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
2022/3/1	2.6	优化 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。

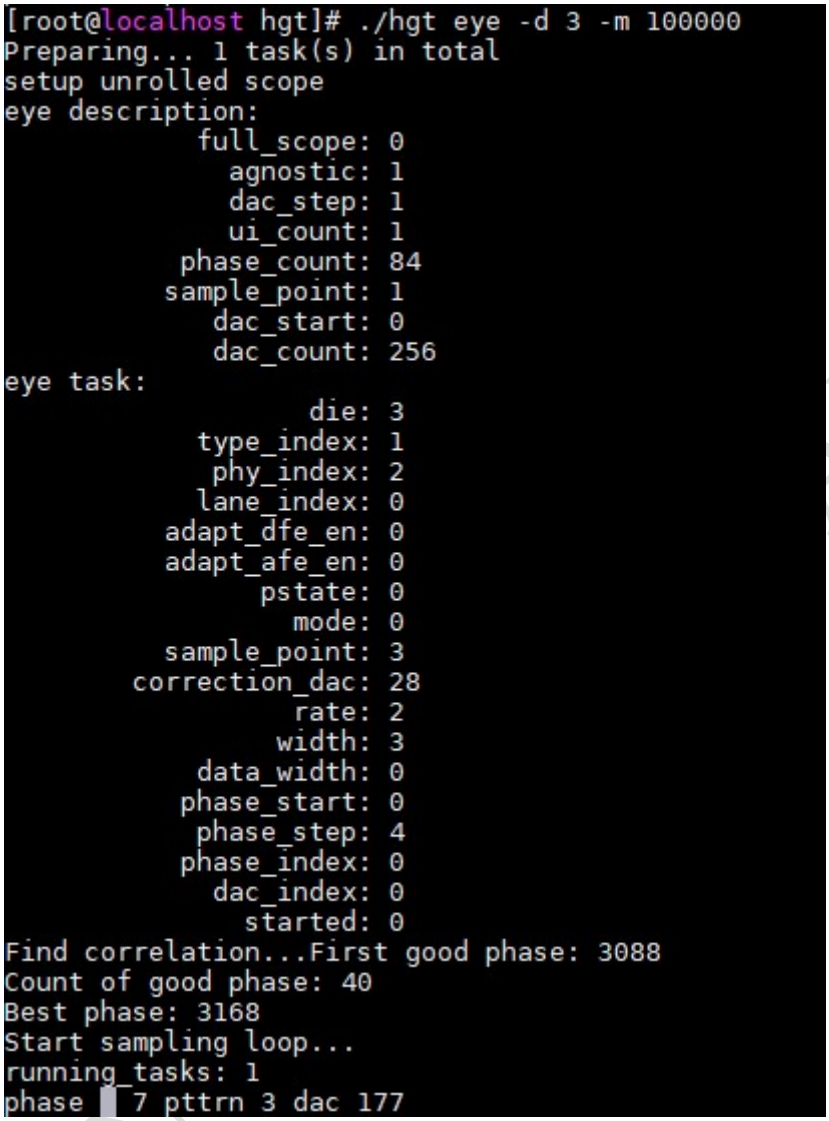

图 2 HGT 绘制眼图

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。

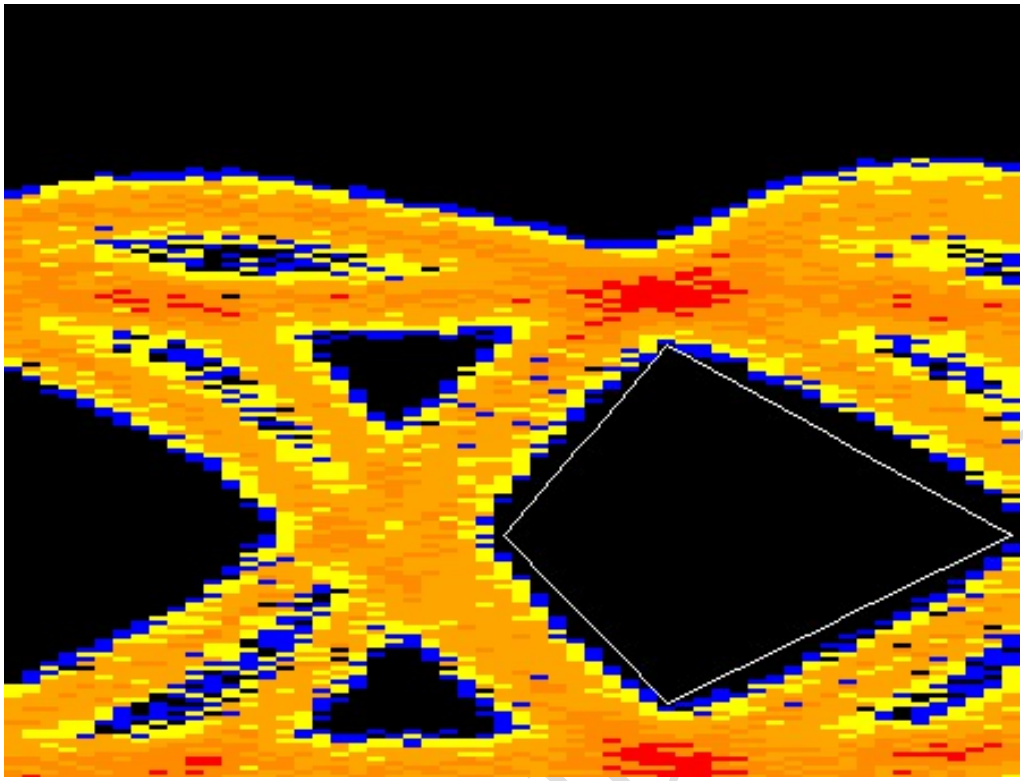

图 3 眼图示意图

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：

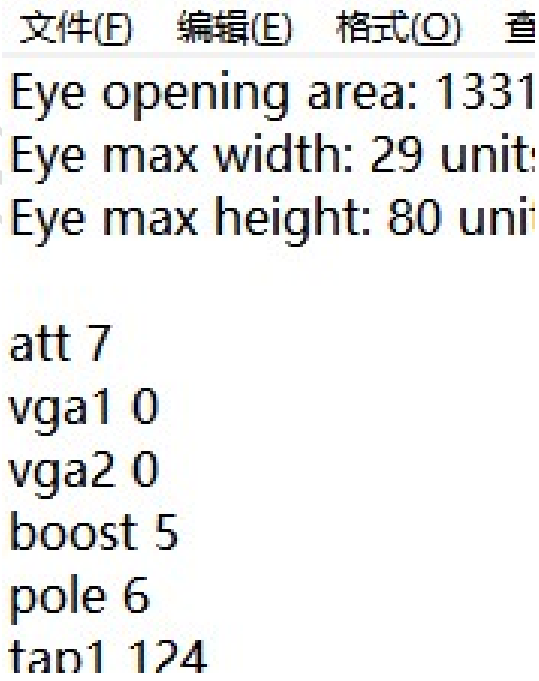

图 4 眼图测量数据

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：

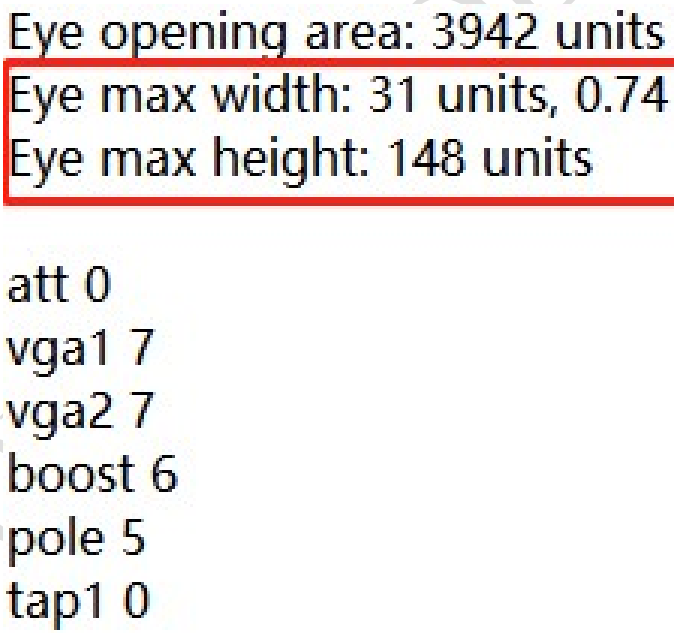

图 5 XGMI 测试举例

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（支持海光 2 号、海光 3 号未解锁 MP 芯片）
- 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. The absence of this option will result in auto pick preset
	-v vendor_id	specify device vendor id and device id.
	-g	just get current preset
	-S preset	just set current preset
	--speed=speedval	specify the speed

注意；

--presetlist 功能可选，如果不指定该参数则会自动挑选 presetlist 进行扫描。

-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。

--speed 功能可选，可以指定链路速度(只支持 GEN3/GEN4), 如果不指定以当前速度运行。指定速度会切换链路到对应的速度。

命令示例：

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

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。

4.2.1 功能说明

1. 获取当前 preset。命令示例：

./ hgt pcietxeqtune -s 5:0.0 –g

```
root@LabB0016:~/fengying/hgt# ./hgt pcietxeqtune -s 5:0.0 -g
current preset :
0 5 2 3 6 7 2 7
root@LabB0016:~/fengying/hgt# ./hgt pcietxeqtune -s 21:0.0 -g
current preset :
5 5 5 5 8 5 6 6 9 6 6 6 4 5 9 6
```

建议在做 preset 挑选前先使用该功能获取下当前的 auto 的 preset。挑选时指定的 presetlist 在获取的当前 preset 附近指定，可以减少掉卡和宕机的概率。例如：

当前所有 lane 的 preset 为“4 4 4 4 5 4 4 7”，可以指定 presetlist=“3 5”。

2. 设置 preset。命令示例：

./ hgt pcietxeqtune -s 41:0.0 –S 4

3. 挑选结果目录优化

现在挑选结果记录保存路径命名如下：

Bridge-桥的 BFD-Device-VDID-GENX。这么命名可以表征测试的槽位和卡类型以及速度。

```
Bridge-40:01.01-Deivce-0xc0101000-GEN3
Bridge-40:01.01-Deivce-0xc0101000-GEN4
```

4. Log 记录

执行的 Log 记录会自动保存在对应的挑选结果保存的目录里。

```
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000# ll
total 24
drwxr-xr-x  4 root root 4096 Jan 25 17:10 ./
drwxr-xr-x 11 root root 4096 Jan 25 17:01 ../
-rw-r--r--  1 root root 3133 Jan 25 17:10 2022-01-25-17-00-37-tx-eq-tune.log
-rw-r--r--  1 root root  145 Jan 25 17:10 HYGON_EQ_CFG
drwxr-xr-x  2 root root 4096 Jan 25 17:10 preset-7-eye/
drwxr-xr-x  2 root root 4096 Jan 25 17:05 preset-8-eye/
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000#
```

5. Continue 功能

现在每次跑完眼图测试都会统计保存眼图测试的结果（ew_average, ew_min）。

```
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000/preset-7-eye# ls
D2T1P0L0.bmp  D2T1P1L0.bmp  D2T1P2L0.bmp  D2T1P2L2.bmp  D2T1P3L0.bmp  D2T1P3L2.bmp  D2T1P4L0.bmp  D2T1P4L2.bmp  ew_average
D2T1P0L0.txt  D2T1P1L0.txt  D2T1P2L0.txt  D2T1P2L2.txt  D2T1P3L0.txt  D2T1P3L2.txt  D2T1P4L0.txt  D2T1P4L2.txt  ew_min
D2T1P0L1.bmp  D2T1P1L1.bmp  D2T1P2L1.bmp  D2T1P2L3.bmp  D2T1P3L1.bmp  D2T1P3L3.bmp  D2T1P4L1.bmp  D2T1P4L3.bmp
D2T1P0L1.txt  D2T1P1L1.txt  D2T1P2L1.txt  D2T1P2L3.txt  D2T1P3L1.txt  D2T1P3L3.txt  D2T1P4L1.txt  D2T1P4L3.txt
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000/preset-7-eye#
```

现在挑选 preset 时, 会先去找当前目录下是否存在对应的路径和保存的结果文件。

先检查是否存在对应的 Bridge-桥 BFD-Device-VDID 目录，然后检查该目录下是否存在测试指定 preset 对应的 preset-x-eye 目录, 再检查里面是否存在 ew_average, ew_min。如果都存在则直接利用上次跑过的测试结果进行挑选，节省跑眼图测试的时间。

```
root@LabB0016:~/fengying/hgt# ./hgt pcietxeqtune  -s 4c:0.0 --presetlist="7 8"
presetlist: 7 8
preset 7 test result exist, do not need run eye test
preset:7, ew_min:0.619718, ew_average:0.670775
current best preset:7
preset 8 test result exist, do not need run eye test
preset:8, ew_min:0.647887, ew_average:0.702465
current best preset:8
```

用户在运行时可能会遇到眼图 timeout 的错误。根据提示需要重启机器，然后再次去上一次执行的路径运行相同的命令。已经跑出的结果不会重复，会接着上次继续。

如果用户想重新对该 preset 跑眼图测试，则删掉对应的目录或者里面的结果文件，然后再运行。

6. 自动挑选功能（推荐）

现在运行时支持自动挑选模式。运行时不指定--presetlist 参数则运行在自动挑选模式。命令示例：

./ hgt pcietxeqtune -s 41:0.0

推荐用户使用自动挑选模式进行测试。该模式无需用户手动指定 presetlist 参数。避免指定到很差的 preset 值，导致掉卡,宕机或者眼图测试失败等。自动挑选模式会根据 auto 的 preset 值挑选出可能的最优 preset, 进行扫描。如当前所有 lane 的 preset 为“6 6 6 5 5 6 5 5 5 6 4 5 8 6 8 8”。自动挑选模式会挑选“5 6 9”进行扫描。

```
root@LabB0016:~/fengying/hgt# ./hgt pcietxeqtune  -s 41:0.0
presetlist is null, auto pick
6 6 6 5 5 6 5 5 5 6 4 5 8 6 8 8
presetlist: 5 6 9
before set preset:
6 6 6 5 5 6 5 5 5 6 4 5 8 6 8 8
set preset 5 ...
die 2 pcie_core 0 pcie_port 0 preset 5 16.0GT/s EQ pass
after set preset:
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

7. 指定速度功能

现在运行时支持指定速度，只支持 GEN3/GEN4。运行时不指定--speed 参数则默认以当前速度运行。命令示例:

./ hgt pcietxeqtune -s 5:0.0 –g -speed=3

```
root@LabB0016:~/fengying/hgt_src/hgt# ./hgt pcietxeqtune -s 5:0.0 -g --speed=3
current_link_speed:GEN3
current preset :
2 1 2 6 1 1 6 3
root@LabB0016:~/fengying/hgt_src/hgt# ./hgt pcietxeqtune -s 5:0.0 -g --speed=4
current_link_speed:GEN4
current preset :
7 7 7 7 7 7 7 7
```

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；

```
|                      \-00.1  Chengdu Higon IC Design Co.Ltd NTBCCP
+-[0000:40]-+-00.0  Chengdu Higon IC Design Co.Ltd Root Complex
|           +-01.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
|           +-01.1-[41-4b]----00.0-[42-4b]--+-00.0-[43-46]----00.0-[44-46]--+-10.0-[45]--
|           |                               |                                \-18.0-[46]--
|           |                               +-0c.0-[47-4a]----00.0-[48-4a]--+-14.0-[49]--
|           |                               |                                \-15.0-[4a]--
|           |                               \-1c.0-[4b]----00.0  LSI Logic / Symbios Logic Device c010
|           +-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-[4c-56]----00.0-[4d-56]--+-00.0-[4e-51]----00.0-[4f-51]--+-10.0-[50]--
|           |                               |                                \-18.0-[51]--
|           |                               +-0c.0-[52-55]----00.0-[53-55]--+-14.0-[54]--
|           |                               |                                \-15.0-[55]--
|           |                               \-1c.0-[56]----00.0  LSI Logic / Symbios Logic Device c010
|           +-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-[57]--+-00.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Function
|           |            \-00.2  Chengdu Higon IC Design Co.Ltd PSPCCP Command DMA Processor
|           +-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 4c:00.0

日志会记录 preset 挑选过程，用于验证挑选的 preset 是否正确。

```
********show pcie_list_t********
auto pick preset
7 7 6 7 6 9 6 5 9 7 7 7 6 8 8 6

presetlist: 7 8

before set preset:
7 7 6 7 6 9 6 5 9 7 7 7 6 8 8 6
set preset 7 ...
set GEN4 preset 7 pass
after set preset:
preset is fixed
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
preset:7, ew_min:0.647887, ew_average:0.670775
current best preset:7

before set preset:
preset is fixed
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
set preset 8 ...
set GEN4 preset 8 pass
after set preset:
preset is fixed
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
preset:8, ew_min:0.647887, ew_average:0.698944
current best preset:8
```

测试结果 HYGON_EQ_CFG 如下：

```
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000# ll
total 24
drwxr-xr-x  4 root root 4096 Jan 25 17:10 ./
drwxr-xr-x 11 root root 4096 Jan 25 17:01 ../
-rw-r--r--  1 root root 3133 Jan 25 17:10 2022-01-25-17-00-37-tx-eq-tune.log
-rw-r--r--  1 root root  145 Jan 25 17:10 HYGON_EQ_CFG
drwxr-xr-x  2 root root 4096 Jan 25 17:10 preset-7-eye/
drwxr-xr-x  2 root root 4096 Jan 25 17:05 preset-8-eye/
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000# cat HYGON_EQ_CFG
{FLAG_FORCE, pcie_gen4_ttx_force_otherside_preset, Cpu_Gen_Dhyana2n7, 2, 0xffffffff, 0x00000000, 16, 31, True, {8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8}}
```

- 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；

```
|                    \-00.1  Chengdu Higon IC Design Co.Ltd NTBCCP
+-[0000:40]-+-00.0  Chengdu Higon IC Design Co.Ltd Root Complex
|           +-01.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
|           +-01.1-[41-4b]----00.0-[42-4b]--+-00.0-[43-46]----00.0-[44-46]--+-10.0-[45]--
|           |                               |                                \-18.0-[46]--
|           |                               +-0c.0-[47-4a]----00.0-[48-4a]--+-14.0-[49]--
|           |                               |                                \-15.0-[4a]--
|           |                               \-1c.0-[4b]----00.0  LSI Logic / Symbios Logic Device c010
|           +-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-[4c-56]----00.0-[4d-56]--+-00.0-[4e-51]----00.0-[4f-51]--+-10.0-[50]--
|           |                               |                                \-18.0-[51]--
|           |                               +-0c.0-[52-55]----00.0-[53-55]--+-14.0-[54]--
|           |                               |                                \-15.0-[55]--
|           |                               \-1c.0-[56]----00.0  LSI Logic / Symbios Logic Device c010
|           +-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-[57]--+-00.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Function
|           |            \-00.2  Chengdu Higon IC Design Co.Ltd PSPCCP Command DMA Processor
|           +-08.0  Chengdu Higon IC Design Co.Ltd PCIe Dummy Host Bridge
```

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

```
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-c0101000# lspci -xxx -s 4c:0.0
4c:00.0 PCI bridge: LSI Logic / Symbios Logic Device c010 (rev b0)
00: 00 10 10 c0 47 05 10 00 b0 00 04 06 10 00 01 00
10: 00 00 00 a1 00 00 00 00 4c 4d 56 00 f1 01 00 00
20: 00 a2 00 a2 f1 ff 01 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 40 00 00 00 00 00 00 00 05 01 12 00
```

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

./hgt pcietxeqtune -s 4c:00.0 -v 0xc0101000

日志会记录 preset 挑选过程，用于验证挑选的 preset 是否正确。

```
root@LabB0016:~/fengying/hgt# ./hgt pcietxeqtune -s 4c:0.0 -v 0xc0101000
auto pick preset
preset is fixed
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

presetlist: 8 7

preset 8 test result exist, do not need run eye test
preset:8, ew_min:0.647887, ew_average:0.698944
current best preset:8

preset 7 test result exist, do not need run eye test
preset:7, ew_min:0.647887, ew_average:0.670775
current best preset:8
```

测试结果 HYGON_EQ_CFG 如下：

```
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000# ll
total 28
drwxr-xr-x  4 root root 4096 Jan 25 17:15 ./
drwxr-xr-x 11 root root 4096 Jan 25 17:01 ../
-rw-r--r--  1 root root 3133 Jan 25 17:10 2022-01-25-17-00-37-tx-eq-tune.log
-rw-r--r--  1 root root 2889 Jan 25 17:15 2022-01-25-17-15-47-tx-eq-tune.log
-rw-r--r--  1 root root  149 Jan 25 17:15 HYGON_EQ_CFG
drwxr-xr-x  2 root root 4096 Jan 25 17:10 preset-7-eye/
drwxr-xr-x  2 root root 4096 Jan 25 17:05 preset-8-eye/
root@LabB0016:~/fengying/hgt/Bridge-40:03.01-Deivce-0xc0101000# cat HYGON_EQ_CFG
{FLAG_SPECIFIED, pcie_gen4_ttx_force_otherside_preset, Cpu_Gen_Dhyana2n7, 2, 0xc0101000, 0x00000000, 16, 31, True, {8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8}}
```

- 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。

4.4.4 注意事项

1. 适用范围

   本工具并不适用于所有的 PCIe error。由于信号质量问题导致在系统启动中，启动完成后静置一段时间，出现 PCIe error，或者在压力测试过程中出现 PCIe error，可以使用本工具进行最优 preset 挑选。其他报错则需要根据问题现象具体分析。对于海光 2 号出现 PCIe error 时，建议先测试一下眼图，看看眼图是否正常。

2. 眼图超时

   运行过程中可能出现眼图超时的情况，如下图所示。可根据提示重启之后继续运行。

```
preset is fixed
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Preparing... 16 task(s) in total
eye description:
            full_scope: 0
              agnostic: 1
              dac_step: 1
              ui_count: 1
           phase_count: 71
          sample_point: 1
             dac_start: 0
             dac_count: 256
[ERROR] e25_scope_dac_search: ERROR:119 sample timeout
eye test fail, stop test. Please reboot and run the same command again
```

   具体参考 4.2.1 中的 continue 功能。

4.5 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
        )
    ),
```