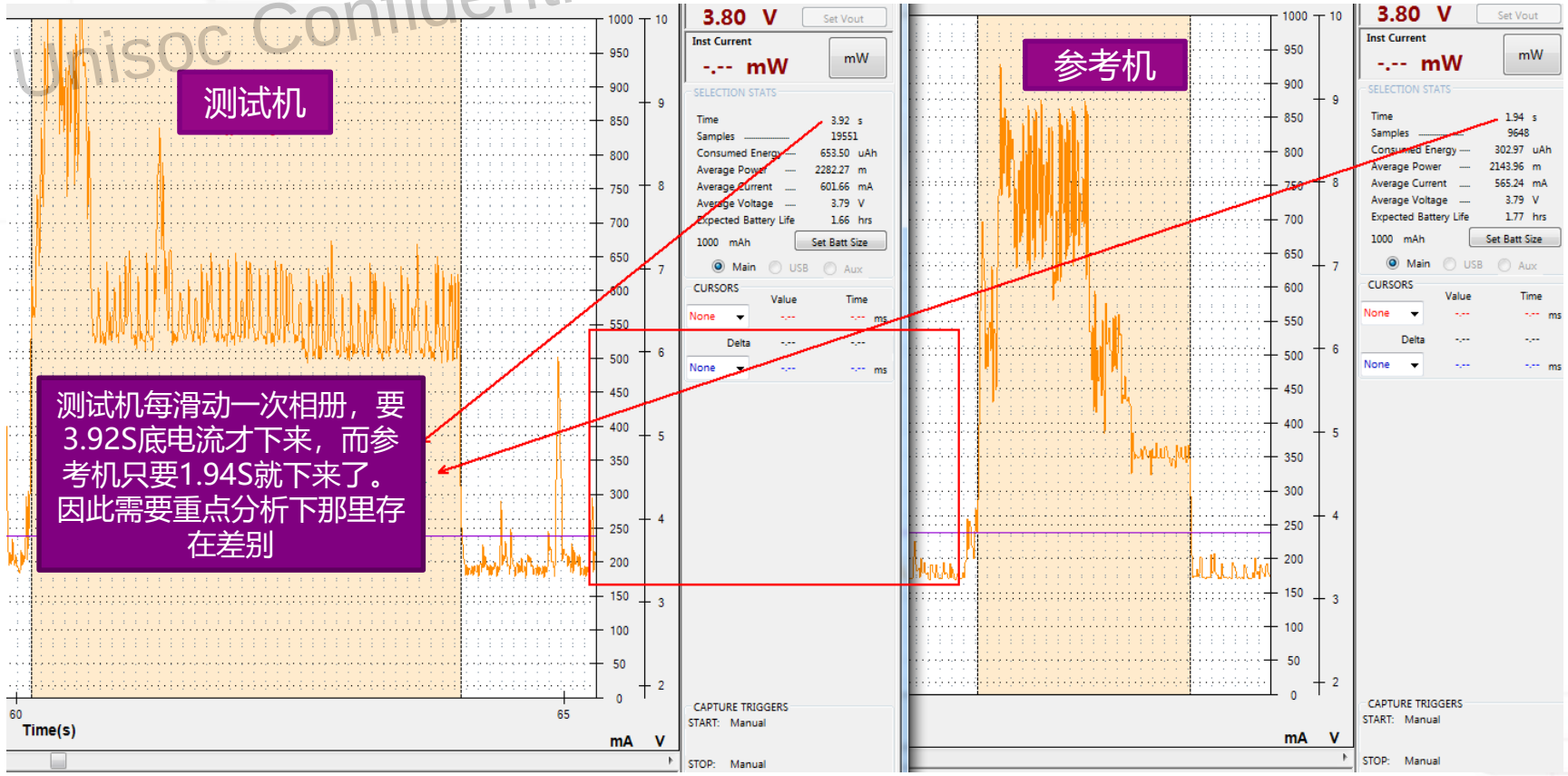
Name	Wall Duration
ext4:BtOpp ClientThr	361.289 ms
bt hc worker	289.118 ms
userial read	129.040 ms
Notification Up	120.373 ms
BTU	105.719 ms
Bluetooth Share	93.906 ms
ndroid.systemui	73.608 ms
kworker/0:1	64.089 ms
Binder 3	56.398 ms
system server	47.813 ms
ActivityManager	30.941 ms
Binder 4	21.273 ms
Binder 2	20.023 ms
Binder 1	18.738 ms
blockmmcqd/0	18.527 ms
droid.bluetooth	17.739 ms
Compiler	16.656 ms
Binder 6	16.076 ms
cfinteractive	15.286 ms
Binder 5	14.889 ms
Binder 8	13.823 ms
Binder 9	12.936 ms
Binder 7	12.634 ms
BTIF	12.443 ms
FinalizerDaemon	11.871 ms

参考机

场景功耗实例分析—浏览相册类 1/2

●CPU功耗高问题分析—浏览相册功耗

像浏览相册这种case都是有周期的，只需要分析一个周期的波形即可。

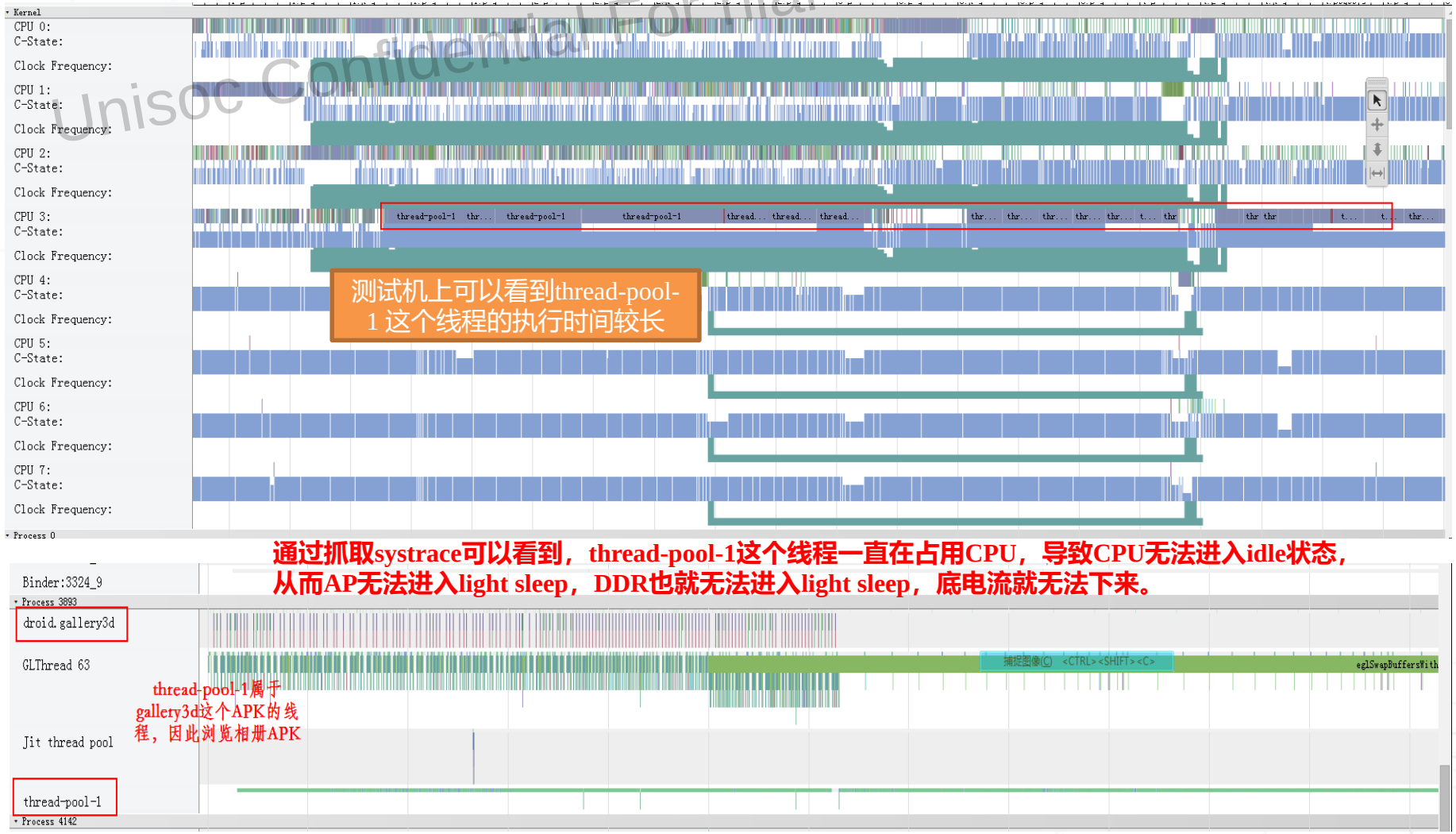


线程耗时长问题分析

通过分析一个周期的波形图可以看到，测试机滑动一张照片之后，电流很久才会下来。
通过分析DDR、GPU、CPU等主要模块的频率和loading，最终锁定问题在CPU上。

场景功耗实例分析—浏览相册类 2/2

●CPU功耗高问题分析—浏览相册功耗



通过抓取systrace可以看到，thread-pool-1这个线程一直在占用CPU，导致CPU无法进入idle状态，从而AP无法进入light sleep，DDR也就无法进入light sleep，底电流就无法下来。

Systrace分析线程耗时长

场景功耗实例分析—图层信息类 1/2

●亮屏前台类—图层信息问题

Temprun2 游戏

```
h/w composer state:
  h/w composer present and enabled
Hardware Composer state (version 01040000):
  mDebugForceFakeVSync=0
  Display[0] configurations (* current):
    * 0: 720x1280, xdpi=268.941010, ydpi=268.694000, refresh=16666000, colorTransform=-22numHwLayers=2, flags=00000000
     type   |  handle  | hint | flag | tr | blnd |   format    |     source crop(l,t,r,b)      |          frame          | name
GLES | 7f407a6451c0 | 0000 | 0000 | 00 | 0100 | RGB_565     |    0.0,    0.0,  720.0, 1280.0 |    0,    0,  720, 1280 | SurfaceView -
com.imangi.templerun2/com.imangi.unityactivity.ImangiUnityNativeActivity
 FB TARGET | 7f407a643d20 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0,  720.0, 1280.0 |    0,    0,  720, 1280
|HWC_FRAMEBUFFER_TARGET
Total allocated (estimate): 37491.56 KB
```

默认版本是GPU合成

```
h/w composer state:
  h/w composer present and enabled
Hardware Composer state (version 01040000):mDebugForceFakeVSync=0
  Display[0] configurations (* current):
  * 0: 720x1280, xdpi=268.941010, ydpi=268.694000, refresh=16666000, colorTransform=-22 numHwLayers=2, flags=00000000
    type   |  handle  | hint | flag | tr | blnd |   format    |     source crop(l,t,r,b)      |          frame          | name
HWC | a93f50e0 | 0000 | 0000 | 00 | 0100 | RGB_565     |    0.0,    0.0,720.0, 1280.0 |    0,    0,  720, 1280 | SurfaceView -
com.imangi.templerun2/com.imangi.unityactivity.ImangiUnityNativeActivity
 FB TARGET | a93f4780 | 0000 | 0000 | 00 | 0105 | RGBA_8888   |    0.0,    0.0,720.0, 1280.0 |    0,    0,  720, 1280 | HWC_FRAMEBUFFER_TARGET
Total allocated (estimate): 37365.00 KB
```

修改之后使用DPU/GSP合成

场景功耗实例分析—图层信息类 2/2

- 亮屏前台类—图层信息问题

```
h/w composer state:
  h/w composer enabled
 comp type :   format : fbc : pitch : height : transform :    blend : alpha : zorder :   dl   dt   dr   db :
       GPU :  RGB565  :   N :   720 :   1280 :         0 :     NONE :   255 :      0 :
       GPU :    OTHER :   N :   640 :    480 : 90&FLIP_H :     NONE :   255 :      1 :    0    0    0    0 :
       GPU :  RGB565  :   N :   144 :    256 :         0 :     NONE :   255 :      2 :    0    0    0    0 :
       GPU : RGBA8888 :   N :   720 :   1280 :         0 :  PREMULT :   255 :      3 :    0    0    0    0 :
       GPU : RGBA8888 :   N :   720 :     48 :         0 :  PREMULT :   255 :      4 :    0    0    0    0 :
Output:GPU - RGBA8888 :   N :   720 :   1280 :
```

默认版本是GPU合成

```
Display 0 HWC layers:
 Layer name
           Z :  Comp Type :   Disp Frame (LTRB) :          Source Crop (LTRB)
 com.tencent.mm/com.tencent.mm.plugin.voip.ui.VideoActivity#0
  rel      0 :     Device :    0    0  720 1280 :    0.0    0.0  720.0 1280.0
 StatusBar#0
  rel      0 :     Device :    0    0  720   48 :    0.0    0.0  720.0   48.0

h/w composer state:
  h/w composer enabled
 comp type :   format : fbc : pitch : height : transform :    blend : alpha : zorder :   dl   dt   dr   db :
       DPU : RGBA8888 :   N :   720 :   1280 :         0 :     NONE :   255 :      0 :    0    0  720 1280 :
       DPU : RGBA8888 :   N :   720 :     48 :         0 :  PREMULT :   255 :      1 :    0    0    0    0 :
```

修改之后使用DPU合成

Wechat video功耗优化
140mA
(356mA->216mA)

场景功耗实例分析—DDR功耗类

●亮屏前台类—DDR功耗问题

Wechat Video DDR功耗分析：

测试机DDR频点分布：
933 = 100%, 622 = 0%, 368 = 0%, 233 = 0%, error = 0

参考机DDR频点分布：
1200 = 1, 600 = 99, 400 = 0, 200 = 0, error = 0

从DDR频率分布来看，测试机基本上都在最高频，所以DDR功耗理论上可以优化。

频点	功耗	性能情况(帧率)
DFS	1251 mA	34.4
622	872 mA	29.5

DDR频率固定622M之后，功耗有优化，说明这个case下可以降低DDR频率来降功耗。

场景功耗实例分析—CPU功耗类

●亮屏前台类—CPU功耗问题

Wechart Video CPU功耗分析

********** module0 freq load **********
freq(HZ)->	624000	936000	1248000	1560000	1872000	2028000	ONLINE	OFFLINE
0(cpuid)	23.387%	18.879%	6.700%	2.115%	1.346%	5.662%	58.088%	41.912%
1	22.093%	17.834%	6.329%	1.998%	1.271%	5.348%	54.874%	45.126%
2	22.164%	17.892%	6.350%	2.004%	1.275%	5.366%	55.051%	44.949%
3	22.641%	18.276%	6.486%	2.047%	1.303%	5.481%	56.234%	43.766%
SUM:2236209.258240

********** module1 freq load **********

freq(HZ)->	624000	936000	1248000	1560000	1872000	2028000	ONLINE	OFFLINE
4(cpuid)	12.413%	19.408%	6.517%	3.175%	1.050%	5.522%	48.085%	51.915%
5	12.897%	20.164%	6.771%	3.299%	1.091%	5.738%	49.960%	50.040%
6	13.256%	20.725%	6.959%	3.391%	1.122%	5.897%	51.350%	48.650%
7	12.452%	19.468%	6.537%	3.185%	1.054%	5.539%	48.234%	51.766%
SUM:2143878.996354

测试机 CPU总loading

TOTAL:4380088.254594

********** module0 freq load **********
|freq(HZ)-> |768000 |1032000 |1154000 |1276000 |1398000 |1490000 |ONLINE |OFFLI
|0(cpuid) |0.170% |0.756% |4.086% |4.438% |27.679% |26.987% |64.116% |35.8%
|1 |0.099% |0.438% |2.369% |2.573% |16.049% |15.647% |37.175% |62.825% |
|2 |0.050% |0.221% |1.198% |1.301% |8.112% |7.909% |18.790% |81.210%
|3 |0.018% |0.080% |0.433% |0.470% |2.934% |2.861% |6.796% |93.204%
|SUM:1784837.222628|

********** module1 freq load **********

freq(HZ)->	1536000	1707000	1874000	2000000	ONLINE	OFFLINE
4(cpuid)	0.097%	0.000%	0.000%	0.000%	0.097%	99.903%
5	0.164%	0.000%	0.000%	0.000%	0.164%	99.836%
6	0.091%	0.000%	0.000%	0.000%	0.091%	99.909%
7	0.092%	0.000%	0.000%	0.000%	0.092%	99.908%
SUM:6821.795185

参考机 CPU总loading

TOTAL:1791659.017813

频点	功耗	性能情况(帧率)
DVFS	1251 mA	34.4
MAX936	**965 mA**	**34**

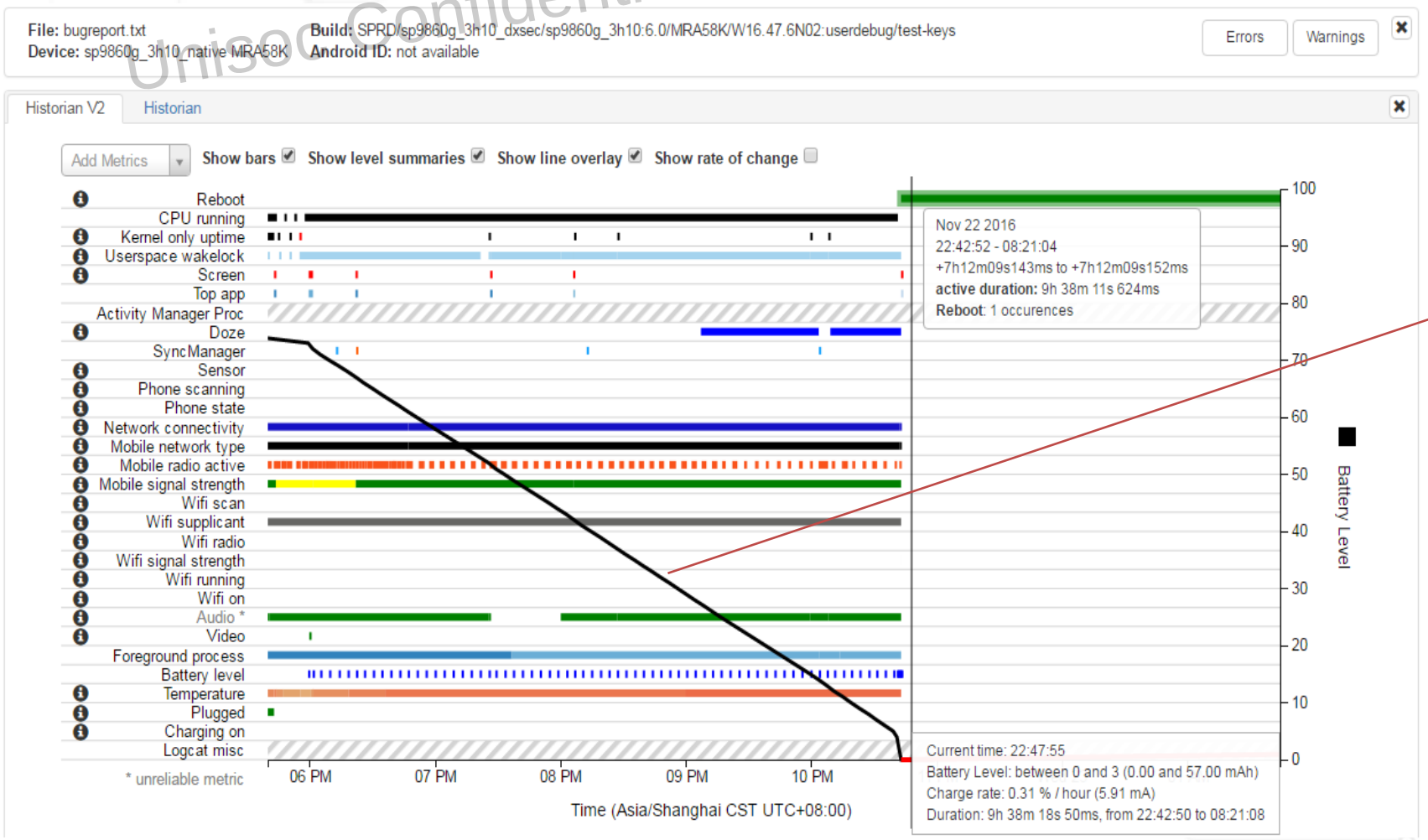
测试机的CPU loading远远大于参考机的CPU loading，因此需要从如下2个方面优化：
1、尝试降低CPU loading
2、优化CPU DVFS和CPU调度算法

可以看到把CPU频率限制到最大936M之后，在不影响性能情况下，功耗下降很多。

场景功耗实例分析—长待机续航类 1/3

- **长待机续航功耗问题**

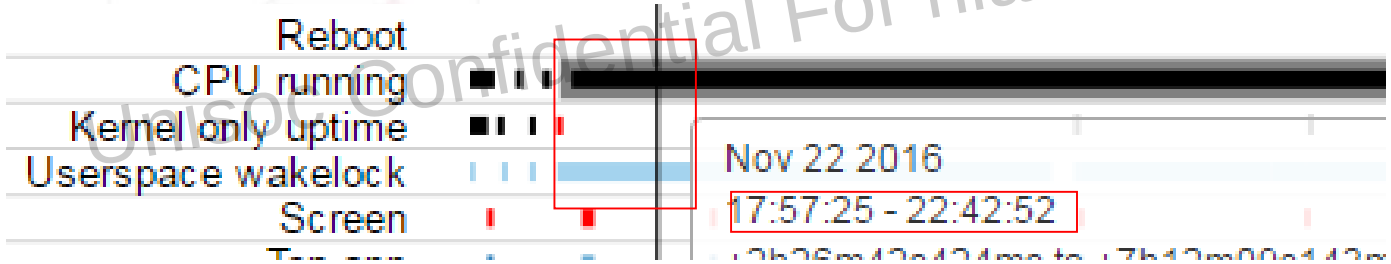
长待机案例分享：手机在晚上8点钟40%电量待机，早上看手机已经没电关机。



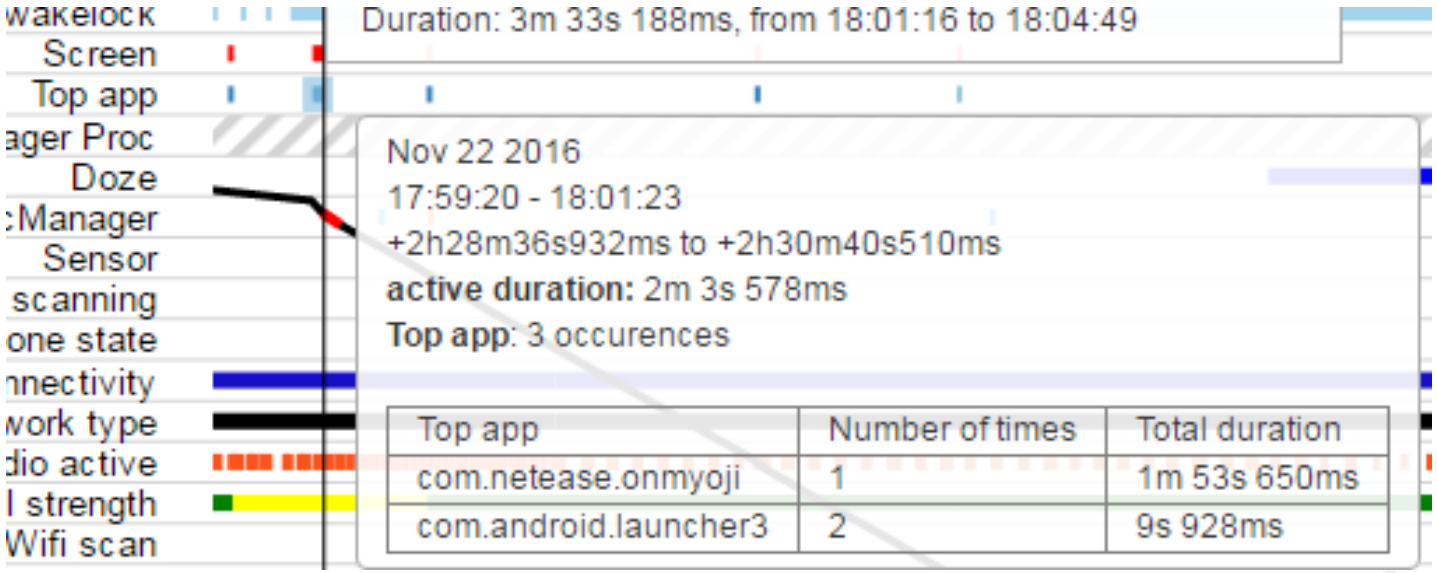
电池电量曲线，可以看到电池电量很快耗尽。

场景功耗实例分析—长待机续航类 2/3

- 长待机续航功耗问题



17:57之后CPU一直在工作，userspace wakelock 被一直拿锁，但是在17：57分之前并无此现象（CPU running 和 userspace wacklock版本）。



通过Top APP栏位发现用户在17：59打开了com.netease.onmyoji（百度搜索为“阴阳师“手游）。

场景功耗实例分析—长待机续航类 3/3

- **长待机续航功耗问题**

Userspace Wakelocks:

Show 5 entries　　Search:　　Copy

Ranking	Name	Uid	Duration / Hr	Count / Hr	Minimum Duration	Total Count
0	MEDIA : AudioMix	1013	44m21s294 ms	3.31	5h8m36.228 s	23
1	com.netease.onmyoji : AudioMix	10090	7m7s576ms	0.14	49m34.906s	1
2	com.tencent.mm : WakerLock:190860297	10087	9s456ms	15.52	1m5.792s	108

然后查看Userspace Wakelocks，发现audiomix 一直处于拿锁的状态，持续时间5小时。

结论：

由以上信息初步判断为阴阳师APK导致audiomix 持续拿锁，导致耗电快。

后续多次验证证实初步判断是正确的，阴阳师APK 在home键退出程序后会有audiomix 拿锁现象，导致耗电高。

04

场景功耗分析总结

场景功耗分析总结 1/2

场景类型	前置条件	抓取信息
待机相关	关闭modem log，网络类型一致，信号强度一致，单双卡一致。	Bugreport(长待机)，电流波形图 提供飞行模式下的功耗数据和波形。
后台运行	后台运行应用一致； 后台运行暂停功耗是否有差异。	CPU loading,systrace,top DDR loading,DDR带宽 HW/SW分解数据 Light/deep sleep的功耗数据和波形。
亮屏case	亮屏静止状态的功耗是否一致	CPU loading,systrace,top DDR loading,DDR带宽 GPU loading，GSP/VSP相关信息 LCD ON的功耗和波形及场景暂停之后的功耗数据和波形。
网络相关	确保网络类型一致； 传输的数据量大小保持一致，信号强度一致。	CPU loading,systrace,top DDR loading,DDR带宽 GPU loading，GSP/VSP相关信息，Cap包 关闭网络之后的功耗数据，排除是否网络影响的功耗。

详细分析

场景功耗分析总结 2/2

对功耗分析方法了解了多少呢?

功耗分析需要不断积累

分析电流波形	HW方法	SW方法
• 找波形规律 • 看波形底电流和高度以及宽度	• HW分解 • HW拆外设 • 测量相关domain波形	• SW逐步拆解场景 • SW check各IP的信息 • Check work flow • Check寄存器，中断等信息。 • 分析log

功耗分析需要对数据敏感

功耗一般从以上三个方面入手进行分析

谢 谢