HYGON

海光信息技术股份有限公司

Hygon Information Technology Co., Ltd.

HYGON HGT User Manual

内容

修订记录

日期	版本	描述
2020/05/06	1.4	初始发布版本
2020/05/23	1.5	修订部分缺陷
2020/05/28	1.6	修订部分缺陷
2020/09/28	2.0	增加 XGMI TX EQ Tune 功能
2020/10/22	2.1	支持在相同主板上搭配不同类别 CPU 的情况
2020/11/23	2.2	修复 SL1/DM1 CPU 封装识别错误问题
2021/10/11	2.5	增加挑选最佳 PCIe EP TX Preset 功能

1. 概述

HGT(Hygon xGMI margin Tool) 是海光公司开发的用于评估和优化海光处理器的 SATA/PCIe/xGMI 信号完整性工具。

1.1 功能描述

HGT 提供了以下子命令：

- eye： 绘制并分析 xGMI、PCIe、SATA 等 RX 链路的眼图。
- xgmitxeqtune：对 XGMI 链路 TX EQ 参数做 Tune 测试
- xgmigenapcbfile：根据 XGMI TX EQ Tune 结果产生 BIOS 所需 APCB 头文件
- pcietxeptune：对 PCIe 链路 EP TX Preset 参数做 Tune 测试

1.2 运行环境

HGT 的运行环境要求如下：

硬件	搭配了 海光处理器的主板
处理器	海光 ES 芯片 海光 MP 芯片（需要 unlock）
BIOS PI 版本	不低于 1.0.1.1
操作系统	Linux（Ubuntu server 16.04，CentOS 7.5）

特别注意

如需在海光量产版本芯片(MP)上运行 HGT，必须先对处理器进行安全解锁(unlock)操作。unlock 工具和使用文档请联系海光 FAE 现场工程师获取，对应工具名称为 HygonSecureUnlockTool。

2. 眼图绘制

本章详细介绍 HGT 的眼图绘制步骤。

2.1 环境需求

- 目标链路应连接了设备。
- 执行 HGT 命令需要 root 权限

2.2 运行方法

2.2.1 确定目标 Die 和 Lane

首先需要确认待测 lane 的编号及 DIE 编号。

DXIO 由 Type A 和 Type B 两部分组成，TYPE A 中的 PHY 是多功能的，支持 SATA/PCIE/ xGMI 三种协议, Type B 下的 PHY 支持 xGMI/PCIE 两种协议。Type A 由 2 个 X2 的 PHY 和 3 个 X4 的 PHY 组成,包含 16 条 Lane， Type B 同样也是由 2 个 X2 的 PHY 和 3 个 X4 的 PHY 组成, 包含 16 条 Lane。

命令： ./hgt eye -s -d 0

Command	Parameter	Notes
eye	-s	显示 DXIO 各个 PHY 的工作模式
	-d	指定 die number

执行命令之后，信息以表格形式输出，如图一所示。“lane”表示 DXIO lane 编号，“mode”表示 lane 的配置模式，“P”表示 PCIe， “S”表示 SATA， “G”表示 xGMI (注：若某条 lane 或者某几条 lane 没有配置，则默认显示为 G)

```
[root@localhost hgt]# ./hgt eye -s -d 0
Die 0
+-------+-------------------------------+-------------------------------+
|       |            Type B             |            Type A             |
+-------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       |3|3|2|2|2|2|2|2|2|2|2|2|1|1|1|1|1|1|1|1|1|1| | | | | | | | | | |
| lane  |1|0|9|8|7|6|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|
+-------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| mode  |   P   |   P   |   P   | P | P |   P   |   G   |   S   | S | S |
+-------+-------+-------+-------+---+---+-------+-------+-------+---+---+
```

图 1 HGT 显示 lane 配置

2.2.2 采集数据并绘制眼图

命令： ./hgt eye -d 3 -m 100000

Command	Parameter	Notes
eye	-d	指定 die number
	-m	以位的形式指定待测试 lane， 16 进制无符号整数，bit 位为“1”表示测试 lane，为“0”表示不测试

使用-m 参数按位指定某个 Die 的若干 Lane(s)，然后采集数据并绘制眼图。

如图 2 中的“-m 100000”，其中的 100000 是 16 进制数，bit20 被置“1”, 表示测试目标是 lane 20, 可以参考上一章节 2.2.1 中‘-s’配合测试，注意被测的 Lane 要连接正常工作的设备，否则无法测试。

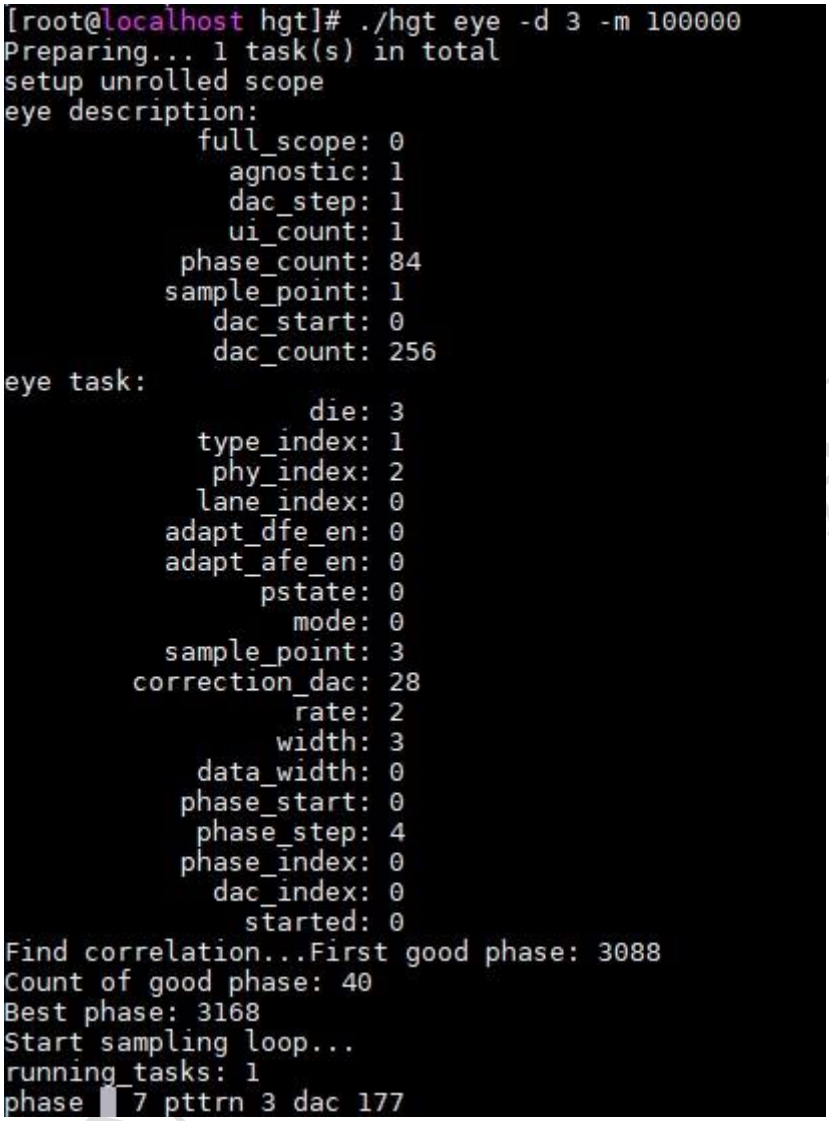

图 2 HGT 绘制眼图

2.2.3 查看结果

眼图绘制完成后，HGT工具会在当前目录下为每条 lane 生成两个文件：*.bmp和*.txt。假设绘制了 Die 3 Lane 20 的眼图，那么会生成D3T1P2L0.bmp 和 D3T1P2L0.txt 两个文件，其中 D3T1P2L0.bmp 是眼图，如图3所示。

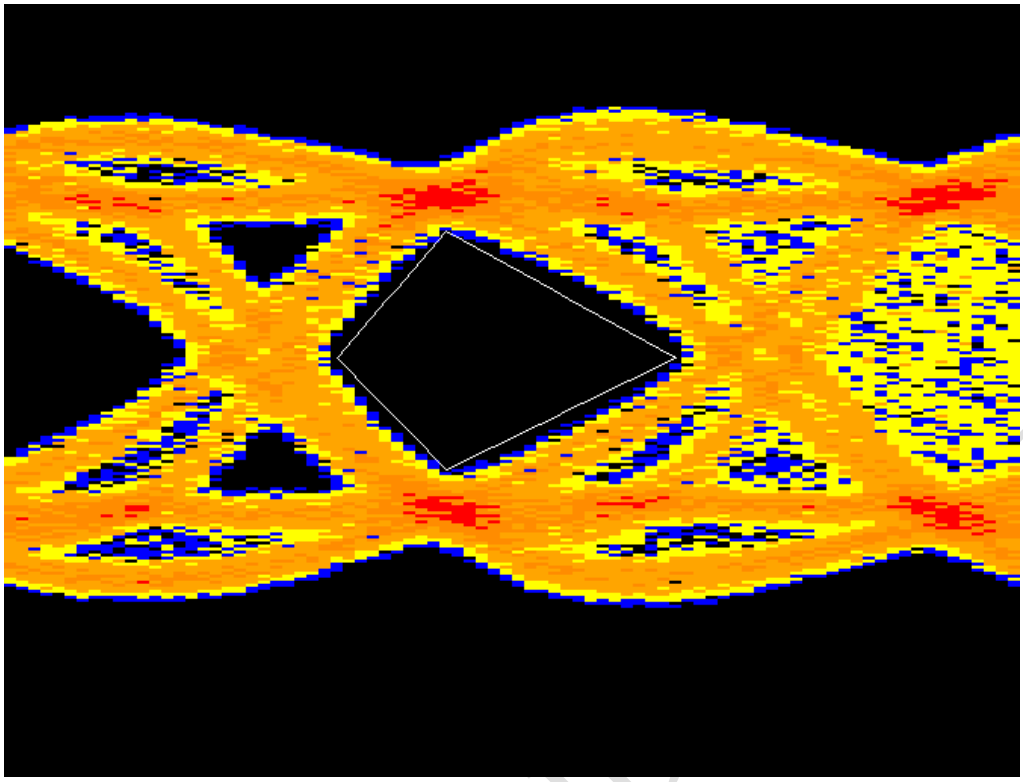

图 3 眼图示意图

D3T1P2L0.txt 中包含了眼图的测量数据和该 Lane 的当前状态，如图4所示：

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
Eye opening area: 1331 units
Eye max width: 29 units, 0.69 UI
Eye max height: 80 units

att 7
vga1 0
vga2 0
boost 5
pole 6
tap1 124
tap2 5
tap3 -7
tap4 14
tap5 14
fom 95
```

图 4 眼图测量数据

2.3 xGMI 测试通过标准

由于 xGMI 是 HYGON CPU 间特有的互联链路，HGT 给出了 xGMI 信号完整性的测试标准。对于 xGMI 测试，满足如下标准即通过。

表格 1 xGMI 测试标准

Max Eye width	Max Eye height
>0.30 UI	>25 units

举例：

在 2.2.3 中介绍了如何查看 HGT 的测试结果，结果中包含了一个 txt 文件用来描述眼图的测量数据，关注下图中红框部分：

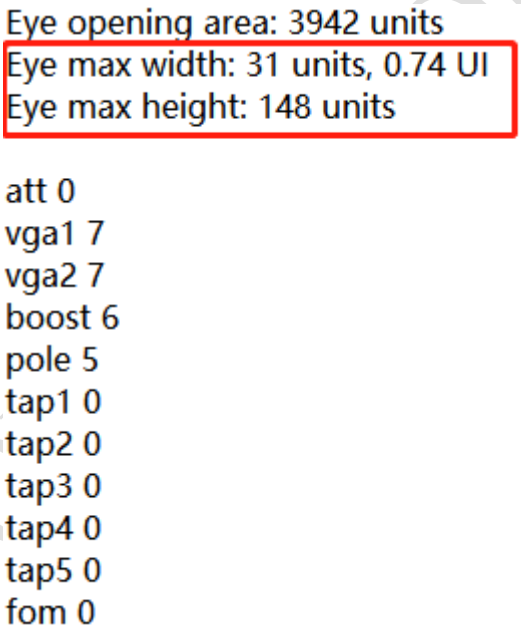

图 5 XGMI 测试举例

可以得知 Max Eye width=0.74 UI, Max Eye height=148 units

那么参考表 1 XGMI 的测试标准：

Max Eye width = 0.74 UI > 0.30UI

Max Eye height = 148 units > 25 units

测试结果表明这条 xGMI lane 测试通过。

3. XGMI TX EQ Tune

XGMI TX EQ Tune 可以测试每条 xGMI lane 的 TX EQ 参数，并以*.h 头文件格式输出测试结果，将测试结果更新到 BIOS 中可以增强 XGMI 信号，改善眼图质量。

3.1 环境需求

- 双路服务器
- 执行 HGT 需要 root 权限

3.2 Tune 测试

XGMI TX EQ Tune测试使用子命令xgmitxeqtune，参数说明如下：

Command	Parameter	Notes
xgmitxeqtune	-d die	specify die index
	-m lanes_bitmask	specify lanes indexed in one die, bit[15:0]: xGMI(default: 0xFFFF)
	-r datarate	specify data rate, 10:10.6G(default) 9: 9.6G 8: 8.5G
	-h	show this message

命令示例：

```
./hgt xgmitxeqtune -d 1 -m 0xffff -r 10
```

Tune测试需要在发送端遍历多组EQ参数，在RX端评估眼图质量，每个die 16条lane的测试时间约1.5小时。

3.3 生成 APCB 头文件

XGMI TX EQ Tune测试完成后可以使用子命令xgmigenapcbfile生成XGMI APCB头文件。

该命令包含-g/-c 参数：

./hgt xgmigenapcbfile -g

该命令生成的 APCB 头文件将包含工具所在执行环境的 CPU 信息前缀，如在海光 2 号 CPU,SL1 封装环境下执行该命令，会生成如下 3 个头文件:

```
DPSL1_xgmi_apcb_settings_10.h
DPSL1_xgmi_apcb_settings_9.h
DPSL1_xgmi_apcb_settings_8.h
```

./hgt xgmigenapcbfile -c

该命令将 hgt 工具所在目录下所有带有 CPU 信息前缀的，相同速度的 APCB 头文件合并成一份，最终生成如下三个文件:

```
xgmi_apcb_settings_10.h
xgmi_apcb_settings_9.h
xgmi_apcb_settings_8.h
```

将这三个头文件拷贝到 PI 目录

AgesaPkg/Addendum/Apcb/#BoardName#/ApcbData/AGT_Data/Instance0/

然后重新编译集成该 PI 的 BIOS 即可。

XGMI APCB头文件内容示例见附录4.1。

3.4 完整 tune 测试

- STEP 1:解压 HGT 工具包到被测环境(注意：在开始前要保证 HGT 工具所在目录下没有 HGT 生成的临时文件)
- STEP 2:使用 xgmitxeqtune_all.sh 对所有 die 的全部 lane，全部速率进行 tune 测试，并生成带有 CPU 信息的 APCB 头文件。7 x x x 双路系统完整 tune 测试时间大约需要 36 小时。

```
#!/bin/bash


DieNum=$(lspci -n -d :1450|wc -l)


for((data_rate = 10; data_rate > 7; data_rate--))
do
        for((die = 0; die < $DieNum; die++))
        do
                ./hgt xgmitxeqtune -d $die -m 0xffff -r $data_rate
        done
done


./hgt xgmigenapcbfile
```

- STEP 3:如果被测环境同时支持多类(不同代/不同封装)CPU，确认 Step 2 测试完毕，关机更换另一款 CPU 后启动到系统，在 Step 2 的 hgt 目录下继续执行 Step 2。直到将该环境支持的所有类别的 CPU 都测试完毕。如果被测环境仅支持一类 CPU 则执行 Step 4。
- STEP 4:执行./xgmigenfinalapcb.sh，生成最终的 APCB,最终的 APCB 将放在 apcb_hgt_data 目录下。
- STEP 5:将生成的 APCB 放到 BIOS 源代码中如下目录

AgesaPkg/Addendum/Apcb/#BoardName#/ApcbData/AGT_Data/Instance0/

4. PCIe EP TX Preset Tune

PCIe EP TX Preset Tune 可以挑选出最优 EP TX Preset 参数，并以 HYGON_EQ_CFG 文件输出测试结果，将测试结果更新到 BIOS 中可以增强 PCIE 信号，改善眼图质量。

4.1 环境需求

- 海光 2 号、3 号处理器
- PCIe Gen3、Gen4 设备
- 执行 HGT 需要 root 权限

4.2 Tune 测试

PCIe EP TX Preset Tune测试使用子命令pcietxeqtune，参数说明如下：

Command	Parameter	Notes
pcietxeqtune	-s bus:device.function	specify device BDF.
	-presetlist=”0 1 2 3 4 5 6 7 8 9”	specify endpoint tx preset list：0~9.
	-v vendor_id	specify device vendor id and device id.

注意；

-v vendor_id 功能可选，在指定设备 vendor、device id 后，只有在指定卡插在指定插槽上时，才会固定 preset。

命令示例：

./hgt pcietxeqtune -s 61:00.0 --presetlist="3 4 5 6 7" -v 0x005d1000

Tune测试需要在发送端遍历多个 preset参数，在RX端评估眼图质量，每个die 16条lane的10 个 preset。

4.3 生成 HYGON_EQ_CFG 文件

PCIe EP TX Preset Tune测试完成后会生成一个 HYGON_EQ_CFG文件，文件内容如下：

```
{FLAG_SPECIFIED, pcie_gen3_ttx_force_otherside_preset,
Cpu_Gen_DhyanaPlus, 6, 0x101315b3, 0x001315b3, 16, 23, True,
{3,3,3,3,3,3,3,3}}
```

使用文件内容更新 BIOS 中的 EqConfigurationTable，然后重新编译 BIOS 即可。

4.4 完整 Tune 测试

4.4.1 指定插槽模式

指定插槽模式是对指定的 Bus、Device、Function 的 PCIe 卡进行最优 EP preset 挑选。测试中工具会根据 CPU 和设备都支持的最大速率选择 Gen3、Gen4 速率测试。在输出结果里，会指定插槽所在的 lane，PCIe 速率、以及最优 Preset。

指定插槽模式示例：

- STEP 1:在指定插槽上插入 PCIE 卡。如果想固定 Gen3 Preset 请插入 PCIe Gen3 卡；如果想固定 Gen4 Preset 请插入 PCIe Gen4 卡；
- STEP 2:启动系统后，解压 HGT 工具包到被测环境；
- STEP 3:使用 lspci -tv 查看需要固定 Preset 的 Bus、device、function；

```
-[0000:00]-+-00.0  Chengdu Higon IC Design Co.Ltd Root Complex
           +-00.2  Chengdu Higon IC Design Co.Ltd I/O Memory Management Unit
           +-01.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-01.3-[01-02]--+-00.0  Intel Corporation I350 Gigabit Network Connection
           |               \-00.1  Intel Corporation I350 Gigabit Network Connection
           +-01.5-[03-04]----00.0-[04]----00.0  ASPEED Technology, Inc. ASPEED Graphics Family
           +-02.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-03.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-03.1-[05-06]--+-00.0  Intel Corporation Ethernet Controller X710 for 10GbE SFP+
           |               \-00.1  Intel Corporation Ethernet Controller X710 for 10GbE SFP+
           +-04.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-07.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-07.1-[07]--+-00.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Function
           |            +-00.2  Chengdu Higon IC Design Co.Ltd PSPCCP Command DMA Processor
           |            \-00.3  Chengdu Higon IC Design Co.Ltd USB 3.0 Host controller
           +-08.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
```

- STEP 4:使用 HGT 工具对指定的 Bus、device、function 的设备做 Preset 测试。

./hgt pcietxeqtune -s 5:00.0 --presetlist="0 1 2 3 4 5 6 7 8 9"

测试完成后会打印 preset 挑选过程，用于验证挑选的 preset 是否正确。

```
preset:0, ew_min:0.788732, ew_average:0.809859
preset:1, ew_min:0.845070, ew_average:0.852113
preset:2, ew_min:0.816901, ew_average:0.823944
preset:3, ew_min:0.845070, ew_average:0.866197
preset:4, ew_min:0.873239, ew_average:0.894366
preset:5, ew_min:0.873239, ew_average:0.873239
preset:6, ew_min:0.845070, ew_average:0.862676
preset:7, ew_min:0.788732, ew_average:0.809859
preset:8, ew_min:0.816901, ew_average:0.830986
preset:9, ew_min:0.816901, ew_average:0.830986
best preset:4
```

测试结果 HYGON_EQ_CFG 如下：

```
-rw-r--r--  1 root root  129 Oct 11 17:54 HYGON_EQ_CFG
drwxr-xr-x  2 root root 4096 Oct 11 17:34 preset-0-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:36 preset-1-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:38 preset-2-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:41 preset-3-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:43 preset-4-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:45 preset-5-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:47 preset-6-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:50 preset-7-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:52 preset-8-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:54 preset-9-eye/
root@518:~/eye-preset-test/20211011173159# cat HYGON_EQ_CFG
{FLAG_FORCE, pcie_gen3_ttx_force_otherside_preset, Cpu_Gen_Dhyana2n7, 0, 0xffffffff, 0x00018086, 16, 31, True, {4,4,4,4,4,4,4,4}}
```

- STEP 5:根据工具的结果更新 BIOS 中的 EqConfigurationTable。

4.4.2 指定插槽和卡模式

指定插槽和卡模式是对指定 Bus、Device、Function、vendor/device id 的 PCIe 卡进行最优 EP preset 挑选。测试中工具会根据 CPU 和设备都支持的最大速率选择 Gen3、Gen4 速率测试。在输出结果里，会指定插槽所在的 lane，vendor、device id、PCIe 速率、以及最优 Preset。

指定插槽和卡模式示例：

- STEP 1:在指定插槽插上指定的 Gen3 或则 Gen4 PCIE 卡；
- STEP 2:启动系统后，解压 HGT 工具包到被测环境；
- STEP 3:使用 lspci -tv 看需要固定 Preset 的 Bus；

```
-[0000:00]-+-00.0  Chengdu Higon IC Design Co.Ltd Root Complex
           +-00.2  Chengdu Higon IC Design Co.Ltd I/O Memory Management Unit
           +-01.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-01.3-[01-02]--+-00.0  Intel Corporation I350 Gigabit Network Connection
           |               \-00.1  Intel Corporation I350 Gigabit Network Connection
           +-01.5-[03-04]----00.0-[04]----00.0  ASPEED Technology, Inc. ASPEED Graphics Family
           +-02.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-03.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-03.1-[05]--00.0  Samsung Electronics Co Ltd Device a824
           +-04.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-07.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
           +-07.1-[06]--+-00.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Function
           |            +-00.2  Chengdu Higon IC Design Co.Ltd PSPCCP Command DMA Processor
           |            \-00.3  Chengdu Higon IC Design Co.Ltd USB 3.0 Host controller
```

- STEP 4: lspci -xs bus:0.0 查看 PCIe 设备的 vendor id、device id；

```
root@518:~/eye-preset-test# lspci -xs 5:0.0
05:00.0 Non-Volatile memory controller: Samsung Electronics Co Ltd Device a824
00: 4d 14 24 a8 46 05 10 00 00 02 08 01 10 00 00 00
10: 04 00 81 f6 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 4d 14 01 a8
30: 00 00 70 f6 40 00 00 00 00 00 00 00 05 01 00 00
```

- STEP 5:使用 HGT 工具对指定的 PCIe 设备做 Preset 测试。

./hgt pcietxeqtune -s 5:00.0 --presetlist="0 1 2 3 4 5 6 7 8 9" -v 0xa824144d

测试完成后会打印 preset 挑选过程，用于验证挑选的 preset 是否正确。

```
preset:0, ew_min:0.450704, ew_average:0.492958
preset:1, ew_min:0.563380, ew_average:0.602113
preset:2, ew_min:0.535211, ew_average:0.573944
preset:3, ew_min:0.591549, ew_average:0.633803
preset:4, ew_min:0.676056, ew_average:0.707746
preset:5, ew_min:0.676056, ew_average:0.714789
preset:6, ew_min:0.676056, ew_average:0.700704
preset:7, ew_min:0.535211, ew_average:0.577465
preset:8, ew_min:0.591549, ew_average:0.630282
preset:9, ew_min:0.647887, ew_average:0.693662
best preset:5
```

测试结果 HYGON_EQ_CFG 如下：

```
-rw-r--r--  1 root root  129 Oct 11 17:22 HYGON_EQ_CFG
drwxr-xr-x  2 root root 4096 Oct 11 17:01 preset-0-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:04 preset-1-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:06 preset-2-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:08 preset-3-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:11 preset-4-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:13 preset-5-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:15 preset-6-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:18 preset-7-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:20 preset-8-eye/
drwxr-xr-x  2 root root 4096 Oct 11 17:22 preset-9-eye/
```

```
{FLAG_SPECIFIED, pcie_gen4_ttx_force_otherside_preset, Cpu_Gen_Dhyana2n7, 0, 0xa824144d, 0xa801144d, 16, 31, True, {5,5,5,5,5,5,5,5}}
```

- STEP 5：根据工具的结果更新 BIOS 中的 EqConfigurationTable。

4.4.3 BIOS 配置

1. 进入 AmdCpmPkg\Addendum\Oem\项目名称\Pei\HygonCpmOemInitPei 目录，打开 HygonCpmOemTable.c。
2. 在 HygonCpmOemTable.c 中添加如下 Table，注意对应的 SubSysVidDid 要修改成 0xFFFFFFFF。

```
HYGON_CPM_EQ_CONFIG_TABLE  gHygonEqConfigurationTable = {
 {CPM_SIGNATURE_EQ_CONFIG, sizeof (gHygonEqConfigurationTable)/sizeof(UINT8), 0, 0, 0, 0x0000000F},
  {
    {FLAG_FORCE,        pcie_gen3_ttx_force_otherside_preset,  Cpu_Gen_Dhyana2,  3,  0x005d1000,  0xFFFFFFFF,  16,  31,  TRUE,  {4} },
    {FLAG_SPECIFIED,  pcie_gen4_ttx_force_otherside_preset,  Cpu_Gen_Dhyana2,  2,  0xFFFFFFFF,  0xFFFFFFFF,  0,  15,  TRUE,  {4,} },
    {EQ_TABLE_END, eq_type_end, 0xFF, 0xFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, TRUE, {0}},
  }
}
```

3. 进入到 AmdCpmPkg\Addendum\Oem\HonghaiSL1SLT\Pei\HygonCpmOemInitPei 目录下，打开 HygonCpmOemInitPeim.c，将步骤 2 中新建的 gHygonEqConfigurationTable 添加到 gCpmTableList[]中。

注意：gHygonEqConfigurationTable 是举例名称，该名字根据具体的项目可以适当更改。

5. 附录

5.1 XGMI APCB 头文件示例

```
////////////////////////////////////////////////////////////////////////////////
/// Copyright 2020 HYGON, INC. All Rights Reserved.
/// This file was generated by hgt xgmitxeqtune. Do not modify.
////////////////////////////////////////////////////////////////////////////////


SL1_MAKE_TX_EQ_FREQ_TABLE_COMPACT(XGMI_TX_EQ_SPEED_107,SOC_GEN_1,SOC_SL1,
    MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET0_DIE0_EACH_LANES_COMPACT(),
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  0
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  1
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  2
        MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  3
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  4
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  5
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  6
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  7
        MAKE_TX_EQ_LANE_DATA( 30, 32,  8 ), // lane  8
        MAKE_TX_EQ_LANE_DATA( 32, 32,  0 ), // lane  9
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane 10
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 11
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 12
        MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane 13
        MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 14
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 )  // lane 15
    ),
    MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET0_DIE1_EACH_LANES_COMPACT(),
        MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane  0
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  1
```

```
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  2
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  3
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  4
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  5
        MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  6
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  7
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  8
        MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  9
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 10
        MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane 11
        MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 12
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 13
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 14
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 )  // lane 15
    ),
    MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET0_DIE2_EACH_LANES_COMPACT(),
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  0
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  1
        MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane  2
        MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  3
        MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  4
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  5
        MAKE_TX_EQ_LANE_DATA( 28, 36, 12 ), // lane  6
        MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  7
        MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane  8
        MAKE_TX_EQ_LANE_DATA( 28, 40,  8 ), // lane  9
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 10
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 11
        MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 12
        MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 13
```

```
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 29, 36,  8 )  // lane 15
        ),
        MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET0_DIE3_EACH_LANES_COMPACT(),
            MAKE_TX_EQ_LANE_DATA( 28, 48,  0 ), // lane  0
            MAKE_TX_EQ_LANE_DATA( 28, 48,  0 ), // lane  1
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  2
            MAKE_TX_EQ_LANE_DATA( 28, 48,  0 ), // lane  3
            MAKE_TX_EQ_LANE_DATA( 28, 40,  8 ), // lane  4
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  5
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  6
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  7
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane  8
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  9
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 10
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 11
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 12
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 13
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 )  // lane 15
        ),
        MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET1_DIE0_EACH_LANES_COMPACT(),
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  0
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  1
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  2
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  3
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  4
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  5
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  6
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  7
```

```
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  8
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  9
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane 10
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 11
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 12
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane 13
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 )  // lane 15
        ),
        MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET1_DIE1_EACH_LANES_COMPACT(),
            MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane  0
            MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane  1
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  2
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  3
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  4
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  5
            MAKE_TX_EQ_LANE_DATA( 29, 32, 12 ), // lane  6
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  7
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane  8
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane  9
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 10
            MAKE_TX_EQ_LANE_DATA( 29, 44,  0 ), // lane 11
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 12
            MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane 13
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 28, 44,  4 )  // lane 15
        ),
        MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET1_DIE2_EACH_LANES_COMPACT(),
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  0
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  1
```

```
            MAKE_TX_EQ_LANE_DATA( 30, 32,  8 ), // lane  2
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  3
            MAKE_TX_EQ_LANE_DATA( 30, 32,  8 ), // lane  4
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  5
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  6
            MAKE_TX_EQ_LANE_DATA( 32, 32,  0 ), // lane  7
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  8
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  9
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 10
            MAKE_TX_EQ_LANE_DATA( 30, 40,  0 ), // lane 11
            MAKE_TX_EQ_LANE_DATA( 29, 40,  4 ), // lane 12
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 13
            MAKE_TX_EQ_LANE_DATA( 29, 36,  8 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 )  // lane 15
        ),
        MAKE_TX_EQ_LIST_COMPACT(MAKE_TX_EQ_SOCKET1_DIE3_EACH_LANES_COMPACT(),
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  0
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  1
            MAKE_TX_EQ_LANE_DATA( 32, 32,  0 ), // lane  2
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  3
            MAKE_TX_EQ_LANE_DATA( 31, 28,  8 ), // lane  4
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane  5
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane  6
            MAKE_TX_EQ_LANE_DATA( 32, 32,  0 ), // lane  7
            MAKE_TX_EQ_LANE_DATA( 32, 32,  0 ), // lane  8
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane  9
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 ), // lane 10
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 11
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 12
            MAKE_TX_EQ_LANE_DATA( 31, 36,  0 ), // lane 13
```

```
            MAKE_TX_EQ_LANE_DATA( 31, 32,  4 ), // lane 14
            MAKE_TX_EQ_LANE_DATA( 30, 36,  4 )  // lane 15
        )
    ),
```