PCIe 通义万问

系列文档之（一）

发现 + 探讨 + 解决 + 记录 + 启发

MangoPapa

2024.10.06

前言

《PCIe 通义万问》系列文档旨在记录 PCIe 相关行业工程师们在 PCIe 协议学习、IP 设计验证、产品测试及使用过程中遇到的问题、迸发的思考、进行的探讨及可能的解决方案，以期给遇到相同相似问题的同行们些许启发。文档涉及的技术方向主要为 PCIe，也包含 PCIe 相关的 CXL、UCIe 及计算机体系结构相关内容。

本文档是《PCIe 通义万问》系列文档之（一），含 999 个问题。问题内容有以下三个来源：MangoPapa（下称博主）的“PCIe 每日一问一答”系列专栏；博主的博文留言及私信讨论；博主作为群主的“PCIe 技术交流群”群聊内容。MangoPapa 小助理、折叠、先杰、CR 小队长、kangling 共同完成了问答内容提取与归纳整理，噫嘘唏及慕荷负责文辞优化及编辑排版，上述人员及 MangoPapa、JasonW、皮塞阿姨、席可儿参与了文档内容的审校。感谢以上人员的无私奉献。

本文档仅对讨论内容进行总结记录，不保证问题答案的正确性，部分问题只提供解决思路或讨论过程。术业有专攻，文内定有偏颇。如您发现谬误，一敬希原宥，二望乞点拨；如您持异见，欢迎来信讨论。如有需要，欢迎联系 MangoPapa 加入相关技术交流群进行交流讨论。我们不惧山呼海啸的批评，只恐心血落地后的寂然无声。

接下来，本文档及文档内容将交由皮塞工坊（PCI Engineers’ Club）进行维护。为打通 MangoPapa 的多个技术群聊及博客平台，由皮塞工坊相关人员对各渠道日常沟通内容进行关键信息提取，并同步在皮塞工坊知识星球，欢迎各位加入围观讨论。本系列文档相关收益用于皮塞工坊日常运营、激励活跃用户及群友节日福利。

找 MangoPapa 聊聊

进 PCI Engineers’ Club 瞧瞧

目录

前言 **1**

目录 **2**

一、拓扑结构 **34**

拓扑组件 34

1. PCIe RC/EP/RCiEP 是什么？ 34
2. EP 是不是没有 Downstream，只有 Upstream？ 34
3. Native EP 和 Legacy EP 是什么？ 34
4. RC 内可以有多个 PF 吗？ 34
5. RCiEP 该怎么理解？ 35
6. PCIe 伪端口（Pseudo-Port）是什么？ 35
7. M-PCIe 是什么？ 35
8. 对于 NUMA Chip，通常是每个 Node 对应一个 RC 吗？ 35

HIERARCHY ID 35

9. 什么是 PCIe Hierarchy？ 35
10. 为何需要 PCIe Hierarchy ID？ 35
11. 在支持 HID 的系统中，所有 TLP 均会带有 HID 吗？ 36
12. Hierarchy ID 的实际意义是什么？是否相当于做了 NTB Msg Register 的事情？ 36

SWITCH 37

13. PCIe Switch 有哪几种传输模式？ 37
14. 挂在同一 Switch 下的 EP 设备可以相互直接通信吗？ 37
15. 现在国产 PCIe Switch 有哪几家，3.0 的成熟吗？ 38
16. Switch 在什么场景下会主动给下面的 EP 发 MWr？ 38
17. Switch 上只有一个 Host 时为什么会出现 Cfg 拦截？ 38
18. 博通 PCIe Gen5 Switch 的 Synthetic Mode 是什么？ 38
19. 如果 Switch 误开了 Synthetic Mode，使用过程中 BAR 会被 Switch 改掉吗？配置空间是否会被 Switch 内的 CPU 动态调整？ 38
20. PCIe Switch 是如何分配每个 DSP 占总 Lane 数中多少 Lane 的？ 39
21. 一个 8 Lane Switch 下可以挂单个 x8 设备，也可以挂两个 x4 设备。挂单个 x8 时，两个 x4 的 LTSSM 都会参与链路训练吗？ 39
22. 除了 BDF 编号的限制外，Switch 扩展设备的数量还有什么其他的限制吗？ 39

BRIDGE 39

23. lspci -tv 扫描出一些 AMD 的 Dummy Host Bridge，这些 Dummy Host Bridge 是什么？ 39
24. 在集成了 PCIe 转 PCI 桥的卡中，桥下接了 Device Function ID 不同的多个设备，但桥的上游不支持 ARI 功能，此时 BDF 是怎么转换的？ 39
25. NTB 是什么? 40
26. Ice Lake CPU 同一 RC 发出请求中的 RID 随机变化或导致 NTB 失效，对于该问题 BRCM 有解决方案吗？ 40
27. 上述设计导致向外部暴露多个 ID，为何 Ice Lake 会有上述设计？ 40

RETIMER 40

28. PCIe Retimer 是什么？ 40
29. 如何判断 PCIe 系统是否要用 Retimer? 41
30. RC 和 EP 如何知道它们中间有 Retimer ? 41
31. 对于 CPU-Device 之间存在 Retimer 的连接场景，建链过程的 TS 码流在 Retimer 中直接转发的还是 Retimer 重新生成的？EIEOS 呢？ 41
32. 如果链路中间有 Retimer 做了 Skip Remove，Rx PIPE 做 128/130 解码时如何保证一个 Block 内不会把非 Skip 的数据包进来？ 42
P2P 42
33. 什么场景下会用到 P2P 功能？ 42
34. PCIe 的 P2P 是指是同一个 Switch 下的两个设备，还是不同 Switch 下的两个设备通信？ 42
35. 请问飞腾平台能在两个 RP 下的 EP 之间进行 P2P 通信吗？ 42
36. P2P 一般支持 Snoop-able 的访问吗？ 43
37. P2P 可以发 RO=1 和 IDO=1 吗？ 43
38. 请问一个 EP 想与另一个 EP 通信，其如何获知对方的 BDF？ 43
39. 已知服务器支持 PCIe Switch 下的 P2P 通信，如何在工程实践上去验证 P2P 功能呢？ 43
40. P2P 对 EP 设备有什么特殊要求吗？ 43
41. 在 Switch 未开启 P2P 功能的前提下，Switch 收到 EP 发来地址落入 Mem Base/Limit 范围内的 Mem 请求，是不是不应该转发而是直接反馈 CpI.UR？ 43
42. Switch 开了 ACS 重定向之后 RC 端还可以支持 P2P 吗？ 44
BIFURCATION 44
43. PCIe Bifurcation 有什么优势吗？ 44
44. 一个 PCIe Controller 可以连接多个 PHY 吗？ 44
45. 同一个 PCIe Controller 可以支持多个 Link 吗？有没有存在 2 个 x4 的 Link 公用一个控制器来替代 1 个 x8 的 Link？ 44
46. 同一插槽下支持多种配置，多种配置是对应多个 EP 设备吗？ 44
47. Host 端 BIOS 把一个 PCIe x16 槽位改成了两个 x8，EP 端是 x16 的，能降 x 识别成 x8 吗？EP 需要支持分叉配置吗？ 44
48. EP 支持 Bifurcation 吗？如果 EP 支持 Bifurcation 是不是也需要多个 EP Controller？ 45
49. Synopsys 公司的 2 Lane 的 Controller+2 Lane 的 PHY 可以接 2 个 x1 的 Device 吗？ 45
50. Cadence 公司的 2 Lane 的 Controller+2 Lane 的 PHY 可以接 2 个 x1 的 Device 吗？ 45
51. 比如一个 x16 和一个 x8 可以工作在单 x16 或 2x8 模式，怎么通过 Bifurcation Mux 跟 PHY 的 PIPE 接起来？ 45
CROSSLINK 45
52. PCIe Crosslink 是什么？ 45
53. 现在会用 Crosslink 这个功能吗？ 45
GPU DIRECT 46
54. GPU Direct 是什么？ 46
55. GPU Direct RDMA 中是如何使用 PCIe 的？ 46

二、传输层 48

TLP 分类 48
56. PCIe TLP Type 有哪些？ 48
57. PCIe Memory 访问事务包分为哪几种？ 48
58. PCIe IO 访问事务包分为哪几种？ 48

59. PCIe Configuration 访问事务包分为哪几种？ 49
60. PCIe 转发、非转发事务是什么？ 49
61. CPL 状态有哪几种？ 49
TLP 组包 49
TLP 字段 49
62. PCIe TLP 包格式是怎样的？ 49
63. PCIe TLP Header 格式是怎样的？ 50
64. PCIe TLP Header 支持哪些 Format? 50
65. PCIe TLP Header 中有哪些 Fields？ 51
66. PCIe TLP Header Base Size 取决于什么？ 54
67. PCIe TLP Header 中的 Length 有何意义？ 55
68. PCIe TLP Header 中的 Transaction ID 是什么？ 55
69. PCIe TLP Header 中的 Lower Address 有何意义？ 55
70. PCIe TLP Header 中的 Byte Count 有何意义？ 55
71. PCIe TLP Header 中的 Byte Count Modified 有何意义？ 55
72. PCIe TLP Length 只有 10 bit 位宽，该如何表示 1024 DW 长度的 Data Payload？ 56
73. PCIe MWr/MRd TLP 中的地址必须 4 B 对齐吗？如何发出非 4 B 对齐的请求？ 56
74. CplD 的 TC 字段需要跟 Req 中的 TC 保持一致吗？ 56
75. PCIe Completion 中的 EP 和 Cpl.Status 之间有什么联系吗？EP=1 时一般后者会是什么值？ 56
76. First DW BE、Last DW BE 是什么？ 56
77. PCIe Zero Write、Zero Read 是什么？ 57
78. 如何知道 TLP Prefix 的长度？ 57
79. PCIe 6.0 OHC-A5 中的 Segment 是什么？ 57
80. PCIe 6.0 的 TLP 里面新增的 Segment 域段是如何分配的？这个 Segment 是 8 bit 的，和 16 bit 的 Hierarchy ID 有何关系？ 58
81. Memory 事务中的 Requester ID 有什么作用？ 58
82. Completer ID、Requester ID 与 BDF 的关系是什么？ 58
TLP 路由 58
83. PCIe 有哪几种路由方式？ 58
84. Message 是如何路由的？ 59
TLP 包处理 59
收发处理 59
85. 几种不同的 TLP 同时要发送时候，该怎么发？ 59
86. 多端口按时间窗口循环发送，既定时间窗口内发不完当前的 TLP 包会怎么办？ 60
87. IO Write 是不是只能由 RC 发出不能有 EP 发出？ 60
88. TLP 请求可以跨越 4 KB 地址边界吗？ 60
89. RC 收到 EP 发来的 Cfg 包后会回复什么？ 60
90. PCIe 发包有什么能控制到 Memory Type 吗？比如 Device-GRE Device-NGRE 等等。 60
91. MRd、MWr TLP 中是否需要同时指定 RID 和对方的 ID？ 61
92. NP 请求需要 Requester ID，那么 MWr 等 Posted 请求中 Requester ID 有用吗？ 61
93. Requester ID 不属于该设备的 Subordinate Bus 会不会被拒收？ 62
94. 收到不支持的 Message 是如何处理的？ 62
95. 当同一 Requester 有多笔 MRd 发出时，加之 Flow Control 中 MRd、CplD 不保序，如何知道 CplD 回的是哪笔 MRd 的数据？ 62

96. 开启 AER Error Reporting 之后，如果 EP 收到了一个 Type 1 类型的 Config 包，EP 是不是既要回复一个 UR 包又要发生 Error Message？ 62
RCB/MPS/MRRS 62
97. MRRS 是什么？跟 MPS 有什么区别？ 62
98. MPS 和 MRRS 有什么关系吗？ 63
99. MRRS 最大读请求 Size 是对自己的要求还是对链路对端的要求？ 63
100. 已有 MPS 来限制包的大小了，为什么还需要 RCB 来拆分 CplD 呢？ 63
101. 为什么 EP 的配置空间内也有 RCB Field？ 63
102. PCIe 协议对 MRd 请求返回的 CplD 数量有限制吗？ 63
103. PCIe IP 内部的 Buffer 深度对 MPS 有影响吗？MPS 设置有何讲究？ 64
104. MPS 的值对性能有影响吗？ 64
105. 相比于 RCB=64，把 RCB 设置为 128 是否效率更高？ 64
106. MPS 怎么设置，和 LCRC/ECRC 有关系吗？ 64
107. 链路上不同节点的 MPS 不同怎么办？ 64
108. MPS/MRRS 和 Byte Enable 有关系吗？拆包时是否需要综合考虑两者？ 65
109. 如果 Requester 发出的 MRd 超 MRRS，Completer 收到后一般怎么处理？回 UR？ 65
110. RC 的 RCB 可配吗？RC 返回包给 EP 的时候，能不能固定拆分方式呢？ 65
111. RC 和 EP 的 MRRS 不同时，应该遵从谁？ 65
112. CplD 拆包必须要满足 RCB 的规则吗？是拆包按 RCB 对齐还是按 RCB 的 N 倍？ 66
113. 回 CplD 拆包时，具体按 RCB 的 1 倍还是 N 倍，是有寄存器去设置吗？ 66
114. 当 MRd 请求超过 MRRS 时，是否可以先改大 MRRS 再一次发出 MRd？ 66
115. PCIe 回 CplD 时单个 CplD 的 Payload Length 是不是也不能超过 MPS？ 66
116. Rx 端会检查接收到的 MRd 中的 Length 是否超没超自己 MRRS 吗？ 66
117. 在主机 Linux 上使用 devmem2 工具给设备发送 MRd，默认读长度只有 2 DW，是因为受 64b 操作系统影响吗？ 66
118. PCIe MWr 有要求跨 256 B 地址边界拆分 TLP 吗？ 67
119. PCIe RC 的 Max_Payload_Supported 一般设置为多大？ 67
120. CplD 可以按照 RCB 和 MPS 进行拆包，一般以哪个为准？ 67
121. MTU 一般设置多大？设置为 MAX 吗？ 67
122. MRRS 是怎么设置的？ 67
CA/UR 68
123. 跨主机访问时 MWr 正常，但 MRd 回 UR。该地址在本机读正常，另一主机过 Switch 来访问本机该地址写正常，读全 F 且 UR，有何定位思路？ 68
124. 请问 UR 类型的 Completion 可以是 CplD 吗？还是说只能是 Cpl？ 68
虚通道/服务质量 69
125. 一般需要开几个 VC Channel？ 69
126. 哪些场景会用到多 VC（Virtual Channel）？ 69
127. PCIe 的多 VC 功能的使用情况如何？ 69
128. PCIe（RC）上来的包有个 AXI QoS Field，这个一般固定给什么值？ 69
129. PCIe 的 VC/TC Mapping 字段和 VC ID 字段需要在 VC Enable 之前提前配好吗？ 69
事务排序 70
Tag 70
130. PCIe TLP Header 中的 Tag 应该怎么理解呢？连续发起 MRd 是否会改变 Tag 的值？ 70
131. 10 bit Tag 为何从 256 开始，为何不和 8 bit 一样从 0 开始？ 70

132. 为何 10 bit Tag 是 256 到 768，而不是跟 8 bit 重用/兼容从 0 开始？ 70
133. 为什么 10 bit 的 Tag 只能用 768 个？ 不应该是 1024 个吗？ 70
134. TLP 中的 Tag 能不能选择 512？ 71
135. PCIe 的同一个 Tag 需要间隔多久才能重复使用？ 71
排序规则 71
136. PCIe 排序规则中 NP 的 Cpl 为什么一定要等前面的 P 都要完成才能返回？ 71
137. PCIe 排序规则中的"Y/N"意思是可以超过、也可以不超过吗？ 72
138. RO=1 的 TLP 可以超过任何 TLP，那么 RO=0 的 TLP 可以超过 RO=1 的 TLP 吗？ 72
139. 不同 ID 的 TLP 排序需要考虑 TLP 类型吗？ 72
140. PCIe 的 SO 强排序在系统级能保证到 DDR 入口也是 SO 的吗？ 72
141. 为什么 Posted 请求在阻塞情况下要超车 Non Posted 请求，这两种请求有关联吗？为什么会造成死锁？ 73
142. 排序时，Posted 包超越 Non-Posted 包的位置是在 VC Buffer 中超的吗？ 73
143. NP Read 之间可以相互超越不保序，其保不保序跟 Relaxed Ordering 有关吗？ 73
144. EP 收到一笔 MRd 后分多笔 CplD 进行回复，同一 MRd 对应的 CplD 采用相同 Tag，不同 MRd 对应的 CplD 可以交叉回吗？ 73
145. EP 发出的两个读请求，后面的会不会超过前面的？ 74
146. PCIe 链路上相同地址的读请求之间会保序吗？ 74
排序应用 74
147. DMA 乱序需要处理吗？ 74
148. EP 发出的 Mem 请求可以直接桥接到 CMN 上，然后通过 TLP 带的原始地址来路由到同样挂接在 CMN 上的目标模块吗？还是说 TLP 里面带的地址不能直接用，需要先改一下？ 74
149. 可以通过设置 Slave 端相同的 AXI ID 来实现两笔 MRd 的保序吗？ 74
150. PCIe Controller 入口和出口 AXI Bus 接口处是如何保序的？ 74
数据完整性 75
ECRC 75
151. TLP Digest 是什么？ 75
Error Poison 75
152. Poisoned TLP 是什么？是干什么的？ 75
153. 上游设备给下游 PCIe 设备发送的数据中毒 TLP 包头中的 EP bit 是谁置 1 的？ 76
154. EP 收到的 Poisoned TLP 后一般怎么处理？MWr 直接丢掉、MRd 回 UR 吗？ 76
完成超时 76
155. Completion Timeout 机制是什么？ 76
156. Completion Timeout 的值为何是一个范围，而不是一个固定的值？ 76
157. 关闭 Cpl Timeout 后 DMA 发出 MRd 但收不到 Cpl 会有什么结果？会重新训练吗？ 77
158. Host 发一个读请求到 EP，EP 挂了没回 CplD 导致超时，Host 超时会挂么？ 77
159. Synopsys PCIe IP 在 FPGA 上作为 RC 扫描桥下设备，RC 能扫到桥内的 USP 及 DSP，但扫描不到桥下的 EP，RC 报 Cfg Timeout，可能是什么原因？ 77
160. Host 遇到 Completion Timeout 错误时一般是什么反应？ 77
161. 请问 Xilinx 的 IP 完成包超时会不会造成不可恢复的错误？ 77
162. MRd 的 Completion Timeout 所用的计数器是软件维护还是硬件维护？ 78

三、数据链路层 79

基本介绍 79

163. PCIe 数据链路层是做什么的？有哪些功能？ 79
DLLP 包与包处理 79
164. DLLP 是什么？ 79
165. DLLP 分为哪几类？ 79
166. NOP DLLP 有什么功能？ 80
167. NOP DLLP 是什么？有何作用？什么时候发 NOP DLLP? 收到后该如何处理？ 80
168. Data Link Feature DLLP 是什么？有何作用？什么时候发？ 80
169. Link Management DLLP 是做什么的？ 81
170. PCIe 6.0 规定的 DLLP Payload 有哪几种？ 81
171. PCIe Sequence Number 是什么？有什么用处？ 81
172. 几种不同的 DLLP 同时要发送时候，该怎么发？ 82
173. PCIe DLLP Blocking 是什么？ 82
DL 状态机 82
174. 数据链路控制状态机是做什么的？ 82
175. DLCMSM 跟 LTSSM 有什么关系？ 83
176. DL_Feature 期间要交换哪些 Feature? 83
流量控制 84
177. PCIe Flow Control 是什么？ 84
178. PCIe Flow Control 用到了哪些 DLLP? 84
179. Flow Control 初始化流程是怎样的？ 84
180. PCIe 流控初始信用量可配吗？ 85
181. InitFC1 和 InitFC2 有何区别？ 85
182. PCIe Spec 有规定 InitFC DLLP 发送的频率吗？ 85
183. 流控 Scale Factor 的值是在固件配置的吗？ 85
184. 为什么 TLP Header 和 Body 要分开做流控？这样要分开方到两个 Buffer 里，岂不是增加了复杂性？其优势在哪？ 86
185. Mindshare PCIe Technology 流控指针计算时 E8h 取补码为何是 11000b 而非 10011000b？ 86
186. PCIe 流控有 N123 算法 N123+算法和 N23 算法，不同节点如果选择不同的效率可能会大受影响，这种情况是否需要添加额外的逻辑来加以协调呢？还是接受浪费？ 86
187. 流控机制要求剩余缓冲数量小于最大缓冲数的一半才能发包，为什么要求一半？ 87
188. FC Credit Timeout 后会发生什么？PHY 重新回到 Detect 重新训练，还是 DL 层 Link Down 但 LTSSM 保持在 L0？ 87
189. 启用了 Scaled Flow Control 后，是不是 Update DLLP 传输的实际 Credits 数量必须是 Factor 的倍数？ 87
190. 完成状态为 UR 或 CA 的 Cpl 包会占用一个 Cpl Header 信用量吗？ 87
191. EP 可以发出超过其 Cpl 信用量的 MRd 请求吗？ 87
192. 发送给对端的 IO 事务，如果对端不接受并回了 UR，会消耗信用量吗？ 88
193. 什么是 Scaled Flow Control? 跟 Flow Control 区别是什么？ 88
194. Optimized_Update_FC 是做什么的？跟常规 UpdateFC 有何区别？ 88
数据完整性 88
195. DLLP 自身是如何保证其数据完整性的？ 88
196. LCRC 和 ECRC 有何区别？ 89
197. PCIe Gen6 中提到的 FBER 是指什么误码率？ 89
198. Cfg 事务和 IO 事务是不是也可以开启 ECRC？ 89

重传机制 89
199. 发生 DLLP Timeout 会导致链路 Retrain 吗？ 89
200. PCIe EP 发送注入 Parity Error 的 TLP 给 Host 后理应会触发重传，如何在 EP 端确认是否发生了重传？ 89
201. Requester 发送 Req TLP 之前需要确认 Completer 有 Credit，Completer 给 Requester 反馈 Credit 时需要 Requester ID，这不是死锁了吗？ 90
202. PCIe 6.0 Retry 机制，越看越迷糊怎么办？ 90
203. IMPLICIT_RX_FLIT_SEQ_NUM 该怎么理解？ 90
204. Rx 端为什么自己要有一个 Retry Buffer？ 90
205. Nak/Ack Flit 里面的应该 TLP 填充什么？ 90
206. 每收到一个 MRd 都要回复一个 Ack DLLP 吗？ 90
207. ACK/NAK 机制是什么？ 91
208. 每个 TLP 都要回复 Ack 吗？Ack 要多久发一次？ 91
209. 什么情况下会发送 Nak DLLP? 92
210. TLP Tx 收到 Nak 后如何处理？ 92
211. 连续多次重传失败后如何处理？ 92
212. Retry 机制中用到的 Tx Retry Buffer Size 是如何计算的？ 92
213. PCIe Flit Mode 时，Selective Nak 是什么？ 92
214. PCIe 6.0 Retry 机制中，Implicit Flit Sequence Number 是做什么的？ 93

四、逻辑物理层 94

包与包处理 94
TS 94
215. TS1 Symbol 6 中，为什么在 2.5 GT/s 及 5 GT/s 的条件下包含 8 GT/s 和 32 GT/s 的描述？ 94
216. TS2 Symbol 6 2.5 or 5.0 GT/s 条件下又包含 8.0/32.0 GT/s 的条件，这该如何理解？ 94
217. Flit 有类似 STP 这样的 Token 吗？接收端如何识别 Flit 起始位置？ 94
218. PCIe TS1/TS2 是如何实现 DC Balance 的？ 94
219. 请教一下 PCIe Gen5 协议物理层逻辑子层 TS1 Ordered Set 的使用问题：TS1 里面的 Symbol 6，用在 2.5 GT/s 和 5 GT/s 速率的时红框和蓝框的两个配置分支的用途该如何解释？ 95
220. Modified TS1/TS2 是什么？Modify 了什么？为什么要 Modify？ 96
Order Set 96
221. PCIe 有序集有哪几种？ 96
222. 128b/130b 时 SKPOS 长度不定，发送端是如何补齐 1 个 OS Block 的？ 96
223. 128b/130b 时 SKPOS 长度不定，接收侧如何确定 SKPOS 的结束？ 96
224. PCIe 有序集什么时候加扰，什么时候不加扰？ 96
225. 为什么下图中的 SKPOS 未遵守 128/130 编码？ 97
226. PCIe SDS/EDS 是什么？ 97
227. EIOS 截断是什么？ 98
228. Logic Idle Symbol 与 Idle Symbol (IDL) 是一样的吗？与 EIOS 有何关系？ 98
229. 为什么在 2.5、8、16、32 GT/s 时 EIOSQ 由 1 个 EIOS 组成，而 5 GT/s 时却需要 2 个? 98
230. STP 中的 FCRC 和 FP 都是干什么的？ 99
231. 如果发送的 FTS 不够，导致从 L0s 回退到 L0 失败了怎么办？ 99
Token 99

232. 在 Gen3 以后，STP Token 中的 TLP Sequence Number 和 TLP Length 是数据链路层产生的，还是物理层产生的？ 99
233. PCIe 设备给主机发 EDB Token 可能是什么原因导致的？ 99
234. PCIe Ordered Set 和 Framing Token 有何区别？ 100
Logic Idle 100
235. Logical Idle 未同时出现在所有有效 Lane 上是否违反协议？ 100
236. PCIe 是如何判断链路处于电气闲的？ 100
包的收发处理 101
237. Multiplexer 和 De-multiplexer 有何作用？ 101
238. Byte Striping/Un-striping 是做什么的？ 101
239. Packet Filtering 是做什么的？ 102
240. TLP、OS、DLLP 同时要发送时，该怎么发？ 102
241. Block Alignment 块对齐是什么？ 103
242. 位锁定是什么? 位有什么好锁定的？ 103
243. PCIe 是如何实现 Block Type 检测的？ 103
244. PCIe 是如何检测块对齐丢失的？ 103
245. PCIe Lane-Lane Skew 产生的原因及 De-skew 方法？ 104
246. 128b/130b 编码发生 TLP 时只有 STP 没有 END，如何知道 TLP 结束？ 104
247. PIPE 口上观测到了 TS1/TS2，为什么 PCIe Controller 内没作统计？ 105
编解码/加扰 105
248. PCIe 预编码（Precoding）是干什么的？ 105
249. PCIe 的物理层的 Precoding 有什么作用？ 106
250. Precoding 可以只在 USP 开启，不应该在 DSP 和 USP 同时开启吗？ 106
251. 128b/130b 编码是如何做到 DC Balance 的？ 106
252. 128b/130b 编码时 Sync Header 和 TS Symbol 0 不做 Scrambling 不会影响 DC Balance 吗？ 106
253. 128b/130b 编码时终止数据流传输的 EDS 为 Framing Token，为什么启动数据流传输的 SDS 却要是有序集的形式？ 106
254. Gen3~5 速率时候采用 128b/130b 编码，是如何保证码流没有连续长 0 或者长 1 呢？单靠加扰能避免吗？ 106
255. PCIe Tx Scramble 放在 Encode 之前之后？为什么？ 107
256. PCIe Scrambling 中有哪两种 LFSR 方案? 107
257. 扰码种子如何传递到对端的？ 108
258. 8b/10b 编码已经能均衡 0 和 1，为何还需要加扰？ 108
259. 为什么要关闭加扰 Scrambling？ 108
260. PCIe 加扰解扰时，什么情况下会初始化 LFSR ？ 108
时钟 108
261. PCIe 参考时钟为何都是 100 MHz？ 108
262. PCIe 的 100 MHz 参考时钟（refclk）是用来帮助 CDR 锁定嵌入在数据流中的时钟的吗？物理层工作的时钟也是这个参考时钟提供的吗？ 108
263. PCIe 有哪几种参考时钟架构？ 109
264. CC 时钟模式时收发端有频差吗？为什么 CC 时还需要 Elastic Buffer？ 109
265. SSC 有什么用？ 109
266. CC 时钟模式可以开 PHY 内部的 SSC 吗，还是说一定要接外部的 SSC？ 109

267. CC 时钟架构下 SSC 可以不加在 Common refclk 上、加在 PHY Internal Clock 上吗？ 110
268. 如何判断 refclk 有没有开 SSC？ 110
269. SSC 扩频时钟和 PCIe 插 SKP 之间有何关系？ 110
270. Host 开启 SSC 后，EP 需要同步开启 SSC 吗？ 110
271. PCIe 扩频前后时钟频率变化了吗？ 110
272. Common Clock 同源时钟模式下，Gen4 PCIe Training 完成之后还可以开展频吗？ 110
273. PCIe Link 上的 RC 和 EP 采用非同源时钟，其中 RC 端启用 SSC 扩频后仍无法 Link 是何原因？是否需要同步打开 PCIe Controller 的 SRIS？ 110
274. 在 Link Up 后再开 SSC 是否合规？如何证明是由于该卡容错性差导致的？ 111
275. PCIe RC 的 refclk 采用内部时钟模式，PHY 输出 100 M 差分时钟，50 Ω 端接是放到 RC 主板的连接器端、还是放在 EP 设备端？ 111
276. 主机 BIOS 里面的 Spread Spectrum 就是 PCIe 的 SSC 吗？ 111
277. 如何证明 BIOS 里的 Spread Spectrum 就是 PCIe 的 SSC，而不是其他地方的？ 111
278. 一般主板里除了 PCIe 哪里还需要 SSC？ 112
279. 如何确认 EP 对接服务器的某个卡槽上 SSC 是否开启呢？ 112
280. 板子提供参考时钟时 PCIe 内还需要 CDR 吗？ 112
281. Synopsys 的 PHY 的 ATE 测试 JTAG Clock 的最大频率是多少？ 112
弹性缓存 112
282. PCIe Elastic Buffer 弹性缓存是做什么的？ 112
283. PCIe 6.0 没有 Elastic Buffer 了，那么是如何做时钟恢复的？如何解决的频差问题？ 112
284. Elastic Buffer 中的 Nominal Empty Mode 大概是什么原理？ 113
285. 为什么弹性 Buff 深度加大，数据传输路径延时会增加？ 113
286. 链路存在 Deskew Error，可以通过增加 Elastic Buffer 深度来解决吗？ 113
链路训练 114
协议理解 114
287. PCIe Link Training 期间都训练了什么？ 114
288. PCIe Gen2 也需要从 Gen1 切速吗？能不能直接进 Gen2？ 114
289. PCIe Link Training 时，双方协商是从 Gen1 直接往最大速率跳转吗？ 114
290. 请教 PCIe LTSSM Detect.Quiet 的理解： 114
291. LTSSM 中的 Polling.Active 时需要发 1024 个 TS1，这些 TS1 是用来 PLL 恢复时钟的吗？ 115
292. Polling.Active 中两端的 1024 个 TS 有什么发送顺序吗？ 115
293. Polling.Active 因超时跳转到 Polling.Configuration 条件中，是说收到第一个 TS1 时，已经发完 1024 个 TS1，此后收到 8 个 TS1 或者 TS2，跳转到 Polling.Configuration 吗？ 115
294. 上述 Polling.Active 跳到 Polling.Configuration 条件中，可以收到 TS1 或 TS2，Polling.Active 状态下不是发 TS1 吗，为什么会收到 TS2？ 115
295. PCIe 必须支持 PN 极性翻转吗？ 116
296. 链路训练到 L0 后还能进行链路反转和极性反转吗？ 116
297. Polling.Compliance 状态是做什么的？Compliance Pattern 是干什么的？ 116
298. 链路训练过程中为什么在 Cfg.LaneNum.Wait、Cfg.LaneNum.Accept 和 Cfg.Complete 这三个阶段发 Modified TS1 序列而非标准的 TS1 呢？ 116
299. Downstream Port 中的默认 Link Number 是否为寄存器可配？有哪些设置规则吗？ 117
300. 状态转移时是否需要判断 TS 中的 Link Number 是否跟 Configuration 阶段协商的 Link Number 一致？ 117
301. Bifurcation 时两个 DSP 的默认 Link Number 可以设置为相同值吗？ 117

302. 如果链路中某条 Lane 一直跑不到 2.5 GT/s，链路训练过程状态机会卡在什么状态? 117
303. 链路建立完成 Lane 数协商之后，还可以写 RP 的配置寄存器实现 Lane 数动态调整吗？ 118
304. 链路训练时为什么有时发 Modified TS 有时发 Standard TS？可以固定发 Standard TS 吗？ 118
305. Tx 在发数据时，Rx 有 Ack 和流控等 DLLP 在传送，为何此时 Rx 可以处于 L0s？ 118
306. PCIe 5.0 规范支持 x32 吗？NV 的卡似乎支持 x32。 118
307. 有哪个应用场景会用到 x12 吗？ 118
308. PIPE 接口中 RxElecIdle 拉低可以理解为 FPGA 检测到吗？ 119
309. EP 寄存器配了 Link Width 后还能自动识别对端有几条 Lane 并做切换吗？ 119
310. Polling.Active 阶段部分 Lane 上收不到 TS，不应该进 Polling.Configuration 降 Lane 吗？ 119
311. 一个 Port 下可以有多条 Link 吗？ 119
312. Switch 上 4x2 的 Link Merge 为 1x8 后，多个 Link 共享同一 LTSSM 吗？ 119
313. PCIe 启动建链之前，对初始化时间有要求吗？ 119
314. RC 在复位 1 s 后才会认定设备是否 Broken，为何要求链路在 100 ms 内 Link Up？ 120
315. Physical Link Up=1 体现在哪，有相关寄存器读取吗？ 120
316. LTSSM Polling.Compliance 跳转条件中提到的 Passive Test Load 是什么？ 120
317. LTSSM 状态机只有一个，但 L0s 状态却分成 Tx、Rx，这是如何做到的？Tx 和 Rx 一个 L0s 一个 L0 时，链路状态算 L0 还是 L0s？ 120
318. Alternate Protocol Negotiation 是什么 ？ 跟 Modified TS 的关系？ 121
建链失败 121
319. 使能了 LTSSM 后，LTSSM Status 一直处于 Detect 状态，可能是哪些原因呢？ 121
320. 成熟的 EP 设备插在商用服务器上，某些槽位会偶发扫不到设备或者是识别到但掉 Lane，这种情况除了用协议分析仪分析还有何定位手段？ 121
321. x4 设备 LTSSM 状态一直在 Polling.Active，大概什么原因？ 121
322. Xilinx PHY 的 PIPE 接口输出信号与仿真一致，但是链路状态没有跳转一直处于 Detect 状态，可能是什么原因？ 122
323. 链路两端只有一端从 L0 退到 Recovery 又回到 L0，另一端一直在 L0，这正常吗？ 122
324. x4 设备只用 x1，建链时 LTSSM 在 Polling.Active 和 Polling.Compliance 之间反复跳转，可能是什么原因？ 122
325. RC 直连网卡建链时找不到网卡设备，查看 RC 状态机处于 Polling.Compliance 状态，大概是什么原因？ 122
326. EP 端的 LTSSM 上电后走到 bcdb 后直接就到 0 了，然后最终 0123 一直卡在 3 出不来，有哪些 Debug 思路？ 123
327. 板卡金手指插在标准服务器上，LTSSM 起来后概率性卡在 Polling.Comliance 状态不动，有什么定位手段？ 123
328. 建链时 EP LTSSM 卡在 Recovery 状态，RC、EP 速率分别为 2.5 GT/s 和 8 GT/s，这可能是什么原因？ 123
329. PCIe EP 设备对接戴尔服务器，Reboot 服务器主机后主机无法扫描到 PCIe EP 设备，可能是什么原因？ 123
330. Target Speed 为 Gen3 速率，建链时 LTSSM 反复从切速后的 Recovery.RcvrLock 掉到 Detect，该怎么分析呢？ 124
链路非预期（掉链/掉宽/掉速） 124
331. 建链时掉 Lane，一部分检测到对端、令一部分没有检测到对端，可能是什么问题？ 124
332. L0p 状态下，不工作的 Lane 失锁了怎么办？直接上调 Lane 会有问题吗？ 124

333. 有个设备在 Gen2 时可以找到，Gen1 有一定概率找不到，这是什么原因？不应该是默认先到 Gen1 再协商到更高速率吗？ 124
334. 为什么有的物理通路信号质量很差，该如何提升物理信号质量？ 125
335. 链路两边的 Lane 数不一样（DSP x4，USP x2），但还是都在 L0 进行数据报传送，没有进入 Recovery，可能是什么原因？ 125
336. RC 与 EP 间通信，初期正常，为什么后期只有 RC 端接收、没有 RC 端发送呢？ 126
337. PCIe EP 偶尔会降 Lane，眼图看着正常，有何好的 Debug 思路吗？ 126
338. 2.5 GT/s 速率下，在枚举阶段 PCIe 位宽由 x4 降到了 x2，一般是什么原因？ 126
339. Gen1 x4 链路建链偶发掉 Lane 到 x2，失效的 Lane 在 Polling 24 ms 内不能 Lock，确认信号质量没问题，可能是什么原因？ 126
340. PCIe 链路两端均具备 x8 能力但建链后为 x1，有什么定位手段？ 126
341. 在未出错的前提下链路会自动从 L0 状态转到 Recovery 状态，该情况正常吗? 126
342. PCIe 建链成功后，链路流量大时每隔几十秒报一两个 Receiver Error、Bad DLLP、Bad TLP，不打流或者流量小时没问题，有何定位思路？ 127
343. 在 x1 Gen1 下链路训练没问题，提升速率或 Lane 数后链路训练失败，是信号质量问题吗？ 128
344. 对设备做 Reboot 后链路掉速，可能是什么原因？ 128
链路操作 128
345. Link Up 之后如何发起更改链路宽度，有寄存器触发吗？ 128
346. smlh_link_up 拉高之后需要如何配置才能让 rdlh_link_up 信号也拉高？ 128
347. 支持 Lane Reversal 的 PCIe x16 板卡插接到不支持 Lane Reversal 的服务器 x16 插槽上，该插槽实际只有 x8 工作，可以建链成功吗？ 129
348. PCIe Link 先建链到 x4 再改 EP 寄存器降 Lane 到 x1 是可以的，但先建链到 x1 L0 再升 Lane 到 x4 失败，是什么原因？ 129
349. 可以直接写配置空间 Link Control Register bit5 触发 Retrain 吗？感觉未生效。 129
350. 在 EP Rx Term 存在的情况下，RC 该如何忽略其中未使用的 Lane? 129
链路训练仿真 130
351. DUT-VIP 通过 PIPE 对接，仿真时报错检测到 Unknow Order Set，该如何定位问题？ 130
352. DUT-VIP 快速仿真模式下，过了 Gen1 L0 后在均衡阶段报超时，是哪里设置不对吗？ 130
353. RC DUT 向 EP VIP 发起读写操作，数据无法写入，可能是什么原因？ 130
354. Synopsys 公司的 VIP 如何配置有序集的 Disable 字段让 Link 进入 Disable 状态？ 130
355. 仿真时，RC 还没有检测到 Rx 端口变化就从 Detect.Quite 跳到 Detect.Active 状态了，可能是什么原因？ 130
356. 采用 Synopsys IP+VIP 仿真，DUT 侧的 iip_chip 层做了替换，波形显示 DUT 的 LTSSM 走到了 L0，smlh_link_up 起来了但 rdlh_link_up 没起来，可能是什么原因导致的？ 131
357. Synopsys PCIe IP 做 EDA 仿真时，Gen1、Gen2 的 Link 正常，Gen3 时 EP 在 Phase 0 出现 12 ms 超时，对于 Gen3 是否有其他寄存器需要配置？ 131
均衡 131
均衡过程理解 131
358. PCIe 采用了哪些主要的信号补偿技术？ 131
359. 均衡阶段交互 TS，TS 中的 P0 ~ P10 Preset 有什么用？ 132
360. 均衡参数中的 Preset 和 Cursor 有具体的数量对应关系吗？ 132
361. LTSSM 均衡 Phase 0 ~ Phase3 都在干什么？ 132

362. Pre-cursor、Cursor 和 Post-cursor 符合这个表。就像这样，是否能得出下表呢？下表里面有的值没有对。 133
363. FIR 均衡器有哪几种输出电平？ 133
364. 上文中，当前 bit 是 1，011 的组合时为何就是 Normal 电平，111 就是 De-emphasized 电平。如果当前 bit 是 x，那么 0x1 就是 normal，1x1 就是 De-emphasized 电平，1x0 就是 Pre-shoot 电平，0x0 就是 maximum-height 电平，是这么规定的吗？ 134
365. Rx 的均衡参数不是自动调整的吗？在初期寄存器配的均衡参数有用吗？是否只做初始参数用，最后使用的还是系统自己调整优化后的均衡参数？ 135
366. 协议里提到的 start_equalization_w_preset 这个字段是做什么的？ 135
367. PCIe Rx AFE 的参数是动态均衡 Phase 2、Phase 3 期间调整的吗？ 135
368. PCIe Adaptive Rx EQ 在 Phase 1 前还是 Phase 1 后？ 136
369. Phase 0/1 阶段不是已经做好 Preset 评估了吗，为什么在 Phase 2/3 还要做 Preset 评估？ 136
370. PCIe Gen5_Eq_Control_Register 中的 USP/DSP Tx Preset 是自己设置的还是对端设置的？ 136
371. 11 个 Preset 都是针对 Tx 设置的参数吗？为什么 Rx 没有 Preset 值？ 136
372. 均衡状态下 TS Symbol 6 的 PRBS Pattern 是做什么的？配置为 0 和 1 的作用是什么？ 136
373. PCIe 3.0 做均衡需要发 PRBS 吗？ 136
374. 如果链路均衡时，DSP 没有给 USP 传 Preset 值，USP 用的是自己 Equalization Control 寄存器里的 Preset 值吗？ 137
375. PCIe Gen3 Tx 均衡参数是在 Recovery 阶段确定的吗？ 137
376. 如果均衡在 Phase 1 失败了，还能通过修改配置完成建链吗？ 137
377. 如果指定选择均衡到 Phase 1，做完 Phase 1 误码率高会怎么处理？ 137
378. 做均衡时只能选择到 Phase 1 或 Phase 3 吗，不能到 Phase 2 吗？ 137
379. 什么时候可以不做均衡的 Phase 2、Phase 3？ 137
380. Recovery.Equalization 阶段确认 Tx Preset 起始是固定从 P0 开始扫描还是可以自定义？ 137
381. Rx PHY 在调 Tx Preset FFE 时如何得到 FOM 的？需要在一次接收过程中画出整个眼图吗？ 138
382. Recovery.Equalization 阶段的时间要求是协议固定死的还是可以通过寄存器来调整？ 138
383. 在 PERST 释放之前可以 Override PHY 内的 Tx FIR 寄存器来修改 Tx Deemphasis 吗？ 138
384. 目标速率为 Gen5 时，均衡是 Gen3 -> Gen4 -> Gen5 逐步做还是只做 Gen5 的？ 138
385. FOM 是 Synopsys 独有的吗？ 138
386. Tx、Rx 均衡时的 C_0、C_{-1}、C_{+1}系数是算法自动调整的吗？可以自己控制吗？ 138
387. 链路训练为什么 Gen1 到 Gen3 再到 Gen4，不能直接上去么？ 139
388. LTSSM 中均衡相关变量 start_equalization_w_preset 是如何跳变的？ 139
389. 当前 USP 为 2.5 GT/s 或 5 GT/s 且最大支持 16 GT/s 时，start_equalization_w_preset 该怎么处理？ 140
390. PCIe Equalization 有哪几种模式？ 140
391. PCIe Full-Equalization 是什么？ 140
392. PCIe Bypass-Equalization-to-Highest-Rate 机制是什么？ 140
393. PCIe No-Equalization-Needed 机制是什么？ 141
394. PCIe 是如何重新发起均衡的？ 141
395. PCIe Quiesce Guarantee 是什么？ 142
396. PCIe Equalization 中提到的 Full Swing、Reduce Swing 是什么意思？ 142
397. 为什么 8 GT/s 及以上速率时需要均衡？ 142
398. PCIe 训练完毕后，如何重新进行链路训练？ 143
399. DSP Equalization 没有 Phase 0，那 DSP 是怎么给 USP 发送 Preset 的？ 143

均衡实际问题 143
400. 在 Link Up 之后想调节 Rx 端的均衡参数，OS 下写完相关寄存器后是否需要重新 Retrain？还是说 OS 下写其实根本不会再生效？ 143
401. 目的为恶化 RC 端 Rx 的信号质量，除了调整 Rx 的 CTLE 和 DFE 等均衡参数，还有其他的恶化手段吗？ 143
402. PCIe 寄存器中有个 Link Equalization Request Interrupt Enable，这个中断是如何产生的？产生后如何处理？ 144
403. PCIe Controller 常用的均衡算法是用 FOM Mode 还是 DIR Mode？能混合用吗？ 144
404. 实际应用中，用 FOM 均衡机制选 Preset 靠谱吗？如果 PHY 的 FOM 和 DIR 结果都不可信，但是 PHY Rx 均衡本身无问题，那系统是否有办法设定固定 Preset 值？该做法实际中会不会带来额外的应用问题？ 144
405. 卡插在 x86 上，链路均衡过程中卡在 Phase 2 中不动，有什么解决思路？ 144
406. 均衡 Slave 收到 Master 发来的 Tx Preset 后可以不改 PMA 电气参数吗？ 144
407. 均衡的 Phase 0 阶段，USP 端发送的 Preset 值为 1111 保留值不在 P0 ~ P10 范围内，这是为什么呢？ 145
408. 链路速率切为 Gen5 做均衡时，LTSSM 状态从 Recovery.Equalization 的 Phase 1 因超时跳到了 Recovery.Speed，速率也切为了 Gen1 速率，这种情况该如何解决？ 145
409. 采用均衡阶段协商的 Tx Preset 测试 Tx 眼图发现不是最佳，有什么办法调整吗？ 145
410. 用 Synopsys 公司的 PHY 测试发现多次均衡结果不一致，这是否正常？ 145
411. PCI-SIG Compliance 测试过了，但在客户那 Tx Preset 测试过不了，该怎么办？ 145
412. Gen4 目标速率开启了 Equalization bypass to highest rate，为什么没起作用？ 145
FLIT MODE 146
413. PCIe 6.0 有哪些新变化？ 146
414. PCIe 6.0 的 Flit Mode 是什么？ 146
415. 为什么需要 Flit Mode? 146
416. PCIe Flit Mode 支持几种 Flit? 147
417. Flit_Marker 是什么？有什么用？ 147
418. Flit 不做编码，不怕出现连续 0 或 1 烧坏耦合电容吗？时钟同步怎么办？ 147
419. Flit 不做编码，难道 PCIe 6.0 不需要控制字符集 K 字符吗？有续集 OS 的字符编码可能会与 Payload 重叠吗？如何区分 Payload 与包头？如果 Order Set 跟 Payload 重叠了怎么办？ 147
420. PCIe 6.0 每个 Flit 最大容纳 TLP 中的 236 B，超过 TLP 超过 236 B 就需要放到多个 Flit 中，那么每个 Flit 中都需要一个 TLP Header 吗？ 148
421. PCIe 6.0 中 CRC 和 FEC 执行顺序是什么 148
422. 为什么 PCIe Flit Mode 采用 3 路交织的方式计算 ECC 就能保证 16 路不出错呢？ 148
423. Flit Mode 下，如果只想发最短的单个 MRd TLP，NOP TLPs 会按照什么频率插入呢？ 148
424. 如果应用层长时间没有 TLP，链路是持续传输 NOP-FLIT 呢，还是停止传输 Flit？ 149
425. PCIe 开启 Flit Mode 需要写哪些寄存器？ 149
426. Flit Mode 时，Tx Buffer 和 Retry 机制应该都做在物理层、而非 Data Link 层了吧？ 149
427. Flit 定界是以 Flit State 是否为 Poisoned 或者是 Nullified 作为结束的标志的吗？ 149
428. Flit 长度是固定的，发送时分发到最大 16 个 Lane 同时传输的，接收端怎么从 16 个 Lane 恢复出这个 Flit 呢？ 149
429. PCIe Flit 打包时，如果有 NOP 插入，TLP 要求 4 DW 对齐，那么在 x8 上也是这样吗？ 149
430. 什么样的 Flit SN 算是隐式的？既然 Flit 中都携带了 SN 字段，为什么还说它是隐式的？ 149
431. PCIe Flit Mode 时不支持动态上调链路宽度吗？ 149

432. Flit 模式下是不再使用令牌了的，那么接收端怎么知道 Flit 的开始位置呢？ 150
433. Flit 是在哪一层封装的？是如何封装的？ 150
434. PCIe 6.0 规定的 Flit Size 有几种？ 150
435. 每个 Flit 都需要 Ack/Nak 握手吗？这不会影响效率吗？ 150
436. DLP 是嵌合在 Flit 中的，当没有要发送的 TLP 时，Flit 中 TLP Payload 位置如何填充？ 151
437. Flit Mode 时，TLP Header 需要放在 Flit 固定位置吗？ 151
438. Flit Mode 时，若 TLP Size 小于 236 B，该如何生成 Flit？大于 236 B 时呢？ 151
439. Flit Mode 与 Non-Flit Mode 时的 TLP Header 有何不同？ 151
440. PCIe TLP PCRC 是什么？ 151
441. TLP Trailer 是什么？ 151
442. TLP Header 中的 OHC 是什么？ 152
443. NOP TLP 是什么？ 152
444. NOP DLLP 是什么？ 152
445. NOP2 DLLP 是什么？ 153
446. NOP DLLP 和 NOP2 DLLP 有何区别？ 153
447. Idle Flit 是什么？有何作用？ 153
448. NOP Flit 是什么？有何作用？ 153
449. Idle Flit 和 NOP Flit 有何区别？ 154
450. NOP TLP 和 NOP Flit 有何区别？ 154
451. NOP Flit 和 NOP DLLP 有何关系？ 154
452. PCIe Gen6 中 DLLP Type 有哪几种？ 154
453. Flit Mode 下的 DLLP 有何变化？ 155

五、电源管理 156

电源管理过程理解 156
454. 设备是否可以只支持 ASPM L1 但不支持 L0s？ 156
455. RC 是在进入 L1 之后拉高 CLKREQ#来进入 L1.2 吗？ 156
456. 追：接 VIP 进行测试时，RC VIP 自动控制进入了 L1，之后手动拉高 CLKREQ#。但由于 EP 反应晚于 VIP，EP 会拉低 CLKREQ#导致 VIP 未进入 L1.2 并退出 L1。若想进入 L1.2，是否需要反复、持续手动拉高 CLKREQ#？ 156
457. USP 处于 ASMP L1 状态，此时让 USP 进入 $D3_{Hot}$，USP 会立刻开始退出 ASPM L1，还是要等接收到协商进入 PCI L1 的 DLLP 包才开始退出？ 156
458. USP 是 ASPM L1 状态，DSP 是 PCI L1 状态，那 USP 退出 ASPM L1 会导致 DSP 退出 PCI L1 吗？反过来，USP 退出 PCI L1 会使 DSP 退出 ASPM L1 吗？ 156
459. Switch 的 DSP 可以向 RC 发 PME_Turn_Off 吗？ 157
460. 在开启 ASPM 的情况下，什么条件可以触发 EP 发出 PM_Active_State_Request 包? 157
461. PCI-PM 电源管理方法中，通过软件调度 RC 写 EP 配置空间相关寄存器为 D1/D2/D3 就会使得 EP 与 RC 协商进入 L1 吗？ 157
462. 现在软件还用不用 PCI-PM 了？ 157
463. PCI-PM、ASPM 这两个标准该怎么理解？ 157
464. 是不是 RC 没有 PCI-PM，只能发给 EP 或者 Switch? 157
465. 如何让 EP 发出 PME Message 请求？ 157
466. PCIe Switch 的 DSP 能生成 PME_Turn_Off 给 EP 吗？ 158
467. LTSSM 跟 Link PM State 是一回事吗？硬件没做电源域划分 LTSSM 能支持 L2、L3 吗？ 158

468. 在 D0 Uninitialized 状态可以发 Type 1 Cfg 包吗？ 158
469. PCIe EP 进 $D3_{Cold}$ L2 后重新上电并复位了 EP 配置空间，RC 会重新枚举这个 EP 吗？ 158
470. 如果在 Non-Posted 请求发出后、其 Cpl 包回来之前把设备置于 $D3_{Hot}$ 状态，这个 Cpl 包还能正常返回吗？是不是进 $D3_{Hot}$ 的条件就会等所有 inflight 的都结束了才能进？ 158
471. 那等从新到 D0 再回吗？如果一直不回会出现什么情况啊？ 158
472. 在 Switch 中 USP 的链路状态一定比 DSP 更活跃吗？可以 USP 在 L1、DSP 在 L0s 吗？ 158
473. 需要达到什么条件 Linux PCIe 标准驱动才会去执行配置 EP 进去 D3 低功耗的流程？ 159
474. 是不是 DSP 要先进入 $D3_{Hot}$，EP 才能发起 PCI-PM L1 握手？ 159
475. 对于 Switch 的 DSP，然后 VIP 作为 EP 发起 PCI-PM L1 请求，这个时候 DSP 应该是 D0 还是 $D3_{Hot}$？ 159
476. D0 能进入 ASPM L1，不能进 PCI-PM L1 吗？这两个有什么区别？ 159
477. ASPM L1 和 PCI L1 的功耗一样吗？还是说哪个更省电？ 159
478. PCI-PM 一般的使用方式是怎么样的？以一对一的 RC-EP 为例，在系统 Suspend 时，是否为 RC 写自己 PM CSR bit[3:0] 为 D1/D2/D3、然后 RC 写 EP PM CSR bit[3:0] 为 D1/D2/D3？RC 和 EP 是否可以自己写自己的 PM CSR bit[3:0] 为 D1/D2/D3？ 159
479. 主机将 EP 功耗状态配置成 $D3_{Hot}$ 或 $D3_{Cold}$ 之前，会主动将 Command 寄存器中的 Bus Master En 置为 0（即关闭 EP 发起的 Mem 请求）吗？ 159
480. 如果 PCIe EP Function 从 D0 转为 $D3_{Hot}$ 前主机不关闭 Bus Master En，那么 EP 一直发 MRd 请求，驱动软件如何确保 Outstanding Completions 已经终止呢？ 160
481. 当前设置 PM CSR 为 D3 直接进入 L1 了，如何操作能进 L2？ 160
482. ASPM Support 和 Enable 这两个字段之间是怎么搭配工作的？ 160
483. 这个 PM 状态机是软件控制还是硬件控制？ 160
484. Power State 只有 2b，是如何区分 D0 中两个子状态的？ 161
485. 一个 EP 设备不支持 ASPM 但是 lspci 看着被写成 D3 了，EP 的 LTSSM 不停在 L0 及 L1 之间切换，可能是什么原因？ 161
486. ASPM 和 PCI-PM 都包括哪些低功耗状态？ 161
487. 请问 L0、L1 是链路的电源状态还是设备的电源状态？ 161
488. Power Budget Capability 中的寄存器如何使用？ 161
489. PCIe Switch 下游端口如何从 L23 ready 状态退出到 L0 状态？ 161
490. EP 发 L1 Request 是纯硬件的行为吗？软件能否控制？ 162
491. 如果 Switch 没有 DSP，是否说 USP 取决于 DSP 进入低功耗电源状态的条件一直满足? 162
492. L0s 状态下是不会发包括 FC 流控在内的任何 DLLP 吗？ 162
493. PCIe L1 发起请求的是 RC 还是 EP 呢？ 162
494. Device 被配置为 D1 后想回到 D0，需要 RC 还是 EP 去配？ 162
495. $D3_{Cold}$ 时 Link 只能为 L2、L3 吗？ 162
496. 如果只支持 $D3_{Hot}$，是不是就不支持 L2、L3？ 162
497. L0p 时没有 Active 的 Lane 是处于 EI 状态吗？ 162
498. RC 和 EP 都可以发请求进 L1 吗？ 163
499. 设备端 ASPM 进入对应子状态，需要跟其他 IP 关联吗？还是只是 PCIe 这个 IP？ 163
唤醒 163
500. 唤醒有哪些方式？ 163
501. 将 PCIe 从 L2 低功耗状态唤醒有哪几种方式？ 163
502. 单个 RC 下挂了多个 Device，这些 Device 的 WAKE#信号线与在一起，当 Host 检测到低电平时如何获知是哪个 Device 拉低的以便退出 L2 对其供电？ 163

503. Wake Up 功能应该如何实现？ 163
504. 所以 RC 先给与该 Device 对接的下行 Port 口供电？因为 Switch 知道是哪个 Device 要退出 L2？ 164
505. WAKE#引脚是 Output，在 RC 模式下 PCIe 控制器怎么通过 WAKE 引脚唤醒呢？ 165
506. CPU 是如何知道 WAKE#引脚高低电平的？ 166
507. RC 怎么知道是哪个 Device 拉低了 WAKE#呢？ 166

六、系统架构 166

中断（INTx/MSI/MSI-X） 166
508. MSI 和 MSI-X 是什么？ 166
509. MSI-X 是如何工作的？ 166
510. Interrupt Line 是干什么的？是代表这个设备用哪个中断信号吗? 166
511. INTx 只能由 Upstream Ports 发起，MSI、MSI-X 有这个要求吗？ 167
512. 在 RC 端，MSI 写跟常规的 MWr 有何区别？ 167
513. MSI 跟常规 MWr 在排序上有什么要求吗？ 168
514. RC 内是在什么时候识别 MSI 还是常规 MWr 的？ 168
515. RC 的 MSI-X 检测一般是软件实现还是硬件实现？ 168
516. RC 端是如何检测 MSI 中的地址的？ 168
517. RC 端是如何监测 MSI-X 中的地址的？ 168
518. RC 给 EP 发 MSI、MSI-X 有实际场景吗？有何意义？ 169
519. 如何实现 RC 到 EP 的 MSI/MSI-X 功能？ 169
520. MSI Capability 中消息数据寄存器的数据是由系统软件写入的，还是由 PCIe 设备负责更新？ 169
521. EP 想要使用 MSI/MSI-X 中断，需要上一层接的 RP 支持 MSI/MSI-X Cap 吗？ 169
522. GIC、APIC 是什么？ 170
523. PCIe MSI 中断的地址是在该 Function 的 BAR 里面挑一段地址空间，还是在 BAR 之外主机重新分配地址空间？ 170
524. 根据 Xilinx PCIe RP 文档对 MSI 描述中，其支持 64 Vector 的 Interrupt，根据协议 MSI 不应该最大支持 32 Vector 吗？这个 32~63 Vector 有何用处？ 170
525. Synopsys PCIe IP MSI-X 要求所有 PF 和 VF 的 Table BIR、Table Offset、PBA BIR、PBA Offset 必须相同，如果相同该怎么区分呢？ 170
526. MSI-X Table 要挂在 BAR 上，为什么不像 MSI 那样把地址数据放在寄存器中？ 171
527. Synopsys PCIe Spec 提到 MSI-X Table 和 PBA 位于 Application Memory Space，这里的 Application Memory Space 是指 DDR 吗？ 171
528. 使用 Synopsys 的 iMSI-X 时，中断源是如何触发 MSI-X 请求的？ 171
529. 如果 EP 要同时发送多个 MSI-X 中断，多个中断的仲裁是在 PCIe 外部实现的吗？ 171
530. MSI-X 中断数量有什么影响吗？为什么我开 2048 个会被裁剪？ 171
531. 为什么 MSI-X 最多只能 2048 个？ 172
532. 配置 MSI-X 时一般开多少 Entry？ 172
533. VF 跟 PF 是共享 MSI-X 数量吗？ 172
534. 为什么 PCIe 读写都要发生 MSI 中断？ 172
535. 在 RP 侧，Synopsys 公司的 Controller 和 ARM 对接，这个 MSIs 是不是也包含了 MSI-X 了，只是命名成了 MSIs？ 173
536. RISC-V 能接收并解析 MSI-X 中断吗？ 173

537. MSI 和 MSI-X TLP 的 Length 是固定 1 个 DW 吗？ 173
538. MSI 和 MSI-X 的 First BE 必须是 4'b1111、Last BE 必须是 4'b0000 吗？MSI 如果没使能 Extended Message Data，First BE 可以是 4'b0011 吗？ 173
539. CPU 如何区分收到的消息中断类型是 MSI 还是 MSI-X？ 174
540. 如果主机收到 EP 发来的 MRd 地址为 MSI 或 MSI-X 的地址，应该怎么回 CplD？ 174
541. MSI-X 发中断之后，CPU 要怎么操作才知道是哪个向量发的中断呢？ 174
542. RC 收到 MSI Message 后怎么写 Memory，之后中断又是怎么产生的？ 174
543. LPI 写是指什么？写 Mem 的动作是 RC 完成还是 GIC 完成？ 174
544. MSI 中断过 MSI64 送往 ITS 时，如果判定地址匹配上了 ITS Target Address，MSI64 Master 口出来的地址会自动被偏移 16b 并在低位补上 0x0040 吗？ 174
545. Linux 的 PCIe EP 驱动中用 mmap 申请一片虚拟地址 DMA，只要文件超过 6 M 就会申请失败，需要调系统里哪个参数来解决该问题？ 174
546. 在 ARM64 系统中，LPI 如何依据收到的 MSI 中断来区分不同 Domain 下 Requester ID（BDF）相同的两个 EP？ 174
547. 系统中有多个 RC 时，ITS 和 RC 是一一对应的吗？ 175
548. EP MSI 中断消息上报后没有被应答，EP 会按写事务超时机制重新上报吗？ 175
549. MSI 支持 32 个中断是否意味着一个 MSI 中断地址对应 32 个中断向量？从应用角度讲，不同的向量号之间有什么区别？ 175
550. RP 只是接收 MSI，是不是 RP 不用配置 MSI Capability？ 176
551. SRIOV 不同 VF 分配到的 MSI 地址可以是一样的吗？ 176
552. 不同 VF 的 MSI 地址一样的场景常见吗？ 176
553. 该如何使能 VF 的 MSI-X 中断？ 176
554. PCIe 的 MSI-X 中断消息要经过 SMMU 吗？ 176
555. 系统中没有 SMMU 这个组件，LPI 如何区分不同 RC 下相同的 BDF？ 176
错误处理 176
556. Link Error 包括哪几种？发生 Link Error 时是如何恢复的？ 176
557. 在 EP 端主动注入 PARITY_SKPOS_ERR 错误，Host 收到后会挂死吗？ 177
558. Synopsys 公司 EP IP 如果发生 CE 错误，有寄存器能读出 CE 错误数量吗？ 177
559. 什么情况下 PCIe 会报 Failed to Check Link？ 177
560. log 显示错误已经被纠正，还用管吗？ 178
561. AER 上报错误是走 MSI、MSI-X 中断方式吗？ 178
562. 如何手动清理 AER 的 CE 错误？ 178
563. PCIe 设备起来后 RC 端 CPU 通过 lspci 查看有 Advisory Non-Fatal Error，可能是什么原因？ 178
564. 有 DPC 和没有 DPC 的情况下，出现 UE 错误会有什么不一样的现象？ 179
565. 怎么知道 Host 有没有 DPC？ 179
566. lspci 看到的 Secondary PCIe 状态中的 lane_err 是指发送侧还是接收侧？AER 里的 rx_err 是指 EP 的 Rx 端么？两者什么关系？ 179
567. AER 里面 Header Log 是不是只有 RC Port 能记录第一条错误？如果有新的 Error 会把前面的记录覆盖吗？ 179
568. dmsg 查看 AER 一直上报 rx_err，但 lspci 看有的会报、有的不报这个错，是什么情况？系统软件会在上报后清这个错误么？ 179
569. RP AER 可以使用 Error Interrupt（INTx/MSI）来上报错误吗？ 179

570. 假设 EP 报告 AER 错误，那么 lspci 究竟查看哪个设备的 AER 为准？只需要查看 EP 本身，还是 RC 也能体现？ 179
571. 如果报了 Mutiple AER，AER 驱动会去遍历这个 RP 下的所有节点看是否发生了错误吗？ 180
572. Linux 默认未开启 ECRC 功能，这是完全由 OS 控制的还是 Device Driver 也可以改？ 180
573. 如何根据 Requester ID 和 Message Code 定位错误报告位置？ 180
复位 180
574. PCIe 有哪几种复位机制？ 180
575. Host 触发了 FLR，等 Firmware 写完对应 Function 的 flr_ack 为 1 后，Host 要做什么操作才能激活被 Reset 的 Function？ 181
576. Host 对 Switch（Pericom）的 Upstream 做 Hot Reset，对于 Switch 的 DSP 的 Hot Reset 有什么时间要求吗？反复对 Switch 的 USP 做 Hot Reset，会发现 Host 偶尔会丢失 Switch 和下面的 Hierarchy，如果 Switch 下面不接设备就不会丢失，可能的原因是什么？ 181
577. SNPS 的 ECAM 如果写 Bridge Control 的 Second Bus Reset，复位后能解复位吗？ 181
578. PCIe 作为 Device 时，Controller 的 perst_n 一般怎么接？ 181
579. 该如何理解 Cold Reset 和 Warm Reset？ 181
580. Hot Reset 和 Warm Reset 后再传输需要重新 Link 吗。 182
581. Link Down 再次回到 L0 后，Config Space 会被复位吗？ 182
582. EP 意外 Link Down 再次 Link Up 后，Host 还能访问吗？ 182
583. Hot Reset 或 Warm Reset 时 Host 会认为这个设备掉了吗？ 182
584. 如何启动 FLR 相关复位？ 182
585. 难道只能对 PF 做 FLR 吗，不能单独控制某一个 VF？ 182
586. 对某个 PF 做 FLR 必须不能影响其他 PF 吗？还是仅仅建议不能影响其他 PF？ 182
587. 如果 EP 响应 Host 的 perst，开始复位，但是先于 Host 释放了，会不会有问题？ 182
588. Cadence 公司的 Gen3 IP 是如何处理 Hot Reset 的？ 183
589. Hot Reset 之后，一定要 Rescan 么？Linux PCI 驱动中为何必须要 Remove->Hot Reset->Rescan，单独做 Hot Reset 不行么？ 183
590. PCIe 设备收到主机发来的 Hot Reset TS1 后，需要等未完成的数据传输完后再 Reset 吗？Flushing Mode 是指传完还是直接清掉？ 183
591. Hot Reset 后需要 Rescan 吗？ 183
热插拔 183
592. PCIe 热切换和热插拔是什么？ 183
593. 热插拔一般用在哪？ 183
594. 热插拔需要物理按键吗？ 184
595. PCIe 设备该如何支持暴力 Hot-Plug？ 184
596. Hot-Plug 需要 Host 配合吗？跟 EP 有关吗？ 184
597. EP 端软件需要支持 Hot-Plug 功能吗？ 184
598. 目前实现 Switch 整体热插拔了吗？ 184
599. 目前 Linux 内核是否已支持整个 PCIe Switch 及其下挂设备的热插拔操作？ 184
600. Switch 支持热插拔吗？ 185
601. Switch 热插拔未实现技术上卡在哪里了？ 185
602. 硬件上是如何检测到热插拔的？ 185
603. Linux 中是如何识别和处理热插拔的？ 185
604. 如何知道是哪个 Slot 发生了热插拔事件？热插设备是如何添加到原 PCI 设备树中的？ 185
605. PCIe 5.0 热插拔的 Async Removal 和暴力热插拔有什么区别吗？ 186

606. 软件上 Remove PCIe 设备和 Hot-Plug 是一回事还是两个概念？ 186
607. 热插拔中断上报是硬件自动报的，还是软件报的？ 186
608. Switch 端口预留空间不够的时候，RC 还有空间，不可以再分配吗？ 186
609. 热添加一个 PCI 子系统，PCI 空间不够分配怎么办？ 186
610. RC 端通知 Hot-Plug 功能跟 PCIe Controller 有关吗？ 186
611. PCIe Hot-Plug Events 中的 Command Completed Event 该怎么理解？ 186
612. 热插拔时主机如何获知设备用的 Device Number 是多少？同一个 Bus 下的设备 Device Number 相同了怎么办？ 187
613. 目前操作系统内核支持对整个 Switch 进行热插拔吗？ 187
扩展能力 187
基本介绍 187
614. 如果 RC 为 PCIe 2.0 但 EP 为 PCIe 2.1，PCIe EP 2.1 相关的 Capability 能否被 RC 识别到？ 187
615. 如何查找扩展配置空间中的各种 Capability？ 187
DRS 187
616. Base Spec 中讲 DRS 时提到，DRS Supported bit clear 时可以发送 DRS Messages，又说 DRS-capable 的 Port 才能发 DRS，两者不是冲突了吗？ 187
RN 188
617. 什么场景下会用 PCIe RN 这个 Feature？ 188
MTCP 188
618. Device 能往 DDR 发 MTCP 的包吗？MTCP 包存在 DDR，用 VMD 来承载 MCTP。 188
IDE 188
619. PCIe 在什么场景下需要用到 IDE？ 188
Atomic 188
620. PCIe 原子操作的原子性体现在哪？ 188
621. PCIe 原子操作地址为 32/64/128 bit 时，要求 4/8/16 Byte 对齐吗？ 189
622. 原子操作 FetchAdd 的 Length=2 同时 BE=0，这符合协议吗？ 189
623. PCIe 原子操作中能开 ID 排序吗？能用 Steering Tag 吗？AT 还能用于地址翻译吗？ 189
624. 设备发出 AtomicAdd 操作 Host Memory，RC 收到该事务后是不是向 Memory 发起读操作、然后 Add 后写入，RC 端再返回 Atomic 完成状态？ 189
OBFF 189
ARI 189
625. 枚举过程中如何发现下游端口连接的 EP 是否支持 ARI 功能？是不是先按照不支持 ARI 进行枚举，读完配置空间发现支持 ARI 后，再对设备进行配置？ 189
626. 在 Linux PCI 枚举时，如果支持 ARI 会去 ARI Capability 寄存器查找 Next Function Number，那么每次去查都是相同的 Next Function Number 吗？ 189
627. ARI 功能与 SR-IOV 有关吗？ 189
LTR 190
628. 软件角度应该如何使用 LTR Message？是不是 EP 业务可以随意发送任意数值的 LTR Message 给 RC？ 190
629. EP 设备访问 PC 端 DDR 同一个地址，首次访问 Delay 时间特别长，再次访问 Delay 就正常了，为什么首次访问 Delay 时间长？ 190
630. 主机收到设备发来的 LTR Message 后会回一个 LTR Message 吗？ 190
631. 如果主机不接受设备发来的 LTR Message 请求值怎么办？ 190
TPH 190

632. TPH 有实际使用场景吗？ 190
633. Switch 中的 Cache 缓存的是 Device Mem 的内容吗？假如是的话，直接写到 Device Mem 中 Device 在每次读的时候不是更快吗？ 191
634. 请问 TPH Capability ST Table 中的 Entry 中要填什么？ 191
635. EP 如何将 TPH 里的 Steering Tag 对应到 RC 相应的 Cache 呢？怎么用？ 191
DMWR 192
636. DMWr 是什么？ 192
637. 为何需要 DMWr ？ 192
638. DMWr 请求的排序规则是怎样的？ 192
639. Deferable Memory Write (DMWr) 的 Defer 体现在哪？Rx 收到 DMWr 后无需延时、立即写并反馈 CplD，本质是 Non-Posted Write 吧？ 193
640. DMWr 的 Completion 应该怎么回？如果是 SC，那是回 Cpl 还是 CplD？ 193
641. 发送端上层软件如何知晓延迟写已经完成？是轮询 RC 寄存器吗？ 193
642. 对于被丢失的 Posted Write，软件一般怎么处理？如果目标地址是关键的控制指针之类，最后软件岂不是动辄崩溃？如果 Posted 写有很大概率静默丢失的话。Local DDR 是不是也有大概率丢失写请求？ 193
643. 为什么要设立这个 DMWr 的重传机制？Requester 目前来看就是 enqcmd 指令自身，该指令必须得到 cmpl 才算执行成功，在这之前线程处于阻塞状态，接收者按理说不需要告知请求者 DMWr 的中途执行状态。如果接收者丢掉 DMWr 让发送者重传，那么说，常规的 Posted MWr 请求，也有可能被丢失？PCIe 不是保证端到端无损的么？ 193
644. PCIe Base Spec 未规定 DMWr 的 Length，但有相关寄存器配置来指示支持的最大 DMWr Length 为 64 B 或 128 B，是根据 DMWr32/64 来间接算 Payload DW 长度吗？ 193
645. 追问：这么说，发送端上层软件可以允许在非阻塞模式了。进一步推测，发送端的 RC 收到 DMWr 的 Cpl 或 Cpl.RRS 后都会发送中断，相对于 PCIe 5.0 需要 RC 在硬件上做中断相关改动？ 194
646. 追问：发送端上层软件如何知晓 DMWr 已经完成？是轮询 RC 寄存器吗？ 194
647. 追问：对于 NP Read，软件发出 MRd 后 CPU 拿到数据自动执行下一指令，软件继续运行；对于 Posted Write，写到 CPU 队列后即可继续执行下一指令，软件知道写操作已完成；同理，对于 DMWr，软件必须知道 DMWr 已完成，这个完成信号是如何传递给上层软件的？是否需要一条特殊指令？内核还需要一个 API？ 194
648. 追问：为何要设立重传机制？目前来看 Requestor 是 enqcmd 指令自身，该指令必须得到 Cpl 才算执行成功，在此之前线程处于阻塞状态，按理说 Completer 无需告知 Requestor DMWr 的中途执行状态。如果 Completer 会丢掉 DMWr 让 Requestor 重发该请求，是否意味着常规的 Posted Write 也会被丢失？PCIe 不是保证端到端无损的么？ 194
649. 追问：对于被丢失的 Posted Wirte，软件该怎么处理？如果目标地址是关键的控制指针之类的，任凭 Posted Write 静默丢失，软件岂不是会崩溃？ 194
650. 追问：处理报错时，状态已经丢失了，Host 如何得知是哪个线程写的哪个地址？线程不会跑飞吗？ 194
651. 发送端上层软件写数据可以运行在非阻塞模式，是否可进行以下推测：发送端的 RC 收到延迟写的 Cpl.SC 或者 Cpl.RRS 后都会发送中断，即需要发送端 RC 硬件上做改动？ 195
652. DMWr 写完成的信号是如何传递给上层软件的？ 195
653. 处理报错时，状态已经丢失了，Host 并不知道是哪个线程当初写入的对应地址，如何通知线程？此时估计线程已经跑飞了吧。 195
Multicast 195

654. 如果 EP 和 Bridge 上都有 Multicast Capability Register，那么在配置时是配 Bridge 的还是 EP 的？如果只需配 Bridge 的，那 EP 为什么要实现这些堆寄存器？ 195
655. PCIe Multicast 功能一般在什么场景应用？目前在服务器上应用需求大吗？ 195
VPD 196
656. 什么设备会用到 VPD 这个功能？需要 RP 配合吗？ 196

七、软件初始化与配置 197

配置空间 197
657. PCIe 配置空间是什么？ 197
658. PCIe 配置空间在哪里？ 197
659. RC 有配置空间吗？ 197
660. 每个 PCIe 设备都有一个配置空间吗？ 197
661. PCIe 配置空间 Size 是多少？ 198
662. PCIe 配置空间分为哪几类？ 198
663. PCIe 地址空间分为哪几块？ 199
664. PCIe 有哪几种配置空间访问机制？ 199
665. PCIe 地址空间有哪几种访问模式？ 200
666. DBI、ELBI、LBC、CDM 是什么？ 200
667. 扩展配置空间怎么找？ 200
668. PCIe Configuration Space Header 是什么？ 201
669. PCIe 配置空间中的 Expansion ROM Base Address 寄存器是干什么的？ 201
670. PCIe Config Space Header 中的 Expansion ROM Base Addr 寄存器是做什么的？ 202
671. 哪里能查询最新的 PCIe Class Code？ 202
672. Linux 系统中 PCIe Class Code 有什么讲究吗？是不是不能为 FF？ 202
673. PCIe Capability 地址是链表形式的，该如何确定哪些地址是省下来可自用的？ 202
674. 如果只用到 4 KB 配置空间中的一部分，剩余空间能省下来吗？ 202
675. RCRB 是什么？ 202
BDF 203
676. 枚举时 Host 是如何给 Device 分配 BDF ID 的？我理解系统上电后，Device 自己不知道 BDF，Host 是如何通过深度优先算法能够实现呢？ 203
677. 手动改 NVMe 盘 Bus Number 大于 0x80 无法工作，可能是什么原因？ 203
678. PCIe 设备最多支持 8 个 Function，该如何理解这里的 Function？ 203
679. PCIe 设备多 Function 一般是什么场景呢？一般是怎么用的？ 203
680. PCIe Phantom Function 是什么？ 204
681. PCIe Phantom Function 有何局限？ 204
682. Shadow Function 是做什么的？ 204
BAR 205
683. PCIe BAR 是什么？ 205
684. IO BAR 是什么？有什么作用？ 205
685. 为什么 PCIe 设备的配置空间里只有 BAR 寄存器定义了起始地址，没有定义 Size？ 205
686. BAR 空间基地址分配时有什么要求么？ 205
687. 有什么场景需要中途改 BAR Mask？需要在什么时候改？ 205
688. RC-EP 直连（非 NTB、Switch，未启动 EA）BAR 分配结果中出现 Virtual 字样是怎么回事？如何解决？ 206

689. 开启 IOMMU 后，PCIe BAR 里面是 Virtual Address 还是 Physical Address？ 206
690. BAR 配置时可操作的最低位为 12，是否表示<12 的位都不可以操作？那可以申请的内存为什么不是 2^12~2^31 大小呢？ 206
691. 现在还会发生 OS 把 BAR 安排到 High Mem 之后，由于设备指针长度不够导致 DMA 寻址不到 High Mem 的问题吗？ 206
692. EP 直接从 BAR 里读出 Pointer 使用可能会导致问题，因为有些平台上，这个地址并不是物理地址，RC 内部会翻译一次。不知道你怎么看? 207
693. 若访问到 BAR 空间中未使用的部分，比如 BAR 的大小是 256 KB，但只使用了 200 KB，当读取的范围包含后面 56 KB，应该如何回复 Cpl 呢？ 207
694. 设置了 64b 可预取的 BAR Base 为 0x2_4000_0000，BAR 配置的低四位为 4'b1100 (0xe)指示 64b/可预取，这样的话 0x2_4000_0000~0x2_4000_000d 会落在该 BAR 范围内吗？ 207
695. IP 生成的 BAR 大小，EP 软件再次配置 Base Address Register 更改 BAR 的大小还会生效吗？ 207
696. Resizable BAR Capability 一般用来做什么？是不是 Function 支持多个 BAR Size，软件选一种 BAR Size？比如 Function 支持 2 M 和 4 M，软件是不是就可以配 2 M 或者 4 M？ 207
697. 哪些设备的驱动支持 BAR Resize？内核支持 BAR Resize 吗？ 208
698. 某个 BAR Mask 空间大小 A 后又 Resize 为 B，A 和 B 之间有什么关系么？ 208
699. BAR Resize 是设备主动发起的是吗？Resize 之后需要通知 Host 重新分配吗？ 208
700. 如果 PCIe 设备只有 Mem Space 没有 IO Space，可以说是一个 IO 设备吗？ 208
701. 有没有这种使用场景：EP 自己从 BAR 里读出基地址以获知自身 BAR 空间所处的 HPA 位置，然后基于这个位置去做下游的一些事情？EP 会拿这个地址做什么事情？ 208
702. RC 没有 BAR，那么 RC 是否会把 EP 发来的任意地址 Mem Req 上传给 Host？ 208
703. PCIe BAR 寄存器存放的是 CPU 域的 System Memory 地址吗？ 208
704. 既然枚举时 Host 从 System Memory 空间找一块合适的空间给 BAR，为什么 BAR 地址不是 CPU 域地址？ 209
705. 如果开了 SRIOV，BAR 的枚举和使用会有什么不同？ 209
706. 对于 64 位的 Host 主机，可以认为物理空间有 2^64 Byte 吗？ 209
707. 实地址模式该怎么理解？有什么应用场景？ 209
708. Windows 系统下怎么使用 Resizable BAR？ 209
709. 如果 Host 要支持 Resize BAR 对 Host 硬件有什么要求吗？ 210
710. BAR Size 只支持二次幂设置，但设备内部只支持 96 G，驱动一般怎么处理该情况？ 210
711. Resize BAR 可不可以按照实际大小设置为非二次幂？ 210
712. Outbound 的地址可以不在 BAR 空间内吗？Controller 会做检查吗？ 210
访问问题 210
713. Host 的 CPU 如果读到 2 个相同的 Vendor ID 和 Device ID 会报错吗？ 210
714. 服务器访问网卡 BAR 空间，发现 BAR 空间数据更新很慢，可能是什么原因？ 210
715. Type 0/1 配置空间 Command Register 的 Memory Space Enable 默认 0 不响应 Memory 空间请求，在使用之前是否需要主动配 1？ 211
716. PCI 设备建链后配置空间变成全 0xffff，可能是什么原因导致的？ 211
717. RC 访问 EP 的 BAR2，却送到了 BAR0 对应的地址范围，可能是什么原因？ 212
718. 39 位的怎么寻址 1T 或者 4T 的硬盘？ 212
719. Host 做 BAR Resize 失败了，可能是什么原因？ 212
720. RP 配置为 Outbound，AP 分配了一个 4 G 的 BAR 给 GPU，这个一般用来干什么的？ 212
枚举 213

枚举规则.....213
721. PCIe 枚举跟 PCIe 初始化是一个意思吗？.....213
722. BIOS 初始化 PCIe 设备以后，在进入到 OS 时 OS 还要对 PCIe 设备进行一次枚举吗？这两次枚举的主要差异是什么？.....213
723. RC 枚举之前怎么确定自己的 ID 的？.....213
724. RC 发的 Cfg 包的 Requester ID 中 Device Number 和 Function Number 只能是 0 吗？.....213
725. EP BDF 的 Function Number 一定大于等于 RC Requester ID 的 Function Number 吗？.....213
726. 枚举过程中，RC 发现某 EP 为 Single Function，在继续扫下一个设备的时 BDF 的 Function Number 不能再继续加 1 了吧？.....214
727. ARM64 如果走 ACPI Scan Root 流程去枚举，还会走到传统 Probe 流程的枚举吗？.....214
728. 基于 ECAM 机制访问 EP PCIe 时，EP 收到的 TLP 是 Cfg 还是 Mem？.....214
729. 枚举过程中，主机会扫描 PCIe Switch 内部虚拟桥并读取 P2P 桥的 Vendor ID，该读请求位于总线上，下游所有 P2P 桥都会收到这个信息，是如何知道哪个是 Device0、哪个是 Device1 的？.....214
730. 枚举完毕之后，EP 设备的 BDF 信息是保存在配置空间中吗？.....214
731. 配置空间里的 Device ID 跟 BDF 中的 D 是一回事吗？.....214
732. 枚举好的 BDF 信息放在哪里？.....214
733. 对于 RC-EP 点对点的 PCIe 拓扑，EP 发起 MRd Request TLP 中 Requester ID 位置的 Bus 号是 0 还是 1？EP 硬件知道自己所处的 Bus 吗？Bridge 是有寄存器记录 primary Bus，EP 中是否有这样的寄存器？.....215
734. PCIe 接多个不同的 Device，是软件可配置还是硬件枚举固定不变的呢？.....215
735. 捕获 Bus Number 怎么理解？在枚举过程中，EP 收到的都是 Type 0 吧？这个时候自己的 Bus Number 还是零，应该匹配不上，捕获了以后怎么用呢？.....215
736. EP 在一条链路一端是只有一个设备 Device0 吗？比如阵列卡，是将每个 SSD 当做一个 EP 吗？.....215
737. 分配 BAR 空间的时候，如果 Device 只需要 512 KB 的空间，我能分给它 1 MB 吗?.....215
738. 对于 RP 和 DSP 而言，在枚举之前不知道其下方为 EP 还是 Bridge 的 USP，扫描设备树时软件如何知道该发 Cfg0 还是 Cfg1？.....215
739. Hot Reset 会影响此前枚举好的 BDF 表吗？.....215
740. 软件上对 PCIe 设备做 Remove，在硬件上有什么动作？枚举树会删除这个节点吗？.....216
741. 如果是 RC-EP 直连的场景，枚举过程可以直接跳过去吗？.....216
枚举故障.....216
742. 系统中只有一个 RC 直连一个 EP，枚举时 RC 可以从 Bus 编号 0 以外的数字开始枚举呢？为什么仿真时发出的 BDF=0x100 即 Bus Num=1？.....216
743. 挂在桥下的设备在上电过程中出现下述 BAR 分配错误，可能是什么原因？.....216
744. 扫描设备时报错 Invalid BAR 是怎么回事呢？.....217
745. lspci 查看和启动虚拟机设备，枚举过程中报 Unknown Header Type 0x7f，Header Type 不正确，一般是什么原因导致的？.....217
746. x86 CPU 在 BIOS 的 PCI 枚举时找到了设备，但进入 Kernel 后没有这个 PCI 设备。难道是进入 Kernel 后会重新枚举吗？.....217
747. 高通主板，RC 主动 Scan 热插入的 EP，PCIe Rescan 之后提示 EP BAR No Space，实际有空间指示 BAR Space 较小，是何原因？.....217

748. PCIe 设备起来后，RC 端 CPU 通过 lspci 查看，出现 Advisory Non-Fatal Error，大概是什么原因？标准 x86 服务器 Intel 网卡没用这个错，用自研网卡（Xilinx PCIe IP）出现该错。上电后系统刚起来就有几百个错，后续业务阶段没用新的错误产生。 217
749. 华三服务器某特定槽位插上自研网卡能够正常 Link Up，但 PCIe 枚举时候服务器卡死，其可能是什么原因？ 219
系统应用 219
启动问题/BIOS 219
750. 重启服务器，PCIe 卡在 BIOS 阶段，用分析仪抓包发现 Host 重启后未下发任何 Cfg 包，该问题有何 Debug 思路？ 219
751. PCIe 开了 OPROM BAR，BIOS 里面有配置可以关掉不读吗？ 219
752. FPGA 开发板插在电脑 PCIe 插槽，电脑黑屏无法启动，可能是什么原因？ 219
753. 系统突然挂了是怎么回事？ 219
754. PCIe 板卡往服务器发数据，导致服务器重启，可能是什么原因导致的？ 220
755. 在 Boot 阶段，CPU RC 除了配置 PCIe 外设 Link Up 之外还会有其他的配置吗？比如配置 PCIe 外设 EP 的 MSI-X Table。 220
756. 成群的服务器如果某个芯片的 A53 死了，我虽然能通过 SMBus 用远端 HostCPU Reset A53，但 PCIe 本身是不是没法重新识别呀？ 220
ROM 220
757. PCI Expansion ROM 这个功能怎么用？需要真实例化一个 ROM 吗？ 220
758. EP 的驱动现在还会从 Expansion ROM 去读取吗？ 220
写合并 221
759. PCIe EP 需要配置什么 Capability 来支持 Host 写合并（Write Combine，WC）？ 221
760. 写合并是什么意思？ 221
761. 为了实现 Host 写合并，PCIe EP 需要配置什么 Capability 吗？ 221
lspci 222
762. lspci 结果中，Region 有时是 Virtual、有时是 Disable，表示什么意思？ 222
763. 分配出来的 IO Mem 空间为什么被标记为了 Disabled？ 222
764. 用 lspci 查看次级 PCIe 扩展能力结构不显示详细字段，可能是什么问题？ 222
765. Windows 下有没有好用类似 lspci 的工具，能够读到超过 0x100 的配置空间？ 222
拓展应用 223
766. CPU 下挂的 NVMe 硬盘容量大小有限制吗？容量大小跟物理地址空间有关系吗？ 223
767. PCIe 的 U.2、U.3 是什么意思？还有 U.2/U.3 的 Single、Dual 是什么意思？ 223

八、电气物理层 224

物理特性 224
768. PCIe Edge Connection 金手指 PIN 脚中的 HSIp、HSOp、Tx、Rx 的方向是相对谁来说的？HS 是什么意思？ 224
769. Synopsys 公司有个叫 VGA 的特殊 PIO，这个是做什么用的？ 224
770. 差分信号 16 GT/s 要求 85 欧姆、8 GT/s 及以下要求 100 欧姆的目标阻抗，这在同一块板子同一组硬件 Trace 下如何实现？是否需要在 16 GT/s 时并电阻进去？ 224
771. PCIe 在 Electrical Idle 时差分对上的电压是多少？ 225
772. PCIe 的 AC 耦合电容为什么多放在发送端？ 225
773. 金手指到 PCIe 芯片的走线长度有要求，板上互联的 PCIe 走线长度有要求么？ 225
774. PCIe PHY 需要用到 Reference Voltage 吗，比如给 vco 一个基准电压什么的？ 225

775. 只用 Serdes IP 可以两边对传不包含任何协议的高速数据吗？ 225
776. PCIe 协议中除了 CEM 规范还有哪个规范定义了 PCB 设计的插损和回损的要求？ 225
777. PCIe 7.0 物理层支持光口，为何不直接走光通信？PCIe 不就淘汰了吗？ 225
778. PCIe5.0 时 x4 槽位功率还是 25 w 吗？ 226
电气特性 226
779. 下图为 Gen1 Rx 的眼图，这个电压幅值 1 V 是不是过高了？ 226
780. PCIe 链路信号质量不好时，实际一般调哪些参数来提高信号质量？ 227
781. 如果 Local Tx 的 FS 没有设置默认参数也没有寄存器可配，那么其 FS 是怎么获取的？ 227
测试 227
测试方法 227
782. PCIe 是如何检测到对端有设备存在的？是靠外围电路检测然后告知 PHY，还是 PHY 内部自己处理？ 227
783. PHY 是怎么在训练过程中评估 BER 的？是通过 PRBS Pattern 吗？ 228
784. 如何统计 PCIe 的误码率？有何标准方法吗？ 228
785. 测 PCIe 链路上的眼图需要发特定 Pattern 吗？必须在 Loopback 下测吗？ 228
仪器设备 229
786. 常用的 PCIe 测试设备有哪些？ 229
787. 协议分析仪是什么？是否需要购买硬件设备？ 229
788. Teledyne Lecroy PCIe Protocol Analysis 是软分析仪吗？是否需要厂家配套的硬件设备？ 229
789. Teledyne Lecroy PCIe Protocol Analysis 软件在哪里可以下载到？ 229
790. 力科分析仪抓包抓到很多 TS，该如何过滤掉这些 TS 序列？ 229
791. 协议分析仪抓取硬盘 Trace 数据，显示 TLP 无效或 LCRC/ECRC 错误，该如何分析错误产生的原因？ 230
792. 能否从分析仪抓的报文里看出是否有 Warm Reset？ 230
793. Xilinx DMA 发送 IO 写请求，接口输出波形但 Analyzer 抓不到包，可能是什么原因？ 230
794. 原型验证环境中降速测试，Host 超时报错，分析仪抓出下述错包，是何原因？ 231
795. 抓取 Trace 数据时只抓到下行的 TLP 数据，抓不到上行 TLP 数据，这是怎么回事呢？ 231
796. 分析仪抓到了 Link Up、Link Down 等 Link Event，这个 Event 是怎么来的？ 231
797. PCIe 5.0 协议分析仪大概什么价格？ 232
798. 有哪些厂家做 Gen5 协议分析仪？ 232
799. PC 接自研 EP，用分析仪解析建链过程中 PC 发来的包全部出错，可能是什么原因？ 232
800. 协议分析仪跟设备间是怎么连接的？ 232
BIST 232
801. PCIe PHY BIST 是什么？ 232
802. Vendor 提供了单独测 PHY BIST 的程序，在 SoC 上怎么用 CR 口测试 PHY BIST？ 232
803. PCIe BIST 电路测试的是 PCS 还是 PMA？ 232
Loopback 233
804. PCIe 有哪几种 Loopback 模式？ 233
805. 接误码仪进行 RX BER 测试时，Gen1 速率进不到 loopback 模式，但 Gen2、Gen3 可正常进 loopback，可能和什么有关？ 233
806. PHY 有专门的内外环测试去检验 PHY 是否正常，控制器有什么类似的方法吗？ 233
807. 有 Loopback BIST implementation in a typical PCIe Interface 相关博客可以看吗？ 233
808. 用误码仪对 Synopsys 公司 PCIe EP 进行 Loopback 测试，测试 PCIe Rx 时，是只在 PHY 内部 Loopback，还是包含控制器一起做 Loopback？ 234

809. PCIe EP Tx 的 Loopback 是怎样测试的？是测 PHY 的吗？ 234
810. PCIe RC Rx 的 Loopback 测试是怎样的？ 234
811. 下图是系统主板 Rx LEQ 测试组网图，其中的 Loopback 是哪种模式？ 234
812. Loopback 模式下 Link Up 状态应该是 1 还是 0？ 235
813. 本地 Rx 回环的时候恰好碰到对端 Tx 发送 TS1 怎么办，是有什么仲裁机制吗？ 235
Compliance 235
814. 只要使能 LTSSM，协议分析仪就可以直接发 TS 进 Compliance 模式吧？ 235
815. Compliance 只能在 Gen1 时候做吗？ 235
816. 对 EP 卡进行 Compliance 测试时，RC 该如何读写 Compliance 状态的 EP 卡的寄存器对 EP 进行配置？ 236
817. PCIe Compliance Pattern 指的是哪些 Pattern ? 236
818. Compliance Test 是不是不区分 RC 和 EP? 236
819. Serdes 上不应避免连 0 连 1 吗，为何测量 8.0 GT/s 及以上速率的 Tx 均衡参数时可以发送 64 个连 0 连 1 的低频 Pattern？ 236
820. 发生连 0 连 1 的 Compliance Pattern 测 Tx Preset 时，会经加扰再发出吗？ 236
821. Compliance Pattern 是只有在 Polling.Compliance 下才可以发的吗？ 236
PCI-SIG 认证 237
822. 如何做 PCI-SIG 认证？ 237

九、虚拟化 238

协议规则 238
823. PF 和 VF 该怎么理解？ 238
824. 启用 SRIOV 时，一个 Bus 下可以挂多个 SRIOV 的 EP 吗？ 238
825. 一个 VF 的 TLP 是怎么经 Switch 路由到这个 EP 的 VF 的？Switch 上的 Mem Range 好像不包含这个 VF Address。 238
826. Scalable IOV 和 SRIOV 是完全不同的东西吗？ 238
827. 多 VF 时，RC 可以发 Cfg 请求来访问 VF2 的 Config Space Header 相关寄存器吗？ 238
828. Multi-Host 是何概念？一个 x16 的物理链路上下挂两个 x8 的 EP 是否为 Multi-Host？ 238
829. MR-IOV 后面是不用了吗？ 239
应用问题 239
830. 有场景需要 RP 支持多 PF/VF 吗？ 239
831. PF 的 SRIOV Capability 寄存器里只有一组 VF BAR 寄存器，那么开启 SRIOV 后如何配置每个 VF 的 BAR 空间？ 239
832. 为了支持 SRIOV，RC 有什么特别的要求吗？比如有没有必须要支持的 Feature？ 239
833. Synopsys 公司的 DMA 支持 SRIOV 但不支持 TLP Prefix，PASID 是在 TLP Prefix 里的，这是否意味着不能使用 SMMU Stage1 的翻译了？ 239
834. 未开启 SRIOV 时，多个虚拟机可以共享同一 PCIe 设备资源吗？ 239
835. Intel 还支持 Requester ID 方式的 SRIOV 吗？ 240
836. 协议规定 SRIOV 中 VF 的 Vendor ID 为 16'hffff，软件如何判断 VF 的存在？ 240
837. Windows 驱动支持 SRIOV 吗？ 240
838. PCIe SRIOV 能力结构中 VF Offset 和 Stride 是设备 EP 决定的还是 Host 上的 RC 决定？ 240
839. VF 数目最大不是 256 么，为什么有时软件能扫描出上千个 VF？ 240
840. 理论上 VF Driver 读不到 PF 配置空间，其是通过什么机制在 PF SRIOV Capability 中拿到 VF BAR 信息的？ 241

841. lspci 结果中，设备 BAR 空间出现 Virtual 标志是怎么回事？ 241
842. 手动使能 SRIOV 功能后用 telscan 软件重新扫描设备，仍然只能扫到 PF 扫不到 VF，可能是什么问题？ 241
843. Synopsys 公司的 PCIe Device 最多支持 4 K 个 Function，这是如何做到的？开了 ARI 不也才最大 256 个 Function 吗？ 241
844. 为什么读不到 VF 的 BAR Size，也配不进去？ 242
845. 如果 EP 需要支持 SRIOV，一般需要预留几个 VF？ 242
846. 下图中每个 Device 中只有一个 Function。一个 Device 可以有多个 PF 吗？是不是只有 Dual Port 才支持多个 PF？ 242
847. 市场上真正实现了 MRIOV 的产品有哪些？ 243

十、地址转换 244

ATS 244
848. 如何在 RC 上验证自家 EP 芯片上的 ATS 功能？ 244
849. PCIe 4.0 的 IP 建链之后工作在 Gen1 速率，还能用 ATS 等较高版本 IP 才支持的功能吗？ 244
850. lspci 能查看设备的 ATS 能力吗？为什么在 ContextX-7 上看不到 ATS Capability？ 244
851. lspci 显示 Intel 网卡支持 ATS 但 ATS 未打开，该如何打开？ 244
852. RC 和 EP 是不是都可以用 ATS？比如 Dual Mode PCIe IP，作为 EP 时支持 ATS，在作为 RC 时还能用 ATS 吗？ 244
853. ATS 转换请求中的 Length 可以设置为转换请求长度的两倍，这句话怎么解读？ 245
854. 设备本地 ATC 用 RAM 来存储被转换后的地址，这样一来当发起请求进行地址匹配时，花费很多个 cycle 去遍历 RAM 里的地址，效率较低，有什么好的解决方案吗? 245
855. 目前哪些产品支持 ATS？有支持 ATS 的显卡网卡吗？比如 Nvidia 显卡和 MLNX 网卡？ 245
856. PCIe 的 ATS 地址转换是 IP 自己主动去发然后存到 ATC 里面吗？ 245
857. ARM 架构有 ATS 吗？ 245
858. 市面上有什么设备支持 ATS 并且内部带 ATC 的吗？ 245
859. PCIe Controller 回 ATS Completion 时，S bit 是 IOMMU 还是 Controller 置起的？ 245
ACS 246
860. PCIe ACS 这个功能重要吗？ 246
861. 如何配置如下 ACS？ 246
862. 技嘉服务器 BIOS 内已开启桥和 ACS，但 lspci 查询结果显示 Switch 未开启 ACS，该如何配置 Switch 上的 ACS？ 247
863. 有哪些支持 ATS、ACS 的设备？ 247
864. 同一个 Switch 下多卡之间进行 P2P 访问时必须开启 ACS 吗？ 247
865. ACS 是如何判定某个请求是否为 P2P 请求？ 247
PRS 247
866. 有了 ATS 为什么还需要 PRS，碰到 Miss 后直接等到页表分配好再返回请求不行吗？ 247
PASID 247
867. PCIe 协议的 PASID 是来干什么的？ 247
868. ATS 里的 PASID 怎么用？ 248
869. PASID 和 Bus Device Function 编码有什么关系？正常 BDF 不是已经可以定位到某个具体的 Function 了吗？ 248
870. 现在有什么能测 PASID 的 Function 吗？ 248

871. IOMMU 方案中，有只用 BDF 的，有在 BDF 基础上启用 PASID 的，是因为只用 BDF 会导致某些情况下地址映射出问题、需要用 PASID 来弥补缺陷吗？还是说只用 BDF 没问题、PASID 从另一个角度完善了地址映射功能？ 248
872. EP 发 TLP 时其中的 PASID 是谁给的？ 249
873. PASID 有何用途？ 249
874. EP 的 DMA 流量里 TLP 需要带上 PASID 吗？ 249
875. DMA 流量里有无 PASID 在效率上有区别吗？ 249
876. 不开虚拟化时，EP DMA 发的是 PA 吗？ 249
877. EP 如何知道自己的 PASID 值是多少？软件是如何通知 EP PASID 值的？ 249
MMU 250
878. ATS、ACS、IOMMU 是什么关系？ 250
879. IOMMU 是什么，跟 MMU 有何区别？ 250
880. 用 DMA 做 EP-EP 间 P2P 通信需要考虑 IOMMU 吗？ 250
881. 若系统中没有虚拟机，IOMMU 在地址翻译中是否也发挥了作用？只是因为虚拟地址和物理地址相同，看起来想没翻译？ 250
882. 在服务器上的测试 IOMMU 时，EP 开启 ACS 后系统的 IOMMU 才生效，没有报错，可能是什么原因？ 250
883. VA->PA 地址转换过程中，什么场景下不能跳过 Stage1、必须先做 GVA->GPA 呢？ 250
884. PCI EP Framework 中的 EPC Addr Space 是指什么？ 251
885. 包括查找表在内的 IOMMU 功能实现是单独一个芯片还是集成在主机芯片内呢？ 251
886. 开启 SSMU 时，EP 发出的地址是先比较 Base/Limit 在哪个 RP 下，还是先送到 SMMU 做转换？ 251
887. PCIe + SMMU 方案什么样的 SoC 用的多？ 252
888. SMMU 读 STE 报 C_BAD_STE Error，可能是什么原因？ 252
ATU 253
889. EP 是如何获知 RP 侧的地址的？ 253
890. ATU 全称是什么?这个是哪家公司定义的名称？和 MMU 有何区别？ 254
891. 对于 Synopsys 公司的 IP，在不开启 iATU 的情况下，访问 ECAM 空间或者 MMIO 空间，Controller 内部是怎么组包的呢？ 254
892. 发送不同包的时候 TLP Header 的有些字段可能会有变化，如果不使能 iATU，是否有某种修改机制？还是说硬件采用默认值？ 254
893. EP 要访问 RP 存储器域，EP 如何知道 RP 存储器域的地址对应 RP PCIe 域什么地址？ 254
894. Inbound ATU 有两种模式：BAR Match Mode 和 Address Match Mode。在设备测，这两种模式都可以配置多条映射吗？还是 BAR Match 模式，一个 BAR 只能映射一项？ 254
895. SNPS PCIe Controller 作 EP 时，Outbound ATU 有何应用场景？ 254
896. iATU 开启时，Mem 包是先做 BAR Mapping 还是先做 iATU 转换？ 254

附、专题 255

一致性 255
897. Prefetchable 和 Non Prefetchable 有何区别？ 255
898. RC 在读写 EP BAR 空间时，TLP 中 Attr[0] Snoopable 属性一般是怎么做的？ 255
899. 有文章说 PCIe DMA 时建议将 No Snoop 属性设置为 1，即不做 Cache Snoop，而是由软件来维护 Cache 一致性。为什么？ 255
900. 有实际场景需要把 PCIe 的 No Snoop 属性配置为 0 吗？ 255

901. No Snoop 属性是谁设置的？是硬件默认还是驱动设置？EP 的驱动可以设置 RC 的吗？ 255
902. 可以直接 setpci 设置 No Snoop 属性吗？ 255
903. 该如何理解 PCIe No Snoop 的优势？ 256
904. 什么时候会用 No Snoop，什么时候不用？ 256
905. DMA 为什么多用 No Snoop？是不是意味着内核 DMA 部分会提前 Flush Cache、Pin Page，所以才不需要 Snoop？ 256
906. EP 使用 No Snoop 时，EP 端的每个请求都必须带 No Snoop 标签么？ 256
907. Snoop 默认开的还是关的？ 257
908. PCIe 中什么场景需要用到 Snoop？ 257
909. 使用 DMA 时是是否都要设置 No Snoop = 1？是使用 No Snoop 好还是 Snoop 好？ 257
910. Snoop=1 时写 Partial Line 和写 Full Line 跟 AXI ID 有什么关系呢？ 257
911. 如果 AXI 不支持 IO-Coherence，那么 PCIe 的 Snoop Feature 也是不是也不起作用？ 258
912. PCIe Prefetchable 和 Non Prefetchable 空间的区别在哪里？ 258
913. Type 1 字段里面没有 Non-Prefetchable 设置，是否意味着 Switch 下方的 EP 不能有 64b Non-Prefetchable BAR？ 258
914. 相对于 Non Prefetchable BAR，Host 读写 Prefetchable BAR 的行为具体是什么样的？x86 上这个内存属性的设置是不是根据这个 BAR 属性自动设置的？ 259
915. NVMe 盘一般发出读写是 Snoop 的还是 No Snoop 的？ 260
916. EP 把 RC 端的 DDR 颗粒当成 EP 自己的内存使用，即 EP 把自身 PCIe 空间当作 Non-Cacheable Memory 去用，在该地址段内做 memcpy、memset、创建结构体等操作。当结构体内存在 char 类型的成员时，对该结构体内多个成员跳着赋值时出现写不进去的情况，可能是什么问题？如何解决的？ 260
PCIe 性能 261
带宽 261
917. 一般如何测试纯 PCIe 的速率？ 261
918. Gen5 x16 Device 往 DDR 方向发 64 B 的 MRd，性能一般多大？ 261
919. PCIe TLP Posted 写没有反馈消息，如果 CPU 一直写把写带宽打满了会怎样？ 261
920. 测试发现按 MPS/MRRS 拆包性能较低，PCIe 怎么拆包性能好点？ 262
921. 理论带宽 7.8 G 的 PCIe Gen3 x8 设备实测 DMA 写带宽只有 4.6 G，可能是什么原因？ 262
922. NVLink 带宽为什么比 PCIe 大这么多？ 262
923. PCIe3.0 x4 实际性能在两个方向分别大概能跑到多少？有 80%利用率吗？ 262
Latency 263
924. PCIe 收发两端的延迟高达 100 ns 以上，这个延迟的主要贡献者在哪里？物理层和数据链路层只占很少一部分吧？ 263
DMA 263
925. PCIe 的 DMA 为何在 EP 端呢？有何好处吗？ 263
926. 一般 EP 端应该都带 DMA，那还需要 RC 端的 DMA 吗？ 264
927. 对 EP 而言，使用 Host 端的 DMA 与使用 EP 自己的 DMA 搬数据两者效果一样吗？ 264
928. PCIe Core 里有 DMA 吗？是协议规定的吗？ 264
929. PCIe DMA 某 Channel 传输异常后不能继续使用了，请问有什么复位的办法吗？ 264
930. HDMA 这个 Feature 用的场景多吗？什么场景会用？ 264
931. Synopsys 公司的 PCIe 用 HDMA 搬数据时需要用到 iATU 吗？ 264
932. DWC 的 PCIe EP Driver 把 DMA 的 Link-List Descriptor 放在 EP BAR 空间，为什么不放在 Host Memory 上？ 264

933. EP 通过 DMA 往 RC 的 DDR 搬数据，搬完最后一笔之后通知 RC，EP 如何保证完成中断到达 RC 之前写数据已经到达了 RC DDR？是否存在可能，RC 收到 DMA 完成中断时后，RC 的 CPU 开始读，但 EP 来的这笔写数据还在路上、还没到达 DDR？ 265
934. 在驱动程序里面将 DMA 描述符配置好写到 DDR 里面，CPU 通过 PCIe 总线写 BAR0 地址数据来配置并开启 DMA，之后写不进去，出错之后就卡死了，可能是什么原因？ 265
935. 基于内核驱动端的 DMA 传输函数计算带宽，传输数据量增大但传输时间无明显变化，可能是什么问题？ 265
936. 使用 DMA 引擎时，软件发 Doorbell 告知 DMA 去搬数据，那怎样控制发的 Doorbell 是 Active 还是 Inactive 的？ 266
937. PCIe 网卡用 DMA 搬移数据，数据会封装成 TLP 进行传输吗？ 266
938. 用 DMA 做 Mem2mem 链式数据搬运，每个描述符搬 1 M 或更大数据量时会卡住，可能是什么原因导致的？ 266
939. EP DMA 不同 Function 发出相同地址的读写 Mem 包，最终映射到内存地址相同吗？ 267
940. EP 初始化后做 EP 通过 DMA 读 RC Memory，开始时 RC 正常回 CplD，过了一会儿后 RC 就不回 CplD 了，可能是什么情况？ 267
941. 用 DMA 往 Host 搬数据，相同 PCIe 链路速率情况下链宽 x8 的性能是否是 x4 的两倍？ 267
942. 用 DMA 往 Host 搬数据，相同 PCIe 速率时链宽 x8 带宽性能相较 x4 提升不明显，可能是什么原因？ 267
IP 设计与仿真 268
设计方法与理解 268
943. Synopsys PCIe Controller 的 sys_aux_pwr_det 输入信号是从哪里拉过来的？ 268
944. PCIe 配置空间本身已有 4 KB，加上自定义的寄存器后不就超过 4 KB 了吗？ 268
945. 请问 Synopsys 家 IP 中哪个寄存器记录了当前链路状态是不是 L1.1/L1.2？ 268
946. Synopsys PCIe Controller 寄存器分下述三类，其中 ViewPort 该如何理解？ 268
947. Synopsys 公司 Controller 配置空间里的 Shadow Register 是做什么的？ 269
948. Synopsys PCIe 配置空间里的 Shadow Register 跟常规影子寄存器的定义有何不同吗？ 269
949. Synopsys PCIe 内 DMA 相关寄存器是否不能用 Cfg 包访问只能用 Mem 包访问？ 269
950. 请问 PCIe 配置空间 0x700 后面这些配置是什么呢？ 269
951. 为什么 ELBI 访问要给 BAR Num 的信息？ 270
952. Synopsys PCIe VIP 可以用协议分析仪进行分析，需要有协议分析仪设备吗？ 270
953. Synopsys 公司的 PCIe Controller 用 AXI Bridge 时，发送 NP 请求中的 Requester ID 和 Tag 放在 AXI ID 上吗？ 270
954. 既然 ATU 能填充 Function Number，为什么还需要 axmisc_info？ 270
955. Synopsys PCIe Controller 作为 EP，配置 IP 时将 Controller 的内部寄存器映射到了 BAR0，那么 Host 访问 Controller 寄存器中的 Capability 寄存器时是用 Cfg 还是 Mem 呢？ 270
956. 1 G 速率下把 PCIe 的 Timing 过约 30%，即只有 0.7 ns 可用，这样是否过于苛刻？ 271
957. Synopsys PCIe IP 给 MSI-X Table 留了 8 K DW 空间，为何经 DBI 只能访问到 1 K DW？ 271
958. Synopsys 公司的 PCIe IP 支持在 IP 里注错吗？ 271
959. AXI 在 PCIe 里是做什么的，为什么很多 PCI IP 用 AXI？ 271
960. PCIe 对 AxPROT 有什么要求吗？ 271
961. ECAM 是必选功能吗？ 271
962. Synopsys 公司的 PCIe EP 发出 MRd，RC VIP 反馈 Cpl.UR 后 EP 把这笔 Cpl 丢掉了，发起 MRd 的 AXI Master 等不到 Rsp，只能等 Cpl 超时吗？ 272
963. Synopsys 公司有 x1 的 PHY 吗？只需要实现 x1，用 x4 的 PHY 好浪费面积。 272

964. PCIe VF 中的寄存器可以用 DBI 访问吗？ 272
965. Xilinx 的 PCIe PHY 连标准的 PCIe Controller，Xilinx 的 PCIe PHY EQ 接口非 PIPE 标准，需要自己设计逻辑转换吗？ 272
966. Synopsys 公司 PCIe IP 的 Port Logic 寄存器中到底包含 iATU 和 DMA 寄存器吗？ 272
967. Synopsys 公司 PCIe DMA Descriptor 中的 SAR 目标地址是 RC 端的 PCIe Bus Address 还是 EP 端的 Bus Address？ 272
968. Synopsys 公司 PCIe Controller 中，Dedicated 和 Shared AXI DBI 模式有什么区别？ 273
969. 一般什么时候用 PCIe 的 Dual Mode 模式？ 273
仿真问题 274
970. 对于 Synopsys PCIe IP，Driver 发起 Mem 请求时 TLP Header 中的信息可以直接从 AXI Bus 拿到吗？该如何提供？ 274
971. Synopsys 的 PCIe VIP 中 Unified 的 VIP 有没有一些说明文档呀？ 274
972. Synopsys PCIe VIP 的 Unified TB 中，选择 Serdes 连接时参考时钟应接到哪个接口上？ 274
973. PCIe VIP Unified TB 中，如果想要连接 DUT 去替换 Root 的话，是不是仍然要实例化 Root 和 Endpoint，只是将 Root 的 connect_active_vip 置 0，Env 中仍然要例化 Root 的 Agent 和 Cfg？ 274
974. Synopsys 的 PCIe VIP 可以支持多 Port 吗？就是一共 8 Lane 分为两个 x4 Port。 274
975. 按照 Synopsys 公司的 VIP 手册写了 Topology File 并例化、连接了 DUT，编译时报下述[XMRE]错误是何原因？ 274
976. Synopsys VIP 中 PIPE 接口 pclkreq_n/pclkack_n 的这两个信号是什么？ 275
977. PCIe 仿真 Link Up 后仿真日志报“Detected EIOS, Receiver entering L0s”，是什么原因？ 275
978. Synopsys 公司 PCIe Controller + Xilinx PHY 接 Synopsys PCIe VIP 做 EDA 仿真，LTSSM 走到了 Polling.Active 但 Xilinx PHY 的输出仍然是 High-X，可能是什么原因？ 275
979. 如何用 PCIe VIP 发起 Stash RNF 访问？ 275
980. 请问 Synopsys 的 VIP 可以发 Cpl 包吗？ 275
981. VIP 报 ECRC 错误，但数据是对的，这个错误可恢复吗？ 275
982. Synopsys PCIe VIP 如何使用 Shadow Memory 来检查写读相同地址的数据是否一致？ 276
983. 用 SNPS PCIe VIP 仿真，有办法从 Transaction Log 上算出收发数据量吗？ 276
984. Synopsys PCIe VIP 如何让 EP 发出 Cfg 报文？ 276
985. Synopsys PCIe VIP，发 MRd 后 Shadow Memory 报错是什么原因？ 276
986. Synopsys PCIe VIP，如何关掉收到 CA 后的 appl_driver_missing_good_status 报错？ 276
987. Synopsys PCIe VIP，在添加频偏时 Serdes PLL Unlock，该如何解决？ 276
988. Synopsys 公司的 PCIe VIP 可以用 Cadence 公司的仿真工具吗？ 276
PIPE 277
989. PIPE 接口上的 rx valid 和 rx data valid 的区别和关系是什么？ 277
990. PIPE 接口 Datak 指示 Datad 哪 8 bit 有效，但对 8b/10b 编码后的 10 bit 该如何表示？ 277
991. Synopsys 的 IP 是有 Gen5 速率的，为什么 PIPE 协议在 5.1 和 6.1 版本都没有定义 Gen5 速率 FS/LF 的寄存器？ 278
992. Synopsys 公司控制器接 Xilinx GTH PHY 只能用 16b 的 PIPE 吗？能支持 32b 吗？ 278
993. FPGA 平台上 Synopsys 公司 Controller 搭配 Xilinx PHY，PIPE 两边均衡算法不一致的问题是如何解决的？ 278
994. 使用 Synopsys PCIe 控制器和 X 家 PHY 做 FPGA 版本，两者之间需要做哪些处理？ 278
995. PIPE 口上的 TxDetectRx 已经起来了，为什么 LTSSM 没有退出 Detect？ 278

996. PIPE 6.1.1 协议新增 Recalibration 允许 PCIe 控制器进入 Recovery 对 PHY 重新进行校准，但是 PCIe 规范中没有涉及这个行为，这块该怎么理解？ 278
997. Gen3 及以上速率时 SerDes Arch PIPE 接口信号中的 TxData 中每 10b 只用 8b 吗？ 278
应用 279
998. 手机用 PCIe 连接什么外设？ 279
999. 如果芯片 DDR 有问题，可以使用 PCIe 接的 SSD 临时充当 DDR 用吗？ 279

致谢 280

一、拓扑结构

拓扑组件

1. PCIe RC/EP/RCiEP 是什么？

   RC，Root Complex，根复合体，PCIe 树的根，连接 CPU、主存及 PCIe Bridge/EP/Switch 等。

   EP，Endpoint，端点，具体的 PCIe 设备，比如带有 PCIe 接口的网卡、存储卡等。

   RCiEP，集成了 EP 的 RC。

2. EP 是不是没有 Downstream，只有 Upstream？

   EP 就是 Upstream Port。

3. Native EP 和 Legacy EP 是什么？

   EP 分两种：Legacy EP 及 Native EP。Legacy EP 是指原本给 PCI-X 总线接口设计的 EP 被改为 PCIe 总线接口 EP。Native EP 是指标准的 PCIe EP。Legacy EP 支持部分 Native EP 禁止的操作，Native EP 全部通过内存映射来操作。两者比较如表所示。

   ▼ 表 Legacy EP Vs. Native EP

	Legacy EP	Native EP
是否支持 IO 事务？	支持	不支持
是否支持锁定事务？	支持（完成者） 不支持（请求者）	不支持
是否支持 64 位寻址？	可以不支持 64 位寻址	对于可预取的存储空间必须支持 64 位寻址，不可预取的存储空间允许映射到低 4 GB
支持的中断类型	支持 INTx 必须支持 MSI	只支持 MSI

4. RC 内可以有多个 PF 吗？

   RC 比较抽象，里边可能包含多个 iEP，即 RCiEP，这样 RC 内可以有多个 PF。如果指 RP，那么没有多 PF。

5. RCiEP 该怎么理解？

   SoC 内部互联的 EP，比如网卡 SATA 控制器，没有 PHY 和 Link，不需要 Serdes 传输。

6. PCIe 伪端口（Pseudo-Port）是什么？

   PCIe 伪端口，顾名思义，假的端口。伪端口仅具有 PCIe 端口的部分属性。比如 Retimer 的 Port 就是伪端口。Retimer 的 Tx 和 Rx 只有物理层部分，没有也不需要数据链路层、传输层。

7. M-PCIe 是什么？

   M-PCIe 是面向移动设备等功耗敏感型设备 PCIe 协议，即 Mobile-PCIe。

   M-PCIe 与标准 PCIe 的 TL、DL 层相同。相比于标准 PCIe，M-PCIe 引入了 MIPI M-PHY。M-PHY 能够进行快速的电源状态切换，从而降低功耗。M-PHY 仅在实际传输时处于最大功耗状态，Burst 传输完毕进入低功耗状态“STALL”，紧接着退至最低功耗“HIBERN8”状态。此外，M-PHY 支持非对称链路，允许链路上存在不同数量的收发器。

   M-PCIe 功耗更低，但其支持的速率不如标准 PCIe 高。M-PCIe 支持 Gear1~3 三种速率，为 1.25 GT/s, 2.5 GT/s 及 5.0 GT/s，Gear2/3 分别对应 PCIe Gen1/2 速率。

8. 对于 NUMA Chip，通常是每个 Node 对应一个 RC 吗？

   这个不一定，要看产品组网的具体需求。个人理解通常不会每个 NUMA 设计一个 RC，RC 的需求不是随着 NUMA 产生，而是根据网络需求，尤其是否有 GDRGDS 需求来定。普通云网络一个 RC 即可，当然该场景设计上需要尽量避开跨 NUMA 的 mem 访问。

Hierarchy ID

9. 什么是 PCIe Hierarchy？

   PCIe I/O 互连的树形拓扑结构称为 PCIe 的 Hierarchy，或称层级、层次（不是事务层、数据链路层的“层”）。层次区域是指与 RC 某一 RP 相关联的所有设备和链路组成的线路结构，单个 PCIe 系统内允许存在多个层次区域。

10. 为何需要 PCIe Hierarchy ID？

    在初始化 PCIe Hierarchy 的时候，软件给每个组件分配 Bus、Device 和 Function Number，以确保每个 Function 在该 PCIe 树中都有一个独一无二的 Routing ID。

在大型 PCIe 系统或 CPU 集群系统中，树多层级多，仅采用原有的 BDF Routing ID 无法唯一确定某个 Function。为此，采用在 Routing ID 上添加额外信息的方式来产生唯一的 ID，即 Hierarchy ID，在庞大的系统中唯一标识一个 Function。

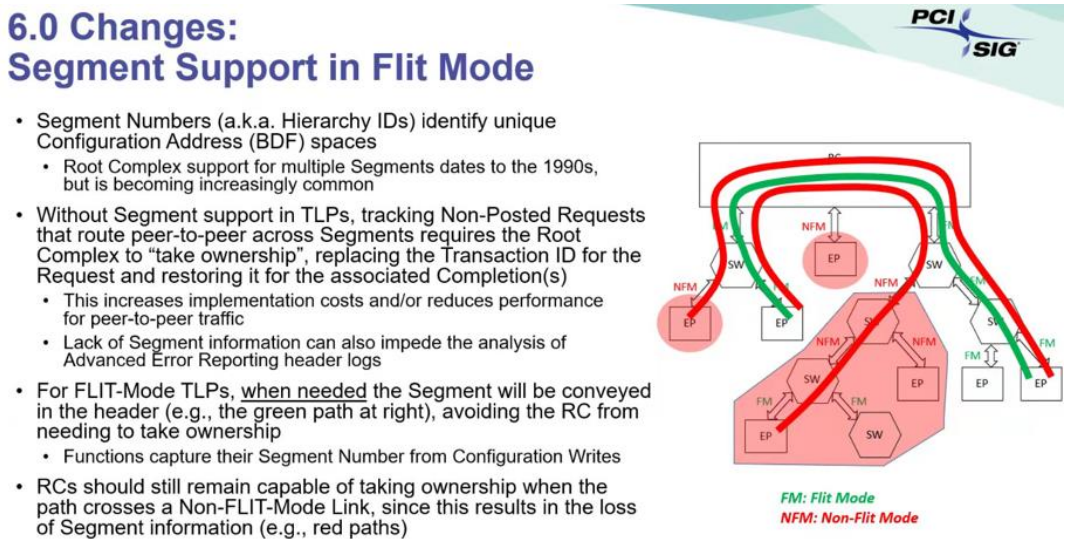
通过以下三者组合可以唯一标识一个 Function：

- ✓ 唯一的系统（RC）标识符；
- ✓ RC 内唯一的层级（Hierarchy）标识符；
- ✓ Hierarchy 内唯一的 Routing ID。

11. 在支持 HID 的系统中，所有 TLP 均会带有 HID 吗？

问题补充：未在 Spec 中找到 HID 被插入到了 TLP 的哪个字段了，难道 HID 只让 EP 和 Switch 知道但不会出现在 TLP 中？

不是的。PCIe 6.0 之前是有专门的 Hierarchy ID Message，PCIe 6.0 Flit Mode 中有 Segment 字段。根据下图，TLP 中携带 Hierarchy ID 会在 PCIe 6.0 中实现，目前 PCIe 5.0 尚未实现。



12. Hierarchy ID 的实际意义是什么？是否相当于做了 NTB Msg Register 的事情？

问题描述：在多 RC 环境下，一般用 Switch Partition 隔离，分区间的通信一般用 NTB 地址窗口映射实现，而 NTB 一般提供了 Message 传递寄存器用于互传 Msg（意见 A），那么 Hierarchy ID 的实际意义是什么？是否相当于做了 NTB Msg Register 的事情？

本问题没有具体解答，罗列一些现场发言：

- ✓ 论意见 A：多 RC 不只有 NTB，还有 Crosslink，需要改 OS。多年前华为就做了 Crosslink，现在多主机的有 PLX 的 97xx 和 mscc 的 pax 系列。

- ✓ 驳论意见 A：Crosslink 很早就有了，内部也是个隐含的 NTB，底层都需要预先配置好，目前 OS 应该都支持了。
- ✓ Hierarchy ID 看来只是给单 Host 多 RC 系统准备的，让 Switch 知道 Msg 应该被路由到哪个 RC。该设计的前提是允许一个 Tree 下存在多个 RC，并解决多 RC 重复枚举的问题。若能解决重复枚举问题，则 Msg 随便返回任意 RC 时 Host 都应能够处理，而非搞 HID，弄得好像 Host 端分别处理每个 RC 收到的 Msg。如此一来，Switch 分区意义何在？这些方案感觉在相互矛盾。
- ✓ 多 Host 路由，其实用 Partition ID 也可，没必要用 Hierarchy ID。对于多 Host，只要是 Switch 内部，都能区分。目前 MCHP 的 Switch 使用 NTB Msg Register 相互传递 Msg。
- ✓ NT 用的人很少，多是存储用户。当前有些边缘计算、自动驾驶、AI 企业逐渐开始用 NTB 来做 Host-to-Host 通信。
- ✓ 多 Host 路由用 Partition ID 亦可。只需 MCHP 的 Switch 使用 NTB Msg Register 相互传递 Msg，则 Switch 内部都能区分。

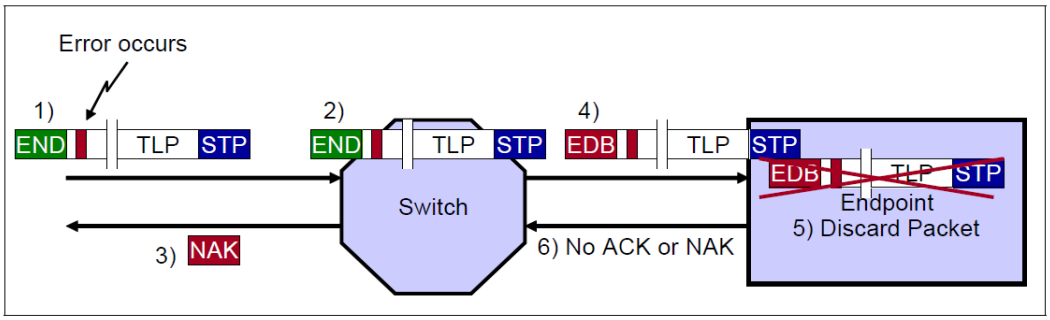
Switch

13. PCIe Switch 有哪几种传输模式？

PCIe Switch 传输 TLP 有两种模式：① Store & Forward；② Cut-through。

Store & Forward，接收报文并进行校验，如有错误走 NAK DLLP 请求重传，否则推送到 Egress Port。这种模式保证送到 Switch 出口的报文是正确的，但也因此增加了延时。对于 Data Payload 比较大的 TLP，采用该模式时延很大。

Cut-through，直截了当，接收报文直接推给 Egress Port，没收完也推，延时低但不保证报文正确。当接收完报文进行校验，发现报文错了，发送 NAK DLLP 给 Requester 请求重传，然后给 Completer 发送 EDB（End Bad）让 Completer 无视该报文（如下图）。当链路质量不好时，频繁出现校验失败，采用该模式很浪费带宽。



▲图： Switch Cut-Through Mode Showing Error Handling

14. 挂在同一 Switch 下的 EP 设备可以相互直接通信吗？

可以。同一个域不需要额外设置。

15. 现在国产 PCIe Switch 有哪几家，3.0 的成熟吗？

国产 PCIe Switch 厂家有国微（2.0）、Asmeida（3.0）。Asmeida 的用起来也有 Bug。

16. Switch 在什么场景下会主动给下面的 EP 发 MWr？

补充：博通 BCM Switch。之所以确认是 Switch 发起的请求，是因为 EP 收到 MWr 中的 Requester ID 为 Switch 的 ID。

- ✓ 可排查是否使用了 NTB。
- ✓ 可排查是否为主机软件给 Switch 发了 Doorbell，Switch Decode Doorbell 之后生成的 MWr 请求。
- ✓ 有些 Switch 是会主动发请求。在 Gen5 Switch 内带有 CPU，CPU 会劫持上游发来的 Cfg 包和各种 Msg 并更改其中的字段，可排查是否存在该情况。

17. Switch 上只有一个 Host 时为什么会出现 Cfg 拦截？

为便于 Switch 上多个 Host 之间共享资源， Switch 会开启 Cfg 拦截，只有一个 Host 不建议开启拦截。如果出现了拦截现象，可确认配置的模式是否正确，确认是否启用了 Synthetic Mode。

18. 博通 PCIe Gen5 Switch 的 Synthetic Mode 是什么？

这应该是 88、89 系列 Switch 的新功能，博通 PCIe Gen5 Switch 内嵌 CPU，根据是否启用 CPU 对应两种工作模式：Base Mode 和 Synthetic Mode。Base Mode 是不启用 Switch CPU，Switch 为标准 Switch；Synthetic Mode 时启用内部 CPU 作为 Host，此时 Switch 充当 Host 角色。可参考 Switch Spec 下述描述：

- ✓ Base mode: The switch functions without any firmware involvement. In this mode, the embedded CPU is disabled and the device will operate as a standard PCIe fan-out switch. The switch may be programmed to provide PCIe fanout using a set of chassis management capabilities with MPT endpoints embedded in the switch.
- ✓ Synthetic Mode: In this mode, the embedded CPU becomes the host and allocates I/O devices and internal resources to external host devices connected to the switch in single or multi-host environment. The embedded CPU synthesizes the hierarchy for each connected host based on firmware loaded in the embedded RAM.

19. 如果 Switch 误开了 Synthetic Mode，使用过程中 BAR 会被 Switch 改掉吗？配置空间是否会被 Switch 内的 CPU 动态调整？

正常应该不会去调整，如果随便调整的话就不能用了。

20. PCIe Switch 是如何分配每个 DSP 占总 Lane 数中多少 Lane 的？

    问题描述：比如一个 PCIe Switch，USP 为 1x16，DSP 有 12 个 Port 共 48 Lane，这 48 条 Lane 可以随意分 12 个 Port 吗？比如 1x16+11x2。

    一般不能随便分，但可以按照固定模式分，这取决于 Switch 支持怎么分。Microchip 的 Switch 分配比较灵活，x2 以上几乎可以随意分；博通的没有那么灵活。

21. 一个 8 Lane Switch 下可以挂单个 x8 设备，也可以挂两个 x4 设备。挂单个 x8 时，两个 x4 的 LTSSM 都会参与链路训练吗？

    不是，只有一个 LTSSM 参与训练。每条 Link 对应一个 Controller 下的一个 LTSSM，当 8 条 Lane 作为 x8 使用时，8 条 Lane 均参与训练，但只使用了一个支持 x8 的 Controller，另一个 x4 Controller 是没有使用到的。

22. 除了 BDF 编号的限制外，Switch 扩展设备的数量还有什么其他的限制吗？

    数量上没有限制，性能上有所限制。

Bridge

23. lspci -tv 扫描出一些 AMD 的 Dummy Host Bridge，这些 Dummy Host Bridge 是什么？

    顾名思义，Dummy Host Bridge 就是假的 Host Bridge，其主要目的是先占据 Slot 或者 Bus，防止后续 OS Rescan 时候没有设备。主要用于占位，没什么实际作用。

24. 在集成了 PCIe 转 PCI 桥的卡中，桥下接了 Device Function ID 不同的多个设备，但桥的上游不支持 ARI 功能，此时 BDF 是怎么转换的？

    问题补充：在内核 OS 中，针对于该 PCIe 转 PCI 桥后做 SMMU 页表时，会有两个 SID：其一为自己本身的 SID，另一为 PCIe 转 PCI 桥的 SID。但是，现在每个设备都使用到桥的 SID 来建立 SMMU 页表，导致只能第一个设备没有问题。捉波形看，下面发上来的也是桥 SID 的 MEM 报文。

    这里跟 ARI 没关系，因为 PCI 是标准的东西，BDF 还在正常的范围内。在 Windows 下试过 PCIe to PCI 桥，可以正常识别桥下的多个设备，未在 Linux 下试过。检查排查内核选项，尝试在 2.6x 等老版本内核。

    反馈：不使用 SMMU 时可以正常识别，使用 SMMU 时只有第一个设备可以正常使用，后面几个设备存在 SID 冲突无法正常使用。

结论：现在 OS 不支持 SMMUv3 的 Sharing SID 的功能导致。

25. NTB 是什么？

NTB 是指非透明桥，RC 不知道 EP 是什么东西。一般的 PCIe Switch 或者 Bridge 都是透明的；Intel CPU 上带 NTB。

26. Ice Lake CPU 同一 RC 发出请求中的 RID 随机变化或导致 NTB 失效，对于该问题 BRCM 有解决方案吗？

问题详述：对于 Ice Lake 及后续 CPU，同一个 RC 发出的请求中 RID 随机变化且毫无规律，这对 NTB 是致命打击，直接导致 NTB 失效。这种情况 BRCM 有解决方案吗？

可考虑用 CPU 自带的 NTB，CPU 自带的 NTB 有 ID 翻译，RC 的 ID 对 NTB 无甚影响。——CPU 自带的 NTB 每一笔 TLP 都需要上 Root 然后下去，速度较慢。

结论：NTB 可以记下 Request 里的 RID。虽然不确定 RC 有多少 RID，但最多 256 个，不担心记满。

27. 上述设计导致向外部暴露多个 ID，为何 Ice Lake 会有上述设计？

因为里面是多个设备，Bus0 上的设备都是 RC 内部的。

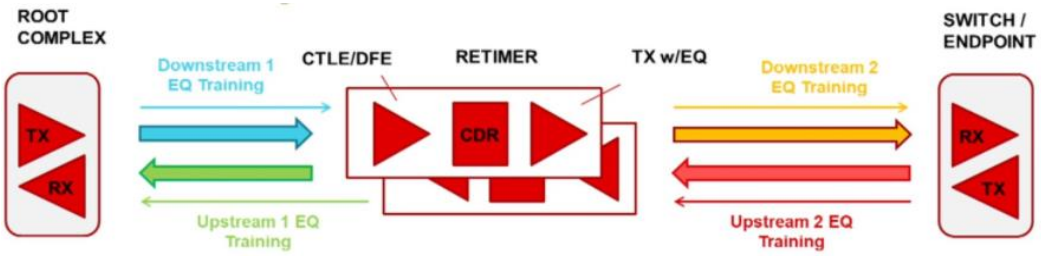
Retimer

28. PCIe Retimer 是什么？

随着 PCIe 的迭代，传输速率越来越高，高速信号传输中的信号衰减问题越来越大。目前解决信号衰减的三大方案：① 高速 PCB 板材；② Retimer；③ Redriver。Retimer 是三者中性价比最高的一种方案，也更为主流。2021 年是 Retimer 发展元年。

Retimer 通过其 Rx 端 CTLE/DFE (连续时间线性均衡/判断反馈均衡) 、CDR (时钟数据恢复) 及 Tx 端 EQ (均衡)，来够补偿信道损耗，消除信号抖动，提升信号完整性，从而增加传输距离。

Redriver 是放大信号，Retimer 是重新生成信号。Retimer 比 Redriver 性能更高，但时延也更大。



29. 如何判断 PCIe 系统是否要用 Retimer?

有两种常用的方法来判断 PCIe 系统是否需要 Retimer：

根据信道衰减：检测端到端的 PCIe 信道插入衰减，若信号衰减大于 PCIe 信道预算，则需要采用 Retimer。信道插入衰减包括 RC 和 EP 的封装损耗。

根据眼图：根据均衡后的信号眼图及误码率来判断是否要用 Retimer。若眼高 EH<15mV，眼宽 EW<0.3UI，或者误码率 BER>1e-12，则建议采用 Retimer。此方法考虑了信号反射及码间干扰，比采用信道衰减的判别法更为准确。

▼ 表 PCIe 系统各部分的信号衰减预算

Root package	Non-root package	CEM connector	Add-in Card(AIC)	System Budget	Total channel budget
9.0 dB	4.0 dB	1.5 dB	9.5 dB	17.5 dB	36 dB

表注：System budget 包括基板、转接卡、基板-转接卡，PCIe CEM 连接器。

30. RC 和 EP 如何知道它们中间有 Retimer ?

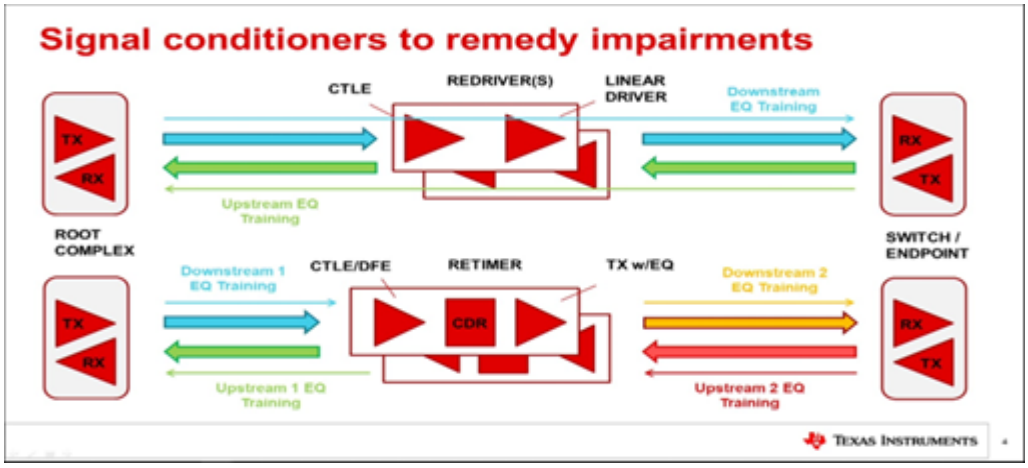
TS2 Symbol 5 有两个 field 来指示是否有 Retimer，TS2 经过 Retimer 时，Retimer 会更改这两个字段。

Symbol 5 bit4 为 Retimer Present Bit，仅在 2.5 GT/s 速率时有效，该位为 1 时表示 RC-EP 间至少有一个 Retimer。

Symbol 5 bit5 为 Two Retimer Preset Bit，仅在 2.5 GT/s 速率时有效，该位为 1 时表示 RC-EP 间至少有两个 Retimer。对于 16 GT/s 及以上速率，必须实现该 field，对 8 GT/s 可选。

31. 对于 CPU-Device 之间存在 Retimer 的连接场景，建链过程的 TS 码流在 Retimer 中直接转发的还是 Retimer 重新生成的？EIEOS 呢？

注：上述场景示意图如下。



TS 和 EIEOS 是在 Retimer 中重新生成的，CPU-Retimer 和 Retimer-Device 作为独立的链路进行建链；TLP 是转发。

相比之下，Redriver 纯粹在电路层的模拟信号层发挥作用。对信号进行放大和增强，比如利用多阶均衡器，对链路信号进行学习然后做预加重或者去加重。

32. 如果链路中间有 Retimer 做了 Skip Remove，Rx PIPE 做 128/130 解码时如何保证一个 Block 内不会把非 Skip 的数据包进来？

    这不是 PIPE 去处理的，而是在 Descrambler 那边处理的。到了 Descrambler 那边，会有一根线指示是否为 Skip Remove，然后再结合 Descramble 处理。

P2P

33. 什么场景下会用到 P2P 功能？

    很多情况，比如 GPU 直接，存储之间。

34. PCIe 的 P2P 是指是同一个 Switch 下的两个设备，还是不同 Switch 下的两个设备通信？

    这两种都应该算是 P2P 通信。

    一般来讲不跨 RC 的 PCIe Device 之间通信都应该算 P2P，当然也有跨 RC 的 P2P。同一 Hierarchy 下不同 EP 之间的访问算是 P2P，不同 Hierarchy 之间 Switch 之间有 NTB 也能实现 P2P 访问。

35. 请问飞腾平台能在两个 RP 下的 EP 之间进行 P2P 通信吗？

不确定。飞腾平台下 P2P 通信，实测写没问题，读回来全是 F。在 x86 平台验证读写都没问题

36. P2P 一般支持 Snoop-able 的访问吗？

不支持吧。EP 不支持 Snoop-able。

37. P2P 可以发 RO=1 和 IDO=1 吗？

注：RO-Relaxed Ordering；IDO-ID Based Ordering。

按理说这些都是可配的。发都能发，主要看路径及双方是否支持。

在 RP 上层见过 IDO 可配，在设备 BIOS 上也看到过可配。这些不暴露给用户，写到内部寄存器了。

38. 请问一个 EP 想与另一个 EP 通信，其如何获知对方的 BDF？

P2P 通信一般走地址路由，不需要 BDF。

39. 已知服务器支持 PCIe Switch 下的 P2P 通信，如何在工程实践上去验证 P2P 功能呢？

可以找个 NVMe 硬盘用 GPU 写一下。GPU 通过 P2P 读写 NVMe 硬盘的驱动程序没有现成的，需要自己开发。

40. P2P 对 EP 设备有什么特殊要求吗？

问题补充：我理解 P2P 主要靠桥来路由，对 EP 设备是不是没什么特殊要求？

如果不考虑 IOV、不开 IOMMU 的话，P2P 对 EP 没什么要求，PCIe 都是地址路由的，EP 对是否 P2P 不敏感，EP 只管处理事务，至于转发给谁或者谁下发的，由桥来做。如果开了 IOV，使用 P2P 需要 ATS、ACS 相关的支持。

41. 在 Switch 未开启 P2P 功能的前提下，Switch 收到 EP 发来地址落入 Mem Base/Limit 范围内的 Mem 请求，是不是不应该转发而是直接反馈 Cpl.UR？

应该不是回 Cpl.UR。在 Type1 Config Header Mem Base/Limit 寄存器介绍中未提及 P2P 开关问题，只说落在 Mem Base/Limit 之内的会被转发到该 DSP 的另一 Secondary Bus。问题所属未开启 P2P 功能，应该是通过 ACS P2P 请求重定向来实现的，当 Switch DSP 收到落入 Mem Base/Limit 范围内的 P2P 请求后不转发到另一 DSP，而是向上转发到 RC。

42. Switch 开了 ACS 重定向之后 RC 端还可以支持 P2P 吗？

    问题详述：两个 GPU 通过 Switch 连接到 CPU，Switch 上打开了 ACS 重定向，此时 RC 端还可以支持 P2P 吗？NV 上层库是怎么判断底层是否支持 P2P 的？

    是否支持 P2P 访问是 Device Driver 自己控制的。NV 的 GPU 印象中只能位于同一 RC 下，即 RC 端不能做 P2P 访问。

Bifurcation

43. PCIe Bifurcation 有什么优势吗？

    相比于传统一个 Controller 配一个 PHY 的设置，Bifurcation 相当于多个 Controller 共用 PHY，可以省一点芯片面积。

44. 一个 PCIe Controller 可以连接多个 PHY 吗？

    如果是同一条 Link，在 PCIe Controller 允许的最大 Link Width 范围内，在 PCS 支持的前提下可以连多个 PHY，多个 PHY 跟 Controller 之间通过 PIPE Mux 连接。比如 Controller 最大支持 x8，那么可以连两个 x4。

    如果说的是多条 Link，即 Bifurcation 的情况，则每条 Link 都需要一个单独的 Controller。

45. 同一个 PCIe Controller 可以支持多个 Link 吗？有没有存在 2 个 x4 的 Link 公用一个控制器来替代 1 个 x8 的 Link？

    不能，一个 Controller 对应一条 Link。

46. 同一插槽下支持多种配置，多种配置是对应多个 EP 设备吗？

    问题详述：在 Mindshare PCI Express Technology 14 章 Link Initialization and Training 中，介绍了一种 x4 拆分为 1x4、2x2、4x1、1x2+2x1 的链路配置方式（Link Configuration Example 2），多种配置是对应多个 EP 设备？还是同一设备的不同功能？该情况只存在于 Switch 中吗？

    不同链路对应不同的 EP 设备。如果是同一个主机接了多个设备，称为 Bifurcation 分叉，典型应用就是阵列卡，1x16 拆分为 4x4 的四个 SSD。如果是同一个设备接了多个主机，称为 Multi-Host 多主机。该情况并非只存在于 Switch。

47. Host 端 BIOS 把一个 PCIe x16 槽位改成了两个 x8，EP 端是 x16 的，能降 x 识别成 x8 吗？EP 需要支持分叉配置吗？

EP 自动识别，全自动协商。可以识别成 x8，不需要支持分叉配置。EP 只能识别为 1 个 x8，而不是两个 x8。

48. EP 支持 Bifurcation 吗？如果 EP 支持 Bifurcation 是不是也需要多个 EP Controller？

日常还是 RP Bifurcation 用得多，比如在电脑主板上，x8 可以分成 2 个 x4 或四个 x2 使用。但也存在 EP Bifurcation 的情况，比如某些 SSD 设备、DPU。

49. Synopsys 公司的 2 Lane 的 Controller+2 Lane 的 PHY 可以接 2 个 x1 的 Device 吗？

不可以。只有支持 Bifurcation 才可以，这种一般要有 2 个 Controller。

50. Cadence 公司的 2 Lane 的 Controller+2 Lane 的 PHY 可以接 2 个 x1 的 Device 吗？

不可以，谁家的都不行。做 Bifurcation 都是要额外设计的，需要两个 MAC。

51. 比如一个 x16 和一个 x8 可以工作在单 x16 或 2x8 模式，怎么通过 Bifurcation Mux 跟 PHY 的 PIPE 接起来？

MUX 左边接 16+8 Lane 的 Controller PIPE，右边接 16 Lane 的 PHY PIPE，里面就是一个简单的逻辑 MUX。这里要求 PHY 也支持 Bifurcation。

Crosslink

52. PCIe Crosslink 是什么？

为了解决不同处理器间的互联问题，PCIe 提供一种 Crosslink 的链接方式。两个同方向的 Port 相接称为 PCIe Crosslink，比如 Downstream Port 接 Downstream Port 或 Upstream Port 接 Upstream Port PCIe。

Crosslink 是一种可选的能力。支持 Crosslink 的两个 PCIe 设备在 Link Training 的时候沟通谁是 DSP 谁是 USP（具体可搜索 Crosslink random time）。

53. 现在会用 Crosslink 这个功能吗？

会用，在类 NVLink 互连场景中可以用 PCIe Crosslink 方案，两个 Die 的 Device Type 不区分 RC 还是 EP，开启 Crosslink 后双 Die 直连自动协商谁是 RC 谁是 EP，本质上还是 RC-EP 的链路。

这个 Feature 比较小众，似乎只有 Synopsys 公司支持，且需要 IP 为 DM 模式。2021 年左右使用 Synopsys 公司 PCIe IP 的 Crosslink 功能，出现 TL Link Up 后双方均为 RC 或

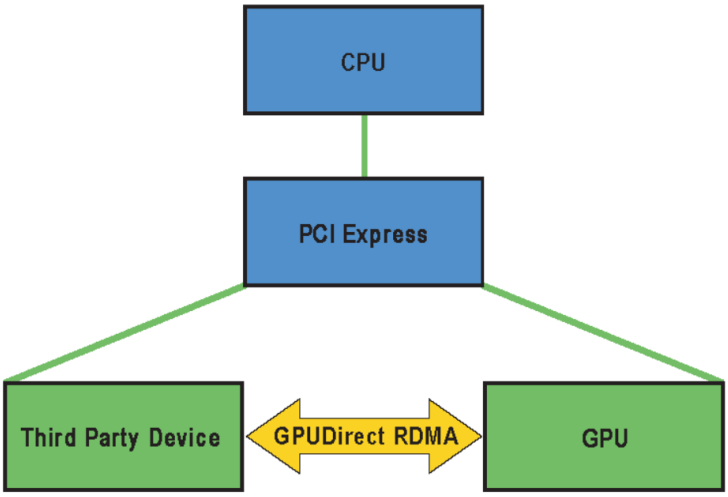
均为 EP 的情况。虽然 Link Up，但 DL 未 Active，无法正常发包。可能为其 IP Bug，也可能为使用配置问题。

当然，如果为 DM 模式，硬件也可以不支持 Crosslink 通过软件实现类似的机制。当两端 Device Type 相同时会 Link Fail，届时再基于相关机制通过软件随机去改 Device Type。相较于开启 Crosslink 硬件自协商的方式，这种软件随机改 Device Type 的方式较慢，可能需要多次 Link Training。

GPU Direct

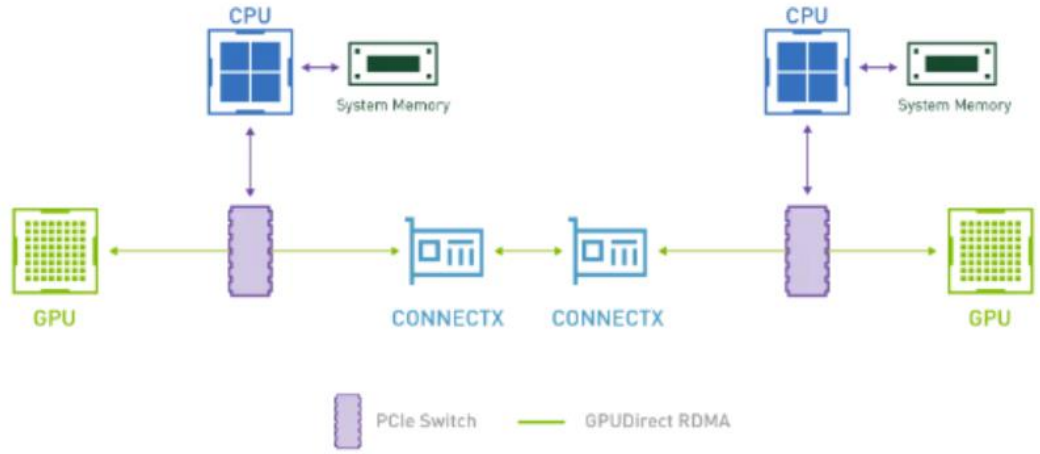
54. GPU Direct 是什么？

GPU Direct 是一种 GPU 绕过 RC 直接与其他设备通信的技术，可以说是一种 P2P 通信方式。下图就是一种 GPU Direct。



GPUDirect RDMA within the Linux Device Driver Model

55. GPU Direct RDMA 中是如何使用 PCIe 的？



网卡驱动直接控制网卡数据写 GPU 显存，相当于绕过 System Memory，GPU 直接调用 RDMA 走 PCIe P2P 进行数据传输，以提高效率。

个人理解，对于自研 GPU，GPU 里面放个小 CPU、PCIe Switch 下挂一个 RDMA 网卡应该也能用起来。GPU Direct RDMA 的方案跟 EP 关系不大，显卡不需要特别支持什么，但显卡驱动需要支持 GPU Direct，Linux Inbound Driver 中有一部分，GPU 里可能也需要一部分 RDMA 网卡驱动，GPU 的 Driver 调用网卡 RDMA 来抓自己 Local Memory 里面的内容。对 RC 而言，其只负责通过 ACS 控制 P2P。在 NVLink 中，这部分驱动未开源，由用户分配 GPU 显存，RDMA 接收驱动要放在 GPU 里面跑。

（参考：https://blog.51cto.com/marsggbo/5821309 ）

二、传输层

TLP 分类

56. PCIe TLP Type 有哪些？

根据 PCIe TLP Header 中的相关字段可以区分不同类型的 TLP，Non-Flit Mode 时看 Fmt[2:0] 及 Type[4:0]，Flit Mode 时看 Type[7:0]。无论 NFM 还是 FM，PCIe 支持的 TLP Type 都是相同的，包括：

- ✓ Memory 相关的 MRd，MWr，MRdLk，DMWr。
- ✓ IO 相关的 IORd，IOWr。
- ✓ Configuration 相关的 CfgRd0，CfgWr0，CfgRd1，CfgWr1，TCfgRd。
- ✓ Message 相关的 Msg 及 MsgD。
- ✓ Completion 相关的 Cpl，CplD，CplLk 及 CplDLk。
- ✓ Atomic 相关的 FetchAdd，Swap，CAS。
- ✓ PASID Prefix 相关的 LPrfx 及 EPrfx。

57. PCIe Memory 访问事务包分为哪几种？

PCIe Memory 访问 TLP 分为：

- ✓ Memory Write Request。该事务为 Posted，无需 Completion。
- ✓ Memory Read Request 及其 Completion。
- ✓ Deferrable Memory Write Request 及其 Completion。该事务为 Non-Posted，需要 Completion。
- ✓ AtomicOp Request 及其 Completion

以上几种访问请求均为地址路由，根据地址格式的不同分为 32 bit 地址请求及 64 bit 地址请求。Memory 访问请求 TLP 中，可根据实际需求选择添加 PASID (Process Address Space ID)。

58. PCIe IO 访问事务包分为哪几种？

PCIe IO 访问 TLP 分为：

- ✓ IO Write Request 及其 Completion
- ✓ IO Read Request 及其 Completion
- ✓ PCIe IO 访问请求只能选用 32 bit 地址格式，不支持 64 bit 地址。

59. PCIe Configuration 访问事务包分为哪几种？

PCIe Configuration 访问 TLP 分为：

- ✓ Configuration Write 及其 Completion
- ✓ Configuration Read 及其 Completion

60. PCIe 转发、非转发事务是什么？

转发事务，请求者发送 TLP 到完成者，无需完成者进行反馈。转发事务有 MWr，Message。

非转发事务，请求者发送 TLP 到完成者，按照分割事务进行处理，需要完成者进行反馈。非转发事务有 MRd，DMWr，IORd，IOWr，CfgRd，CfgWr。

由于转发事务无需反馈，其比非转发事务性能更高。非转发事务出错时，可以得到硬件通知；转发事务出错时，只能通过完成者记录错误并上报 RC，由软件进行处理。

参考：https://blog.csdn.net/icxiaoge/article/details/80456837

61. CPL 状态有哪几种？

PCIe Requester 发出的所有 PCIe Rd、NPWR、Atomic 请求及 PCIE 6.0 的 DWMR 请求均需要 Completer 返回 CPL/CplD，CPL/CplD 中含有 Cpl.Status 字段，指示完成情况。那么 CPL 状态有哪几种呢？如下：

- ✓ SC，Successful Completion，请求成功；
- ✓ UR，Unsupported Request，不支持的请求；
- ✓ CA，Completer Abort，支持但由于违例等原因被终止了；
- ✓ RRS，Request Retry Status，需要重新发生请求。在 Gen6 之前叫 CRS，即 Configuration Request Retry Status，PCIe 6.0 的 DMWR 也支持请求重传，所以把 configuration 去掉了；
- ✓ 其他预留。

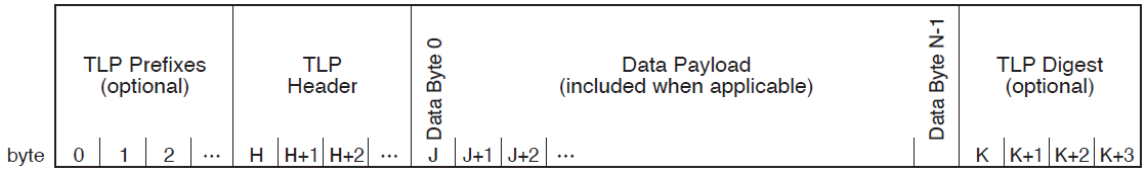
TLP 组包

TLP 字段

62. PCIe TLP 包格式是怎样的？

简单来讲，PCIe TLP 包由四部分组成：TLP Prefix、TLP Header、Data Payload 及 TLP Digest（如下图所示）。其中，TLP Header 是必须有的，不同 TLP 类型的 TLP Header 有

所不同；根据 TLP 类型的不同 Data Payload 可以有也可以没有；TLP Prefix 及 TLP Digest 是可选的。

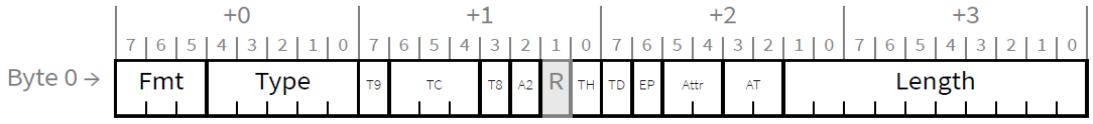


▲图： Serial View of a TLP

63. PCIe TLP Header 格式是怎样的？

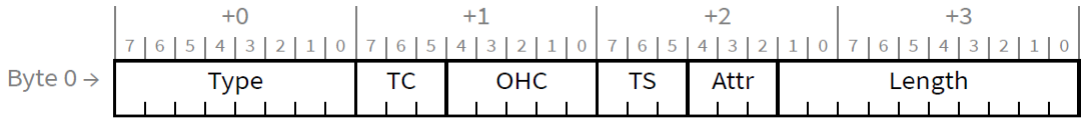
Non-Flit Mode 与 Flit Mode 的 TLP Header 有所不同，同一 Mode 不同 Format/Type 的 TLP Header 也存在差异。但无论如何，同一 Mode 下 TLP Header 的第 1 DW （TLP Header Base）是相同的。

Non-Flit Mode 时的 TLP Header Base 格式如下：



▲图： PCIe Non-Flit Mode TLP Header Base

Flit Mode 时的 TLP Header Base 格式如下：



▲图： PCIe Flit Mode TLP Header Base

64. PCIe TLP Header 支持哪些 Format?

PCIe Non-Flit Mode 时，TLP Header 中的 Fmt 位域指示当前 TLP Header 的 Format，支持的 Format 有以下几种：

▼表：PCIe TLP Header Formats

Fmt[2:0]	TLP Format Description
000b	3 DW Header，No Data
001b	4 DW Header，No Data

010b	3 DW Header，With Data
011b	4 DW Header，With Data
100b	TLP Prefix
Others	Reserved

65. PCIe TLP Header 中有哪些 Fields？

PCIe TLP Header 中常见的 Fields 有：

▼表：PCIe TLP Header 中的常见 Fields

Fields	Description
Fmt[2:0]	Format，指示 Header 的格式，是否为 TLP Prefix，Header 的长度 3 DW/4 DW，是否携带 Data Payload 等，仅在 Non-Flit Mode 时有效。
Type[4:0]@NFM Type[7:0]@FM	用以指示 TLP 的类型。
Len[9:0]	Length，用以指示当前 TLP 的 Data Payload 长度，仅对带有 Data Payload 的 TLP 有效，单位为 DW。
Attr[5:0]	Attributes，通过该字段提供附加信息给接收端，告知接收端采用非默认方式来处理当前 TLP，常用于 TLP 排序、Snoop 硬件一致性管理等方面。
Requester ID[15:0]	Request TLP 中 Transaction ID 的组成部分，请求者的 ID，包括 Bus Number、Device Number 及 Function Number。
Completer ID[15:0]	Completion TLP 中 Transaction ID 的组成部分，完成者的 ID，包括 Bus Number、Device Number 及 Function Number。
Tag[9:0]@NFM Tag[13:0]@FM	Transaction ID 的组成部分，用以 Request Outstanding。

TC[2:0]	Traffic Class，PCIe QoS 相关，用以指示 Transaction 的服务优先级。
TD[0:0]	TLP Digest，用以指示 TLP 末尾的 Digest 字段是否含有 ECRC。
First DW BE[3:0]	First DW Byte Enables，用以指示当前 TLP Data Payload 中第一 DW 中的有效 Byte。
Last DW BE[3:0]	Last DW Byte Enables，用以指示当前 TLP Data Payload 中末尾 DW 中的有效 Byte。
BC[11:0]	Byte Count，用以指示为完成当前 Request 所剩余的 Data Payload Byte 数目。
BCM[0:0]	Byte Count Modified，用以指示 Byte Count 字段是否被修改，仅用于 PCI-X Completer/Bridge，PCIe Completer/Bridge 不能设置该位。BCM 仅在 Non-Flit Mode 时有效，Flit Mode 没有 BCM。
EP[0:0]	Error Poisoned，PCIe E2E 数据完整性相关，用以指示当前 TLP 的 Data Payload 被污染。
AT[1:0]	Address Type，一般用于 ATS，仅对 Memory Read TLP 有效，用以指示当前地址的类型，00b Untranslated，01b Translation Request，10b Translated，11b Reserved。
Message Code[7:0]	Message 编码，用以指示当前 Message 为哪种 Message。
Message Routing[2:0]	该 Field 位域 Type 中，用以指示当前 Message 的路由信息。对于 Message TLP，其 Type[2:0] 用以指示 Message Routing 信息，000b 路由到 RC，001b 根据地址路由，010b 根据 ID 路由，011b 来自 RC 的广播消息，100b 至于本端 Rx 的消息，101b 仅用于 PME_TO_Ack，收集并路由到 RC，110~111b Reserved。

CS[2:0]	Completion Status，在 Completion TLP 中指示 Completion 完成状态
TH[0:0]	TLP Hints，用以指示 TLP Header 中是否含有 TPH 及 TPH TLP Prefix。 参考《PCIe TPH (TLP Processing Hints) 介绍》
PH[1:0]	Processing Hints，位于 Address[1:0] 所在位置，TH=1 时 PH 有效，用以提供较粗粒度的 Processing Hints 控制能力。
ST[15:0]	Steering Tag，对 TPH Hint 作更详细的分类，用以提供较细粒度的 Processing Hints 控制能力。ST[7:0] 位于 TLP Header， ST[15:8] 位于 TPH TLP Prefix。
TS[2:0]	Trailer Size，用以指示该 TLP 是否存在 Trailer 并指示 Trailer Size，仅用于 Flit Mode。
Address[31/63:2]	地址
LA[7:0]	Lower Address，一般用于 Memory Read 的 Completion TLP 中，用以指明当前 Completion 所携带 Data Payload 的起始 Byte 地址。
OHC[4:0]	Orthogonal Header Content，正交头内容，用以来指示其后是否存在 OHC、是何种 OHC，是 PCIe Flit Mode 下新加的一个 TLP 字段。
PV[0:0]	PASID Valid，用于 Flit Mode。
PASID[19:0]	Process Address Space ID，跟 Requester ID 一起作为 Request TLP 读职空间的唯一标识。Non-Flit Mode 时位于 TLP Prefix 中，Flit Mode 时位于 OHC 文本中。
ER[0:0]	Execute Request，仅在 PV=1 时有效。
PMR[0:0]	Privileged Mode Request，仅在 PV=1 时有效。

NW[0:0]	No Write，仅用于 Translation Request，用以指示当前地址不可写。
HV[1:0]	Hints Valid，出现于 Flit Mode，用以指示 PH 及 ST 相关字段是否有效。
AV[0:0]	AMA Valid，出现于 Flit Mode，用以指示 AMA 字段是否有效。
AMA[2:0]	ATS Memory Attributes，出现于 Flit Mode，为优化 Memory 或 Cache 性能提供 Hints。
RSV[0:0]	Requester Segment Valid，出现于 Flit Mode。
Requester_Segment[7:0]	Flit Mode 时，用以指示 Requester 所在的 Hierarchy，位于 OHC-C 中。
DSV[0:0]	Destination Segment Valid，出现于 Flit Mode。
Destination_Segment[7:0]	Flit Mode 时，用以指示 ID 路由的 TLP 的 Detination 所在的 Hierarchy，位于 OHC-A3/A4/A5 中。
Completer_Segment[7:0]	Flit Mode 时，用以指示 Conpleter 所在的 Hierarchy，位于 OHC-A5 中。
PR_Sent_Counter[7:0]	应用于 IDE TLP。
Stream_ID[7:0]	用以指示 IDE Stream 的 Stream ID。
Sub-Stream[3:0]	用以指示 Sub-stream。
P[0:0]	IDE TLP 中用以指示当前 TLP 包含 PCRC。
M[0:0]	IDE TLP 中用以指示当前 TLP 包含 MAC。
K[0:0]	IDE TLP 中用以指示当前 TLP 包含 Key 设置。
T[0:0]	IDE TLP 中用以指示当前 TLP 源于可信的运行环境。

66. PCIe TLP Header Base Size 取决于什么？

取决于 TLP Format 及 TLP Type。NOP TLP 及 TLP Prefix 的长度为 1 DW；常规 Memory 读写访问请求（包含 Atomic）的 Header Base Size 根据 Address 位宽可以为 3 DW 或 4 DW；IO 及 Cfg 访问请求、DMWr 及 Cpl/CplD/CplLk/CplDLk 的 Header Base Size 均为 3 DW；Msg/MsgD 的 Header Base Size 为 4 DW。

67. PCIe TLP Header 中的 Length 有何意义？

PCIe TLP Header 中的 Length 字段共 10 bit，用以指示当前 TLP 的 Data Payload 长度，仅对带有 Data Payload 的 TLP 有效。Length 单位为 DW，非 0 值为几就代表有多少 DW 的 Data Payload，全 0 代表 024 DW。

68. PCIe TLP Header 中的 Transaction ID 是什么？

PCIe TLP 中的 Transaction ID 是给 Transaction 的 Outstanding 用的，用以唯一标记一笔处于 Outstanding 中的 TLP。Transaction ID 由 Requester/Completer ID 及 Tag 组成。Flit Mode 时，还可能包含 Requester Segment、Destination ID 或 Completer ID。

69. PCIe TLP Header 中的 Lower Address 有何意义？

Lower Address (LA) 一般用于 Memory Read 的 Completion TLP 中，用以指明当前 Completion 所携带 Data Payload 的起始 Byte 地址。Completion 中不含 First DW BE 字段，接收端通过 LA 来确定起始地址。

从名称也能看出来，这里的地址并非完整的地址，而是完整地址截取后的低位地址(位宽 7 bit)。一笔 Memory Read 可能会返回多笔 Completion，对于这多笔 Completion，第一笔 Completion 的 LA 字段即 First DW BE 所对应的有效地址，后续 Completion 的 LA 跟 RCB 有关。对于 RC 返回的 Completion，其 RCB 为 64 B，相邻 Completion 的 Data Payload 真实地址间隔 64*N，此时 LA[6:0] 为全 0 或 100_0000b；对于其他 RCB=128 B 的情况，相邻 Completion 的 Data Payload 真实地址间隔 128*N，LA[6:0] 必须为全 0。

70. PCIe TLP Header 中的 Byte Count 有何意义？

Byte Count[11:0]一般用于 Completion TLP，用以指示剩余 Byte 数目。Byte Count 非零时其值即为完成当前请求前未返回的 Data Payload 的 Byte 数目（含当前 Completion 中的 Data Payload Byte 数目），Byte Count=0 表示剩余 4096 B（跟 Memory 请求 Length 最大值 1024 DW 相呼应）。

71. PCIe TLP Header 中的 Byte Count Modified 有何意义？

Byte Count Modified (BCM) 位宽只有 1 bit，用以指示 Byte Count 字段是否被修改，仅用于 PCI-X Completer/Bridge，PCIe Completer/Bridge 不能设置该位。BCM 仅在 Non-Flit Mode 时有效，Flit Mode 没有 BCM。

对于 Multi-Completion 的情况，第一笔 Completion 中的 Byte Count 值跟读请求要求的 Byte 数目相同。在 PCI-X 的一些应用中，其需要修改第一笔 Completion 的 Byte Count 字段为第一笔 Completion 所携带的 Data Payload Byte 数目，并将 BCM 置一，告知请求方 Byte Count 被修改，其后仍然有该笔读请求所对应的 Completion（不然会认为 Completion 已经发送完毕）。BCM 只能在同一读请求的多笔 Completion 中的第一笔 Completion 中置一，后续 Completion 不能置一，也不能修改 Byte Count 字段的值。

72. PCIe TLP Length 只有 10 bit 位宽，该如何表示 1024 DW 长度的 Data Payload？

Length=0 表示 1024 DW。

73. PCIe MWr/MRd TLP 中的地址必须 4 B 对齐吗？如何发出非 4 B 对齐的请求？

MWr、MRd TLP 中地址必须是 4 B 对齐的，因为 TLP 地址中的 addr[1:0]不存在，这 2b 用作了其他用途。虽说地址必须 4 B 对齐，但仍然可以通过 TLP Header 中的 First DW Byte Enable（FBE）、Last DW Byte Enable（LBE）字段来实现非对齐访问。

74. CplD 的 TC 字段需要跟 Req 中的 TC 保持一致吗？

是的，Completion 中的 TC 必须跟其请求中的 TC 保持一致。Spec 中解释如下：

A Completer must accept requests with TC label other than TC0, and must preserve the TC label. That is, any completion that it generates must have the same TC label as the label of the request.

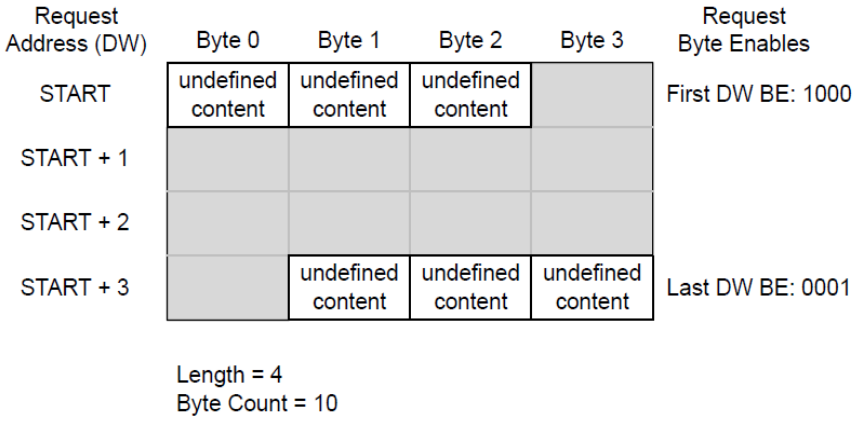
75. PCIe Completion 中的 EP 和 Cpl.Status 之间有什么联系吗？EP=1 时一般后者会是什么值？

EP=1 指示 CplD Data poisoned，都返回 CplD 了而非 Cpl，所以 CplD.Status 应该是 SC。

76. First DW BE、Last DW BE 是什么？

First DW BE 和 Last DW BE 是 Memory 读写请求 TLP Header 中的一个字段，指示读写请求数据载荷中的第一个及最后一个 DW 中的数据 Byte 是否有效。First DW BE[3:0] 对应第一个 DW 的 Byte[3:0]，Last DW BE[3:0] 对应最后一个 DW 的 Byte[3:0]，如图所示。

若 BE 某 bit 为 0，表示该 byte 数据不能写入 Memory 或不可预取、不可读。若请求长度大于 1 DW，First DW BE 不能为 0；若请求长度等于 1 DW，Last DW BE 只能为 0；First/Last DW BE 的 4 bit 可以不连续。对于 Zero Read/Write，First/Last DW BE 均为 0。



▲图： First/Last DW Byte Enable 位置对应关系

77. PCIe Zero Write、Zero Read 是什么？

Zero Write 和 Zero Read 是指 PCIe 发送的数据载荷长度为 0 的内存读写请求，常用于 Flush 操作，以确保早前 Posted 发送的数据传输完成。

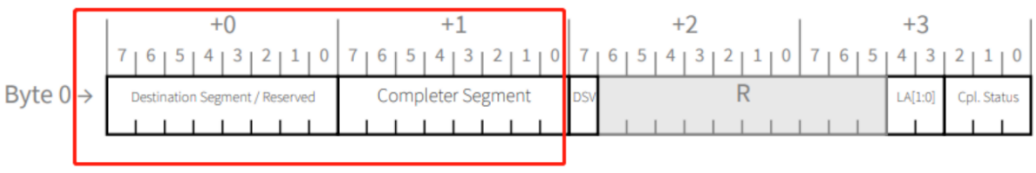
Zero Write、Zero Read 请求 TLP 中，Length 设置为 1 而非 0，0 表示 1024 DW，但 Byte Enable 均设置为 0。Zero Write TLP 中不携带数据载荷，Zero Read 的 CplD 携带有 1 DW 数据载荷（协议未指定，可为任意值）。

78. 如何知道 TLP Prefix 的长度？

每个 TLP Prefix 固定长度 1 DW，其中第 1 Byte 是 Fmt 及 Type，用以指明当前为 TLP Prefix。一个 TLP Prefix 不够可以继续追加，第二个 TLP Prefix 的第 1 Byte 再来一次 Fmt 及 Type。

虽然 TLP Prefix 中有 Fmt 及 Type，但那是 TLP Prefix 的，正式 TLP Header 中仍然需要 Fmt 及 Type。

79. PCIe 6.0 OHC-A5 中的 Segment 是什么？



§

Figure 2-11 ***OHC-A5***

这里的 Segment 是指 Segment ID，是给多 RC 域用的，一般只有大型项目才会实际用到 Segment ID（比如 Server）。

此前只有 BDF，现在需要 Segment ID + BDF，在 Bus Capture 之前需要先匹配 Segment ID，Completer EP 回 Response 时也需带上 Segment ID。

80. PCIe 6.0 的 TLP 里面新增的 Segment 域段是如何分配的？这个 Segment 是 8 bit 的，和 16 bit 的 Hierarchy ID 有何关系？

根据协议，Flit Mode 中的 Segment Number 就是此前版本中的 Hierarchy ID，这个 Segment Number 是在枚举期间由软件分配的。可以肯定的是，16b 的 Hierarchy ID 包含 8b 的 Segment Group Number，不确定其余 8b 是预留还是留作他用。

81. Memory 事务中的 Requester ID 有什么作用？

Memory 事务是按地址路由，Requester ID 在 Memory 事务的路由中作用不大，但当 Completer 在反馈 CplD 时需要依据该 Requster ID 把 Cpl/CplD 反馈给 Memory 事务的请求方。

82. Completer ID、Requester ID 与 BDF 的关系是什么？

TLP Header 中的 Requester ID, Completer ID 用以在 PCIe Hierarchy 中唯一标识生成、完成该 TLP 的 PCIe Function，两者均由 Bus Number、Device Number 及 Function Number 组成的 （ARI 没有 Device Number）。Requester 的 BDF 称为 Requester ID，Completer 的 BDF 称为 Completer ID。

对于 MWr 等基于地址路由的转发事务而言，Requster ID 可有可无；对于 MRd 等基于地址路由的非转发事务而言，要求有 Reqesuter ID，其 Completion 中携带有 Completion ID 及 Requester ID。

TLP 路由

83. PCIe 有哪几种路由方式？

PCIe 路由一般指 TLP 路由，DLLP 只在相邻设备 DL 间传递，不携带路由信息也无需路由。PCIe TLP 路由方式有基于地址、ID 的路由及隐式路由 3 种，各组件根据 TLP Header 中的相关字段决定以何种方式路由该 TLP。

地址路由：基于设备地址的路由，适用于 Memory 及 IO 请求（含 DMWr 及 Atomic）。Address 位于 TLP Header 的 Address 字段。基于地址路由的 Memory 请求，地址可以为 32 bit 或 64 bit 地址；IO 请求采用 32 bit 地址。

ID 路由：基于设备 ID 的路由，适用于 Cfg 请求及 Completion。ID 位于 TLP Header 的 BDF 字段（对于 ARI 而言，只有 BF 没有 D）。

隐式路由：适用于广播 Message，比如：中断、Error、电源管理、功率闲置、Lock 事务等消息及厂商自定义的消息。

84. Message 是如何路由的？

Message 路由方式有三种，以隐式路由为主，但也支持基于地址或 ID 的路由。TLP Header 中 Type 字段（10r2r1r0）指定了该 Message 的路由方式，如表。

▼ 表：Message 路由

r2r1r0	描述	Message TLP Header Byte8~15
000	隐式路由（路由到 RC）	Reserved
001	地址路由	地址
010	ID 路由	BDF 等
011	隐式路由（RC 广播）	Reserved
100	隐式路由（本地消息，止于接收端，Legacy PCIe 采用这种路由方式传递 PCI 中断消息）	Reserved
101	隐式路由（收集并路由到 RC，目前仅用于电源管理 PM Message 中的 PME_TO_Ack）	Reserved
110~111	Reserved，止于接收端	Reserved

对于与 RC 之间通信的 Message，RC 为 Message 的发起者或接收者，没必要指定地址和 ID，采用隐式路由。目前除了 Vendor Define 的 Message，还没有基于地址进行路由的 Message 类型。

TLP 包处理

收发处理

85. 几种不同的 TLP 同时要发送时候，该怎么发？

若 Cfg/Mem/IO Write/Read、Completion、Message 等几种不同的 TLP 同时需要发送，Synopsys 推荐按照以下优先级进行发送：

- ✓ 当前传输中的 TLP 请求的 Completion
- ✓ Retry Buffer 内需要重传的 TLP
- ✓ PCIe Controller 内部事务层新生成的 Message
- ✓ PCIe Controller 内部事务层新生成的 Completion
- ✓ PCIe 用户接口新生成的待发 TLP

86. 多端口按时间窗口循环发送，既定时间窗口内发不完当前的 TLP 包会怎么办？

不太清楚这种机制。个人理解，TLP 没发完的继续发完再切窗口吧。

87. IO Write 是不是只能由 RC 发出不能有 EP 发出？

Legacy EP 可以，PCIe EP 不可以。

88. TLP 请求可以跨越 4 KB 地址边界吗？

不能。数字总线协议中绝大多数都规定请求不跨 4 KB 边界。如果遇到需要跨 4 KB 边界的访问，需要拆包。不仅跨 4 KB 需要拆分，超过 MPS、MRRS 也需要拆包。

89. RC 收到 EP 发来的 Cfg 包后会回复什么？

回复 Cpl.UR。

90. PCIe 发包有什么能控制到 Memory Type 吗？比如 Device-GRE Device-NGRE 等等。

GRE 释义如下：

- ✓ Early Write Acknowledgment-An early acknowledgement for a write transaction can be given before the transaction reaches the peripheral.
- ✓ Reordering-Transactions to the same peripheral can be re-ordered with respect to each other.
- ✓ Gathering-Multiple transactions can be merged into a single transaction.

PCIe Header 没法控制内存属性，得看 MMU、SMMU 配置的内存地址属性，有个转换关系。PCIe 不修改 mem 属性，但 SMMU 可以修改 mem 属性，由 SMMU 做 gre 到 mem 属性的转换。

PCIe 中有个类似的功能，通过 TPH 中的 Steer Tag 来控制。我一般拿 TPH 告诉上面需要 Stash。

从 SMMU 的 LA 是不是拿不到，只能去 LR 获取？是在配置 STE 和 CD 时设置的。

91. MRd、MWr TLP 中是否需要同时指定 RID 和对方的 ID？

ID 路由按照 BDF 号找；地址路由按照 BAR 地址找。对于地址路由的 Memory 请求不需要对方 ID，其根本不知道是给谁的，因此请求中无需指定 Completer ID；Completion 是 ID 路由，需要根据 Mem 请求中的 Requester ID 将 Cpl/CplD 路由回请求方，因此 Mem 请求中需要携带有有效 Requester ID。对于 Cpl/CplD 而言，其中携带有 Completer ID，该 ID 不用于路由，而是用于传输匹配、乱序处理、错误定位等。

以下为 AI 对 Completer ID 用途的解释：

PCIe Completer ID 字段是 PCI Express 规范中的一项功能，用于指示 PCIe 设备生成的传输完成数据包所对应的传输请求。该字段将原始请求的 Tag 信息传回给发送者，用于匹配传输请求和完成数据包。完成数据包包含了传输的结果和状态信息。

Completer ID 字段的主要用途如下:

- ✓ 传输匹配：当一个 PCIe 设备对请求做出响应时，Completer ID 字段将原始请求的 Tag 信息传回给发送者。发送者可以根据 Completer ID 字段来匹配传输请求和完成数据包，以确定请求被正确处理。
- ✓ 乱序处理：PCIe 中支持乱序的传输，Completer ID 字段用于确保返回的完成数据包按照请求的顺序进行匹配。通过 Completer ID 字段，发送者可以按照请求的顺序重新排列完成数据包，以确保数据的正确性。
- ✓ 错误处理：Completer ID 字段还可以用于处理传输错误。如果传输过程中发生错误，如超时、数据异常等，Completer ID 字段可以用于确定传输错误所对应的传输请求，以便进行错误处理和故障排除。

总之，Completer ID 字段在 PCIe 传输中起到了重要的作用，用于传输匹配、乱序处理和错误处理确保数据的正确性和完整性。

TLP 类型	路由方式
Memory Read/Write	地址路由
Cfg Read/Write	ID 路由
Completion	ID 路由
Message	地址路由、ID 路由、隐式路由

92. NP 请求需要 Requester ID，那么 MWr 等 Posted 请求中 Requester ID 有用吗？

有用。Requester ID + Tag 组成 Transaction ID，能够唯一标识一个 Transaction。如果系统支持的 Outstanding 数目特别大、仅用 Tag 不够，可以在开启 Phantom/Shadow Functiom 的基础上基于 Requester ID 字段来扩展 Outstanding 数目。此外，在排序时也会用到 Requester ID。

93. Requester ID 不属于该设备的 Subordinate Bus 会不会被拒收？

Memory 事务是地址路由，RC->EP 这个方向的 Requester ID 本来就不在 EP Sub Bus 下，肯定不会被拒收；EP->RC 这个方向上，如果没开 ACS，不会被拒收，开了 ACS 会被拒收。

94. 收到不支持的 Message 是如何处理的？

回复 UR。不支持的 Message 是指 Message Code 未识别。

95. 当同一 Requester 有多笔 MRd 发出时，加之 Flow Control 中 MRd、CplD 不保序，如何知道 CplD 回的是哪笔 MRd 的数据？

对于不同的 Completer 或 VC，可以根据 CplD TLP Header 中的 TC 和 Completer ID 来判断该笔 CplD 归属于哪一笔 MRd。

对于同一 Completer 同一 VC，问题所言的情况（多笔 MRd，MRd/Cpl 失序）不会发生。Flow Control 机制对于不同 VC 不保序，但是对同一 VC 是保序的，MRd 在 Flow Control NPR Buffer 中按照先入先出的顺序发出。

96. 开启 AER Error Reporting 之后，如果 EP 收到了一个 Type 1 类型的 Config 包，EP 是不是既要回复一个 UR 包又要发生 Error Message？

是的，根据 PCIe Base Spec 6.2.3.2.4.1 Completer Sending a Completion with UR/CA Status 解释，需要同时回复 Cpl.UR 和 Error Message。

RCB/MPS/MRRS

97. MRRS 是什么？跟 MPS 有什么区别？

MPS 只跟 MWr 和 CplD 等带 Data Payload 的 TLP Length 相关。

MRRS，Max Read Request Size，最大读请求长度，本端发出 MRd TLP 的 Length 大小不能超过 MRRS，超过后就拆分为多笔 MRd 发出。除了 MRRS 的限制，做得好的 EP 还会根据 RC 配置的 RCB 切 MRd（非强制要求），这样 RC 回 Cpl 效率会更高。RCB 可能就是 RC Cacheline 的长度。

98. MPS 和 MRRS 有什么关系吗？

MPS 跟 MRRS 没有直接关系。一般设置 MRRS 小于等于 MPS，如果大于的话回来的 Data Payload 可能存不下。

99. MRRS 最大读请求 Size 是对自己的要求还是对链路对端的要求？

MRRS 是对自己的要求，即自己不能发出 Payload 超过 MRRS 的读请求。MRRS 跟发送端的 Buffer 有关，这个缓存一方面用来，存放发出的 MRd 的 TLP Header 中的相关信息，一方面存放该 MR 对应的 CplD。在发出请求之前，本端需要确保有足够的 Buffer 来存放 CplD 中的 Data Payload，用于判断 CplD 是否回完整了，以及把同属一个 MRd 的、拆分开的多笔 CplD 合并起来。

100. 已有 MPS 来限制包的大小了，为什么还需要 RCB 来拆分 CplD 呢？

当 CplD 载荷超过 MPS 时必须拆分，载荷在 MPS 及以内时可以只回一笔不拆分。

RCB 可以规范 Cpl 的最小拆分粒度。每个 Cpl 都需要一块空间来放 Header，如果拆包粒度太小，最大拆包数量就太多了。有了 RCB，最大可能的回的 Cpl 的个数也就确定了，便于规划相关 Buffer Size。可以参考外网下述解释：

To make room for packet-header overhead, the user application must allocate enough space for the maximum number of completions that might be returned. To make this process easier, the PCI Express Base Specification quantizes the length of all completion packets such that each completion must start and end on a naturally aligned read completion boundary (RCB).

101. 为什么 EP 的配置空间内也有 RCB Field？

因为 RC 的 RCB 可以是两种，所以 EP 要存下 RC 的 RCB，以便 EP 检测 RC 回的 Completion，此外 EP 可以根据 RCB 来拆分自己的 MRd，以便 RC 更高效的处理收到的 MRd 回 Completion。

102. PCIe 协议对 MRd 请求返回的 CplD 数量有限制吗？

协议没有明确限制一笔 MRd 能拆分为多少 CplD 进行回复，只是说可以拆分，但存在理论上的最大值。

单笔 CplD 的最大 Length 不能超过 1024 DW，即 Payload 不能超过 4 KB，这也是对所有带有 Length 字段的 TLP 的要求。

单笔 CplD 的 Payload 还受 MPS 的影响，如果 MRd 请求的数据量超过了 MPS，则需要拆分为不超过 MPS 的多笔 CplD。拆分 CplD 时候，需要满足 RCB 的要求，即地址边界需要为 RCB 的整倍数。

举个例子，单笔 MRd 请求读 2 KB，MPS=512 B，RCB=128 B，理论上最少需要回复 4 个 512 B 的 CplD，但也可以拆为 8 个 256 B，或者 16 个 128 B，以及其他 128 B 整倍数的组合，但 CplD 的数量最多不会超过 16。

103. PCIe IP 内部的 Buffer 深度对 MPS 有影响吗？MPS 设置有何讲究？

Buffer 深度对 MPS 的设置肯定有影响。对 TLP 而言，其 Payload Length 理论最大为 1024，但实际 MPS 受限于 PCIe IP 内部 Buffer 深度。IP Buffer 深度一般设置为 256 DW 或者 512 DW。

除了 Buffer 深度，从系统性能角度看，为了获得更改的系统性能，设置 MPS 时还需参考 MTU、RCB 的值。

104. MPS 的值对性能有影响吗？

MPS 对性能有一定的影响。MPS 决定了数据吞吐量，如果 MPS 比较小，那么小包就多，性能就低。考虑到 MPS 对 Buffer/Memory 也有要求，如果 MPS 设置太太，所需的 Memory 也多，对性能提升帮助不大。在 Gen5 之前，MPS 一般设置为 256 或 512。

在 Synopsys PCIe IP Spec 附录有讲 Buffer 大小、MPS 对 PCIe 效率提升的对比，可以参考。

105. 相比于 RCB=64，把 RCB 设置为 128 是否效率更高？

不是，RCB=128 不一定比 RCB=64 效率高。

举个例子：EP 请求从 RP 中读 4 KB 数据，MPS=512。不管 RCB=64 还是 128，可能都是回来 8 个 512 B 的 CplD，在 CplD 包数量上无甚区别。在 RP 侧，4 KB 数据也是拆了之后才送给 RP PCIe Controller 进行 CplD 组包的，这 4 KB 数据过 ACE LITE 接口要被 MTU 拆一次，更往请求上过 CMN 时也是拆成 64 B 一个个去处理，CMN 回下来也是 64 B 粒度逐笔回下来。RCB=128 能拆的组合，RCB=64 都能拆成相同的组合，显然 RCB=64 在拆包、组包时比 RCB=128 更灵活，在使能 Data Chuning 时，性能反而更好。

106. MPS 怎么设置，和 LCRC/ECRC 有关系吗？

MPS 设置和 LCRC/ECRC 没关系，配置空间的 Capability 寄存器中指示当前设备所支持的 MPS，可通过固件配置来查看。

107. 链路上不同节点的 MPS 不同怎么办？

补充：假设一个 PCIe 设备 MPS 是 128 B，DMA 写主存，经过两级 PCI 桥，两个桥的 MPS 都是 256，那么 MWr TLP 经过第一个桥的时候 MPS 是 128，经过第二个桥的时候是多少呢？直接将 TLP 发到上游，还是两个 TLP 合并成 MPS 是 256 的包向上发？

Device Control/Capability 寄存器内有 MPS 信息，枚举完后采用整个系统中最小的 MPS 作为最终 MPS。

108. MPS/MRRS 和 Byte Enable 有关系吗？拆包时是否需要综合考虑两者？

问题补充：比如 MPS 是 512 B，如果请求 512 B 数据但地址没按 DW 对齐，Length 会变成 129 DW 吗？这样算不算超了 MPS？要不要拆包？

TLP 中的地址一定是 DW 对齐的，地址低 2b 被别的字段占用了。DW 非对其通过 BE 来指示。如果一定要从非 DW 对齐地址开始往后读写 128 DW，实际需要 129 DW。超不超 MPS 是按照 Length 算的，对于读/写，Length=129 DW 超过 MRRS/MPS=512 B 了，那么就需要拆包。

109. 如果 Requester 发出的 MRd 超 MRRS，Completer 收到后一般怎么处理？回 UR？

MRRS 是给发端发 NP Req 时切包用的，只对发端有约束，对端不 Check 是否满足 MRRS。

110. RC 的 RCB 可配吗？RC 返回包给 EP 的时候，能不能固定拆分方式呢？

问题补充：比如 EP 读请求包的长度是 512 B，双方支持的 RCB 是 64 B，MPS 是 256 B，能否通过设置固定 RC 的返回包一定是两个包，每包数据大小是 256 B 呢？

RCB 可以配置，RC 通常配 64 B，兼容性好。RCB 是 boundary，能不能返回固定的 256 B，取决于所请求的地址、长度及拆包策略。如果请求包的长度超过 MPS，则肯定会被拆包，拆包地址边界为 RCB 对齐，最小包粒度为 RCB。显然让返回包的数据长度固定为 MPS 大小提高带宽利用率，但至于能否固定 Size 为 MPS，理论上是可以的，取决于 Controller 拆包策略。

对，读请求包里地址都是 RCB 对齐的（低 8 bit 都是 0），但是返回包有多种拆分情况。一个 Tag 最小的返回包 payload 是 64 B，总共是够读请求包里的长度的。

RCB 相关介绍可参考：【PCIe】PCIe 读完成边界 (RCB) 介绍_PCIe RCB-CSDN 博客

111. RC 和 EP 的 MRRS 不同时，应该遵从谁？

问题补充：RC 端的 MRRS 是 256 B，EP 端的 MRRS 是 4096 B，那 EP 端发送 MRd 的 Length 能超过 256 吗？这个限制是遵从本端还是对端？

取决于 CPU 那边的设计了：如果严格按 Spec 实现，MRRS 是不限制 Receiver 侧的，即 MRRS=256 的 RP 依然可以收 4 KB；如果不严格按 Spec 设计 RP，为了节约资源，要在实现文档上说一下，不支持收到超出 MRRS 的。

追：EP 发出超过 256 B 的 MRd，RP 侧回了 UR。

附：If software chooses to program the Max_Payload_Size of various System Elements to non-default values, it must take care to ensure that each packet does not exceed the Max_Payload_Size parameter of any System Element along the packet's path. Otherwise, the packet will be rejected by the System Element whose Max_Payload_Size parameter is too small.

112. CplD 拆包必须要满足 RCB 的规则吗？是拆包按 RCB 对齐还是按 RCB 的 N 倍？

必须满足，可以是 RCB 的 N 倍，它是一个地址对齐概念。

113. 回 CplD 拆包时，具体按 RCB 的 1 倍还是 N 倍，是有寄存器去设置吗？

协议未要求，取决于设计。根据协议，有多种拆包方式，满足在拆包边界 RCB*N 地址对齐即可。

114. 当 MRd 请求超过 MRRS 时，是否可以先改大 MRRS 再一次发出 MRd？

问题补充：如果我想读 4 KB 数据，MRRS=512 B 时至少需要发 8 次读请求。我是否可以通过设置自身 MRRS=4 KB，这样发一次读请求就可以了？

根据 PCIe Spec，MRRS 是个 Control 寄存器，而非 Capability，无法在 Link 两端进行协商。MRRS 是对 Requester 即发送端的限制，其对 Completer 即接收端没有限制。比如，对于 MRRS=64 的 RP，其仍然可以接收 4 KB 的读请求。

作为发送方，其可以通过设置较大的 MRRS 来减少发送读请求的数量，但实际使用中不一定是一次请求的数据越大性能越好。作为接收方，出于资源考虑或本身理解实现的错误，有些 CPU 会在产品说明书上写明其支持的最大读数据量。

115. PCIe 回 CplD 时单个 CplD 的 Payload Length 是不是也不能超过 MPS？

是的。如果收到的 CplD 超过 MPS，可能会报 Malformed TLP 触发 UCE 并 Hang 机。

116. Rx 端会检查接收到的 MRd 中的 Length 是否超没超自己 MRRS 吗？

MRRS 是 Tx 的自我约束，约束自己发出的读请求不超过 MRRS，PCIe Base Spec 里未要求要不要检查 Rx MRd Length 跟自己的 MRRS，各家实现方式不一，一般不检查，不过听说 PLDA 的会检查，超过之后当无效处理。

117. 在主机 Linux 上使用 devmem2 工具给设备发送 MRd，默认读长度只有 2 DW，是因为受 64b 操作系统影响吗？

工具原因，devmem2 一次读最大只能到 64b，大于 64b 要用专门的汇编指令，找其他工具。

参考：https://blog.csdn.net/weixin_40837318/article/details/130198897

118. PCIe MWr 有要求跨 256 B 地址边界拆分 TLP 吗？

PCIe Base Spec 中没有要求不能跨 256 B 地址边界。如果发现发出的 TLP 按照 256 B 地址边界做了拆分，可能跟 MPS、MTU 等设置有关。

119. PCIe RC 的 Max_Payload_Supported 一般设置为多大？

512 B。

120. CplD 可以按照 RCB 和 MPS 进行拆包，一般以哪个为准？

补充：正常 MPS 比 RCB 大，那 RCB 还有何用处？

这两个都要用。如果要对 CplD 进行拆包，拆包地址边界应该为 RCB 整数倍。无论拆不拆包，单个 CplD 内的 Data Payload 都不应该超过 MPS。

121. MTU 一般设置多大？设置为 MAX 吗？

一般需要大于 SoC 中的 AXI 数据位宽，最高 4 KB，但并非越大越好。设置 256 B 的比较多，512 B 的也有，再大未见过。如果是类 NVLink 自己连自己，肯定大点好，但是也要注意面积的均衡。

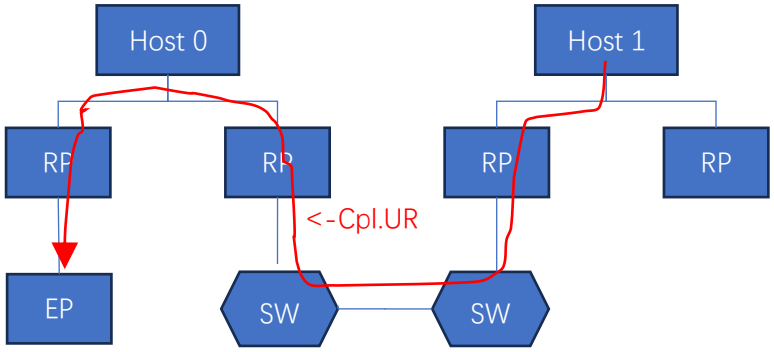
个人理解，对于 DMA 场景而言设置 512 B 好点，512 B 基本能覆盖对接 Host 的场景，再低了 DMA 利用率下降、带宽利用率降低，其他没什么影响。DMA 做 H2D 时是走 MRd，基本和 MPS 无关；D2H 时走 MWr，MPS 越大 Overhead 越小。

122. MRRS 是怎么设置的？

这个好像是被动的，由其他几个参数设置自动计算的。

CA/UR

123. 跨主机访问时 MWr 正常，但 MRd 回 UR。该地址在本机读正常，另一主机过 Switch 来访问本机该地址写正常，读全 F 且 UR，有何定位思路？



补充说明：测试链路图如下。该现象仅出现在跨 RP 交互的时候，设备和 Switch 在同一个 RP 下没问题。CPU 的 RC 支持 P2P，且 RC1 和 RC2 的 ID 不同、两边 Host 的 Bus num 不同，NT 的 ID 查找表配置成功且正常，Config Style 正常。

一般情况下，如果读的地址不能读或者不存在，就会返回 UR。鉴于本机可读，可排除该情况。

根据上图，该测试场景为 B2 B 连接，即 NT 对 NT。配置分区或者 Crosslink 了么。地址和 id 都是重叠的，无法访问。这种连接，枚举也会遇到问题，根据配置的 Upstream 和 Downstream 实际只有一边生效。

可尝试所有路径上的 Switch 及 EP 的 MPS 和 MRRS 设置为相同值。

可排查 RC 是否支持 NP P2P。有些 RC 只支持 Posted Req 的 P2P 访问。

可排查 Addr Base 和 Limit，有没有开 SMMU。

猜测 MWr 也未成功，只是 MWr 是 Posted 可以 Drop 且没有 Response 回来。

124. 请问 UR 类型的 Completion 可以是 CplD 吗？还是说只能是 Cpl？

下文为 Spec 内关于 UR 的解释：

✓	A statues that applies to a posted or non-posted Request that specifies some action or access to some space that is not supported by the Completer.
✓	A statues indication returned with a Completion for a non-posted Request that suffered an Unsupported Request at the Completer.

Cpl.Status=UR 时不带 Data，即 UR 对应 Cpl，非 CplD。虽然 Cpl.Status=UR 不带数据，但其 Byte Count 仍应按照规则给，即剩多少 Byte 就给几，而不是给 0 值。

在枚举过程中，若不存在某个 Port，会回复一个 Data Payload 全 1 的 Cpl 来指示设备不存在，这也是一种 UR。有些系统通过 CplD 中的 Data 是否为全 1 来判断是否为 UR，此时 Cpl.Status=SC，但系统配置软件仍需要认为当前 Completion 状态为 UR。

虚通道/服务质量

125. 一般需要开几个 VC Channel？

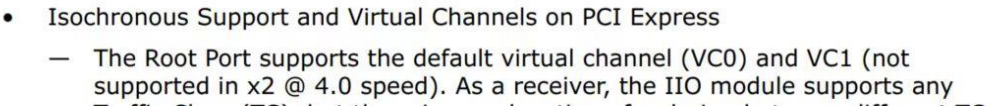
大部分需求应该 1 个就够了，大部分设备都是开一个 VC。个人理解，需要全链路均支持多 VC 时多 VC 才有实际作用。

126. 哪些场景会用到多 VC（Virtual Channel）？

需要提高某些数据的优先级时可以用多 VC，需要整条链路都支持 VC，否则多 VC 没有意义。PLX 好像支持 2 VC，网络、存储 NVMe、HPC、显卡等都有用到。

127. PCIe 的多 VC 功能的使用情况如何？

- ✓ 一般就一个 VC，Server CPU 会有 2VC。最多见过也就俩 VC，没见过超过 2VC 的。
- ✓ 多 VC 会让 Order 变得更复杂，即便实现了多 VC，Queue 的资源后端也不接受。
- ✓ 我记得我看到的第二个 VC 是特殊用途的，不是给一般 EP 用的。
- ✓ 用过 VC1，做 CCIX 用的。
- ✓ 你去看看 CPU 支持程度，就可以把这个否掉了。
- ✓ DMA 方向见过有多 VC，不过内部是 Virtual Link。



128. PCIe（RC）上来的包有个 AXI QoS Field，这个一般固定给什么值？

固定值的话无所谓给什么值，默认给个 0。

129. PCIe 的 VC/TC Mapping 字段和 VC ID 字段需要在 VC Enable 之前提前配好吗？

需要提前配好。

事务排序

Tag

130. PCIe TLP Header 中的 Tag 应该怎么理解呢？连续发起 MRd 是否会改变 Tag 的值？

Tag 跟 Outstanding 有关。对于 NP 请求，在 Response 回来之前及 Tag 用光之前，哪怕地址相同，Tag 也会一直增加。

131. 10 bit Tag 为何从 256 开始，为何不和 8 bit 一样从 0 开始？

附：With 10-Bit Tags, valid Tag[9:8] values are 01b, 10b, or 11b. 10-Bit Tag values with Tag[9:8] equal to 00b are invalid, and must not be generated by the Requester.

Tag[9:8]是用来标记当前为 10b Tag 的。某一 Requester 发送了 10b Tag，Completer 如果不支持 10b Tag，其会丢弃 Tag[9:8]只保留 Tag[7:0]并在 Cpl 中回复 Tag=Tag[7:0]。如果 10b Tag 从 0 开始，Requester 就无法得知其收到的 Tag 是 8b 还是丢了高 2b 的 10b Tag，造成误判。同理，14b Tag 的 Tag[13:10]也不能为 0。

协议里也给出了大致解释：This enables a Requester to determine if a Completion it receives that should have a 10-Bit Tag contains an invalid Tag value, usually caused by the Completer not Supporting 10-Bit Tag Completer Capability.

132. 为何 10 bit Tag 是 256 到 768，而不是跟 8 bit 重用/兼容从 0 开始？

先纠正问题，10b Tag 是 256~1023，共 768 个 Tag，不是 256~768。

为了给 8b Tag 设备留点可用 Tag。比如 RC 下挂了两个 EP，一个支持 10b Tag 一个支持 8b Tag，0~255 全发到 10b Tag 设备了，接下来给 8b Tag 设备发就没有可用的了。

10 bit Tag，最高 2 bit 不能为 0，所以不是从 0 开始吧。如果你发一个 10 bit Tag 给不支持 10 bit Tag complete 的设备，他会返回一个最高两 bit Tag 为 00 的 Tag，这样来提示自己不支持改特性。如果 10 bit Tag 中最高两 bit 允许为 0 的话，就实现不了这个功能了。

133. 为什么 10 bit 的 Tag 只能用 768 个？ 不应该是 1024 个吗？

https://www.synopsys.com/designware-ip/technical-bulletin/accelerating-32gtps-PCIe5-designs.html

回答同上。

134. TLP 中的 Tag 能不能选择 512？

10 bit Tag 模式时可选择 512，其他 8b、14b Tag 不能选择 512。

135. PCIe 的同一个 Tag 需要间隔多久才能重复使用？

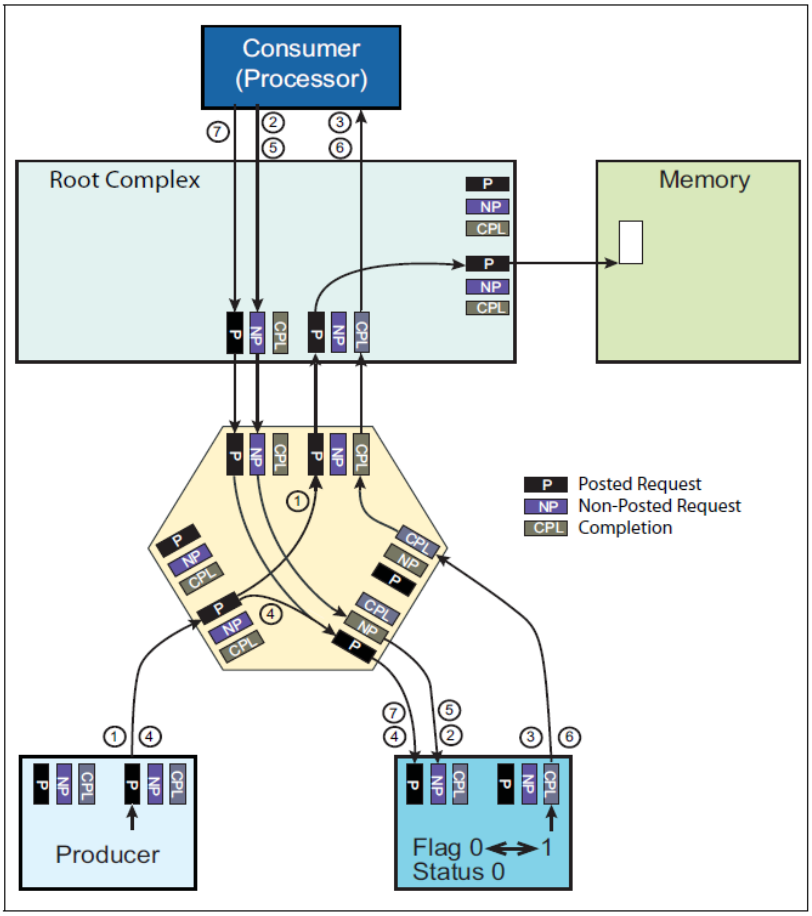
对于 NP 请求，Tag 的重复使用不是看时间间隔，而是看该 Tag 的 CplD 有没有回来，回来表示释放，释放了才能用。

排序规则

136. PCIe 排序规则中 NP 的 Cpl 为什么一定要等前面的 P 都要完成才能返回？

问题不正确。根据 Mindshare PCIe Technology Table 8-1: Simplified Ordering Rules Table 中的 D2a/D2b，原因是为了满足生产者、消费者模型，例如 Figure 8-2 所示的情况；相同 Req_ID 的 Cpl 不能超越 P，IDO=1 且 Req_ID 不同时 Cpl 可以超越 P。同理，A2a/B2a/C2a 规则中的 P/N 均不能超过此前的 P。

Figure 8-2: Producer/Consumer Sequence Example — Part 1



137. PCIe 排序规则中的“Y/N”意思是可以超过、也可以不超过吗？

是的。这是针对 Performance 的。

138. RO=1 的 TLP 可以超过任何 TLP，那么 RO=0 的 TLP 可以超过 RO=1 的 TLP 吗？

按照 PCIe Base Spec2.4 节里的 Ordering Rules Summary 表格排序即可。该表格是针对相同 TC 而言的，不同 TC 之间没有排序要求。

139. 不同 ID 的 TLP 排序需要考虑 TLP 类型吗？

问题补充：对于 IDO=1，书中提到不同的数据流（ID）可以“自由乱序”，未要求 TLP 类型，但是又存在一个排序表约束了各种 TLP 包的序，比如 A3，这不是自相矛盾吗？

附：IDO 模型由 PCIe V2.1 版本引入。该模型引入了“数据流”的概念，即相同的数据源发出的 TLP 属于相同的“数据流”，而不同数据源发出的 TLP 属于不同的“数据流”。PCIe 链路可以根据“数据流”对 TLP 进行区分。IDO 模型允许分数不同“数据流”的 TLP 之间没有序的要求，即可以自由乱序。（源于王齐《PCI Express 体系结构导读》11.4 节）

Can Row pass Column? (Col 1)		Posted request (Col 2)	Non-Posted Requests		Completion (Col 5)
			Read Request (Col 3)	NPR w/ data (Col 4)	
Posted request (Row A)		a) **No** (unless b) applies) b) **Y/N** (if RO=1 or IDO=1 and different ReqID)	**Yes** (deadlock avoidance)	**Yes** (deadlock avoidance)	a) **Y/N** (unless b) applies) b) **Yes** (special case for PCI/PCI-X bridges)
Non-Posted Requests	Read Request (Row B)	a) **No** (unless b) applies) b) **Y/N** (if IDO=1 and different ReqID)	**Y/N**	**Y/N**	**Y/N**
	NPR w/ data (Row C)	a) **No** (unless b) applies) b) **Y/N** (if RO=1 or IDO=1 and different ReqID)	**Y/N**	**Y/N**	**Y/N**
Completion (Row D)		a) **No** (unless b) applies) b) **Y/N** (if RO=1 or IDO=1 and different CmpID / ReqID or if completion for IO or Config Write)	**Yes** (deadlock avoidance)	**Yes** (deadlock avoidance)	a) **Y/N** (for completions from different read requests) b) **No** (for completions from same read request)

这个表应该是按照同 TC 下的包来做的。对于 IDO=1，不同 TC 间不存在排序关系，相同 TC 时，即便 ID 不同仍然需要遵循图中的排序规则。整体来看，图中对 IDO=1 的情况开了很多绿灯（Y/N），只在几处为了避免死锁需要 IDO=1 时遵循先排序规则。

140. PCIe 的 SO 强排序在系统级能保证到 DDR 入口也是 SO 的吗？

问题补充：我看 PCIe 上去后地址和 ID 不同 Hash 到不同 SNF 节点，无法保证 SO 了。

仍然可以保证 SO。RP 控制器发出 SO 的前提同一个 AXI Port 以及相同的 AWID，系统保序点在 HN，即使 SO 请求按地址被 Hash 到多个 HN，CHI 协议的 OWO 任然可以保证 PCIe 的 SO 请求可以按照 Requester 发出的顺序被观测到。为实现该功能，需要在 RN 上配置一个 Field，确保其可以识别出跟其对接的为 PCIe。

141. 为什么 Posted 请求在阻塞情况下要超车 Non Posted 请求，这两种请求有关联吗？为什么会造成死锁？

注：如下图 PCIe 排序中的 A3、A4 规则。

Row pass Column? (Col 1)		Posted Request (Col 2)	Non-Posted Request		Completion (Col 5)
			Read Request (Col 3)	NPR with Data (Col 4)	
Posted Request (Row A)		a) No b) Y/N	Yes	Yes	a) Y/N b) Yes
Non-Posted Request	Read Request (Row B)	a) No b) Y/N	Y/N	Y/N	Y/N
	NPR with Data (Row C)	a) No b) Y/N	Y/N	Y/N	Y/N
Completion (Row D)		a) No b) Y/N	Yes	Yes	a) Y/N b) No

Non Posted 阻塞可能是外设的问题，不是发生在 Switch 中，不应该影响其他的传输。如果 Non Posted 发生阻塞时把 Posted 的也给阻塞掉是很不好，Posted 不超车可能就死锁了。

142. 排序时，Posted 包超越 Non-Posted 包的位置是在 VC Buffer 中超的吗？

不是，Buffer 只是一个存包的地方，不做排序。P、NP 包放在独立的 Buffer 中，外部的排序逻辑依照排序规则来 Buffer 中取包。

143. NP Read 之间可以相互超越不保序，其保不保序跟 Relaxed Ordering 有关吗？

无关。

144. EP 收到一笔 MRd 后分多笔 CplD 进行回复，同一 MRd 对应的 CplD 采用相同 Tag，不同 MRd 对应的 CplD 可以交叉回吗？

可以。Transaction ID（含 Tag）不同的 CplD 之间可以乱序，相同 Transaction ID 需要按顺序返回。

145. EP 发出的两个读请求，后面的会不会超过前面的？

根据 PCIe 排序规则，允许两个读请求之间互相超越。

146. PCIe 链路上相同地址的读请求之间会保序吗？

根据 PCIe 排序规则，允许两个读请求之间互相超越，即便地址相同也可以相互超越。PCIe 链路上是按照包类型排序的，而非按照地址。

即便 PCIe 链路上能够保序，考虑到端到端之间多径的可能，PCIe 同地址读请求之间仍无法保序。若想保序，需要请求者自行维护。

排序应用

147. DMA 乱序需要处理吗？

根据协议，肯定需要。一般 EP 需要处理乱序的问题，具体要看 EP 是否遵循 Spec 要求，实际应用中碰到过设备本身设计不遵循 PCIe 规范、也就是收到的包不会重排序的情况。这种乱序多在 EP 主动发起 DMA Read 的时候会遇到。

148. EP 发出的 Mem 请求可以直接桥接到 CMN 上，然后通过 TLP 带的原始地址来路由到同样挂接在 CMN 上的目标模块吗？还是说 TLP 里面带的地址不能直接用，需要先改一下？

不能直接利用 TLP 的地址将其路由到 CMN 上的其他目标。TLP 内的地址是 Host 分配的，和 CMN 地址是两个地址域。

149. 可以通过设置 Slave 端相同的 AXI ID 来实现两笔 MRd 的保序吗？

可以，但不建议这么做。靠总线 ID 保序限制较多，最好在 Master 端做保序，不要依靠 Slave 端的总线。

150. PCIe Controller 入口和出口 AXI Bus 接口处是如何保序的？

问题补充：例如 EP 访问 Host DDR，先 MWr 后 MRd，在 EP 入口和 RP 出口后，如何保证到达 Host DDR 是严格按照先 MWr 后 MRd 的？

在 PCIe 拓扑内会保证 NP 类型的 MRd 不会超越 P 类型的 MWr。在 SoC 上，NoC 应该可以做读写合并，确保同一 Master 进入 NoC 的请求顺序得到保证；请求到达 DDR 后在 DDRC 内也会做保序，确保先进来的同地址写和后进来的同地址读不会被优化乱序。

数据完整性

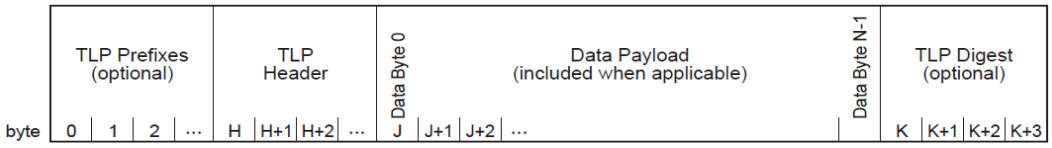
ECRC

151. TLP Digest 是什么？

Digest，又名 ECRC。

TLP 包格式：Prefix + Header + Data Payload + Digest。TLP Digest 便是这 TLP 包的组成部分，位于 TLP 末尾，长度 1 DW，存放了该 TLP 的 ECRC 值。

TLP 四大组成中，除了 Header，其他都是可选项，包括 Digest。TLP Header 中有一个字段 TD (TLP Digest)，用以标记该 TLP 是否使用 Digest，或者说该 TLP 是否使用 ECRC。如果 Header 显示该 TLP 不含 Digest，但最后收到了 Digest，这个 TLP 就是个畸形包，要上报错误。



▲图： TLP 包格式

Error Poison

152. Poisoned TLP 是什么？是干什么的？

Data Poisoning 或称 Error Forward，是一种事务层报告错误的方法，报告错误时发送的 TLP 为 Poisoned TLP。Poisoned TLP，有毒的 TLP，仅用于含有 Data 的 Request 或 Completion，比如写请求、MsgD、读完成、Atomic 请求及完成。

Poisoned TLP 通过 TLP Header 中的 EP 位来指示，EP=1 时，即表示当前 TLP 数据被破坏了。注意，是 TLP Data 被破坏了，不是 Header，Header 出错可不归他管，Header 出错那就是另外的故事了。Poisoned TLP 用于控制 Error 在整个系统内的传播，协助软件诊断该 TLP 路由路径上的 Switch 是否给该 TLP 贴上了有毒标签。

在生成这个 TLP 的时候就知道这是个 Data 错误的 TLP。既然知道 Data 出错，为什么还要继续发？—— 以读请求为例，Requester 发出读请求后，在等待 Completion，现在数据读回传了，但是 Switch 在半路就检测到 Data 出错了，你这 Completion 还要不要继续发？不发的话，Requester 还在等，直到 Timeout。倒不如给个错误信号，至少不

会让 Requester 久等。Requester 收到 Poisoned TLP，知道发生了事务层错误，至于这数据要不要用，都不违法。

153. 上游设备给下游 PCIe 设备发送的数据中毒 TLP 包头中的 EP bit 是谁置 1 的？

真实场景中由 Controller 发现并置起。如果是验证，要在 SII 中实现注错功能，或者 iATU 实现对该 Field 功能的注入。

154. EP 收到的 Poisoned TLP 后一般怎么处理？MWr 直接丢掉、MRd 回 UR 吗？

Poisoned 是对 Data Payload 而言的，即 Poisoned 会出现在 MWr 或 CplD 中，指示 TLP Data 有问题，TLP Header 没问题。PCIe 协议没有要求如何处理 Poisoned TLP，可以继续向前透传或者直接丢弃并上报错误。

完成超时

155. Completion Timeout 机制是什么？

PCIe 设备发出的请求中有些请求需要 Completer 反馈 Completion，此时 Requester 会等待 Completion 再进行下一步操作。在某些异常情况下，比如配置不当、系统故障等，Requester 无法收到 Completion。为了不影响进一步使用，需要一种超时退出机制让 Requester 从这种等待状态恢复过来，这就是超时退出机制。

几点注意事项：

- ✓ 发生 Completion Timeout 时，Requester 上报错误。
- ✓ 由于 Switch 仅用于交换请求不生成请求，Completion Timeout 机制仅限于用在 RC、EP 及 Bridge 上，不适用与 Switch。
- ✓ Completion Timeout 机制默认打开，用户可通过配置 Device Control 2 寄存器来关闭该机制。Completion Timeout Value 有几种选择，用户可通过配置 Device Control 2 寄存器进行选择。如果设备不支持可变的 Timeout Value 配置，那么在设计的时候，FLIT Mode 下应把 Timeout Value 设置为 40 ms~50 ms，非 Flit Mode 下应设置为 50 us~50 ms（建议最少 10 ms）。
- ✓ 对于有多笔 CplD 的 MRd 请求，若 CplD 被中断，此前 Requester 收到的 Data 可以保留也可以丢弃。

156. Completion Timeout 的值为何是一个范围，而不是一个固定的值？

因为在范围内要进行重传，范围下限以外重传太早是没有意义的。

157. 关闭 Cpl Timeout 后 DMA 发出 MRd 但收不到 Cpl 会有什么结果？会重新训练吗？

Cpl Timeout 是 Error Handling 机制，不建议关闭。如果关闭又遇到上述情况，会卡死 DMA，不会重新训练，因为 Cpl Timeout 不代表链路质量存在问题。如果 DMA 有相关保护机制的话，DMA 会报超时等异常。

158. Host 发一个读请求到 EP，EP 挂了没回 CplD 导致超时，Host 超时会挂么？

Host 可能会挂。有 DPC、eDPC 来管理的话可能不会挂。

159. Synopsys PCIe IP 在 FPGA 上作为 RC 扫描桥下设备，RC 能扫到桥内的 USP 及 DSP，但扫描不到桥下的 EP，RC 报 Cfg Timeout，可能是什么原因？

问题补充 1：确定 EP 和桥以及 RC 和桥之间建链成功且稳定，EP 能收到 RC 发来的 Cfg 包但 RC 报 Cfg Timeout。

问题补充 2：扫描 NTB 非透明桥是可以的（FPGA RC+桥片+PCI 设备），中间有 Type 0/Type 1 Cfg 包的切换和桥 0x18 寄存器相关桥寄存器的配置（Primary Bus，Sub- Bus 等）。

如果 EP 能收到 Cfg，可进一步确认 EP 有没有相关错误报出，判断 EP 是否正确解析处了该笔 Cfg 请求。如果 EP 没有任何异常报出，可进一步确认 EP 是否反馈了 Cpl/CplD、反馈的相关 Transaction ID 是否正确。若 EP 反馈 Cpl 正确，再排查 Switch、RC 的问题。

160. Host 遇到 Completion Timeout 错误时一般是什么反应？

现象：Host 遇到 Completion Timeout 后，有的服务器会挂掉重启，有的不会。

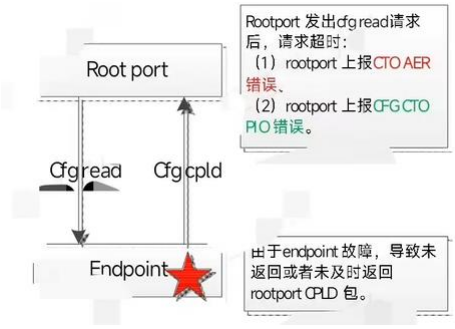
猜测跟服务器的错误处理机制有关，有的会把 CE 当成 UC 处理，导致服务器挂掉。

猜测跟 Link 饿死有关，大量 Req 发出后收不到 Cpl，Req Queue 一直占用得不到释放，Credit 用完之后也无法发出新的请求，时间一长发送端报出大量 Timeout。

猜测系统挂掉跟 DPC 有关，有 DPC/EDPC 的话应该好一点，有 EDPC 时超时会主动回一个 Cpl 包。

161. 请问 Xilinx 的 IP 完成包超时会不会造成不可恢复的错误？

A 给 B 发请求，B 的 Completion 超时了，A 会 Completion Timeout 置位。



162. MRd 的 Completion Timeout 所用的计数器是软件维护还是硬件维护？

计数器由硬件实现，Timeout 的值软件可配。

三、数据链路层

基本介绍

163. PCIe 数据链路层是做什么的？有哪些功能？

PCIe 数据链路层位于事务层与物理层之间，用以保证数据传输的可靠性。

其主要功能如下：

TLP 数据交换。在发送端，数据链路层负责接收事务层发来的 TLP，添加 Sequence Number 及 LCRC 后交给物理层；在接收端，数据链路层负责接收物理层发来的 TLP，数据完整性检测通过后交给事务层。此外，数据链路层负责 TLP 流控。

错误检测及重传。数据链路层基于 ACK/NAK 机制来检测有无 TLP 损坏或丢包，在出现损坏或丢包时通过 Retry 机制请求重传。若发送端长时间等不到相应 TLP 的 ACK，还有 Timeout 机制来保证重传。

链路初始化及电源管理。PCIe 初始化过程中，数据链路层监测 PCIe Link 的 Reset、Active、Disconnected 等状态并传递到事务层。在 LTSSM 跳转到 L0 状态后，数据链路层交换 FC DLLP 进行流控初始化。此外，在允许过程中，PCIe Link 双方的数据链路层交换 PM DLLP 实现功耗管理及电源管理。

DLLP 包与包处理

164. DLLP 是什么？

DLLP，全称 Data Link Layer Packet，数据链路层包，用以 PCIe 链路管理及维护（链路初始化、电源管理等）。DLLP 仅在 PCIe 链路两端设备的数据链路层之间传输，其源于发送端设备的数据链路层，终于接收端设备的数据链路层。当 TLP 及 DLLP 均需要发送时，按照一定规则交替发送。

165. DLLP 分为哪几类？

根据功能的不同，DLLP 大致可分四类：

ACK/NAK DLLP（TLP 应答 DLLP）。包括 ACK、NAK，用以决定重传还是丢弃 Tx Retry Buffer 中的 TLP。

FC DLLP（Flow Control DLLP）。包括 InitFC1-P、InitFC1-NP、InitFC1-Cpl、InitFC2-P、InitFC2-NP、InitFC2-Cpl、UpdateFC-P、UpdateFC-NP 及 UpdateFC-Cpl，用以初始化及更新流控信用量（Flow Control Credit）。

PM DLLP (Power Management DLLP)。包括 PM_Enter_L1、PM_Enter_L23、PM_Active_State_Request_L1 及 PM_Request_Ack，用以进行功耗及电源管理。

Vendor Defined DLLP。厂商自定义的 DLLP。

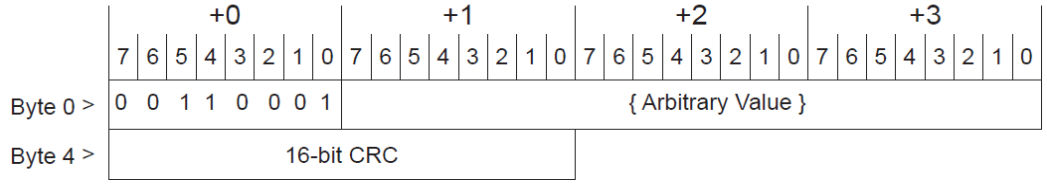
ACK/NAK、FC 及 PM DLLP 均为链路管理 DLLP。此外，还有 MR、NOP 及 Data_Link_Feature DLLP。各 DLLP 类型之间通过 DLLP 中的 Tyte 字段 (Byte0) 进行区分。

166. NOP DLLP 有什么功能？

Non-Flit Mode 下 NOP DLLP 仅用于链路测试，不能发，且收到后需忽略；Flit Mode 下，Flit 中 DLP 字段没有其他有效的待发送 DLLP 时填充 NOP DLLP。

167. NOP DLLP 是什么？有何作用？什么时候发 NOP DLLP? 收到后该如何处理？

NOP（No Operation）DLLP 是 PCIe 3.0 引入的概念，其帧格式下图所示，Payload 字段（Byte1~3）为任意值。NOP DLLP 仅用于链路测试场景，常规操作模式时不应发送 NOP DLLP，接收端接收到 NOP DLLP 后，在做完数据数据完整性检测后直接丢弃。

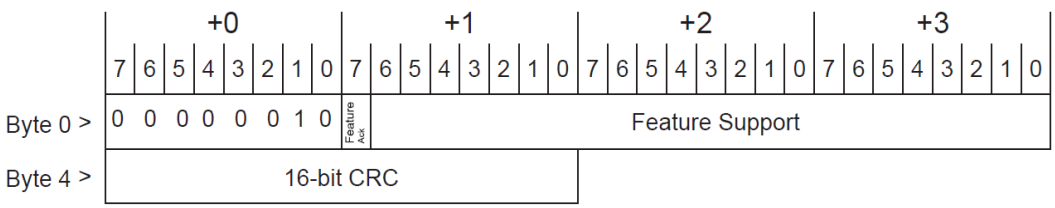


▲图： NOP DLLP Format

168. Data Link Feature DLLP 是什么？有何作用？什么时候发？

Data Link Feature DLLP 是一种携带有数据链路层 Feature 及相关 Ack 信息的 DLLP，数据链路层处于 DL_Feature 状态（可选）时，PCIe 链路两端的数据链路层通过 Data_Link_Feature DLLP 来告知彼此支持的数据链路层特性。

Data_Link_Feature DLLP 帧格式如下图所示，其中 Feature Support 字段来源于其 Data Link Feature Capabilities Register 的 Local Data Link Feature Supported 字段，目前只有 Local Scaled FC 这一个 Feature。当本端接收到对端发来的 Data_Link_Feature DLLP 后，回复以 Feature Ack 置一的 Data_Link_Feature DLLP。

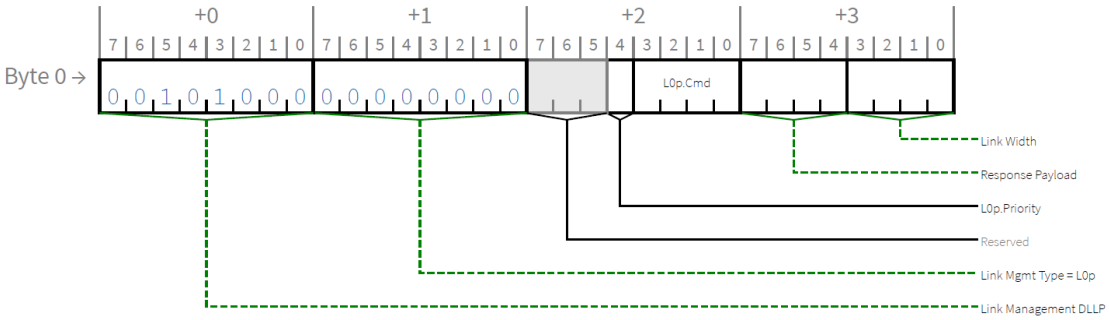


▲图： Data_Link_Feature DLLP Format

169. Link Management DLLP 是做什么的？

Link Management (LM) DLLP 首次出现于 PCIe 6.0 中，是一种用于链路管理的 DLLP。LM DLLP 包格式如下图所示，其中关键位域释义如下：

- ✓ DLLP Type，值为 28h，用以指示当前 DLLP 为 LM DLLP。
- ✓ Link Mgmt Type，用以指示链路管理的类型，目前 LM DLLP 仅用于 L0p 相关链路状态管理，Type=00h。
- ✓ L0p.Cmd，用以指示当前 L0p DLLP 是 Request 还是 Response、是何种 Response (Ack/Nak)。
- ✓ L0p.Priority，用以指示当前 L0p 请求的优先级，支持常规优先级及高优先级两种模式。
- ✓ Link Width，当 L0p.Cmd 为 L0p Request 时，该位域表示 L0p Request 请求的链路宽度。
- ✓ Response Payload，当 L0p.Cmd 为 L0p Request Ack/Nak 时，该位域表示对某链路宽度 L0p Request 的响应。



170. PCIe 6.0 规定的 DLLP Payload 有哪几种？

DLLP Payload Type 不同于 DLLP Type。DLLP Payload Type 分为**常规 DLLP Payload、Optimized_Update_FC 及 Flit_Marker**。常规 DLLP Payload 即 FC/PM/LM 等这些 DLLP，Optimized_Update_FC 及 Flit_Marker 属于非常规的 DLLP。DLLP Payload Type 及 DLLP Payload 最高位共同决定当前的 DLLP Payload 类型。

171. PCIe Sequence Number 是什么？有什么用处？

DL 层在发生 TLP 前会给每一个 TLP 添加一个序列号，即 Sequence Number。

接收端 DL 层在检查完 LCRC 后对 Sequence Number 进行检查，判断接收到的序列号是否与期望值一致，以此判断是否有丢包。出现丢包的话，依据 Sequence Number 发送 NAK 请求重传丢掉的包。短时间内收到 Sequence Number 相同的重复包不会被视为错误。

172. 几种不同的 DLLP 同时要发送时候，该怎么发？

若几种不同的 DLLP 同时，建议按照以下由高到低的优先级进行发送：

- ✓ 当前传输中的 DLLP 请求的 Completion
- ✓ NAK DLLP
- ✓ ACK DLLP
- ✓ FC DLLP
- ✓ Vendor Defined DLLP
- ✓ PM DLLP 及其他 DLLP

173. PCIe DLLP Blocking 是什么？

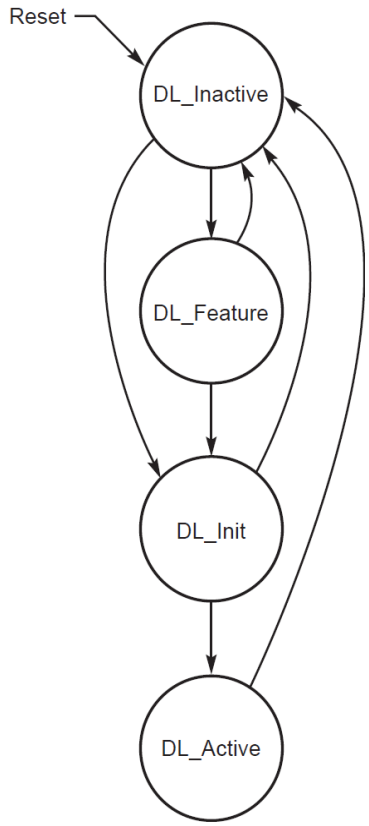
对于采用自动均衡机制的 PCIe 设备，USP 在收到 DSP 发来的 DLLP 之前不能发送任何 DLLP，DSP 在均衡完成之前也不能发送任何 DLLP，称为 DLLP Blocking。链路训练进入 L0 状态后，如果 PCIe 设备知道均衡未完成、其接下来要进行均衡，无论当前速率是多少（即便已经达到目标速率），双方都不能发送任何 DLLP。

若由于 DLLP Blocking 导致设备从 L0 延迟进入 Recovery，USP 或 DSP 端有可能会发生推测电气闲超时或 DLLP 超时，这两种超时均是不可报告的错误。当发生上述两种超时时，链路会进入 Recovery 重新对链路进行训练，不会影响正确的链接操作。

DL 状态机

174. 数据链路控制状态机是做什么的？

数据链路层状态机（Data Link Control and Management State Machine, DLCMSM），用以①控制及管理数据链路层工作状态，②在事务层、物理层之间沟通链路状态，③触发物理层进行链路状态管理等。DLCMSM 状态转移图如下图所示。



▲图： 数据链路层控制管理状态机

175. DLCMSM 跟 LTSSM 有什么关系？

DLCMSM 是数据链层状态机，LTSSM 是物理层链路训练状态机，两者各自为各自层次负责，同时又有一定的联系。

在物理层 LTSSM 进入 Target Speed L0 之前，DLCMSM 都处于 DL_Inactive 状态，物理层训练完毕（Target Speed L0，PL_LinkUp=1）之后，DLCMSM 进入 DL_Feature 交换 DL Feature 或直接进入 DL_Init 进行流控初始化。流控初始化完成之后才进入 DL_Active 状态，至此 DL_LinkUp=1。

176. DL_Feature 期间要交换哪些 Feature?

DL_Feature 是一个可选的状态，只有 ①PCIe 设备支持且开启了 Data Link Feature Exchange 且 ②物理层 PL_LinkUp=1 之后，DLCMSM 才会由 DL_Inactive 进入 DL_Feature 状态。

数据链路层处于 DL_Feature 状态（可选）时，PCIe 链路两端的数据链路层通过 Data_Link_Feature DLLP 来告知彼此支持的数据链路层特性。

Data_Link_Feature DLLP 的 Feature Support 字段来源于其 Data Link Feature Capabilities Register 的 Local Data Link Feature Supported 字段，目前只有 Local Scaled FC 这一个

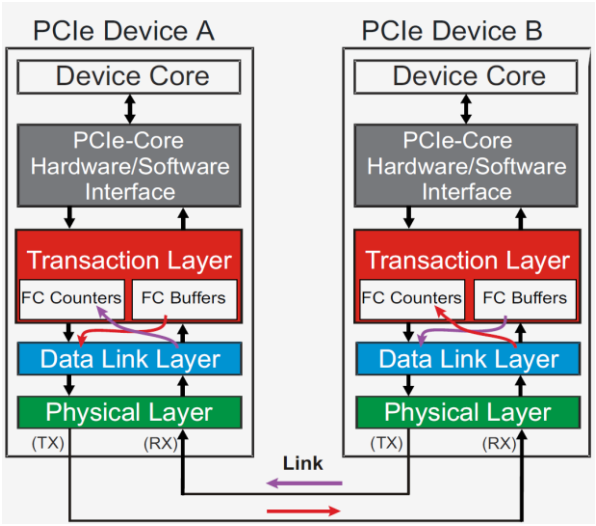
Feature。当本端接收到对端发来的 Data_Link_Feature DLLP 后，回复以 Feature Ack 置一的 Data_Link_Feature DLLP。

流量控制

177. PCIe Flow Control 是什么？

PCIe Flow Control (FC) 是一种基于信用量 (Credit) 的点到点的 TLP 流控机制（非端到端），确保接收端有足够缓存的前提下才会发送 TLP。

PCIe 流控逻辑位于事务层，通过 FC DLLP 来进行流控初始化及 Credit 更新。PCIe 多个虚通道（Virtual Channal, VC）之间独立进行流控，同一 VC 内，也按照根据事务类型（P/NP/CPL）及 Header/Data 分为多种 Credit 独立流控。



▲图： PCIe Flow Control 基本逻辑框图

（图源 Mindshare PCIe Technology）

178. PCIe Flow Control 用到了哪些 DLLP?

PCIe 基于 FC DLLP 来实现流控初始化及流控 Credit 的更新。流控初始化用到的 FC DLLP 包括 InitFC1-P、InitFC1-NP、InitFC1-Cpl、InitFC2-P、InitFC2-NP、InitFC2-Cpl，Credit 更新用到的 DLLP 包括 UpdateFC-P、UpdateFC-NP 及 UpdateFC-Cpl。

179. Flow Control 初始化流程是怎样的？

PCIe 流控初始化发生于物理层 LinkUp=1 之后、数据链路层 DLCMSM=DL_Init 状态时。

PCIe 流控初始化先后分为两个阶段：

FC_Init1 阶段，PCIe 设备先后重复发送 InitFC1-P、InitFC1-NP、InitFC1-Cpl 这 3 笔 FC DLLP 到对端设备，至少每 34 us 发送一次，告知其 Rx Buffer 的 Size。直到接收到对端发来的 InitFC1 DLLP 之后才停止发送，并把对端的 Buffer Size 保存起来。

FC_Init2 阶段，PCIe 设备先后重复发送 InitFC2-P、InitFC2-NP、InitFC2-Cpl。此时的 FC DLLP 中不携带初始化 Credit 信息。

180. PCIe 流控初始信用量可配吗？

流控初始信用量跟 Buffer 深度有关，一般是静态配置，有些 IP（比如 PLX）也留了寄存器允许在最大支持的范围内调整初始信用量的值。

181. InitFC1 和 InitFC2 有何区别？

InitFC2 期间发送的 InitFC2 DLLP 不携带 Credit 信息，仅用以确认流控初始化是否成功。InitFC1 期间不能发送 TLP，InitFC2 期间可以。

182. PCIe Spec 有规定 InitFC DLLP 发送的频率吗？

只看到不超过 34 us 的限制，34 us 内需要发一轮 InitFC。

183. 流控 Scale Factor 的值是在固件配置的吗？

Table 3-3 Scaled Flow Control Scaling Factors

Scale Factor	Scaled Flow Control Supported and Activated	Credit Type	Min Credits	Max Credits	Field Width	FC DLLP field	
						Transmitted	Received
00b	No	Hdr	1	127	8 bits	HdrFC	HdrFC
		Data	1	2,047	12 bits	DataFC	DataFC
01b	Yes	Hdr	1	127	8 bits	HdrFC	HdrFC
		Data	1	2,047	12 bits	DataFC	DataFC
10b	Yes	Hdr	4	508	10 bits	HdrFC >> 2	HdrFC << 2
		Data	4	8,188	14 bits	DataFC >> 2	DataFC << 2
11b	Yes	Hdr	16	2,032	12 bits	HdrFC >> 4	HdrFC << 4
		Data	16	32,752	16 bits	DataFC >> 4	DataFC << 4

是的，是在固件中配置的。IP 有一套默认配置参数，但可以通过边带接口去修改，一般给固件用。

184. 为什么 TLP Header 和 Body 要分开做流控？这样要分开方到两个 Buffer 里，岂不是增加了复杂性？其优势在哪？

- ✓ 快。无需重新解析哪是 Header 那是 Data。
- ✓ 方便传输。有些包没 Data 只有 Header。

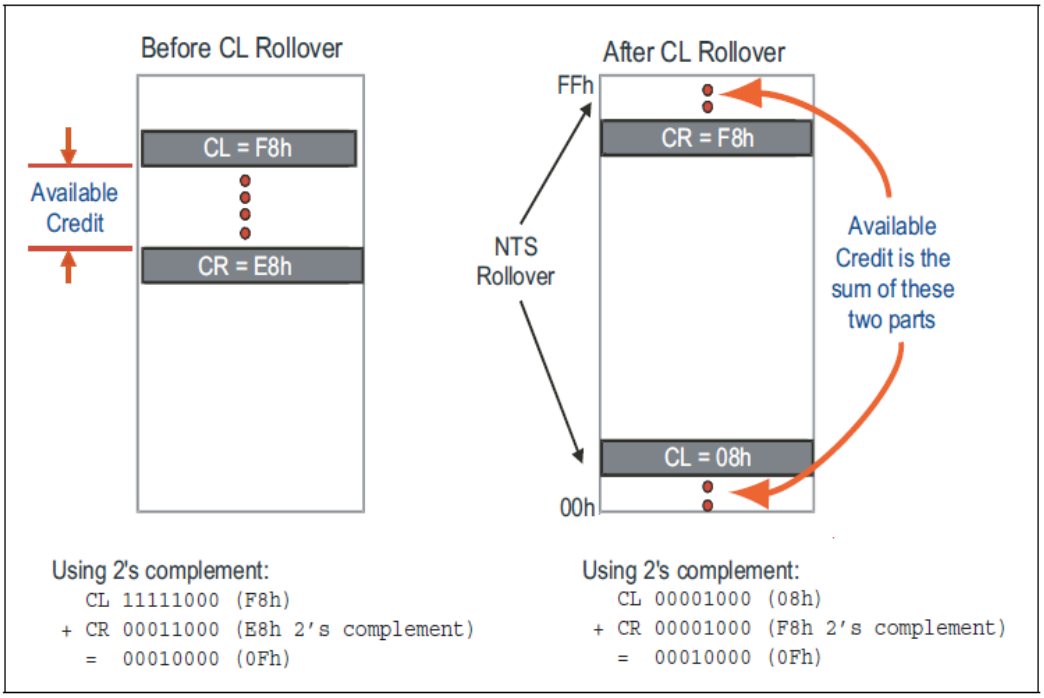
下述为引用：[PCIe 流量控制-Peter Blog (peterhu.github.io)]

PCIe 总线将 Header 和 Data 缓存分离的主要原因是，通常 TLP 的 Header 的大小是固定的但是 Data 的大小并不固定，Data 的长度通常由 TLP 的 Length 字段确定。将 Header 和 Data 缓存分离有利于合理利用 Data 缓存，

因为 TLP 的传输需要知道接收端的可用的 Credit Buffer，所以要先对接收端和发送端的 Flow Control 进行初始化也就是初始化它们的 Credit Buffer。Flow Control 的初始化是在链路训练完成之后，当物理层的 Link Up 信号触发之后就会开始，这个过程也被称为 DLCMSM(Data Link Control and Management State Machine).

185. Mindshare PCIe Technology 流控指针计算时 E8h 取补码为何是 11000b 而非 10011000b？

Figure 6-13: Flow Control Rollover Problem



正常是这样，但 PCIe Spec 上说了无符号数，没负数。

186. PCIe 流控有 N123 算法 N123+算法和 N23 算法，不同节点如果选择不同的效率可能会大受影响，这种情况是否需要添加额外的逻辑来加以协调呢？还是接受浪费？

N123、N23 算法应该是 ATM 网络中 Link-by-Link 的流控， PCIe 流控是单条 Link 两端 DL 层之间的流控，不是同一个概念。如需更多上述 ATM 流控的介绍，可参考以下论文：

H. T. Kung and A. Chapman, “The FCVC (Flow Controlled Virtual Channels) Proposal for ATM Networks,” Version 1.3, October 1993. A summary of the paper is to appear in Proc.1993 International Conf. on Network Protocols, San Francisco, California, October 19-22, 1993, pp. 116-127.

187. 流控机制要求剩余缓冲数量小于最大缓冲数的一半才能发包，为什么要求一半？

不是最大数目的一半。仔细看你会发现 Field Size 有的是 8，但是允许的最大 Credit 数量是 127。个中缘由需要细细品味。

188. FC Credit Timeout 后会发生什么？PHY 重新回到 Detect 重新训练，还是 DL 层 Link Down 但 LTSSM 保持在 L0？

协议中只提到了在 Flit Mode 时必须实现 FCP Timeout 机制，FCP 仅对非 Infinite Credit 类型有效。当出现 FCP Timeout 时，链路需要回退到 Recovery 状态。对于非 Flit Mode，未看到有相关要求。个人理解，如果没有实现 FCP Timeout 机制，且没有按规则反馈 Credit 导致流控出问题，会报出相关 Buffer 溢出的错误，进一步处理该错误即可。如果偶然 Buffer 溢出，重传即可；频繁溢出可能就是链路问题，需要重新训练。

189. 启用了 Scaled Flow Control 后，是不是 Update DLLP 传输的实际 Credits 数量必须是 Factor 的倍数？

是。比如 Scale Factor 为 10b，那我们在 DLLP HdrFC Field 放个 1，那么实际传输的 Credits 数量其实是 4。

190. 完成状态为 UR 或 CA 的 Cpl 包会占用一个 Cpl Header 信用量吗？

会。分层考虑。

191. EP 可以发出超过其 Cpl 信用量的 MRd 请求吗？

补充描述：比如说 EP 发给 RC 的 Cpl 信用量只有 10，但是 EP 想要发起需要 100 信用量的 MRd 请求，理论上这笔请求的 Cpl 回来后需要小花 100 个 Cpl 信用量。这笔请求可以发吗？对端收到后可以回吗？

个人理解，如果允许发送超出 Cpl 可用信用量的 MRd，对端在 RCB 等拆包规则内能够拆分为发端 Cpl 信用量允许范围内的小包，那么可以先行反馈部分 CplD，等 Cpl Credit

更新后在反馈后续 CplD；如果无法拆分为小包，那么可能会造成死锁。保险起见，一般情况下 Cpl 的默认 Credit 为 0，即无穷大。

问题追踪：实测过程中，Cpl 的 InitFC 直接给 10，不更新 Cpl 的 UpdateFC 直接发需要消耗 100 Data Credit 的 MRd，对端也回了 CplD 且未拆分为小包。如果上述测试过程描述无误，这种情况是违反协议的，需要进一步排查。

192. 发送给对端的 IO 事务，如果对端不接受并回了 UR，会消耗信用量吗？

会消耗信用量，但回了 UR 之后就释放了。

193. 什么是 Scaled Flow Control? 跟 Flow Control 区别是什么？

常规情况下，FC DLLP 中 Header Credit 位宽为 8 bit，Header Credit 最大为 127；Data Credit 位宽 12 bit，最大为 2047。对于一些 Rx Buffer 较大的设备，这些最大 Credit 不够用，如果不加大，势必会影响传输性能。

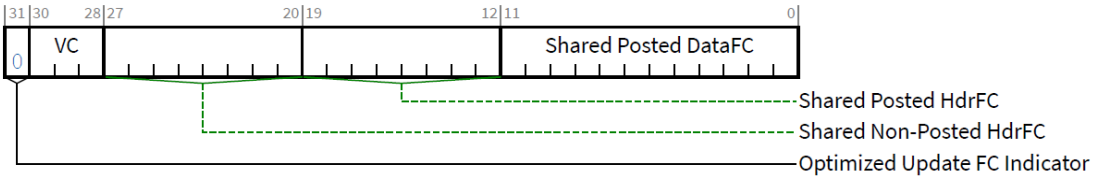
从 PCIe 4.0 开始，PCIe 支持 Scaled Flow Control，通过 FC DLLP 中的缩放因子 HdrScale 及 DataScale，在 DLLP 中相关 Credit 位宽不变的情况下，来代表更多的 Credit 数目，从而支持更多数量的 Credit 通报。这种开启了 Scaled Factor 的流控即 Scaled Flow Control。

194. Optimized_Update_FC 是做什么的？跟常规 UpdateFC 有何区别？

Optimized_Update_FC 是流控包的一种，仅用于 Flie Mode。

Optimized_Update_FC 格式如下如所示，其中包含 Shared NPR HdrFC、Shared PR HdrFC 及 Shared PR DataFC，没有 HdrScale 及 DataScale。Completion HdrFC 及 DataFC 在 Merged FC 机制中合入了 PR。

Optimized_Update_FC 无需保存进 Retry Buffer。



数据完整性

195. DLLP 自身是如何保证其数据完整性的？

首先，若能够接收到 DLLP，每一笔 DLLP 都携带有 2 B CRC，来保证接收 DLLP 的正确性。

其次，若 DLLP 丢包，对于 Ack/Nak DLLP，要么没有影响（短暂丢 Ack），要么有 Timeout 机制来保证 TLP 重传（丢 Nak，长时间丢 Ack）。对于 PM DLLP，会重复多次或者持续发送相关 DLLP，确保对端能够收到。对于 FC DLLP，发送端在收到对端发来的 InitFC 前会一直定时发送 InitFC 给对端；对于 UpdateFC，其是定时发送的，偶尔丢一个影响不大，采用最新收到的 Credit 就好了。

196. LCRC 和 ECRC 有何区别？

ECRC，End-to-End CRC，由事务层生成及解析，是**可选的**，是 TLP 的组成部分，Size 4 B，位于 Data Payload 之后。ECRC 始于 Requester，终于 Completer，不会被 Switch 等中间节点修改。ECRC 错误是不可纠正错误，若 Switch 发现 ECRC 出错，其记录错误并继续转发；若 Completer 发现 ECRC 出错，则上报错误请求系统处理。

LCRC，Link CRC，由数据链路层生成及解析，是必选的，Size 4 B，添加在 TLP 之后。LCRC 始于，终于 PCIe 链路的发送端数据链路层，终于 PCIe 链路的接收端数据链路层，途径 Switch 时需要进行校验并重新进行计算填充。LCRC 是可纠正错误，若接收端检测到 LCRC 出错，其请求重发该 TLP。

197. PCIe Gen6 中提到的 FBER 是指什么误码率？

好像是不考虑 Burst Error 情况下的误码率。

198. Cfg 事务和 IO 事务是不是也可以开启 ECRC？

可以。ECRC 跟 TLP 事务类型没关系，TLP 中 TD=1 就表示有 ECRC。

重传机制

199. 发生 DLLP Timeout 会导致链路 Retrain 吗？

会先 Replay 重传，四次重传仍还收不到 Ack 则进行 Retrain。

200. PCIe EP 发送注入 Parity Error 的 TLP 给 Host 后理应会触发重传，如何在 EP 端确认是否发生了重传？

注：采用 Synopsys IP。

在 EP DUT 内 diag bus 上可以看到 Nak。如果对端挂的是个 VIP，VIP Port 上也能检测到，同时仿真 Log 里有记录。

201. Requester 发送 Req TLP 之前需要确认 Completer 有 Credit，Completer 给 Requester 反馈 Credit 时需要 Requester ID，这不是死锁了吗？

问题所述不正确。流控包为 DLLP，Completer 的 Credit 并非夹带于 Cpl TLP 中，而是有专门的 Flow Control DLLP。在 DL Link Up 之前 Requester 跟 Completer 之间就已经完成初始 Credit 数目的协商，不需要对端通过 TLP 触发，而是定时反馈。

202. PCIe 6.0 Retry 机制，越看越迷糊怎么办？

看不懂可能是 Spec 写错了。PCIe Base Spec 6.0.0 Gen6 Retry 部分有不少错误，在 6.0.1、6.1 及之后的版本中都有 Retry 相关 Errata Fix，建议看最新版。

203. IMPLICIT_RX_FLIT_SEQ_NUM 该怎么理解？

Flit Sequence Number 要么是 Tx Flit 的 Explicit Flit Sequence number，要么是 Ack/Nak Sequence Number。Rx 自己维护一个 Counter，就是这个 Implicit Rx SN Counter，当接收到 Ack/Nak SN 时，根据这个 Counter 来推断当前 Flit 的 Explicit SN。

204. Rx 端为什么自己要有一个 Retry Buffer？

Selected Retry 机制需要 Rx 自己有个 Buffer 来暂存重传前正常发送的 Flit。Selected Retry 机制在于 Rx 即使收到一个错误的 Flit，后面好的 Flit 也要存入 Rx Buffer，等 Retry 的特定 Flit 来了后放到缺失的位置，一次就把 Rx buffer 的 Flit 都发给协议层。

205. Nak/Ack Flit 里面的应该 TLP 填充什么？

问题补充：比如收到这个 Nak Flit，然后知道是要 Replay，里面的 TLP 要关注吗？

Ack/Nak 是 Flit 中 DLP 字段，TLP 字段不受影响该怎么发仍然怎么发，没有流量时发生 NOP TLP。

206. 每收到一个 MRd 都要回复一个 Ack DLLP 吗？

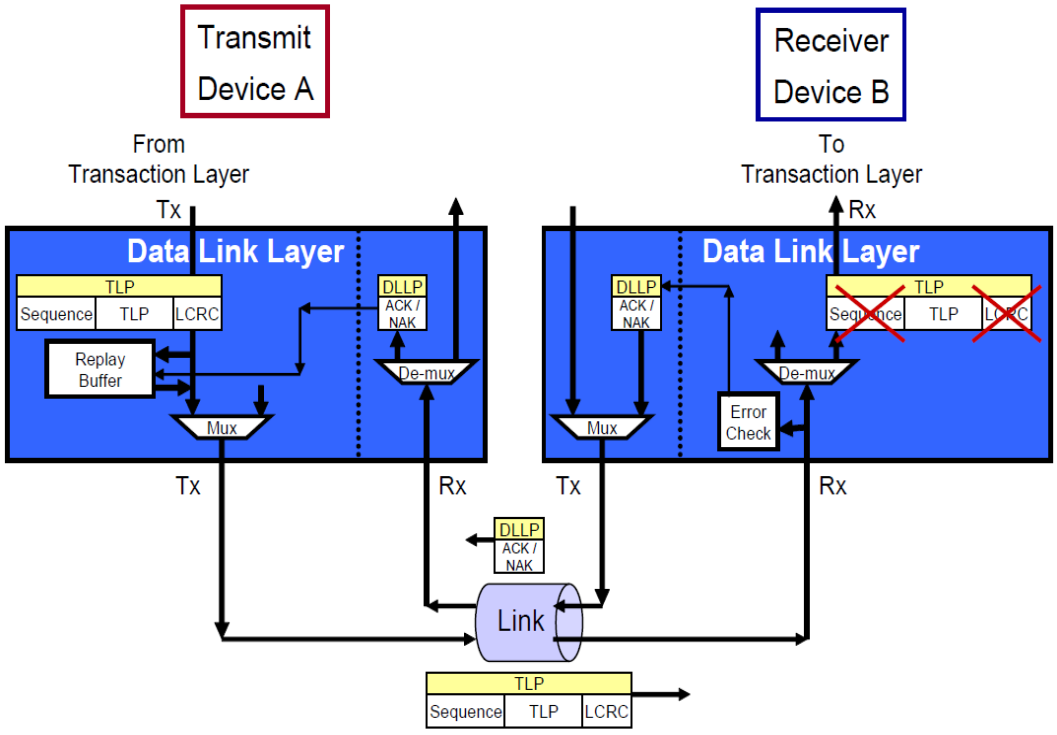
不是，Ack/Nak 机制跟 TLP 类型无关。发送端发送一直 TLP，接收端每间隔一定 Latency 时间累计反馈一次 Ack/Nak 进行确认，不需要每个 TLP 都反馈。Ack/Nak 到达发送端后，发送端清理 Replay Buffer 中的相关 TLP 或进行重传。这个 Latency 在是 PCIe Base Spec 有公式定义，跟 MPS、Bandwidth 等有关。

207. ACK/NAK 机制是什么？

PCIe 通过 ACK/NAK 机制来保证 TLP 在 PCIe 链路上的 End-to-End 可靠传输。

PCIe ACK/NAK 机制示意图如下图所示，通过几个关键要素来简单介绍 ACK/NAK 机制：

- ✓ Sequence Number，LCRC。发送端数据链路层在每一笔要发送的 TLP 前添加连续编号的 Sequence Number（SN）及 LCRC，然后发送到对端。
- ✓ ACK/NAK DLLP。接收端数据链路层接收到 TLP 后，依据 SN 及 LCRC 来判断 TLP 是否被正确传输、有无丢包/坏包，并给发送端反馈 ACK 或 NAK。
- ✓ Tx Retry Buffer。发送端发出的每一笔 TLP 均在 Tx Retry Buffer 中进行备份。若后续接收到了 ACK，则清理 Tx Retry Buffer 中的相关 TLP；若收到了某 TLP 的 NAK 或超时未收到 ACK，则从 Tx Retry Buffer 中取出相关 TLP 进行重传。



▲图： PCIe ACK/NAK 机制示意图

208. 每个 TLP 都要回复 Ack 吗？Ack 要多久发一次？

并非每一笔 Good TLP 都需要回复一笔 Ack DLLP，这样效率太低了。

PCIe Rx 数据链路层内部有个 AckNak_Latency_Timer，从收到某笔没有回复 Ack 的 TLP 就开始计时，达到指定时间后（取决于 Timer 设计）才会要一次 Ack。此前有可能会收到多比 Good TLP，这些 TLP 只回复一次 Ack，Ack 的 Sequence Number (SN) 为最新 Good TLP 的 SN。Tx 端收到 Ack 后，会把 Tx Retry Buffer 中 SN 之前的 TLP 清理掉。

209. 什么情况下会发送 Nak DLLP?

简单而言，Rx 发现 Bad TLP 或 TLP 丢包时会发送 Nak。通过检测 LCRC 来检测 TLP 是否为 Bad TLP，通过 SN 还判断是否有 TLP 丢包。

210. TLP Tx 收到 Nak 后如何处理？

根据 Nak DLLP 中的 SN，从 Tx Retry Buffer 中取出相应 TLP 进行重传，重传次数计数器 REPLAY_NUM 加一，重传计时器 REPLAY_TIMER 启动。

211. 连续多次重传失败后如何处理？

若 TLP 多次重传均以失败告终，重传计数器 REPLAY_NUM 会溢出时，此时考虑是物理链路出现了问题，应记录错误事件并通知物理层重新对物理链路进行训练。此时数据链路层仍然是 Active 的，其 Tx Retry Buffer 不清空，相关计数器及寄存器值也不复位，等待物理链路训练完成后接着进行重传。

212. Retry 机制中用到的 Tx Retry Buffer Size 是如何计算的？

跟绝大多数 Buffer Size 的设计原则相同，在设计 Tx Retry Buffer Size 时应遵循两大原则：

- ✓ Size 足够大，确保常规操作时收到 Ack 并清掉其对应 TLP 前 Buffer 不溢出。
- ✓ 在满足以上需求的前提下，Size 尽量小，以降低成本。

第一条“收到 Ack 并清掉其对应 TLP 前 Buffer 不溢出”，这里有两个关键因素：①收到 Ack 前，②不溢出。也就是说，从某 TLP 发出并存到 Tx Retry Buffer，到收到其 Ack 后从 Tx Retry Buffer 清掉该 TLP 为后来者腾空间，这段时间内发出的所有 TLP 都能够存放在 Buffer 中且 Buffer 不溢出。需要考虑以下时延：

- ✓ Rx 收到 TLP 到开始回复 Ack 的时延。
- ✓ Ack 的处理时延。
- ✓ TLP、DLLP 在物理层及物理链路上的时延。
- ✓ Tx Retry Buffer 所在 Device 的 Rx 从 L0s 退出到 L0 的时延。

此外，对于 SRIS 时钟的 Device 或者 PCIe Retimer，也会引入额外时延。

213. PCIe Flit Mode 时，Selective Nak 是什么？

Selective Nak 是和 Standard Nak 相对的一个概念，初现于 PCIe 6.0。PCIe Flit Mode Ack/Nak 机制中，并非每一笔 Flit 都需要响应以 Ack/Nak。当收到 Nak 之后，根据 Nak 类型来选择需要重传的 Flit。若为 Standard Nak，则 Nak Flit Sequence Number 之前未经

Ack 确认的 Flit 均需要重传；若为 Selective Nak，则仅需要重传 Nak 所对应的 Flit。Selective Nak 实现的前提是接收端具备 Rx Retry Buffer。

214. PCIe 6.0 Retry 机制中，Implicit Flit Sequence Number 是做什么的？

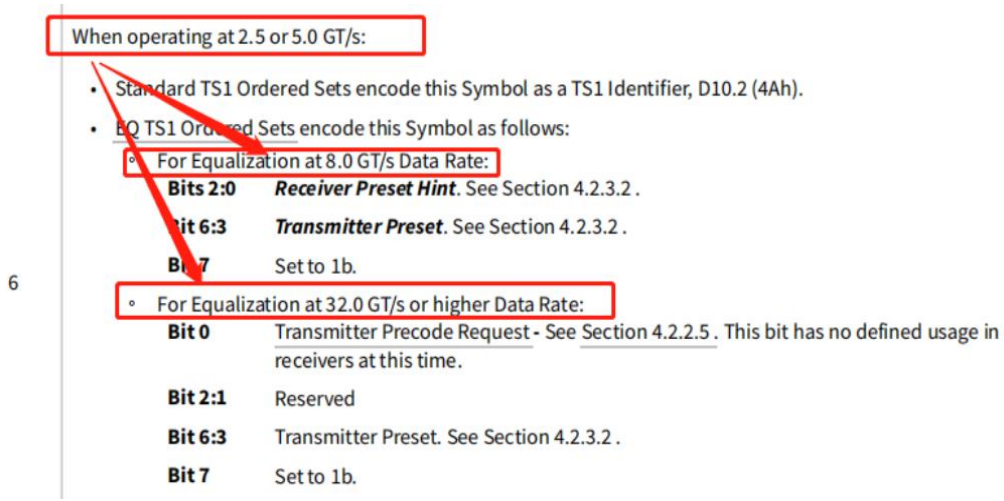
Implicit Flit Sequence Number，或者说 IMPLICIT_RX_FLIT_SEQ_NUM，是 PCIe Flit Mode 流控机制中接收端用来计数/编号接收到的 Flit 的。对于接收到的 Flit，并非所有 Flit 都携带了 Explicit Flit Sequence Number，也并非所有携带 Explicit Flit Sequence Number 的 Flit 都需要计数。Rx 端按照一定规则（参考文档 IMPLICIT_RX_FLIT_SEQ_NUM Rules）赋予接收到的 Flit 一个编号（即 IMPLICIT_RX_FLIT_SEQ_NUM），保证流控机制的正常运转。

四、逻辑物理层

包与包处理

TS

215. TS1 Symbol 6 中，为什么在 2.5 GT/s 及 5 GT/s 的条件下包含 8 GT/s 和 32 GT/s 的描述？



切到 Gen3 以上速率之前，先在 Gen1/2 低速发 Preset，看下均衡那一节的 Phase 0。

216. TS2 Symbol 6 2.5 or 5.0 GT/s 条件下又包含 8.0/32.0 GT/s 的条件，这该如何理解？

When operating at 2.5 or 5.0 GT/s 是指进 Recovery 时的速率，子条件 For equalization at 8.0/32.0 GT/s 是指目标速率。当前处于 2.5 或 5.0 GT/s 时，目标速率为 8.0 GT/s 为正常均衡，目标速率为 32.0 GT/s 是 bypass_eq_to_highest_rate=1 的情况。如果目标速率为 16.0 GT/s，则先均衡到 8.0 GT/s，然后 8.0 GT/s 下走 When operating at 8.0 GT/s or higher 的分支切速到 16.0 GT/s。

217. Flit 有类似 STP 这样的 Token 吗？接收端如何识别 Flit 起始位置？

没有 STP Token，有 EIOS EIEOS SKP，Rx 通过接收到的 Data Bytes 来推断当前为 OS 还是 Flit Data。可参考 PCIe 6.0 Spec 4.2.3.4.5，Ordered sets insertion in data stream in flit mode，介绍了 Flit 模式下如何插入有序集。

218. PCIe TS1/TS2 是如何实现 DC Balance 的？

对于 Gen1/Gen2 ， 8b/10b 编码自带 DC Balance 功能。

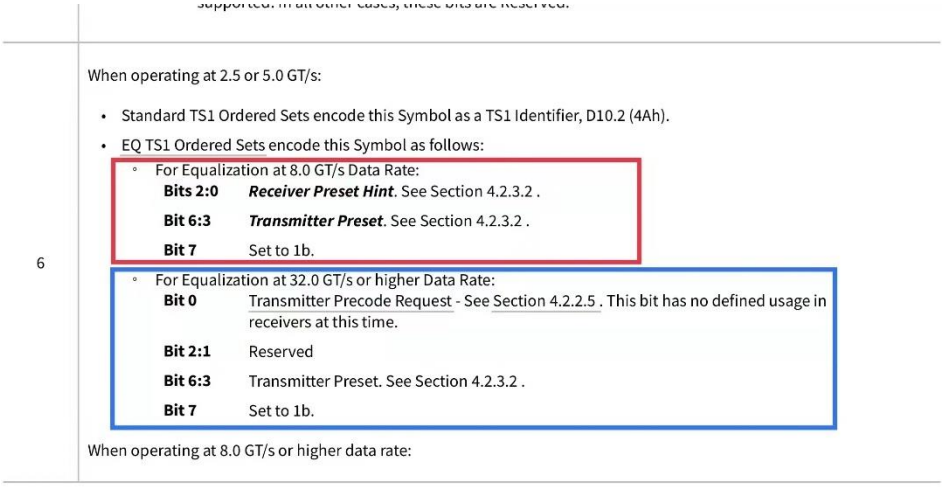
对于 Gen3 及以上，128b/130b 只在 Payload 前添加 2 bit 同步头，不具备 8b/10b 那样的 DC Balance 能力，依靠加扰来实现一定程度的 DC Balance。TS1 或 TS2 的 16 个 Symbol 中，Symbol 0 无需加扰，Symbol 1~Symbol 13 需要加扰，Symbol 14~Symbol 15 根据 DC Balance 的需要可加扰可不加扰。

为了采用 Symbol 14、Symbol 15 实现 DC Balance，每条 Tx Lane 都有一个独立的模块监测其已发送的 bit1 和 bit0 的数量差。当差值达到 511 之后不再继续加，但是可以减少。具体规则如下：

- ✓ 如果发完 Symbol 11 后差值 >31，Symbol 14、Symbol 15 均不加扰，均用作 DC Balance。若 1 比 0 多，则多发点 0，Symbol 14 发送 20h（00100000b），Symbol 15 发送 08h（00001000b）；若 0 比 1 多，则多发点 1，Symbol 14 发送 DFh（11011111b），Symbol 15 发送 F7h（11110111b）。
- ✓ 如果发完 Symbol 11 后差值 >15，Symbol 14 加扰照常发送 TS 标识符，Symbol 15 不加扰用作 DC Balance。若 1 比 0 多，则多发点 0，Symbol 15 发送 08h（00001000b）；若 0 比 1 多，则多发点 1，Symbol 15 发送 F7h（11110111b）。
- ✓ 如果发完 Symbol 11 后差值 <=15 时，Symbol 14、Symbol 15 均加扰，照常发送 TS 标识符。
- ✓ 当监测到 EI 或 EIEOS 后，计数器清零。

Rx 可以检测 Symbol 14、Symbol 15 的合法性，也可以不检测。

219. 请教一下 PCIe Gen5 协议物理层逻辑子层 TS1 Ordered Set 的使用问题：TS1 里面的 Symbol 6，用在 2.5 GT/s 和 5 GT/s 速率的时红框和蓝框的两个配置分支的用途该如何解释？



红框是指从 2.5 GT/s 或 5 GT/s 进 Recovery.Speed 切换速率进入 8 GT/s 之前的其他状态下使用；蓝框是指从 2.5 GT/s 或 5 GT/s 进 Recovery.Speed 跳过中间速率直接切换速率进入 32 GT/s 之前的其他状态下使用。

220. Modified TS1/TS2 是什么？Modify 了什么？为什么要 Modify？

非 PCIe 协议运行在 PCIe PHY 上或发送 TS Message 时，在 LTSSM 部分子状态采用 Modified TS1/TS2。

允许发送 Modified TS1/TS2 的 LTSSM 子状态有 Cfg.Lanenum.Wait、Cfg.Lanenum.Accept、Cfg.Complete。在此之前的，必须在 Polling.Active、Polling.Cfg、Cfg.Linkwidth.Start、Cfg.Linkwidth.Accept 状态将 Standard TS1/TS2 Symbol 5 的 bit[7:6] 置为 2'b11，以此在收发端协商是否支持 Modified TS。只有在收发端均支持 Modefied TS 时，才能发送 Modified TS，否则仍然发送 Standard TS。

跟 Standard TS1 / TS2 比，Modified TS 更改了 Symbol 6~15，这些 Symbol 主要是均衡相关字段，采用 Modified TS 是为了在协商完毕链路宽度后，同时协商 Lane Number 及 Alternate Protocol，并 Bypass EQ。

Order Set

221. PCIe 有序集有哪几种？

PCIe 物理层包（Physical Layer Package, PLP）又称有序集（Ordered-Set），长度为 4 B 整倍数，用于链路训练、时钟偏差补偿、使链路进入电气闲或退出电气闲。PCIe 有序集有以下几种：

- ✓ TS0 / TS1 / TS2，用于链路初始化、训练、协商等,TS0 出现于 PCIe 6.0。
- ✓ SKPOS，用于收发端时钟补偿。
- ✓ EIOS，使链路进入电气闲。
- ✓ EIEOS，使链路退出电气闲。
- ✓ FTSOS，使链路退出低功耗模式。
- ✓ SDS/EDS，标志数据流开始/结束，仅用于 Gen3 及以上。

222. 128b/130b 时 SKPOS 长度不定，发送端是如何补齐 1 个 OS Block 的？

128b/130b 时发送端 SKPOS 内固定由 16 个 SKP Symbol 组成，在 Retimer 内可能会更改 SKP Symbol 数量，最终接收端 SKPOS 内的 SKP Symbol 可能有 8、12、16、20、24 个 SKP。

223. 128b/130b 时 SKPOS 长度不定，接收侧如何确定 SKPOS 的结束？

最后一组（4 个）SKP 内有 SKP_END/SKP_END_CTRL 指示 SKPOS 是否结束。

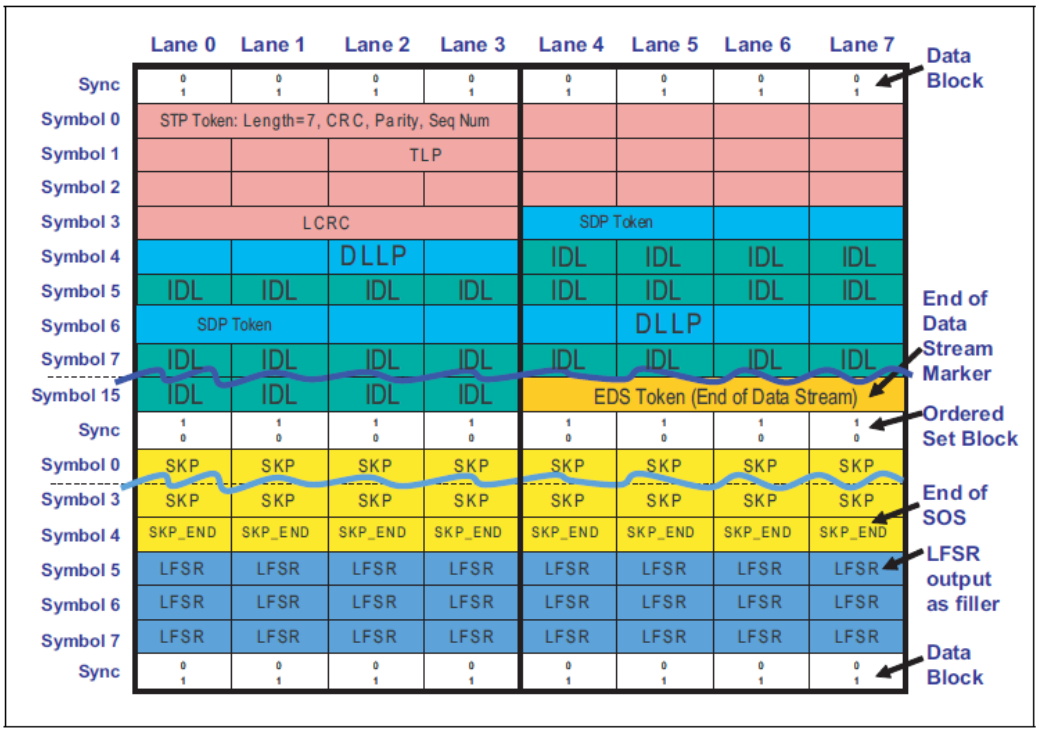
224. PCIe 有序集什么时候加扰，什么时候不加扰？

Gen1/Gen2 时，K 码不加扰，除 OS 外的 D 码、Compliance Pattern、Modified Compliance Pattern 都需要加扰。为了便于调试，也可以通过 TS sym5 bit3 软件控制关闭加扰功能。

Gen3 及以上时，对于 TS1 及 TS2，Symbol 0 无需加解扰，Symbol 1~Symbol 13 需要加解扰，Symbol 14~Symbol 15 根据需要可加解扰可不加解扰，为了 DC Balance 可以不加解扰。FTSOS、SDS、EIEOS、EIOS、SKPOS 不加解扰。除 SKPOS 外，其他 OS 即便不加解扰，LFSR 也照常推进。

225. 为什么下图中的 SKPOS 未遵守 128/130 编码？

Figure 12-15: Gen3 x8 Skip Ordered Set (SOS) Example



SKPOS 比较特殊，有 SKPOS 的 OS Block Symbol Length 跟 SKPOS 中的 SKP Symbol 数量有关，规则如下：

The one exception is the SOS (Skip Ordered Set), because it can be changed by intermediate receivers in increments of 4 Bytes at a time for clock compensation. Consequently, an SOS is legally allowed to be 8, 12, 16, 20, or 24 Symbols in length.

226. PCIe SDS/EDS 是什么？

SDS，全称 Start of Data Stream Ordered Set，在发送 Data Block 第 1 DW 前发送。与之对应的是 EDS，发送 Data Block 最后 1 DW 后发送 EDS，标志数据块的结束，下一笔就是 OS 有序集了。

发送 SDS 对链路状态有所要求，在 LTSSM 跳转到 L0 状态前的最后一个子状态发送，包括 Cfg.Idle、Rcvr.Idle、Tx_L0s.FTS，且只能在这些状态时发送。

SDS/EDS 仅适用于 Gen3 及以上速率，不适用于 Gen1/Gen2 是因为 Gen1/Gen2 时还没出现 SDS/EDS，协议要兼容老版本。Gen1/Gen2 时，LTSSM 跳到 L0 前不发 SDS 发什么？——在所有 Lane 上同时发 Logic Idle Symbol，即 0x00。

Loopback 时，OS Block -> Data Block 切换时 Tx 无需发送 SDS 或 EDS，Rx 也无需检查 SDS 或 EDS。

227. EIOS 截断是什么？

128b/130b 编码时，EIOS 由 16 个 66h 组成。若当前发送的 EIOS 之后紧接着还有 EIOS，那么必须发送完完整的 16 个 66h 当前 EIOS 才算发送完毕。若发送完当前 EIOS 接下来便进入电气闲，那么发送完 14 个 66h 后就可以停止发送剩下的 66h，仍认为当前 EIOS 发送完毕，称为 EIOS 截断。

EIOS 截断是为了解决 128b/130b 编码期间发送端可能存在的符号边界上的时钟边界不对齐情况。EIOS 截断不会影响接收端对 EIOS 的判断，因为接收端只需要收到 6 个 EIOS 就可以做出接收到 EIOS 的判断了。

228. Logic Idle Symbol 与 Idle Symbol (IDL) 是一样的吗？与 EIOS 有何关系？

不一样。

Logic Idle Symbol，逻辑闲符号数据是 0x00，与 EIOS 没什么特定关系，当链路上临时没有数据包要发送时候（链路逻辑闲），发送器继续发送逻辑闲符号。

Idle Symbol （IDL），数据是 0x7c，是 EIOS @Gen1/Gen2 中的一个符号。Gen1/Gen2 时，一个 EIOS 由 1 COM + 3 IDL 组成，Gen3 及以上时 EIOS 由 16 个 0x66 组成。

Tx Logic Idle 时，所有 lane 上都发 Logic Idle Symbol。

（参考：【PCIe 有点闲】电气闲，逻辑闲，到底谁更闲？）

229. 为什么在 2.5、8、16、32 GT/s 时 EIOSQ 由 1 个 EIOS 组成，而 5 GT/s 时却需要 2 个?

这个 Spec 上没讲明，也没有找到明确的解答。以下是个人猜测的原因：

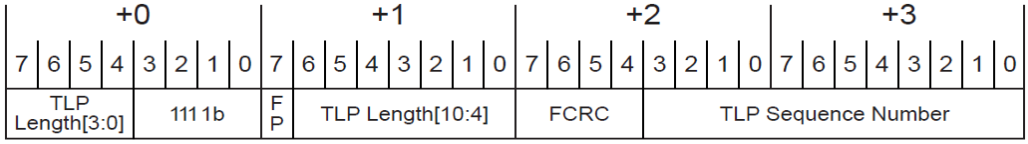
2.5 GT/s 时，速率相对较低，信号质量好，EIOSQ 由 1 个 EIOS 组成。速率从 2.5 GT/s 上升为 5 GT/s 后，并没有经过均衡，信号质量不如 2.5 GT/s 时好。为了确保 Rx 能够正确进入电气闲，EIOSQ 由 2 个 EIOS 组成，Rx 只需接收到 1 笔 EIOS 即可进入电气闲。

8 GT/s，16 GT/s，32 GT/s 时，EIOSQ 仍然只需 1 个 EIOS，是因为进入这些速率是经过了均衡的，信号质量有保证。

230. STP 中的 FCRC 和 FP 都是干什么的？

STP，Start of TLP，在 128b/130b 编码 Data Block 中指示开始发送 TLP 的 Frame Token，如图所示。该 Token 中携带有接下来要发送的 TLP Length 及 TLP Sequence Number。TLP Length 是个相当重要的字段，为了保护该字段，STP 中规划了 FCRC (Frame CRC) 及 FP (Frame Parity) 两个字段来对 TLP Length 进行保护。

FCRC 通过 10 Bit TLP Length 按照一定算法计算获得，只保护 TLP Length，能够检测出 2 Bit 错误。常规 FP 奇偶校验则是校验 TLP Length 及 FCRC 字段，缩减版 FP 只检验 TLP Length 中的特定位，能够检测出奇数位的 Bit 错误。



STP Token

▲图： STP Frame Token

231. 如果发送的 FTS 不够，导致从 L0s 回退到 L0 失败了怎么办？

从 L0s 回退到 L0 时，Tx 发送 N_FTS 个 FTS。如果 Rx 在此期间实现了位锁定、符号锁定及块对齐，那么能够成功回退到 L0。如果 Rx 再次期间没有完成位锁定、符号锁定及块对齐，那么 Rx 会超时退到 Recovery，沟通加大 N_FTS，在 Recovery 阶段完成未完成的锁定及对齐工作，然后进入 L0。

Token

232. 在 Gen3 以后，STP Token 中的 TLP Sequence Number 和 TLP Length 是数据链路层产生的，还是物理层产生的？

从 PIPE 接口上看，我理解都是 MAC 产生的，不需要 PHY 处理。

233. PCIe 设备给主机发 EDB Token 可能是什么原因导致的？

情况补充：在 LTSSM 的 Polling 阶段就出现了上述情况，IP 给主机发的第一个 OS 中就出现了 EDB Token。

正常来讲，EDB 表示代表 TLP 出错了，TLP 传递到物理层发现了问题，由物理层添加了 EDB。但目前还处于 Link Training 的 Polling 阶段，且是 EP 发给主机的第一个 OS，猜测是此时主机 Rx 还没锁定，解析错位了。

234. PCIe Ordered Set 和 Framing Token 有何区别？

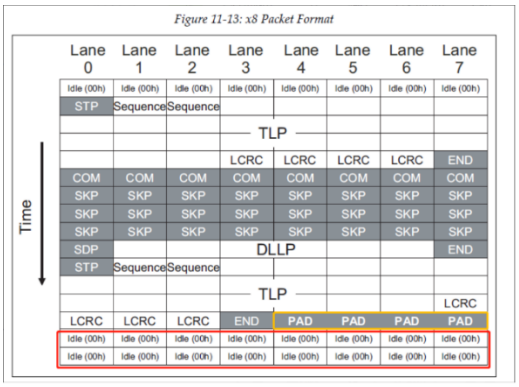
有序集用于链路训练、时钟偏差补偿、进出电气闲等。有序集长度为 4 B 整倍数，一般 16 Symbol（SKPOS/EIOS 4 symbol @Gen1/2）。Gen3+ 时，有序集的同步头为 01b，在 Orderer Sets Block 中发送，有序集的 Symbol 0 来指示当前有序集的类型。

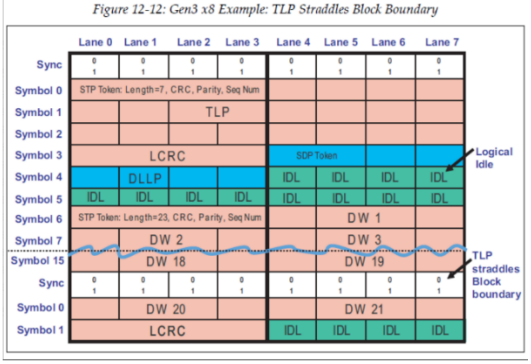
Framing Token 是帧令牌，标记帧的类型、起始、长度、状态、结束等等。Framing Token 长度较短，为 1, 2, 4 Symbol。Framing Token 是 PCIe Gen3 出现的概念，Framing Token 中的 STP、SDP、EDB、IDL 在 Gen1/2 时候以特殊符号（K Code）的形式存在。Gen3+ 时，Framing Token 的同步头为 10b，在 Data Block 中发送。

Logic Idle

235. Logical Idle 未同时出现在所有有效 Lane 上是否违反协议？

问题详述：Mindshare 书中讲 Logical Idle 时提到，Idle 必须同时出现在所有 Lane 上，故此前需要用 PAD 填充（下左图黄色），但右图 Block Boundary 示意图中并未遵循该规则，这两处是否存在矛盾？





不矛盾，左图是针对 8 B/10 B 编码，右图针对 128 B/132 B 编码，两者规则不同。

236. PCIe 是如何判断链路处于电气闲的？

PCIe 设备有两种方式来获知当前链路是否处于电气闲状态：

- ✓ 直接检测：模拟电路直接检测电路是否电气闲。
- ✓ 间接推测：依据收到的进行推测链路是否处于电气闲。

模拟电路直接检测的方法不必多说。出现以下情况时，PCIe 设备可推测链路处于电气闲。

- ✓ L0 状态下，128 us 时间窗内未检测到流控 DLLP 更新和/或未检测到 SKP。

- ✓ Loopback.Active 状态下，Loopback Slave 在 128 us 内未检测到退出电气闲（仅 Gen1）。
- ✓ Recovery.RcvrCfg 状态下，1280 UI 时间内未检测到 TS1/2 @Gen1/Gen2，4 ms 时间内未检测到 TS1/2 @Gen3+。
- ✓ Recovery.Speed 状态下，若上次速率协商成功，1280 UI 时间内未检测到 TS1/2 @Gen1/Gen2，4680 UI 时间内未检测到 TS1/2 @Gen3+。
- ✓ Recovery.Speed 状态下，若上次速率协商失败，2000 UI 时间内未检测到 TS1/2 @Gen1，16000 UI 时间内未检测到 TS1/2 @Gen2+。

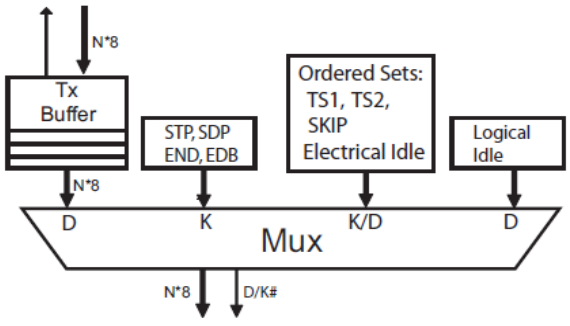
若此前已经处于电气闲，不可依据推测结果判断是否已经退出电气闲。在具体实现时，为了节省面积，往往在 LTSSM 中统一设置一个 Common Counter 来进行推测，而非每条 Lane 单独设置一个 Counter。

包的收发处理

237. Multiplexer 和 De-multiplexer 有何作用？

Multiplexer 多路复用器位于 Tx 端，负责按需选通要发送的数据，包括 DL 层下来的 Data、K 控制字、Framing Token、有序集及逻辑闲等，如图所示。在链路训练阶段，选通 K/D 通道，发送 TS1/TS2/SKIP/EIOS 等有序集进行链路训练；链路训练完毕后，若 DL 层有数据下来，则有序选通 D 通道及 K 通道，在 TLP/DLLP 头尾插入 STP/SDP/END/EDB，插入 SKPOS 等；没有数据下来时，选通逻辑闲 D 通道，发送逻辑闲。

De-multiplexer 是 Multiplexer 的逆过程，位于 Rx 端，对组装好的数据进行解析及拆分。



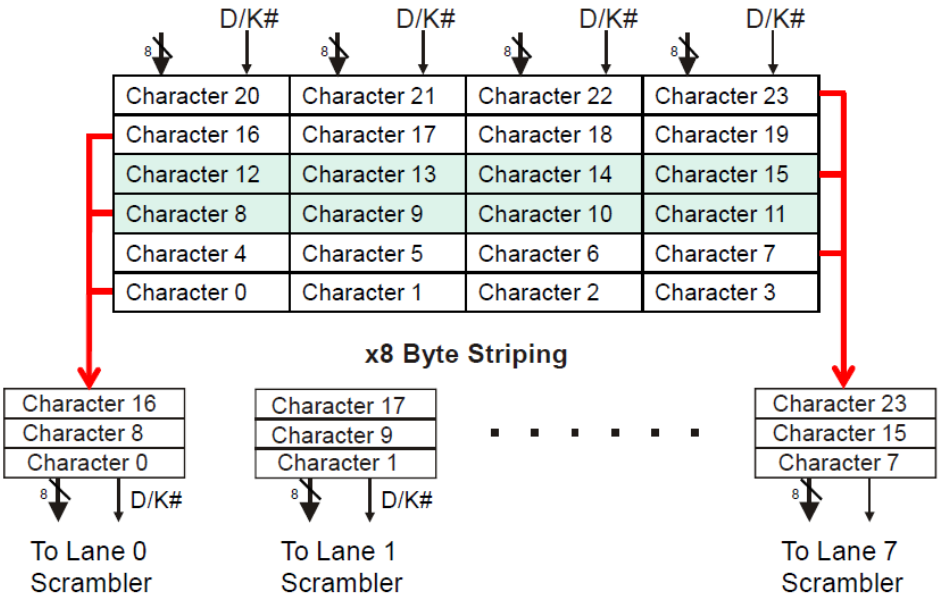
▲图： Tx Multiplexer 示意图

238. Byte Striping/Un-striping 是做什么的？

Byte Striping 位于 Tx 端，是对于有多条 Lane 的 Link 而言的。Byte Striping 内，Tx 把 Multiplexer 出来的待发送的串行 Byte 数据按照 Byte 依次分配到该 Link 多条 Lane 上并

行发送出去。图是 x8 link Byte Striping@8b/10b 的一个例子。128b/130b 时，各条 Lane 先发送 Sync Header 再发送 Block Symbol。

Byte Un-striping 是 Byte Striping 的逆过程，位于 Rx 端，对各条 Lane 上的数据进行解析重组为串行 Byte，送给 De-multiplexer。



▲图： x8 Byte Striping@8b/10b 示意图

239. Packet Filtering 是做什么的？

Rx 收到的数据中，有 OS、TLP、DLLP、IDL 及控制字/Framing Token 等。这些数据中，OS、IDL 等是 PL 层自用的，TLP、DLLP 需要上传到 DL 层。Packet Filtering 就是干这个的，来过滤 Packet，决定是否将 Packet 送给 DL。

240. TLP、OS、DLLP 同时要发送时，该怎么发？

若数据链路层同时有 TLP、TS、DLLP 要发送时，是需要按照一定的优先级进行发送的。官方文档中给了一直建议的发送顺序（不强制），优先级从高到低依次为：

- ✓ 进行中的 TLP 或 DLLP 的 Completion
- ✓ Ordered Set（这个是在 Mindshare 文档提到的）
- ✓ Nak DLLP
- ✓ Ack DLLP
- ✓ FC DLLP（遵循流控规则要求的比较紧急的 FC DLLP）
- ✓ Retry Buffer 内需要重传的 TLP
- ✓ FC DLLP（其他不紧急的 FC DLLP）
- ✓ 事务层待发的 TLP

- ✓ 其他 DLLP，比如 NOP DLLP 及 Vender Defined DLLP

241. Block Alignment 块对齐是什么？

Block 是 Gen3 及以上速率的概念。Gen3 及以上速率时，在完成 Bit Lock 后进行 Block Alignment Lock。Block Alignment 块对齐是在 Recovery 状态进行的。Tx 通过发送 EIEOS 来分隔 Block，Rx 依据收到的 EIEOS 界定 Block 边界。若 Rx 收到了 EIEOS 且接下来 130 bit 正确收到了有序集，则 Block 边界查找成功。否则继续检测 EIEOS。Block Alignment 有 3 个 Phase：

- ✓ Unaligned Phase，未对齐阶段。Rx 收到 EIOS 后进入 Unaligned Phase。在此阶段 Rx 依据 SKPOS 调整 Block Alignment，Rx 收到 EIEOS 后进入 Aligned Phase。
- ✓ Aligned Phase，对齐阶段。在此阶段 Rx 依据 SKPOS 调整 Block Alignment。若 Rx 收到 EIEOS 则对 Block Alignment 进行调整。若收到 SDS，进入 Locked Phase。若 Rx 收到未定义的同步头，退至 Unaligned Phase。
- ✓ Locked Phase，锁定阶段。为防止数据丢失，Rx 进入此阶段后不再调整其 Block Alignment。若 Rx 收到未定义的同步头，退至 Unaligned Phase 或 Aligned Phase。

242. 位锁定是什么? 位有什么好锁定的？

PCIe 的位锁定**是指 Rx PLL 锁定在了与 Tx 相同的频率上，不是说锁定了某个 Bit**。Rx PLL 依据接收数据流的 Bit 跳变，使得 Rx 还原时钟频率同步于 Tx 时钟。如果没有位锁定，PCIe Rx 无法正确重建 Rx 时钟，也无法正确解析数据。

243. PCIe 是如何实现 Block Type 检测的？

Block Type 检测是检测当前数据流为 Ordered Set Block 还是 Data block，是 PCIe Gen3 及以上的概念，因为 Gen3 时才出现 Ordered Set Block 与 Data Block 之分。

Block Type 检测原理很简单，就是根据同步头中的两个 bit 进行判断，但在判断之前，必须先完成 Block Alignment。如果检测到 Block 为 Ordered Set Block，则当前 Block 为 PL 层自用，不会上传到 DL 层；如果检测到 Data Block，则会上传至 DL 层。

考虑到链路中所有 Lane 上都携带有相同的 Block 信息，可以只在 1 条 Lane 上检测 Block Type。

244. PCIe 是如何检测块对齐丢失的？

在链路训练期间，若 Rx 检测到了未定义的 Sync Header，则认为块对齐丢失（Loss of Block Alignment）。块对齐丢失是因为 Rx Block Alignment 失败，需要重新回退到 Unaligned Phase 进行块对齐。

245. PCIe Lane-Lane Skew 产生的原因及 De-skew 方法？

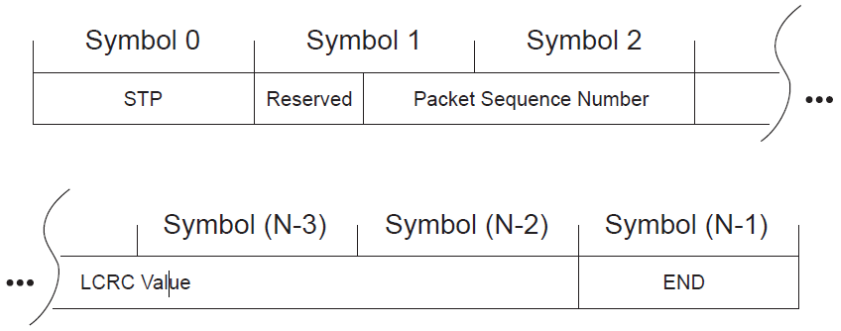
同一 Link 多条 Lane 之间难免存在 Skew，即便 Tx 端采用同一 Clock 同时发送，不同 Lane 上到达 Rx 时间也难以相同。PCIe Lane-Lane Skew 一般源于以下几点：

- ✓ 电气 Driver 及 Receiver 之间存在差异。
- ✓ 印刷线路板阻抗变化。
- ✓ 不同 Lane 的走线长度存在差异。

Gen1/Gen2 时常用检测 OS 序列中的 COM 符号来实现多 Lane 的 De-skew；Gen3 及以上常采用检测 SDSOS、EIEOS、SKP 等方式来实现 De-skew，具体情况需具体分析。

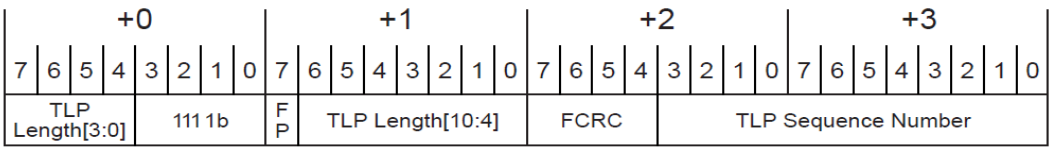
246. 128b/130b 编码发生 TLP 时只有 STP 没有 END，如何知道 TLP 结束？

8b/10b 编码发送 TLP 时，有 STP 及 END 来标志 TLP 的开始结束，STP (K27.7) 及 END (K29.7) 均为 8b/10b 的控制字，两者均不携带 TLP 相关信息，有必要采用 STP 及 END 来标志 TLP 的开始和结束。其帧格式如下图所示。

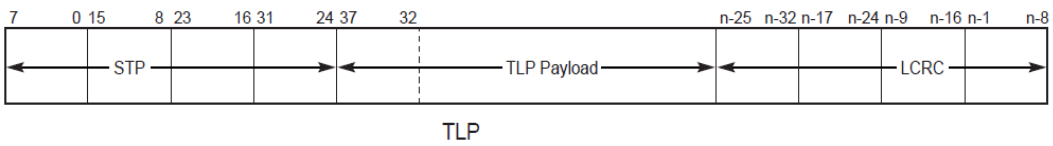


▲图： 8b/10b 编码时的 TLP 帧

128b/130b 编码时，STP 为 Framing Token（图），其中包含了数据链路层提供的该 TLP 的 Length、FCRC、Sequence Number 等信息。Rx 可以依据 STP Token 中的 TLP Length 来获取 TLP 的长度，无需专门的 END。其帧格式如图所示。



▲图： STP Framing Token 格式



▲图： 128b/130b 编码时的 TLP 帧

247. PIPE 口上观测到了 TS1/TS2，为什么 PCIe Controller 内没作统计？

不一定有 TS1、TS2 就要统计。对于 8 GT/s 及以上速率，LTSSM 只对 Block Aligned 块对齐之后收到的 TS 进行统计。块对齐的一大标志是 Rx 收到了 EIEOS。

对于 Recovery.RcvrLock 状态而言，若其上一状态为 L0s、L1 或 Recovery.Speed，检测到 EIEOS 后视为块对齐。若其上一状态为 L0，收到 EDS 后视为块对齐。以上说的是初次在某速率下（8 GT/s 及以上）进入其 Recovery.RcvrLock 状态的情况。

若上一状态为 Recovery.Equalization、Recovery.Idle，块对齐的依据是什么？还需要重新检测到 EIEOS 后再计数 TS 吗？Spec 中未明确说明。依据个人经验，无论 Recovery.Equalization -> Recovery.RcvrLock 还是 Recovery.Idle -> Recovery.RcvrLock，此前 LTSSM 已经在该速率下进入过 Recovery.RcvrLock，块对齐已经完成，此时收到 TS 可以直接计数，无需等待 EIEOS。

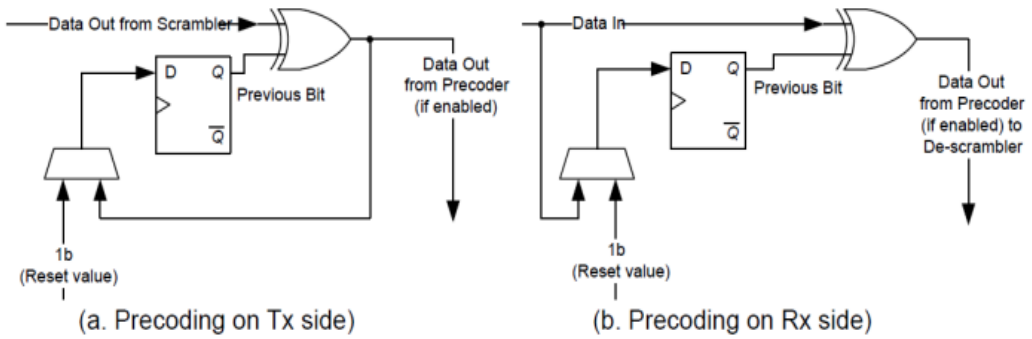
编解码/加扰

248. PCIe 预编码（Precoding）是干什么的？

PCIe 接收端采用 DFE (决策反馈均衡器) 进行判决，在 32 GT/s 速率下比 16 GT/s 更容易发生 Burst Error (连续突发错误)。为了减少 Burst Error 的发生，Gen5 速率下可以选配 Precoding。Precoding 仅适用于 32 GT/s 及以上速率，其他速率时不开启。PCIe Rx 通过 EQ TS2 请求 Tx 开启 Precoding。

Precoding 可以把连续多 bit 的 Burst Error 拆分为 2 bit 的 Entry Error 及 Exit Error，从而达到消除 Burst Error 的目的。对于单 bit 的突然错误，开启 Precoding 后会变成 2 bit 错误，从而使系统误码率 BER 会提升为之前的两倍。

Precoding 发生于 Tx Scramble 之后、Rx De-scramble 之前。开启 Precoding 不会影响信号质量及完整性。PCIe Precoding 示意图如下：



▲图： Precoding working the scrambler/ de-scrambler

249. PCIe 的物理层的 Precoding 有什么作用？

Precoding 可以把连续多 bit 的 Burst Error 拆分为 2 bit 的 Entry Error 及 Exit Error，从而达到消除 Burst Error 的目的。对于单 bit 的突然错误，开启 Precoding 后会变成 2 bit 错误，从而使系统误码率 BER 会提升为之前的两倍。

（参考：https://blog.csdn.net/weixin_40357487/article/details/124049739）

250. Precoding 可以只在 USP 开启，不应该在 DSP 和 USP 同时开启吗？

引文：协议 4.2.2.5。

没有要求 USP 开了 Precoding 后 DSP 也必须开 Precoding。USP 的 Tx 开启 Precoding，DSP 的 Rx 必须开启 Decoding。

251. 128b/130b 编码是如何做到 DC Balance 的？

128b/130b 编码本身没有 DC Balance 功能，Scramble 有一定的 DC Balance 能力，Gen3/4/5 速率时候靠的是 Scramble 来做 DC Balance。频率升高后，CDR 对长 0、长 1 敏感度降低，DC Balance 也没那么严重。所以 PCIe 到 Gen3 后，换成效率更高的 128b/130b，加扰的 PRBS 伪随机序列也更长。

252. 128b/130b 编码时 Sync Header 和 TS Symbol 0 不做 Scrambling 不会影响 DC Balance 吗？

128b/130b 编码时 Sync Header 和 TS Symbol 0 不做 Scrambling，不会影响 DC Balance。Sync Header 为 01b 或 10b，TS Symbol 0 为 1Eh(00011110b)，0、1 数量相同，本身已经实现了 DC Balance。

253. 128b/130b 编码时终止数据流传输的 EDS 为 Framing Token，为什么启动数据流传输的 SDS 却要是有序集的形式？

128b/130b 编码时，PCIe 的 Block 分为 OS (Ordered Set，有序集) Block 及 Data Block，OS Block 期间传输 OS，Data Block 期间传输 Data Stream。

链路通过发送 SDS 从 OS 传输切到 Data Stream。发送 SDS 的时候，链路仍处于 OS Block 传输，按道理也应该发送有序集，所以 SDS 为有序集的形式。

链路通过发送 EDS 从 Data Stream 传输切到 OS 传输，此时链路处于 Data Block 传输时段，EDS 不应为有序集的形式，又不便采用 TLP、DLLP，采用 Framing Token 是最佳选择。

254. Gen3~5 速率时候采用 128b/130b 编码，是如何保证码流没有连续长 0 或者长 1 呢？单靠加扰能避免吗？

Gen3~5 时加扰能在一定程度上避免长 0 长 1，但不能保证没有连续长 0 或者长 1。何况 128bb/130b 编码 Sync Header 中有 2b 的 01 跳变，连续也连续不了太长，簇发的连续 100 多个连 0/1 几乎没有影响。再说，不需要保证极端情况，PCIe 本身有重传机制，失锁大不了 Retrain。

255. PCIe Tx Scramble 放在 Encode 之前之后？为什么？

乱序加扰应该放在编码之前。

发送序列中存在连续重复序列时，发送能量会集中在特定频点附近较窄的范围，产生较大的电磁辐射。scramble 乱序，是为了消除连续出现的重复序列，把能量集中的离散能量谱打散成白噪声，从而大大减小 EMI。

8b/10b，128b/130b，1b/1b等编码，在串行数据流中加入时钟信息，使得接收端PLL能够从数据流中恢复出时钟。编码有以下益处：① 减少时钟布线的问题；② 减小时钟线引入的电磁干扰；③实现直流平衡；④增强误码检测性能。

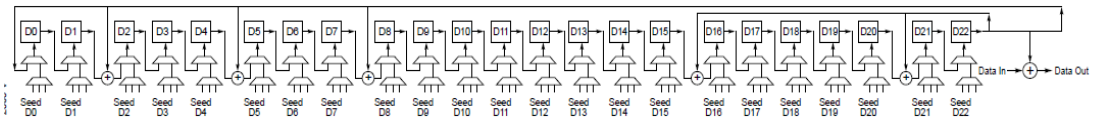
若先编码再乱序，数据流中的时钟信息会被破坏，直流均衡也难以保证。

256. PCIe Scrambling 中有哪两种 LFSR 方案?

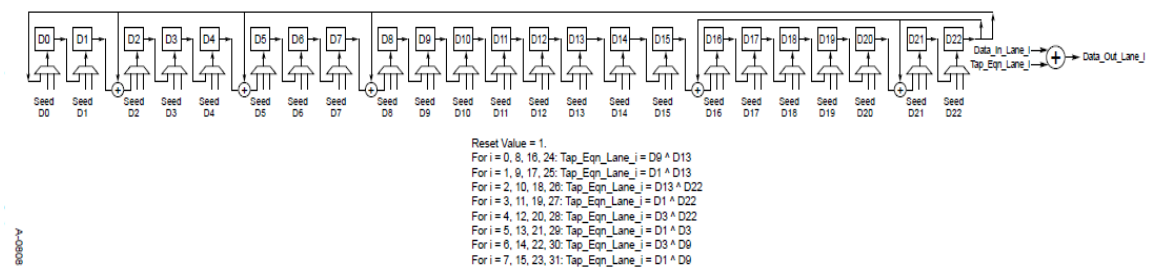
PCIe Scrambling 中常用的两种 LFSR 方案为 Separated LFSR 与 Shared LFSR。

Separated LFSR 方案，或称 Individual LFSR，是指 PCIe 链路中每条 Lane 都单独配一个 LFSR 进行 Scrambing，各条 Lane 的 LFSR 有自己的 Seed。其原理图如图所示。

Shared LFSR，从名字也看得出来，共享的 LFSR，是指 PCIe 链路中所有链路共享同一 LFSR，不同 Lane 采用不同的抽头点位，其原理图如图所示。



▲图： Individual LFSR 原理图



▲图： Shared LFSR 原理图

257. 扰码种子如何传递到对端的？

每一条 Lane 的 Seed 都是固定的，无需传递给对端或从对端获取。

258. 8b/10b 编码已经能均衡 0 和 1，为何还需要加扰？

Gen1~Gen5 中都需要加扰。8b/10b 是为了 01 数目均衡做 DC Balance，扰码是为了间隔均衡做展频。编码后实现了直流均衡，但仍然可能频谱比较集中、辐射较大，加扰能够把频谱打散一点，降低对外部的电磁干扰（EMI）。在开了 SSC 后，为了降低 EMI 必须要加扰。

259. 为什么要关闭加扰 Scrambling？

默认情况下 Scrambling 是打开的。Scrambling 的确有很多好处，比如减低 EMI，但在设备调试阶段，工程人员想要更容易地查看传输的数据，数据 Scrambling 无疑增加了数据辨识难度。为了更容易地查看传输数据，可以选择关闭 Scrambling 功能。只能在链路训练的 Configuration.Complete 阶段通过发送 Disable Scrambing 位（Symbol 5 Bit 3）为 1 的 TS1 或 TS2 来关闭对端的 Scrambling 功能，且只能在 8b/10b 编码期间来关闭 Scrambling。

260. PCIe 加扰解扰时，什么情况下会初始化 LFSR ？

Gen1/Gen2 时，COM 符号初始化 LFSR 为 FFFFh。

Gen3 及以上时，发送/接收完 EIEOS 最后一个 Symbol 初始化 LFSR，lane_num[2:0] 不同初始值也不同。

时钟

261. PCIe 参考时钟为何都是 100 MHz？

应该是因为 100 MHz 数字比较整，内部 mplla 倍频到 16 GHz 相对容易。当然并非一定要 100 MHz，也可以 125 MHz 等其他频率，只要 PHY 内部参数配置正确即可。

262. PCIe 的 100 MHz 参考时钟（refclk）是用来帮助 CDR 锁定嵌入在数据流中的时钟的吗？物理层工作的时钟也是这个参考时钟提供的吗？

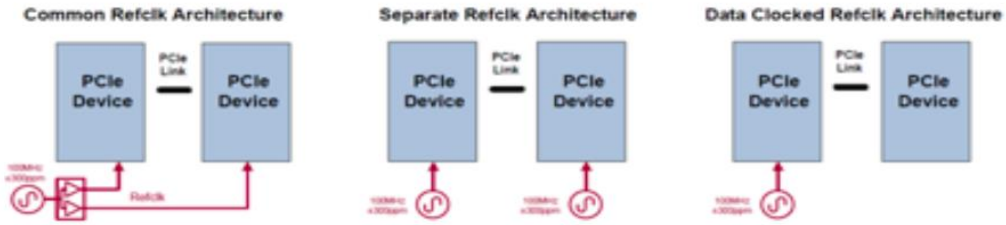
Refclk 可作为基准，通过 PHY 内的 PLL 产生 PIPE 时钟。在 Tx/Rx 端，PCS 都是用 refclk 作为基准通过 PLL 产生的 PIPE 时钟。在 Rx 端，refclk 可辅助 Rx CDR PLL 锁定频率往 Elastic Buffer 内写数据，同时以 refclk 为基准的 Local PLL 生成的时钟从 Elastic

Buffer 内读数据把接收数据同步到 PIPE 时钟域。Rx 端 CDR 恢复时钟到 Elastic Buffer 就结束了，后面用的都是 refclk 分频倍频的时钟。

263. PCIe 有哪几种参考时钟架构？

PCIe 有 3 种时钟架构，分别为：

- ✓ **Common Refclk Architecture**，通用参考时钟，收发端共享参考时钟。
- ✓ **Data Clocked Refclk Architecture**，仅发送端需要 refclk，接收端 CDR Refclk 参考时钟从数据流中恢复。
- ✓ **Separate Refclk Architecture**，收发端采用独立的参考时钟，根据有无时钟扩频（SSC）可进一步分为 SRNS 及 SRIS。



三种基本 PCIe 参考时钟架构

264. CC 时钟模式时收发端有频差吗？为什么 CC 时还需要 Elastic Buffer？

CC 时钟没频差，SRIS 和 SRNS 有频差。虽然 CC 时钟链路两端的 Tx/Rx 之间频率相同，但两者之间的随机 Jitter 抖动或相位差异较大，CDR 恢复出来的时钟和本地 refclk PLL 之间已经不能算作同步，依然需要 Elastic Buffer，只是 Buffer 深度比 SRIS 的浅很多。

265. SSC 有什么用？

抗电磁干扰的。一方面认证时会看是否支持 SSC，另一方面系统对 EMI 有要求的话会开 SSC。Gen5 中 SSC 已经快变成必选项了。

参考：PCIe 参考时钟架构 (Refclk Architecture)-MangoPapa

266. CC 时钟模式可以开 PHY 内部的 SSC 吗，还是说一定要接外部的 SSC？

题中意思为，CC 时钟模式时 refclk 本身不做 SSC，refclk 给到 PHY 内的 PLL 后由 PHY 的 PLL 加 SSC。

SSC 一般是加在 refclk 上的，即接外部的 SSC，这样难度较小。在 PHY PLL 支持自己加 SSC 的情况下，题中这种方式也是可以的。

267. CC 时钟架构下 SSC 可以不加在 Common refclk 上、加在 PHY Internal Clock 上吗？

可以，要在 PHY 内加的话，两边 PHY 都要加，只在一边加没有意义，且会给对端 Rx 造成困难。采用 CC 时钟且在两边 PHY 上加 SSC，两侧展拼不同步，其实已经不属于 CC，物理行为类似于 SRIS。

268. 如何判断 refclk 有没有开 SSC？

SSC 为频偏 5000 ppm，可以用 LeCory 协议分析仪可以粗略看 refclk 的频率，如果大约 100 M 就是 refclk 没开 SSC，如果是是 99.75 M 说明 refclk 开了 SSC。

269. SSC 扩频时钟和 PCIe 插 SKP 之间有何关系？

不同时钟模式的误差不同，所需插入 SKP 的频率也不同。

270. Host 开启 SSC 后，EP 需要同步开启 SSC 吗？

如果 SSC 是 Common RefClk 上，那 EP 不用开 SSC，且 RC 和 EP 都不用开。如果 refclk 采用固定 100 M，则需要对 PHY 做些设置加 SSC。

271. PCIe 扩频前后时钟频率变化了吗？

变了，频率范围变宽了，在 5000 ppm 以内变动。

272. Common Clock 同源时钟模式下，Gen4 PCIe Training 完成之后还可以开展频吗？

可以，配好后再 Recovery 或者 Retrain 一次。

同源的本质就是两边的误差可以同步抵消，PLL 可以跟随几十 k 的扩频频率，本地 RX CDR 和对方 Tx 发过来的数据也基本无频偏。如果是在同源时钟的源端参考时钟做展频，从 PLL/CDR 角度看 Link Up 后再开展频对链路没什么影响；如果是在两端的 PHY 内做展频，跟 SRIS 差不多。

如果担心 Link Up 后再开 SSC 的性能，可以在打开/关闭展频后测一下回环 BERT，噪声容限结果应该是一样的。

273. PCIe Link 上的 RC 和 EP 采用非同源时钟，其中 RC 端启用 SSC 扩频后仍无法 Link 是何原因？是否需要同步打开 PCIe Controller 的 SRIS？

需确认设备端是否支持 SSC 模式。对于同源时钟，可以只在 Link 一端开 SSC，对于非同源时钟，需 Link 两端同时支持并开启 SSC。

反馈：设备芯片采用了 8748，不支持 SSC。

274. 在 Link Up 后再开 SSC 是否合规？如何证明是由于该卡容错性差导致的？

补充 1：同样环境测试 2 万张卡就这一家卡有两张出问题，出问题时候链路报 CE 错误。

补充 2：上来就打开 SSC 不会出现该情况，Link Up 后开 SSC 会小概率异常。出现异常时 LTSSM 会在 L0 和 Recovery 之间循环，卡无法正常工作，最终大概率卡会丢了或者重启。

建议：怀疑是瞬时频偏引起的异常。可测试展频打开的瞬间源时钟频率有没有跳变稳定一段时间之类，有的话系统还是得反应一下。有测试手段的话，可以查查此时的 pipe rxstatues，rxvalid 等等。

275. PCIe RC 的 refclk 采用内部时钟模式，PHY 输出 100 M 差分时钟，50 Ω 端接是放到 RC 主板的连接器端、还是放在 EP 设备端？

有意见表明，如果 EP 需要 RC 提供 100 MHz 参考时钟，50 Ω 单端电阻需要放在 EP PHY 内部以防止反射。

另有近似意见表明，可以在 EP 外部靠近 refclk_p/n 的 PAD 处外置 50 Ω端接，同时 PHY 内部也放置 50 Ω端接并支持寄存器配置启闭。

276. 主机 BIOS 里面的 Spread Spectrum 就是 PCIe 的 SSC 吗？

是的。

277. 如何证明 BIOS 里的 Spread Spectrum 就是 PCIe 的 SSC，而不是其他地方的？

可以用分析仪量一下金手指上的 100 MHz 参考时钟，开了 SSC 后时钟频率为 99.75 MHz 左右，没开是 100 MHz 左右。

278. 一般主板里除了 PCIe 哪里还需要 SSC？

SSC 等高速接口也有 SSC，但 SSC 没有输出时钟。USB 不是时钟源头展频，都是 PHY 内展频，类似于 PCIe 的 SRIS 类似。

279. 如何确认 EP 对接服务器的某个卡槽上 SSC 是否开启呢？

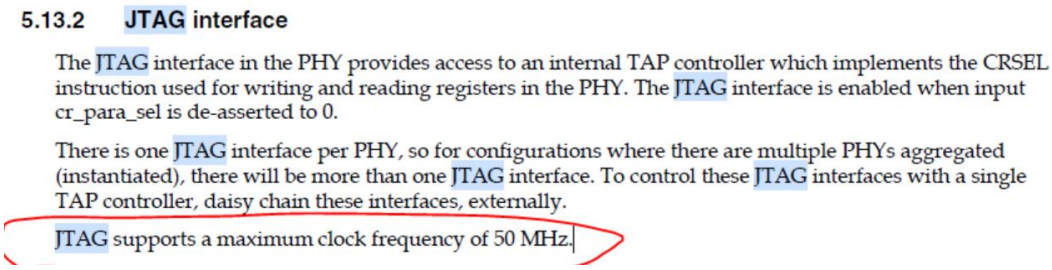
可以看 BIOS 上相关选项是否打开。如果是 Separate Clock，在 Link Control Status 也能看 Clock Mode 是否为 SRIS，但好像要到 Gen5 及以上才有。

280. 板子提供参考时钟时 PCIe 内还需要 CDR 吗？

需要。PHY 内需要 CDR 来恢复时钟。

281. Synopsys 的 PHY 的 ATE 测试 JTAG Clock 的最大频率是多少？

跟 PHY 有关。对于 Gen5 PHY，最大 125 MHz，一般用个 20 MHZ 就可以了，下图例子中的 PHY 最大支持 50 MHz。详参 PHY Databook。



5.13.2 JTAG interface

The JTAG interface in the PHY provides access to an internal TAP controller which implements the CRSEL instruction used for writing and reading registers in the PHY. The JTAG interface is enabled when input cr_para_sel is de-asserted to 0.

There is one JTAG interface per PHY, so for configurations where there are multiple PHYs aggregated (instantiated), there will be more than one JTAG interface. To control these JTAG interfaces with a single TAP controller, daisy chain these interfaces, externally.

JTAG supports a maximum clock frequency of 50 MHz.

弹性缓存

282. PCIe Elastic Buffer 弹性缓存是做什么的？

Elastic Buffer 是 Rx 端的一种异步 FIFO，通过在 Elastic Buffer 中插入或删除 SKP，来解决 PCIe 接收数据 Tx/Rx 跨时钟域问题。

Elastic Buffer 输入时钟为 Recovered Clock，输出时钟为 Rx Local Clock。当 Rx Local Clock 比 Recovered Clock 快时，Elastic Buffer 会出现 Underflow，在 Empty 之前往 Elastic Buffer 中插入 SKP；当 Rx Local Clock 比 Recovered Clock 慢时，Elastic Buffer 会出现 Overflow，在 Full 之前从 Elastic Buffer 中删除 SKP。

283. PCIe 6.0 没有 Elastic Buffer 了，那么是如何做时钟恢复的？如何解决的频差问题？

问题不正确，PCIe 6.0 里既有 CDR 也有 Elastic Buffer，且时钟恢复不依赖于 Elastic Buffer。

284. Elastic Buffer 中的 Nominal Empty Mode 大概是什么原理？

简单而言，Nominal Empty Mode 时 PHY 会移除所有接收到的 SKP 符号，且不会主动插入 SKP，使 Elastic Buffer 尽可能保持在空的状态，并通过 RxDataValid 信号来告知 MAC 是否有可用的数据。相比于另一种模式（Nominal Half Full Mode），Nominal Empty Mode时延时更低。具体可查看 PIPE Spec 中 Clock Tolerance Compensation 章节。

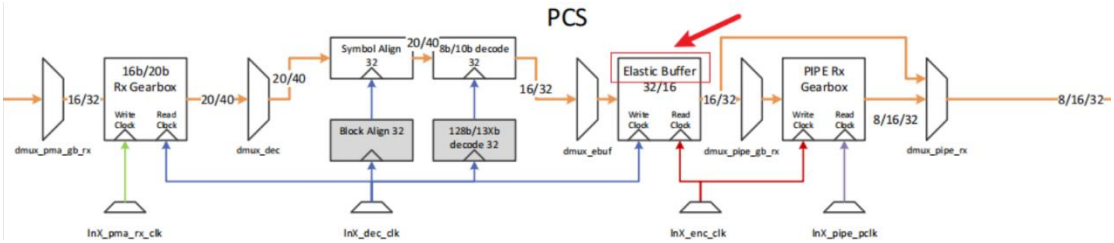
285. 为什么弹性 Buff 深度加大，数据传输路径延时会增加？

引文：弹性缓存深度加大后，数据传输路径延时也增加了。（出自 MangoPapa 的博文《PCIe 参考时钟架构》）

一方面，由于 SSC 的引入，Elastic Buffer 中读写指针间距增加，导致延时加大。另一方面，SSC 时候插入 SKP 较多，加大了 Busy 状态下的总延时。

286. 链路存在 Deskew Error，可以通过增加 Elastic Buffer 深度来解决吗？

问题详述：EP 为自研芯片，基于 PLDA 的 Controller 及 GF 的 PHY，CC 时钟结构。Gen4 x16 链路在 L0 状态反复 LTSSM_Down 进 Recovery 状态并最终回到 L0，此间 EP 报出 Deskew Error 及 128b/130b Frame Error，且 dl_link_up 一直为 1。降速到 Gen3 就没问题。



不能靠增加 Elastic Buffer 深度来解决 Deskew Error。

对于上述情况，可尝试调整 Preset 值来获取一个更优的 FOM 值，或者尝试调节 Rx Lane Margin。CC 场景时，如果开启 EP PHY 的 SSC，必须关闭 SSC。此外，也许关注 Channel 的情况，如果 Channel 太差的话，调整 Preset 也不一定能完全解决问题。

链路训练

协议理解

287. PCIe Link Training 期间都训练了什么？

位锁定、符号锁定、Block 对齐、链路宽度、链路速率、极性、链路反排、Lane 间消抖。

288. PCIe Gen2 也需要从 Gen1 切速吗？能不能直接进 Gen2？

需要。不能。

无论收发端支持的最高速率是多少，第一次 Training 必须进 Gen1 L0，这是协议要求的。

即使是 Gen2 速率，第一次也要按照 Gen1 进行 Training。到达 Gen1 L0 后，由于还未达到最高速率，LTSSM 进入 Recovery，状态跳转如下：

L0 (Gen1) -> Recovery.RcvrLock -> Recovery.RcvrCfg -> Recovery.Speed -> Recovery.RcvrLock -> Recovery.RcvrCfg -> Recovery.Idle -> L0 (Gen2)。

289. PCIe Link Training 时，双方协商是从 Gen1 直接往最大速率跳转吗？

不是，必须做 Gen1，如果为 Gen2 则进一步做 Gen2，否则跳过 Gen2 继续做 Gen3、Gen4、Gen5 直到协商好的最高速率。对于 Gen5，如果设置了 bypass_eq_to_highest_rate=1，可以从 Gen1 直接跳到 Gen5。

如果最高速率为 Gen4，有些 VIP 作为 RC 对接 EP 时，可以 Bypass 掉均衡的 Phase 2 和 Phase 3，但是会经历 Phase 1，还是从 Gen1 -> Gen3 -> Gen4。经过 Phase 0/1 后，TS1 Order Set 可以被正确接收，BER 达到了<1e-4 的要求，如果认为此时信号质量达到要求，可以不进行 Phase 2/3 的调节。

290. 请教 PCIe LTSSM Detect.Quiet 的理解：

All Receivers must meet the the ZRX-DC specification for 2.5 GT/s within 1 ms (see Table 8-10) of entering this substate. The LTSSM must stay in this substate until the ZRX-DC specification for 2.5 GT/s is met. 1. 第一句的 1 ms，有什么根据吗？是否跟第二句矛盾？ 2. 第二句是否可以理解为，只要 ZRX-DC 满足 40-60 Ω(GEN1&GEN2)就立刻退出 Detect.Quiet？如果是这么解释的话，如何做到的呢，不停发检测脉冲？

可以理解为 12 ms 的检测脉冲间隔。

291. LTSSM 中的 Polling.Active 时需要发 1024 个 TS1，这些 TS1 是用来 PLL 恢复时钟的吗？

不全是。Polling 状态主要是用来做 Bit Lock、Symbol Lock 和极性翻转。

292. Polling.Active 中两端的 1024 个 TS 有什么发送顺序吗？

没有顺序要求。

假设 Compliance Bit = 0，LinkNum = PAD，LaneNum = PAD 。

Detect 完成之后 Tx/Rx 进入 Polling.Active 状态，任一端确保其 Tx 部分稳定后即可发送 TS1，没有先后顺序要求，具体取决于退出 Detect 的时刻。两端 Rx 依据其收到的 TS1 进行位锁定和符号锁定。任一端所有有效 Lane 上发送完 1024 TS1 且收到 8 TS1/TS2 后进入 Polling.Configuration，改发 TS2。

以相连的 A、B 两个 PCIe 设备为例，重点说下为什么收到 8 TS1 或 8 TS2 ：A 发完 1024 TS1 收到 8 TS1 后进 Polling.Configuration，A 开始发 TS2；A 开始发 TS2 了，B 还处于 Polling.Active，这时候 B 有可能还没完成锁定，那么 B 一个 TS1 都收不到。等完成锁定，收到的已经是 A 发来的 TS2 了。所以收到 8 TS2 也可以进 Polling.Configuration。

293. Polling.Active 因超时跳转到 Polling.Configuration 条件中，是说收到第一个 TS1 时，已经发完 1024 个 TS1，此后收到 8 个 TS1 或者 TS2，跳转到 Polling.Configuration 吗？

不是。解释如下：

- ✓ 进入 Polling.Active 后 24 ms 内，如果发完了 1024 个 TS1 且在 Detect 到的所有 Lane 上都收到了 8 个连续的 TS1 或 TS2，则可以跳到 Polling.Configuration；其中 TS1/TS2 的 Link Num 需要为 PAD，TS1 的 Compliance Bit 不为 1；
- ✓ 若 24 ms 内未在所有 Detect 到的 Lane 上收到上述 8 个连续的 TS1 或 TS2，但在某些 Lane 上收到过上述 TS1 或 TS2，则在收到第一个 TS1/TS2 后至少应继续发送 1024 个 TS1 才能跳转至 Polling.Configuration。

294. 上述 Polling.Active 跳到 Polling.Configuration 条件中，可以收到 TS1 或 TS2，Polling.Active 状态下不是发 TS1 吗，为什么会收到 TS2？

考虑以下情况：Polling.Active 期间对端快速完成 Bit Lock、Symbol Lock 并先于本端进入 Polling.Configuration，对端由发 TS1 转为发 TS2，此时本端可能还未完成 Lock 一个 TS1 没看到，等 Lock 时就开始收到 TS2 了，所以也可以看 TS2。

295. PCIe 必须支持 PN 极性翻转吗？

必须支持。如果不支持，一旦出现需要极性反转的情况，就无法成功建链。协议原文如下：Support for Lane Polarity Inversion is required on all PCI Express Receivers across all Lanes independently.

296. 链路训练到 L0 后还能进行链路反转和极性反转吗？

Polarity Inversion、Lane Reversal 分别在 Polling、Configuration 状态下完成，如果抱持在 L0 不进这两个状态是无法进行极性反转、链路反转的。如有需要，可触发 Retrain 重新建链。

297. Polling.Compliance 状态是做什么的？Compliance Pattern 是干什么的？

Polling.Compliance 是 LTSSM Polling 子状态之一，用于 PCIe 链路的合规性测试，与 PCIe 测试设备配合使用。Polling.Compliance 期间，收发端 PCIe 设备发送 Compliance Pattern，在相邻通路间产生最坏的干扰及 EMI，测试设备来评估待测 PCIe 链路上的电压、时序是否符合规范，并测试 EMI、BER 及串扰程度。Loopback 模式下，32 GT/s 及以上速率时，Loopback Master 发送 Modified Compliance Pattern。TS 或 Link Control Register 2 也可以特别要求发送 Modified Compliance Pattern。

正常操作中一般不会进 Polling.Compliance 状态，但是所有的 PCIe 设备必须实现该功能，这是其 DFT 不可或缺的一环。

Polling.Active 状态进入 Polling.Compliance 的 3 个条件（满足其一即可）：

- ✓ Link Control 2 Register 的 Enter Compliance 位置一，即我方主动要求进入 Compliance。
- ✓ 24 ms 内未进入 Polling.Configuration，且检测到的 Lane 中，有 lane 未退出电气闲。此时该 PCIe 设备在所有检测到的 Lane 上发送 Compliance Pattern。无源负载，比如 50 Ω 阻抗的探针或 50 Ω 接地阻抗连接到任一差分信号对上，都会使设备进入 Polling.Compliance。
- ✓ 24 ms 内未进入 Polling.Configuration，且收到了对端发来的 8 个 TS1 中，Compliance Receive bit 置一，且 Loopback bit 为 0（非 Loopback 模式）。

所有 Lane 退出电气闲且发送完 1024 个 TS1 后，Polling.Compliance 退回到 Polling.Active。

298. 链路训练过程中为什么在 Cfg.LaneNum.Wait、Cfg.LaneNum.Accept 和 Cfg.Complete 这三个阶段发 Modified TS1 序列而非标准的 TS1 呢？

并非一定要用 Modified TS。Polling 的时候依据标准 TS 的 Symple5 bit[7:6]来协商是否要用 Modified TS。若双方均支持 Modified TS，则在上述三个状态用 Modified TS，有一方不支持就用 Standard TS。

Modified TS 是为了兼容其他协议，比如 CXL，在 Modified TS 中协商 CXL 相关的 Feature。此外，Equalization Bypass 相关也在 Modified TS 中协商。

299. Downstream Port 中的默认 Link Number 是否为寄存器可配？有哪些设置规则吗？

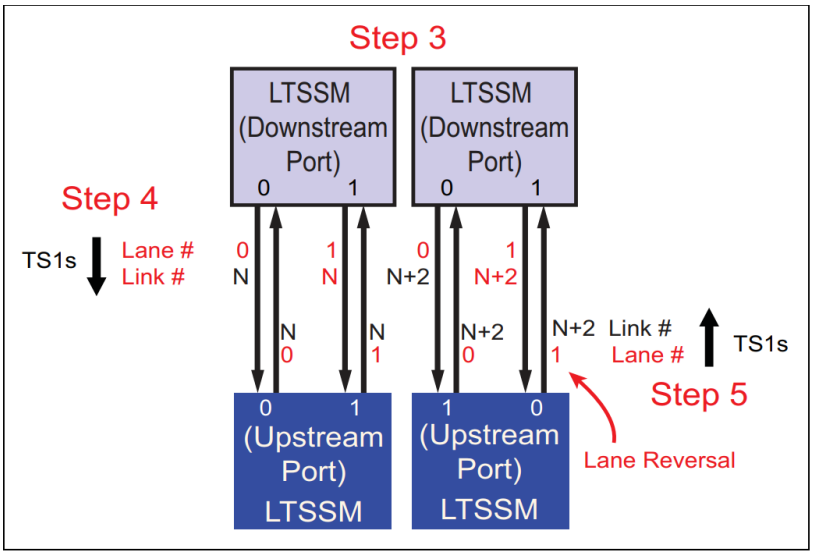
是内部值，一般寄存器可配。Link Number 在 Link Training 的 Configuration 阶段由 DSP 通过 TS 中相关 Symbol 告知 USP，仅对该 Link 有效，可以随便设置。对于 Synopsys Controller，该寄存器在 PORT_LOGIC 寄存器组的 PORT_FORCE_OFF 寄存器内。

300. 状态转移时是否需要判断 TS 中的 Link Number 是否跟 Configuration 阶段协商的 Link Number 一致？

一般不会在 Link Training 期间动态改 Link Number。对于采购的 IP 无需关注，对于自研 IP 可进一步深究。

301. Bifurcation 时两个 DSP 的默认 Link Number 可以设置为相同值吗？

附：下图为 Mindshare 书中举的一个 Bifurcation 双 Link 例子。



一般两条 Bifurcation Link 的 Link Number 是一样的，如果需要单独配置，应该会说明的。这是两条相对独立的 Link，Link Number 一样应该没关系。如果是 RC 多端口，其本身内部应该由多个 Default Link Number。

302. 如果链路中某条 Lane 一直跑不到 2.5 GT/s，链路训练过程状态机会卡在什么状态?

Gen1 都没跑上的话，一般在 detect->polling->detect 循环转圈的现象比较多。

303. 链路建立完成 Lane 数协商之后，还可以写 RP 的配置寄存器实现 Lane 数动态调整吗？

不行，需要重新进行 Link Training。对于 PCIe Gen6，可以走 L0p 来动态调整 Link Width，但仅限于在 Link Training 协商好的 Lane 之内进行调整。

304. 链路训练时为什么有时发 Modified TS 有时发 Standard TS？可以固定发 Standard TS 吗？

Physical Layer 32.0 GT/s Extended Capability 中可以查看 Modified TS 相关状态并进行开关控制。

对于题中的问题，可能有以下原因：

- ✓ 可能存在随机使能 Modified TS 的情况。只有 Link 两端均使能 Modified TS 才能发送 Modified TS，可排查是否存在未开启 Modified TS Enabled 的情况。
- ✓ 若确定两端均开启了 Modified TS，
  - ■ 对于 CXL，可能是 Target Speed 不同导致。只有两端协商速率在 Gen3 及以上速率时才会发送 Modified TS 进行 CXL 协商，不够 Gen3 时则发送 Standard TS。
  - ■ 对于纯 PCIe，可能是 bypass_eq_to_highest_rate 配置不同导致。只有两端均使能 bypass_eq_to_highest_rate=1 时才会发送 Modified TS，否则发送 Standard TS。

305. Tx 在发数据时，Rx 有 Ack 和流控等 DLLP 在传送，为何此时 Rx 可以处于 L0s？

引文：发送逻辑和接收逻辑可以一个处于 L0，一个处于 L0s，不要求同步转换状态。例如当一个 EP 进行 DMA 写操作时，其发送逻辑 Tx 一直被使用，因此处于 L0 状态，而接收逻辑 Rx 可能长时间没有被使用，从而可以暂时处于 L0s 状态，以降低功耗。（原文引自：https://blog.csdn.net/highman110/article/details/132235892）

这里面的例子举得有问题，Ack Latency，Flow Control DLLP Latency > L0s Entry Latency 的话，对端 Tx（本端 Rx）可以短暂进 L0s。但是一旦收到对端发来的这类 DLLP，本端 Rx 还是要退出 L0s 的。

306. PCIe 5.0 规范支持 x32 吗？NV 的卡似乎支持 x32。

PCIe 5.0 协议支持 x32，但没见用用的。PCIe 6.0 时提到会删除 x32 及 x12。

NV 的 x32 卡并非一条 Link 的 x32，而是两个 x16，支持多主机模式。

307. 有哪个应用场景会用到 x12 吗？

协议是说支持 x12，但实际没见过，也没听说过有用 x12 的。

308. PIPE 接口中 RxElecIdle 拉低可以理解为 FPGA 检测到吗？

不是。Gen1 时 RxElecIdle=1 标志接收端检测到了电气闲 EI，在 Gen2~5 时应根据 MAC 的接收逻辑来判断电气闲。有没有检测到 Rx 要看 RxStatus。MAC 一般有 Test 接口可以看 Tx Detect Rx 结果。

309. EP 寄存器配了 Link Width 后还能自动识别对端有几条 Lane 并做切换吗？

可以的，只要 EP 支持相关 Lane 数，硬件可以自动协商并切换 Link Width。

310. Polling.Active 阶段部分 Lane 上收不到 TS，不应该进 Polling.Configuration 降 Lane 吗？

不是。如果 Detect 电气检测阶段能够检测到 Rx，但 Polling.Active 阶段又收不到 TS，则进 Polling.Compliance。

311. 一个 Port 下可以有多条 Link 吗？

Mindshare 书中讲 LTSSM 时提到，Configuration 状态下 DSP 会根据各个 USP 反馈的 TS1 中的 Link 字段来判断当前 Port 连接了多少 Link，但实际上不存在单 Port 对应多 Link 或者动态拆分为多 Link 的设备，都是固定配置的。即便是 Bifurcation 的情况，也是需要提前配置好，用两个 Controller 各对应一条 Link。

312. Switch 上 4x2 的 Link Merge 为 1x8 后，多个 Link 共享同一 LTSSM 吗？

不是。4x2 是四个独立的 Link，各有一个独立的 LTSSM；1x8 是 4x2 之外的单个独立 Link，只有一个 LTSSM。这里的聚合并非指在空间上把 4 条 x2 链路聚合在一起，而是指单个 x2 转为 x8、四个 x2 分时复用这个 x8 链路。

313. PCIe 启动建链之前，对初始化时间有要求吗？

有要求。PCIe 协议要求 PERST 基础复位完成后 20 ms 内需要进 Detect 开始 Link Training，也就是说复位后 20 ms 内要完成 PCIe PHY 及 Controller 的初始化工作。如果要走 cr_port 更新 PHY Firmware，也需要在复位后 20 ms 内完成。当前 PHY 速率越来越高且都有 Firmware，走 cr_port 更新 Firmware 很难在 20 ms 内完成。此前同行测试了 Intel Gen11 CPU，超过 20 ms 进 Detect 也能成功建链（下附链接）。

最新进展：在 PCIe 6.0 Base Spec 中发现 20 ms 这个参数有更新，基本复位完成之后，5.0 GT/s 及以下速率时要求 LTSSM 在 20 ms 内进 Detect，5.0 GT/s 以上速率时要求 100 ms 内进 Detect。这一点在 PCIe 5.0 Spec 中还未体现。

附：https://blog.csdn.net/linjiasen/article/details/121685493

314. RC 在复位 1 s 后才会认定设备是否 Broken，为何要求链路在 100 ms 内 Link Up？

问题描述不够准确。只有 5.0 GT/s 及以下速率时才才要求 100 ms 内 Link Up，5.0 GT/s 以上没有这么严格。另外，判定设备是否 Broken 不是依据是否 Link Up，而是依据是否正确响应了 Cfg 请求。

根据协议 PCIe Base Spec 第六章，不考虑 RN 机制，目标速率为 5.0 GT/s 及以下速率时，传统复位完成后 RC 软件至少需要等待 100 ms 才能发 Cfg 请求；目标速率为 5.0 GT/s 以上速率时，则是要求在 DL 层 Link Up 后至少等待 100 ms 发 Cfg 请求。这两个 100 ms 计时的起点不同，5.0 GT/s 以上速率建链耗时较久，从复位后开始计时显然是不够的。对设备端而言，则没有 100 ms 的要求，仅要求设备能够在传统复位完成后的 1s 内能够响应 RC 发来的 Cfg 请求。在设备 DL 层 Link Up 之后，设备能够接收 RC 发来的 TLP、DLLP，但其可能还没准备好，需要更多时间才能反馈响应，这个时间限制即复位后的 1s 之内。对 RC 而言，虽然其发出了 Cfg 请求，但只要设备能够在 1s 内响应以 Cpl.SC，就算正常，否则就算是 Broken。

如果设备支持 RN 机制，那么 RC 软件可以不受上述 100 ms 及 1s 的限制，在收到设备发来的 Ready 通知后可以立刻发送 Cfg 请求。这种情况下如果发生了 Cpl 超时，即便复位后不足 1s 也判定设备 Broken。

除了 RN 机制还有一个类似的机制，即 Readiness Time Reporting 扩展能力。如果设备支持 Readiness Time Reporting，在复位前其可以通报给 RC，告诉 RC 自己在复位之后多久时间后能够处理 Cfg。软件可以依据这个提前通报的时间，在复位完毕该时间后直接发送 Cfg，不必等待设备发来的 Ready 信号。

315. Physical Link Up=1 体现在哪，有相关寄存器读取吗？

补充说明：说是 Link Disable 被置 1 后，就会物理层报告 Physical Link Up=0。

可查看 LTSSM State 寄存器是否走到了 L0，另外需要关注链路宽度及速率是否符合预期。

316. LTSSM Polling.Compliance 跳转条件中提到的 Passive Test Load 是什么？

Passive Test Load 是指无源负载，比如 Termination 电阻或者示波器端口，本质都是单端 50 Ω 电阻，能让 DUT Detect 成功。

317. LTSSM 状态机只有一个，但 L0s 状态却分成 Tx、Rx，这是如何做到的？Tx 和 Rx 一个 L0s 一个 L0 时，链路状态算 L0 还是 L0s？

LTSSM 只有一个，L0s 分 Tx 和 Rx 的逻辑不在 LTTSM 大状态机内实现，而是在 LTSSM 外部实现。根据 PCIe Base Spec，Tx L0s 和 Rx L0s 是独立的；根据 Cadence LTSSM L0s

实现逻辑，Tx 处于 L0s 即代表 LTSSM 进入 L0s，然后用另外的 Rx Link State 表示 Rx L0s 的状态机。

318. Alternate Protocol Negotiation 是什么？跟 Modified TS 的关系？

为了扩展 PCIe 设备的适用范围，PCIe 5.0 开始支持 Alternate Protocol。Alternate Protocol 是指运行在 PCIe PHY 上的非标准 PCIe 协议，目前有 CXL 协议。

CXL 运行在 PCIe 5.0 PHY 上，支持 CXL 协议的设备也支持 PCIe 协议。那么到底采用 CXL 还是 PCIe 协议，收发端在进行链路训练的时候可以进行协商，即 Alternate Protocol Negotiation。Alternate Protocol Negotiation 采用 Modified TS1/TS2，跟 Cfg.Lanenum.Wait、Cfg.Lanenum.Accept、Cfg.Complete 并行。

建链失败

319. 使能了 LTSSM 后，LTSSM Status 一直处于 Detect 状态，可能是哪些原因呢？

检查 PHY 的复位是否释放，pclk 是否出来了。

320. 成熟的 EP 设备插在商用服务器上，某些槽位会偶发扫不到设备或者是识别到但掉 Lane，这种情况除了用协议分析仪分析还有何定位手段？

补充：相同 EP 插在不同插槽上没问题，相同插槽接其他网卡也没问题。

只要是插槽，不管插的 CPU、内存还是 PCIe 设备，都可能存在接触问题。此外，还有走线、信号干扰等问题。

服务器是商用的，EP IP 也是成熟的，问题偶发，虽说理论上两端设备存在问题的概率不大，但实际上由于兼容性引发的问题也蛮多的。

建议如下：先判断是 SI 的问题，还是协议的问题。如果和 SI 有关的话，上分析仪没有帮助，先从 EP 的 LTSSM 切换看看有没有异常。如果跟 SI 无关，可以用协议分析仪抓包并进一步分析有何异常。

321. x4 设备 LTSSM 状态一直在 Polling.Active，大概什么原因？

现象描述：RC 这边的 LTSSM 显示一直为 Polling.Active，外设的 LTSSM 目前还没法获取到。

应该不是一直在 Polling.Active，Polling.Active 会超时退出的。这种情况可能是因为链路质量不好，位锁定、符号锁定等一直锁不住。

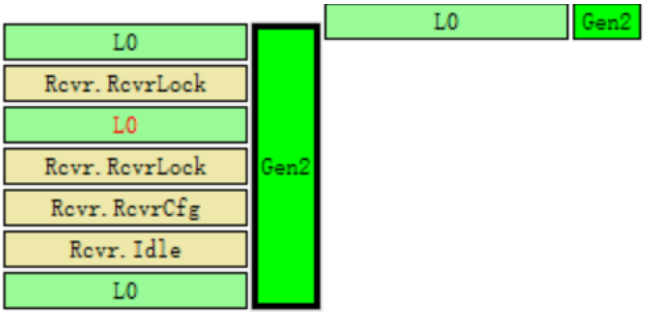
322. Xilinx PHY 的 PIPE 接口输出信号与仿真一致，但是链路状态没有跳转一直处于 Detect 状态，可能是什么原因？

背景：FPGA 平台，Synopsys 公司的 PCIe RC IP + Xilinx 的 FPGA PHY

应该是 FPGA 侧没有 Detect 到 Rx，可确认下 RC 那端是否有 Enable Termination。

323. 链路两端只有一端从 L0 退到 Recovery 又回到 L0，另一端一直在 L0，这正常吗？

问题描述：下图为协议分析仪抓到链路两端的 LTSSM 状态，一端是 L0->Recovery->L0，另一端一直是处于 L0，这个正常吗？我理解两边应同时进入 Recovery、同时进入 L0。



正常。在 L0 被动进 Recovery 的条件中，需要收到 TS1/TS2 或 EIOS 后会进入 Recovery，但前提是当前速率为 8.0 GT/s 及以上。图中 Gen2 速率为 5.0 GT/s，不符合 L0->Recovery 的条件。如果图中为 Gen3，那就不正常了。

324. x4 设备只用 x1，建链时 LTSSM 在 Polling.Active 和 Polling.Compliance 之间反复跳转，可能是什么原因？

进 Compliance 有三种情况：自己主动要求进，对方主动要求进，24 ms 内未检测到 8 个 TS1。如果确认双方均未主动要求进 Compliance，则进 Compliance 为第三条原因。具体可能是：

- ✓ 在 Detect 阶段 Detect 到的部分 Lane 上存在无源负载，导致 Tx Detect Rx，但对端这条 Lane 上实际没有发送 TS。
- ✓ 或者由于链路质量差或者时钟质量差，无法完成锁定，导致无法在 24 ms 内收到指定数量的 TS1。

能够从 Polling.Compliance 跳回 Polling,Active，说明在 Compliance 阶段存在 Lane 上能够收到 TS1 并退出电气闲。

325. RC 直连网卡建链时找不到网卡设备，查看 RC 状态机处于 Polling.Compliance 状态，大概是什么原因？

补充说明：把其他正常设备飞线过去也找不到设备。

进入该状态通常是因为链路信号质量较差，Tx 发送的序列不能被对端正确识别。可尝试在该状态下调整去加重等级或均衡参数，或尝试改为 Gen1 速率。

326. EP 端的 LTSSM 上电后走到 bcdb 后直接就到 0 了，然后最终 0123 一直卡在 3 出不来，有哪些 Debug 思路？

注：0 Idle, 1 Detect, 2 Polling.Active, 3 Polling.Compliance, c Recovery.Equalization, D Recovery.Speed.

看下差分线短路了没，PHY PLL 配好了没，中间是否有插探针。

327. 板卡金手指插在标准服务器上，LTSSM 起来后概率性卡在 Polling.Comliance 状态不动，有什么定位手段？

大概率是连接问题，可以检查下 Tx 和 Rx 的 Lane 连接，也可以换个服务器及板卡做交叉验证。

328. 建链时 EP LTSSM 卡在 Recovery 状态，RC、EP 速率分别为 2.5 GT/s 和 8 GT/s，这可能是什么原因？

问题补充：在华为服务器上存在该问题，Dell 服务器上正常。

EP 处于 8.0 GT/s 速率下的 Recovery 状态，说明链路曾经到达 Gen1 的 L0 状态，RC 的 LTSSM 也曾到达 Gen1 L0。后续观测到 RC 在 2.5 GT/s 下发送 TS1，说明 RC 的确从 L0 退出了，但由于某些因素没有升速到 8.0 GT/s。可查看 RC 当前 LTSSM 是什么状态、是否曾经到达过 Gen3 的 Recovery，如果曾经在 Recovery 下切速到 Gen3 速率，可能是引文链路信号质量问题无法升速；如果没有切速，可能是这个 RC 不支持 Gen3 速率。

329. PCIe EP 设备对接戴尔服务器，Reboot 服务器主机后主机无法扫描到 PCIe EP 设备，可能是什么原因？

补充说明：问题设备为网卡 ASIC 芯片，默认多 PF 且未开启 VF。戴尔服务器主机对接其他 PCIe EP 设备可以扫描成功，当前的 PCIe EP 设备对接其他服务器也可以扫描成功。通过对比实验发现服务器和 PCIe EP 设备本身都是正常的。在扫描失败场景中，通过读寄存器发现 LTSSM 状态机从来没有进入过 L0 状态（包括 Gen1 L0），也没收到过 DL 层的 NAK 和 ACK。

如果怀疑是建链问题或信号问题，可尝试一下手段：

- ✓ 可以降速到 Gen1 重新尝试。
- ✓ 可排查 RC 和 EP 的 Link Width 是否匹配，如果 RC 的 Link Width 更宽，可以排查下 Lane Reversal 相关。

✓ 检查一下 CDR LOCK 问题，是不是 Skew 设计的 Margin 不够，挑设备。

如果跟建链无关，则可以 Rescan 看能否扫描到，也可能是 BIOS 等待时间太短了。

此外，戴尔服务器有设备认证问题，其 BIOS 自检时会认证设备，只有戴尔认证过的才能被扫描到，自检结果可以在 BIOS 下看到。如果确认为认证问题，可以把 PCIe 认证选项关闭重试。不过，认证的前提是 Link Training 成功，如果 LTSSM 状态从未到过 L0，应该跟这个认证机制无关。

330. Target Speed 为 Gen3 速率，建链时 LTSSM 反复从切速后的 Recovery.RcvrLock 掉到 Detect，该怎么分析呢？

可以先把 Target Speed 设置为 Gen1 速率，确保启动 Sequence 没有问题，然后再设置 Target Speed 为 Gen3。如果 Gen1 没问题，那大概率链路质量差，切速到 Gen3 后没锁住。可以排查初始 Preset 是否为一个比较恶劣的值。如果接了分析仪，除了看 LTSSM 状态机，还可以看 Event View 里的数据进一步分析。

链路非预期（掉链/掉宽/掉速）

331. 建链时掉 Lane，一部分检测到对端、令一部分没有检测到对端，可能是什么问题？

猜测是 Detect 信号交互存在问题：

✓ Detect 信号有问题，保持时间上升时间之类不符合要求；

✓ EP 过早发送 Detect 信号，导致没有检测到对端，状态机就着急进入下个阶段。

解决方案：让 CPU 的 pcie_reset 信号延时后给到 EP，让 EP 晚于 CPU 发送 Detect 信号，从而让两者正常交互。

332. L0p 状态下，不工作的 Lane 失锁了怎么办？直接上调 Lane 会有问题吗？

L0p 状态下激活休眠状态的 Lane 是有一个协商过程的，即需要在待激活的 Lane 上互发 TS，在此期间重新锁定。如果 24 ms 内无法激活，则则上调失败，需要进 Recovery。

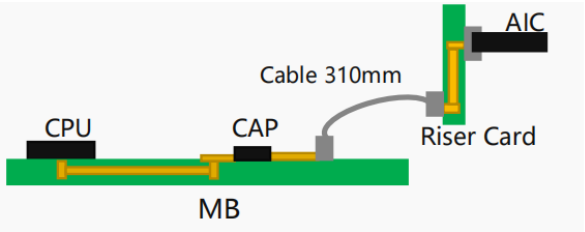
333. 有个设备在 Gen2 时可以找到，Gen1 有一定概率找不到，这是什么原因？不应该是默认先到 Gen1 再协商到更高速率吗？

问题补充：Gen1 是在桥上直接硬件配置的，即 PowerUp 后直接强制进 Gen1，而非 Gen2 切过去的。

是先到 Gen1 再到更高的速率。Gen1 对链路质量要求较低，甚至飞线都能成功建链。如果是强制进 Gen1 仍建链失败，建议读 LTSSM Track 查看中间经历了哪些状态再做进一步判断分析。

334. 为什么有的物理通路信号质量很差，该如何提升物理信号质量？

测试场景为：RC 是服务器 CPU，非 PC 电脑，服务器裸板上插上测试 PCIe 设备，然后测试信号。物理通路信号质量差体现为：示波器跑出来的眼图信号都是 Pass 的，但是 Message Log 会出现一些可纠正错误（CE），还有降速掉带宽问题。下图为测试链路示意图。



可以做 Loopback JTOL，即用 BERT 仪做 Rx BERT 测试。如果 Rx BERT 测试通过，难以说明是信号质量问题；单纯的 Rx BERT 测试通过，也难以说明信号质量不存在问题。如果的确是信号质量问题，可以找 SI 工程师调 Serdes，降低链路损耗。->测试结果如下：BERT 测试中 Rx Bert 有一定误码率，但测试通过。

可尝试接分析仪复现问题。单测和协议分析仪场景有点小差异，尽量用分析仪现场抓下，看看是哪一层出的问题。->用协议分析仪测试结果如下：

- ✓ 未发现有 8 B10 B, 128/130 编码错误；
- ✓ 产品为 Gen4，LTSSM 未训练到 Gen4 速率，掉速到 Gen3；
- ✓ 时有抓出 ECRC 和 LCRC 错误，设备侧（SSD 卡）日志报 Rx Err、Bad TLP、Bad DLLP；
- ✓ TLP 中存在 CA（Completion abort），CA 由 RC 端发出。

可尝试交叉测试。根据上述协议分析仪结果，没有直接证据证明 SSD 侧收错，更像是 DSP 发错了。考虑到有的外设会有问题，所以要交叉测试，换几种厂商的设备侧，以排除外设问题。之前服务器用了一家厂商的盘也会出现 LCRC 的错误；当时是 Retimer 卡和 SSD 之间产生了 LCRC Error。若不同外设都有链路问题则排除外设问题。

可评估下 CPU-AIC 之间的插损。->评估下来插损裕量充足。

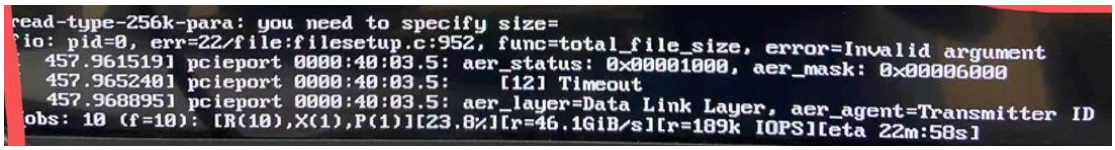
插 LeCory 分析仪掉速比较正常，LeCory 抓错包一般对分析链路质量问题作用不大。LeCory 5.0 的分析仪的 Interposer 卡做的不够好、信号衰减严重，可以用他们 6.0 的 Interposer。原环境不要插分析仪，读数字眼图，和 Rx Err 的 DFX 一起做进一步分析。

335. 链路两边的 Lane 数不一样（DSP x4，USP x2），但还是都在 L0 进行数据报传送，没有进入 Recovery，可能是什么原因？

首先，可能是你看错了，不应该出现这种情况；其次，可能是 Downstream 开了 x4，建链协商结果实际只有 x2。

336. RC 与 EP 间通信，初期正常，为什么后期只有 RC 端接收、没有 RC 端发送呢？

问题补充：使用的是服务器，x4 卡槽，在 Riser 上接 Retimer 卡进行测试。状态机正常，双方处于 L0 状态。此间多次出现 L0 退到 Recovery，系统显示 Completion Timeout。RC 发送的最后一条 Ack 报文后又占用了 2.002sec，RC 收到 EIEOS 后进入 Recovery。



猜测是 EP 反复收到 RC 发来的包 Bad TLP，EP 先进 Recovery 发 EIEOS，带动 RC 进 Recovery。

337. PCIe EP 偶尔会降 Lane，眼图看着正常，有何好的 Debug 思路吗？

可能是均衡问题。可以查看 LTSSM 过往状态进一步排查。

可能有 Error。可以查看相关错误状态寄存器进一步排查。

可能出于共和考虑，Power 太大自动降 Link Width。

338. 2.5 GT/s 速率下，在枚举阶段 PCIe 位宽由 x4 降到了 x2，一般是什么原因？

可能是信号质量差导致的，但 2.5 GT/s 下信号不至于这么差，可尝试换个 Host 或者换个插槽看现象是否一样，或者看下 dmesg log 有没有相关提示信息。

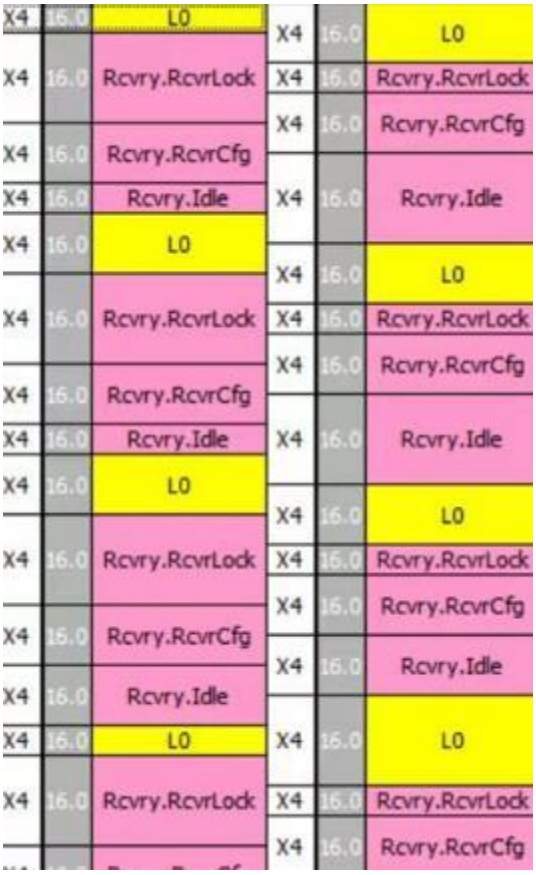
339. Gen1 x4 链路建链偶发掉 Lane 到 x2，失效的 Lane 在 Polling 24 ms 内不能 Lock，确认信号质量没问题，可能是什么原因？

根因：Firmware 配置 Sequence 问题，Serdes Rx 信号检测使能位在 PIPE_reset 释放之后晚了一点点才 set 上。

340. PCIe 链路两端均具备 x8 能力但建链后为 x1，有什么定位手段？

简单快捷的方式是祭出橡皮大法——拿橡皮擦一擦金手指，重新插接看能否建链成 x8。如果不成功，可以查看相关状态寄存器，确认是从 x8 掉到了 x1 还是上来就是 x1。如果是从 x8 掉到了 x1，可能是存在 Lane 上有信号质量问题；如果上来就 x1，可能存在部分 Lane 未 Detect 到对端的情况。如果是信号质量问题，则需要调信号质量；如果是未 Detect 到，可以拿万用表量一下是否线断开了，或者是否存在电容虚焊。

341. 在未出错的前提下链路会自动从 L0 状态转到 Recovery 状态，该情况正常吗?



是正常的。如果长时间没有数据，会发 EIEOS 到 Recovery。

342. PCIe 建链成功后，链路流量大时每隔几十秒报一两个 Receiver Error、Bad DLLP、Bad TLP，不打流或者流量小时没问题，有何定位思路？

现象补充：商用服务器接自己的 EP 芯片（非 Synopsys 公司 PHY），EP 接受方向相同流量下 Gen1、Gen2 速率没问题，Gen3、Gen4 时出现上述报错，误码比较少，仅几十秒报一两个错，链路一直正常稳定在 Gen3/4 没有跳回 Gen1。大流量时所有 Lane 都有问题，非固定 Lane，且 Gen4 x1 时同样存在问题。测试环境温度不高。

提议 1：可尝试换服务器或 EP 设备，确认是个体现象还是普遍现象。

反馈 1：换同类型服务器、换 EP 芯片、换芯片板子，上述问题现象均一样。

提议 2：除了切换同类型的服务器及芯片，也可以切换不同型号的服务器及芯片，排除是否兼容性问题。

提议 3：大概率是信号质量问题，如果怀疑和 SI 有关，可以看一下 PHY 内部眼图，还有进出 Recovery 的频率。如果跟信号质量有关，可尝试链路均衡参数调优，包括 EP Rx 的 CTLE 参数、对端 Tx 的 Preset 参数。Gen4 时曾遇到类似问题，FPGA 上的 EP PRBS 测试没问题，后来发现 PRBS 测试和真实使用的 PCIe 环境参数不一致，后来调了 PCIe 环境的 CTLE AC Gain 参数解决的。另外，如果 EP Rx Serdes 的 CDR 带宽可调，可以把 CDR 带宽调小，以更大限度地抑制抖动，其他针对低频噪声的参数、针对 Vendor 开放的 CDR 参数，能调的都可以调节试试。

反馈 3：用单 PMA 外部环回，PRBS31 测试没有错误。Gen4 内部眼图看着没问题，但眼图扫描时间短，有可能扫不到问题点。<-回环测试没参考性，真实场景的内眼扫描有一定参考价值。

提议 4：排查隔直电容大小是否符合标准。该电容在服务器发送端及设备接收端都有。

提议 5：可查看 RAS Debug Feature 相关寄存器，比如信道插损、均衡 FOM 值、Elastic Buffer 是否 Overflow。如有异常可尝试调整 SKP 发送频率。

反馈 5：Deskew 未报异常，Elastic Buffer Overflow 情况待确认。

提议 6：从电源角度考虑，流量大会导致功耗大、IR 压降增大，可尝试抬高供电电压，或者简单尝试上下调动供电电压排查电压有无影响；另外，可排查电源噪声情况，是否在流量大时电源噪声过大。Gen2 和 Gen4 的链路 Margin 不一样，对噪声和抖动的容忍度也会不同。

反馈 6：抬电压有点影响，Receiver Error 相关的 128b/130b Frame Error 未再报出，但其他原因导致的 Rx Error 仍然存在。电压已上调至极限，无法继续调节了。

343. 在 x1 Gen1 下链路训练没问题，提升速率或 Lane 数后链路训练失败，是信号质量问题吗？

大概率是。但仍需提供更多细节进行分析。

344. 对设备做 Reboot 后链路掉速，可能是什么原因？

现象描述：飞腾服务器过 PLX 的 Switch（PEX8748）接自研网卡，在 Gen3 建链成功后做 Reboot，再次建链成功后降速到了 Gen1，此间未看到任何错误。手动配置 Switch 的 DSP 做 Retrain 又能够恢复到 Gen3。插别家卡未出现该问题。

如果是 PLX 87XX 系列的 Switch，有可能导致均衡失败，可查看 EP 的 Log 是否存在均衡未正常完成的情况。PLX 87XX 的 Switch 在均衡过程中发送的 TS 有可能 Parity 是计算有问题的，导致均衡没法走下去，难以从 EP 端解决该问题。

链路操作

345. Link Up 之后如何发起更改链路宽度，有寄存器触发吗？

不在 PCIe Spec 规定的范围，厂商自己搞。

346. smlh_link_up 拉高之后需要如何配置才能让 rdlh_link_up 信号也拉高？

注：Synopsys 公司的 PCIe 控制器。

不用特意配。LTSSM 走到 Gen1 速率 L0 后 smlh_link_up 就能起来了，rdlh_link_up 需要 DL 层 DLCMSM 状态机走到 DL_Active 才能起来，期间做 DL_Feature 和流控初始信用量相关协商。如果 smlh_link_up 起来了但 rdlh_link_up 迟迟起不来，可看下 LTSSM 是否最终落在 L0 了，是否 DL 相关协商卡住了。

347. 支持 Lane Reversal 的 PCIe x16 板卡插接到不支持 Lane Reversal 的服务器 x16 插槽上，该插槽实际只有 x8 工作，可以建链成功吗？

Link 两端只要一端支持 Lane Reversal 就可以。

348. PCIe Link 先建链到 x4 再改 EP 寄存器降 Lane 到 x1 是可以的，但先建链到 x1 L0 再升 Lane 到 x4 失败，是什么原因？

不回 Detect 切 Lane 数（不考虑 L0p），降 Lane 可以直接降，升 Lane 的需要 Link 两端的 RC 和 EP 均支持 Upconfigure Capability。

Resizing (up or down) only occurs if both Link partners advertised the upconfigure Capability bit. In cases of link reliability, the PCIe specification permits the remote partner to initiate downsizing regardless of the value of this bit.

349. 可以直接写配置空间 Link Control Register bit5 触发 Retrain 吗？感觉未生效。

写入 Link Control Register 的 Retrain 之后会进入 Recovery，然后自动进入 L0 并清除该 bit，但从这个寄存器看不出是否发生了 Retrain。

可尝试改写 Link Control 2 Register(Offset 30h)的 Target Link Speed 为一个不同于当前值，然后写 Link Control Register 的 retrain bit，根据最终的 Link 速率或 Link Bandwidth Management Statues 来判断是否发生了 Retrain。

350. 在 EP Rx Term 存在的情况下，RC 该如何忽略其中未使用的 Lane?

问题详述：RC 配置为 4 Lane，EP 是 4 Lane 金手指但实际只有 1 Lane 工作。由于 EP 的 4 条 Lane 均存在 Rx Term（Rx PN 之间有 100 Ohm 差分阻抗），导致 RC Detect 到 4 Lane, LTSSM 在 Polling.Active 和 Polling.Compliance 反复跳转。在 RC 端实时修改 Link Width 到 x1 重新训练未解决问题；修改镜像代码中的 Link Width 为 x1 重新编译后烧录到 RC 镜像，可以 Link Up。该 EP 在 x86 的 x16 插槽中可以正常识别为 x1。

可以在启动 Link Training 之前人为把 EP 的 Link Capable 寄存器中的 Link Width 配置为 x1。如果无法配置 EP，可以尝试拿掉未使用 Lane 上的电阻，也能使 Rx Detect 失效。实在不行，可以用薄层绝缘胶带把金手指上未使用的 Lane 贴上，进行物理屏蔽（此即胶带大法）。

链路训练仿真

351. DUT-VIP 通过 PIPE 对接，仿真时报错检测到 Unknow Order Set，该如何定位问题？

排查发生错误时 LTSSM 是否过了 Gen1 L0 了：

- ✓ 如果过了 Gen1 L0 在均衡阶段 VIP 检测到 Bad OS，可能是 VIP Clock Tolerance 设置太严格了导致锁不住。
- ✓ 如果没过 Gen1 L0，确认 Symbol Lock 状态下有没有锁住，Link 两端的 Speed 速率是否一致。

352. DUT-VIP 快速仿真模式下，过了 Gen1 L0 后在均衡阶段报超时，是哪里设置不对吗？

如需在加速模式下进行仿真，VIP 和 DUT 都要开 Fast Link Mode，且加速倍数应合理设置，不是越快越好。

排查时，可根据仿真日志初步确认是谁报的超时，如果是 VIP 先报超时，确认是否 VIP 的 EQ Timeout Limit 设置过小；如果是 DUT 先报超时，确认是否 fastsim 加速倍数设置过大，导致 Lock 不住。

如果以上排查无误，可继续排查：对于带 PHY Model 的 Serial Connection 仿真，PHY Model 要添加 Fast Sim 相关的 Macro，必要时可以选用 Fastsim 模式的 PHY Firmware，并确认 Firmware 有没有方法能把能关的中断关掉、有没有 Macro 支持 Force Calibration Done =1。

353. RC DUT 向 EP VIP 发起读写操作，数据无法写入，可能是什么原因？

可能是没落在 EP BAR 范围内。

354. Synopsys 公司的 VIP 如何配置有序集的 Disable 字段让 Link 进入 Disable 状态？

可以调 VIP 的 PL Service 的 set_phy_en_sequence 这个 Sequence 让 VIP 进 Disable 状态。具体可以参考 Solvnet 上的这篇文章：

https://solvnetplus.synopsys.com/s/article/PCIe-UVM-VMT-An-Example-Showing-How-to-Bring-the-Link-to-a-DISABLE-State-and-Then-Back-to-DETECT-1576070074545

355. 仿真时，RC 还没有检测到 Rx 端口变化就从 Detect.Quite 跳到 Detect.Active 状态了，可能是什么原因？

从 Detect.Quite 跳到 Detect.Active 有两个条件，12 ms 超时或者退出 Electric Idle。如果没有退出 Electric Idle 且仿真时间未到 12 ms，可能是仿真 Speed Up 加速太快提前超时了。

356. 采用 Synopsys IP+VIP 仿真，DUT 侧的 iip_chip 层做了替换，波形显示 DUT 的 LTSSM 走到了 L0，smlh_link_up 起来了但 rdlh_link_up 没起来，可能是什么原因导致的？

查看 VIP 吐出的 Transaction Log 中 DUT 跟 VIP 之间有无沟通 InitFC。如果 DUT 没发 InitFC，可排查 DUT 的解复位是否正确；如果是 VIP 没法，可排查 VIP dl_cfg 是否开了 InitFC 等配置。

357. Synopsys PCIe IP 做 EDA 仿真时，Gen1、Gen2 的 Link 正常，Gen3 时 EP 在 Phase 0 出现 12 ms 超时，对于 Gen3 是否有其他寄存器需要配置？

补充：仿真时带 PHY，对端接的 Synopsys VIP，VIP 配置了 Fast Sim。

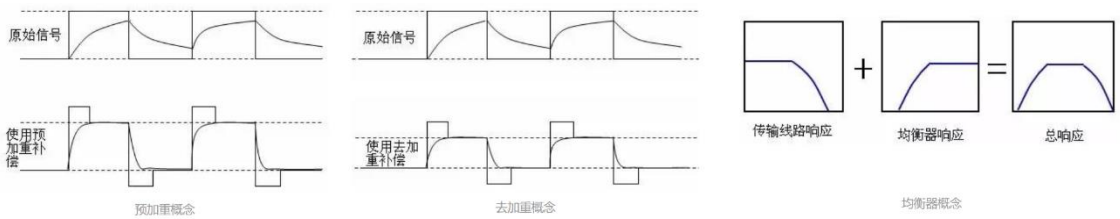
首先，VIP 和 IP 都需要开启加速模式；其次，开启加速后，加速倍数需要合理设置，加速倍数太大的确可能导致在均衡成功之前过早超时退出。建议排查 DUT 的仿真加速倍数是否设置合理。正常加速模式都加速 10~1000 倍或者更大加速倍数，很少有加速 10 倍以内的。

均衡

均衡过程理解

358. PCIe 采用了哪些主要的信号补偿技术？

信号补偿三大技术：发送端预加重（Pre-emphasis）、去加重（De-emphasis）和接收端均衡（Equalization）。预加重、去加重也被称为发送端均衡。各技术基本原理如图所示。



▲图： 预加重、去加重、接收端均衡原理

无源传输线路像是一个低通滤波器，PCIe 高速串行信号经信道从发送端传输到接收端后，其高频分量比低频分量衰减要大，而高频分量主要集中在信号的上升沿及下降沿。为了补偿这种高频衰减，在信号发送的时候，有意增强信号跳变沿的信号幅度，增高

高频分量，即信号预加重；相比于预加重的方案，去加重则是降低跳变沿之外的信号幅度，削弱低频分量，同样能够达到目的。

接收端均衡器相当于高通滤波器，来补偿失真的波形。

359. 均衡阶段交互 TS，TS 中的 P0 ~ P10 Preset 有什么用？

简单来说，尝试不同的 Preset 找到信号质量最好的那个 Preset，然后使用。

P0 ~ P10 共 11 种可能的预设以及它们对应的系数值和电压比率为 Tx 均衡提供了一个粗调的起点。如果使用预设的信号达到了期望的 10^-12 误差率，则不需要进一步训练。如果测量到的误差率过高，则使用均衡序列通过尝试不同的 C_{-1} 和 C_{+1} 值来微调系数设置，重复该序列直到达到所需的信号质量或误差率。

360. 均衡参数中的 Preset 和 Cursor 有具体的数量对应关系吗？

公式如下，可对照协议上 PSET 表根据计算公式推算一遍。推算完后再把 MAC 和 PHY 之间这几个 value 的交互以及指示信号看一下，理解其具体使用。

$$\text{De-emphasis} = 20\log_{10} Vb/Va$$
$$\text{Preshoot} = 20\log_{10} Vc/Vb$$
$$\text{Boost} = 20\log_{10} Vd/Vb$$

Table 8-1 Tx Preset Ratios and Corresponding Coefficient Values

Preset #	Preshoot (dB)	De-emphasis (dB)	c_{-1}	c_{+1}	Va/Vd	Vb/Vd	Vc/Vd
P4	0.0	0.0	0.000	0.000	1.000	1.000	1.000
P1	0.0	-3.5 ± 1 dB	0.000	-0.167	1.000	0.668	0.668
P0	0.0	-6.0 ± 1.5 dB	0.000	-0.250	1.000	0.500	0.500
P9	3.5 ± 1 dB	0.0	-0.166	0.000	0.668	0.668	1.000
P8	3.5 ± 1 dB	-3.5 ± 1 dB	-0.125	-0.125	0.750	0.500	0.750
P7	3.5 ± 1 dB	-6.0 ± 1.5 dB	-0.100	-0.200	0.800	0.400	0.600
P5	1.9 ± 1 dB	0.0	-0.100	0.000	0.800	0.800	1.000
P6	2.5 ± 1 dB	0.0	-0.125	0.000	0.750	0.750	1.000

361. LTSSM 均衡 Phase 0 ~ Phase3 都在干什么？

PCIe LTSSM 链路均衡，即 EQ (Equalization)，是 LTSSM Recovery 的一个子状态。PCIe 在首次进入 8 GT/s 及以上速率时要进行均衡，调整收发端电气参数以改善信号质量，满足高速信号传输的信号质量需求。

均衡有 phase0 ~ phase3 四个阶段。TS1 或 TS2 symbol 6 中包含 EC 字段来指示当前处于均衡的哪个 Phase。各 Phase 工作内容如下：

- ✓ EQ Phase 0：EC=0。USP 回传 preset 及 coefficients 给 DSP。DSP 没有 Phase 0。

- ✓ EQ Phase 1：EC=1。DSP 及 USP 发送其 LF (Low Frequency，symbol7)、FS (Full Swing，symbol8) 、Post-cursor (symbol9) 给对端 ，在 Phase 2/3 使用。
- ✓ EQ Phase 2：EC=2。USP 作为 Master，调节 DSP 的 Tx 系数。具体可分为 preset 调节及 coefficient 调节。
- ✓ EQ Phase 3：EC=3。DSP 作为 Master，调节 USP 的 Tx 系数。同 phase2。

以上四个 Phase 中，Phase 0/1 采用 Preset 粗调，Phase 2/3 精调。如果粗调阶段就达到了信号质量需求，也可以不进行精调。

除非有特别配置，8 GT/s 以上速率时必须进行均衡，至少在最高速率上要进行均衡，中间速率可以跳过均衡。当然仿真的时候可以设置 bypass_eq，或者设置 bypass_eq_to_highest_rate，仅在最高速率时进行均衡。比如最高支持 32 GT/s，那么在 8 GT/s，16 GT/s 的时候可以跳过均衡。从 32 GT/s 降速时，需要重新进行链路训练并做均衡。

362. Pre-cursor、Cursor 和 Post-cursor 符合这个表。就像这样，是否能得出下表呢？下表里面有的值没有对。

a) $|C_{-1}| <= \text{Floor}(FS/4)$
b) $|C_{-1}|+C_0+|C_{+1}| = FS$ (Do not allow peak power to change with adaptation.)
c) $C_0-|C_{-1}|-|C_{+1}| >= LF$

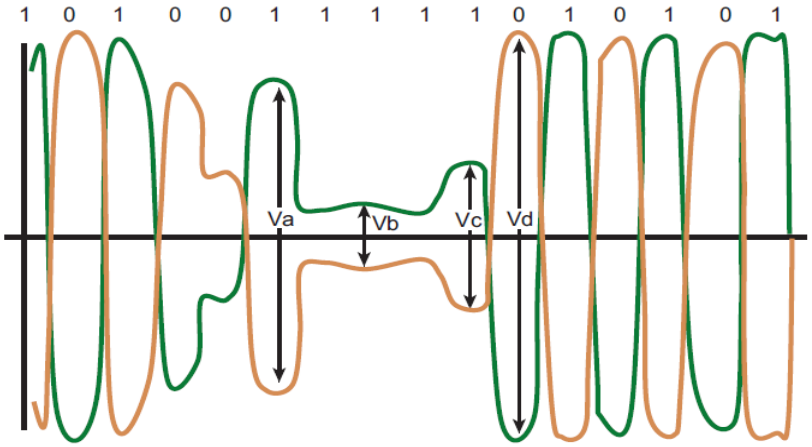
Preset #	Preshoot (dB)	De-emphasis (dB)	c-1	c+1
P4	0.0	0.0	0.000	0.000
P1	0.0	-3.5 ± 1 dB	0.000	-0.167
P0	0.0	-6.0 ± 1.5 dB	0.000	-0.250
P9	3.5 ± 1 dB	0.0	-0.166	0.000
P8	3.5 ± 1 dB	-3.5 ± 1 dB	-0.125	-0.125
P7	3.5 ± 1 dB	-6.0 ± 1.5 dB	-0.100	-0.200
P5	1.9 ± 1 dB	0.0	-0.100	0.000
P6	2.5 ± 1 dB	0.0	-0.125	0.000
P3	0.0	-2.5 ± 1 dB	0.000	-0.125
P2	0.0	-4.4 ± 1.5 dB	0.000	-0.200
P10	0.0	Note 2.	0.000	Note 2.

```
Table"
preset      0   1   2   3   4   5   6   7   8   9"
pre_cursor  0   0   0   0   0   2   3   5   3   4"
cursor      18  20  19  21  24  22  21  34  18  20"
post_cursor 6   4   5   3   0   0   0   9   3   0"
```

0+18+6=24，0+20+4=24。不一定得出这个表，但需要再这个表的配置下，得到的 dB 值符合协议。即归一化之后，能对应上$|C_{+1}| + |C_{-1}| + C_0 = 1$。

363. FIR 均衡器有哪几种输出电平？

采用三抽头 FIR 均衡器时，对于某比特信号而言，其前后两比特有 4 种不同的组合形式，这也意味着该比特输出电平有 4 种可能，如图所示。电平幅值取决于三个抽头系数 C_{+1}、C_0 及 C_{-1}。



▲图： 三抽头均衡器的四种输出电平

举个例子：当前比特为 1，C_{+1}=-0.2，C_0=0.7，C_{-1}=-0.1,

- ✓ 连续 3 比特为 011，输出电压为 $-C_{+1}+C_0+C_{-1}$=0.8，对应图中 Va，称为 **Normal 电平**。
- ✓ 连续 3 比特为 111，输出电压为 $C_{+1}+C_0+C_{-1}$=0.4，对应图中 Vb，称为 **De-emphasized 电平**。
- ✓ 连续 3 比特为 110，输出电压为 $C_{+1}+C_0-C_{-1}$=0.6，对应图中 Vc，称为 **Pre-shoot 电平**。
- ✓ 连续 3 比特为 010，输出电压为 $-C_{+1}+C_0-C_{-1}$=1.0，对应图中 Vd，称为 **Maximum-height 电平**。

根据以上四种电平，可计算出去加重、Pre-shoot 及 Boost 幅度，表示为：

$$De-emphasis = 20\log_{10} V_b/V_a$$

$$Preshoot = 20\log_{10} V_c/V_b$$

$$Boost = 20\log_{10} V_d/V_b$$

364. 上文中，当前 bit 是 1，011 的组合时为何就是 Normal 电平，111 就是 De-emphasized 电平。如果当前 bit 是 x，那么 0x1 就是 normal，1x1 就是 De-emphasized 电平，1x0 就是 Pre-shoot 电平，0x0 就是 maximum-height 电平，是这么规定的吗？

这是概念、规定。概念解释可查看下图。

PCIe 3.0 Term	Corresponding Condition on Input Signal
Flat Level	A constant voltage will appear when bits of same polarity are being transmitted (Refer to Vb in Fig. 2)
Pre-shoot	A boost appears just before the polarity inversion (Refer to Vc in Fig. 2)
Maximum Boost	A major boost appears when there is polarity inversion only for one bit interval (Refer to Vd in Fig. 2)
De-emphasis	A boost appears just after the polarity inversion (Refer to Va in Fig. 2)

365. Rx 的均衡参数不是自动调整的吗？在初期寄存器配的均衡参数有用吗？是否只做初始参数用，最后使用的还是系统自己调整优化后的均衡参数？

有两种模式：自适应和 Override 模式。如果要用寄存器配置的参数，可关闭自适应调整。

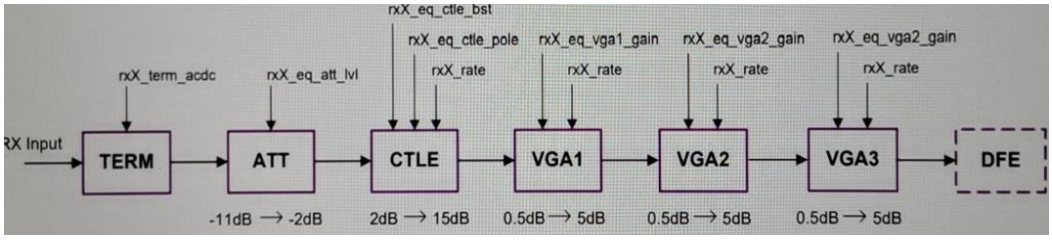
366. 协议里提到的 start_equalization_w_preset 这个字段是做什么的？

这里的 w 是指 with，字面解释为启动基于 preset 的均衡，是 LTSSM 从 Recovery.RcvrLock 到 Recovery.Equalization 跳转的一个条件/信号，不是用于测试模式的。深层次来看，这个信号是跟信号完整性有关。

如果需要做均衡，该信号必须实现。

367. PCIe Rx AFE 的参数是动态均衡 Phase 2、Phase 3 期间调整的吗？

注：AFE，Analog Front End，模拟前端。



各家做法不一，一般不调。

Rx 均衡参数可以通过 Rx Preset Hint 进行调整，但是 Synopsys PCIe IP 不建议用这个参数。Rx Preset Hint 只在 Gen3 定义为 Optional，到 Gen4 及以后没有了。Rx 均衡可以理解为 Tx Preset 固定时 Rx 端 CTLE、DFE 的自适应均衡。Phase 2、Phase 3 时看起来是

在调 Tx Preset 等 Tx 参数，但本质上是在 Rx 做均衡收敛、均衡效果评估的基础上对 Tx 所做出的调整。

368. PCIe Adaptive Rx EQ 在 Phase 1 前还是 Phase 1 后？

Rx Adaptive Eq 属于精调，在 Phase 1 之后。具体看是 Upstream 还是 Downstream，一个是 Phase 2，一个是 Phase 3。

369. Phase 0/1 阶段不是已经做好 Preset 评估了吗，为什么在 Phase 2/3 还要做 Preset 评估？

Phase 0/1 是粗调，误码率要求 10e-4，能够正确接收 TS 即可；Phase 2/3 是精调，误码率要求 10e-12。调完 Preset 还要调 Coefficient。

370. PCIe Gen5_Eq_Control_Register 中的 USP/DSP Tx Preset 是自己设置的还是对端设置的？

对端设置的。Tx Preset 初值是自己设置的，但基于该值不保证误码率达到要求，一般需要对端来调。在 EQ Phase 2/3 阶段，上下游的 Rx 根据均衡评估结果互相调整对方的 Tx Preset，是对方要求的且不能拒绝。

371. 11 个 Preset 都是针对 Tx 设置的参数吗？为什么 Rx 没有 Preset 值？

P0~P10 这 11 个 Preset 都是对 Tx 而言的，对应发送信号不同的 Pre-Shoot 和 Deem-Phasis。Rx 也有 Preset，即 TS 中 Rx Preset Hint，但其仅限于 8 GT/s 速率，且是可选的。

高速 SerDes 最难、最核心的就是两个：Adaptive Rx EQ 和 CDR，Tx 难度不大。如此前所述，Phase 2、Phase 3 时看起来是在调 Tx Preset 等 Tx 参数，但本质上是 Rx 在均衡效果评估之后对 Tx 做出的调整。

Rx 是基于 CTLE+DFE 的，PCIe Spec 只是定义最低要求，具体实现也不要求和 Spec 完全一致。不同厂家的 PCIe PHY（PMA） IP 的 CTLE/DFE 特性并非完全相同，PCIe 并没有定义 Adaptive Rx EQ 需要怎么实现，主要靠 Vendor 自己发挥。就比如 DFE 的 Tap 数，Spec 定义用 1 Tap 的，实际实现可能都是 5 个 Tap 起。换一句话说，不同 Vendor 的 PCIe PHY 的 Rx 能力还是有区别的，有的做得好，有的可能差一点。具体表现为，同样质量的信号，有的 PHY 可以非常稳定地接收，有的 PHY 就不行。

372. 均衡状态下 TS Symbol 6 的 PRBS Pattern 是做什么的？配置为 0 和 1 的作用是什么？

TS 中没有 PRBS Pattern 这个字段。

373. PCIe 3.0 做均衡需要发 PRBS 吗？

PCIe 均衡不发 PRBS Pattern，发 TS1/TS2。

374. 如果链路均衡时，DSP 没有给 USP 传 Preset 值，USP 用的是自己 Equalization Control 寄存器里的 Preset 值吗？

协议未定义具体规则，厂家自行实现。

375. PCIe Gen3 Tx 均衡参数是在 Recovery 阶段确定的吗？

对于 5 GT/s 及以下速率不需要协商均衡参数，为固定的经验值。对于 8 GT/s 及以上速率，Tx 均衡参数是在 Recovery.Equalization 阶段协商确定的。

简单来说，均衡参数协商机制就是 Tx 端用预设好的 Preset 调节 Tx 的均衡，然后对端的 Rx PHY 会做评估并给出 FOM，即该组 Preset 好坏的数值，最后根据不同 Preset 的 FOM 结果综合来看哪个 Preset 最好，就是反过来告诉 Tx 用这个 FOM 结果最优的 Preset。该方向调整好之后，然后 Link 两端再互换角色进行另一个方向的均衡。

376. 如果均衡在 Phase 1 失败了，还能通过修改配置完成建链吗？

正常情况下不能。如果怀疑是 Rx 有问题，在 Phase 1 超时之前 Rx 内部偷偷做一次 Rx Reset，或许能绕过去。

377. 如果指定选择均衡到 Phase 1，做完 Phase 1 误码率高会怎么处理？

如果是硬件均衡，Phase 1 做完后误码率不达标会继续做后面的 Phase；如果是软件均衡且强制只做到 Phase 1 的话，Phase 1 做完后误码率不达标会重新做均衡。

378. 做均衡时只能选择到 Phase 1 或 Phase 3 吗，不能到 Phase 2 吗？

不可以。Phase 2、Phase 3 是一对，Phase 2 是由 USP 的 Rx 来微调 DSP 的 Tx，Phase 3 是由 DSP Rx 来微调 USP 的 Tx。

379. 什么时候可以不做均衡的 Phase 2、Phase 3？

在 Phase 1 传递 Tx Preset 值，做完 Phase 1 后误码率一般较高，需要 Phase 2 和 Phase 3 来进行微调。如果 Phase 1 阶段就达到了误码率要求，可以在 Phase 1 结束均衡直接进 Recovery.RcvrLock。如果打算跳过 Phase 2、Phase 3，可以在此前建链均衡结果基础上手动选择一个比较好的初始 Preset。

380. Recovery.Equalization 阶段确认 Tx Preset 起始是固定从 P0 开始扫描还是可以自定义？

可以设置 MAC（Controller）里的寄存器选择起始 Preset 的选取值，不需要连续。Synopsys 公司这个寄存器为 GEN3_EQ_PSET_INDEX_OFF。

381. Rx PHY 在调 Tx Preset FFE 时如何得到 FOM 的？需要在一次接收过程中画出整个眼图吗？

FOM 应该是在 PMA 里实现的，看不到实现方法。计算 FOM 的时间最多 2 ms，这么短时间扫描整个 Eye 不太够，可能只是粗粒度测几个点。

382. Recovery.Equalization 阶段的时间要求是协议固定死的还是可以通过寄存器来调整？

超时时间协议固定死了，一共 24 ms。当然可能有后门寄存器可以改这个超时时间，但这样是不符合协议的。

383. 在 PERST 释放之前可以 Override PHY 内的 Tx FIR 寄存器来修改 Tx Deemphasis 吗？

问题详述：PCIe Gen1 的 Tx Deemphasis 幅度是由 MAC 通过 PIPE 接口信号管脚下发到 PHY 进行配置的，一般 Gen1 硬连接为 3.5dB、Gen2 硬连接为 -6dB 或-3.5dB、Gen3 及以上 LT 的 Recovery 状态均衡出合适的 Preset 值接管。现在的问题是，假设 Gen1 想把 Tx Deemphasis 从-3.5dB 改成 0dB，通过 MAC 发给 PHY 的路径是不行了，是否可以在 PERST 释放之前 Override PHY 的 Tx FIR 寄存器来完成这个事情？

可以这样做，没有问题。经过一段 Channel 后，对端无法得治用的 Deemphasis 是多少了。不过，除非 Channel 损耗很小（比如本板 Chip2Chip 连接），不然不建议改成 0dB。

384. 目标速率为 Gen5 时，均衡是 Gen3 -> Gen4 -> Gen5 逐步做还是只做 Gen5 的？

默认是 Gen3->Gen4->Gen5 速率递增做均衡知道最高目标速率，如果配置了 bypass_eq_to_highest_rate=1 也可以只做 Gen5 的均衡。

385. FOM 是 Synopsys 独有的吗？

FOM 和 DIR 模式是 PIPE Interface 定义的，不是 SNPS 独有的。

大部分厂家的 PHY 都支持 FOM，但很多不支持 DIR。

386. Tx、Rx 均衡时的 C_0、C_{-1} 、C_{+1} 系数是算法自动调整的吗？可以自己控制吗？

问题补充：11 个 Preset 可以自由选择，但是这三个系数能自己控制吗？

可以自动调整，也可以自己控制。

一般来讲，均衡过程分粗调和细调，是先扫描 P0~P10 粗调，在这 11 组预设好的 Preset 系数中选择一组最优的，然后在这个 Preset 基础上细调三个系数。对应到链路训练过程中，TS1 Symbol 6 中的 use_preset 字段选择是用 Preset 粗调还是 Coefficient 细调。

个人理解，FOM 和 DIR（Direction Change）都是做均衡的调整方式，FOM 是直接给 Rx 质量打分，DIR 是更进一步给出了 Coefficient 调整的方向，DIR 属于手动控制的一种。一般来讲，采用 FOM 模式扫描 Preset 能够找到次优解，已经满足 BER 要求，DIR 属于锦上添花。

根据 SNPS PCIe Controller Databook，其支持 DIR，在其 PHY Databook 又备注其 PHY 不支持 DIR 模式。官方宣称不支持 DIR，但是实际上可以用。不让用 DIR 是因为 DIR 模式存在不收敛的风险：DIR 要是真是反复+-调系数，一次评估耗时 2 ms，24 ms 内最多调节 12 次，很容易超时。如果采用 DIR，可以考虑控制做均衡的 Loop 总数，最后即使 DIR 没收敛也可以靠这个 Loop 数目来结束，避免超时。

另外，根据使用经验，Cadence 公司的 PCIe IP 在均衡时候不扫码 Preset，只盯着一个固定的 Tx Preset 死磕。

387. 链路训练为什么 Gen1 到 Gen3 再到 Gen4，不能直接上去么？

一方面，应该是 Link 电气特性的要求，直接上最高速有较大概率会失败；另一方面，当初的设计应该是想可以回退到 Gen3。当然，如果为 Gen5 及以上版本的 IP，也可以设置 Bypass_eq_to_highest_rate=1，这样就不会做 Gen3、Gen4 的均衡，且是协议允许的。

对于 Synopsys 公司的 PCIe Controller，也可以设置 GEN3_EQUALIZATION_DISABLE=1 来直接到 Gen4。虽然可以这样测，但这并不符合 Base Spec，会过不了兼容性测试，多用于仿真加速。

388. LTSSM 中均衡相关变量 start_equalization_w_preset 是如何跳变的？

DSP 进入均衡后，start_equalization_w_preset 清零；

USP 进入均衡后，start_equalization_w_preset 清零；

进入 Recovery.RcvrCfg 后，start_equalization_w_preset 清零，

DSP start_equalization_w_preset 再置一

USP 满足以下情况时 start_equalization_w_preset 置一：

- ✓ 若当前 2.5 GT/s 或 5 GT/s，且最大支持 8 GT/s
- ✓ 若当前 2.5 GT/s 或 5 GT/s，最大支持 32 GT/s 且 bypass_to_highest_rate
- ✓ 若当前 8 GT/s，且支持 16 GT/s（最大不一定是 16 GT/s）
- ✓ 若当前 16 GT/s，最大支持 32 GT/s

389. 当前 USP 为 2.5 GT/s 或 5 GT/s 且最大支持 16 GT/s 时，start_equalization_w_preset 该怎么处理？

USP 当前 2.5 GT/s 或 5 GT/s，最大支持 16 GT/s 时，需要经过两次均衡调速，多次调整 start_equalization_w_preset 的值。

第一次切速进入 Recovery.RcvrCfg 后收到的 TS 中显示最大支持 8 GT/s，符合规则 (i)，start_equalization_w_preset 置一，Gen1 -> Gen3 做一次均衡进行第一次调速，进入均衡后 start_equalization_w_preset 清零 。

第二次切速进入 Recovery.RcvrCfg 后收到的 TS 中才切为最大支持 16 GT/s，符合规则 (iii)，start_equalization_w_preset 置一，Gen3 -> Gen4 做一次均衡进行第二次调速，进入均衡后 start_equalization_w_preset 清零 。

也就是说，切速过程为 Gen1 -> Gen3 -> Gen4，每次切速都做一次均衡。

390. PCIe Equalization 有哪几种模式？

PCIe Equalization 有 3 中模式：

- ✓ Full EQ Mode：全均衡模式。采用该模式时，PCIe 链路会对 8 GT/s 及以上每一档速率都要进行一次均衡。对于 32 GT/s，要在 8 GT/s、16 GT/s 及 32 GT/s 时都进行一次均衡，耗时较久。
- ✓ EQ Bypass to Highest Rate：只在最高速率进行一次均衡。该模式仅适用于 32 GT/s 及以上速率，当 8 GT/s、16 GT/s 开启该模式时，8 GT/s、16 GT/s 仍然不做均衡。
- ✓ No EQ Needed：采用该模式时，PCIe 链路不做均衡。采用该模式能够大大节省均衡的时间，以最快速度建立连接。

391. PCIe Full-Equalization 是什么？

Full EQ Mode，全均衡模式，是最传统的 PCIe 均衡模式。采用该模式时，链路训练到达 2.5 GT/s L0 后，返回 Recovery 继续针对 8 GT/s、16 GT/s 及 32 GT/s 等速率从低到高依次进行均衡，直到完成最高速率均衡。随着传输速率的提高，需要均衡的次数越来越多（单次约 100 ms），均衡耗时越来较久。

392. PCIe Bypass-Equalization-to-Highest-Rate 机制是什么？

如前文所述，采用 PCIe Full-EQ 模式链路训练耗时很久，EQ-to-Highest-Rate 模式是缩减均衡时间的方式之一。对于支持 32 GT/s 及以上速率的设备，其可能根本无需工作在 8 GT/s 或 16 GT/s 速率，这时再针对 8 GT/s 或 16 GT/s 速率进行均衡，多少有点浪费。

只有链路两端设备均支持 Bypass-Equalization-to-Highest-Rate 时才能采用该模式时，否则还是会按照 Full-EQ 模式进行链路训练及均衡。链路双方在 8b/10b 编码期间通过 TS

进行协商，决定是否要采用 Bypass-Equalization-to-Highest-Rate 模式进行均衡。如果协商后决定采用 Bypass-Equalization-to-Highest-Rate 模式进行链路训练，链路到达 2.5 GT/s L0 后，返回 Recovery 直接切速到最高速率并针对最高速率（32 GT/s 或以上）进行 1 次均衡。对于两端有能力最高支持 32 GT/s 及以上速率的设备，如果把目标速度设置为 8 GT/s 或 16 GT/s 速率，是不建议采用 Bypass-Equalization-to-Highest-Rate 模式的，即便采用了，也不会针对 8 GT/s 或 16 GT/s 的目标速率进行均衡。

393. PCIe No-Equalization-Needed 机制是什么？

采用 PCIe Full-EQ 模式链路训练耗时很久，No-Equalization-Needed 模式是除 Bypass-Equalization-to-Highest-Rate 模式外的另一种缩减均衡时间的方式。

从名字也看得出来，No-Equalization-Needed 模式比 Bypass-Equalization-to-Highest-Rate 模式更狠，采用 No-Equalization-Needed 模式时一次均衡都不做，能够大大节省均衡的时间，以最快速度建立连接。如果本地存有之前均衡预设好的值，或者对链路质量足够有信心觉得不用均衡，可以选用此模式不做均衡。

只有链路两端设备均支持 No-Equalization-Needed 时才能采用该模式时，链路双方在 8b/10b 编码期间通过 TS 进行协商，决定是否要采用 No-Equalization-Needed 模式进行链路训练。如果协商后决定采用 No-Equalization-Needed 模式进行链路训练，链路到达 2.5 GT/s L0 后，返回 Recovery 直接切速到最高速率且不做均衡。对于两端有能力最高支持 32 GT/s 及以上速率的设备，如果把目标速度设置为 8 GT/s 或 16 GT/s 速率，是不建议采用 No-Equalization-Needed 模式的。非要采用的话，也行，同样不会针对 8 GT/s 或 16 GT/s 等任何速率进行均衡，但是链路信号质量得不到保证。

394. PCIe 是如何重新发起均衡的？

当 USP 请求重新均衡时，USP 进 Recovery 状态，在 Recovery.RcvrCfg 状态时给 DSP 发送 TS2，Request Equalization 位置一，Equalization Request Data Rate 置为需要均衡的速率，Quiesce Guarantee 位可以（非必须）置一，通知 DSP 其在 1 ms 内将启动均衡，以确保不会对 DSP 其他操作产生影响。

DSP 收到 USP 的均衡请求后（处于 Recovery.RcvrCfg 且连续收到 8 个 Request Equalization 位置一的 TS2），可以（非必须）依照 USP 发来 TS2 中的 Quiesce Guarantee 位来判断是否会对 USP 产生影响。只有在确保 DSP 启动均衡不会对 DSP 和 USP 的其他操作产生影响的前提下，DSP 才能启动均衡进程。如果 DSP 确定均衡不会对 DSP 和 USP 的其他操作产生影响，其需要在 1 ms 内按照 USP 请求的均衡速率发起均衡进程。否则，DSP 链路状态寄存器中相应速率的 Link Equanlization Requset 位置一，由 DSP 作为均衡请求方，查询 USP 状态（Quiesce Guarantee 位置一）。

USP 收到 DSP 的均衡请求后，可以（非必须）暂停其正在执行的操作，并按照收到 DSP 的均衡请求，再次向 DSP 发起均衡请求（TS2 的 Quiesce Guarantee 位必须置一）。USP 响应 DSP 均衡请求的时间没有限制，但应尽快执行。DSP 收到 USP 响应后，在保证均衡对自身操作没有影响的前提下，需要在 1 ms 内发起均衡，否则把相应速率的 Link Equanlization Requset 位置一稍后再来一次。

DSP 主动发起均衡请求也含在了以上介绍中。

395. PCIe Quiesce Guarantee 是什么？

Quiesce Guarantee，静默保证，是 TS 中的一个字段（Symbol 6 bit6），常用于 DL_Active 之后重新进行均衡的情况。Quiesce Guarantee 是一种可选的手段，跟 TS 中的 Request Equalization、Equalization Request Data Rate 字段配合使用。

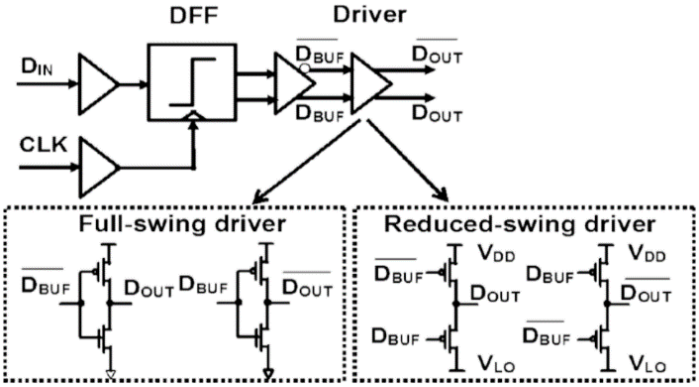
链路工作在 8 GT/s 或以上速率时，如果 DSP 检测到均衡问题或者 USP 想要重新均衡，则需要重新进行均衡。此时 DL 层可能正在进行数据传输，如果贸然启动均衡，有可能会在 Port、Device、System 层面引发错误。比如，DSP 发起了一笔 MRd，在收到 CplD 之前进入均衡，鉴于均衡耗时较久，会导致 Completion Timeout。Quiesce Guarantee 正是这样一种机制，告知对端我已准备好，你随时可以发起均衡。

396. PCIe Equalization 中提到的 Full Swing、Reduce Swing 是什么意思？

PCIe 均衡中的 Swing 是指 Tx 的输出电压摆幅，分两种模式：

- ✓ Full Swing：全摆幅模式，超高速 Tx 的必选模式，能够在较低的电压范围内保持较高的噪声容限。
- ✓ Reduced Swing：降低摆幅模式，是一种可选的低功耗摆幅模式，多用于对电源和噪声干扰比较敏感的系统。

两种模式的典型电路如图所示。



▲图： Transmitter with full-swing and reduced-swing drivers.

图源：https://www.researchgate.net/figure/Transmitter-with-full-swing-and-reduced-swing-drivers_fig10_224085165

397. 为什么 8 GT/s 及以上速率时需要均衡？

LTSSM 中的 Recovery.Equalization 的确是到 Gen3 (8 GT/s) 时才有的，但均衡不是到 Gen3 才有的。2.5 GT/s 及 5 GT/s 时，信号速率相对较低，在 Tx 端采用信号去加重即

可实现较好质量的信号传输，这里的信号去加重即是均衡技术的一种。由于信号去加重时参数固定，收发端无需协商，不需要 Recovery.Equaliation 这一过程进行协商。

数据速率上升到 8 GT/s 及以上后，信号传输面临的信号完整性问题更加突出，仅采用传统的去加重无法满足高质量的信号传输，需要收发端动态协商 Tx/Rx 参数来优化信号质量。

398. PCIe 训练完毕后，如何重新进行链路训练？

只需要将 Link Control Register 的 Retrain Link 置一就好了。如果需要在重新训练时改变链路宽度、链路目标速度，需要在将 Link Control Register 的 Retrain Link 置一前完成相关参数的修改。

399. DSP Equalization 没有 Phase 0，那 DSP 是怎么给 USP 发送 Preset 的？

DSP 在切速前的 Recovery.RcvrCfg 状态给 USP 发送 Preset。至于为什么要放在这个状态，spec 中未明确说明。从 Equalization 前后状态来看（Recovery.RcvrLock -> Recovery.RcvrCfg -> Recovery.Speed -> Recovery.Equalization -> Recovery.RcvrLock），肯定是要放在 Equalization 之前的，放在之后没有意义。Equalization 前只有 RcvrLock、RcvrCfg、RcvrSpeed 三个状态，其中 RcvrLock 要做位锁定符号锁定，RcvrSpeed 要切速，只有个 RcvrCfg 还比较闲。如果 DSP 也来一个 Phase 0，由于在 phase0 时速度已经切完了，此时链路质量很差的话，USP 就解析不出 DSP 发给他的 preset 了。

注意：如果 USP 当前已经为 128b/130b 编码切要切到更高一档的速度，USP 也是要在 Recovery.RcvrCfg 状态给 DSP 发送 preset 的。

均衡实际问题

400. 在 Link Up 之后想调节 Rx 端的均衡参数，OS 下写完相关寄存器后是否需要重新 Retrain？还是说 OS 下写其实根本不会再生效？

改均衡参数后 Retrain 是肯定需要的，但在 OS 下改写均衡参数不一定会生效，可 Retrain 后读这几个均衡参数看是否生效。实在不行可调参固件里的均衡参数，重新编译固件后再建链。

401. 目的为恶化 RC 端 Rx 的信号质量，除了调整 Rx 的 CTLE 和 DFE 等均衡参数，还有其他的恶化手段吗？

可考虑把 EP 端 Tx 的均衡参数进行恶化，相当于 Rx 恶化了。Tx 端信号电气一致性测试时会进行该测试，用误码仪加抖动。

402. PCIe寄存器中有个Link Equalization Request Interrupt Enable，这个中断是如何产生的？产生后如何处理？

个人理解，这个中断是在发起链路均衡后硬件自动触发的，协议中未规定应该如何处理，应该是厂家自己决定。

403. PCIe Controller常用的均衡算法是用FOM Mode还是DIR Mode？能混合用吗？

FOM和DIR两种模式基本是分开用的，目前未碰到过两者混用的情况。

原理上FOM+DIR搭配没什么问题，但根据个人使用DIR Mode的经验，DIR Mode下PHY直接给DIR进行反馈，没用到FOM提供初始参数。直觉感觉先粗调把Tx Preset扫一遍，找个FOM最大的，然后到微调。

虽说采用DIR调整似乎更顺畅一些，但DIR Mode存在不收敛的风险。此外，FOM和DIR的最后结果差不多，但DIR花费的时间太久，一般建议关闭DIR，只用FOM。

404. 实际应用中，用FOM均衡机制选Preset靠谱吗？如果PHY的FOM和DIR结果都不可信，但是PHY Rx均衡本身无问题，那系统是否有办法设定固定Preset值？该做法实际中会不会带来额外的应用问题？

不清楚FOM在PMA中的实现机制，我们感觉有时候用FOM选的Preset不是最优的。之前也曾尝试开DIR Mode，但Synopsys说怕没法收敛，所以就关了DIR Mode，靠FOM找个次优的Preset也能用。或者DIR Mode能把Loop数给限制住，以免超时。

确实有IP只要求对端使用固定Tx Preset的，相当于不调整对方的Tx，只调整自己的Rx。结果如何就看自己Rx的能力了。

我们当时跑过固定的Preset和auto Preset的PCIe Margin，挑选出最优的Preset的值是Preset7，后来请BIOS重新适配的。

405. 卡插在x86上，链路均衡过程中卡在Phase 2中不动，有什么解决思路？

可以修改一下要求对端的TX Preset，自己的Rx端做一下RX Reset或者CDR Relock。

406. 均衡Slave收到Master发来的Tx Preset后可以不改PMA电气参数吗？

问题补充：即采用固定Tx Preset，但谎称自己改过了让均衡流程往下走。

不应该这样。Tx Preset是强制支持的，在PCIe电气测试规范中，进入Compliance Mode后会扫描不同Tx Preset并发送Pattern，然后进行信号分析。即便能这样，骗过均衡Master，但信号质量无法作弊，BER不收敛仍然无法完成均衡。

407. 均衡的 Phase 0 阶段，USP 端发送的 Preset 值为 1111 保留值不在 P0 ~ P10 范围内，这是为什么呢？

初始 Preset 在 32.0 GT/s Lane Equalization Control Register (Offset 20h)寄存器中进行配置，后面 Preset 交互未硬件自动交互。Equalization 阶段总归要有一个初始用的 Preset，1111b 是该寄存器 Default 值，表示把选择权交给对方，让对方选择初始使用的 Preset。

408. 链路速率切为 Gen5 做均衡时，LTSSM 状态从 Recovery.Equalization 的 Phase 1 因超时跳到了 Recovery.Speed，速率也切为了 Gen1 速率，这种情况该如何解决？

如果为仿真，VIP 报超时，可以加大 VIP 均衡相关 Timeout 时间；如果是 DUT 报超时，可以调节 DUT PCIe Controller 仿真加速相关参数，加速倍数别那么大。

如果该问题发送于系统实测场景，可以调节 Preset 值。考虑到 Equalization 的 Phase 0、Phase 1 阶段误码率要求比较低（<10-4），这个阶段超时可能是交互时有一方未收到对端的 TS1，可尝试调节 Preset 来解决。调节 Preset 时，Host Tx Preset 由 Host 自己控制，并强制设置 Device Tx Preset。

409. 采用均衡阶段协商的 Tx Preset 测试 Tx 眼图发现不是最佳，有什么办法调整吗？

这不是常规做法，不应依据 Tx 眼图来判断 Tx Preset 的好坏。Tx 眼图经过 Channel 到 Rx 均衡之后的 Eye 才是 PHY 采样数据的信号，所以协议依附于 Rx 的 FOM 来判决 LP 的 Tx Preset 是否合适。

410. 用 Synopsys 公司的 PHY 测试发现多次均衡结果不一致，这是否正常？

问题补充：即同一设备多次上电完成建链后选出来 Link Partner 的 Tx Preset 不同。

正常。个人理解均衡只是找到一个符合误码率要求的均衡参数，这个参数不一定是最优解。寻找均衡最优解需要花费很长的时间，而建链过程中显然不会留这么长时间给均衡。取决于均衡算法，每次均衡结果只能说大致一致，很难完全一致。如果已知某个 Tx Preset 是最优的，可以尝试修改均衡算法采用这个固定的 Tx Preset，不要让 IP 自己挑选。但是，考虑到板子之间存在差异，某个板子上的均衡最优解不一定适用于其他板子。

411. PCI-SIG Compliance 测试过了，但在客户那 Tx Preset 测试过不了，该怎么办？

根据经验，PCI-SIG 的 Compliance 测试要求比较低，为最低要求，只要有一个 Tx Preset 过了就算过了，客户的要求往往比 PCI-SIG 严苛得多。如果其他 Tx Preset 过不了，可尝试优化 PCB 材料、严格阻抗控制、采用优质连接器或者减少 PCB 走线长度。

412. Gen4 目标速率开启了 Equalization bypass to highest rate，为什么没起作用？

当最高速率为 Gen5 且开启了 Equalization bypass to highest rate 时，只进行一次切速/均衡，即 Gen1 -> Gen5。Equalization bypass to highest rate 是 Gen5 的 Capability，若目标速率为 Gen3/Gen4 且开启了 Equalization bypass to highest rate，不是像 Gen5 那样只在最高速率进行一次均衡，而是不进行均衡，直接从 Gen1 -> Gen3/4，没有中间速度。用 VIP 进行仿真时，VIP 会报错。

Flit Mode

413. PCIe 6.0 有哪些新变化？

- ✓ 单 Lane 传输速率由 32 GT/s 变为 64 GT/s；
- ✓ 传输信号由 NRZ 两电平变为 PAM4 四电平；
- ✓ 编码方式由 128b/130b 变为 FLIT 编码；
- ✓ 引入了 FEC 前向纠错机制，并维持既有的 Retry 重传机制；
- ✓ LTSSM 新增加了 L0p 低功耗状态，允许部分 Lane Electric Idle、部分 Lane Active；
- ✓ 新加了几种机制：DOE、CMA、DMWr、IDE。

（参考：https://blog.csdn.net/weixin_40357487/article/details/120714950）

414. PCIe 6.0 的 Flit Mode 是什么？

在 PCIe 1.0~5.0 中，TLP 是事务层包的基本传输单元。考虑到 TLP Header 的 Prefix 及 Data Payload Length 不定，TLP 总长度也是不定的。尤其当 Data Payload Length 较长时，传输一笔 TLP 耗时较长，需要较长的时间才能对该 TLP 有效性进行确认，时延较大。为了降低该时延，PCIe 6.0 中采用了基于 Flit 的传输方案，以 Length 更小的 Flit 为基本单位进行 Retry 及 Flow Control。

PCIe 6.0 采用 256 B 固定长度的 Flit，单个 Flit 由 236 B TLP + 6 B DLP + 8 B CRC + 6 B FEC 组成。每一笔 Flit 中都有 6 B DLP，此外无需单独传输 DLLP。对于 Length 大于 236 B 的 TLP，需要进行拆分，分发到多笔 Flit 中进行发送；对于 Length 小于 236 B 的 TLP，多笔 TLP 可以共享同一 Flit；TLP 在 Flit 中的起始位置不局限于任何边界。

PCIe Gen6 速率下必须采用 Flit Mode，若开启了 Flit Mode，在 Gen6 以下速率时仍然需采用 Flit Mode。即便为 8b/10b 或 128b/130b 编码，考虑到 Flit 中的 TLP、DLP 等有其固定位置，Flit Mode 时不再使用 STP、SDP、END、END 等 Packet Marker 及 Frame Token。

415. 为什么需要 Flit Mode?

搜索的结果多数是 Low Latency，以及 FEC 针对的是 Fixed Sized。但是直观来看，似乎只有 MWr 和 CplD 的时候，Flit Mode 中 236 B TLP 的有效利用率才会较高。

实际使用场景中，背靠背大量发送 MRd 或 Msg 等较短 TLP 的场景也不多吧。EP 从主存读数据就是 MRd，这也是为什么网卡发送带宽远低于接收带宽的最大影响原因

416. PCIe Flit Mode 支持几种 Flit?

从 Flit Size 看，目前（PCIe 6.0）仅支持 256 B 固定长度的 Flit。

按照 Flit Type 分类，Flit Mode 支持 3 中 Flit，分别为 Idle Flit、NOP Flit 及 Payload Flit。Flit 中 DLP Bytes 的 DLP[0] 中 Flit Usage 及 Prior Flit was Payload 位域来指示当前及前一 Flit 的类型。考虑到 Idle Flit 和 NOP Flit 在 Rx 端的处理方式相同，协议中并未对 Idle Flit 和 NOP Flit 进行区分，Rx 端更关心是否为 Payload Flit。

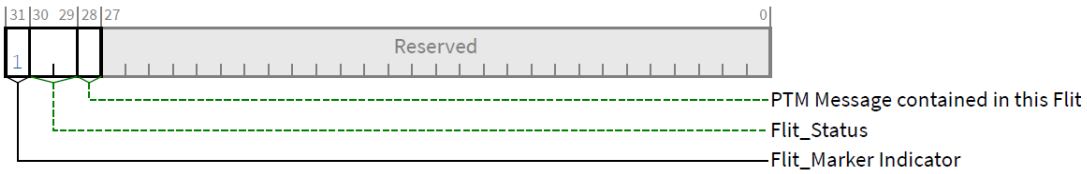
417. Flit_Marker 是什么？有什么用？

Flit Marker 出现于 PCIe 6.0，Flit Mode 下用以指示当前 Flit 中最后一笔 TLP 的有效性。

Flit Marker Size 为 4 B，是 DLLP Payload 的一种，位于 Flit DLP Byte 的 DLP[2:5] 位置。DLP[0:1] 中的 DLLP Payload Type 指示当前 DLLP Payload Type。DLLP Payload Type 值为 1b 时，表示当前 DLLP Payload Type 为 Flit_Marker 或 Optimized_Update_FC （两者共用），两者通过 Payload bit[31] 进行区分，bit[31]=1 时为 Flit_Marker。

Flit Marker 格式如下图所示，包含 Flit_Marker Indicator、Flit_Status 及 PTM Message contained 位域。Flit Status 能够指示 ①No special information、②Last TLP Nullified、③Last TLP Posioned 这三个状态，接收端需要根据该状态对接收到的 TLP 进行处理。Flit Marker 中的 PTM Message Contained bit 指示当前 Flit 中是否包含有效的 PTM Message 的最后一个 Symbol。

注意：Flit_Marker 跟 TLP 状态有关，需要跟 TLP 一起保存在 Retry Buffer 中。



418. Flit 不做编码，不怕出现连续 0 或 1 烧坏耦合电容吗？时钟同步怎么办？

有 Scramble，Scramble 能在一定程度上避免连续的 0 或 1。时钟同步同理。

419. Flit 不做编码，难道 PCIe 6.0 不需要控制字符集 K 字符吗？有续集 OS 的字符编码可能会与 Payload 重叠吗？如何区分 Payload 与包头？如果 Order Set 跟 Payload 重叠了怎么办？

存在重叠也没问题。Flit 属于 Data Block，OS 属于 OS Block。虽然 PCIe 6.0 没有 Sync Header 来指示 OS/Data Block，但仍然有相关机制来识别当前属于哪个 Block。既然能够识别出是哪个 Block，就不担心把 Flit 中的 Payload 误识别未 OS。

420. PCIe 6.0 每个 Flit 最大容纳 TLP 中的 236 B，超过 TLP 超过 236 B 就需要放到多个 Flit 中，那么每个 Flit 中都需要一个 TLP Header 吗？

TLP 拆开到多个 Flit 之后，不需要每个 Flit 中都放一个 TLP Header，直接把拆开的 TLP Bytes 填充到 Flit TLP Bytes 位置即可。

421. PCIe 6.0 中 CRC 和 FEC 执行顺序是什么

在发送端，先加 CRC 再加 FEC。在接收端，接收端先做 FEC 纠错，再做 CRC 检查。

在接收端，FEC 每组 ECC 能够纠正单 Byte 错误，超过单 Byte 就纠正不过来了。无论是否纠正过来，均会进行 CRC 检查，CRC 检查出错后进行 Retry。

422. 为什么 PCIe Flit Mode 采用 3 路交织的方式计算 ECC 就能保证 16 路不出错呢？

问题补充：PCIe Flit Mode 采用 3 路交织的方式计算 ECC，对 236 B TLP + 6 B DLP + 8 B CRC 进行计算，三个连续字节属于三个不同的 ECC 组，保证线路中长度≤16 的任何突发不会影响码字中的一个以上符号。

三路交织算 ECC 的方案不是为了保证 16 路（x16?）不出错，而是为了保证同一条 Lane 发生的 16b 以内的 Burst Error 不会影响每组 ECC 组超过 1 个以上的 Symbol。

PCIe FEC 机制每组 ECC 都能够纠正该组内 1 B 错误。如果一个 Flit 内某条 Lane 上发生了单次 16b 及以内的 Burst Error，16b 最多出现在连续的 3 个 Symbol 上，无论链路宽度为 x1、x2、x4、x8 还是 x16，这 3 个出错的 Symbol 都会落在三组交织的 ECC 码组内且每组受影响不超过 1 B，从而确保错误可纠正。也就是说，只要该 Flit 内发生了单次 Burst Error 且不超过 16b，都能够纠正过来。

423. Flit Mode 下，如果只想发最短的单个 MRd TLP，NOP TLPs 会按照什么频率插入呢？

问题描述：PCIe 6.0 Spec 讲 It is required to transmit NOP TLPs while TLP transmission is active if there are no other TLPs to transmit. 在 PCIe Gen6 Flit Mode 下，如果只想发最短的单个 MRd TLP（3 DW Header + 1 DW LCRC = 16 B），NOP TLPs 会按照什么频率插入呢？会每个周期检查一次是否有 Active TLP 吗？如果当前周期没有有效 TLP 会插入多长的 NOP TLP 呢？

似乎 Spec 没有明确指出 Flit Accumulation Latency 的限制，至于等多久插入 NOP TLP，猜测由各个厂商在满足 Flit 打包规则前提下自行决定。即，没有有效数据就插 NOP，Half Flit 中 TLP 不超 8 个，NOP 后的 TLP 起始位置 4dw 对齐。

424. 如果应用层长时间没有 TLP，链路是持续传输 NOP-FLIT 呢，还是停止传输 Flit？

取决于低功耗设置。

425. PCIe 开启 Flit Mode 需要写哪些寄存器？

相关能力寄存器中需显示支持 Flit Mode，控制寄存器中需 Enable Flit Mode。

426. Flit Mode 时，Tx Buffer 和 Retry 机制应该都做在物理层、而非 Data Link 层了吧？

两种模式可选。

427. Flit 定界是以 Flit State 是否为 Poisoned 或者是 Nullified 作为结束的标志的吗？

不是。Flit 是定长物理包，不需要定界。

428. Flit 长度是固定的，发送时分发到最大 16 个 Lane 同时传输的，接收端怎么从 16 个 Lane 恢复出这个 Flit 呢？

怎么把 Flit 分发到各 Lane 上，就再逆操作收集起来。

429. PCIe Flit 打包时，如果有 NOP 插入，TLP 要求 4 DW 对齐，那么在 x8 上也是这样吗？

协议原文：Once a NOP TLP is scheduled to be sent, NOP TLPs must be scheduled till the next 4 DW aligned boundary within the Flit or the Flit boundary, whichever is earlier.

这句话意思是说，Flit 打包时，NOP 一旦安排上了，就一直安排直到下一个 4 DW 对齐。这句话中并未提到是针对哪个链路宽度，x16 时好理解，当前 Symbol Time 内所以 Lane 上都为 NOP；当链路宽度为 x8 时，则需要每条 Lane 上至少需要排两个 Symbol 的 NOP，其中第二个感觉必要性不大。协议这里似乎未说清楚，但多排几个肯定没有问题。

430. 什么样的 Flit SN 算是隐式的？既然 Flit 中都携带了 SN 字段，为什么还说它是隐式的？

DLP cmd 指示当前 DLP 内的序列号 SN 是 Explicit SN 还是 Ack/Nak SN。当为 Ack/Nak SN 时，接收端根据之前的 Explicit SN 推算出当前的真实 SN，即 Implicit SN。之所以说它是隐式的，是因为 Flit 中这个 SN 并非是当前 Flit 的 SN，真实的需要推算出来。

431. PCIe Flit Mode 时不支持动态上调链路宽度吗？

引文：With flit Mode, link upconfigure, which performs dynamic link width adjustment through the state transition L0 -> Recovery -> Configuration -> L0, is not supported.

是的，Flit Mode 下不支持走 Recovery 进行升 Lane，即便不支持 L0p 也不能用这种方法。

432. Flit 模式下是不再使用令牌了的，那么接收端怎么知道 Flit 的开始位置呢？

边界只能出现在四个位置，电路并行检测一下即可。

433. Flit 是在哪一层封装的？是如何封装的？

Flit 最终是在数据链路层封装完毕的，大致的封装过程如下：

事务层提供完整 TLP，该 TLP 下发给数据链路层。

数据链路层对事务层发来的 TLP 进行切片/拼接、缓存，用作 Flit 中的 TLP Bytes 。

接着，数据链路层提供 Flit 传输过程中所需的 DLP，包括 Flit Sequence Number、Ack/Nak、Flow Control 及 Power Management 等信息。

获取到 Flit 所需的所有 TLP 及 DLP Bytes 后，计算 CRC 及 FEC。其中 CRC 计算的输入信息为 236 B TLP + 6 B DLP 共 242 B，FEC 计算的输入信息为 TLP+DLP+CRC 共 250 B。NOP Flit、Idle Flie 也是由数据链路层进行添加。

434. PCIe 6.0 规定的 Flit Size 有几种？

PCIe 6.0 规定的 Flit Size 只有一种，即 256 B Flit。即便 TLP Length 远远不够 236 B，甚至只需发送 DLP 的场景，仍然需要发送 256 B 长度的 Flit。

256 B Flit 中，TLP 236 B 占比 92%，即带宽有效率 92%。若采用 Size 更大的 Flit，带宽有效率能够得到提升，同时也使得 Flit 时延更大；若采用 Size 更小的 Flit，能够降低 Flit 传输时延，但是带宽有效率会打折扣。256 B Flit Size 是综合考虑 PCIe 带宽效率及 Flit 时延后选择的一种较优方案。

435. 每个 Flit 都需要 Ack/Nak 握手吗？这不会影响效率吗？

首先，跟 Non-Flit Mode 相同，没有必要对每一个 Flit 相应 Ack，可以对多个正确接收到的 Flit 统一回复一个 Ack。若接收到的某个 Flit 的 Ack 信息，则该 Ack DLP 中 Flit Sequence Number 之前未进行确认的 Ack 都可以认为是已经正确接收到了。

其次，Ack/Nak 并不会影响传输效率。PCIe 6.0 Flit 中给 DLP 分配了 6 B，其中 DLP[1:0] 用来传输 Sequence Number 及 Ack/Nak，并不会额外占用其他 Slot。

436. DLP 是嵌合在 Flit 中的，当没有要发送的 TLP 时，Flit 中 TLP Payload 位置如何填充？

当没有 TLP 要传输时，TLP Bytes 位置需要填充 NOP TLP。NOP TLP 对应的 Flit 可以为 NOP Flit 或 Idle Flit。NOP Flit 中的 DLP Byte 仍然填充有有效的 DLP，Flit Seqence Number 有实际值，跟前一 Flit Sequence Number 相同且不消耗新的 Sequence Number，NOP Flit 出错时也无需 Retry；Idle Flit 中 DLP 字段也是空的，DLP 字段填充了 NOP2 DLLP（全零），Flit Sequence Number 为 0，出错时也无需 Retry。

跟 NOP/Idle Flit 相对的 Flit 便是 Payload Flit。当前 Flit Type 存放在了 DLP[0] 中。

437. Flit Mode 时，TLP Header 需要放在 Flit 固定位置吗？

Flit Mode 时，TLP 起始位置是 4 DW 对齐的。Flit Mode 时只规定了 TLP、DLLP、CRC、ECC 等在 Flit 的位置，但没有要求 TLP 中的 Header 具体应该放在哪儿，可以放在 Flit TLP Bytes 任意 4 DW 对齐的位置。接收端通过 NOP 来查找 TLP 边界。

438. Flit Mode 时，若 TLP Size 小于 236 B，该如何生成 Flit？大于 236 B 时呢？

TLP Size 大于 236 B 时，一个 Flit 不够用，肯定需要拆分到多笔 Flit 中去。TLP Size 小于 236 B 时，有可能该 TLP 能够完成得放在一个 Flit 中，也有可能会被拆成两笔放在两笔相邻的 Flit 中。不论 TLP Size 是多少，放在 Flit TLP Bytes 位置就好了。

439. Flit Mode 与 Non-Flit Mode 时的 TLP Header 有何不同？

主要看到以下几点不同：

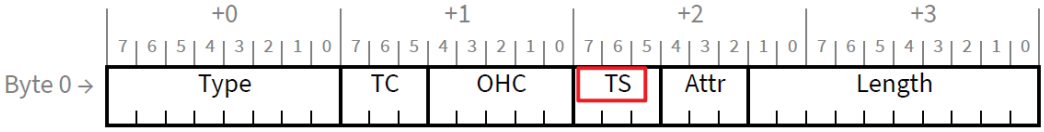
- ✓ 取消了 Fmt，合并到 Type，Type 种类更多。
- ✓ 新加了 TS (Trailer Size)，用来指示 Trailer Size。目前，TS=000b 时表示没有 Trailer，TS=001b 时表示当前 TLP 包含 1 DW 的 ECRC。OHC-C 时，TS=101b 指示 3 DW IDE MAC，TS=110 指示 4 DW IDE MAC 及 PCRC。其他的预留。
- ✓ 新加了 OHC (Orthogonal Header Content)，
- ✓ TLP Prefix 相关内容合并到 OHC 中，通过 OHC 来指示。
- ✓ 预留了 Suffix 后缀。

440. PCIe TLP PCRC 是什么？

PCRC 全称 Plaintext CRC，纯文本 CRC，是 IDE TLP 中的概念，PCIe 6.0 才出现。具体请查看 IDE 机制或者 Plaintext 机制。

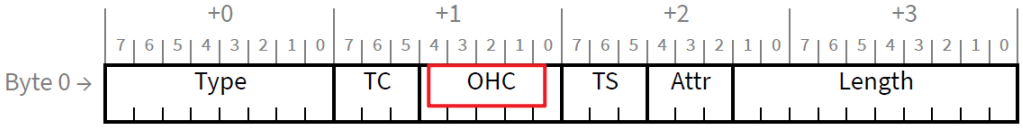
441. TLP Trailer 是什么？

PCIe TLP 后可以跟一些 Trailer，比如 ECRC 及 IDE 机制的 IDE MAC。Flit Mode 下的 TLP Header 中新加了 TS (Trailer Size) 位域（如下图）来指示该 TLP 是否存在 Trailer 并指示 Trailer Size。目前，TS=000b 时表示没有 Trailer，TS=001b 时表示当前 TLP 包含 1 DW 的 ECRC。对于 IDE TLP 而言，其 TLP Header 必须采用 OHC-C，TS=101b 指示 3 DW IDE MAC，TS=110 指示 4 DW IDE MAC 及 PCRC。其他的预留。



442. TLP Header 中的 OHC 是什么？

OHC，全称 Orthogonal Header Content，正交头内容，是 PCIe Flit Mode 下新加的一个 TLP 字段。TLP Header Base (TLP 第一 DW，如下图) 中的 OHC 用来指示其后是否存在 OHC、是何种 OHC。不同的 OHC 代表了不同的 TLP Header 格式。更多 OHC 介绍请见《PCIe OHC 介绍》。



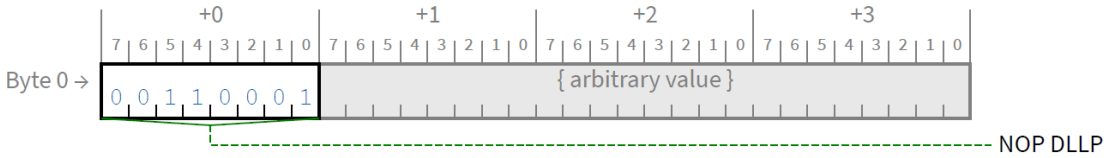
443. NOP TLP 是什么？

PCIe 工作在 Flit Mode 下，发送端没有 TLP 要发送时需要插入 NOP TLP。NOP TLP Header 中的 Type 字段全 0，其他所有字段 Reserved。NOP TLP 不携带 Data Payload，一笔 NOP TLP Size 为 1 DW，且不消耗流控机制的 Credit。接收端收到 NOP TLP 后进行丢弃。

若开启了 IDE 机制，NOP TLP 不能作为 IDE TLP 进行传递，接收端收到 NOP IDE TLP 后按照 Malformed TLP 处理。

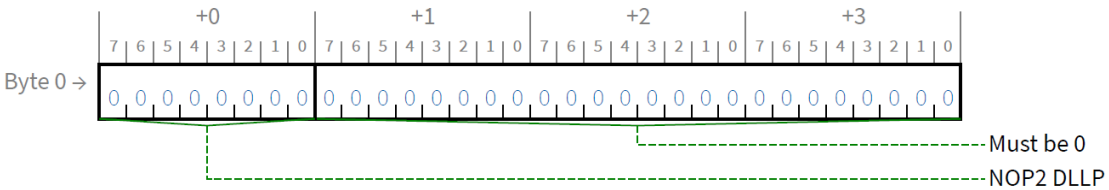
444. NOP DLLP 是什么？

NOP DLLP Payload 格式如下如所示，其 DLLP Type 为 31h，Payload 字段可以为任意值。Flit Mode 及 Non-Flit Mode 时都有 NOP DLLP 的概念。Non-Flit Mode 时，NOP DLLP 仅用于链路测试场景，常规操作模式时不应发送 NOP DLLP；Flit Mode 时，发送端没有 DLP 要发送时需要插入 NOP DLLP。接收端接收到 NOP DLLP 后，在做完数据数据完整性检测后直接丢弃。



445. NOP2 DLLP 是什么？

NOP2 DLLP Payload 格式如下如所示，其 DLLP Type 及 Payload 字段必须为全 0。NOP2 DLLP 是 PCIe 6.0 出现的概念，仅用于 Flit Mode。Idle Flit 时，DLP[2:5] 必须填充 NOP2 DLLP。



446. NOP DLLP 和 NOP2 DLLP 有何区别？

大致有以下几点不同：

- ✓ 格式不同。NOP DLP 是有具体意义的 DLP Type，DLP Payload 字段为任意值；NOP2 DLLP 的 DLP Type 及 Payload 均为全零。
- ✓ 用处不同。NOP DLLP 支持 Flit Mode 及 Non-Flit Mode，而 NOP2 DLLP 仅出现于 Flit Mode。

447. Idle Flit 是什么？有何作用？

Idle Flit 的 236 B TLP Byte 全部为 NOP TLP，且 DLP Bytes 为全零，DLP[2:5] 为 NOP2 DLLP。Idle Flit 的 Sequence Number 为 0，且无需 Retry。

Idle Flit 用于 PCIe 训练过程中。Flit Mode 下，LTSSM 处于 Configuration.Idle 或 Recovery.Idle 状态时，为 Idle Flit Handshake Phase，Tx 持续发送 Idle Flit。Rx 连续接收到两笔有效的 Idle Flit 后跳转到 Sequence Number Handshake Phase。

448. NOP Flit 是什么？有何作用？

NOP Flit 是一种 TLP Byte 全部为 NOP TLP 的 Flit，出现于 PCIe 链路训练完毕之后。NOP Flit 的 236 B TLP Byte 全部为 NOP TLP，DLP Byte 可以为任意有效的 DLP。只要有一笔为 Non-NOP TLP，该 Flit 就称不上是 NOP Flit。NOP Flit 的 DLP 字段仍然是有效的，可以承载流控等相关 DLP。

NOP Flit 不占用流控相关 Credit，不占用 Flit Sequence Number 且无需 Retry，其 Sequence Number 采用前一 Flit Sequence Number。

449. Idle Flit 和 NOP Flit 有何区别？

- ✓ 格式不同。Idle Flit 的 236 B TLP Byte 全部为 NOP TLP，且 DLP Bytes 为全零，DLP[2:5] 为 NOP2 DLLP。NOP Flit 的 236 B TLP Byte 全部为 NOP TLP，DLP Byte 可以为任意有效的 DLP。
- ✓ 应用场景不同。Idle Flit 用于 PCIe 训练完毕进入 L0 之前，LTSSM 处于 Configuration.Idle 或 Recovery.Idle 时，Tx 持续发送 Idle Flit，Rx 连续接收到两笔有效的 Idle Flit 是 Rx LTSSM 进入 L0 的条件（之一）。NOP Flit 出现于 PCIe 链路训练完毕之后，当没有 TLP 要发送时，Tx 发送 NOP Flit。NOP Flit 中的 TLP Bytes 全部填充以 NOP TLP，DLP Bytes 仍照常发送 DLP，仍具备流控等相关功能。
- ✓ 其他不同，比如 Flit Sequence Number，Idle Flit 的 SN 为 0，NOP Flit 的 SN 不一定为 0。

450. NOP TLP 和 NOP Flit 有何区别？

同样都是 NOP 操作，但两者的单位不同。NOP TLP 是 TLP Level 的 NOP，其粒度为 1 DW，该 TLP 内无需任何操作；NOP Flit 是 Flit Level 的 NOP，其粒度为 1 Flit，该 Flit 内无需任何操作。存在 NOP TLP 的 Flit 不一定为 NOP Flit ，但 NOP Flit 的 TLP Bytes 一定为 NOP TLP。但凡 TLP Bytes 内存在 1 DW 数据非 NOP TLP，该 Flit 便称不上是 NOP Flit。

451. NOP Flit 和 NOP DLLP 有何关系？

仅对 Flit Mode 而言，NOP Flit 和 NOP DLLP 没有太大关系。NOP Flit 中要求 TLP Bytes 全部为 NOP TLP，DLP Bytes 不要求是 NOP DLLP。当没有 DLLP 要发送时候，在 Flit 的 DLP Bytes 插入 NOP DLLP。NOP DLLP 不一定出现在 NOP Flit 中，Payload Flit 中的 DLP Bytes 也可以为 NOP DLLP。

452. PCIe Gen6 中 DLLP Type 有哪几种？

Non-Flit Mode 下，DLLP Type 跟 PCIe 5.0 时的 DLLP Type 完全相同，有 Ack/Nak、Flow Control、PM 及 Vendor Defined、NOP 等这几大类 DLLP。

Flit Mode 下，不支持 Ack/Nak，废弃了 MRInit、MRInitFC1、MRInitFC2 及 MRUpdateFC 等 MR-IOV 相关的 DLLP，新加了链路管理类的 LM DLLP（目前只服务于 L0p）及 NOP2 DLLP。

453. Flit Mode 下的 DLLP 有何变化？

DLLP 格式上，Non-Flit Mode 的 DLLP 由 1 B Type + 3 B Info + 2 B CRC 组成，Flit Mode 时 DLLP 由 1 B Type + 3 B Info 组成，没有独立的 CRC。

支持的 DLLP Type 上，Flit Mode 下不支持 Ack/Nak，废弃了 MR-IOV 相关的 DLLP，新加了链路管理类的 LM DLLP 及 NOP2 DLLP。

发送模式上，DLLP 不能独立发送，而是随 Flit 发送且发送 DLLP 之前不需要 SDP 等 Token。

五、电源管理

电源管理过程理解

454. 设备是否可以只支持 ASPM L1 但不支持 L0s？

可以。

455. RC 是在进入 L1 之后拉高 CLKREQ#来进入 L1.2 吗？

是的。准确说不是拉高，是设置 HighZ 高阻。

CLKREQ#是一个 OD（Open Drain，开漏）信号，两边同时控制，可以选择控制为拉低或高阻。两边有一方拉低就是低；当两边都是 HighZ 时，即没有驱动时，外边电路会上拉，CLKREQ#就是高，两边就会检测到高。

Host 设置 CLKREQ#高阻之后，如果 EP 不想进 L1.2，则 EP 会把 CLKREQ#拉低，最终无法进入 L1.2；如果 EP 也想进入 L1.2，其会设置 CLKREQ#为高阻，最终进入 L1.2。

456. 追：接 VIP 进行测试时，RC VIP 自动控制进入了 L1，之后手动拉高 CLKREQ#。但由于 EP 反应晚于 VIP，EP 会拉低 CLKREQ#导致 VIP 未进入 L1.2 并退出 L1。若想进入 L1.2，是否需要反复、持续手动拉高 CLKREQ#？

CLKREQ#应该使用 tri 三态信号，拉高不要 force 1 而是 force z，并设置驱动强度为 Weak Pull Up。

457. USP 处于 ASMP L1 状态，此时让 USP 进入 $D3_{Hot}$，USP 会立刻开始退出 ASPM L1，还是要等接收到协商进入 PCI L1 的 DLLP 包才开始退出？

我理解的是立刻退出。因为配成 D3 后，USP 是协商发起者，有 DLLP 包要发就会退出 ASPM。

458. USP 是 ASPM L1 状态，DSP 是 PCI L1 状态，那 USP 退出 ASPM L1 会导致 DSP 退出 PCI L1 吗？反过来，USP 退出 PCI L1 会使 DSP 退出 ASPM L1 吗？

ASPM PM 是硬件控制下的链路上的电源状态，需要链路两端协商；PCI-PM 是软件控制下的设备电源状态，只跟本端有关。对链路而言，只有 DSP 可以发起进入 ASPM L1 请求，DSP 及 USP 均可以请求退出 ASPM L1。

至于设备和链路 L1 是否同步的问题，需要进一步讨论。

459. Switch 的 DSP 可以向 RC 发 PME_Turn_Off 吗？

不能，PME_Turn_Off 只能从上往下发。

460. 在开启 ASPM 的情况下，什么条件可以触发 EP 发出 PM_Active_State_Request 包?

EP 自己有个 Timer，记录处于 Idle 的时间，如果 Timer 超时了，EP 自动发起 PM_Active_State_L1_Request。L0 和 L0s 各有一个 Timer，一般 L0s 的 Timer 配的值更小，谁先到就先进哪个状态。

一般也可以单独配置使能 ASPM L0s 和 L1，也可以配置是直接 Idle 进去 L1 还是 Idle 先进 L0s 然后再进 L1。Synopsys 公司的可以是 Timer 计数进去，也可以拉 app_req 信号进去。

461. PCI-PM 电源管理方法中，通过软件调度 RC 写 EP 配置空间相关寄存器为 D1/D2/D3 就会使得 EP 与 RC 协商进入 L1 吗？

是的。

462. 现在软件还用不用 PCI-PM 了？

在支持休眠唤醒的设备，做 D0 -> D3 低功耗的时候会用到，服务器一般不用。设备关机流程中可能会用到 PCI-PM，该观点需要进行确认。

463. PCI-PM、ASPM 这两个标准该怎么理解？

PCI-PM 是符合 ACPI 标准的、系统层面进行的电源控制，ASPM 是 PCIe 协议的、PCIe 硬件本身的功耗管理。PCI-PM 相当于强制进入的，而 ASPM 是硬件自动检测的，进一步省电模式。

464. 是不是 RC 没有 PCI-PM，只能发给 EP 或者 Switch?

RC 的配置空间也有 PM 相关的状态寄存器和控制寄存器。PM 需要从 RC 开始一级级往下传。

465. 如何让 EP 发出 PME Message 请求？

问题补充：直接配置 EP 的 Power State 为 $D3_{Hot}$ 然后打流未出现 PME 且触发了 AER 报错。

设置 $D3_{Hot}$ 只是使链路处于 L1 状态，打流只是发起 L1->L0 的转换，只有 L0 时才能发送 PME。

PME 是系统发的，是 EP 设备请求主机唤醒时发起的，需要看下 EP 的唤醒条件，比如网卡收到特殊的唤醒包之类的。对 EP PCIe IP 而言，设计需要知道什么时候触发，以及如何触发。有种方式是 Vendor 自己设计寄存器，通过写寄存器触发硬件信号（例如 pm_pme），该信号拉高送给 EP Controller 并进一步触发 EP 给 RC 发 PME Message。可以看下系统里有没有设计类似的寄存器。

466. PCIe Switch 的 DSP 能生成 PME_Turn_Off 给 EP 吗？

可以。

467. LTSSM 跟 Link PM State 是一回事吗？硬件没做电源域划分 LTSSM 能支持 L2、L3 吗？

LTSSM 中的 L2、L3 跟设备 PM State 不完全是一回事，但是存在对应关系。严格遵循协议的话，不同 LTSSM 低功耗状态下对应不同 Power State 下电源域的开关。如果后端不想划分这么多电源域，在不掉电的情况下，设备可以谎称自己进入了某些电源状态并于对端进行相关协商，使得链路 LTSSM 进入相关低功耗状态。

468. 在 D0 Uninitialized 状态可以发 Type 1 Cfg 包吗？

设备处于 D0Unitialized 状态时，设备未被系统软件使能，此时只能接收 Cfg 请求，但不能主动发包。没有 Credit Init 之前不能发包，且想发包需要处于 D0 Active 状态。

469. PCIe EP 进 $D3_{Cold}$ L2 后重新上电并复位了 EP 配置空间，RC 会重新枚举这个 EP 吗？

不会重新枚举。根据驱动的做法，配置空间的值在 Probe 结束时会保存在 EP 本地，一般是 resume 函数中取重新初始化。再说，一般是设备驱动想要休眠才去置的 D3 状态，进 D3 状态后自己想唤醒只能通过驱动来操作了。

470. 如果在 Non-Posted 请求发出后、其 Cpl 包回来之前把设备置于 $D3_{Hot}$ 状态，这个 Cpl 包还能正常返回吗？是不是进 $D3_{Hot}$ 的条件就会等所有 inflight 的都结束了才能进？

如果是单个 Function 的 EP，Cpl 回不去了。如果是多个 Function 的 EP，有些 IP 是能返回 Cpl 的。

471. 那等从新到 D0 再回吗？如果一直不回会出现什么情况啊？

一直不回到 D0 就无法发包，会阻塞，时间过长对端会超时的。

472. 在 Switch 中 USP 的链路状态一定比 DSP 更活跃吗？可以 USP 在 L1、DSP 在 L0s 吗？

不太可能 USP 处于 L1、DSP 处于 L0s。好像是 USP 比 DSP 更活跃，因为都是下游先发起这个动作，所以下游先进低功耗。

473. 需要达到什么条件 Linux PCIe 标准驱动才会去执行配置 EP 进去 D3 低功耗的流程？

驱动里面一般是做挂起操作会设置 D3 状态。

Windows 环境下，在设备管理器里 Disable 设备会使设备进入 D3。

474. 是不是 DSP 要先进入 $D3_{Hot}$，EP 才能发起 PCI-PM L1 握手？

应该不是。RC 配了 EP 的 PM Statues 为 D3 后，EP 会发出 PM_ENTER_L1 DLLP，RC 收到后再回复 Ack，然后才一起进 D3。

475. 对于 Switch 的 DSP，然后 VIP 作为 EP 发起 PCI-PM L1 请求，这个时候 DSP 应该是 D0 还是 $D3_{Hot}$？

跟上文相同，DSP 是 D0，两边协商完成才会一起进 D3。

476. D0 能进入 ASPM L1，不能进 PCI-PM L1 吗？这两个有什么区别？

从 D0 可以进 PM L1，但 PM 需要由 RC 发起。ASPM 就是 D0 的功耗管理，PM 是 D1/D2/D3 的功耗管理。至于进 PM L1 后是否还在 D0，可参考协议上的对应关系。

477. ASPM L1 和 PCI L1 的功耗一样吗？还是说哪个更省电？

我理解 ASPM 是硬件自发的行为，不需要软件干预；L1 更偏向于被更上层软件操作下的行为。在底层 PHY 的行为是一样的。PHY 不区分因为那种原因进 Low Power。

478. PCI-PM 一般的使用方式是怎么样的？以一对一的 RC-EP 为例，在系统 Suspend 时，是否为 RC 写自己 PM CSR bit[3:0] 为 D1/D2/D3、然后 RC 写 EP PM CSR bit[3:0] 为 D1/D2/D3？RC 和 EP 是否可以自己写自己的 PM CSR bit[3:0] 为 D1/D2/D3？

一般是 RC 写 EP 的 PM CSR，不涉及 RC 写自己的 PM CSR。PCI-PM 机制是 EP 的 D 状态改变后发起链路进 L1 的，RC 写自身的这个寄存器没有什么作用。

479. 主机将 EP 功耗状态配置成 $D3_{Hot}$ 或 $D3_{Cold}$ 之前，会主动将 Command 寄存器中的 Bus Master En 置为 0（即关闭 EP 发起的 Mem 请求）吗？

应该不会。

480. 如果 PCIe EP Function 从 D0 转为 $D3_{Hot}$ 前主机不关闭 Bus Master En，那么 EP 一直发 MRd 请求，驱动软件如何确保 Outstanding Completions 已经终止呢？

引文：As part of this quiescence process the Function's software driver must ensure that any outstanding transactions (i.e., Requests with outstanding Completions), are terminated prior to handing control to the system configuration software that would then complete the transition to $D3_{Hot}$.（PCIe Spec 中 $D3_{Hot}$ 相关描述）

一般设备没什么事要挂起的时候才会被配成 $D3_{Hot}$ 进入省电模式，如果 EP 一直发 MRd，Sycle 驱动是不会进低功耗状态的。一旦进入 D3，EP 是发不出 MRd 的，只能接收 Config 和 Message。

回到上述 PCIe Spec 中的描述，此处说的是进 D3 之前要保证 Request 都收到 Completion。这步操作是芯片对应 PCIe Function（EP）的驱动软件保证的，驱动做的就是执行相关的 Supend 函数，写 D3 状态。

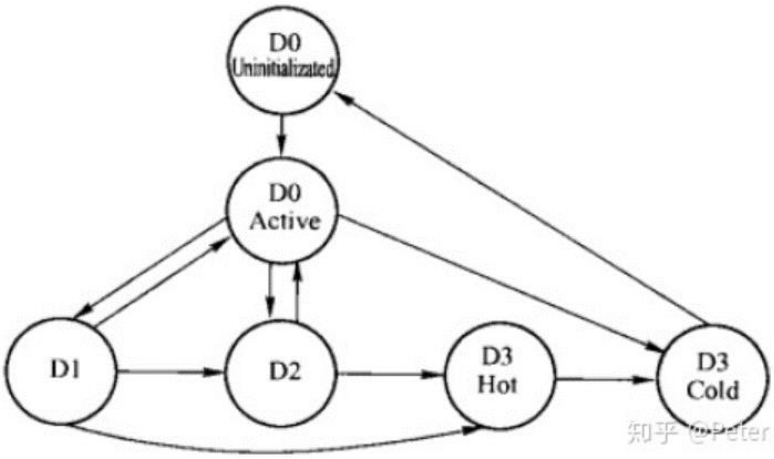
481. 当前设置 PM CSR 为 D3 直接进入 L1 了，如何操作能进 L2？

PCI-PM 配置只能到 L1，能否进 L2、L3 取决于 PCIe 硬件状态，需要关 V_{core} 电压：Main Power V_{main} 掉了后进 L2，全 Power V_{aux} 掉了后进 L3。

482. ASPM Support 和 Enable 这两个字段之间是怎么搭配工作的？

Support是支持，Enable是使能，支持可以不使能，使能前提是支持。Support不可配，Enable 可配，Host 枚举时看的是 Support。

483. 这个 PM 状态机是软件控制还是硬件控制？



软件控制。

484. Power State 只有 2b，是如何区分 D0 中两个子状态的？

不区分，都是 D0。

485. 一个 EP 设备不支持 ASPM 但是 lspci 看着被写成 D3 了，EP 的 LTSSM 不停在 L0 及 L1 之间切换，可能是什么原因？

被写 D3 后 EP 就会进 D3 L1，EP 被唤醒就会回到 D0 L0，但唤醒后未重新写 D0，就会再次进 L1。

486. ASPM 和 PCI-PM 都包括哪些低功耗状态？

ASPM 有 L0、L0s、L1；PCI-PM 有 L0、L1、L2/L3 Ready、L2、L3、LDn。

487. 请问 L0、L1 是链路的电源状态还是设备的电源状态？

L0、L1 是链路状态，D0、D1 是电源状态。

488. Power Budget Capability 中的寄存器如何使用？

看 Spec。

489. PCIe Switch 下游端口如何从 L23 ready 状态退出到 L0 状态？

需要硬件做总线上的处理，不能从 L23 ready 直接回到 L0。可参考 PCIe Spec Figure5-1。

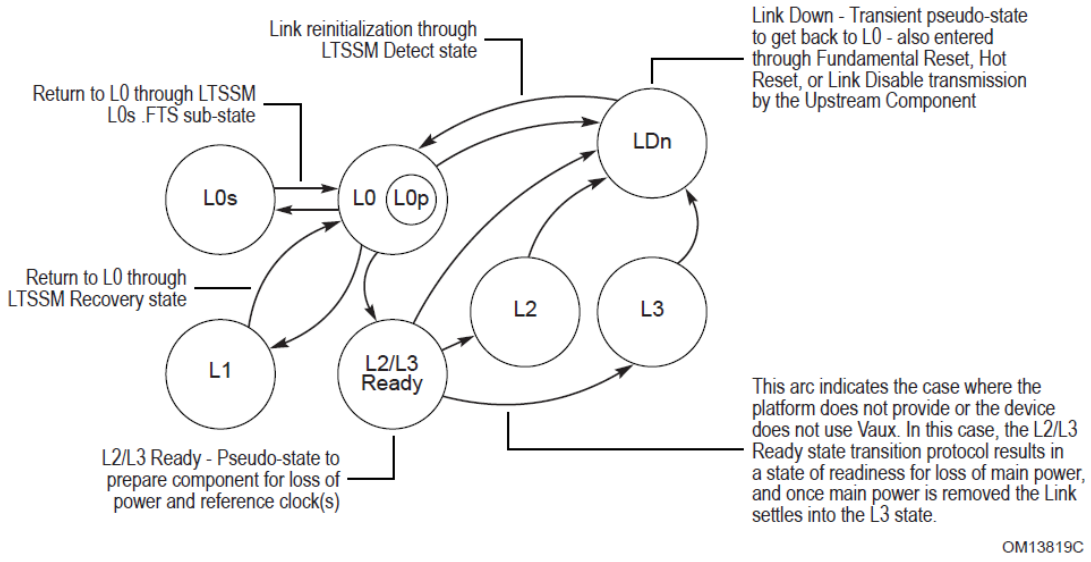


Figure 5-1 Link Power Management State Flow Diagram

490. EP 发 L1 Request 是纯硬件的行为吗？软件能否控制？

两种机制可以生成 L1 Request，硬件自发生成的是 PM_Active_State_Request_L1，软件控制生成的是 PM_Enter_L1。若需硬件自动发起，配置寄存器开启 ASPM 即可。

491. 如果 Switch 没有 DSP，是否说 USP 取决于 DSP 进入低功耗电源状态的条件一直满足?

不是，Switch 没接 DSP 但是 DSP 还是有的，这种情况下 Spec 说把 Switch 当成一个 EP 看就可以了，EP 没有数据的时候就可以进入低功耗了。

492. L0s 状态下是不会发包括 FC 流控在内的任何 DLLP 吗？

应该是每过一段时间发 FC。

493. PCIe L1 发起请求的是 RC 还是 EP 呢？

L1 的 PM Request 由 EP 直接发起，但此前需要 RP 发 CfgWr 来配置 EP 的 PM CSR 寄存器。

494. Device 被配置为 D1 后想回到 D0，需要 RC 还是 EP 去配？

RC 负责配。

495. $D3_{Cold}$ 时 Link 只能为 L2、L3 吗？

D3Cod 时设备已经没电了，PCIe Link 主电源肯定没电了。此时如果 Slot 有 Aux 辅助电源就是 L2，没有就是 L3。

496. 如果只支持 $D3_{Hot}$，是不是就不支持 L2、L3？

$D3_{Cold}$ 不需要支持，断电就是 $D3_{Cold}$。就是 L23 Ready，然后断电。

497. L0p 时没有 Active 的 Lane 是处于 EI 状态吗？

问题补充：看协议上的描述，PLL、驱动电平之类的都未写明需要关闭，只提及进入 EI 保持 DC 共模。

我理解 L0p 只关闭了 Tx PHY，其他都没有关。Rx 方向也没法关，PLL 和 CDR 都要用，所以省电省的应该就 Tx PHY 这部分。

Tx 主驱动和 Serial 电路不工作也能省不少电，Tx 从 EI 恢复很快；Rx 因为恢复慢，在 L0s 不会关。PCIe 和 PIPE 协议只定义不同 Power Mode 的 Recovery 恢复时间，不同 Vendor 的 PHY 实现方式相似，但具体模块的开关各有不同。Synopsys 公司 Databook 有描述其 PHY 在各种 Power 模式时内部模块工作情况。

498. RC 和 EP 都可以发请求进 L1 吗？

都可以，可以主动进也可以被动进。进 L1 的过程可以理解为双方都没有数据要发送，链路闲置一定时间后，Tx/Rx 各自进入了 L0s。在打开 ASPM 的情况下，硬件自动判断进入 L1。

499. 设备端 ASPM 进入对应子状态，需要跟其他 IP 关联吗？还是只是 PCIe 这个 IP？

都可以。可以跟 SoC 上的其他 IP 进行联动，做系统级低功耗方案。

唤醒

500. 唤醒有哪些方式？

可以通过 Beacon 或者 WAKE#引脚唤醒。

501. 将 PCIe 从 L2 低功耗状态唤醒有哪几种方式？

有两种方式将 PCIe 从 L2 低功耗状态唤醒：

WAKE#，是一种 Sideband 信号。

Beacon，是一种 30 kHz ~ 500 MHz 的低频 In-band 信号。在 L2 状态时，主电源及时钟都没有，但 Beacon 电路使用的 Aux 电源仍然保留。UPS 发送 Beacon，DSP 接收 Beacon，方向不能反。DSP 检测到 Beacon 后退回到 Detect。

502. 单个 RC 下挂了多个 Device，这些 Device 的 WAKE#信号线与在一起，当 Host 检测到低电平时如何获知是哪个 Device 拉低的以便退出 L2 对其供电？

不存在问题所述情形。RC 挂多个 Device 时，多个 Device 并非直接挂在了 RC 下，而是挂在了 Switch 下，实际和 Device 对接的是 Switch 的 DSP，和 RC 对接的是 Switch 的 USP。

503. Wake Up 功能应该如何实现？

问题补充：看协议进了 $D3_{Cold}$ 后主电源断了，只有辅助电源，IP 提供一个接口，该由谁来触发 Wakeup 呢？

取决于设备。对于 USB Controller，插入 USB 设备就有可能唤醒。对于网卡，需要智能网卡在辅电下还能工作，这时候在 BMC 下发命令，有个物理引脚来唤醒。

504. 所以 RC 先给与该 Device 对接的下行 Port 口供电？因为 Switch 知道是哪个 Device 要退出 L2？

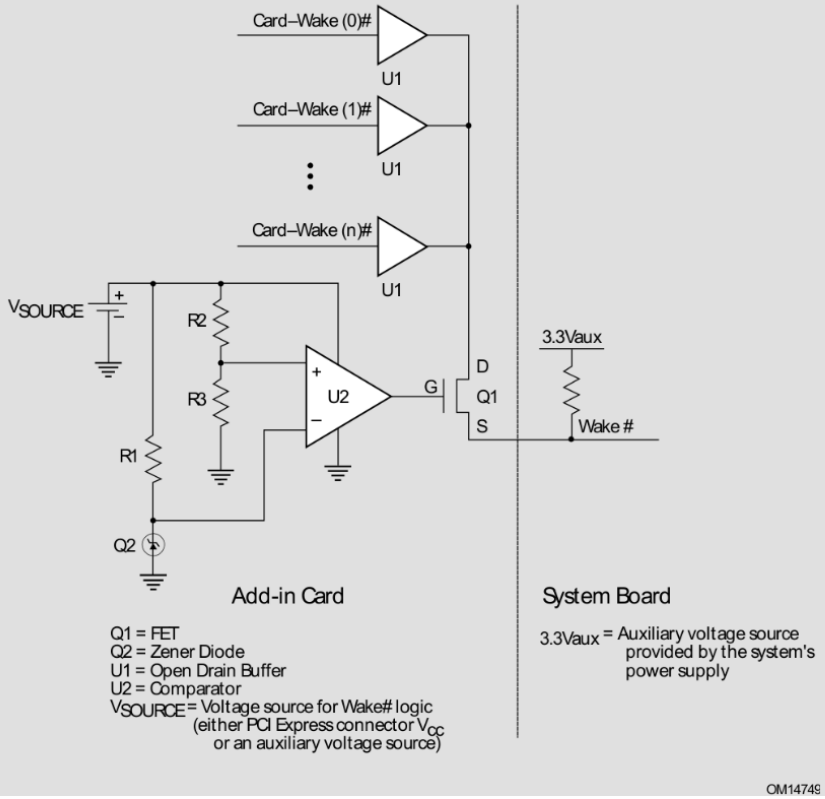
具体要看 Host 的设计。如果默认接口带有 WAKE#信号的话，可参考 CEM 手册 WAKE#连接的例子，每个 PCIe 插槽都有一个 WAKE#线练到 Host 主板的电源控制模块。

Example WAKE# Circuit Design

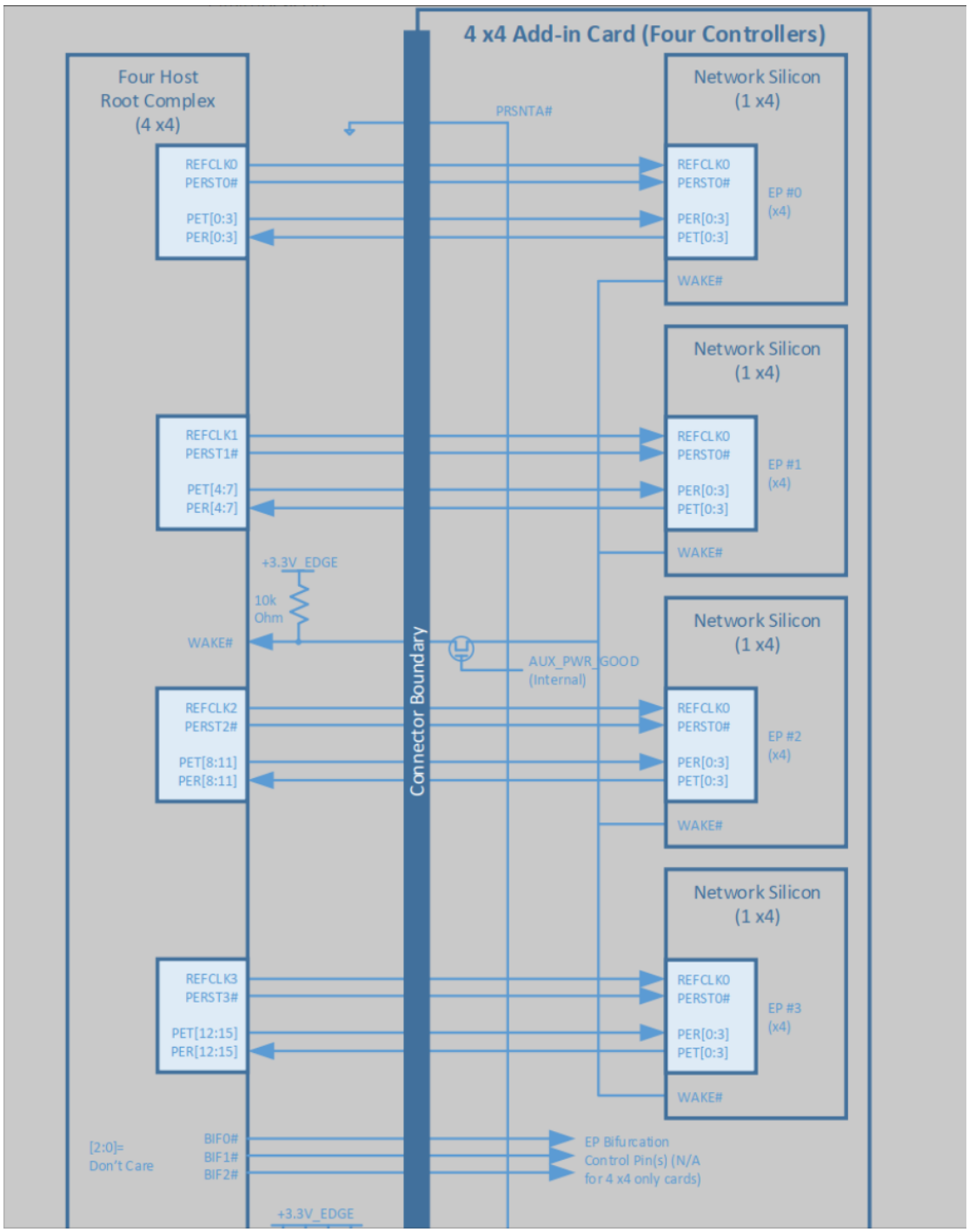
The following diagram is an example of how the WAKE# generation logic could be implemented. In this example, multiple PCI Express functions have their WAKE# signals ganged together and connected to the single WAKE# pin on the PCI Express Add-in Card connector.

The circuit driving the gate of transistor Q1 is designed to isolate the Add-in Card's WAKE# network from that of the system board whenever its power source (V_{SOURCE}) is absent.

If the card supplies power to its WAKE# logic with the PCI Express connector's +3.3V supply (i.e., it does not support wakeup from $D3_{cold}$), then all WAKE# sources from the card will be isolated from the system board when the Add-in Card's +3.3V rail is switched off. Add-in Cards that support wakeup from $D3_{cold}$ have an auxiliary power source (+3.3Vaux) to power the WAKE# logic which maintains connection of these WAKE# sources to the system board's WAKE# signal network even when the Link hierarchy's power (+3.3V) has been switched off.



我看一般不连这个 WAKE#，就见网卡会连这个 WAKE#。给到 BMC，BMC 再对电源做控制。多个线或在一起，唤醒整个系统。



505. WAKE#引脚是 Output，在 RC 模式下 PCIe 控制器怎么通过 WAKE 引脚唤醒呢？

问题观点有误，WAKE#不是输出信号，而是 OD 信号。跟 CLKREQ#信号原理类似，该 OD 引脚既可以检测外部输入的 WAKE#信号状态，又可以输出 0。

WAKE#默认是上拉的。在 L2 时候，Link 双方看到的 WAKE#信号电平都是 1. 当任何一方想要退出 L2，就主动拉低 WAKE#，然后对方就可以看到外部的 WAKE#被拉低了。就知道对方想要要退出 L2，从而唤醒。

实际应用中，也有不用 WAKE#做唤醒的，而是通过板子上的 CplD 芯片处理金手指的 Sideband 信号，但不确定该做法是否符合标准协议。

506. CPU 是如何知道 WAKE#引脚高低电平的？

就是一个中断。可以把 WAKE 管脚理解为一个 CPU 的输入管脚，同时还有输出 0 的功能。

507. RC 怎么知道是哪个 Device 拉低了 WAKE#呢？

又不是 MCU，不需要知道哪个唤醒的，不需知道是哪个设备拉的。现在休眠没做到那么细颗粒度，是整个系统级休眠，WAKE#直接拉起整个系统和所有设备。当有设备 WAKE#拉低时，不想退出 L2 的 Device 也会被动退出。一共没几个设备，想单独控制可以靠其他机制。

六、系统架构

中断（INTx/MSI/MSI-X）

508. MSI 和 MSI-X 是什么？

MSI 和 MSI-X 都是消息信号中断。它们虽然叫 Message Signaled Interrupt，但它们不是 Message，而是带有 1 DW 数据载荷的一笔存储器写事务。

MSI 能力结构中定义了消息地址（即接收中断的存储器地址，32 bit 或 64 bit）、消息数据等，不同的消息数据代表不同的中断。MSI 最多支持 32 个中断，这 32 个中断对应唯一的地址。

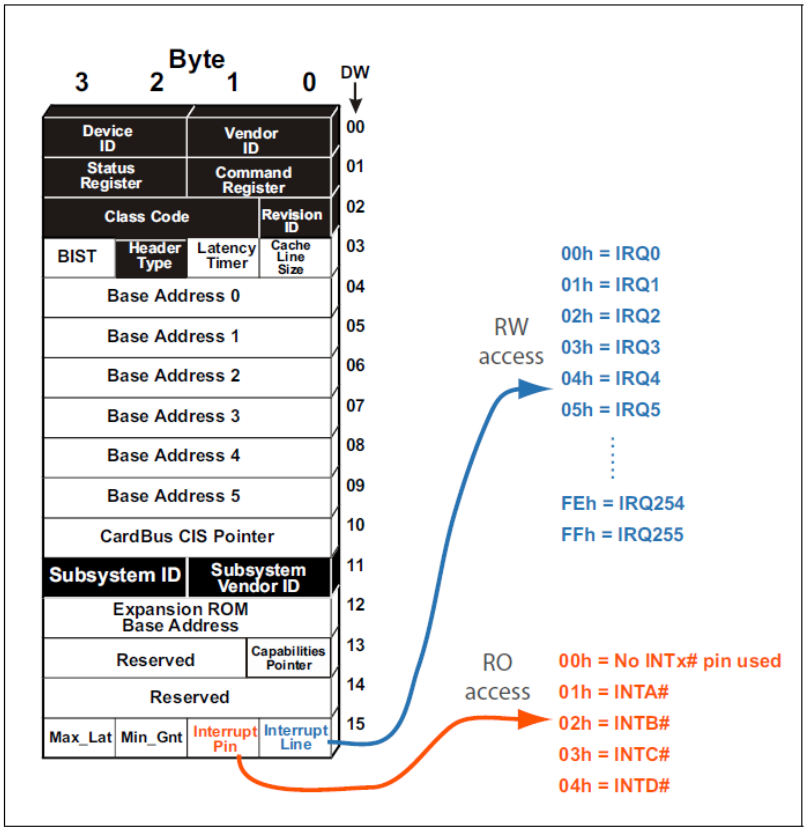
随着系统的发展，单一消息地址及最高 32 个中断的现状无法满足需求。MSI-X 对 MSI 做了扩展，最多支持 2048 个中断，且每个中断可以对应不同的地址。为了实现以上功能，MSI-X 能力结构中不再指定单一的消息地址，而是替换为 MSI-X 表地址，这个表中每一个 Entry 对应控制一条中断。

509. MSI-X 是如何工作的？

简单而言，在 EP 端，MSI-X Table 的位置是 EP 决定的，RC 只需配置 MSI-X Table Entry 的 Addr 和 Vector，至于 EP 如何触发 MSI-X，可遵循 EP IP 的设计文档；在 RC 端，软件直接配置 GIC ITS 相关映射表和寄存器，来监控 MSI-X。

510. Interrupt Line 是干什么的？是代表这个设备用哪个中断信号吗?

Figure 17-5: Interrupt Registers in PCI Configuration Header



PCI 是通过引脚触发中断，通过 Interrupt Pin 来选择哪个 INTx 中断。PCIe 中通过 MSG 来传中断。

511. INTx 只能由 Upstream Ports 发起，MSI、MSI-X 有这个要求吗？

PCIe Base Spec 里未限制 MSI、MSI-X 只能由 USP 发起。

理论上，MSI、MSI-X 只是一个 MWr，MSI、MSI-X TLP 跟常规 MWr TLP 没有任何区别，RC 可以发 MSI、MSI-X 到 EP，只要 Message 地址落在 EP BAR 范围内就可以了。EP 端需要有相关逻辑或 GIC 来处理这笔 MSI、MSI-X，才算是有效的 MSI、MSI-X、才能触发 EP 端的中断，否则也只能是一笔普通的 MWr。

512. 在 RC 端，MSI 写跟常规的 MWr 有何区别？

在 TLP 层面，MSI 中断的写与常规 MWr 没有区别。RC 收到之后，处理方式上有些区别。虽然 MSI 写也是挂在 RN-I 节点上，但它和普通 MWr 的出口不同，MSI 是给到 GIC，常规 MWr 是给到 Memory。

在 x86 上，MSI 写请求到了 HIF 应该就算结束了，HIF 会根据 MSI 写数据去设置某个 APIC，由 APIC 来响应这笔写请求。

513. MSI 跟常规 MWr 在排序上有什么要求吗？

MSI 与常规 MWr 也要遵循 PCIe 排序规则，根据 x86 Spec，MSI 写一般是普通 MWr。不过有的设计认为 MSI 跟排在其后的常规 MWr 没有关系，认为 MSI 可以被超越，并做了开关让软件人员自己去选。

514. RC 内是在什么时候识别 MSI 还是常规 MWr 的？

写请求在过了 SMMU 的 VF-PF 转换后，根据写请求的地址范围来判断是 MSI 还是常规写，地址范围寄存器软件可配。Synopsys 公司的 IP 还会判断写地址是否 Cacheable，必须是 No-Snoop 的才能去往 ITS 然后去往 GIC，否则 MSI 跟常规写请求一样走相同出口去往 CMN，最终还是路由给 GIC。

上述说法是 LTIM 和 SMMU Inlined 集成方式下的处理，否则控制器也不感知内存属性。另外 ITS 和控制器本身有两种集成方式，上不上总线取决于用哪种集成方式。MSI 的地址属性也是 Device 空间，也是内存地址，内存地址中的某段特定地址表示 APIC MSI。

515. RC 的 MSI-X 检测一般是软件实现还是硬件实现？

硬件实现。

516. RC 端是如何检测 MSI 中的地址的？

PCIe Spec 中未规定具体的 MSI 硬件实现方式，根据 RC 端有无 MSI Receiver、GIC 是否支持 MSI，大致有下述两种方式：

- ✓ Synopsys IP 内有 MSI Receiver，其在 Port Logic 里有配置监控地址的编程接口。如果 RC 端集成了 Synopsys 的 MSI Receiver，此时地址可随意配置，只需确保 EP 端 MSI CAP 里配置的地址跟和 RC 端 Port Logic 所配置的 MSI 监控地址相同即可，由 IP 来监控 MSI 中的地址。该方法不依赖于 GIC 对 MSI 的支持，但需要在 RC 端增加额外的逻辑来检测 MSI。
- ✓ 如果 RC 端的 GIC 支持 MSI，EP 端 MSI 可以直接对接 RC 端 GIC 地址。即把 RC 端 GIC GICD_SETSPI 寄存器在 PCIe 域内的地址配置进 EP 端 MSI CAP 内。该方法无需在 RC 端做 MSI 接收逻辑，但使用不当会有安全问题：若未接入 GIC ITS LPI，考虑到 GIC 及 MSI 触发寄存器为同一个、没有做到隔离，PCIe EP 可以触发任意可用的中断号。

GIC ITS 在 GICv3 才引入，如果使用了 GICv2 及之前版本，建议使用第一种方式，以避免 GIC v2m 机制做不到中断隔离引发的安全问题。

517. RC 端是如何监测 MSI-X 中的地址的？

RC IP 应该无法检测 MSI-X 的地址。MSI-X 地址是 Host 环境配置的，需要 Host 环境自己做 MSI-X 地址检查。RC 配置该地址时，可配置成 GIC 地址由 GIC 监控 MSI-X 触发中断，也可以配置成 DDR、IRAM 等其他不被 GIC 监控的地址。

对 Synopsys PCIe IP 而言，默认没有 MSI-X Receiver。如果 RC 端有 GIC ITS，可以基于 GIC ITS 来实现 MSI-X 的硬件监控；如果没有，则需要自己集成 Application Logic 来实现该功能。较新版本的 Synopsys RC IP 支持可选的 MSI-GIC DVSEC，可参考设计。

518. RC 给 EP 发 MSI、MSI-X 有实际场景吗？有何意义？

一般来讲，RC 很少有需要 EP 处理的中断，但仍然有其应用场景。

当需要从 RC 端搬大块数据到 EP 时，优先是调用 EP 的 DMA。当 EP 没有 DMA 时，可用 RC 端的 DMA 往 EP 搬，搬完后通过 MSI/MSI-X 触发 EP 中断，以便 EP 快速处理。

此外，在一些片间互联场景下，RC 和 EP 的区别不大，两侧的 DMA 都会有读写。通过在底层驱动和用户层加入数据互联的驱动，来屏蔽掉底层 RC 和 EP 的区别，向上支持双向传输，根据实际应用业务在哪端发起来决定调用哪方的 DMA。

如果要用，需要改 RC 端的中断子系统，需要做各种 PCIe 重定向，尤其是在 Linux 下做定制，比较麻烦。

519. 如何实现 RC 到 EP 的 MSI/MSI-X 功能？

在 RC 端，RC 发起 MSI/MSI-X 方法跟 EP 类似，可①通过指定事件，比如写几个指定寄存器，触发硬件生成指定地址的 MWr，或②CPU 直接 ldr/str EP 端 MSI/MSI-X 指定地址。

在 EP 端，为正确解析出 MSI/MSI-X，可① EP BAR 指向 EP 的 GIC，或者 ② 在 BAR 内指定 Offset 上设置寄存器，写入对应寄存器自动触发 EP 内部中断。RC 发 MSI/MSI-X 到该 BAR/Offset，理论上就可以触发 EP 中断了。据传，英伟达的 PCIe 数据互联（非 GPU Direct）就采用了第①种方法。其对内用 NVLink，对外支持这种通用的片间互联机制。该方法已商用，稳定性有待提升。

520. MSI Capability 中消息数据寄存器的数据是由系统软件写入的，还是由 PCIe 设备负责更新？

是系统软件写的。

521. EP 想要使用 MSI/MSI-X 中断，需要上一层接的 RP 支持 MSI/MSI-X Cap 吗？

不需要，非必需，当作写操作转发就行。

522. GIC、APIC 是什么？

GIC 是 ARM 的中断控制器，APIC 是 x86 的中断控制器。

523. PCIe MSI 中断的地址是在该 Function 的 BAR 里面挑一段地址空间，还是在 BAR 之外主机重新分配地址空间？

一般 MSI 地址是主机中断控制器的地址。

在 ARM 下的一些实际应用中，Host 会把 MSI/MSI-X 地址直接写到 EP 端相关 Cap 寄存器内，EP 直接写这个地址就可以，在 Host 侧有两种实现方式，一种是 RC Controller 自己监控这个地址(Synp 做法)，也可以是中断控制器来监控的地址。

524. 根据 Xilinx PCIe RP 文档对 MSI 描述中，其支持 64 Vector 的 Interrupt，根据协议 MSI 不应该最大支持 32 Vector 吗？这个 32~63 Vector 有何用处？

> ***Note:*** MSI Message Data [5:0] will always be decoded as MSI Message vector regardless of how many vectors are enabled at your Endpoint.
>
> When an MSI interrupt is received, the Root Port MSI Interrupt Decode 1 or Root Port MSI Interrupt Decode 2 register is set. If the Root Port MSI Interrupt Decode 1 or Root Port MSI Interrupt Decode 2 register is also set, the `interrupt_out_msi_vec*` pins are asserted. `interrupt_out_msi_vec0to31` corresponds to MSI vector 0 - 31, and `interrupt_out_msi_vec32to63` corresponds to MSI vector 32 - 63. After receiving this interrupt, the user application must follow this procedure to service the interrupt:

感觉是其想要超过 32 个中断，又不想用 MSI-X，就扩展出 32-63，配合其驱动也能用。

525. Synopsys PCIe IP MSI-X 要求所有 PF 和 VF 的 Table BIR、Table Offset、PBA BIR、PBA Offset 必须相同，如果相同该怎么区分呢？

Operational Limitations (General)

- Table BIR must be identical[2] for VFs and PFs.
- PBA BIR must be identical for VFs and PFs.
- Table Offset must be identical for VFs and PFs.
- PBA Offset must be identical for VFs and PFs.
- Table Offset must be aligned to 8 KB boundary. Recommended by PCI specification.
- PBA Offset must be aligned to 4KB boundary. Recommended by PCI specification.
- `MSIX_TABLE_SIZE` (used to size Table RAM) must be identical for all PFs.
- Sending MSI-X interrupt through the SII interface signals `ven_msi_req`, `ven_msi_func_num`, `ven_msi_tc`, and `ven_msi_grant` is not supported.

如果 EP 支持 MSI-X，需要在该 EP 每个 Function 的 Config Space 内都实现一个 MSI-X Capability，MSI-X Capability 内记录了 MSI-X Table、PBA 的 BIR 及 Offset，即位于该 Function 下哪个 BAR 内、在该 BAR 内的 Offset 是多少。

上图的意思是说，在实现 Synopsys iMSI-X 时，不同 PF、VF 的 MSI-X Capability 内记录的 MSI-X Table、PBA 的 BIR 或 Offset 应该保持一致。举个例子，不能在 PF0 中位于 BAR0、在 PF1 中位于 BAR1，也不能在 PF0 中位于 BIR=BAR0 的 Offset=0x0000、在 PF1 内位于 BIR=BAR0 的 Offset=0x8000。

由于 Host 在分配 BAR 时不同 Function 的 BAR 空间是不同的，所以即便不同 PF、VF 的 MSI-X Table/PBA BIR、Offset 相同，Host 仍然可以通过地址对齐加以区分。

526. MSI-X Table 要挂在 BAR 上，为什么不像 MSI 那样把地址数据放在寄存器中？

MSI 最多支持 32 个中断，地址数据直接放 Capability Register 寄存器组就够。

对 MSI-X 而言，单 Function 的 MSI-X Entry 数量最大支持 2048，每个 Entry 需要 4 DW，这样光 MSI-X Table 就需要 32 KB 空间。32 KB 空间全做成寄存器太大了，更适合做到 BAR 上的 SRAM 或 DDR 里，也便于 Host 访问或配置。

527. Synopsys PCIe Spec 提到 MSI-X Table 和 PBA 位于 Application Memory Space，这里的 Application Memory Space 是指 DDR 吗？

没有要求必须是 DDR。可以是 Controller 内部的 SRAM，或者 Controller 外部的 DDR。不管位于哪里，这块空间都应该是挂在 BAR 下的，确保 Host 能够访问到，以便 Host 对 MSI-X 进行配置。

528. 使用 Synopsys 的 iMSI-X 时，中断源是如何触发 MSI-X 请求的？

通过 DBI 接口去写 MSI-X Doorbell 寄存器，来触发 MSI-X 请求。

529. 如果 EP 要同时发送多个 MSI-X 中断，多个中断的仲裁是在 PCIe 外部实现的吗？

对 EP PCIe Controller 内部而言，DBI 一次只能写一个，不会同时到达多个 MSI-X Doorbell 写请求，不存在仲裁。在 PCIe Controller 外部 Application 层可能需要仲裁。

530. MSI-X 中断数量有什么影响吗？为什么我开 2048 个会被裁剪？

MSI-X 数量会影响芯片面积。在 EP 端，如果要用 SRAM 来存放 MSI-X Table，Entry 越多所需的 SRAM Size 就越大。在 RC 端，MSI-X 中断发到 GIC，GIC 查 LPI 中断，MSI-X 数量不同，所需的 LPI、Collect Table、Device Table Interrupt Translation Table Size 也不同，如果这些表未放在 DDR 里（一般是放 DDR 里的），同样影响面积。此外，还有一些其他相关逻辑门的增加。

除了上述原因，MSI-X Entry 数目被裁剪，也可能是由于 EP 产品上根本不需要那么多中断。

531. 为什么 MSI-X 最多只能 2048 个？

MSI-X Capability Structure 中有个 Table Size 字段，该字段只有 10b，最大也只能表示 2048 个 MSI-X Entry。

这里的 2048 是对单个 Function 而言的，如果支持多个 Function，那还能支持更多数目的 MSI-X 中断。

532. 配置 MSI-X 时一般开多少 Entry？

- ✓ 我用的场景比较单一，MSI-X Entry 数量很少超过 32 个。我们使用 MSI-X 代替 MSI，只是为了可以中断向量号不连续。
- ✓ 看设备需求。在服务器上 MSI-X 用得多，最大见过支持 256 个 MSI-X 中断的 NVMe 盘；在 PC 上用得不多。
- ✓ 一般而言，队列多 Function 多，对应的 MSI-X 数量就多，见过单 Function 最大支持 2048 个 MSI-X 中断，且多个 Function 总计支持 5000 多个 MSI-X。
- ✓ 这个可以问产品、问架构的人，但大概率他们也讲不清楚需要多少。如果需求不明确，此时留 256 个 Entry 稳妥一点，有些 WIFI 芯片需要 100 多个独立中断。
- ✓ 在 EP 端资源允许的情况下可以多开点 MSI-X Entry，用不用先放在那，实际可能用不了那么多，但至少确保数量不会少。届时，如果 EP 自身用不了那么多 Entry，EP 软件可以在 RC 来配 MSI-X Table 之前自行配置 MSI-X Entry 的个数。实际使用中 Host 可能由于中断号有限不会分那么多中断号给 MSI-X，到时候多个业务之间共享中断号。
- ✓ 可以把 MSI-X Table 映射到 EP 的 DDR 里，想用多少用多少，还不占面积。弊处在于，这种方法软硬件具体实现时可能存在难度，会损失一定的软件性能。
- ✓ 如果说的是在 RC 端应该给 MSI-X 留多少中断号，如果确定是一对一，那 MSI-X 中断数目跟 EP 保持一致即可。如果是要接 Switch，则需要尽量多留点，不确定会扩展出去多少设备。

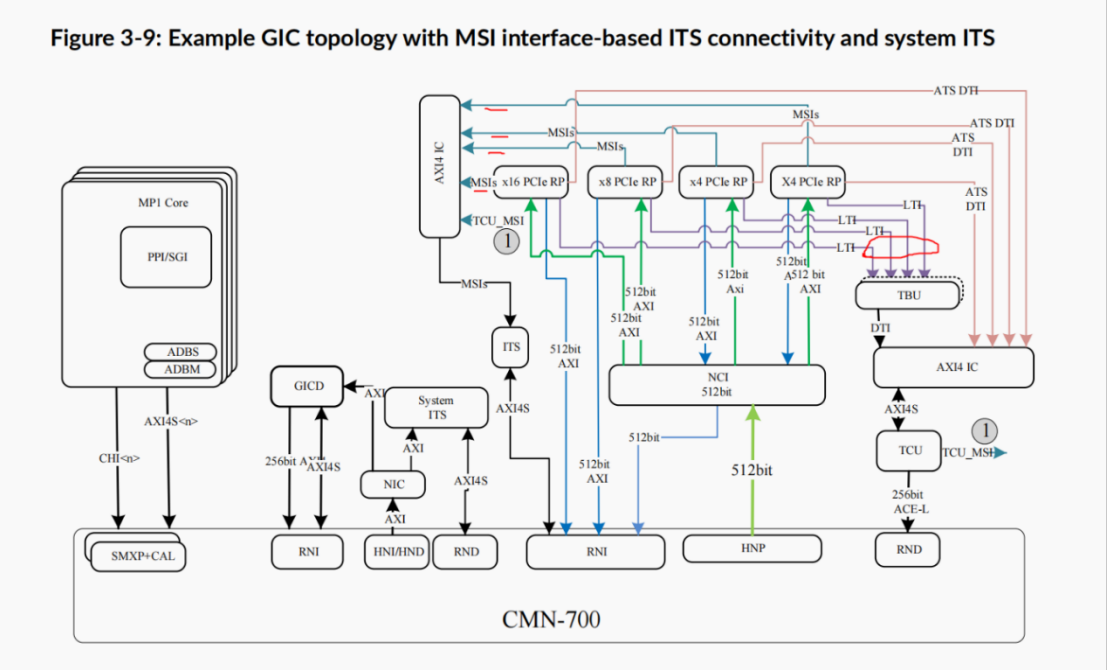
533. VF 跟 PF 是共享 MSI-X 数量吗？

PF 和 VF 的 MSI-X 没有直接关系，VF 的 MSI-X 是独立的。

534. 为什么 PCIe 读写都要发生 MSI 中断？

EP 的 DMA 模式多是采用中断的，一般 EP 发 MSI 给 RC，通知 RC 读写数据完成。

535. 在 RP 侧，Synopsys 公司的 Controller 和 ARM 对接，这个 MSIs 是不是也包含了 MSI-X 了，只是命名成了 MSIs？



是的，对 RC 来说不用区分 MSI 和 MSI-X。

536. RISC-V 能接收并解析 MSI-X 中断吗？

详细描述：比如 PCIe 作为 RC，收到 Remote Device 的 MSI-X 报文，之前用 ARM 的 CPU 可以接到 GIC600，现在用 RISC-V 不确定能否接收并解析 MSI-X 中断。

肯定可以。可以参考 GIC400 和 GIC600 的实现，GIC400 需要自己做一些转换逻辑来实现，GIC600 IP 原生支持。RISC-V 有 AIA 高级中断，转成线中断 PLIC 也可以收，至于 PCIe Controller 上的 AXI Master 经 Bus 到 RISC-V 之间是否需要做逻辑，具体要看 RISC-V 中断控制器了。

537. MSI 和 MSI-X TLP 的 Length 是固定 1 个 DW 吗？

是的，MSI/MSI-X 是 MWr，Payload 只有 Message Data，Length 固定 1 DW。

538. MSI 和 MSI-X 的 First BE 必须是 4'b1111、Last BE 必须是 4'b0000 吗？MSI 如果没使能 Extended Message Data，First BE 可以是 4'b0011 吗？

协议明确说了，First BE 需要设置为 4’b1111，即便 Message Data 只有 16b，但设置为 4’b1111 肯定是没问题的。除非有特殊需求，没必要设置 First BE 为 4’b0011。

539. CPU 如何区分收到的消息中断类型是 MSI 还是 MSI-X？

MSI、MSI-X 本质是 MWr，且地址是 CPU 来指定的，CPU 可以根据地址来区分两者。

540. 如果主机收到 EP 发来的 MRd 地址为 MSI 或 MSI-X 的地址，应该怎么回 CplD？

正常回。

541. MSI-X 发中断之后，CPU 要怎么操作才知道是哪个向量发的中断呢？

RC 可以通过 MSI/MSI-X Data 来知道查询哪个向量。如果 GIC 不支持 ITS、查不到 MSI/MSI-X Data，可以查寄存器 bit 位。

MSI 的话，RC 会注册一个 IRAQ Domain，可通过该 Domain 查询向量中断，MSI-X 类似。ITS 也一样，会有一个 IRQ Domain。

542. RC 收到 MSI Message 后怎么写 Memory，之后中断又是怎么产生的？

具体要看 RC 端的 CPU 架构，如果是 ARM 还要看 GIC 版本，有的 GIC 可以把 MSI 转为 LPI 写，有的只能自己处理把 MSI 转为 SPI 之类的中断线。

543. LPI 写是指什么？写 Mem 的动作是 RC 完成还是 GIC 完成？

LPI 是 ARM 定义的一套写 Mem 触发中断的机制，这个可以跟 MSI 对应起来。对于 ARM 架构，如果 GIC 支持 LPI，实际是 RC 把 MSI 转为 AXI Write 写到 LPI 对应的地址，从而触发中断。AXI 写时要注意，由于 ARM GIC 只支持 32b 的写，所以 AXI 写 Size 一定要是 2，即 4 B，如果使用 SNPS IP，一定要配置一下 SUB_BUS 参数以支持 Narrow 操作，不然发不出来。

544. MSI 中断过 MSI64 送往 ITS 时，如果判定地址匹配上了 ITS Target Address，MSI64 Master 口出来的地址会自动被偏移 16b 并在低位补上 0x0040 吗？

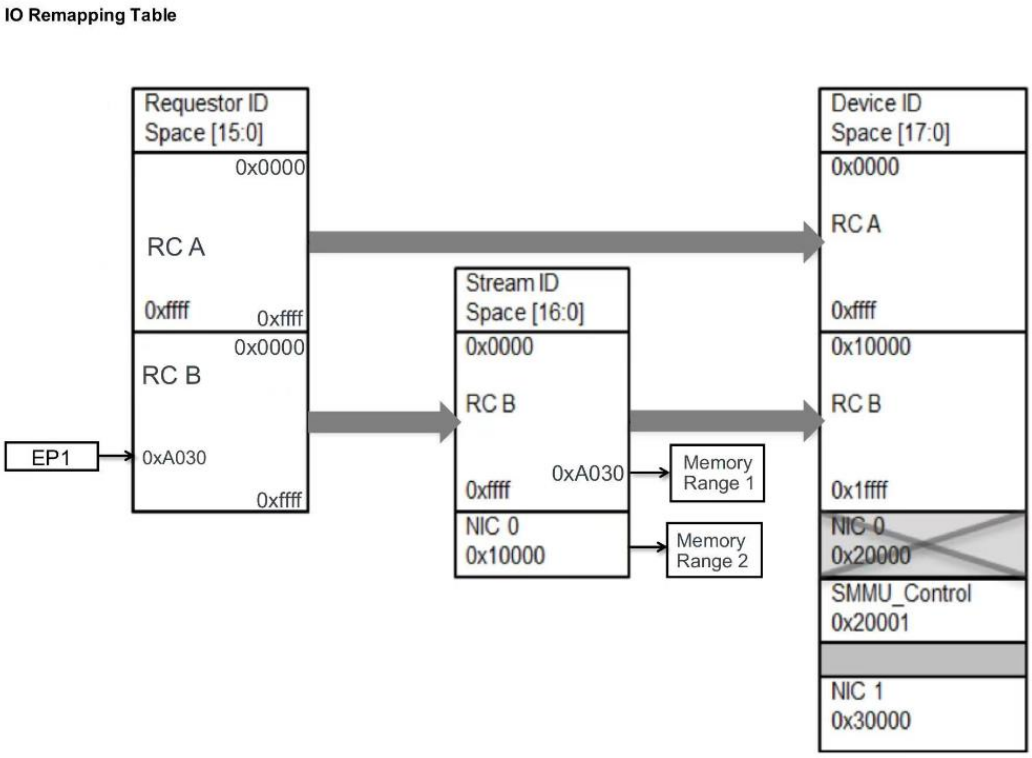
会。

545. Linux 的 PCIe EP 驱动中用 mmap 申请一片虚拟地址 DMA，只要文件超过 6 M 就会申请失败，需要调系统里哪个参数来解决该问题？

可参考： dma_alloc_coherent 申请内存用法和问题总结

546. 在 ARM64 系统中，LPI 如何依据收到的 MSI 中断来区分不同 Domain 下 Requester ID（BDF）相同的两个 EP？

靠 SMMU 来区分。Linux Kernel 支持 Domain，可以依据最前边的 Domain ID 来区分 BDF 相同的两个设备，并进一步根据 IO Remapping Table 映射为不同的 Device ID（16b Segment ID + 16b BDF），最终再给到不同的 ITS。



547. 系统中有多个 RC 时，ITS 和 RC 是一一对应的吗？

不是一一对应关系，多个 RC 可以对应同一 ITS。如果不同 RC 使用同一 ITS，在 DTS 会划分开。如果做成 RC-ITS 一一对应，需要在系统中加 CCI，保证 CPU 和 ITS 之间的一致性。

548. EP MSI 中断消息上报后没有被应答，EP 会按写事务超时机制重新上报吗？

背景描述：Reboot 后未触发设备的 PERST，Host CPU 还没注册中断就收到了 EP 发来的 MSI，Host 侧 CPU 对该 MSI 中断不做处理。

PCIe 写事务没有超时机制。EP 发出 MSI 后并不知道 Host 端 MSI 处理是否超时，除非自行设计判断逻辑。

上述问题为设计缺陷。正常情况下，在 Reboot 后设备会被 PERST 复位关闭 MSI，设备驱动需要重新 Load 才能开启设备的 MSI，在开启 MSI 之前设备是无法上报 MSI 的。

549. MSI 支持 32 个中断是否意味着一个 MSI 中断地址对应 32 个中断向量？从应用角度讲，不同的向量号之间有什么区别？

是的，一个地址对应 32 个中断向量。不同中断号的优先级、对应的场景是人为赋予的，协议层面不同向量号之间没什么区别。

550. RP 只是接收 MSI，是不是 RP 不用配置 MSI Capability？

跟平台有关。一般来讲，MSI 是 EP 下的，只需要配置 EP。有的平台，比如 ARM 服务器，根据 SBSA 规范，RP 自己的中断也要走 MSI 这套软件流程来做自己的错误处理，RP 也要实现 MSI。

551. SRIOV 不同 VF 分配到的 MSI 地址可以是一样的吗？

不同 VF 可以分配相同的 MSI 地址，配好 IOMMU，根据 ID 区分不同的 VF，再转成不同地址出去即可。不论是否一样，都是硬件上的配置，对软件人员而言，入 IOMMU 之前都是 VA，出了 IOMMU 后必须保证地址不一样。

552. 不同 VF 的 MSI 地址一样的场景常见吗？

只见过一样的，没见过不一样的，都是 GIC ITS Translator 的地址。GIC Cap 的资源有限，MSI 到了 ITS-GIC 那里不关心地址，更关心 MSI 的 Payload 内容。

553. 该如何使能 VF 的 MSI-X 中断？

跟 PF 一样，每个 VF 都有自己的 MSI-X Capability，通过配置 VF 的 MSI-X Capability 来是能 VF 的 MSI-X 中断。

554. PCIe 的 MSI-X 中断消息要经过 SMMU 吗？

要经过 SMMU Stage2，因为 MSI-X 本质是写。

555. 系统中没有 SMMU 这个组件，LPI 如何区分不同 RC 下相同的 BDF？

如果没有 SMMU，那么只支持一个 RC，不会存在多 RC 下 BDF 相同的问题。

错误处理

556. Link Error 包括哪几种？发生 Link Error 时是如何恢复的？

Link Error，链路错误，是指连接两个 PCIe 设备的物理链路出现错误，通常在物理层检测出并传达到数据链路层。Link Error 包括以下几种：8b/10b 解码错误、framing 出错、符号/Block 锁定后失锁、弹性缓存上下溢出及 Lane 间消抖失败。

如果是在链路 Training 期间发生链路错误（非 Detect、Disable 状态），无需特殊操作，LTSSM 正常运转，各个状态的 Timeout 机制保证 LTSSM 回转到 Detect 状态进行 Retraining。

如果是在 L0 状态发生链路错误（frame error 等），可以由软件控制 LTSSM 由 L0 -> Recovery 进行链路恢复，或者重新训练。需要注意的是，出现链路错误后，下游设备是无法通过错误链路告知上游设备的，需要由错误链路上游的交换开关 USP 或 RC 上报错误，开启 Retraining。软件通过查验相关寄存器确认是否训练成功。

557. 在 EP 端主动注入 PARITY_SKPOS_ERR 错误，Host 收到后会挂死吗？

该问题答案不明确，需要实测。

一说根据注错的位置及 EP 的设计，这种错误可能在 EP 端就能够检测出来，即不会到 Link 上，也不会发送到 Host，如果是这种情况当然也就不会触发 Host 挂死。这种情况下，EP 端的 AER 机制可能记录一些错误信息。至于是否会触发 AER，则要看 AER Status 中是否存在该错误。据了解是不存在这种错误状态的。

如果这个 Error 能够到达上 Link 到对端，对于 Order Set 有续集而言，可以确定的是这个 Error 不会触发 Retry。如果 SKPOS Parity Error 意味着 Link 两端频率不一致、出现大量 SKP 无法识别，Host 端可能会出现 Elastic Buffer Overflow/Underflow，并进一步引发 Host 挂死。一旦挂死，就需要做系统复位重新 Link Training。

另有观点表示，SKPOS Parity Error 只是 Parity 问题，不影响 SKP Payload，也不会影响 Elastic Buffer，即是一种可纠正、不致命的错误，不影响链路上也无用数据的传输，也不会影响链路状态。

总结下来，需要对照 PCIe Spec 查看该 Error 属于哪一种类型，看起来会是 Receiver (Rx) Error，一般 Physical Layer 的 Link Error 都是归为此类。这个 Rx Error 在 AER 里面定义是 CE (Correctable Error)，EP 如果支持 AER 会在 AER CE Status Register 记录一个 Rx Error，并发送 ERR_COR Msg 给 Root Port，Root Port Error Statues Register 的 CE Bit 会设置起来。对于 x86 系统，通常会触发 SMI 调用 BIOS 代码来识别此错误并生成 Error Record 给 OS。系统不会挂，针对 CE 错误只是记录这笔错误而已。

558. Synopsys 公司 EP IP 如果发生 CE 错误，有寄存器能读出 CE 错误数量吗？

可以。Synopsys 公司 PCIe IP 支持有一组 Statistics and Error Injection Features 可以对各类型错误进行计数。

559. 什么情况下 PCIe 会报 Failed to Check Link？

热插 SSD 但是实际上没有 Link Up 的时候会报上述信息。以此类推。

560. log 显示错误已经被纠正，还用管吗？

取决于错误发生概率、是否影响实际业务等因素。偶发或不影响实际业务的可忽略。

561. AER 上报错误是走 MSI、MSI-X 中断方式吗？

EP AER 上报错误给 RC 有专门的 Message，不一定是 MSI。

对 EP 而言，若 EP 开了 AER，其检测到相关错误后会自动给 RP 上报一个 Error Message。RP 收到后，如果其也开了 AER，RP 会记录这笔错误到其 AER 寄存器中，由 RP 产生中断继续上报。从 RC 这边看，理论上 EP 送过来的 AER 就是一笔 Error Message 写操作，至于这个写操作怎么 Trigger 到 GIC，跟 RC IP 的实现、GIC 版本都是强相关的，完全取决于设计实现方式。

参考：The PCI Express Advanced Error Reporting Driver Guide HOWTO

562. 如何手动清理 AER 的 CE 错误？

CE 信息为：CESta: RxErr+ BadTLP+ BadDLLP- Rollover- Timeout+ AdvNonFatalErr+

setpci AER Statues bit 置 1，写 1 清零，哪个 Error+就哪个写 1。如果链路很差，清掉后会再次出现，可以写个循环读取这个 bit、清除这个 bit。

563. PCIe 设备起来后 RC 端 CPU 通过 lspci 查看有 Advisory Non-Fatal Error，可能是什么原因？

问题补充：采用标准 x86 服务器 Intel 的网卡未出现该错，自研网卡（Xilinx PCIe IP）出现该错。上电后系统刚起来还未传业务就报错了，一共几百个错，传业务期间错误数量未增加。

发送业务之前就报错了且发送业务之后错误数量未增加，说明报错跟业务没关系。猜测可能是枚举时 EP 报了错，比如枚举期间 EP 报了 UR。如果是 UR 的话，可能未以下原因：

- ✓ 可能是在枚举过程中访问到了不该访问的寄存器，比如函数访问了 A 地址，但是包对齐到了 B 地址，触发报错。
- ✓ 枚举时候会访问 Function 0-7，如果设备只有 Function0，访问 1-7 就会回复一个 UR。一般设备会将 AER 中的 UR 对应的 Fatal 设置为 0，即 Non-Fatal Error，在 AER 内就会记录为 Advisory Non-Fatal ERROR。正常服务器都会这么枚举，不是错误。如果是该问题引起，可以参考 Spec 6.2.3.2.4，7.8.4，将相关 Severity 位置为 1。

总之，一般仅在开机时才会出现的 Advisory Non-Fatal Error 不是什么大问题。如果不影响数据传输、不影响实际业务，可以忽略该错，在开机后相关 AER 指示位写 1 清 0 即可。

564. 有 DPC 和没有 DPC 的情况下，出现 UE 错误会有什么不一样的现象？

简单说就是主机死机和不死机的区别。

565. 怎么知道 Host 有没有 DPC？

可以查看是否支持 DPC Extended Capability。

566. lspci 看到的 Secondary PCIe 状态中的 lane_err 是指发送侧还是接收侧？AER 里的 rx_err 是指 EP 的 Rx 端么？两者什么关系？

lane_err 和 rx_err 都是接收侧的。

567. AER 里面 Header Log 是不是只有 RC Port 能记录第一条错误？如果有新的 Error 会把前面的记录覆盖吗？

568. dmsg 查看 AER 一直上报 rx_err，但 lspci 看有的会报、有的不报这个错，是什么情况？系统软件会在上报后清这个错误么？

可能是 CE 错误，上报完成后 BIOS 就把错误状态清掉了。至于有的报有的不报的问题，不报的可能是相关 CE 被 Mask 掉了。

569. RP AER 可以使用 Error Interrupt（ INTx/MSI）来上报错误吗？

问题描述：根据 PCIe Base Spec 6.2.6，RP 的 AER 可以通过 INTx/MSI 来上报，实际支持这样吗？

协议本身支持这种上报形式。EP 发生 AER 错误时可以通过 AER Message 或发中断来上报错误。在 ARM 架构上，RP 收到后上报的错误后可以直接送到 GIC，可以转换为 MSI 中断送到 GIC，也可以 SPI 转 LPI 再送给 GIC。如果直接 SPI 到 GIC，不符合 Kernel 流程，需要软件 Workaround 到 Kernel 处理 AER 的流程。

570. 假设 EP 报告 AER 错误，那么 lspci 究竟查看哪个设备的 AER 为准？只需要查看 EP 本身，还是 RC 也能体现？

看 EP 的最直接。有些被 EP mask 的就不往上报了，还有些 DPC 能够覆盖。

一般也不需要特意用 lspci 来查看错误。开启了 DPC 或者 BIOS Enable AER 之后，当收到 AER 相关错误的 Message 时会触发中断并调用 AER 处理函数，Linux Kernel 会即使打印 Error 信息，告知 Linux AER Driver 处理了何种 AER 错误。在打印处理之后，对于 Non-Fatal Error，在错误处理之后 EP 的 AER Status 会被清掉，采用 lspci 不一定能看到错误现场；对于 Fatal Error，系统会挂掉，可以用 dmesg 来查看此前打印的错误信息。这里的驱动为 Linux 内核自带的 PCI 驱动，AER 中断处理函数也是 OS 内写好的，不需要 Device Driver 提供。

571. 如果报了 Mutiple AER，AER 驱动会去遍历这个 RP 下的所有节点看是否发生了错误吗？

可以肯定的是驱动中会调用 find_source_device 这个函数来查找出错节点。个人理解，PCIe AER 上报错误时携带有 BDF 号，实际上不需要遍历 RP 下的所有节点。

如果多个 AER Msg 同时到，PCIe 驱动会先取走 BDF（Requester ID）放在 AER FIFO 里等待后续处理。硬件实现 FIFO 还好，如果靠软件缓存 AER Msg，在既定时间窗口内再来多个 Msg 可能会丢失 Msg，需要进一步确认该机制。（后续：从软件看，来了一个 AER 中断就从 AER Source 寄存器读取 Requester ID 存早 AER FIFO 等后续处理，软件没有并发处理的逻辑，感觉应该是 rp 硬件里面会有些处理。）

572. Linux 默认未开启 ECRC 功能，这是完全由 OS 控制的还是 Device Driver 也可以改？

补充描述：aer_probe 好像在 Device Driver 启动前就执行完了。

ECRC Enable 可以通过内核参数进行控制，也能 BIOS 控制。默认为保持 BIOS 中的设置，也可以改 grub 来改内核参数，不需要重新编译内核。

573. 如何根据 Requester ID 和 Message Code 定位错误报告位置？

跟预期 Message Code 比对。

复位

574. PCIe 有哪几种复位机制？

PCIe 有四种复位方式：冷复位（Cold Reset）、暖复位（Warm Reset）、热复位（Hot Reset）及功能层复位（Function Level Reset）。简介如下：

- ✓ **冷复位** - 关主电源，可通过边带信号或 Power State 两种方法实现冷复位。
- ✓ **暖复位** - 不关主电源，无特定规范，可选方案，可自行设计。
- ✓ **热复位** – 带内实现。Switch 的 Upstream Port 收到热复位时，复位其自己并广播给其 Downstream Port 来复位 Downstream Port；Switch 的 Downstream Port 收到热复位时，只复位其自己。

- ✓ **功能层复位** - 对于多 Function 的 Device，可以只复位特定的 Function。FLR 只复位内部状态及寄存器。

冷复位和暖复位基于边带信号由系统生成，称为基本复位，由硬件触发；热复位由上游设备生成，以包（TS 有序集）的形式传达到下游，由软件触发。

PCIe 设备复位释放后，重新进行链路训练及初始化，需等待至少 100 ms 才能向其发出访问请求。若初始化之前便收到了配置请求，需返回 Cpl.RRS 要求重传。

575. Host 触发了 FLR，等 Firmware 写完对应 Function 的 flr_ack 为 1 后，Host 要做什么操作才能激活被 Reset 的 Function？

补充说明：复位之后，Function 的寄存器能正常访问到，但 Host 给 Function 发 MWr/MRd 没有响应。

Function 复位之后，其内部一些寄存器被复位了，包括 BAR。根据 PCIe Base Spec，在 Host 软件发起 FLR 之后，需要等待 100 ms，然后 Host 软件重新对 Function 进行配置。

576. Host 对 Switch（Pericom）的 Upstream 做 Hot Reset，对于 Switch 的 DSP 的 Hot Reset 有什么时间要求吗？反复对 Switch 的 USP 做 Hot Reset，会发现 Host 偶尔会丢失 Switch 和下面的 Hierarchy，如果 Switch 下面不接设备就不会丢失，可能的原因是什么？

Hot Reset 后包括配置信息在内的大部分寄存器数据没有了，但可以重新枚举。感觉像是在规定时间内没 Reset 完。

577. SNPS 的 ECAM 如果写 Bridge Control 的 Second Bus Reset，复位后能解复位吗？

补充说明：Cadence 的就不行，会复位 AXI 总线，导致无法解复位。

没有做好 Flush 就会这样。（答非所问哦。）

578. PCIe 作为 Device 时，Controller 的 perst_n 一般怎么接？

对于 Device，Controller 的 perst_n 就是 Host 插槽下来的金手指上的一个 Input PIN，但通常也可以直接接 EP 芯片的 Power On Reset（por）。

理论上 perst_n 释放后金手指上的 refclk 应该是稳定的，但有些主机未遵循该规则。在依赖金手指 refclk 作为系统时钟的设备中，为了避免上述情况及提高复位的灵活性，可以把 perst_n 接在 GPIO 上，由 EP CPU 内的 Firmware 去读 GPIO 状态判断 perst_n 是否被释放，然后判断 refclk 是否稳定，待 refclk 稳定后由 FW 去释放 PHY 和 MAC 的 Reset。

579. 该如何理解 Cold Reset 和 Warm Reset？

Cold Reset 是指没电，Warm Reset 是指有电但 PERST 未释放。比如按下电脑开机键，属于 Cold Reset；开机后按重启键，属于 Warm Reset。不论是 Warm Reset 还是 Cold Reset，都需要重新进行链路训练、重新做枚举。

580. Hot Reset 和 Warm Reset 后再传输需要重新 Link 吗。

需要。上述复位后 LTSSM 都回到 Detect 状态了，需要重新 Link Up。再次 Link Up 后，Warm Reset 需要重新配寄存器，Hot Reset 不用重新配寄存器。

581. Link Down 再次回到 L0 后，Config Space 会被复位吗？

跟 Hot Reset 相同，Non-Sticky 寄存器会被复位，Sticky 的保持不变。

582. EP 意外 Link Down 再次 Link Up 后，Host 还能访问吗？

个人理解跟系统有关，不同 Server 反应不一，有时候意外 Link down 后主机会把这个口给禁用了。可能跟主机 DPC 有关，需要确认。

583. Hot Reset 或 Warm Reset 时 Host 会认为这个设备掉了吗？

Hot Reset 不会掉，Warm Reset 就是 Reboot 了会掉设备。

584. 如何启动 FLR 相关复位？

Host 下发 CfgWr 写 EP 相关 Function 配置空间头 Device Control Register 中相关字段。

585. 难道只能对 PF 做 FLR 吗，不能单独控制某一个 VF？

VF 自己也有 FLR。PF FLR 之后，对应 PF 的 VF Enable 就归 0 了。

586. 对某个 PF 做 FLR 必须不能影响其他 PF 吗？还是仅仅建议不能影响其他 PF？

不是建议，是必须。每个 PF 是独立的，驱动也是完全独立的，某个 PF 的 FLR 必须不能影响其他 PF。

587. 如果 EP 响应 Host 的 perst，开始复位，但是先于 Host 释放了，会不会有问题？

EP 先释放应该没问题。

588. Cadence 公司的 Gen3 IP 是如何处理 Hot Reset 的？

看手册。

589. Hot Reset 之后，一定要 Rescan 么？Linux PCI 驱动中为何必须要 Remove->Hot Reset->Rescan，单独做 Hot Reset 不行么？

问题补充：不做热插拔，只是想装自定义的 Driver。系统起来的时候，内核已经扫描到了所有 Device，为了装驱动，再做一次 Remove 和 Rescan 有必要么？有什么场景非要这样做。

如果单纯是不想用系统自带的驱动，可以重新编译内核，没必要做 Remove+Rescan。一半只有热插拔时才需要先 Remove 再 Rescan。

590. PCIe 设备收到主机发来的 Hot Reset TS1 后，需要等未完成的数据传输完后再 Reset 吗？Flushing Mode 是指传完还是直接清掉？

需要等传完再 Reset，Flushing 是指传完而非直接清掉。

591. Hot Reset 后需要 Rescan 吗？

Host 侧是需要的。

热插拔

592. PCIe 热切换和热插拔是什么？

PCIe 热切换和热插拔是一种在不关闭系统电源情况下更换 PCIe 卡设备的技术，能够确保换卡的同时 PCIe 系统持续不间断的运行。热插拔适用于 PC 及服务器主板上的 PCIe 板卡链接，热切换适用于 Compact PCI。

PCIe 卡有两个为热插拔而设计的引脚——PRSNT#1 和 PRSNT#2，这两个引脚位于金手指两侧，比其他引脚要短。当 PCIe 板卡被拔出时，这两个引脚能够先于其他引脚检测到板卡正在被拔出。在这两个引脚检测到 PCIe 卡拔出、其他引脚断开连接之前这段时间内，系统软件协同主板热插拔驱动器、PCIe 卡设备驱动以及 PCIe 卡硬件进行热插拔处理，有序切断电源。

593. 热插拔一般用在哪？

服务器中用得多，电脑、嵌入式系统平台及一些热插拔模块也会用。在通信厂的一些嵌入式系统中，有些 PCIe 设备初始化比内核驱动初始化晚，需要用到热插功能。

594. 热插拔需要物理按键吗？

标准的热插拔需要按键。在 PCIe Base Spec 中列出的热插拔相关元素中包括 Attentiob Button，即按键，但说明了可以按需实现。

595. PCIe 设备该如何支持暴力 Hot-Plug？

标准板卡要软硬件支持：RC 或 PCIe Switch 芯片上要有热拔插控制器 HPC，Host 也要配置 Hot-Plug 相关的寄存器。Hot-Plug 可参考 Linux 下 Hot-Plug 相关代码。

普通 PC 没有 HPC，服务器有没有 HPC 可查看手册。

596. Hot-Plug 需要 Host 配合吗？跟 EP 有关吗？

热插拔是个系统功能，PCIe IP、硬件、OS 都要支持，缺一都难以稳定工作。

关于 EP，一说 EP 也需要支持热插拔，又说和 EP 无关，因为标准热拔插电源都会关闭掉。

597. EP 端软件需要支持 Hot-Plug 功能吗？

取决于 EP 产品。考虑到热插拔成本较高，一般不建议支持热插拔的方案。

大部分 EP 设备软件是不需要支持 Hot-Plug 功能的，比如大多无线 WIFI 芯片、低端网卡不支持热插拔，但有些高端网卡、矿机、服务器 SSD 等设备需要软件支持热插拔，OCP 显卡网卡也多支持热插拔。

598. 目前实现 Switch 整体热插拔了吗？

背景：感觉实现上没有难度，实测关掉开发板电源后，主机未蓝屏，但重新加电后主机无法识别 Switch。既然主机没崩溃，证明里面已经打扫干净了，重新加电按理说没问题吧。

没实现。上述现象跟 Switch 没关系，跟 BIOS、OS 甚至 CPU 等有关系。

599. 目前 Linux 内核是否已支持整个 PCIe Switch 及其下挂设备的热插拔操作？

注：此处指 Surprise 热插拔，非标准热插拔。

对于纯粹热插 Switch 上去，可能会出现无法识别 Bridge 的情况，原因为预留的 Bus 号数目不够，默认预留一个，而 Switch 需要不止一个 Bus 号。如果是先热拔再插，不会出现预留 Bus 号不够的情况，内核很早就支持这种方式了，rescan 重新扫描即可。

Surprise 热插拔也有热插拔信号。正常卡应该不支持 Surprise 热插拔，只有 SSD 等部分设备支持 Surprise 热插拔。此外，GPU 设备对 Surprise 也有热插拔的需求，这些设备本身支持热插拔，但带上 Switch 后就不支持 Surprise 了。

Surprise 热插拔是个比较复杂的系统工作，如果设计不好，存在烧主板的风险。金手指形态的热插拔大概率会烧板，M.2、U.2 设计不会烧，其电源设计不同。

相关连接：飞腾 S2500 平台 PCIe SWITCH 下热插拔验证

600. Switch 支持热插拔吗？

可以支持。Switch 中的 DSP 和 USP 都可以支持，一般 Switch 上只有 DSP 支持且可以部分 DSP 支持热插拔。博通有支持热插拔功能的 PCIe Switch。

默认情况下每个 Switch 端口下预留一个 Bus 号，但也存在预留多个 Bus 的情况。比如某支持热插拔的 8 端口 Switch，其每个端口下都预留 9 个（1 个给当前端口，8 个给下游 Switch）Bus 号及对应的 Memory 空间，对 Bus 和 Memory Space 资源浪费较大。这种情况下，可以想办法替换 Port 的驱动，每个端口只留 1 个 Bus，或者改用 Switch Fabric 功能。

601. Switch 热插拔未实现技术上卡在哪里了？

问题描述：既然实现了 EP 的热插拔，Switch 整体实现热插拔根本就不是问题，Switch 热插拔未实现技术上卡在哪里了？

Switch 热插拔意义不大，很少人愿意去做。

EP 热插拔做了很多年了，整体实现效果也不咋地。

602. 硬件上是如何检测到热插拔的？

PCIe 金手指上有一根金手指比别的短，能够在正式 Link Down 之前提前感知到热插拔并进行电源保护、业务相关处理工作。

603. Linux 中是如何识别和处理热插拔的？

Linux 通过中断来处理热插拔。检测到热插拔中断后，重新识别 Slot。

604. 如何知道是哪个 Slot 发生了热插拔事件？热插设备是如何添加到原 PCI 设备树中的？

热插拔的中断由 Port 的驱动来处理，驱动程序能知道是哪个 Port 报上来的中断，从而确定是哪个 Slot。Hot-Add 是直接在原来树上简单加叶子节点，不改变此前 PCI 设备树中的其他部分。

605. PCIe 5.0 热插拔的 Async Removal 和暴力热插拔有什么区别吗？

我理解是一个东西，区别 PCIe 3.0 之前的 Surprise Hot-Plug。

我理解两者只是 OS 那边处理的方式不一样，硬件这边都当作暴力热插拔处理。

606. 软件上 Remove PCIe 设备和 Hot-Plug 是一回事还是两个概念？

Remove 和 Hot-Plug 是两个概念。Remove 设备不涉及拔卡、卡不掉电，设备相对安全；Hot-Plug 会拔卡掉电，电源做不好容易把主板烧坏。标准热插拔接口边有个按钮+LED 指示，需要在拔卡前按下按钮。

607. 热插拔中断上报是硬件自动报的，还是软件报的？

问题补充：发生热插拔后热插拔寄存器状态位发生变动，监控到状态位变动后上报热插拔中断。

都可以吧，硬件也行，软件也行。

做过 Windows 下的热插拔，内核软件轮询热插拔寄存器状态。

608. Switch 端口预留空间不够的时候，RC 还有空间，不可以再分配吗？

一般不会再分配。RC 再分配会影响上游端口已分配的空间，影响其他端口使用。而且热插拔一般不会单独热插，往往是先拔再插，这种替换的话就没分配问题。

609. 热添加一个 PCI 子系统，PCI 空间不够分配怎么办？

Switch 端口会预留空间，每个端口默认预留 2 M，预留空间不够就会识别失败。

610. RC 端通知 Hot-Plug 功能跟 PCIe Controller 有关吗？

有关。具体通知机制取决于硬件方案，热插拔上报可以用 PCIe Controller 内的 DPC，也可以用 PCIe Controller 内的 Slot Capability 相关寄存器配合热插拔控制器 HPC。

611. PCIe Hot-Plug Events 中的 Command Completed Event 该怎么理解？

PCIe 热插支持多种事件触发方式，Command Completed Event 是其中一种可选的方式。Command Completed Event 是一种可选的热插触发方式，协议并未介绍具体实现细节，每家厂商具体实现细节不一。

个人理解，非事件触发方式中，软件需要通过轮询热插状态寄存器来获知 Hot-Plug 是否完成，而 Command Commend Completed Event 则提供了中断告知方式，一旦命令完成则触发 Command Completed Interrupt，告知 Host 此前命令执行完毕，可以执行下来的命令了。

612. 热插拔时主机如何获知设备用的 Device Number 是多少？同一个 Bus 下的设备 Device Number 相同了怎么办？

不存在该情况，PCIe 是点到点连接，同一个 Bus 下只能有一个 Device，Device Number 为 0。即便是多个 Slot，这些 Slot 之间的 Bus Number 是不同的。即便是同一 Switch 下的多个 DSP，Bus Number 也都是独立的。

613. 目前操作系统内核支持对整个 Switch 进行热插拔吗？

不支持。软件上资源分配之类的不好做。

扩展能力

基本介绍

614. 如果 RC 为 PCIe 2.0 但 EP 为 PCIe 2.1，PCIe EP 2.1 相关的 Capability 能否被 RC 识别到？

可以获取到，但不一定被识别。Capability 获取就是一个读链表的过程，读能读到，读到后不一定能被识别。

615. 如何查找扩展配置空间中的各种 Capability？

就是一条链表。0xff 之前一个链表，从 Offset=0x34 开始找；0x100 之后一个链表，从 0x100 开始找。

DRS

616. Base Spec 中讲 DRS 时提到，DRS Supported bit clear 时可以发送 DRS Messages，又说 DRS-capable 的 Port 才能发 DRS，两者不是冲突了吗？

不冲突。DRS Supported bit clear 是从 Capability 寄存器角度讲的，DRS-capable 是从 DRS 逻辑是否具体实现角度讲的。如果电路中实现了 DRS 的逻辑，即便其能力寄存器

Supported bit 未置一也能发。协议中也提到，如果实现了 DRS，强烈建议把 DRS Supported bit 置起来，而非必须。

RN

617. 什么场景下会用 PCIe RN 这个 Feature？

RN（Readiness Notification）是为了快启动，非必须。根据协议，在 Detected（5 GT/s 及以下）或 Link Up（5 GT/s 以上）后需要等待 100 ms 才能发 Cfg 请求，而采用 RN 时 EP 可以提前告知自己是 Cfg Ready 状态，以缩减该时间。相比于 CRS 机制，协议更推荐采用 RN 进行快启动。

MTCP

618. Device 能往 DDR 发 MTCP 的包吗？MTCP 包存在 DDR，用 VMD 来承载 MCTP。

取决于芯片里具体功能的具体实现方式。

IDE

619. PCIe 在什么场景下需要用到 IDE？

补充描述：Link IDE 功能有人用么？感觉根本没人用，也几乎没场景，不知道是不是。

在对数据安全要求高的场景中需要用到 IDE，比如机密计算。

Atomic

620. PCIe 原子操作的原子性体现在哪？

问题补充：是体现在一次操作完成多件事情？还是锁定目标地址不许其他写入？还是锁定总线？

PCIe 原子操作不会锁定总线。PCIe 原子操作可以理解为存储访问指令上叠加一个运算操作，一般由总线和缓存系统执行，不锁定总线，同地址的请求会有 Hazrd 处理排队执行。比如 Read Modify Write，在一个指令里完成这三个操作。

621. PCIe 原子操作地址为 32/64/128 bit 时，要求 4/8/16 Byte 对齐吗？

要求对齐。协议原文如下：

For AtomicOp Requests, the Address must be naturally aligned with the operand size.

622. 原子操作 FetchAdd 的 Length=2 同时 BE=0，这符合协议吗？

符合协议，Atomic BE 为 Rsvd。

623. PCIe 原子操作中能开 ID 排序吗？能用 Steering Tag 吗？AT 还能用于地址翻译吗？

可以开 IDO。ST 可以开，但目前 ST 多用于主机支持 Cache Stash 的场景，Atomic 没有这种概念。AT 可以作为地址翻译请求。

624. 设备发出 AtomicAdd 操作 Host Memory，RC 收到该事务后是不是向 Memory 发起读操作、然后 Add 后写入，RC 端再返回 Atomic 完成状态？

不是。AtomicOp 到对端后，可以理解转换为 AXI 上的 Atomic 操作出去到 Host Memory。

OBFF

ARI

625. 枚举过程中如何发现下游端口连接的 EP 是否支持 ARI 功能？是不是先按照不支持 ARI 进行枚举，读完配置空间发现支持 ARI 后，再对设备进行配置？

应该是先读 Capability, 来确定是否支持 ARI。若支持 ARI，则预留 Bus num，然后继续枚举。（参考 PCIe ARI (Alternative Routing-ID Interpretation)介绍-CSDN 博客）

626. 在 Linux PCI 枚举时，如果支持 ARI 会去 ARI Capability 寄存器查找 Next Function Number，那么每次去查都是相同的 Next Function Number 吗？

不是的。到扫描某 Function 的 ARI Capability 获取到 Next Function Number 之后，下次扫描就是 Next Function 中的 ARI Capability 了。

627. ARI 功能与 SR-IOV 有关吗？

ARI 目的是扩展 Virtual Function，为 SRIOV 服务的。

LTR

628. 软件角度应该如何使用 LTR Message？是不是 EP 业务可以随意发送任意数值的 LTR Message 给 RC？

理论上 EP 使能 LTR 可以在任意时间发送任意数量的 LTR 给 RC，但 PCIe Base Spec 建议 EP 开了 LTR 之后立即给 RC 发 LTR，如协议未特别说明，强烈建议 500 us 内不要发送超过 2 个 LTR。

关于 LTR 数值，理论上 EP 可以发任意值的 LTR Message 给 RC，但 RC 可能无法满足这个时间。比如 EP 发 LTR 说 10 ms 进 L 几，但是 RC 这边需要 20 ms，则 EP 发的 LTR 无法满足。

629. EP 设备访问 PC 端 DDR 同一个地址，首次访问 Delay 时间特别长，再次访问 Delay 就正常了，为什么首次访问 Delay 时间长？

猜测受 PC 端对 PCIe 功耗管理的影响，可尝试关闭 ASPM，或采用 LTR 来避免。

LTR 会影响 CPU 的状态切换，可尝试在读内存之前读配置空间，保证 CPU 处于正常状态、Link 处于 L0。

630. 主机收到设备发来的 LTR Message 后会回一个 LTR Message 吗？

首先 LTR Message 是 Posted 请求，不需要回响应；其次，LTR 不是直接发给主机的，而是发给主机电源管理模块的。LTR 只是告知，主机收到该 Message 后，会根据包括 LTR 值在内的当前系统综合指标决策决定是否进入休眠或者更深状态。

631. 如果主机不接受设备发来的 LTR Message 请求值怎么办？

发送给 Switch 的 Downstream Port 的 LTR 必须反映 Upstream Port 下所有 Device 的功能所能忍受的共同的 LTR 值。最终发上去的也只能有一个 LTR 值，这个值是一个共同的 LTR 值。经过 Switch 汇总，产生一个新的合适的 LTR 值。

TPH

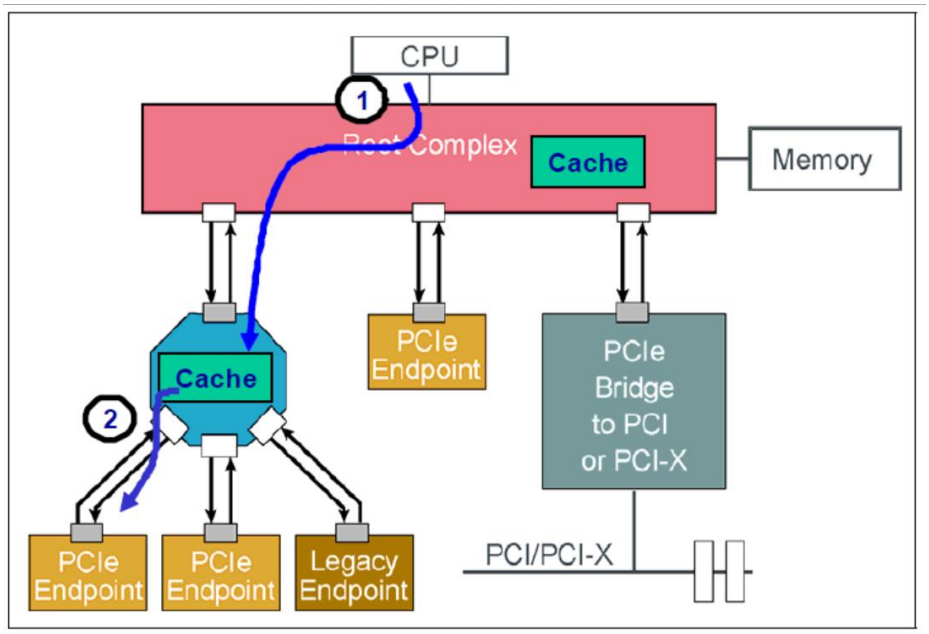
632. TPH 有实际使用场景吗？

TPH 的实际应用场景不常见，Intel 配合它的网卡可以用。

633. Switch 中的 Cache 缓存的是 Device Mem 的内容吗？假如是的话，直接写到 Device Mem 中 Device 在每次读的时候不是更快吗？

补充：下面是 Mindshare TPH 章节的描述，有些疑惑。毫无疑问，对于写，有 Cache 肯定比没 Cache 快，没 Cache 还得等实际发出去才能完成。根据上图文字描述，写的频率很低，主要是读。

Host Write to Device Read. To illustrate the concept going the other way (called Host Write to Device Read or HWDR), consider the example shown in Figure 20-13 on page 903. In this example, the CPU initiates a memory write whose address targets the PCIe Endpoint in step one. The packet contains TPH bits that tell the RC that it should be stored in an intermediate cache near the target, instead of the cache in the RC that was used in the previous example. In this case a cache built into the Switch serves the purpose. The TLP is then forwarded on to the target Endpoint in step two. This model is beneficial when the data is updated infrequently but read often by the Endpoint. That allows several memory reads that would normally go to system memory to be handled by the cache instead, off loading both the Link from the Switch to the RC and the path to memory.



PCIe Switch 内的 SRAM 很小，一般只在出错时起个缓冲作用，实际使用时大部分都是 Cut Through 传递。具体可参考 Cut Through 章节。

634. 请问 TPH Capability ST Table 中的 Entry 中要填什么？

一般都是 Vendor Specific, 或者不用，我遇到的基本都是不用的。

填写需要 Stash 的 Node 或者 SLC 信息，配合 Cache Stash 用。

635. EP 如何将 TPH 里的 Steering Tag 对应到 RC 相应的 Cache 呢？怎么用？

TPH Steering Tag 映射不到具体的 Cacheline，最多可以指定 Core 映射和 SLC，其跟 Cache 之间的具体对应关系需要 EP 跟 RC 商量。

本质上 TPH 用法跟 PASID 一样，就是软件给任务打个标签，主机通过描述符发下来，DMA 执行时即可，硬件上没有什么。在这个过程中，EP 不需要获取 RC 端的 Cache 拓扑，由 RC 端的 EP 驱动（网卡驱动）负责填充 EP 端的 ST Table，直接填 CPU Cache ID；在发送 TPH 请求时，EP 硬件根据 EP 驱动之前的配置决定哪些业务是否使用 TPH 以及使用 ST Table 中的哪个 Entry。可以利用 Steering Tag 去做 Stash，由 Host 来定义映射 Stash IO Deallocte 等操作类型，还有需要 Stash 的核心。

对 RC 设计而言，跟 Intel 兼容即可，因为 EP 基本是按照 Intel 标准设计的。

DMWR

636. DMWr 是什么？

DMWr，全称 Deferrable Memory Write，可延迟的内存写入，是一种新的 PCIe TLP 类型。CXL 1.1 中就已经有 Deferrable Writes 了，正式出现在 PCIe 协议中是 PCIe 6.0。

可延迟的内存写入，顾名思义，Completer 在收到 DMWr 后，可以选择立即执行，也可以选择延迟执行。相比于转发请求 MWr，DMWr 是非转发请求（NPR），无论请求是否被执行，Completer 都应根据完成情况回复指定状态的 Completion。

637. 为何需要 DMWr ？

DMWr 的设计初衷是实现一种有效的软硬件接口控制机制。在 DMWr 出现之前，为了实现软硬件交互，每个进程都需要一份单独的硬件逻辑进行处理，且每个进程都需要分配 4 KB 空间以隔离进程。随着进程数量增大，硬件空间并不会增大，很容易出现硬件空间不够的情况。DMWr 实现了一种共享工作队列，来解决以上问题。

在 DMWr 出现之前，PCIe 不能或者很难实现延迟写入。DMWr 实现了这样一种机制，允许 Host 或 EP 直接执行或延迟执行内存写请求。该机制可应用于 Host 及 EP，以简化流控部分的设计工作。基于 DMWr，PCIe 设备可以在单个共享工作队列内以一种非阻塞的方式接收多个非协作软件 Agent 的工作请求。换言之，PCIe 设备可以实现一个工作队列寄存器，允许多个 Requester 通过单笔 DMWr 请求向设备发出命令，设备通过 DMWr 的 Completion 状态来指示某 Requester 命令是否被接受。

638. DMWr 请求的排序规则是怎样的？

根据事务排序规则，有以下几点要注意：

- ✓ 转发请求可以超过 DMWr 请求，DMWr 请求不能超越转发请求。
- ✓ DMWr 和 MRd 都是 NPR，根据事务排序规则，NPR 之间也可以互相超越。

- ✓ 若是先发的 DMWr 后发的 Mem_Rd，且两笔 TLP 目的地址相同，那么 MRd 读回来的数据有可能是 DMWr 写之前的旧数据，也可能是 DMWr 写之后的新数据。

639. Deferable Memory Write (DMWr) 的 Defer 体现在哪？Rx 收到 DMWr 后无需延时、立即写并反馈 CplD，本质是 Non-Posted Write 吧？

DMWr 本质就是 NP Write。Defer 体现在 Completer 短时间内无暇处理该笔写请求时，回复 Cpl.RRS，告知 Requester 过段时间重传，而非一定要加点延时再回 Cpl。正常 MWR 是 Posted 的，DMWR 是 Non-Posted 的。更多 DMWr 可参考《PCIe 6.0 新特性 - DMWr (Deferrable Memory Write) 详解》。

640. DMWr 的 Completion 应该怎么回？如果是 SC，那是回 Cpl 还是 CplD？

DMWr 是写请求，不需要回 Data。不管写成功还是失败，Completer 均回不带 Data 的 Cpl。

641. 发送端上层软件如何知晓延迟写已经完成？是轮询 RC 寄存器吗？

是一条读写命令返回，一条汇编，中间不会去干别的，等执行完了再去拿结果。

642. 对于被丢失的 Posted Write，软件一般怎么处理？如果目标地址是关键的控制指针之类，最后软件岂不是动辄崩溃？如果 Posted 写有很大概率静默丢失的话。Local DDR 是不是也有大概率丢失写请求？

链路不好的会静默丢失，丢失了应该会报错，Local DDR 内部的比较难有写丢失了。此外，没那么容易丢，PCIe 不是有 Error 吗，去处理 Error，致命性和非致命性错误。

643. 为什么要设立这个 DMWr 的重传机制？Requester 目前来看就是 enqcmd 指令自身，该指令必须得到 cmpl 才算执行成功，在这之前线程处于阻塞状态，接收者按理说不需要告知请求者 DMWr 的中途执行状态。如果接收者丢掉 DMWr 让发送者重传，那么说，常规的 Posted MWr 请求，也有可能被丢失？PCIe 不是保证端到端无损的么？

这里的重传不是由于包丢了才重传，而是由于接收者自己的原因，比如资源占用无暇处理，告知发送者后续择机重传。相比之下，Posted write 发出后不许要接收者状态，直接就返回了，实际该请求可能并未写入内存。对 PCIe 链路而言，无论 Posted 还是 Non Posted Write，其能保证把包无损传到对端，但不保证写到最终目的，包出错请求包重传，重试次数达到上限后会被列入 Error。

644. PCIe Base Spec 未规定 DMWr 的 Length，但有相关寄存器配置来指示支持的最大 DMWr Length 为 64 B 或 128 B，是根据 DMWr32/64 来间接算 Payload DW 长度吗？

DMWr TLP 中仍然有 Length 来指示当前请求携带的 Payload Size，这里的 Length 不超过上述寄存器中配置的 64 B/128 B 即可，类似于 Atomic，超过会报错。

645. 追问：这么说，发送端上层软件可以允许在非阻塞模式了。进一步推测，发送端的 RC 收到 DMWr 的 Cpl 或 Cpl.RRS 后都会发送中断，相对于 PCIe 5.0 需要 RC 在硬件上做中断相关改动？

x86 没用中断，Cpl 状态是一个返回值，上层软件直接根据返回值判断是否 Retry。

646. 追问：发送端上层软件如何知晓 DMWr 已经完成？是轮询 RC 寄存器吗？

不是轮询 RC 寄存器。就是一条读写命令返回，一条汇编指令，中间没有其他操作。

647. 追问：对于 NP Read，软件发出 MRd 后 CPU 拿到数据自动执行下一指令，软件继续运行；对于 Posted Write，写到 CPU 队列后即可继续执行下一指令，软件知道写操作已完成；同理，对于 DMWr，软件必须知道 DMWr 已完成，这个完成信号是如何传递给上层软件的？是否需要一条特殊指令？内核还需要一个 API？

内核应该还没有该 API，x86 是实现了一条特殊访存指令 enqcmd，底层通过该指令发出 DMWr。

648. 追问：为何要设立重传机制？目前来看 Requestor 是 enqcmd 指令自身，该指令必须得到 Cpl 才算执行成功，在此之前线程处于阻塞状态，按理说 Completer 无需告知 Requestor DMWr 的中途执行状态。如果 Completer 会丢掉 DMWr 让 Requestor 重发该请求，是否意味着常规的 Posted Write 也会被丢失？PCIe 不是保证端到端无损的么？

PCIe 不保证端到端无损，重试 N 次后报错。此外，Completer 收到 DMWr 后不是告知中途执行状态，而是由于 Completer 自身原因，比如资源占用，请求 Requestor 重传。对于 Posted Write，Requestor 无需等 Cpl 就返回。

649. 追问：对于被丢失的 Posted Wirte，软件该怎么处理？如果目标地址是关键的控制指针之类的，任凭 Posted Write 静默丢失，软件岂不是会崩溃？

链路不好的时候是会静默丢失 Posted Write，但丢失了会报错，软件去处理 Error 就好了。对于 Local DDR 内部的很难有写丢失。

650. 追问：处理报错时，状态已经丢失了，Host 如何得知是哪个线程写的哪个地址？线程不会跑飞吗？

报错后上层会知道丢指令了，一般丢几个指令上层也能处理掉。报错上传时含有 BDF 信息等 tag，上传能知道丢失的是哪条指令。

651. 发送端上层软件写数据可以运行在非阻塞模式，是否可进行以下推测：发送端的 RC 收到延迟写的 Cpl.SC 或者 Cpl.RRS 后都会发送中断，即需要发送端 RC 硬件上做改动？

x86 中没用中断，DMWR 的 Cpl.RRS 就是一个返回值，上层软件直接判断返回值决定是否重传，无需硬件改动。

652. DMWr 写完成的信号是如何传递给上层软件的？

问题详述：对于 NP Read 请求，软件发出 Mem Read，CPU 拿到了数据就会自动执行下一条指令，软件继续运行，软件便知道读已经完成。对于 Posted 写，写到 CPU 队列中即可继续运行下一条指令，软件知道写已经完成。但是对于延迟写，软件必须明确的收到延迟写完成信号，这个信号怎么传递给上层软件的？更进一步，延迟写肯定是有一条特殊访存指令，对应内核里某个 API。底层会发出延迟写，不知道是不是这样。

x86 是实现了一个新指令 enqcmd，内核应该还没有 API，现在主要是应用判断是否需要延迟写，不需要就是正常的 Mem Write，需要就发 enqcmd。

653. 处理报错时，状态已经丢失了，Host 并不知道是哪个线程当初写入的对应地址，如何通知线程？此时估计线程已经跑飞了吧。

对于中断设备而言，如果是终端设备认为需要丢掉 Request，被丢的 Request TLP 的 Header 能存在配置空间寄存器内，这部分存放错误 Header 的空间可大可小，看具体实现，同时还会上报 AER 给 Host，终端自身也会上报异步事件给自己的 Firmware 记录。

对于 Host 而言，上层应用有自己的超时逻辑，也有办法知道写数据丢失了，取决于代码实现方式。一般而言，针对少量的关键数据，需要重新读进行确认，发出的命令有 tag，丢几个命令一般能处理掉；对于大块数据区，静默损毁无解，丢了就丢了。相比硬盘，PCIe 链路层面丢数据的概率极低。

Multicast

654. 如果 EP 和 Bridge 上都有 Multicast Capability Register，那么在配置时是配 Bridge 的还是 EP 的？如果只需配 Bridge 的，那 EP 为什么要实现这些堆寄存器？

两处都需要配置。为了兼容有些不支持 MC 的 EP，Bridge 有个 MC_Overlay 机制可以提供支持。

655. PCIe Multicast 功能一般在什么场景应用？目前在服务器上应用需求大吗？

服务器好像没有这种需求。GPU 上有用，在集中式存储中也会用，用以同时往多控内存写数据，比如数据同时送往盘阵和镜像服务器。现在的 PCIe Switch 也要求支持组播功能，博通的和微芯的 Switch 都是支持的。

VPD

656. 什么设备会用到 VPD 这个功能？需要 RP 配合吗？

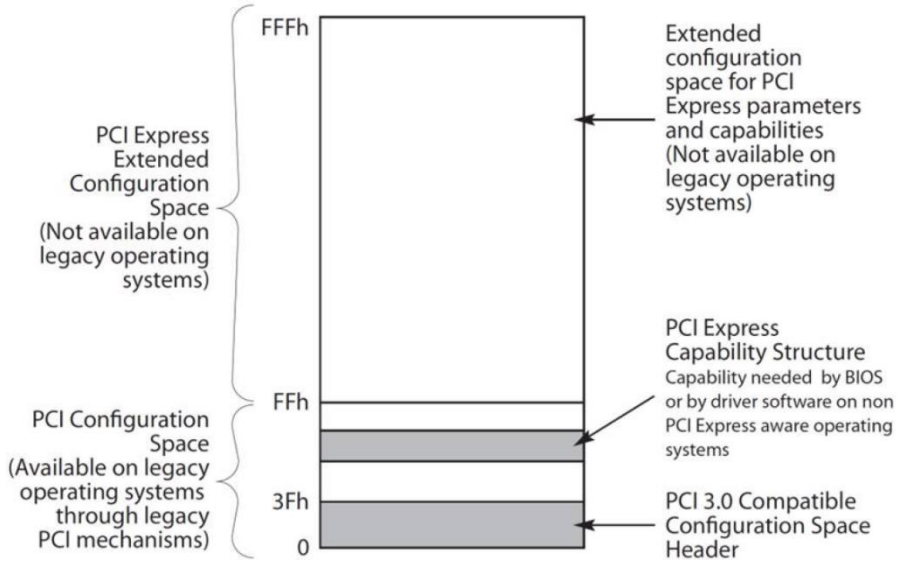
MLX 网卡中有用到。VPD 一般是用于 RP 查看 EP 的一些信息，EP 实现该功能即可。

七、软件初始化与配置

配置空间

657. PCIe 配置空间是什么？

PCIe 配置空间是 PCIe 物理地址空间的一部分，是一组地址连续的寄存器，其 Layout 如下图所示。PCIe 配置空间中记录了该 PCIe 设备的相关配置信息，通过配置该配置空间中相关寄存器可以对该 PCIe 设备进行系统配置。系统上电后，BIOS 检测 PCIe 总线并扫描配置空间，来获取 PCIe 设备的配置要求并进行系统配置。配置空间的存在，为 PCIe 即插即用提供了可能。



▲图： PCI Express Configuration Space Layout

658. PCIe 配置空间在哪里？

PCIe 配置空间位于该 PCIe 设备内存空间内，通常在上电后由硬件从 E2PROM 读取默认值填写到配置空间。

659. RC 有配置空间吗？

RC 也有配置空间，RC 配置头与桥设备的配置头格式相同。

660. 每个 PCIe 设备都有一个配置空间吗？

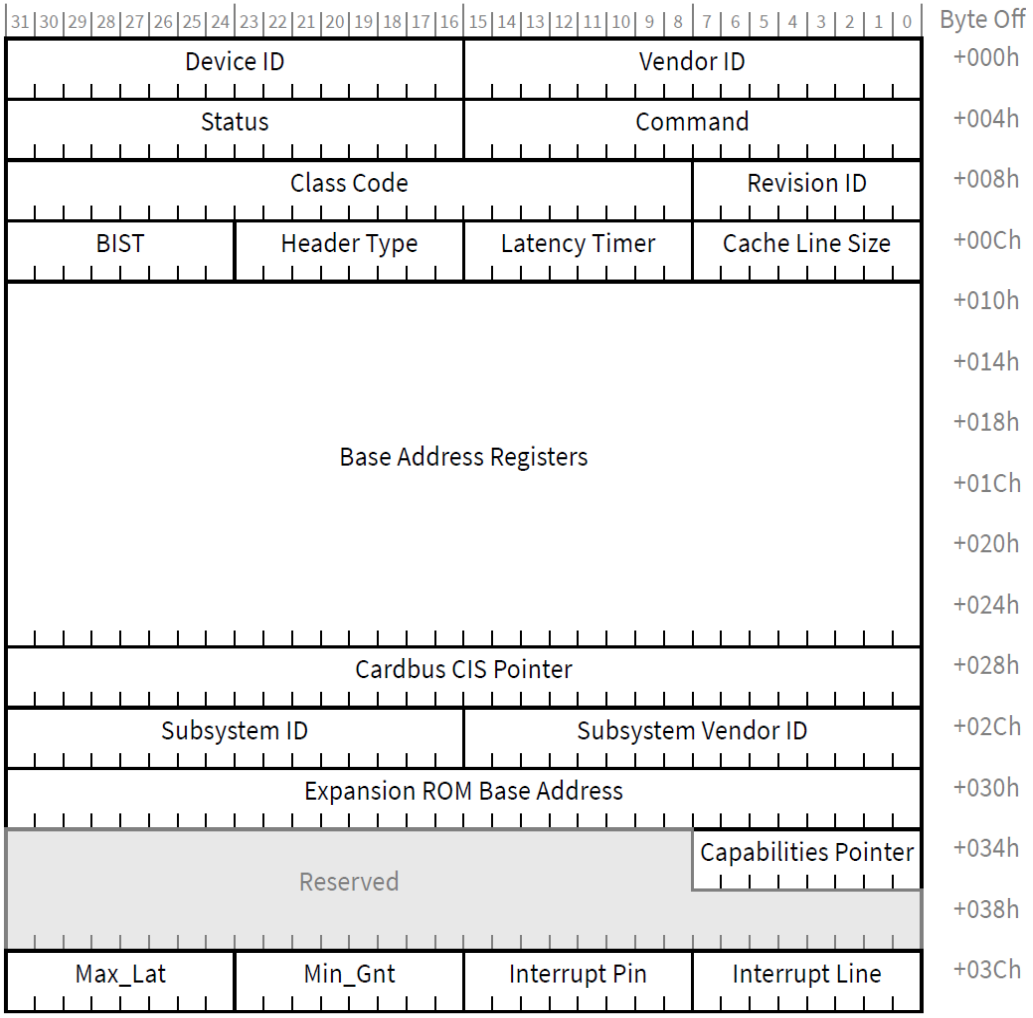
每个 Physical Function 都有一个独立的配置空间。或者说，每个 BDF 号都对应一个独立的配置空间。

661. PCIe 配置空间 Size 是多少？

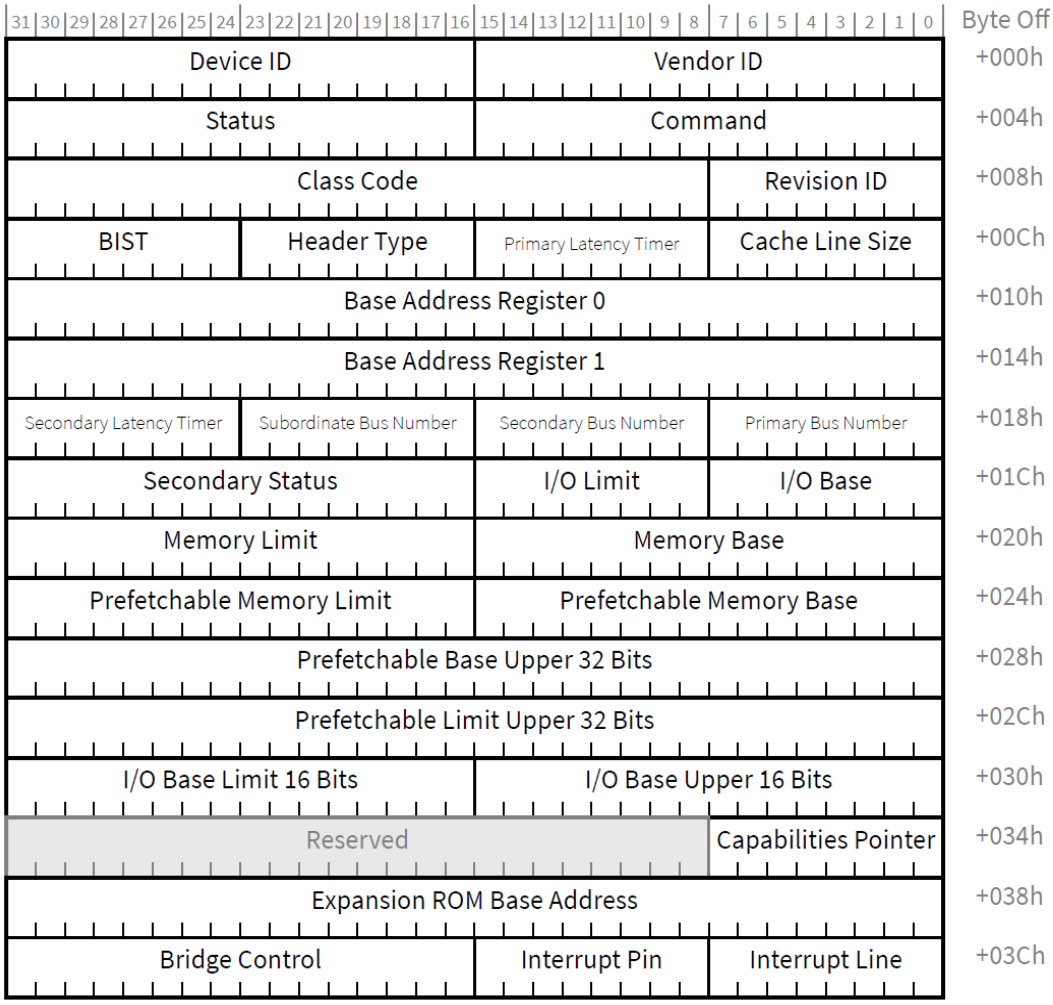
PCI 设备的配置空间为 256 B，PCIe 设备的配置空间为 4 KB。PCIe 4 KB 配置空间的前 256 B 空间即 PCI 的配置空间。对于最大支持 256 Bus * 32 Device * 8 Function 的 PCIe 设备，其内存需要给该 PCIe 设备预留 4 KB*256*32*8=256 MB 作为配置空间。

662. PCIe 配置空间分为哪几类？

根据 PCIe 配置头的不同，PCIe 配置空间可以分为 Type 0 和 Type 1 两类，PCIe 配置头 Header Type 寄存器中的 Header Layout 来指示当前为何种配置头。Type 0/1 配置头如下图所示。Type 0 配置头用于 EP 等非桥设备，Type 1 配置头用于 RC/Switch 等桥设备。桥设备在向下转发配置请求时会将 Type 1 类型的配置读写请求转换为 Type 0 类型，非桥设备收到 Type 1 类型配置访问请求后直接忽略。



▲图： Type 0 Configuration Space Header



▲图： Type 1 Configuration Space Header

663. PCIe 地址空间分为哪几块？

PCIe 地址空间分为 Memory、IO、Configuration 及 Message 这四块。

664. PCIe 有哪几种配置空间访问机制？

PCIe 支持两种配置空间访问机制：

- ✓ CAM，Configuration Access Mechanism，兼容 PCI 的配置访问机制，每个 Function 的配置空间大小为 256 B。
- ✓ ECAM，Enhanced Configuration Access Mechanism，PCIe 增强版的配置访问机制。ECAM 在 CAM 基础上扩展了单个 Function 的可访问配置空间的范围到 4 KB，优化了访问机制。

前 256 B 配置空间可以采用 CAM 或 ECAM 两者机制进行访问，256 B 之后的配置空间只能通过 ECAM 机制去访问。

665. PCIe 地址空间有哪几种访问模式？

PCIe 寄存器访问支持 Wire Access 和 DBI Access 两种模式。Wire Access 是对端 PCIe 访问本地寄存器（CDM 内部寄存器或 ELBI 外部应用寄存器）。DBI Access 是本地 CPU 等 Master 访问本地寄存器。

666. DBI、ELBI、LBC、CDM 是什么？

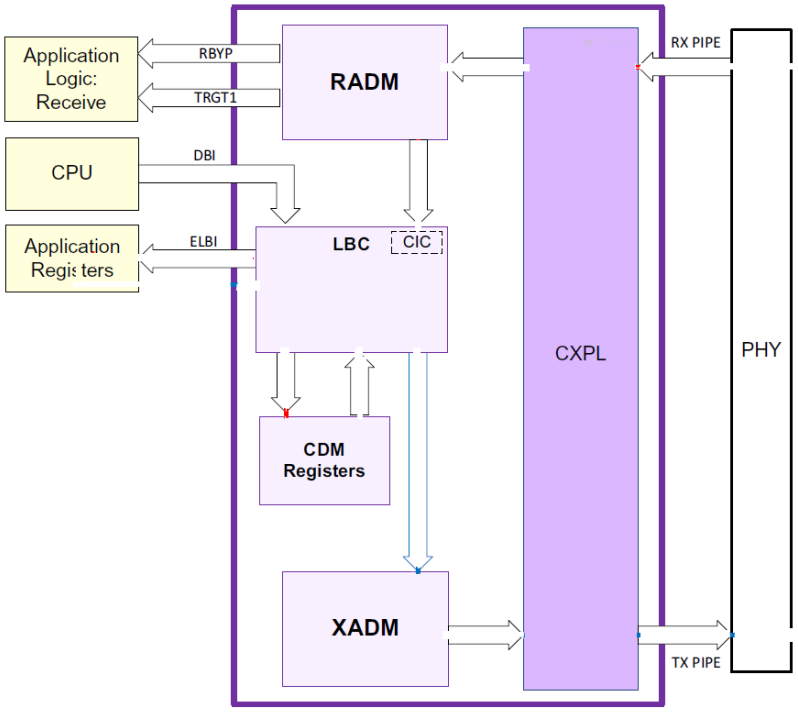
这是 Synopsys PCIe IP 的几个概念，介绍如下：

DBI，Data Bus Interface，EP 端的 CPU 可以通过 DBI 接口 PCIe Controller 的 CDM 内部寄存器，也可以访问 ELBI 外部应用寄存器。

ELBI，External Local Bus Interface，把 PCIe Controller 收到的 Inbound 寄存器读写请求转发到外部应用寄存器。

CDM，Configuration-Dependent Module，含标准的 PCIe 配置空间及 PCIe Controller 指定的寄存器空间（端口逻辑寄存器）。

LBC，Local Bus Controller，EP 的 LBC module 给对端 USP 或本地 CPU 提供一种通过 DBI 访问 CDM 内部寄存器及 ELBI 外部应用寄存器的机制。

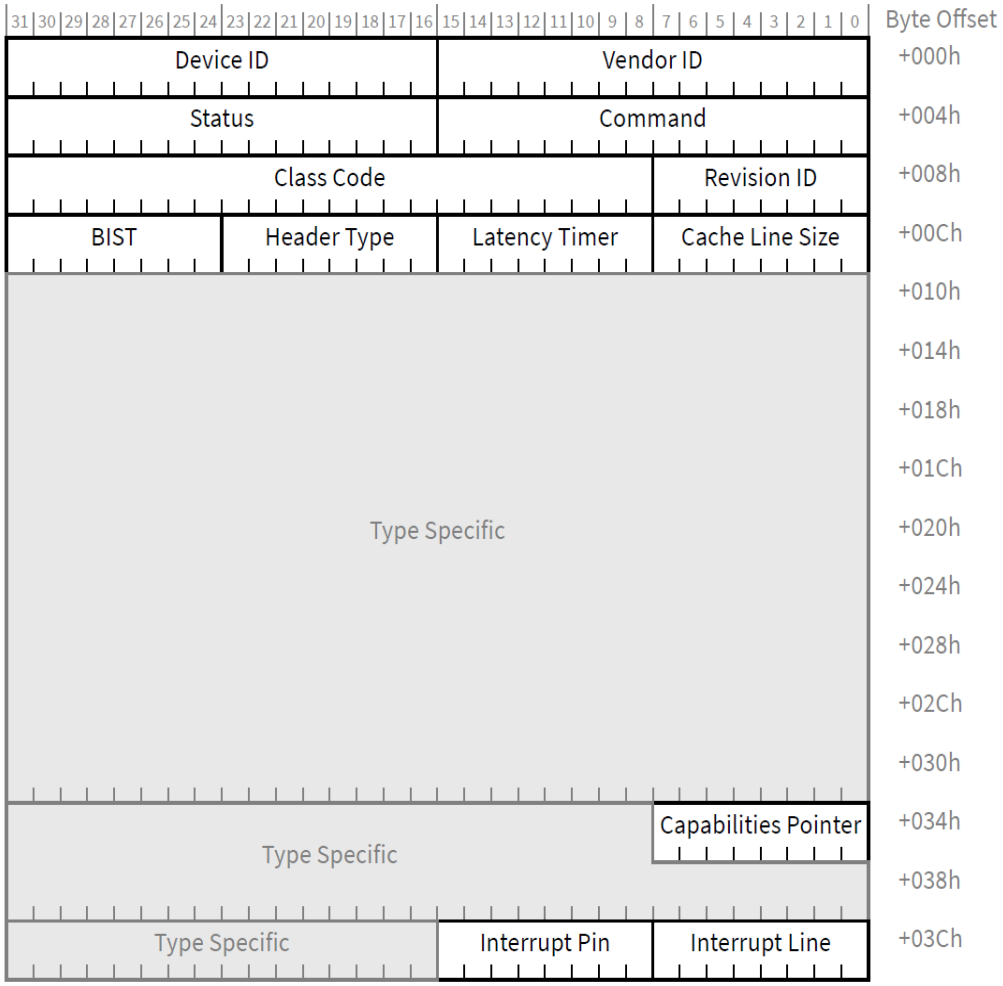


667. 扩展配置空间怎么找？

FFh 之前有一个链表，Offset=34h 开始找。100h 之后有一个链表，从 100h 开始找。扩展配置空间是从 100h 开始的。

668. PCIe Configuration Space Header 是什么？

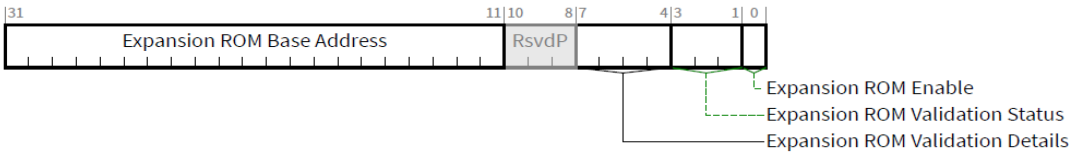
PCIe Configuration Space Header 是 256 B PCI 配置空间的前 64 B，称为配置头。PCIe 配置头的通用格式如下图所示。64 B 配置头主要用于设备识别、定义 IO/Memory/Interrupt 等访问方式等。剩余 192 B 称为本地配置空间，PCI 设备自定义，主要用以定义 Local Bus 的相关特性、Local Space 基地址及范围等。



▲图： Common Configuration Space Header

669. PCIe 配置空间中的 Expansion ROM Base Address 寄存器是干什么的？

有些 PCIe 设备（显卡、硬盘、键盘等）在系统初始化之前就需要预先执行一段 Firmware，以对该设备的基本初始化。这段 Firmware 放在扩展 ROM 中，PCIe 配置头中的 Expansion ROM Base Address 寄存器用来指示及控制 ROM 相关信息。该寄存器格式如下图所示，其中指定了是否开启扩展 ROM、ROM 状态、ROM 基地址等。



▲图： Expansion ROM Base Address Register

670. PCIe Config Space Header 中的 Expansion ROM Base Addr 寄存器是做什么的？

这个寄存器存放了上位机扩展 ROM 的基地址，ROM 可以是 EP 上的 EPROM。这个 ROM 是给上位机用的，内部存放驱动程序，由上位机加载后运行。比如 ROM 内烧一个 PXE 系统用来做远程启动。

671. 哪里能查询最新的 PCIe Class Code？

这里：https://admin.PCI-ids.ucw.cz//read/PD

672. Linux 系统中 PCIe Class Code 有什么讲究吗？是不是不能为 FF？

Config Space 中的 Class Code 是用来指示设备类型的，0xFF 表示未分配的设备类型。

673. PCIe Capability 地址是链表形式的，该如何确定哪些地址是省下来可自用的？

虽说 PCIe Capability 寄存器地址是链表形式的，但在同一 RTL 内实现 Capability 寄存器时候，其地址大多是固定地址的，Synopsys、Cadence、PLDA 都是这么干的。比如把所有 Capability 寄存器固定放在 0x700 之前，那么 0x700~0x1000 这段地址空间就可以省下来自用。对于不同厂家、不同版本的 PCIe IP 设计，同一 Capability 地址可能不同。

674. 如果只用到 4 KB 配置空间中的一部分，剩余空间能省下来吗？

可以。Controller 单个 Function 内一般用不了 4 KB 那么多寄存器，不需要完整分配 4 KB RAM 来放寄存器，只有有意义的地址才需要实现实体寄存器，其他未占用的地址可以放其他自定义的寄存器。比如 Synopsys PCIe Controller 中，便把它的 Port Logic 寄存器放在了 4 KB 配置空间内。

675. RCRB 是什么？

RCRB 全称根复合体寄存器块（Root Complex Register Block），在常规配置空间之外额外指定 4 KB 存储空间给 RC 用，该 4 KB 空间所在的地址空间不能与常规配置空间或

内存空间出现重叠。RCRB 使用方式与常规配置空间类似，可用作 PCIe 的扩展能力寄存器或其他指定寄存器

BDF

676. 枚举时 Host 是如何给 Device 分配 BDF ID 的？我理解系统上电后，Device 自己不知道 BDF，Host 是如何通过深度优先算法能够实现呢？

枚举时，Host 只负责分配 Bus Number，Device 和 Function Number 是硬件写死的。PCI 同一 Bus 下可以有多个 Device，但 PCIe 是点对点的，PCIe 同一 Bus 下只有一个 Device 即 Device 0，枚举时 Host 会把该设备下已开启的 Function 扫一遍。即便某一 Bridge 下挂了两个 Device，这两个 Device 也都是 Device 0，两者 Bus Number 不同。

677. 手动改 NVMe 盘 Bus Number 大于 0x80 无法工作，可能是什么原因？

现象描述：手动修改 NVMe 盘的 Bus Number 为 0x80 或以上时无法工作，直接 Panic，小于 0x80 正常工作

暂无结论，怀疑 Controller 限制了最多 128 个总线号。

678. PCIe 设备最多支持 8 个 Function，该如何理解这里的 Function？

在 PCIe 中，每个连接到主板上的设备都有唯一的地址，被称为 BDF（Bus，Device，Function）。其中，Function 表示同一个设备上不同功能的编号。在 PCIe 设备 ID 和驱动程序匹配过程中，使用 Root BDF 和设备的 BDF 地址来确认特定设备的位置和身份。每个设备必须要有功能 0（Fun0），其他的 7 个功能（Fun1~Fun7）都是可选的。

这是 Physical Function，还有 Virtual Function 的。

比如一种 PCIe 芯片，可以同时支持声卡、网卡；对于上层驱动而言就是 2 类驱动，声卡驱动和网卡驱动。对于这个 PCIe 芯片（PCIe Device）而言就是两个 Function 来实现的。虽然在一个 Device 内部，但实现了两种不同的功能，即 Function。

那网卡声卡同时和 Host 通信，需要分时复用，2 个 Function 也只有一个 PIPE 接口吧。是的，一个 PCIe 接口，实际上就是分时复用。但是，是硬件层面做的分时复用，软件感知不到的。

679. PCIe 设备多 Function 一般是什么场景呢？一般是怎么用的？

譬如一个 PCIe 的无线网卡，同时支持蓝牙，这就是两个 Function;；再比如显卡上集成声卡，GFX 作为 Function 0，声卡 Function 1；再比如网卡开 SRIOV，也是多个 Function。

在硬件上多 Function 共存、共享同一物理接口、共享带宽、分时复用。

上层应用通过 BDF 来区分不同的 Function，通过内存映射将不同 Function 映射到不同 Memory Space 上，比如 0x8000000-0x9000000 给 Function 0，0xa000000-0xc000000 给 Function 1。

释放 CPU 资源，每个 VF 和虚拟机一样，可以单独 Reset 和其他管理，可以做硬件隔离，硬件隔离就可以做安全了。

680. PCIe Phantom Function 是什么？

Phantom Function，“幻影” Function，看似存在实则不存在的 Function，用于增加单 Function 设备暂存的 NP 事务数量，从而增大 Outstanding 能力。

一般情况下，PCIe 设备中每个 Function 都有一块空间暂存该 Function 发出去的 NP 请求，共 8 块，每块暂存的 NP 请求数量上限取决于 Tag 字段宽度（8-bit TAG -> 256）。若设备只有一个 Function，那多多少少有点浪费。Phantom Function 正是看中了这一点，当设备只有一个 Function 且支持 Phantom Function 时，通过使能设备控制寄存器中 Phantom Function Enable 位，将 BDF 中的 Function Number 字段当 Tag 用，这样每一个 Function 的 NP 暂存空间都能被利用到，从而将设备的 Outstanding 能力变为之前的 8 倍。

注意：Phantom Function 不是 Virtual Function，两者不是一个概念。

681. PCIe Phantom Function 有何局限？

由于开启了 Phantom Function 的 PCIe 设备其 Function Number 被挪作他用，开启了 Phantom Function 的设备有以下局限：

- ✓ 不支持 ARI（Alternative Routing-ID ）；
- ✓ 不支持多 VF；
- ✓ 开启 VF 后不支持 PF；
- ✓ 不支持 ATS（Address Translation Service）；
- ✓ 不支持 IDO（ID-Based Ordering）。
- ✓ 由于 Phantom Function 有以上诸多局限，非必要时一般不采用 Phantom Function。PCIe 4.0 引入了 10-bit Tag 能力，Outstanding 数目变为了 1024，绝大部分情况够用了。

682. Shadow Function 是做什么的？

采用 Shadow Function 能在一定程度上增大 PCIe Outstanding 的数目上限。

不同 Tag 宽度对应的 Outstanding 请求数不同，5 bit Tag 最大支持 32 Outstanding，8 bit-256，10 bit-768，之后再往上没法扩展了。而 Shadow Function 为了在不增加 Tag 字段

的情况下提供 Multiple Transaction ID Spaces，允许超出规范架构特性限制的 Functions 使用其他未实现 Functions 的资源。

假如采用 10 bit Tag，最大 768 用完的时候。那么再加一个 Shadow Functions，它有自己的 Requester ID，意味着又可以用 768 个 Tag 了。

BAR

683. PCIe BAR 是什么？

BAR，Base Address Register，是 PCIe 配置空间中从地址 0x10 开始的 6 个基地址寄存器，用以存储 PCIe 设备在 **PCIe 地址空间中的基地址**。该基地址**非 CPU 内存的映射地址**，两者间存在映射关系。通过配置 PCIe 设备 BAR 可以实现 PCIe 地址空间的动态分配。

684. IO BAR 是什么？有什么作用？

PCI 最先是由 Intel 提出来的，IO BAR 或称 IO Space，是 x86 Legacy 使用的空间，x86 指令集层面就有 IO 指令。

x86 对于 IO 是有独立总线的，所以才会有特殊的 IO 指令。由于总线宽度有限、IO 地址不够用，需要扩展 PCI 设备，就把一些 PCI 设备的地址放到 Memory 地址上，称为 MMIO。不管设备地址放到 Mem Space 还是 IO Space，都可以说是一个 IO 设备。对于普通的 Device，尽量不要去使用 IO Space, 因为根本用不着。

另外，ARM 没有 IO Space 这一说。ARM 总线从诞生开始就是统一编址的，ARM 指令集原生就没有 x86 那样额外的 IO 指令，只有 Load/Store 一种访存指令，对应 Mem BAR 和 Mem 访问。为了兼容 x86 Legacy 的设备，可以 map 一段 Load/Store 空间，在 PCIe Controller 控制器上拿 ATU 转一下。

685. 为什么 PCIe 设备的配置空间里只有 BAR 寄存器定义了起始地址，没有定义 Size？

BAR Size 是 PCIe 设备自己通过 BAR Mask 指定的，枚举时先向 BAR 寄存器写全 F 再读回，BAR 内指示 BAR Size 的 Mask 位是写不进去的，根据读回的值判断 Size 是多大。

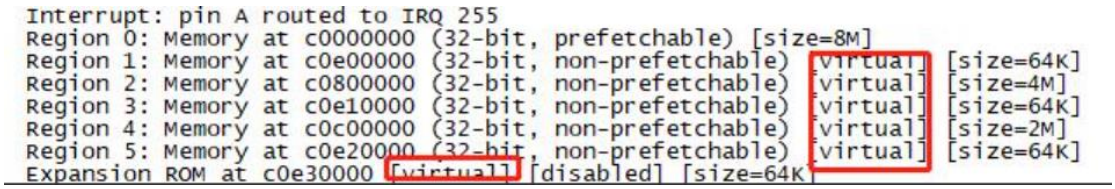
686. BAR 空间基地址分配时有什么要求么？

分配的基地址需要按照 BAR Size 地址对齐。

687. 有什么场景需要中途改 BAR Mask？需要在什么时候改？

在 Link Up 之前，可以通过 DBI 改 BAR Mask 的方式来指示 BAR Size。在 Link Up 之后，Host 既有可分配资源无法满足 Device 请求的 BAR Size 时，如果 Device 支持 Resize BAR，Host 可以来改 BAR Size 的方式来改 BAR Mask。BAR Resize 时，Host 可以来读取 EP Resize BAR 相关寄存器查看 EP 支持的 BAR Size，在 Host 能力范围之内选择一种 BAR Size 进行配置并给 EP 分配资源。

688. RC-EP 直连（非 NTB、Switch，未启动 EA）BAR 分配结果中出现 Virtual 字样是怎么回事？如何解决？



Virtual 表示这是一段并不存在的虚拟区域。

结论：是 EP 出现异常重启导致的。

689. 开启 IOMMU 后，PCIe BAR 里面是 Virtual Address 还是 Physical Address？

补充描述：印象中 BAR 里面是真实物理地址，只有 PCIe 设备往 DDR RAM 读写时使用的才是虚拟地址。

应该是 PA，因为 RC 的出方向上没有 SMMU，RC 的入方向才有。

690. BAR 配置时可操作的最低位为 12，是否表示<12 的位都不可以操作？那可以申请的内存为什么不是 2^12~2^31 大小呢？

枚举之前，主机不知道下游 PCIe 设备所需空间大小，通过向 BAR 内写全 1 来确定 BAR 空间的大小。可操作、不可操作就是对此处写全 1 的结果得出的。可操作的位与不可操作的位用途不同：

- ✓ 不可操作的位为 BAR Mask，指示 BAR Size。枚举时，主机向 BAR 内写全 1 然后读回，不可操作的位写不进去，故不可操作位读回值为 0。读回的值从低位到高位第一次出现 1 的位指示 BAR Size。
- ✓ 可操作的位用来设置改 BAR 的基地址。主机向 BAR 寄存器写全 1，读回值为 1 的位为可操作位。主机通过配置 BAR 内的可操作位，来配置改 BAR 的基地址。可操作的最低位为 12，也说明了不可操作的位至少为 12，即申请的 BAR Size 最少为 2^12 即 4 KB。

691. 现在还会发生 OS 把 BAR 安排到 High Mem 之后，由于设备指针长度不够导致 DMA 寻址不到 High Mem 的问题吗？

不会，RC 会根据 32b BAR 还是 64b BAR 做兼容。对于 32b BAR，正确操作是把该 BAR 基地址分配到 4 GB 以内，高于 4 GB 就可能会出现指针长度不够的问题。

我们原来设计了一个 Function BAR 512 M，开了 8 个 Function，服务器的 OS 直接分配失败。

692. EP 直接从 BAR 里读出 Pointer 使用可能会导致问题，因为有些平台上，这个地址并不是物理地址，RC 内部会翻译一次。不知道你怎么看?

这个过程是比较常见的，物理地址与虚拟地址转换，一般用到 ATS 或者 MMU。

代码里面做一下映射就行了。

693. 若访问到 BAR 空间中未使用的部分，比如 BAR 的大小是 256 KB，但只使用了 200 KB，当读取的范围包含后面 56 KB，应该如何回复 Cpl 呢？

总是要回的，回 sc 数据要么给全 ff 要么给个特殊值自己能够识别出即可。若访问前即知道实际使用了 200 K，可控制请求的长度，不去访问不存在的 56 KB。

694. 设置了 64b 可预取的 BAR Base 为 0x2_4000_0000，BAR 配置的低四位为 4’b1100 (0xe) 指示 64b/可预取，这样的话 0x2_4000_0000~0x2_4000_000d 会落在该 BAR 范围内吗？

会落在该 BAR 内，在判断是否落在 BAR 内时不看 BAR 的控制位，BAR Mask Bits 都不看，不影响 BAR Base。

695. IP 生成的 BAR 大小，EP 软件再次配置 Base Address Register 更改 BAR 的大小还会生效吗？

BAR Size 不一定能改，这取决于 IP 设计。Synopsys 的 IP 可以通过改 BAR Mask 寄存器修改 BAR Size，Xilinx 的 IP 未看到有相关寄存器。如果支持改 BAR Size，一般在 Link Up 之前配置的才会生效，即放在固件或配置 EEPROM 里，芯片上电后、Link Up 之前就进行 BAR Size 配置。

PCIe NT 可以通过配置改 BAR 大小。

696. Resizable BAR Capability 一般用来做什么？是不是 Function 支持多个 BAR Size，软件选一种 BAR Size？比如 Function 支持 2 M 和 4 M，软件是不是就可以配 2 M 或者 4 M？

Resize BAR 多用于申请大空间，显卡用的多。BAR 只是最小空间，当申请空间较大时可用 Resize BAR 进行申请。

正常情况下，在设计人员生成 EP IP 时就需配置/选择各个 BAR 的 BAR Size，设计人员根据评估设计所用的存储资源来选择。IP 生成后，BAR 的容量就是固定下来了。与

BAR 相比，Resize BAR 可理解为 EP 给出多个 BAR Size 供软件选择，Host 根据情况选择一个 BAR Size。Resize BAR 底层仍然是配置 BAR，只是多了个识别、协商的过程。

697. 哪些设备的驱动支持 BAR Resize？内核支持 BAR Resize 吗？

Resize BAR 是现在推的一个 Feature，尤其是影响到 eGPU，AMD 和 NV 显卡都有 Resize BAR 驱动。内核也有相关流程，Kernel 里面搜索 “pci_rebar*”即可搜索词 BAR Resize 相关函数。

698. 某个 BAR Mask 空间大小 A 后又 Resize 为 B，A 和 B 之间有什么关系么？

A 和 B 之间没有直接关系，相当于用 B 替换了 A。最终 Host 分配 BAR 空间依据的是当前 BAR Mask，软件上相当于直接把原来的 BAR 资源替换了。

699. BAR Resize 是设备主动发起的是吗？Resize 之后需要通知 Host 重新分配吗？

如果设备的 BAR Mask 支持动态改，设备可以通过主动写 MAR Mask 的方式来走 BAR Resize；如果设备支持 BAR Resize Capability，则需要 Host 来写该 Capability 中的寄存器来做 BAR Resize。无论如何，改了 BAR Size 之后都需要 Host 重新分配才能生效。

700. 如果 PCIe 设备只有 Mem Space 没有 IO Space，可以说是一个 IO 设备吗？

可以。PCIe Space 只有 IO Space 和 Mem space，不论过放在哪个 space 都可以叫 IO 设备。

701. 有没有这种使用场景：EP 自己从 BAR 里读出基地址以获知自身 BAR 空间所处的 HPA 位置，然后基于这个位置去做下游的一些事情？EP 会拿这个地址做什么事情？

我们是有业务的 Doorbell 需要精确匹配 64 bit 地址，采用了以上方式。

702. RC 没有 BAR，那么 RC 是否会把 EP 发来的任意地址 Mem Req 上传给 Host？

不是。如果想要把 EP 发来的 Mem 请求发给 Host，首先 RC 得打开 MSE，其次地址不能落在 Mem Base/Limit 范围内。。

703. PCIe BAR 寄存器存放的是 CPU 域的 System Memory 地址吗？

不是，至少在 x86 中不是。在 x86 中，PCI 域地址一般跟物理地址相同，即 BAR 内地址为物理地址；CPU 域地址是虚拟地址，CPU 发出的地址通常需要经过内部的桥进行转换才能访问到 PCI 域的地址，映射关系是要通过相应的寄存器配置的。

当然，如果有地址转换单元处理，也可以把 BAR 配成和系统地址空间一样。

704. 既然枚举时 Host 从 System Memory 空间找一块合适的空间给 BAR，为什么 BAR 地址不是 CPU 域地址？

因为 BAR 中的地址是针对 PCI 域的，PCI 域地址一般跟物理地址相同，BAR 内采用 HPA。CPU 访问 BAR 实际上是经过 MMU 映射 HVA 到 HPA 的过程的。MMU 是 CPU 侧的，IOMMU 是外设侧的，服务于 PCIe 地址空间。如果关闭了 MMU，操作系统就不分 VA/PA 了，对 CPU 而言 VA=PA；如果关闭了 IOMMU，IOMMU 直通就可以了。

705. 如果开了 SRIOV，BAR 的枚举和使用会有什么不同？

如果开了 SRIOV，就不仅仅 Host 有 VA/PA，每个 Guest（即 VM ）也有自己的 VA/PA。这样一来就有了很多地址域。无论如何，BAR 的地址还是 Host PA，但是 PCIe EP 的 Master Cycle 发出去的 Request 就变得复杂了，需要 IOMMU 统一处理成 HPA。GVA 到 HPA 的转换，可以通过 TLP 的 Requester ID，也可以通过 Prefix 上的 PASID，有几种方式。

706. 对于 64 位的 Host 主机，可以认为物理空间有 2^64 Byte 吗？

不是，一般只有 48 位左右，没有真正的 64 位地址。之所以没用 64 位地址，个人理解是因为 48 位地址空间已经非常大了，加之出于总线面积的考虑，暂无必要支持到 64 位。

707. 实地址模式该怎么理解？有什么应用场景？

可以理解为关闭 CPU 的 MMU 地址隔离跑裸机，风险较大，多用于关闭 MMU 跑 DOS。实地址模式有一个好处，即 Debug 硬件特别方便，DOS 比 JTAG 好用多了。

708. Windows 系统下怎么使用 Resizable BAR？

问题描述：似乎 OS 默认不会去改，等 KMD 起来又来不及了。

可以在 BIOS 下面开启。Resize BAR 的启闭是 UEFI BIOS 自己控制的，Device 的 VBIOS 最多是控制功能。如果无法使用 Resize BAR 功能，可能是 BIOS 不支持（没有该选项），也可能是 BIOS 内未打开。

如果是 BIOS 不支持且希望通过 Resize BAR 来扩大 BAR Size，一方面可以改设备固件的 BAR Size 大小，另一方面可以联系主板厂商更新 BIOS 支持。需要注意的是，如果一开始的 BAR Size 就报的很大，32 位的 UEFI 可能会起不来，需要 UEFI 启动之后再改大。

709. 如果 Host 要支持 Resize BAR 对 Host 硬件有什么要求吗？

Host 端 BIOS 支持即可，硬件应该没有特殊要求。

710. BAR Size 只支持二次幂设置，但设备内部只支持 96 G，驱动一般怎么处理该情况？

对于单个 BAR，设备的 BAR Size 可以多设置一点到 128 GB，在用的时候注意不要用 96 G 以上的部分；可以少设置一点为 64 GB，剩余 32 GB 不用；或者多配一个 BAR，一个开 64 GB，另一个开 32 GB。

711. Resize BAR 可不可以按照实际大小设置为非二次幂？

不能。都是 2 的幂次方。

712. Outbound 的地址可以不在 BAR 空间内吗？Controller 会做检查吗？

对 RC 而言，Controller 会检查，Outbound 方向的地址需要落在 Memory Base 和 Limit 内，跟自己的 BAR 无关。对 EP 而言，Outbound 方向的地址跟自己的 BAR 无关，Controller 不会做检查。

访问问题

713. Host 的 CPU 如果读到 2 个相同的 Vendor ID 和 Device ID 会报错吗？

Host 读到相同 Vendor ID 和 Device ID 的设备不会报错，Host 通过 BDF 对这些设备加以区分。

就像是同一电脑上插了多个同款 SSD 或 U 盘，其 Vendor ID 和 Device ID 是相同的，并不会报错。

714. 服务器访问网卡 BAR 空间，发现 BAR 空间数据更新很慢，可能是什么原因？

问题详述：在海光服务器上访问客户网卡设备，遇到下述 BAR 空间访问问题：BAR 数据空间里的数据更新很慢，无法读到最新数据，甚至出现读到新数据之后又读到老数据的情况。猜测是存在缓存不一致？该设备在 Dell 服务器上不存在该问题。

可排查设备的 BAR 设置是否开启了 Cache/Prefetch 属性（如下图 bit3）。若透过桥看不到设备 BAR Prefetchable 的实际值，可能需要根据上面的 PCI 桥去看该设备的设备空间具体是在哪个空间的。

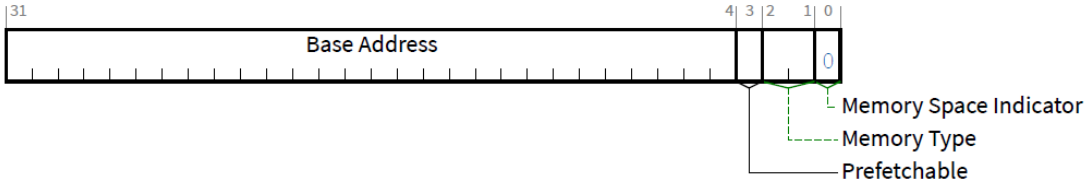


Figure 7-11 Base Address Register for Memory

反馈：lspci 看 BAR0 是 no Prefetch，BAR2 是 Prefetch，但是两个 BAR 都不行。

可排查是否为工具问题，是否工具未写进去。

可排查是否驱动问题，驱动理应维护缓存一致性。

其他经验参考：我遇到的 RC 不管是 Prefetchable=0/1，均按照 Non-Prefetch 来处理。不知道海光策略如何。

715. Type 0/1 配置空间 Command Register 的 Memory Space Enable 默认 0 不响应 Memory 空间请求，在使用之前是否需要主动配 1？

是的，Enable Device 或者枚举之前需要置一。

716. PCI 设备建链后配置空间变成全 0xffff，可能是什么原因导致的？

大概率是链接失败了，可以读一下状态机确认是否到 L0 了。

也可能是进入 L1 后未退出，可以查看 D3 状态。如果不想进 L1，可以把 D3 关闭。

反馈：把 D3 关死后，不会进入 L1 了。通过 D3 睡眠再写 D0 唤醒，唤醒后配置空间还是会极小概率变为全 F，协议分析仪抓包分析发现还是有 D3 写入，且接了分析仪上述问题出现的概率会小很多。

猜测 D0 唤醒动作之后，Link 还处于 L1 状态时就读了，即读配置空间时还未唤醒。也就是说，出问题的时候，很有可能还是在 L1 状态，没有进入 L0。进出 Low Power 是容易出问题，可以尝试把 PIPE Interface 拉出来看看，看下出问题的时候 tx_elecidle，rx_elecidle 都是什么状态。如果是 L0 的话，应该都是 0。

如果怀疑概率性卡在 L1 没法退出，可以尝试主动让反复进 D3 状态。每次写 D3 都进 L1，再写 D0 回到 L0，如此反复，加速问题复现。

反馈：但是我 D3 关掉了哦，应该不会再有 L1 了吧，睡下去的时候也没进 L1 呀。

如果确认关闭了 D3 在 L0 状态下发生了上述问题，可能是 PHY 没调好，Link 上了后来又掉了，就会出现上述问题。这种情况不能用协议分析仪，因为它改变了你链路。协议分析仪是分析协议用的，不能分析链路。如果怀疑是跟电气有关的问题的话，用协议分析仪，通常会改变电气行为，会看到不一样的现象。可尝试以下方法来排查电气问题：

- ✓ 条件好用示波器，误码仪好好量一量；
- ✓ 条件一般，用一块 loopback 卡跑 external loopback；
- ✓ 条件再差，不停跑 Hot Reset 去抓 eq complete 值和 serdes 值。

717. RC 访问 EP 的 BAR2，却送到了 BAR0 对应的地址范围，可能是什么原因？

问题补充：BAR0 Size 1 MB，基地址 0x3000_0000；BAR2 Size 64 MB，基地址 0x3010_0000。

BAR2 基地址设置不争取，如果 Size 为 64 MB，应该设置为 64 MB 地址对齐。虽然设置了 0x3010_0000，但实际地址的 bit 20 未写进去，刚好跟 BAR0 基地址重叠了，因此走 BAR0 出去。

718. 39 位的怎么寻址 1T 或者 4T 的硬盘？

问题描述：即便是靠 MMU 做地址转换，39b 地址位数也不够 1T，CPU 域没有足够与之匹配的地址。比如我们点鼠标保存 Word 文档，文档存在硬盘里，这个过程大概是什么？

硬盘地址并非直接由 CPU 地址转换而来，而是有跟物理地址相隔离的另外的地址域，通过 SATA、NVME 等协议转换而来。简单解释如下：对 CPU 的计算空间来说，硬盘的地址是 Data 而非 Addr。CPU 只需要能处理 64 位 Data，CPU 看到 SATA Controller 的控制地址，完成 SATA 协议的控制即可，SATA 协议里面封装了硬盘地址。

点击保存前，文档保存前存放在 DDR 里面；点击保存后，SATA 控制器里的 DMA 从 DDR 搬运到硬盘内，由 SATA 内部的 DMA 完成地址域的转换。

719. Host 做 BAR Resize 失败了，可能是什么原因？

BAR Resize 一般需要配合 BIOS 中的 “resize bar”和“above 4 G”一起使用，跟 Host 可分配资源大小也有关。

比较大的 BAR Size，比如 4 GB Size 及其以上，有时候由于一些限制不一定能重新分配成功。

另外，假如设备起初分配的地址在一堆设备中间，或者桥下有多个设备，可能都会造成 Resize 后重新分配失败。

720. RP 配置为 Outbound，AP 分配了一个 4 G 的 BAR 给 GPU，这个一般用来干什么的？

映射显存用的，直接把显存暴露给 UMD。

通过 BAR 映射的显存，Host CPU 一般也不太会直接对其发起数据面的访问，效率太低了。数据面还是走 IO 通过主存拿或者是 P2P 方式。

显存都是 GPU 发起的，可以理解为 GPU 的 DMA 来搬运，可能为别的方式，肯定不是 CPU。

枚举

枚举规则

721. PCIe 枚举跟 PCIe 初始化是一个意思吗？

PCIe 初始化跟 PCIe 枚举是两个不同的概念。PCIe 枚举可以理解为软件扫描设备树、分配 Bus 号、分配 BAR 空间、扫描 Capability 等等；PCIe 初始化一般是在枚举之前 Firmware 的操作，比如配置 Device ID、BAR Size、开 Capability 以及建链所需的 Controller、PHY 配置等等。

722. BIOS 初始化 PCIe 设备以后，在进入到 OS 时 OS 还要对 PCIe 设备进行一次枚举吗？这两次枚举的主要差异是什么？

是的，进入 OS 后还要再做一次枚举。关于两次枚举的差异，了解不多，列举如下：

两次枚举在 BAR 上有些区别。在 x86 中，BIOS 在初始化 PCIe 设备后分配的 BAR，OS 驱动再去枚举时，除了错误资源外不会重新分配 BAR；当然，如果 x86 Linux 系统参数来配置了预留 BAR 资源，会走调整资源分配的流程并调整 BAR。对 ARM 而言，OS 二次枚举时会重新分配 BAR。

723. RC 枚举之前怎么确定自己的 ID 的？

所有直接连接到 RC 的设备都被认为在总线 0 上，RC 首先访问总线 0，并开始查询每个可能的设备位置。

724. RC 发的 Cfg 包的 Requester ID 中 Device Number 和 Function Number 只能是 0 吗？

理论上是的，响应方实现时候硬件不要做的太死，请求方来了什么 Requester ID 就响应什么 ID 即可。

725. EP BDF 的 Function Number 一定大于等于 RC Requester ID 的 Function Number 吗？

问题存在问题。RC 和 EP 的 Function Num 没有直接关系，倒是 Bus 有关系。EP 发出的 Requester ID 中的 Bus Num 和 Device Num 由 RC 分配，EP 无法决定。

726. 枚举过程中，RC 发现某 EP 为 Single Function，在继续扫下一个设备的时 BDF 的 Function Number 不能再继续加 1 了吧？

枚举时软件能够从 Header Type Register 中的 Multi-Function 指示位得知当前 EP 是单 Function 还是多 Function。如果是单 Function，则 Function Num 归零继续扫下一 Device Num；如果为多 Function，则保持 Bus Num 和 Device Num 不变扫描 Function 0~7，Function Num 不一定连续，必须完整扫描 0~7 后才能扫描下一 Device Num。另外，在 PCIe 中，一个 Bus 下只有一个直连的 Device，Device Number 为 0。

727. ARM64 如果走 ACPI Scan Root 流程去枚举，还会走到传统 Probe 流程的枚举吗？

应该是只走一个。Probe 流程会调用 Kernel 里 DesignWare 相关配置，ACPI 不需要 DW 配置也不会去调。

728. 基于 ECAM 机制访问 EP PCIe 时，EP 收到的 TLP 是 Cfg 还是 Mem？

ECAM 是走 Cfg。

729. 枚举过程中，主机会扫描 PCIe Switch 内部虚拟桥并读取 P2P 桥的 Vendor ID，该读请求位于总线上，下游所有 P2P 桥都会收到这个信息，是如何知道哪个是 Device0、哪个是 Device1 的？

是的，Device ID 在 Switch 内设定死了。

730. 枚举完毕之后，EP 设备的 BDF 信息是保存在配置空间中吗？

配置空间中没有保存 BDF 的地方，EP 设备自定义寄存器来实现。EP 设备会记录收到 Cfg 请求中的 Bus Number，Device Number 为 0 是定死的，Function Number 是 EP 设备内部决定的。

731. 配置空间里的 Device ID 跟 BDF 中的 D 是一回事吗？

配置空间地址 0x2 的 Device ID 跟 BDF 里的 D 不是一回事。配置空间中的 Device ID 是厂商自定义的一个设备码，用来区分不同设备，跟驱动做匹配。BDF 中的 D 指示当前 Device 在设备树中的位置，对 PCI 而言这个 D 是软件枚举出来的、不固定的，对 PCIe 而言这个 D 只能是 0。

732. 枚举好的 BDF 信息放在哪里？

对 RC 而言，只要 RC 发出某 DBF 的第一笔 CfgRd 后，EP 回了 RC 就知道 BDF 是否存在，可以不用存。对 EP 而言，EP 收到 RC 发来的第一笔 CfgWr 后，EP 会解析包里的 BDF 值当成自己的 BDF，并进一步放在 EP 内部寄存器中。

733. 对于 RC-EP 点对点的 PCIe 拓扑，EP 发起 MRd Request TLP 中 Requester ID 位置的 Bus 号是 0 还是 1？EP 硬件知道自己所处的 Bus 吗？Bridge 是有寄存器记录 primary Bus，EP 中是否有这样的寄存器？

枚举时候 Bus 号随着配置读写发过去了，EP 端会记录 BDF 号，具体记录位置取决于设计具体实现。一般记录在 MAC 层寄存器中，EP 里 CPU 的 Firmware 直接能读取到。

734. PCIe 接多个不同的 Device，是软件可配置还是硬件枚举固定不变的呢？

OS/BIOS 枚举。

735. 捕获 Bus Number 怎么理解？在枚举过程中，EP 收到的都是 Type 0 吧？这个时候自己的 Bus Number 还是零，应该匹配不上，捕获了以后怎么用呢？

捕获就是提取 TLP{中的 Bus Number，赋值给自己的 Bus Number 寄存器。配置空间中没用 Bus Number 这个寄存器，在 Controller 内部自己实现。

736. EP 在一条链路一端是只有一个设备 Device0 吗？比如阵列卡，是将每个 SSD 当做一个 EP 吗？

是。

737. 分配 BAR 空间的时候，如果 Device 只需要 512 KB 的空间，我能分给它 1 MB 吗?

不可以。PCIe 硬件只识别 512 KB。BAR 大小是 EP 定，RC 只分配地址。

738. 对于 RP 和 DSP 而言，在枚举之前不知道其下方为 EP 还是 Bridge 的 USP，扫描设备树时软件如何知道该发 Cfg0 还是 Cfg1？

软件发的时候都是发 Cfg1。该笔 Cfg1 到达 RP 或 DSP 后，如果 Cfg1 中的 Bus Number 等于其 Secondary Bus Number，即该 Cfg 请求是给 RP、DSP 直系下属节点的，其会转为 Cfg0 向下转发；如果 Bus Number 位于 Secondary Bus Number 即 Subordinate Bus Number 之内，则继续向下转发 Cfg1。在配置所有节点时，都是发 Cfg0。

739. Hot Reset 会影响此前枚举好的 BDF 表吗？

内核里有所有设备信息的表，对某个设备做 Hot Reset 并不影响 BDF 张表。Remove 和 Rescan 会改这张表：Remove 操作就会删除这个设备节点，Rescan 后会执行 Driver 的 probe 函数。

740. 软件上对 PCIe 设备做 Remove，在硬件上有什么动作？枚举树会删除这个节点吗？

个人理解软件上 Remove 在硬件上其实没做动作，无论是否删除该节点，能确定的是该节点软件无法用了。

741. 如果是 RC-EP 直连的场景，枚举过程可以直接跳过去吗？

理论上不可以，枚举期间不止扫描设备树，还会扫描设备能力、对设备做 BAR 等相关配置。如果 RC、EP 都是自己的芯片想要做类似 NVLink 的片间连接，且固件、软件都是自己写的，那随便。

枚举故障

742. 系统中只有一个 RC 直连一个 EP，枚举时 RC 可以从 Bus 编号 0 以外的数字开始枚举呢？为什么仿真时发出的 BDF=0x100 即 Bus Num=1？

可以从非 Bus0 开始枚举。Bus0 是一般 RC 内部的 Bus Number，所以会从 Bus1 开始枚举。不论 RC 用什么 Bus Number 进行枚举，EP 都会记录第一个 CfgRd0 中的 Bus Number。如果后续收到了其他 BDF 的 CfgWr0 请求，EP 仍然会记录并更新其记录的 BDF 值。

743. 挂在桥下的设备在上电过程中出现下述 BAR 分配错误，可能是什么原因？

```
pci 0000:04:00.0: BAR 15: no space for [mem size 0xc00000000 64bit pref]
pci 0000:04:00.0: BAR 15: failed to assign [mem size 0xc00000000 64bit pref]
pci 0000:04:00.0: BAR 14: assigned [mem 0xdb600000-0xdb6fffff]
pci 0000:04:00.0: BAR 0: assigned [mem 0xdb700000-0xdb703fff]
pci 0000:04:00.0: BAR 13: assigned [io  0x2000-0x2fff]
pci 0000:05:00.0: BAR 15: no space for [mem size 0xc00000000 64bit pref]
pci 0000:05:00.0: BAR 15: failed to assign [mem size 0xc00000000 64bit pref]
pci 0000:05:00.0: BAR 14: assigned [mem 0xdb600000-0xdb6fffff]
pci 0000:05:00.0: BAR 13: assigned [io  0x2000-0x2fff]
pci 0000:06:00.0: BAR 0: no space for [mem size 0x800000000 64bit pref]
pci 0000:06:00.0: BAR 0: failed to assign [mem size 0x800000000 64bit pref]
pci 0000:06:00.0: BAR 2: no space for [mem size 0x00200000 64bit pref]
pci 0000:06:00.0: BAR 2: failed to assign [mem size 0x00200000 64bit pref]
pci 0000:06:00.0: BAR 5: assigned [mem 0xdb600000-0xdb67ffff]
pci 0000:06:00.0: BAR 6: assigned [mem 0xdb680000-0xdb69ffff pref]
pci 0000:06:00.0: BAR 4: assigned [io  0x2000-0x20ff]
```

设备 BAR 要的资源太多了，导致 RP 上的资源无法在 Host Bridge Windows 资源分配。一句话，给 Host 的资源池不够。

744. 扫描设备时报错 Invalid BAR 是怎么回事呢？

错误信息：PCI 0000:ff:1e.3: [Firmware Bug]: reg 0x10: invalid BAR (can't size)

这个 Device ID 和 Function Number 全 1 比较奇怪，需要确认这个 Device 是否存在。

如果确认这个 Device 和 Function 都存在，那么可能是 EP 驱动代码未对这个 BAR 做初始化，BAR Size 为 0。

745. lspci 查看和启动虚拟机设备，枚举过程中报 Unknown Header Type 0x7f，Header Type 不正确，一般是什么原因导致的？

7f 是全 1，可能是设备没插紧、设备不在线了，软件默认是全 1。

746. x86 CPU 在 BIOS 的 PCI 枚举时找到了设备，但进入 Kernel 后没有这个 PCI 设备。难道是进入 Kernel 后会重新枚举吗？

是的，进入 Kernel 后会重新枚举。枚举是操作系统做的，若是 Linux 会尽量保持独立，重新枚举以确保准确获得所有设备的最新状态和配置信息，地址空间、I/O 等也会重新分配（若开了 resize 等功能）。对于题中的现象，可以是否驱动的原因导致，卸载 EP 对应驱动试试。

747. 高通主板，RC 主动 Scan 热插入的 EP，PCIe Rescan 之后提示 EP BAR No Space，实际有空间指示 BAR Space 较小，是何原因？

需要预留空间，在 Linux 配置里做修改。（后续：改后有效）

748. PCIe 设备起来后，RC 端 CPU 通过 lspci 查看，出现 Advisory Non-Fatal Error，大概是什么原因？标准 x86 服务器 Intel 网卡没用这个错，用自研网卡（Xilinx PCIe IP）出现该错。上电后系统刚起来就有几百个错，后续业务阶段没用新的错误产生。

可能是枚举时 EP 回了 UR 导致的。如果是 UR 的话，大概是下述两个原因引起：

- ✓ 枚举的过程中访问到了不该访问的寄存器；
- ✓ 函数访问的 A 地址，但是包对齐到了 B 地址，导致出错。

更多讨论：

枚举时候，会访问 Function0-7，如果设备只有 Function0，访问 1-7 就会回复一个 UR Cpl，一般的设备会将 AER 中的 UR 对应的 Fatal 置为 0（就是这个错误是 non-Fatal 的，不是 Fatal 的），就会记录为 Advisory Non-Fatal ERROR。所以一般开机时出现了

Advisory Non-Fatal Error 可以忽略，开机后可以 AER 写 1 清 0，此后没有再置位就说明就是枚举时候访问不存在的 Function 的问题，这个是个正常服务器都会这么枚举，不是错误。Severity 位置 1（这个错误是 non-Fatal 的，不是 Fatal 的）。不过你也要看看 UR 有没有置位，如果 UR 置位说明就是上述问题，是其他 AER 置位需要再研究研究。

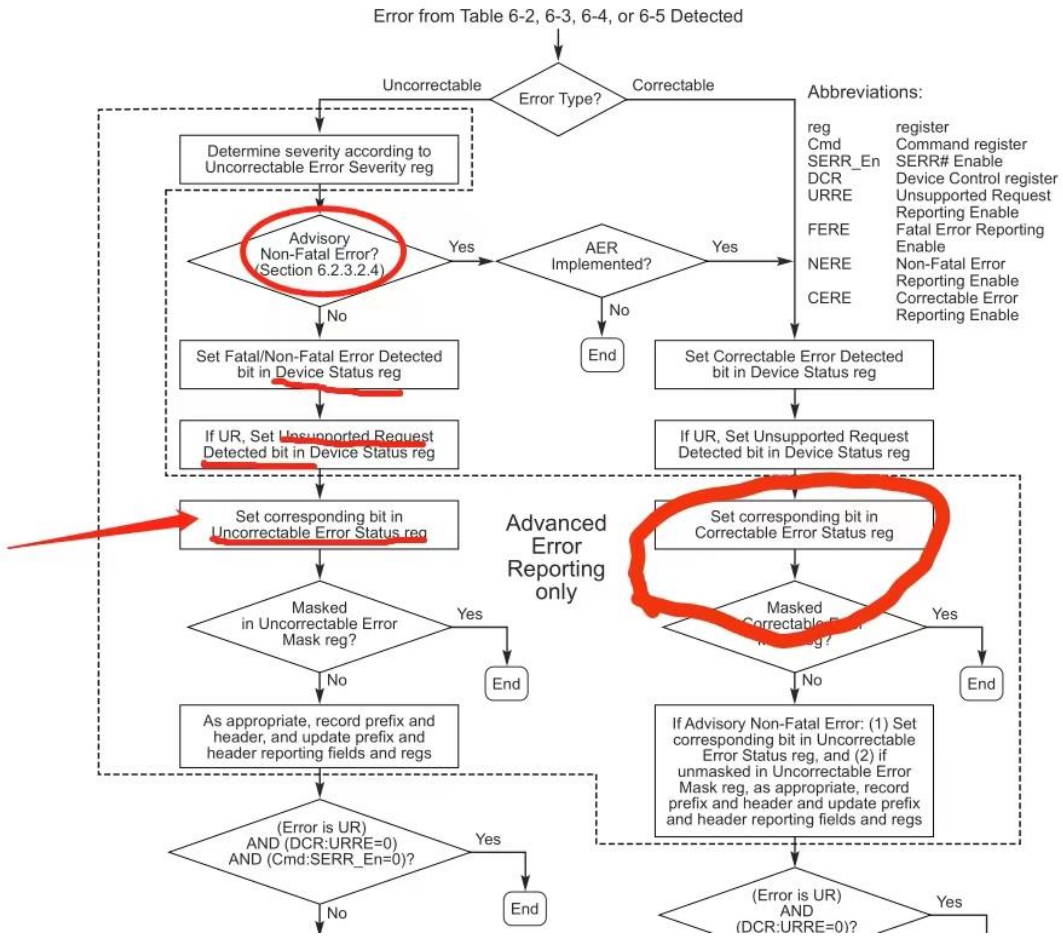
现在配置空间看 AER 里面就报了这个 Advisory Non-Fatal Error 其他 bit 都是 0。是的 Advisory Non-Fatal Error 包含的那些 UR 或者 Cplto 或者有毒包什么的错都没报。

通过 Device Statues 寄存器看到有 Unsupported Request detect 和 Correctable err detected

COR 指的就是 Advisory Non-Fatal Error，UR 大概率就是枚举访问了不存在的 Function，所以应该不是什么大问题，不过按照 AER 处理机制，AER 中的 UR 应该也要置位的，不知道你这个为何不置位。

这种不影响实际业务，不影响数据传递可以不用管，如果影响了再分析。

转化成 Advisory 的 UR，不置 AER 中的 UR。



这个是 No 分支，没有转化的走做成分值，转化的走右侧，Device Status 会置 UR，AER 不置，转化的目的就是不报 Non-Fatal，报 CE 不报 UE。

749. 华三服务器某特定槽位插上自研网卡能够正常 Link Up，但 PCIe 枚举时候服务器卡死，其可能是什么原因？

补充说明：该自研网卡插在其他槽位没有问题，该槽位插别家网卡也没有问题。

如果是建链失败，可能是由于服务器 BIOS 对不同设备在 PCIe 链路上的参数有差异，存在兼容性的问题。但链路已经到 L0 了正常的，基本可以排除建链问题。

当然，LTSSM 走到 L0 也不代表 DL 层 LinkUp 成功，可进一步排除是否 DL 未 Link Up。如果 DL Link Up 失败，可尝试在 BIOS 里彻底关掉该槽位的 DLF（Data Link Feature），有些很早的设备不支持该 Feature。

此外，可尝试用示波器测板卡上的 refclk 和 perst，对比一下有问题卡槽和没问题的卡槽是否存在差异。

如果还不能解决问题，可以找华三帮忙。

系统应用

启动问题/BIOS

750. 重启服务器，PCIe卡在BIOS阶段，用分析仪抓包发现Host重启后未下发任何Cfg包，该问题有何 Debug 思路？

描述：该现象非 100%可复现，但出现概率较大，冷重启、热重启均不行，只有拔电源线再插上才可以。

751. PCIe 开了 OPROM BAR，BIOS 里面有配置可以关掉不读吗？

不同 BIOS 有的有，有的没有。AMD 和 Intel 的应该有，鲲鹏的没有。如果有，可进 BIOS 的 PCIe Device List，可选择或关闭 Option ROM 加载选项、UEFI 等等。如果看不到可尝试 Ctrl+F11 加载隐藏项。

实在找不到的话，可尝试直接把 OPROM 刷掉。

752. FPGA 开发板插在电脑 PCIe 插槽，电脑黑屏无法启动，可能是什么原因？

RC 发的配置包 PCIe 卡回的有问题或者超时没回会出现上述现象，有的服务器就会卡在 BIOS。

753. 系统突然挂了是怎么回事？

一般有下述两种情况造成：

- ✓ PCIe 设备链路断开，Host 系统可能会挂死或重启；
- ✓ 上报 Error Message 也会导致重启，看 Error 的级别，可以去系统的 log 日志里面找找。

754. PCIe 板卡往服务器发数据，导致服务器重启，可能是什么原因导致的？

可能是 RC 端 ACS 没有设置好，导致板卡写到了其他空间地址。

755. 在 Boot 阶段，CPU RC 除了配置 PCIe 外设 Link Up 之外还会有其他的配置吗？比如配置 PCIe 外设 EP 的 MSI-X Table。

是设备驱动调用内核的 API 设的 Linux 的 API，不是 RC 的驱动，比如 NVMe 的驱动会配置。（参考：https://www.kernel.org/doc/html/latest/PCI/msi-howto.html ）

756. 成群的服务器如果某个芯片的 A53 死了，我虽然能通过 SMBus 用远端 HostCPU Reset A53，但 PCIe 本身是不是没法重新识别呀？

上游 Slot 有个 Rescan，echo 1 到这个位置试试。

ROM

757. PCI Expansion ROM 这个功能怎么用？需要真实例化一个 ROM 吗？

ROM 里可以放固件，固件中存放设备初始化或者 System Boot 的 Code。比如，对于网卡来说可以当作 PXE 使用；做 SSD Controller 可以把 UEFI Driver 放在 Expansion ROM 里，供主机使用；另外，一般 RAID 卡里面会用这个 ROM BAR，在 BIOS 阶段跑一段自己的固件代码做配置。

ROM 是真实的 ROM，具体实现方式比较自由，可以通过 SPI 或 NOR Flash 等实现，也可以用偏上 RAM 在 FW 上电初始化的时候写进去。如果不用 Expansion ROM 最好关闭 ROM BAR，因为枚举会去读这个 BAR，要是总线上没有做处理可能会 Hang 死。

758. EP 的驱动现在还会从 Expansion ROM 去读取吗？

不会，只有 BIOS 下才会去读 Expansion ROM。EP 的驱动是合到 Linux 里的，Linux 识别出 Vendor ID、Device ID 后调用对应的驱动程序就可以了。

写合并

759. PCIe EP 需要配置什么 Capability 来支持 Host 写合并（Write Combine，WC）？

当前现象：Host Enable Cache WC，一次写 64 B，但是 PCIe 波形内看到的仍然是 4 B 一次的写数据。

补充：对于不同的 PCIe EP 卡，有些可以收到一笔 64 B，但有些 EP 卡收到的就被拆分，可以排除非 Host CPU/RC 设置问题。

该问题暂无答案，可参考该帖子中的相关讨论：在用户空间中启用写合并 IO 访问

760. 写合并是什么意思？

写合并（Write Combine，WC）是 CPU Cache 属性，能够将多笔写合并为一笔，比如 8 个 8 B 的写合并成一笔 64 B 一次性写下去。

761. 为了实现 Host 写合并，PCIe EP 需要配置什么 Capability 吗？

问题补充： Host 端 AMD 的 CPU 调用 memcopy 往 Synopsys 公司 EP 卡高 4 G 的 DDR 空间写 64 B Cacheable 数据，汇编为 8 个 8 B 的 mov 指令。Host CPU 已开启 Cache 写合并，软件建页表时也配对了 Cacheable 属性。Host 理应把这 8 笔合为 1 笔 MWr，预期在 EP 端看到一笔 64 B 的写操作，但实际在 EP 内看到 8 笔 8 B 的写。

问题补充 2：换张 EP 卡后，同样的操作在 EP 端可以看到合并之后的一笔 64 B 写操作。说明 Host 写合并配置没有问题。这张写合并成功的卡关闭 WC 后，也能收到一个 64 B 合并后的写操作。

WC 写合并是 CPU 内部总线的行为，写 CPU Cache 时起作用，跟 CPU 往外写无关，跟 RP、EP 无关。

该问题跟下述问题类似，可参考：https://www.it1352.com/1562710.html

lspci

762. lspci 结果中，Region 有时是 Virtual、有时是 Disable，表示什么意思？

见下文问答。

763. 分配出来的 IO Mem 空间为什么被标记为了 Disabled？

```
00.0 PCI bridge: Intel Corporation Sky Lake-E PCI Express Root Port A (rev
     Physical Slot: 4
     Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- St
     Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort
     Latency: 0, Cache Line Size: 64 bytes
     Interrupt: pin A routed to IRQ 29
     NUMA node: 0
     Bus: primary=16, secondary=17, subordinate=22, sec-latency=0
     I/O behind bridge: 0000f000-00000fff [disabled]
     Memory behind bridge: 71800000-735fffff [size=30M]
     Prefetchable memory behind bridge: 0000000073600000-00000000753fffff [
     Secondary status: 66MHz- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort
```

Native EP 不支持 IO Transaction，不分配 IO Space。IO 与 Mem 地址独立，IO 没分配不影响下边 Memory 地址的访问。

764. 用 lspci 查看次级 PCIe 扩展能力结构不显示详细字段，可能是什么问题？

注：所用 lspci 版本为 v3.5.6。

一方面，可能是 OS 原因，实在不行可以把配置空间 Hex 整个 Dump 处理再查看；另一方面，可能是 lspci 版本问题，可尝试升级版本。

反馈：更新 lspci 版本后问题解决。

765. Windows 下有没有好用类似 lspci 的工具，能够读到超过 0x100 的配置空间？

有个 Windows 下 lspcie 的工具，链接为：PCI Utilities (lspci, setpci) for Windows，或者试试 SIV。也可以试试 LeCory 的 Telescan，安装协议分析仪软件的时候自带的。

拓展应用

766. CPU 下挂的 NVMe 硬盘容量大小有限制吗？容量大小跟物理地址空间有关系吗？

容量大小跟物理地址没关系，NVMe 硬盘不会映射到内存里。但是大小和文件系统还有硬盘分区有关系，有的不支持大容量。

767. PCIe 的 U.2、U.3 是什么意思？还有 U.2/U.3 的 Single、Dual 是什么意思？

U.2、U.3 就是个接口，企业级 SSD 的接口。Single、Dual 就是单口、双口，单口 x4，双口就是 2x2。

U.2 是 SFF8639 标准，U.33 实际没大规模用起来。U.2 接口现在走 SATA、SAS、PCIe 各种协议，在中国比较流行。北美比较流行 E1 和 E3。个人感觉 E1 和 E3 协议对信号完整性更友好，毕竟不需要一个额外连接器，阻抗更连续。

八、电气物理层

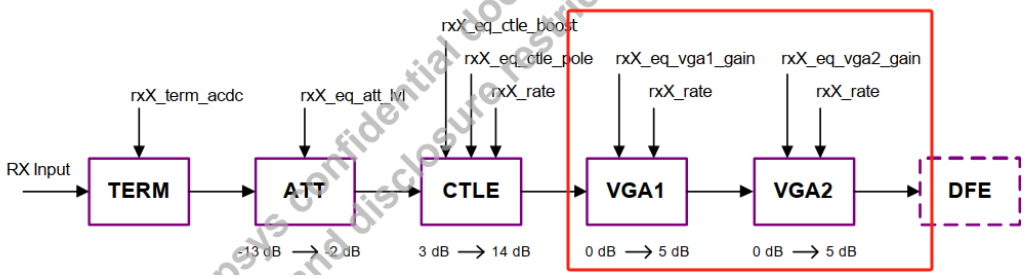
物理特性

768. PCIe Edge Connection 金手指 PIN 脚中的 HSIp、HSOp、Tx、Rx 的方向是相对谁来说的？HS 是什么意思？

HSIp/HSIn/HSOp/HSOn 的 HS 全称 High Speed，I/O 即 Input/Output，p/n 即 Posive/Nagative，Input/Output/Tx/Rx 是站在母板/系统板（RC）角度讲的。

769. Synopsys 公司有个叫 VGA 的特殊 PIO，这个是做什么用的？

Figure 4-3 RX Analog Front End (AFE) Diagram



5	**VGA Palette Snoop** ***Not supported*** Cleared, as required by the *PCI Express Base r2.1*.	RsvdP	No	0

VGA 是 Rx 模拟前端的放大器，控制器输出了一个 vga_en。

770. 差分信号 16 GT/s 要求 85 欧姆、8 GT/s 及以下要求 100 欧姆的目标阻抗，这在同一块板子同一组硬件 Trace 下如何实现？是否需要在 16 GT/s 时并电阻进去？

引文：The impedance targets for the Rx tolerancing interconnect environment are 100 Ω differential and 50 Ω single-ended for the 2.5, 5.0, and 8.0 GT/s channels and 85 Ω differential and 42.5 Ω single-ended for the 16.0 GT/s, 32.0 GT/s, and 64.0 GT/ s channels; the impedance tolerance should be maintained within ±5% or better.（PCIe Base Spec: 8.4.1.2 Calibration Channel Insertion Loss Characteristics）

协议中没有说是同一块板子。这块可参考下图。

4.7.8. Differential Data Trace Impedance

The PCB trace pair differential impedance for a 5.0 GT/s capable data pair must be in the range of 68 Ω to 105 Ω. The PCB trace pair differential impedance for an 8.0 GT/s capable data pair must be in the range of 70 Ω to 100 Ω. The PCB trace pair differential impedance for 16.0 GT/s and higher data rate capable data pair must be in the range of 72.5 Ω to 97.5 Ω. These limits apply to both the Add-in Card and the system board.

771. PCIe 在 Electrical Idle 时差分对上的电压是多少？

都在共模电压上，Tx 一侧共模为 Vswing/2。Rx 对地 term 的情况，共模为 0。TRx 中间是隔直电容。

772. PCIe 的 AC 耦合电容为什么多放在发送端？

AC 耦合一般都是靠近发送端放，目的是在信号一出来的时候就去掉共模部分，发送信号质量相对较好。当然你也可以 Tx 和 Rx 两边都放。一些走 Cable 的协议会建议两边都放，比如 DisplayPort，V-by-One 等。

对 PCIe 而言，要求 AC 耦合电容靠近 Tx 放，还与 Detect 机制有关系。在 PCIe 中，这个电容主要不是用来隔直流，而是 Device Detect 用的，放在 Tx 是为了方便 PCIe 插卡的应用场景。如果是 PCIe 本板互联，理论上放哪儿都一样，但实际行不通（原因未知）。无论如何，PCIe 耦合电容统一放 Tx 端，有个约定也避免了 Host 跟 Device 冲突。

773. 金手指到 PCIe 芯片的走线长度有要求，板上互联的 PCIe 走线长度有要求么？

有要求。

774. PCIe PHY 需要用到 Reference Voltage 吗，比如给 vco 一个基准电压什么的？

没见过需要参考电压的情况，一般会有一个外接参考电阻。听闻高通自行设计的 PCIe PHY 需要一个 QREFS，可能外围电路较复杂给了个参考电压。

775. 只用 Serdes IP 可以两边对传不包含任何协议的高速数据吗？

可以传自定义协议，但流控、可靠性等无法保证。

776. PCIe 协议中除了 CEM 规范还有哪个规范定义了 PCB 设计的插损和回损的要求？

CEM 是针对 AIC 的，如果产品是 AIC 那就至于 CEM 这一个规范。除了 CEM，PCIe 背板应用还有 M.2 相关 Spec。

777. PCIe 7.0 物理层支持光口，为何不直接走光通信？PCIe 不就淘汰了吗？

光通信是一种基于光的信号传输方式，在信道中传递光而非电信号。不管底层用物理层电还是光传输，不影响应用层、TL 层。而且 PCIe 涉及到平台架构、软硬件生态等，不会说淘汰就淘汰。

778. PCIe5.0 时 x4 槽位功率还是 25 w 吗？

槽位功率跟速率没关系，跟宽度有关系。不同宽度槽位初始功率及最大功率如下表。

Table 4-3: PCI Express CEM Add-in Card Edge-Finger Initial Permitted Power Draw at System Power Up

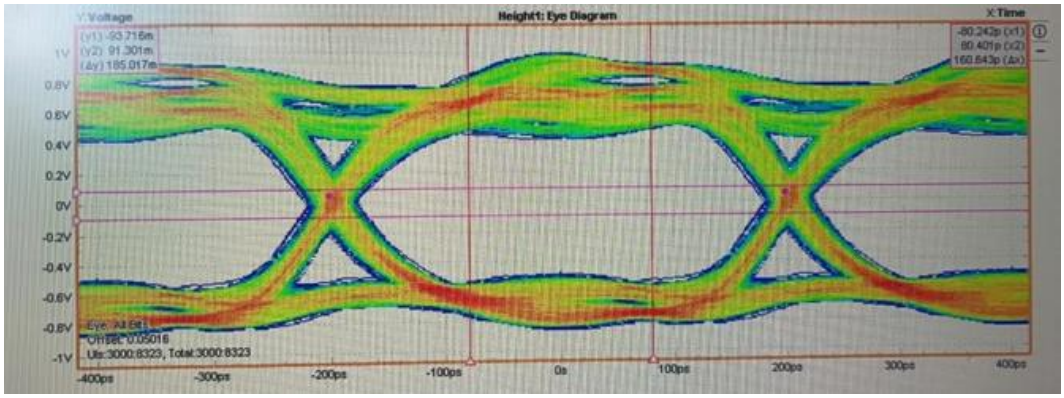
Card Connector Interface	Power Draw Permitted at System Power Up
x1	10 W available from edge connector
x4/x8/x16	25 W available from edge connector

Table 4-4: PCI Express CEM Add-in Card Edge-Finger Maximum Permitted Power Draw after Software Configuration

Card Connector Interface	Maximum Power Draw Permitted after Software Configuration
x1 low-profile	10 W available from edge connector
x1 half-length	10 W available from edge connector
x1 full-length–standard height	75 W available from edge connector
x4 / x8 / x16	75 W available from edge connector

电气特性

779. 下图为 Gen1 Rx 的眼图，这个电压幅值 1 V 是不是过高了？



协议规定 Vpp 不超过 1.2 V（如下），这里是有点高了。

4.3.4.5. Common Receiver Parameters

The following table defines the parameters for 2.5 GT/s, 5.0 GT/s, and 8.0 GT/s Receivers.

Table 4-24: Receiver Specifications

Symbol	Parameter	2.5 GT/s	5.0 GT/s	8.0 GT/s	Units	Comments
UI	Unit Interval	399.88 (min) 400.12 (max)	199.94 (min) 200.06 (max)	124.9625 (min) 125.0375 (max)	ps	UI does not account for SSC caused variations.
$V_{RX-DIFF-PP-CC}$	Differential Rx peak-peak voltage for common Refclk Rx architecture	0.175 (min) 1.2 (max)	0.120 (min) 1.2 (max)	See Table 4-22 and Table 4-23.	V	See Section 4.3.7.3.2.
$V_{RX-DIFF-PP-DC}$	Differential Rx peak-peak voltage for data clocked Rx architecture	0.175 (min) 1.2 (max)	0.100 (min) 1.2 (max)	See Table 4-22 and Table 4-23.	V	See Section 4.3.7.3.2.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

780. PCIe 链路信号质量不好时，实际一般调哪些参数来提高信号质量？

SI 跟 PHY、Package、PCB 都有关系，有时候单靠调整 PHY 参数难以彻底解决问题。对 PHY 而言，一般调 PHY 里面的寄存器，比如 Rx 端的 CTLE、DFE 等模块。至于 SI，我觉得最重要的一条就是阻抗要连续。

781. 如果 Local Tx 的 FS 没有设置默认参数也没有寄存器可配，那么其 FS 是怎么获取的？

PHY 内有寄存器来选择 Full Swing 或 Half Swing。至于 FS 的值，Cadence 的 IP 是根据每次 Power Up External Resistor Calibration 结果算出来的，SNPS 的应该是 PHY 设计的时候固定下来了，不会改变。PIPE 接口上有 LocalFS 这个信号，是 PHY 输出给 MAC 的，在均衡过程中，双方会互相发送自己的 FS，这样对端就能知道了。

测试

测试方法

782. PCIe 是如何检测到对端有设备存在的？是靠外围电路检测然后告知 PHY，还是 PHY 内部自己处理？

PHY 内部自己实现的，无需外围电路的参与。书上解答如下：

PCIe 总线规范定义了接收逻辑 RX 在正常工作状态下的 DC 共模输入阻抗 Z_{RX-DC}，其值在 40 ~ 60 Ω 范围之内。当接收逻辑 RX 的 V_{cc} 没有上电，V_{D+} 信号或者 V_{D-} 信号的电压伏值大于 0 时，其 DC 共模输入阻抗 $Z_{RX-HIGH-IMP-DC-POS}$ 最小为 50 kΩ；当 V_{D+} 或者 V_{D-} 小于 0 时，其 DC 共模输入阻抗 $Z_{RX-HIGH-IMP-DC-NEG}$ 最小为 1.0 kΩ。

由此可见当对端接收逻辑 RX 可以正常工作时，其 DC 共模输入阻抗远小于“没有上电”时的状态。发送逻辑 TX 通过监控 V_{D+} 和 V_{D-} 信号的电压伏值，可以获得接收逻辑在正常工作状态和 V_{cc} 没有上电时的电流曲线，从而判断接收逻辑 RX 是否处于正常工作状态。在 PCIe 总线中，发送逻辑 TX 通过“发送 Detect 序列”判断对端接收逻辑 RX 是否存在。

783. PHY 是怎么在训练过程中评估 BER 的？是通过 PRBS Pattern 吗？

不是。链路训练的均衡状态下，通过发送 TS 来评估 BER。

784. 如何统计 PCIe 的误码率？有何标准方法吗？

纯软件没法测，软件拿不到底层数据。

误码率一般用误码仪测，通过比对发出和接收的数据来计算误码率；没有误码仪的话，有些高速示波器也能测误码率；如果示波器没法测误码率，可以抓出眼图，根据眼图数据计算一个理论误码率出来。

上述方法用到了高速仪器，仪器比较贵，可以找第三方测试或者找厂家借仪器。目前能测误码率的没几家，有安捷伦、Tek 泰克、安立、Keysight。

如果不需要上实际系统，Serdes 一般都设计了多个层级的 Loopback 功能，Loopback 测试结果给个 Pass 或者 Fail 一般也能知道误码率满不满足要求。如果不能测 Loopback，对系统进行功能级压力测试，压测通过的话误码率大概也没问题。

785. 测 PCIe 链路上的眼图需要发特定 Pattern 吗？必须在 Loopback 下测吗？

没有要求必须在 Loopback 状态下通过发特定 Pattern 的方式测眼图。

对于 PCI SIG，其通过夹具连接到示波器，然后调用 sigtest 进行分析，这个是 Polling Compliance 下发特定 Pattern 测眼图。Intel 的 IOMT 还有海光的 HGT 都是在 OS 上用软件跑的，其中 IOMT 是 Loopback，HGT 是 L0。Synopsys 公司的 IP 提供一种不依赖示波器的眼图测试方法，可以在 L0 或 Loopback 时测眼图，通过 Dump PHY 寄存器来统计。在 L0 状态下看眼图属于 Non-Destructive 非侵入式，不影响正常业务传输，但存在一定可能会影响链路上的电气特性，导致进 Recovery。

仪器设备

786. 常用的 PCIe 测试设备有哪些？

大的有示波器、误码仪、网分、模拟器、协议分析仪，小的就比较杂了。

787. 协议分析仪是什么？是否需要购买硬件设备？

是硬件设备，需要购买。

788. Teledyne Lecroy PCIe Protocol Analysis 是软分析仪吗？是否需要厂家配套的硬件设备？

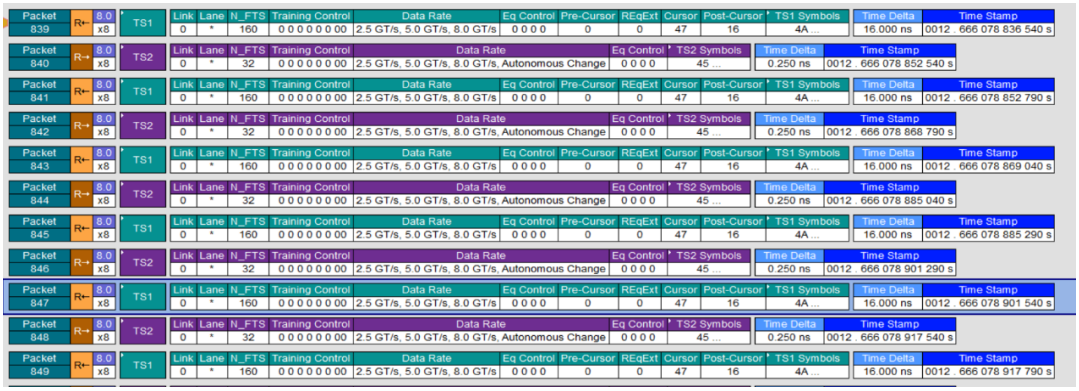
是个软件，可以离线使用，也可以搭配 Lecroy 的协议分析仪使用。如果需要抓包，则需要协议分析仪配合使用；没有设备只有软件的话，只能用来看寄存器，或者分析已经抓好的包数据。

789. Teledyne Lecroy PCIe Protocol Analysis 软件在哪里可以下载到？

找 Lecroy 的 AE 要，一般给个百度网盘的链接。他们网上上下不到了，要提交一堆信息等审批。

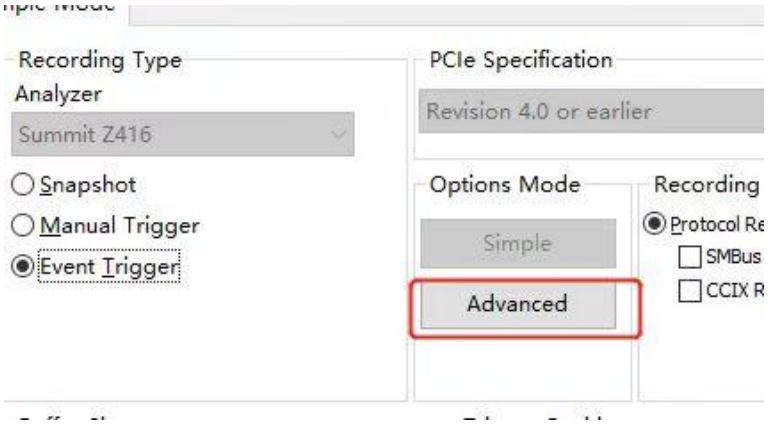
【资料】https://pan.baidu.com/s/1ZjJ0cHayvGhvyKGsGD7jqA?pwd=8888

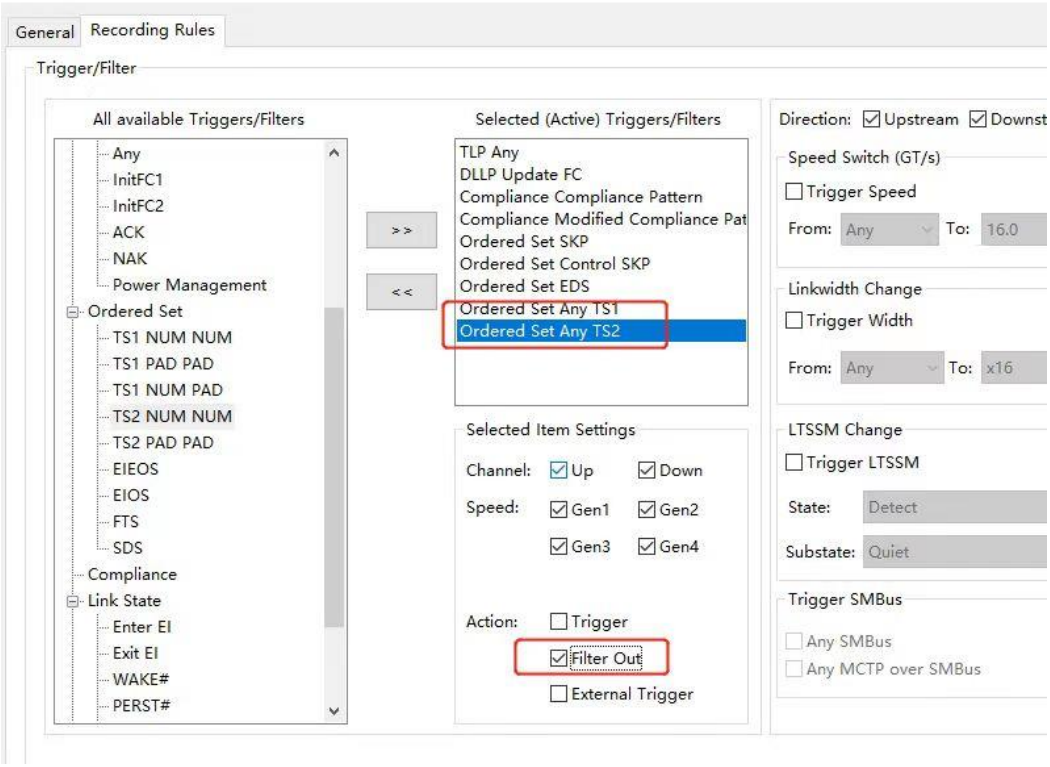
790. 力科分析仪抓包抓到很多 TS，该如何过滤掉这些 TS 序列？



若已抓包完毕，软件工具栏有个按钮，TS 相关点开打叉就可以滤掉 TS。

若需要在抓包过程中屏蔽 TS，即不抓取 TS，可在抓取设定->Advanced 中按下图选择 TS 类型并勾选 Filter Out。





791. 协议分析仪抓取硬盘 Trace 数据，显示 TLP 无效或 LCRC/ECRC 错误，该如何分析错误产生的原因？

讨论：如果链路训练没问题，查看是否协议分析仪抓包设置错误或链路不稳定。

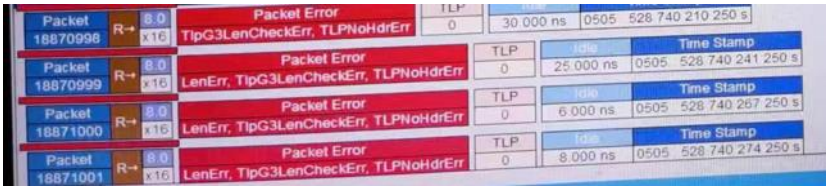
792. 能否从分析仪抓的报文里看出是否有 Warm Reset？

无法根据报文直接看出 Warm Reset。Warm Reset 非 Inbound，自定义成分较大，无法从报文中看出，可以在较大的时间窗口上根据报文推断出 WARM Reset 的发生。

793. Xilinx DMA 发送 IO 写请求，接口输出波形但 Analyzer 抓不到包，可能是什么原因？

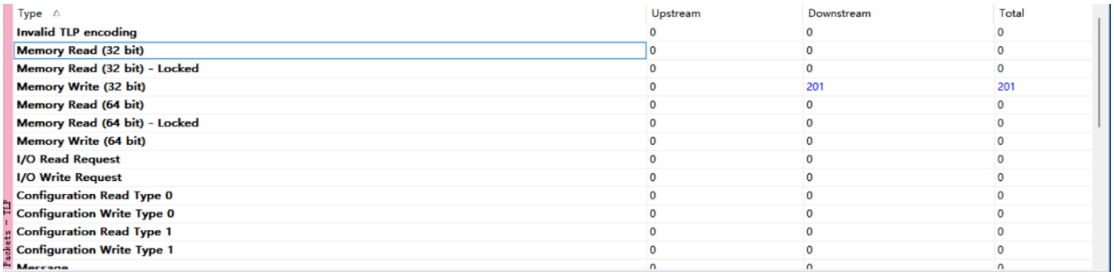
接口抓到波形代表不了 IO Write 发出来了， 可先确认 Link Training 是否完成。分析仪抓不到不代表没有，可先确认分析仪此前是否抓到了 TS 等包，已排除分析仪连接问题。如果确认分析仪连接正确且 PCIe 链路已经 Link Up，可排查分析仪抓包选项是否勾选 IO 请求。

794. 原型验证环境中降速测试，Host 超时报错，分析仪抓出下述错包，是何原因？



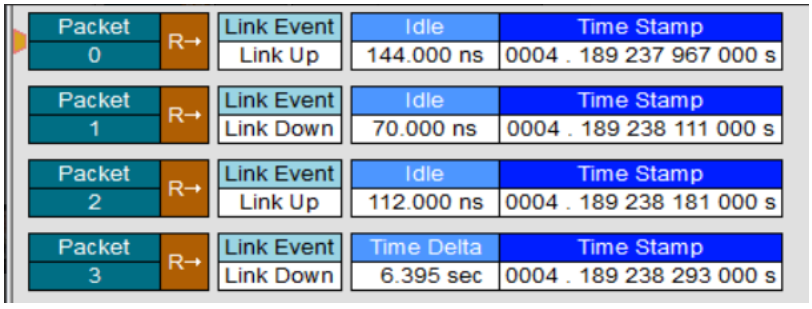
直接原因为发出的 MRd 在既定时间内没有得到 Completion。考虑到原型验证环境做了降速处理，回复 CplD 相对 Host 超时可以理解。至于接收到的错包，只根据上述包看不出原因。可以用 Gen1 测试基本功能。

795. 抓取 Trace 数据时只抓到下行的 TLP 数据，抓不到上行 TLP 数据，这是怎么回事呢？



可能是 USP 没发数据，或者是 Trace 配置中关闭了 USP 数据抓取。

796. 分析仪抓到了 Link Up、Link Down 等 Link Event，这个 Event 是怎么来的？



LeCroy 的协议分析仪应该是把进入 Electrical Idle 事件，包括没有 Common 电平，或者 Common 电平不符合要求等情况，视为 Link Down 事件。反之，退出 Electrical Idle 则视为 Link Up 事件。所以进入 L0s、L1 等状态时，用协议分析仪也能看到 Link Down 事件。

797. PCIe 5.0 协议分析仪大概什么价格？

如果仅仅是分析功能的话，Gen5 在 100 万到 200 万，具体要看配置，比如 Lane 数、搭配 Interposer 的数目、内存大小。

798. 有哪些厂家做 Gen5 协议分析仪？

有 LeCory，Viavi，Serialtek，还有 Keysight。

799. PC 接自研 EP，用分析仪解析建链过程中 PC 发来的包全部出错，可能是什么原因？

可能是因为分析仪 Rx 没收对，分析仪的 Interposer 是 Re-driver，把模拟信号一分为二分给 EP 和分析仪主机。分析仪主机为 FPGA，可能需要调整一下分析仪 Rx 的 DC Gain 等参数。

800. 协议分析仪跟设备间是怎么连接的？

一般是 Interposer 夹在主机和 DUT 之间，Interposer 和分析仪主机用数据线连起来。

BIST

801. PCIe PHY BIST 是什么？

根据 Digital、Analog、Near End、Far End 测试层级的不同，PCIe 有多种 Loopback 测试模式。PHY BIST 是 PHY 的 Internal Loopback 测试，Pattern 一般采用 PRBS。

802. Vendor 提供了单独测 PHY BIST 的程序，在 SoC 上怎么用 CR 口测试 PHY BIST？

无非是写一系列 PHY 的寄存器，让 PHY 内的 BIST 电路动起来，具体可以参考 PHY 手册，或者直接找 Vendor 要 Sequence。

如果是 PHY ATE 测试 Pattern 的移植，CR 口跟 JTAG 类似，JTAG 在内部也是转为了 CR。把单独测 PHY BIST 的 Sequence 移植过来就好了。

803. PCIe BIST 电路测试的是 PCS 还是 PMA？

两者均测。

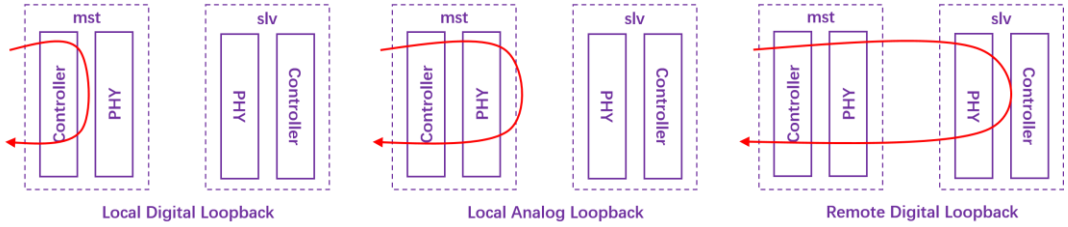
Loopback

804. PCIe 有哪几种 Loopback 模式？

PCIe 有三种 Loopback 模式：

- ✓ Local Digital Loopback (PIPE)，单个 PCIe 仅需 Controller 即可完成，没有 PHY 的参与。
- ✓ Local Analog Loopback，单个 PCIe 即可完成，需要 Controller，过了 PHY。
- ✓ Remote Digital Loopback，需要 Master 和 Slave 两个 PCIe 完成，两端的 Controller 及 PHY 都需参与。

各种模式示意图如图所示。



▲图： PCIe Loopback 示意图

805. 接误码仪进行 RX BER 测试时，Gen1 速率进不到 loopback 模式，但 Gen2、Gen3 可正常进 loopback，可能和什么有关？

不应该呀。

试了一下 Gen2 进 Loopback 后，再测 Gen1 就可以了。

806. PHY 有专门的内外环测试去检验 PHY 是否正常，控制器有什么类似的方法吗？

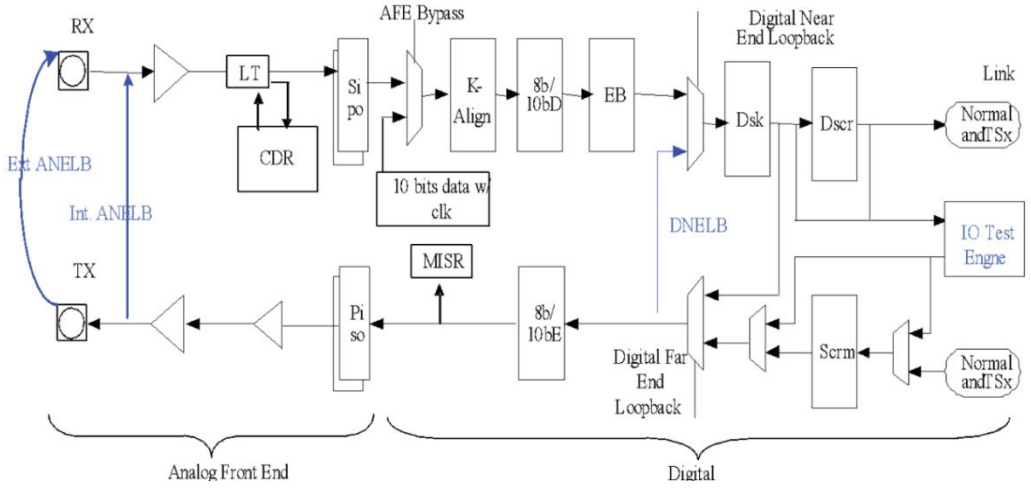
测试 PHY Loopback 时也会跑到控制器，带 PHY Loopback 测试通过也能证明 Controller 没问题，PHY Loopback 测试不过无法证明 Controller 没问题。

如果要脱离 PHY，让 Controller 自证自己没问题，需要 Controller 自己能进行 Loopback。Controller LTSSM 有 Loopback 模式，Intel 的 iomt 就是跑 loopback，另外 rx leq 也是 loopback。此外，有些厂家的 Controller 也有基于 PIPE 口的 Loopback 功能，可以证明控制器协议处理主功能没问题，但是建链功能没法单独验证。LTSSM 本身的建链状态机环回时不工作。厂家有单独的协议用例集，可以掏钱买。

807. 有 Loopback BIST implementation in a typical PCIe Interface 相关博客可以看吗？

看这个图。

Fig. 5. Loopback BIST implementation in a typical PCIe interface.



or instruments. For manufacturing test, DFT has enabled
cost solutions.

Eye-diagram is an accumulative measurement of jitter
the time. Data pulses of the signal stream are a layover o
of each other over a period resulting in an eye-shaped dia
Ideally, the data eye shows no jitter as Figure 6 (a) illustra

808. 用误码仪对 Synopsys 公司 PCIe EP 进行 Loopback 测试，测试 PCIe Rx 时，是只在 PHY 内部 Loopback，还是包含控制器一起做 Loopback？

PCIe EP Rx Loopback 测试需要 EP Controller 参与，即包含 Controller 一起做 Loopback。按了误码仪的 Linkup 按钮后，会先进 L0，然后误码仪会给 EP 发 TS 使 EP 从 L0 退到 Loopback 模式。

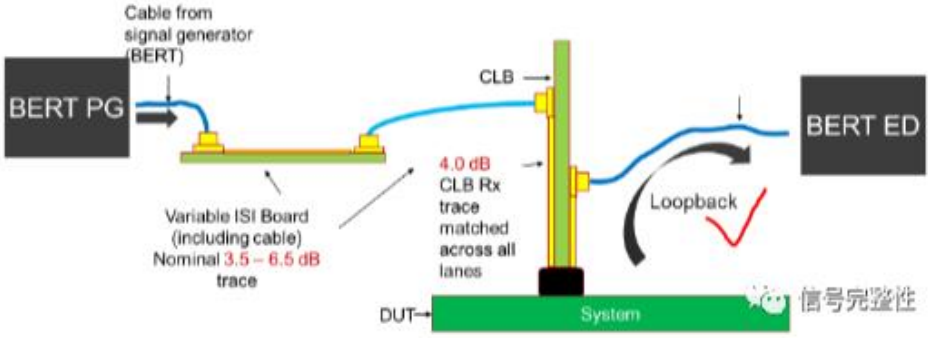
809. PCIe EP Tx 的 Loopback 是怎样测试的？是测 PHY 的吗？

相比于 PCIe EP Rx 的 Loopback 测试，Tx Loopback 测试前链路不会进 L0，而是直接进入 Compliance 状态。Synopsys 公司的 EP 可以发送 BERT 序列，Loopback 回来同进 BER 写入相关寄存器。BERT 在 FT 阶段筛片做的 ATE 测试，除了验证 PHY IP 本身是不是好的，还用来验证 PCB 走线、封装等硬件问题。

810. PCIe RC Rx 的 Loopback 测试是怎样的？

对于 RC，则是只在 PHY 内部做 Loopback。通过写 RC PHY 的寄存器使 PHY 进入 Slave Remote Loopback 的 rx2tx parallel loopback 模式，PHY Rx 收到后立刻转发到 Tx，不需要 Controller 参与 Loopback 但需要 LTSSM 进 Loopback 状态。

811. 下图是系统主板 Rx LEQ 测试组网图，其中的 Loopback 是哪种模式？



这里的 Loopback 不是在 PHY 里环回，需要 MAC 参与，个人理解环路为：BERT Tx -> PHY Rx -> MAC -> PHY Tx -> BERT Rx。

考虑到该 Loopback 是把 DUT 收到的数据环回到并行 Tx Data，然后经过 Tx 发射电路送到 Tx 的差分线上，此间经过了 Tx 的路径，存在信号衰减，BERT 收到的数据误码率可能会会更大。为了剔除 Tx 的影响，从 DUT 的 Tx 到 BERT Rx 路径需要尽可能短，或者加 Repeater。

812. Loopback 模式下 Link Up 状态应该是 1 还是 0？

该模式不需要 Link Up，Link Up 状态为 0，测试数据仍然能够正常发出。

813. 本地 Rx 回环的时候恰好碰到对端 Tx 发送 TS1 怎么办，是有什么仲裁机制吗？

如果仅指单个 PCIe 设备 PHY 的 Loopback，其进入 Loopback 模式后，不再接收对端 Tx 发送的 TS1 (通过 Mux 实现)。

如果是两个 PCIe 设备间的 Loopback，两者进入 Loopback 是经过了协商的（Master 发 Loopback=1 的 TS1，Slave 回相同的 TS1），否则不会开始 BIST 测试。即不存在问题所述情况。

Compliance

814. 只要使能 LTSSM，协议分析仪就可以直接发 TS 进 Compliance 模式吧？

Spec 上，能 Detect 到对端，Polling.Active 超时 24 ms 就自动进 Compliance Mode 了。不过 Compliance Mode 基本都是接示波器用的，不需要协议分析仪。

815. Compliance 只能在 Gen1 时候做吗？

不是。从 Gen1 速率的 Polling.Active 进入 Polling.Compliance 之后，会根据此前 TS 中指示的双方共同支持的最高速率来做 Compliance 测试，不需要进 Recovery 状态切速。

816. 对 EP 卡进行 Compliance 测试时，RC 该如何读写 Compliance 状态的 EP 卡的寄存器对 EP 进行配置？

Compliance Mode 本来就是测试模式，EP 处于 Compliance 状态时还未建链成功，RC 无法从给 EP 发寄存器读写请求，建议在 EP 端自己配。

817. PCIe Compliance Pattern 指的是哪些 Pattern ?

8b/10b、128b/130b、1b/1b 编码时的 Compliance Pattern 不同，具体可参考 PCIe Spec 4.2.9~4.2.11，Compliance Pattern in xxx。

818. Compliance Test 是不是不区分 RC 和 EP?

协议层 Transaction Test 区分 RC、EP。PCIe 4.0/5.0 协议分析厂家现在都只开发针对 EP 的 Compliance test，RC 的协会没有做要求。

819. Serdes 上不应避免连 0 连 1 吗，为何测量 8.0 GT/s 及以上速率的 Tx 均衡参数时可以发送 64 个连 0 连 1 的低频 Pattern？

引文：8 GT/s、16 GT/s、32 GT/s 速率时，PCIe 高频信号的自身特性使得难以测量单 UI 脉冲高度。此时，Tx 电压摆幅及 EQ Preset 可通过 Compliance Pattern 中的低频 Pattern 测得。低频 Pattern 由重复的 64 个连 0 及 64 个连 1 组成，这种 Pattern 下 ISI 大大减小，且信号接近稳定状态，因此能够较为精确地测得各种情况下的电压幅度。（《PCIe 均衡技术介绍（电气物理篇）》）

引文里解释了，这里的连 0 连 1 是为了测试电压摆幅这个静态参数。01 跳变会引入码间干扰 ISI，在没有 ISI 时静态参数测得更准。在正常传输 PCIe 业务时，CDR 长时间没有 Edge 会失锁，因此需要通过编码、加扰的让是来避免连 0 连 1。单纯测量 Tx 参数时，不用担心失锁。

820. 发生连 0 连 1 的 Compliance Pattern 测 Tx Preset 时，会经加扰再发出吗？

不过加扰。测试 Preset 时，在示波器上看到就是 64 个长 1 长 0。

821. Compliance Pattern 是只有在 Polling.Compliance 下才可以发的吗？

不是。也可以自己做，自测试端绕开控制器，兼容低成本测试。

PCI-SIG 认证

822. 如何做 PCI-SIG 认证？

可以付费找第三方实验室进行认证，比如百佳泰。

协议 Transaction 认证一般第三方没有，得找协议分析仪厂家。

九、虚拟化

协议规则

823. PF 和 VF 该怎么理解？

理解成真实物理设备和虚拟设备。虚拟设备就是，不是一个设备、看上去是一个设备、且可以当成一个物理设备来使用的设备。虚拟设备和物理设备共享硬件资源，可以跟物理设备共用，也可以独享。

824. 启用 SRIOV 时，一个 Bus 下可以挂多个 SRIOV 的 EP 吗？

可以。

825. 一个 VF 的 TLP 是怎么经 Switch 路由到这个 EP 的 VF 的？Switch 上的 Mem Range 好像不包含这个 VF Address。

个人理解，Downstream 下来的 TLP 中的地址需要转成 PA ，如果 Switch 支持 SRIOV 特性，这个 VF BAR 的 PA 应该已经包含在 Switch Memory Range 内了。这也意味着 VF BARs 和 PF BARs 的 PA 仍然占用不同的地址空间。具体可以参考 Linux 中 pbus_size_mem这个函数，这个函数计算 Bus 的 Children 时已经考虑了 SRIOV 的情况。

```
/* Put SRIOV requested res to the optional list */
if (realloc_head && i >= PCI_IOV_RESOURCES &&
        i <= PCI_IOV_RESOURCE_END) {
    add_align = max(pci_resource_alignment(dev, r), add_align);
    r->end = r->start - 1;
    add_to_list(realloc_head, dev, r, r_size, 0 /* Don't care */
    children_add_size += r_size;
    continue;
```

826. Scalable IOV 和 SRIOV 是完全不同的东西吗？

是不同的东西。一个是靠 PASID，一个是靠 Requester ID

827. 多 VF 时，RC 可以发 Cfg 请求来访问 VF2 的 Config Space Header 相关寄存器吗？

可以。

828. Multi-Host 是何概念？一个 x16 的物理链路上下挂两个 x8 的 EP 是否为 Multi-Host？

两个 EP 不是 Multi-Host。Multi-Host 需要是两棵 PCIe 树，得接两个 RC。

829. MR-IOV 后面是不用了吗？

是的不用了，MRIOV 无法落地。

应用问题

830. 有场景需要 RP 支持多 PF/VF 吗？

PCIe Base Spec 内似乎没有明确说明 RP 不支持多 Function，即 RP 可以实现多 Function，但没见过这么用的。我的理解是，RP 作为 P2P Bridge，它自己没有 PF/VF；而对于 EP 的 PF/VF，跟 RP 没有关系，也没有所谓的需要 RP 支持的说法。

831. PF 的 SRIOV Capability 寄存器里只有一组 VF BAR 寄存器，那么开启 SRIOV 后如何配置每个 VF 的 BAR 空间？

所有 VF BAR 空间大小是一样的，除第一个 VF 外其他 VF 的 Base 不能独立配置。只需要配第一个 VF 的 Base，后续 VF 的 Base 直接按照固定的 Offset 铺开即可。VF0 的 Base Addr 加上 VF0 的 BAR Size 就是 VF1 的 Base Addr，以此类推。

832. 为了支持 SRIOV，RC 有什么特别的要求吗？比如有没有必须要支持的 Feature？

PCIe Spec 对 RC 没有过多要求，主要是对 RP 和 EP 有要求。

833. Synopsys 公司的 DMA 支持 SRIOV 但不支持 TLP Prefix，PASID 是在 TLP Prefix 里的，这是否意味着不能使用 SMMU Stage1 的翻译了？

如果说的是非虚拟化场景，Stage1 单 CD 地址翻译跟 PASID 无关，没有 PASID 也能做 Stage1 地址翻译。只要 RC 能够区分 Stream ID （SID，通常用 BDF）且使能了 STE 就可以。

在虚拟化场景长，通常使用 PASID 作为 Sub-Stream ID（SSID）在 Stage1 多个 CD 内进行选择，但硬件上 SSID 可以不依赖于 PASID，只要能区分进程并传给 SSID 即刻。

834. 未开启 SRIOV 时，多个虚拟机可以共享同一 PCIe 设备资源吗？

问题详述： SRIOV 是为了让多个虚拟机共享一套硬件资源，如果没有 SRIOV，就是一个普通的 PCIe 设备，难道多个虚拟机就不能访问这个 PCIe 设备资源了吗？SRIOV 的优势到底在哪？

正常来讲不能共享，配置空间就一个，设备内部也没有划分资源，如果允许多虚拟机共享会存在冲突。但如果多虚拟机之间采用时分复用机制来访问该 PCIe 设备资源，那么多虚拟机可以共享该 PCIe 设备资源。相较于时分复用机制，开启 SRIOV 后每个 VF 都有一套独立的配置空间，互相之间资源不冲突，不同虚拟机之间可以同时使用该 PCIe 设备，是一种实现设备 Pass Through 的更高效的标准手段。

835. Intel 还支持 Requester ID 方式的 SRIOV 吗？

应该支持的，SIOV 相当于 Intel 提出了另外一种更好的方式，是在 2020 年新提出的。

836. 协议规定 SRIOV 中 VF 的 Vendor ID 为 16'hffff，软件如何判断 VF 的存在？

VF 的发现过程不是扫出来的，而是由 PF 通过相关 Capability 寄存器中的 Total VFs 告诉系统该 PF 下属多少个 VF。

837. Windows 驱动支持 SRIOV 吗？

支持。可参考：

https://learn.micrOSoft.com/en-us/Windows-hardware/drivers/network/single-root-i-o-virtualization--sr-iov-

838. PCIe SRIOV 能力结构中 VF Offset 和 Stride 是设备 EP 决定的还是 Host 上的 RC 决定？

背景： 我们自己的设备在 x86 服务器上 Stride=4 正常，到龙芯上 Stride 从 4 变成了 256，中间没有更新设备固件；换设备比对，82599 设备的 VF Offset 在 x86 服务器上是 128，龙芯服务器上变成 384 了，VF Stride 正常。

猜测是不是在 82599 中使能 SRIOV 的驱动中，改写了 Offset/Stride 相关寄存器。虽然协议是规定这几个寄存器是 RO 的，但有后门寄存器可以修改这些 RO 寄存器值。

现在现象是 x86 服务器上这个寄存器可以手动配置，龙芯上手动配置不了，不知道是不是因为这个导致什么初始值没配置成功。

一般其它设备不知道这个读改写的寄存器，即只有设备自己能改，其他改不了。猜测可能在 BIOS 阶段把这个寄存器打开啦，然后在 OS 下可以读写了。

可尝试跳过加载这个设备驱动，系统只做扫描，扫描完毕查看这个配置是多少；然后加载驱动，查看配置值有无变化。->驱动加载前后 Stride 值不变，创建 VF 有问题。Stride=256 太大，创建多个 VF 时第二个 PCI 号偏移太大，工作异常。

可尝试比对 82599 在两台服务器上面 4 K 配置空间的配置差异，lspci -xxxx -s BDF。

839. VF 数目最大不是 256 么，为什么有时软件能扫描出上千个 VF？

可以超过 256，超过 256 会跨 Bus 号，Bus Number 的一部分也可以用作 Function num。

840. 理论上 VF Driver 读不到 PF 配置空间，其是通过什么机制在 PF SRIOV Capability 中拿到 VF BAR 信息的？

就是读的 PF 的 SRIOV Capability 中的 BAR，通过软件结构体传递。

841. lspci 结果中，设备 BAR 空间出现 Virtual 标志是怎么回事？

类似这样：

```
[virtual] Memory at fa200000 (32-bit, non-prefetchable) [size=1M]
[virtual] Memory at fa100000 (32-bit, non-prefetchable) [size=1M]
[virtual] Memory at fa300000 (32-bit, non-prefetchable) [size=8K]
```

PCIe 虚拟 BAR 空间或机制可能指的是在虚拟化环境中实现的一种 PCIe 设备地址空间管理技术。它旨在通过虚拟化技术来优化 PCIe 设备的资源分配和性能表现，并提高系统的整体效率和安全性。

在虚拟化环境中，多个虚拟机（VM）可能共享同一物理 PCIe 设备。为了实现资源的有效隔离和共享，系统可能会采用虚拟化技术来创建虚拟 BAR 空间。这些虚拟 BAR 空间允许每个虚拟机拥有独立的地址空间视图，从而确保设备访问的隔离性和安全性。

842. 手动使能 SRIOV 功能后用 telscan 软件重新扫描设备，仍然只能扫到 PF 扫不到 VF，可能是什么问题？

一方面，需要确认 SRIOV 配置是否符合预期，比如 SRIOV 功能是否的确被使能了，支持多少 VF、NumVFs 寄存器显示实际可见的 VF 数量为多少等等。

另一方面，需要确认驱动是否支持 SRIOV、是否会对 VF 进行扫描。

此外，可以根据软件扫描 Log 进行分析，确认是否 Host 资源不够导致未给 VF 分配资源。

843. Synopsys 公司的 PCIe Device 最多支持 4 K 个 Function，这是如何做到的？开了 ARI 不也才最大 256 个 Function 吗？

通过 SRIOV 的 Virtual Function 来实现。在 SRIOV Capability 中，TotalVFs、InitialVFs、NumVFs 寄存器做得很宽（16b），SRIOV 一个 Device 可以占用多个 Bus，这样就没有 256 个数限制了。如果 VF 超过 256，下个 VF 就是 Bus 号加一了。VF 的 BDF 是通过 First Offset 和 Stride 算出来的，如果算出来的 DF 大于 256 了，Bus Num 就加一。PF 数目是定死的，VF Bus 加 1 的时候，不需要多加一个 PF。

该 Feature 无需 Switch 特别支持，也无需在 Device 内虚拟一个 Switch，只需 Host 在 SRIOV 分配 VF 的 BDF 时遵循这个逻辑就行了。

VFs Spanning Multiple Bus Numbers

As an example, consider an SR-IOV Device that supports a single PF. Initially, only PF 0 is visible. Software Sets ARI Capable Hierarchy. From the SR-IOV Capability it determines: InitialVFs is 600, First VF Offset is 1 and VF Stride is 1.

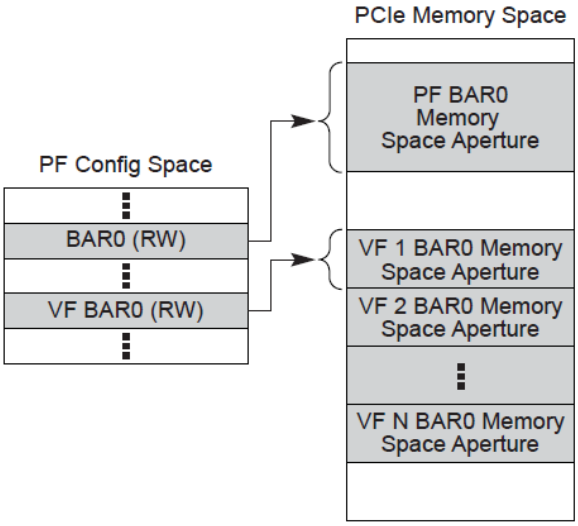
- If software sets NumVFs in the range [0 .. 255], then the Device uses a single Bus Number.
- If software sets NumVFs in the range [256 .. 511], then the Device uses two Bus Numbers.
- If software sets NumVFs in the range [512 .. 600], then the Device uses three Bus Numbers.

PF 0 and VF 0,1 through VF 0,255 are always on the first (captured) Bus Number. VF 0,256 through VF 0,511 are always on the second Bus Number (captured Bus Number plus 1). VF 0,512 through VF 0,600 are always on the third Bus Number (captured Bus Number plus 2).

844. 为什么读不到 VF 的 BAR Size，也配不进去？

可能是误操作了 VF 配置空间中的 BAR 寄存器。

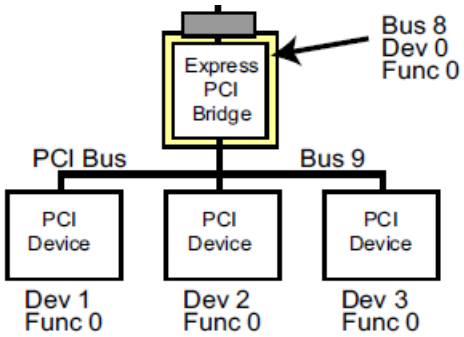
VF 配置空间中的 BAR 寄存器属性为只读且读数据全 0，即无法对 VF 配置空间中 BAR 全 1 再读回的方式获取 VF BAR Size，也无法通过写 VF 配置空间的 BAR 来配置 BAR Base。VF BAR 的 Base 和 Size 都是在其所属的 PF 的 SRIOV Capability 中分配及配置的。



845. 如果 EP 需要支持 SRIOV，一般需要预留几个 VF？

看需求了，多少的都有。NVMe 预留了几个 VF，网卡上千 VF，DPU 的 VF 更多。

846. 下图中每个 Device 中只有一个 Function。一个 Device 可以有多个 PF 吗？是不是只有 Dual Port 才支持多个 PF？



不是。一个 Bus 下只有一个 Device，一个 Device 下可以有多个 Function，跟是否 Dual Port 无关。

847. 市场上真正实现了 MRIOV 的产品有哪些？

没听说有支持 MRIOV 的成熟产品。Broadcom 和 Microsemi 的交换机貌似都不支持 MRIOV，他们实现的好像是自定义的。

十、地址转换

ATS

848. 如何在 RC 上验证自家 EP 芯片上的 ATS 功能？

问题补充说明：我是 Linux Kernal PCIe 驱动，主机为 Ubuntu 环境，装有支持 ATS 的 Intel 集成显卡。已知 ATS 由 EP 发起，IOMMU/SMMU 回复；或者 IOMMU/SMMU 发起，EP 回复。

如果带有支持 ATS 的 RDMA 网卡，可构造 RDMA 测试场景来发出 ATS 的包。

可尝试用协议分析仪测试。协议分析仪无直接测试 Case，但是可以查看相关 Register 来判断是否具备或者开启 ATS 能力，也可以把交互过程可以抓下来进行分析。

849. PCIe 4.0 的 IP 建链之后工作在 Gen1 速率，还能用 ATS 等较高版本 IP 才支持的功能吗？

可以。ATS 等跟速率无关的 Capability，只要链路两端的 IP 均支持 ATS，不论链路实际工作在 Gen 几速率下都仍然能使用相关 Capability。

850. lspci 能查看设备的 ATS 能力吗？为什么在 ContextX-7 上看不到 ATS Capability？

一般能看到，比如 Intel 的就能看到。Nv 的网卡有几个特性没做到 Capability 里，可排查一下。

851. lspci 显示 Intel 网卡支持 ATS 但 ATS 未打开，该如何打开？

Intel 的不清楚。MLX 的可以通过 mxlconfig 来配置，命令如下：

mlxconfig -d <dev> set ATS_ENABLED=1

852. RC 和 EP 是不是都可以用 ATS？比如 Dual Mode PCIe IP，作为 EP 时支持 ATS，在作为 RC 时还能用 ATS 吗？

理论上 RC 不应该作为 Requester 发起 ATS 请求。大致有以下原因：

- ✓ ATS 需要 RC、EP 均支持 ATS Capability，不但要求 EP 能发出 ATS 请求，RC 请求接收方也需要有 IOMMU/SMMU 等 TA ATPT 做地址转换。虽然 IP 是 Dual Mode 的，但其内部不一定有 TA。
- ✓ 即便 EP 设备实现了 TA、能够接收 ATS 请求，其也难以保证转换后地址落在 RC 分配的 BAR 地址范围内。

- ✓ ATS 是做 VA->PA 的转换，作为 RC，需要由 EP 做这个转换吗？即便需要转换且 EP BAR 全开，这个 PA 是到 RC 还是 EP？

853. ATS 转换请求中的 Length 可以设置为转换请求长度的两倍，这句话怎么解读？

这里的转换请求长度是指需要几笔地址转换，一笔地址转换需要 2 DW 的 Data，对于 Length=2，N 笔就需要 Length=N*2。

854. 设备本地 ATC 用 RAM 来存储被转换后的地址，这样一来当发起请求进行地址匹配时，花费很多个 cycle 去遍历 RAM 里的地址，效率较低，有什么好的解决方案吗?

ATC 是 Cache，何来很多个 Cycle 之说？Cache 不是一个 Cycle 一个 Entry 去比的，要是用这个方法去看 cache hit/miss，那这个 cache 不要也罢。遍历肯定不会去遍历的，那就不叫 Cache 了。

支持 ATS 的设备，肯定得实现某种 Cache，即便不是 SRAM 而是 DDR RAM，那也会使用快速定位方法比如 Index，组关键等等当时，来加速查找。

Gen6 里 AMA (ATS Memory 属性) 也可以加速查找的过程。

855. 目前哪些产品支持 ATS？有支持 ATS 的显卡网卡吗？比如 Nvidia 显卡和 MLNX 网卡？

e810 支持 ATS。Intel 和 NV 都有支持 ATS 的显卡。

856. PCIe 的 ATS 地址转换是 IP 自己主动去发然后存到 ATC 里面吗？

应该是设备主动去发送的地址转换请求，然后把返回的地址存到自己的 ATC 中。不然就只能发出未转换的地址。

857. ARM 架构有 ATS 吗？

有。

858. 市面上有什么设备支持 ATS 并且内部带 ATC 的吗？

有，NV 的光纤网卡。

859. PCIe Controller 回 ATS Completion 时，S bit 是 IOMMU 还是 Controller 置起的？

问题补充：U bit 需要配置一下 IOMMU，S bit 需要配置一下 IOMMU 吗？还是说不用配置、直接发一个 Page Size 请求大于 4 KB 的 ATS Req 就行？

当然是 IOMMU 返回是自己置起的，置起表示 8 KB Size 及以上的 Page。ATS Req 内不能指定 Page Size。

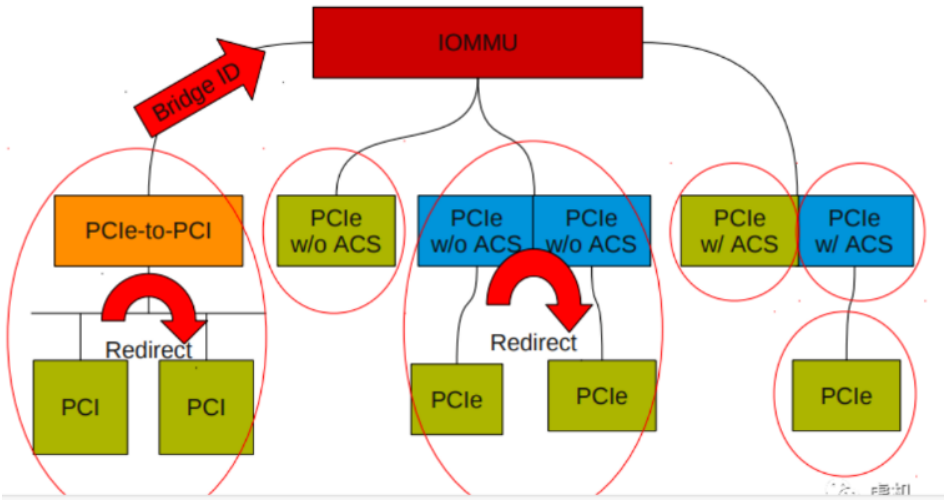
ACS

860. PCIe ACS 这个功能重要吗？

重要。ACS 能解决 P2P 访问中的安全问题。

如果 Switch 下接了两个支持 ATS 的 EP，那就支持绕过 RC 的 P2P 传输了。如果不开 ACS，EP 可以谎称自己发出的为转换后的地址，绕过 RC 去恶意访问甚至破坏另一个 EP。开了 ACS 后，所有 Switch 内的 P2P 转发都会绕道 RC（IOMMU），由 RC 来保证访问的安全性。也就是说，ACS 是个 P2P 开关，防止被偷读数据、防攻击的。

IOMMU 依据其下属节点的 ACS 属性为依据来分组（如下图），一般开了 IOMMU 就需要开 ACS。开 IOMMU 时，Host 设下去的都是 IOVA，要过 IOMMU 才是物理地址，不开 ACS 就可能会有误访问，拿不到想要的值。（参考：解惑 IOMMU）



不过，开了 ACS 后会影响 P2P 效率，如果 P2P 用得多，为防止效率低，可以关掉 ACS。

861. 如何配置如下 ACS？

```
Capabilities: [1cc v1] Access Control Services
        ACSCap: SrcValid+ TransBlk+ ReqRedir+ CmpltRedir+ UpstreamFwd+ EgressCtrl+ DirectTrans+
        ACSCtl: SrcValid- TransBlk- ReqRedir- CmpltRedir- UpstreamFwd- EgressCtrl- DirectTrans-
Capabilities: [1f4 v1] #25
```

用 setpci，在 BIOS 里配。

追问：BIOS 里配 PCIe 总线的吧，能否配 Switch 上面的？现 BIOS 里 ACS 已开，桥已开。

无回复。

862. 技嘉服务器 BIOS 内已开启桥和 ACS，但 lspci 查询结果显示 Switch 未开启 ACS，该如何配置 Switch 上的 ACS？

试试 setpci。

863. 有哪些支持 ATS、ACS 的设备？

比如 Intel 的电脑集成显卡、AMD 200 G/400 G 光纤网卡、某些 GPU。还有一些支持虚拟化的网卡也会支持 ATS，比如 Nvidia 的 ConnectX-5/7 通过 ATS 来加速地址翻译。

864. 同一个 Switch 下多卡之间进行 P2P 访问时必须开启 ACS 吗？

没开 IOMMU 的情况下可以不用开 ACS。

865. ACS 是如何判定某个请求是否为 P2P 请求？

Memory 请求地址落在自己 Memory Base 和 Limit 范围内就可以认为是 P2P 请求。

PRS

866. 有了 ATS 为什么还需要 PRS，碰到 Miss 后直接等到页表分配好再返回请求不行吗？

ATS 和 PRS 是两个独立的 Feature，ATS 为请求 TA 反馈页表里的 PA 地址，PRS 为请求 CPU 分配一个新的页表。理论上没有 ATS 也可以发 PRS，没有 ATS 也能发 PRS，但实际使用中可以只开 ATS 不开 PRS，开了 PRS 要开 ATS。

如问题所述，如果把 ATS 和 PRS 功能做到一块，即 EP 发出 ATS 在 RC 端 Miss 了，RC 端自动申请新页。分配成功了还好，RC 反馈 ATS Hit 的 Response；如若 IOMMU 不肯分配新的页，还是得反馈 ATS Miss。此间申请新页的时间浪费较大，尤其是大量 ATS Miss 时每个都要申请新页然后被拒，时间开销更多，倒不如在 ATS Miss 后直接反馈一个 Miss Response，EP 直接发送 Untranslated Addr 到 RC，交给 RC 端 MMU 来翻译地址。

PASID

867. PCIe 协议的 PASID 是来干什么的？

在 SMMU、虚拟化地址翻译时要用。ARM 的 SMMU 类似于 x86 的 IOMMU，PASID 作为 SMMU 的 SSID，是可选的，可以拿来做 Stage1 的 Translation。

PASID 在地址翻译时非必需。一般来讲，Host 翻译不需要 PASID，Host 一般不带这个字段，但开了 TA 后也能用，开了 TA 后要求发的地址是虚拟地址。对 Device 而言，其地址翻译需要 PASID，但如果 EP 禁掉 PASID，不给 PASID 时也能正常翻译。

868. ATS 里的 PASID 怎么用？

ATS 和 PASID 没有直接联系。如果是做虚拟化，ATS 有点用，可以找 Driver 的人问下怎么用；如果不做虚拟化，ATS 不用也罢。PASID 可以用于 SMMU 查表，但非必须，仅靠 BDF 也能查表。两者结合的情况，即 ATS 里的 PASID 很少用到，知道的只有 Intel 实现了 PASID，ARM 也支持 PASID 但 EP 很实现 PASID。如果一定要用，可以查看 SMMU 相关资料查看 PASID 用法。

869. PASID 和 Bus Device Function 编码有什么关系？正常 BDF 不是已经可以定位到某个具体的 Function 了吗？

PASID 多用于 Virtualization 场景，是 IOMMU 做二级页表翻译时用，一般用不到。

不用 PASID 时，同一 BDF 只能一个进程用；开启 PASID 后，多进程可采用不同 PASID 共享同一 BDF 设备。

870. 现在有什么能测 PASID 的 Function 吗？

网卡。

871. IOMMU 方案中，有只用 BDF 的，有在 BDF 基础上启用 PASID 的，是因为只用 BDF 会导致某些情况下地址映射出问题、需要用 PASID 来弥补缺陷吗？还是说只用 BDF 没问题、PASID 从另一个角度完善了地址映射功能？

只用 BDF 做映射没问题。

MMU 中根据 Stream ID（SID）来选择 Stream Table Entry（STE），同一 STE 内可以根据 Sub-Stream ID（SSID）选择不同的 Context Descriptor（CD）。这里的 SID 即可对应 PCIe 的 BDF，SSID 可对应 PCIe PASID。如果只用 BDF，则只支持一个 CD。

PASID 多用于虚拟化场景中选择不同的 Stage1 转换表。当然，不支持 PASID 时候也能通过其他方式实现 SSID 选 Stage1 转换表的功能，但在 PCIe 中一般是这么干的，即把 PASID 接到 SSID 上。

参考阅读：

- ✓ SMMU 架构手册之数据结构和转换流程（2）
- ✓ MMU-700

- ✓ 多个设备是否可以共用一个 SMMU StreamID？
- ✓ IOMMU 和 Arm SMMU 介绍

872. EP 发 TLP 时其中的 PASID 是谁给的？

PASID 是用户自己提供的，需要配合 SMMU 配置使用。一般通过进程号来给的，MM->PASID，然后 SMMU 使用该 PASID 号建立一个指定 CD 表，最后设备驱动将该 PASID 发送到设备上去。具体给到 EP 设备上面，好像是 EP 厂商来实现的方式，在有些 GPU 里面是通过命令队列的方式发过去的。更多信息可查看 Intel 的 SRIOV 文档，其中有详细介绍该机制。

873. PASID 有何用途？

PASSID 在虚拟化、云上还是很有用的，结合 VMID 和 ASID 可以做进程和虚机隔离。

874. EP 的 DMA 流量里 TLP 需要带上 PASID 吗？

问题补充： Synopsys PCIe IP 的 HDMA 不支持 PASID，是不是实际业务中并不需要？DMA 不带上的话，怎么知道是哪个 Process 的流量呢？

PASID 是给 IOMMU/SMMU 使用的，有没有都行。如果支持 PASID，走 Stage1 的 CD 2 Level 再查找 2 级；如果不支持 PASID，SMMU 走 Stage2 翻译即可。

875. DMA 流量里有无 PASID 在效率上有区别吗？

一般不带 PASID 时走 Stage2 会快一点，毕竟不需要 CD 了。

876. 不开虚拟化时，EP DMA 发的是 PA 吗？

个人理解，不开虚拟化的情况下也要开 IOMMU，EP DMA 发送的仍然是 VA。

PA 地址空间可能是非常离散的，如果 DMA 直接用 PA，一方面 DMA 必须要支持 Scatter Gather 才能效率较高，另一方面 DMA 发出的 TLP 中 AT 就得设置为 1。

个人实际使用 DMA 过程中，没有用 PA，直接 Host 配置的 VA，AT 还是 0，由驱动把 VA 转成 PA。比如 Driver 配置 EP 内部的 DMA，配置的地址仍然是 Host DDR 对应的 VA。

理解下来，每家采用的方法能够自洽即可，只是哪种方法效率更好。

877. EP 如何知道自己的 PASID 值是多少？软件是如何通知 EP PASID 值的？

PASID 是 EP 的本地 ID，由 EP 端的系统软件进行分配，一般是把进程号用作 PASID。理论上 EP 每个 Function 分到的 PASID 都应该是正交的，但实际上大家似乎没有遵守这个规则。至于软件是如何通知 EP PASID 值的，PCIe Spec 没有明确指出，应该是系统软件写 PCIe Wrapper 中的相关寄存器吧，厂商自行实现。

MMU

878. ATS、ACS、IOMMU 是什么关系？

RC 端 ACS、ATS 的物理实现叫 TA，就是 IOMMU。

ATS 是提前翻译地址，只有 ATS 时存在 EP 绕过 IOMMU 恶意 P2P 访问的风险。

ACS 是访问权限控制，可以把从 EP 上来的地址翻译之后的请求全部重定向到 RC，让请求处于 IOMMU 监管之下。

链接：PCIe ACS，PCIe 访问控制服务（ACS）

879. IOMMU 是什么，跟 MMU 有何区别？

IOMMU 是用在设备端，可用以翻译设备发出的 IOVA 地址，防止设备直接操作物理地址引起访问错误。当然 IOMMU 作用不止地址翻译。

MMU 位于 CPU 内部。

880. 用 DMA 做 EP-EP 间 P2P 通信需要考虑 IOMMU 吗？

正规的都要有 IOMMU，具体看 DMA 使用场景、系统是否启用 IOMMU。

881. 若系统中没有虚拟机，IOMMU 在地址翻译中是否也发挥了作用？只是因为虚拟地址和物理地址相同，看起来想没翻译？

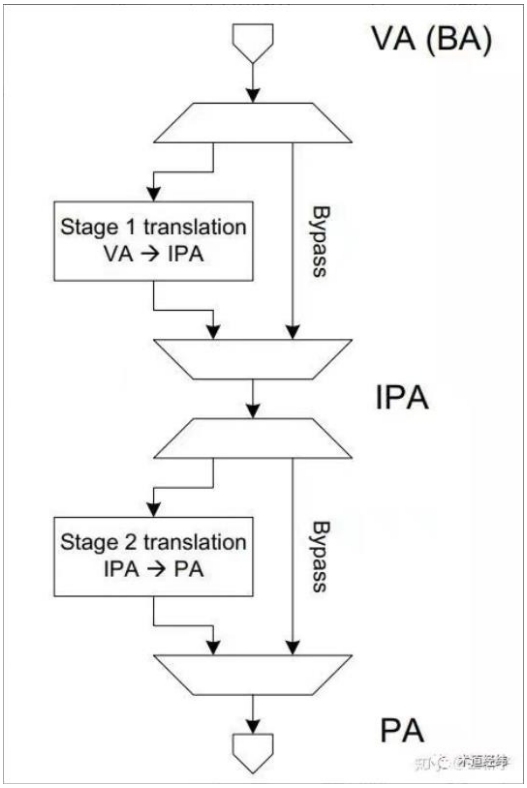
不是。IOMMU 并非一直开启，默认是关闭的，有需要时才开。IOMMU 开启了就发挥作用、就会有地址转换，没开就不会转换。

882. 在服务器上的测试 IOMMU 时，EP 开启 ACS 后系统的 IOMMU 才生效，没有报错，可能是什么原因？

正常。用 IOMMU 必须得开 ACS，否则地址可能会碰撞，地址路由该走错了。

883. VA->PA 地址转换过程中，什么场景下不能跳过 Stage1、必须先做 GVA->GPA 呢？

问题详述：下图中，Host Translation 只做 Stage2 的 GPA->SPA，Guest Translation 则 Stage1 和 Stage2 都做完成 GVA->GPA->SPA。那么，什么场景下不能 Bypass Stage1、必须先做 GVA->GPA 呢？



虚拟化场景中必须走 Stage1。

在有 SMMU 的情况下 Stage1 的 VA 到 IPA 很普遍，用于进程地址隔离。

884. PCI EP Framework 中的 EPC Addr Space 是指什么？

EPC Addr Space 是用来访问 RC 内存的，像是通过 IOMMU 将 PCI Device 域地址映射为 Host Physical Address (HPA)。

885. 包括查找表在内的 IOMMU 功能实现是单独一个芯片还是集成在主机芯片内呢？

是一个片内的功能模块，AMD 叫 IOMMU，ARM 叫 SMMU，Intel 叫 VT-D，应该就在 RC 旁边。

886. 开启 SSMU 时，EP 发出的地址是先比较 Base/Limit 在哪个 RP 下，还是先送到 SMMU 做转换？

问题补充：下文讲默认是先进 SMMU 做转换再比较，不应该先比较 Base/Limit 吗？

引文：ATS（Address Translation Services）是一种基于信任的服务协议。如果 EP 端 ATC（Address Translation Cache）声称其发出的访问请求是经过转换后的地址，且该地址刚好落在 PCIe 交换开关的 BAR 范围内，则该访问请求不会到达 RC，而是被交换开关路由到该地址所对应的 EP。也就是说，该访问请求绕过了 IOMMU 的隔离，进行了 P2P（peer-to-peer）传输。

上文为 P2P 场景，EP 发出的地址在 EP 内已经转换好了，并不需要送到 SMMU。

887. PCIe + SMMU 方案什么样的 SoC 用的多？

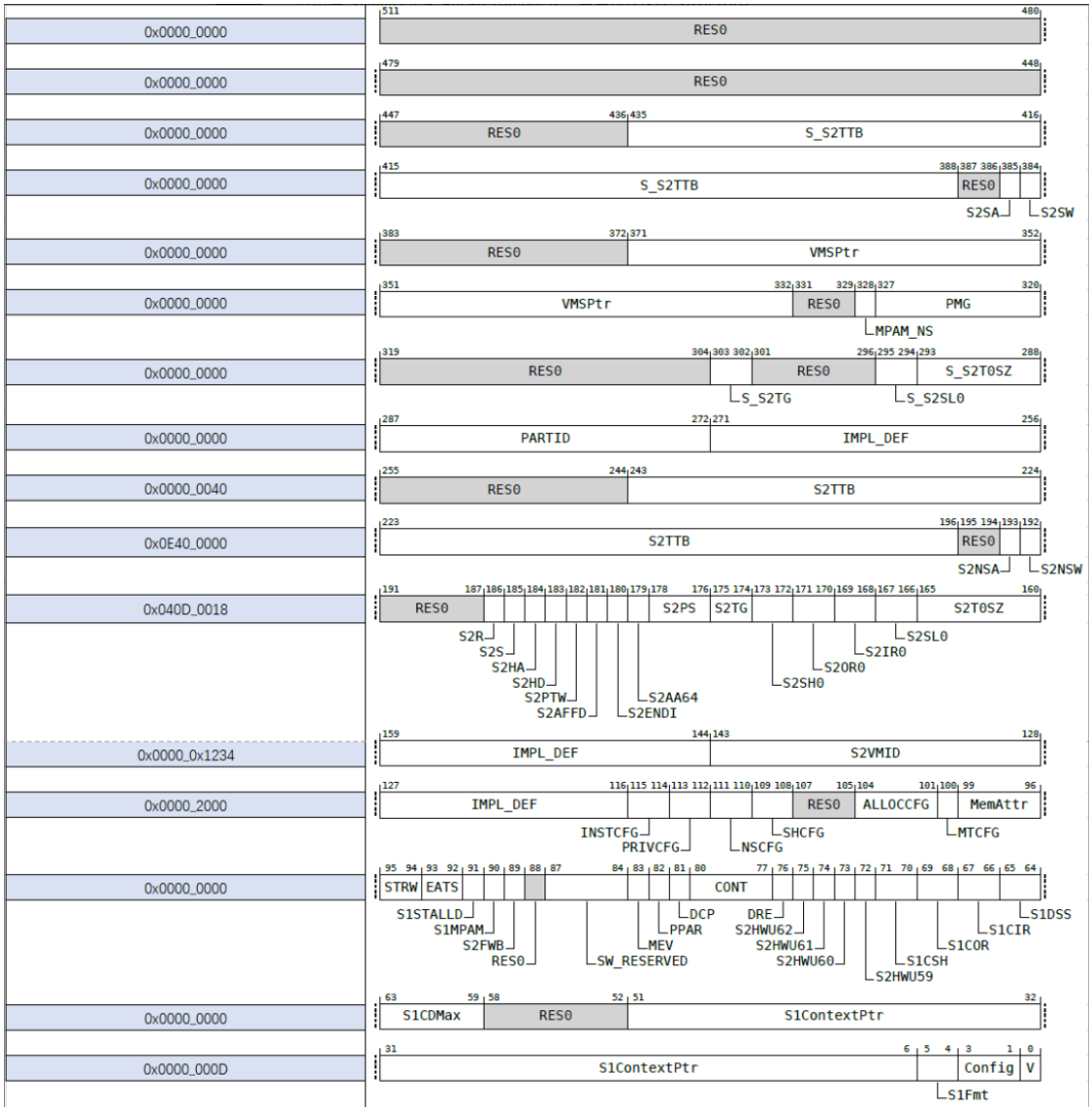
手机，平板，PC，服务器。

888. SMMU 读 STE 报 C_BAD_STE Error，可能是什么原因？

详述 1：SMMU 读到下图 STE 时报 C_BAD_STE Error，预期使用 none-security mode 的 STE，stage1 bypass，stage2 translate。OAS = 48 bits，grannule = 4 KB。S2AA64 = 1。是否该 STE Structure 配置错了？

详述 2：MMU600 已经通过 TCU 的 QTW interface 读到了 STE Structure，然后就在 EVENT Q 上报了 C_BAD_STE event。

详述 3：我配置的 S2TG = 0'b00 ：4 KB； S2T0SZ = 0x18 ：（64- IAS = 24）； S2SL0 = 0 （因为是 48 bits 的，所以我设置是 0，预期它从 Level0 Page Table 开始查找）。



原因很多，需要根据 log 具体分析。可以肯定的是你至少读到了 STE 了，你往 STE 的 S2TG S2TG S2T0SZ S2SL 这些 field 上查查。

ATU

889. EP 是如何获知 RP 侧的地址的？

PCIe 域地址是 Host 配置过去的。需要驱动读出来告诉 EP，底层应该有某个寄存器存放这个地址。

如果是 RP 后面整个地址范围，EP 不需要知道整个 RP 地址范围，RP 照单全收。考虑到 EP 访问整个 RP 地址空间存在安全问题，RC 端往外会配置 SMMU 来阻挡某些地址的访问。

890. ATU 全称是什么?这个是哪家公司定义的名称？和 MMU 有何区别？

ATU 全称 Address Translation Unit，Synopsys PCIe IP 自己做的，和 MMU 没有直接关系。

891. 对于 Synopsys 公司的 IP，在不开启 iATU 的情况下，访问 ECAM 空间或者 MMIO 空间，Controller 内部是怎么组包的呢？

ECAM 控制器里有个窗口寄存器配置一下，落入这个窗口的 AXI 提取 BDF 转成 Cfg 0/1；MMIO 就是下游 BAR 空间访问。如果说的是 io BAR 访问，那就需要 ATU 出马了，或者 SoC 自己定制。

892. 发送不同包的时候 TLP Header 的有些字段可能会有变化，如果不使能 iATU，是否有某种修改机制？还是说硬件采用默认值？

可通过 misc 信号覆盖。

893. EP 要访问 RP 存储器域，EP 如何知道 RP 存储器域的地址对应 RP PCIe 域什么地址？

这是驱动的事，由 Host 侧的驱动高速 EP 其能访问哪些地址，EP 直接访问即可。RP 内存在映射关系，由 BIOS 做映射，RP 自己做转换，EP 无需知道对应 RP 的真实地址。

PCIe Device 能访问的资源是由 IOMMU 决定的。没有 IOMMU 的平台只要使能了 Master Device Bit，所有 Address Map 上的资源都能访问。

894. Inbound ATU 有两种模式：BAR Match Mode 和 Address Match Mode。在设备测，这两种模式都可以配置多条映射吗？还是 BAR Match 模式，一个 BAR 只能映射一项？

可配置多条映射。若某 Inbound 请求落在多条映射中，按照匹配上的第一条映射进行转换。

895. SNPS PCIe Controller 作 EP 时，Outbound ATU 有何应用场景？

比如用 CPU 给 RC 写数据，比如发 MCTP 包。

896. iATU 开启时，Mem 包是先做 BAR Mapping 还是先做 iATU 转换？

先做 BAR Mapping 再进 ATU 做转换，如果 BAR Mapping 不上就不做转换直接丢掉或报错了。

附、专题

一致性

897. Prefetchable 和 Non Prefetchable 有何区别？

在一些应用中我们会遇到“可预取”内存的概念，可预取性是指存储器空间的可预取能力。如果读操作没有副作用，即如同从 RAM 中读数据一样不会破坏数据，则称存储器空间可预取。一般情况下，常规存储器是预提取的，而存储器映射 I/O 或其他性能差的存储器区是非预取的。在 PCIe BAR 中，有 1 bit 来指示该 BAR 范围是可预取或不可预取。

898. RC 在读写 EP BAR 空间时，TLP 中 Attr[0] Snoopable 属性一般是怎么做的？

BAR 内一般是 Non-Cacheable，MWr/MRd TLP 中的采用 No Snoop。

899. 有文章说 PCIe DMA 时建议将 No Snoop 属性设置为 1，即不做 Cache Snoop，而是由软件来维护 Cache 一致性。为什么？

当一个 PCIe 设备对存储器进行 DMA 读操作时，如果传送的数据非常大，比如 512 MB，Cache 的一致性操作不但不会提高 DMA 写效率，反而会降低。因为这个 DMA 读访问的数据在绝大多数情况下，并不会被 Cache 命中，但是 FSB 依然需要使用 Snoop Phase 进行总线监听。而处理器在进行 Cache 一致性操作时仍然需要占用一定的时钟周期，即在 Snoop Phase 中占用的周期，Snoop Phase 是 FSB 总线事务的一个阶段。

900. 有实际场景需要把 PCIe 的 No Snoop 属性配置为 0 吗？

有，高性能网卡需要把 No Snoop 配置为 0。

对于一些需要立马处理的数据，读进来的要数据立马处理，不能被换出，应该使用 Snoop。如果 No Snoop 了，还得从 DDR 读出来，影响效率。目前 Cache 足够大，几十 M 了，所以网卡比较适合这种操作，读进来数据立马放到 cache 里面交给上层处理。存储就不行，存储的数据基本都是读进来很久以后采用。

901. No Snoop 属性是谁设置的？是硬件默认还是驱动设置？EP 的驱动可以设置 RC 的吗？

BAR 里的 No Snoop 属性是 EP 设置的，RC 的 RC 自己设置。RC、EP 的 No Snoop 属性不一致时听 RC 的，EP 只提供建议。

902. 可以直接 setpci 设置 No Snoop 属性吗？

不能通过 setpci 设置 No Snoop 属性。

903. 该如何理解 PCIe No Snoop 的优势？

补充：下附 No Snoop 相关描述，是否可以理解为 No Snoop 时只有设备自己会去读写数据，不需要 CPU 保证 Cache 一致性，同时数据量比较大 Cache 存不下用不到 Cache 的速度优势?

The PCI express protocol includes a "no snoop required" attribute in the transaction descriptor.

For a PCIe non-snooped read, the request can go directly to the DRAM controller to obtain the data. The processor caches do not need to be snooped, and the PCIe device does not need to wait for a snoop response before using the data. This can reduce the latency for obtaining the data, which can increase the sustained read bandwidth in the common case that the hardware supports a limited number of concurrent read transactions.

For a PCIe non-snooped store, the request can go directly to the DRAM controller to store the data. The processor caches do not need to be snooped to invalidate any copies of that cache line. This reduces the amount of time that the buffer handling the store is occupied, so that a fixed number of buffers can deliver higher throughput.

These "no snoop required" transactions are typically "safe" for accesses to address ranges for which processor cacheing is prohibited.

An example use case is a GPU that needs to "borrow" extra memory from the processor(s) for "spill" and "restore" traffic. Only the GPU will be accessing that memory, so it does not need to look in the processor caches to see if any of them has modified copies of the cache lines. The improvement in bandwidth due to the elimination of snooping can improve graphics frame rates.

The PCIe read/write bandwidth improvement is typically the primary reason to implement the "no snoop required" functionality.

Secondary benefits include reduction in snooping traffic on the processor caches, reducing coherence traffic on the chip-to-chip links in multi-chip systems, and reducing overall power consumption.

Since the original message mentioned PCIe-related CBo events, it seems reasonable to assume that the counters are referring to PCIe transactions with the "no snoop required" bit set.

Snoop 会过一下 HNP 的 Snoop Filter，可能会从 SLC 取回，至于会不会刷 MESI，PCIe 的 Read Snoop 转 CHI 的只有 Read-Once，都不会刷和改变。No Snoop 一定会对 DDR 有动作。

904. 什么时候会用 No Snoop，什么时候不用？

看具体访问的 Memory 属性，DMA 一般用 No Snoop 的居多

905. DMA 为什么多用 No Snoop？是不是意味着内核 DMA 部分会提前 Flush Cache、Pin Page，所以才不需要 Snoop？

因果关系反了，是因为想用 No Snoop，所以才去 Invalid 的 Cache。这里的 No Snoop 意愿无需给内核，无需预先静态设定固件/驱动，是全硬件行为。

用 DMA 时，PCIe 一个请求包如果比较大，需要在 MTU 内进行分包，到 CMN（ARM CMN）还要再次分包。如果开启了 Snoop，去查 Filter 然后从 Cacheline 取数据，性能反而不如 No Snoop。

906. EP 使用 No Snoop 时，EP 端的每个请求都必须带 No Snoop 标签么？

具体看怎么用的，仿真的话 Synopsys VIP 默认不开 Snoop 的，要 Cfg 一个配置，然后报文中约束才会变成 No Snoop，否则就是 READ_ONCE。

907. Snoop 默认开的还是关的？

不一。最近看的 Intel 的服务器 Bridge 默认关掉 No Snoop 的，也不可配置。AMD 的 Bridge 默认开的 No Snoop，可以配置。

908. PCIe 中什么场景需要用到 Snoop？

笼统来讲，需要 Coherence 的场景需要使用 Snoop。

909. 使用 DMA 时是是否都要设置 No Snoop = 1？是使用 No Snoop 好还是 Snoop 好？

答案非绝对，取决于业务。像 ARM AMBA 里定义了多种 IO 一致性请求，比如 stash，io deallocat，具体怎么用都给系统设计者决定。

实测抓取 RX500 显卡数据时，读 4 G 以上空间时带的是 No Snoop =0, 读 4 G 以下空间时带的是 No Snoop=1。即对该网卡而言，描述符是 Snoop 设计，数据是 No Snoop 设计。

实际设计时可参考下述两种方案：

- ✓ 如果是 DMA 搬大块数据，No Snoop 好，此时 Snoop 对 DMA 操作意义不大。
- ✓ 如果是搬描述符，是 Snoop 好。比如一个核给 EP 设备下描述符，通过 Snoop，EP 就可以直接从缓存里获取不用刷缓存。处理完成的数据，也可以直接送到 Cache 里、甚至是某一个核的 Cache。

有观点称，站在 RC 角度，Inbound 方向的是 IO Coherence 的，但一般不会看这个 bit，都是系统自己决定的。比如通过叶表页面属性 SMMU 转出来，或者是 TLP 带的其他一些 Hint，TPH 这种。例如 EP 发到 RC 的 MSI，MSI Addr 这个地址在 Host 侧可能是 Non-cacheable 的，直接发到中断控制器；如果是数据或者拉描述符，那就可能是 Cacheable 的，从而产生 IO Cache Coherence 的请求。

也有不同声音表明，从 RC 角度，PCIe 收到来自 EP 的 Snoop=1 很常见。Snoop=1 时：

- ✓ 读请求会转换成 Read-Once 发上去给 CMN；PCIe 因为是利用 Read Once 的，所以只做 Resolve 不会写进 Cache 也不会改变 Cache 状态，所以 Read 的 Snoop=1 不需要维护 MESI 的一致性。
- ✓ 对于写请求，写请求会先转成 Write-Full，如果发现 AXID 相同，再降级为 Write-Partial。

Snoop=1 另外一个用途为 RC 做自研 Prefetch 加速，利用 CHI 的 Stash 报文来做 SLC 加速。

910. Snoop=1 时写 Partial Line 和写 Full Line 跟 AXI ID 有什么关系呢？

对于 PCIe 而言，采用相同 AWID 进行强排序的情况是比较多见的。对于 Cacheable Write 来说，理论上 Write Full 的包完全可以由 Write Partial 来代替，但由于 QOS 等缘故，CMN 里面做了 Downgrade 的设计，一旦 CMN 逻辑监测到几个 Write Full 是相同的 AWID，其后续的 Write Full 会 Downgrade 成 Write-Partial。

911. 如果 AXI 不支持 IO-Coherence，那么 PCIe 的 Snoop Feature 也是不是也不起作用？

是的。

912. PCIe Prefetchable 和 Non Prefetchable 空间的区别在哪里？

在 ARM 系统中，两者基本一致，未作区分。从硬件来看，两种请求均从总线 HN-I (非一致性主节点)节点下来到 PCIe，走的同一条路。

具体来讲，Prefetchable 及 Non Prefetchable 两者都是分的 Device Memory 的属性，都是 No Cacheable 的，一般是映射的 nGnRnE 属性的 Memory。通过软件配置 SMMU 可以选择空间属性为 Device No Cacheable 还是 Normal No Cacheable。即 Prefetch 也可以配成 Normal No Cacheable，但一般不这样用。此外，Prefetch 一般是用 64 bit BAR 分配在系统 4 GB 以上的地址。

实际使用过程中，分成 Device 和 Normal 后遇到下述情况：在 CMN 上拆包行为不一样，Device 的会按 asize 来拆，Normal 按 64 B 拆。

如果把 BAR 配置成 Cache，更新 CPU Cache 数据到 PCIe 外设 BAR 空间的延迟会比较大。如果非要把 BAR 配置成 Cache 空间，个人理解也没问题，但需要严格配置确保硬件执行时访存指令字长和程序里的访存指令的字长保持一致，且能够保证指令执行顺序。

913. Type 1 字段里面没有 Non-Prefetchable 设置，是否意味着 Switch 下方的 EP 不能有 64b Non-Prefetchable BAR？

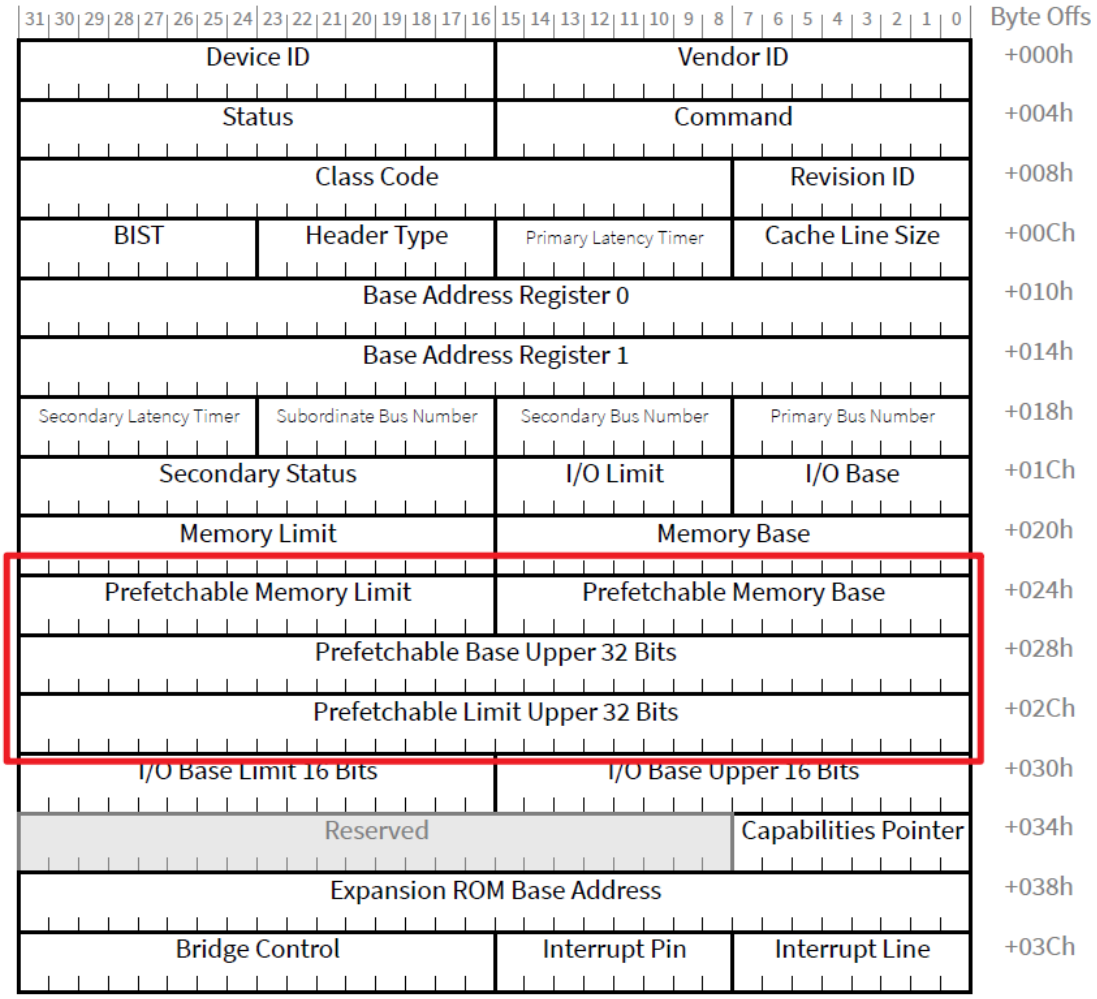


Figure 7-14 Type 1 Configuration Space Header

不是。0x10、0x20 寄存器 BAR 可配成 64b Prefetchable。

914. 相对于 Non Prefetchable BAR ， Host 读写 Prefetchable BAR 的行为具体是什么样的？x86 上这个内存属性的设置是不是根据这个 BAR 属性自动设置的？

附： LDD 上有如下解释，但是有些笼统。

Accessing the I/O and Memory Spaces

A PCI device implements up to six I/O address regions. Each region consists of either memory or I/O locations. Most devices implement their I/O registers in memory regions, because it's generally a saner approach (as explained in the section "I/O Ports and I/O Memory," in Chapter 9). However, unlike normal memory, I/O registers should not be cached by the CPU because each access can have side effects. The PCI device that implements I/O registers as a memory region marks the difference by setting a "memory-is-prefetchable" bit in its configuration register.* If the memory region is marked as prefetchable, the CPU can cache its contents and do all sorts of optimization with it; nonprefetchable memory access, on the other hand, can't be optimized because each access can have side effects, just as with I/O ports. Peripherals that map their control registers to a memory address range declare that range as nonprefetchable, whereas something like video memory on PCI boards is prefetchable. In this section, we use the word *region* to refer to a generic I/O address space that is memory-mapped or port-mapped.

ARM 是设置区域的内存属性，然后根据 BAR 属性申请对应区域的地址资源，主要是看 PCI 地址空间配置的内存属性。对于 ARM，不管 BAR 的属性为 Prefetchable 还是 Non-Prefetchable，都是 Non-cacheable 空间，软件处理是一样的，硬件上也一条通路，理论上可以放开 GRE 属性。对于 Non-Prefetchable 没有要求必须要是 nGnRnE，两者没有绑定关系，都是设置的 Device 属性，即都是 Non-Cacheable 的，GRE 可能有些区别也可能没有。

在 x86 中，64 bit Non-Prefetchable BAR 地址空间有限，个人理解软件上还是会把 Host 预留的 BAR 空间再划分成 Prefetchable 和 Non-Prefetchable 的。比如 Host 硬件上总共预留 64 G 的 64 bit BAR，软件可能在其中划 4 G 给 64 bit Non-Prefetchable。这时候枚举，下面申请的 64 bit Non-Prefetchable BAR 总和超过 4 G 就会分配不出来。

915. NVMe 盘一般发出读写是 Snoop 的还是 No Snoop 的？

取决于配置。

916. EP 把 RC 端的 DDR 颗粒当成 EP 自己的内存使用，即 EP 把自身 PCIe 空间当作 Non-Cacheable Memory 去用，在该地址段内做 memcpy、memset、创建结构体等操作。当结构体内存在 char 类型的成员时，对该结构体内多个成员跳着赋值时出现写不进去的情况，可能是什么问题？如何解决的？

EP 直接写本地内存没问题的话可能是对齐问题，因为应用程序端不保证访问的变量是对齐的。

此前层遇到过类似问题，RC 读写 EP BAR 空间，64 位对齐时没问题，非对齐时读不到最新数据。解决方案是关闭 RC 上的 No Snoop。可参考。

PCIe 性能

带宽

917. 一般如何测试纯 PCIe 的速率？

如果为设计阶段性能仿真，可以用 UVM 环境造场景测 RTL 性能。

如果为回片后性能实测，考虑到软件开销，不太容易测出纯 PCIe 速率。不考虑软件开销的话，如果为 Device 可以上真机跑 Benchmark；如果是 Server，可以跑 iperf，但 iperf 有 TCP 的拆包解包，不是真的 PCIe 速率，但印象中 iperf 结果和硅前测出的性能数据很接近。

有公司会接一个专门的测试 EP，上 Bare Metal 测试纯 PCIe 速率，Xilinx IP 可以用 XDMA 来测试 PCIe 带宽。

协议分析仪抓包写脚本也能分析。

918. Gen5 x16 Device 往 DDR 方向发 64 B 的 MRd，性能一般多大？

下图为理论最大带宽，具体性能取决于诸多因素，比如 DDR 内存带宽。

PCI Express 版本	编码方案	传输速率	吞吐量			
			x1	x4	x8	x16
1.0	8b/10b	2.5GT/s	250MB/s	1GB/s	2GB/s	4GB/s
2.0	8b/10b	5GT/s	500MB/s	2GB/s	4GB/s	8GB/s
3.0	128b/130b	8GT/s	984.6MB/s	3.938GB/s	7.877GB/s	15.754GB/s
4.0	128b/130b	16GT/s	1.969GB/s	7.877GB/s	15.754GB/s	31.508GB/s
5.0	128b/130b	32 or 25GT/s	3.9 or 3.08GB/s	15.8 or 12.3GB/s	31.5 or 24.6GB/s	63.0 or 49.2GB/s

CSDN @小孟boy

PCIe3 之前内存带宽基本不是瓶颈，到了 PCIe4、PCIe5 要看 DDR 带宽了：

- ✓ 对于 DDR4 3200，不考虑 Interleaving 时通道带宽: 64x3.2/8*80%(效率)=20 GB/s，仅够 1 个 PCIe5 x4 的单向吞吐量；对于多通道内存且开启 Interleaving 时能有所改善。
- ✓ 对于 DDR5 4800：4.8x64/8*80%=30 GB/s，大概能匹配 1 个 PCIe5 x8 的单向，如果考虑双向，只有 x4 了。

上述效率 80%是比较理想的情况，实际很多差的 CPU 70%到不了。实际使用过程中 PC 申请的内存不连续时也会影响效率。此外，对于 64 B 的有效载荷还需乘以 70%以扣除编码开销、TLP 包头包尾、DLLP 等开销。

919. PCIe TLP Posted 写没有反馈消息，如果 CPU 一直写把写带宽打满了会怎样？

PCIe Link 上有基于信用量的流控机制，带宽打满前会写不出去。

920. 测试发现按 MPS/MRRS 拆包性能较低，PCIe 怎么拆包性能好点？

这里的系统性能低应该跟按 MPS 拆包没关系，瓶颈不在拆包粒度，拆包粒度比 MPS 小性能可能更低。从 TLP 开销角度讲，在 MPS 之内拆包粒度越大，有效数据占比越高，整体性能越高。大概率还是在系统层面 SoC 内部其他地方堵塞，或者其他地方拆包、组包导致性能差。

921. 理论带宽 7.8 G 的 PCIe Gen3 x8 设备实测 DMA 写带宽只有 4.6 G，可能是什么原因？

可能为链路信号质量问题，信号质量不好、错包太多导致重传次数多。

如果为 DMA 写，可能是写到 Cacheable 地址区域了，CPU 侧硬件做了大量一致性操作影响性能。DMA 写操作不需要报错 Cache 一致性，可以配置 TLP 数据为 No Snoop。

可能为流控的原因，Credit 不够、接收端收不过来，影响性能。

922. NVLink 带宽为什么比 PCIe 大这么多？

一方面看单 Lane 速率，NVLink 也要比 PCIe 速率高很多，单 Lane 速率已经能够达到一两百 GT/s；另一方面，NVLink 物理 Lane 数多，且只有 Nvidia 自己用。

923. PCIe3.0 x4 实际性能在两个方向分别大概能跑到多少？有 80%利用率吗？

PCIe 性能跟诸多因素有关，比如 RCB、MPS、Payload 粒度、访问方向、地址分布、物理 Lane 数量、响应方的接收能力等等。如果是读，跟 RTT 延时、NP Outstanding 数量也有关。如果用了 DMA，还跟 DMA 的 Channel 数量、读写权重等配置有关。从效率上来看，一般在 75%~80%，能达到 80%属于不错的情况。

以下为部分同行们在 PCIe 性能方面的使用经验：

- ✓ Payload 粒度为 256 B 比 128 B 的性能更好。
- ✓ DMA 两个方向性能差距感觉较大，DMA 写 85%，读方向只有 71%左右，不确定是否因厂而异，也可能是因为 DMA 读写权重设置问题。
- ✓ 在做 SSD，Gen4 x8 顺序只读带宽能达到 7 GB/s，超 80%了；写操作有数据落盘问题，速度稍慢。
- ✓ 此前 Cadence 公司 PCIe DMA IP 存在性能问题，单 Channel 有气泡、无法打满带宽。

Latency

924. PCIe 收发两端的延迟高达 100 ns 以上，这个延迟的主要贡献者在哪里？物理层和数据链路层只占很少一部分吧？

Layer	Typical latency	Latency source	Minimization methods
PCIe Driver & OS kernel	~ms to seconds	OS scheduler and PCIe driver	Use kernel bypass (e.g., DPDK), HW assist acceleration
Transaction layer	~500ns	Protocol-related credit & transaction	Use alternative protocol (e.g., CXL)
Data Link layer	No errors: few ns Retry: ~us to ms	Error triggered retransmit	Improve the system or link Bit Error Ratio (BER) by using retimers
Physical layer – logical portion	Few ns	Encode/decode serial to parallel	Optimize the silicon architecture
Physical layer – electrical portion	No errors: few ns Link recovery: ~ms	Poor Signal Integrity (SI) and noise can trigger Link Recovery	Ensure that there is enough SI margin
Retimers	64ns or less	Elastic buffer, lane to lane skew compensation	Use common clocks and control skew on a system basis allowing the use of a short bypass buffer

（图源：Solving PCIe Latency with Retimers and Elastic Buffers - EDN）

100 ns 的延时算小的，实际服务器中，EP 发起的 DMA 读操作返回数据延时高达 1 us 以上的也很常见。

个人理解，PCIe 端到端延迟最大的贡献者应该在 TL 层以上的系统/架构层面，比如软件层面的拆包/组包，比如硬件层面的地址转换、在 NOC/Bridge 里的仲裁；次之是 TL、DL、Logic PHY 等 PCIe IP 内部 TLP 拆包组包、编解码、插 DLLP 引入的延迟，以及为满足数据处理复杂度、高频时序收敛索隐入的 Pipeline 级数；最后才是金手指走线等物理链路引入的延时。

DMA

925. PCIe 的 DMA 为何在 EP 端呢？有何好处吗？

也可以放在 RC 端，但地址要落在 EP BAR 范围内，比较复杂。PCIe DMA 大多做在 EP 端原因有以下几点：

- ✓ 如果 RC、EP 端都有 DMA，双方需要协商好用哪个 DMA，这决定了写主机驱动还是设备驱动，驱动程序不好写，比较麻烦。
- ✓ 如果放在 RC 端，同一 RC 下可能挂了很多 EP，就需要多个 DMA。DMA 数目放多了浪费，放少了不够用、多个 EP 之间还需要仲裁、还需要等待。即便数目刚好够用，不同 EP 对 DMA 能力的需求也不同，RC 端的 DMA 不一定适合 EP。

- ✓ 根据使用场景的不同，不同 EP 对 DMA 能力的需求不同，可能需要很多定制化设计，这时显然直接使用 EP DMA 是最合适、最方便的。DMA 驱动直接跑在 Device 侧的 CPU 上，也能减轻主机负载。
- ✓ 产品应具备自我完备性，对 EP 而言，如果其需要用到 DMA，其不能假设 RC 端有 DMA，万一 RC 没有 DMA 就没法用了，不如直接自己带一个，也方便针对性能做定制设计。
- ✓ 有些 EP 需要 P2P 通信，如果没有 EP DMA，P2P 访问就需要绕道 RC DMA，效率较低。

926. 一般 EP 端应该都带 DMA，那还需要 RC 端的 DMA 吗？

RC 端仍然需要 DMA，有几个场景需要用，以加速 EP DMA 用不了的地方。

927. 对 EP 而言，使用 Host 端的 DMA 与使用 EP 自己的 DMA 搬数据两者效果一样吗？

应该没什么区别。

928. PCIe Core 里有 DMA 吗？是协议规定的吗？

附：PCIe 传输和 DMA 传输有什么区别吗？

PCIe Core 里加 DMA 为非协议标准，厂商自己加的，一般加在 MAC 更上的层次中，比如 TL 层、Application 层。

929. PCIe DMA 某 Channel 传输异常后不能继续使用了，请问有什么复位的办法吗？

该问题难以回答，需具体情况具体对待。自研的 DMA 就去问开发的人，用的谁家的 IP 就去提 Case。

930. HDMA 这个 Feature 用的场景多吗？什么场景会用？

现在用 HDMA 的不多，在存储上用的多。

931. Synopsys 公司的 PCIe 用 HDMA 搬数据时需要用到 iATU 吗？

可用可不用，如果不用可在相关 ATU Entry 内设置 hdma_bypass=1。

932. DWC 的 PCIe EP Driver 把 DMA 的 Link-List Descriptor 放在 EP BAR 空间，为什么不放在 Host Memory 上？

EP DMA LL Descriptor 如果放在 Host Memory，那 EP 启动 DMA 之前还得去 Host 把这些 Descriptor 读回来，无异于舍近求远。

933. EP 通过 DMA 往 RC 的 DDR 搬数据，搬完最后一笔之后通知 RC，EP 如何保证完成中断到达 RC 之前写数据已经到达了 RC DDR？是否存在可能，RC 收到 DMA 完成中断时后，RC 的 CPU 开始读，但 EP 来的这笔写数据还在路上、还没到达 DDR？

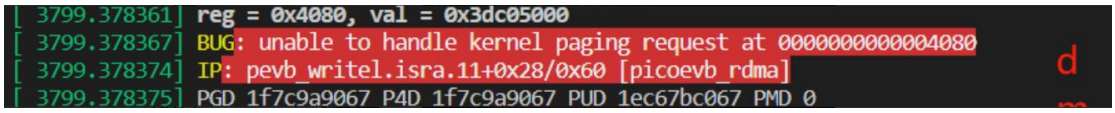
跟 EP MWr 中的 Relax Order（RO）设置有关，EP MWr 和 CPU MRd 都是 Memory 请求，两者之间有保序机制。一般是禁止开启 RO 的，如果开启了 RO=1，Cpl 和 Posted MWr 之间没有严格的保序机制，但 RC 内部一般无阻塞设计，一般不会卡在去 DDR 的路上。

如果 Host CPU 访问 DDR 跟 EP PCIe MWr 写 DDR 在 Host 侧走的是不同的 AXI 口，有的系统能保序，有的不能。

此外，还得考虑 Cache 的影响，需要是 No Snoop 属性。

934. 在驱动程序里面将 DMA 描述符配置好写到 DDR 里面，CPU 通过 PCIe 总线写 BAR0 地址数据来配置并开启 DMA，之后写不进去，出错之后就卡死了，可能是什么原因？

问题补充：错误信息如下。



根据报错信息，DMA 访问地址为 0x4080，虚拟地址没有这么低的，可排查该地址是否没有映射。

反馈：该地址采用 ioremap 有误，DMA 地址应该是 mem。

935. 基于内核驱动端的 DMA 传输函数计算带宽，传输数据量增大但传输时间无明显变化，可能是什么问题？

问题详述：下图为测试 DMA 带宽的代码，其中红框处为内核驱动端的 DMA 传输函数。在传输前后记录时间 ts/te，传输数据量/(te-ts)计算带宽，这种方式是否有问题？实测增大传输数据量后传输时间(te-ts)无明显变化，是计数函数的问题还是传输函数的问题？

```
ts = ktime_get_ns();

ret = pevb_dma_h2c_multi(pevb, &src_ubuf, dma_params.dst, dma_params.len);
te = ktime_get_ns();
if (ret)
    goto put_userbuf_src;
printk("dma_params.len = %llu",dma_params.len);
printk("ts = %llu",ts);
printk("te = %llu",te);
```

方法没问题，但仅凭上述代码无法定位问题。建议发包时尽可能打满 PCIe 带宽，同时加大数据量进行测试。

936. 使用 DMA 引擎时，软件发 Doorbell 告知 DMA 去搬数据，那怎样控制发的 Doorbell 是 Active 还是 Inactive 的？

问题补充：Active 是指 DMA IP 要继续处理，Inactive 是 DMA IP 根据自身需求来判断是否要处理。

如果是 NVMe 里定义的，可以查看 NVMe 文档。

如果是自定义的，那么可以由 IP、固件来处理，或者由驱动来控制 DMA Channel 的 Enable/Disable。Doorbell 应该也能 Disable Channel。

937. PCIe 网卡用 DMA 搬移数据，数据会封装成 TLP 进行传输吗？

PCIe 链路上都是打包成 TLP 传输。就 PCIe 网卡而言，这个时候网卡是 Device，搬到 Host(RC)肯定是 TLP 报文，经过 Serdes 差分线传输。

938. 用 DMA 做 Mem2mem 链式数据搬运，每个描述符搬 1 M 或更大数据量时会卡住，可能是什么原因导致的？

问题详述：在 EVB/FPGA 平台上用 DMA 做 Mem2mem 链式数据搬运，每个描述符搬 1 M 或更大数据量时会卡住，数据量小时卡住概率变低，PCIe 链路正常一直处于 L0 状态，可能是什么原因导致的？

问题补充：协议分析仪抓包发现，卡住是因为发出 MRd 无返回包，又发了一次 MRd。速率由 Gen3 降为 Gen1 后卡住的概率也会降低。

导致该问题的可能原因有很多，且在 PCIe 方便这个问题的确不好排查，大致有以下几点：

- ✓ 可查看 DMA 相关状态寄存器有无异常，是否 DMA 处理不过来丢数据了。
- ✓ 可查看 DDR 内存相关状态寄存器，是否内存控制器数据出了问题。
- ✓ 可查看 NoC 等总线状态寄存器，确认是否拥塞、死锁。

✓ 可查看 PCIe 读请求是否为 Relaxed Ordered，或者有没有死锁。

后续：DDR 卡住时的速率为 2133 MHz，降速后问题未再出现，持续压测几小时均无问题，故大概率跟 DDR 初始化有关。

939. EP DMA 不同 Function 发出相同地址的读写 Mem 包，最终映射到内存地址相同吗？

取决于 IOMMI 是否打开及 TLP 中的 AT 字段。IOMMU 打开时，如果 AT=2'b00，表示地址未翻译，那么不同 BDF、PASID 在 RC 端对应的 PA 不一定相同，即不同 Function 最终映射的 RC 内存不一定是同一个地方；如果 AT=2'b10，表示转换之后的 PA，不同 Function 发起相同 PA 的 Mem 请求能够映射到 RC 内存同一个地方。

940. EP 初始化后做 EP 通过 DMA 读 RC Memory，开始时 RC 正常回 CplD，过了一会儿后 RC 就不回 CplD 了，可能是什么情况？

可查看 LTSSM 状态机状态，排查 Link 是否发生了异常。（反馈：Link 正常，异常时有 MRd 的 Ack 回来但没有 CplD。）

可排查正常与异常时的地址，确认是否访问了非法地址。（反馈：地址为 CPU 域地址，反复读相同地址，不存在地址异常的情况。）

反复同一地址会性能较差但不至于一直不回，还可以排查下是否中间夹杂了 Posted 写 MSI 等，确认是否地址有异常。

941. 用 DMA 往 Host 搬数据，相同 PCIe 链路速率情况下链宽 x8 的性能是否是 x4 的两倍？

在其他条件全部不变的情况下，大致可以认为 x8 性能是 x4 的两倍。

PCIe DMA 性能取决于诸多因素，比如 DMA 的 Outstanding 配置、PCIe 侧及 SoC 侧的 Round Trip Latency，跟 PCIe 侧跟 SoC 侧的 MPS、MRRS 也有关系。如果 x8 带宽足 x4 带宽的两倍，大概率是 DMA 性能配置不足，无法打满 x8 带宽。

942. 用 DMA 往 Host 搬数据，相同 PCIe 速率时链宽 x8 带宽性能相较 x4 提升不明显，可能是什么原因？

补充说明：Gen3 速率下 C2H 方向搬数据，Payload 粒度为 512 B 及 256 B 时 x8 带宽均能达到 x4 的两倍左右，但粒度为 128 B 时 x8 带宽仅比 x4 多了不到 800 Mb/s（理论上应大几十 Gb/s），

如果粒度影响 AXI 的 Burst Length，则可能是 AXI Outstanding 问题。C2H 搬数据是往 Host 发 MWr，不需要 Response，基本可以排除 Host 端 AXI Outstanding 问题。DMA 写 Host 前需要先读本地 DDR，读需要 Response，粒度较小时 Burst Length 也小，在读延迟不变的情况下，撑满 x8 带宽所需的 Outstanding 数目更大。因此，猜测是受限于设备本地 AXI Outstanding 数目，无法打满 x8 带宽。

IP 设计与仿真

设计方法与理解

943. Synopsys PCIe Controller 的 sys_aux_pwr_det 输入信号是从哪里拉过来的？

可以从外部寄存器或 PHY 接出来。

944. PCIe 配置空间本身已有 4 KB，加上自定义的寄存器后不就超过 4 KB 了吗？

补充说明：Synopsys PCIe Controller Databook 说每个 Function 有 4 KB 的寄存器空间，其中包括 PCIe Spec 上规定的 PCIe 配置空间和 Synopsys 特定的寄存器空间。

协议规定 PCIe 配置空间是 4 KB，但实际用不满、只用了一部分，且 Capability 是以链表结构存放的，中间可以存在空隙。Synopsys 在 4 KB 配置空间内 PCIe Capability 未使用的部分加了其自定义的寄存器，比如做了 0x700 开头的 PORT/IATU/DMA 等寄存器，再比如 CXL 相关的 DVSEC 寄存器，整体未超过 4 KB。

945. 请问 Synopsys 家 IP 中哪个寄存器记录了当前链路状态是不是 L1.1/L1.2？

可查看 Silicon Debug 寄存器组中的 SD_STATUES_PM_REG，其中 L1SUB_STATE 能查看 L1 的子状态。

RSVDP_31	31
PM_USP_PROT_STATE	30:27
PM_L0S_STATE	26:24
LATCHED_NFTS	23:16
L1SUB_STATE	15:13
PME_RESEND_FLAG	12
PM_USP_LINK_STATE	11:8
PM_DSP_PROT_STATE	7:4
PM_DSP_LINK_STATE	3:0

946. Synopsys PCIe Controller 寄存器分下述三类，其中 ViewPort 该如何理解？

3.5.1.1 Register Types

There are three types of registers in the controller as shown in Table 3-5 on page 134.

Table 3-5 Types of Registers in PCIe Controller

Register Type	Description	Access/Addressing Method
Simple	Single register accessed directly by its address. Most of the registers are of this type.	Direct addressing method. You just supply the address.
Shadow	Another register exists at the same address as a Simple register. For example the BAR Mask registers have the same address as the BAR registers.	Normally can only be accessed through the DBI where you select between the two registers using the dbi_cs2 input. When you are using an AXI DBI slave, shadow registers can only be accessed through the DBI where you select between the two registers using the CS2 address bit. This is called DBI2, CS2, dbi_cs2, DBI_CS2, or Dbi2 access; all of these terms mean the same thing.
Viewport	Multiple (n) registers existing at the same address as a Simple register. For example Gen3 Coefficient Preset registers.	Indirect addressing method where you first write to a special index register to select which of the n registers at the address you want to access. You then proceed to write to the register address.

iATU 或 DMA 寄存器会有这种 ViewPort 寄存器。ViewPort 可以理解为软件的“片选”信号，就是 ViewPort 配置在某个寄存器内作为 Index，接下来访问 iATU 寄存器地址，就是对应 Index 的窗口 iATU 的配置。

947. Synopsys 公司 Controller 配置空间里的 Shadow Register 是做什么的？

比如 BAR 的 Mask。

948. Synopsys PCIe 配置空间里的 Shadow Register 跟常规影子寄存器的定义有何不同吗？

定义上没什么不同。

949. Synopsys PCIe 内 DMA 相关寄存器是否不能用 Cfg 包访问只能用 Mem 包访问？

是的，DMA 寄存器只能通过 Mem 包访问。Synopsys 公司 DMA 寄存器在 4 KB 之外，对于其他 S 自定义的寄存器，如果在 4 KB 配置空间内，可以通过 Cfg 或者 Mem 包来访问。

950. 请问 PCIe 配置空间 0x700 后面这些配置是什么呢？

问题补充：Realtek 的网卡，PCIe Extend Capabilities 到 0x178 的位置就结束了，但是后面还能读出来有效数值。

放的设备自定义的寄存器，放什么都行。如果是 Synopsys 的 PCIe 控制器，0×700 后面是是 IP 自定义的 Port Logic 寄存器，其他家 IP 不清楚是否类似。

951. 为什么 ELBI 访问要给 BAR Num 的信息？

During a DBI access, when you are not using the AXI DBI slave, you can specify the function number using the `dbi_func_num` input signal instead of `dbi_addr[18:16]` input signal. In addition, the controller drives the ELBI outputs (`lbc_ext_io_access`, `lbc_ext_rom_access`, and `lbc_ext_bar_num`) as follows:

- `lbc_ext_io_access =dbi_io_access`
- `lbc_ext_rom_access =dbi_rom_access`
- `lbc_ext_bar_num =dbi_bar_num`

为了访问挂在 BAR 上的 ELBI 寄存器。

952. Synopsys PCIe VIP 可以用协议分析仪进行分析，需要有协议分析仪设备吗？

不需要仿真设备，用 Verdi 看即可。VCS PCIe Native Protocol Analyzer 配置方式如下：

【VCS】PCIe Native Protocol Analyzer 使用方法_pcie protocol analysis-CSDN 博客

953. Synopsys 公司的 PCIe Controller 用 AXI Bridge 时，发送 NP 请求中的 Requester ID 和 Tag 放在 AXI ID 上吗？

一般 Requester ID 中的 Bus Number 和 Device Number 不需要特别指定，只需指定 Function Number。AXI Master 给带有 AXI Bridge 的 PCIe IP 发 NP 请求时，有两种方式可以提供 Function Number：

- ✓ 一种方式为通过 aw/armisc_info 等 Sideband 信号给 Controller 提供 Function Number，这种方式比较灵活，每一笔 AXI Burst 的 Function Number 都可以独立配置。misc_info 可以接到 axuser 或者其他自定义寄存器上，如果接到 axuser 上则 AXI Master 支持灵感的 axsuer 配置。
- ✓ 另一种方式为通过 iATU 来提供。这种方式能够减少 axmisc_info 等边带信号，但需要提前配置 ATU，且发出的请求需要落在相关 ATU Entry 内。

至于 TLP 中的 Tag 字段，Controller 内部有一套 AXI ID 跟 PCIe Tag 的 Mapping 关系，会自动把 awid/arid 转换为 TLP 所需的 Tag。

954. 既然 ATU 能填充 Function Number，为什么还需要 axmisc_info？

一方面，只有请求落在 ATU Entry 时相关 Function Number 才能上 TLP；另一方面，ATU 需要提前配置，且无法做到 AXI Burst 粒度的 Function Number 控制。

955. Synopsys PCIe Controller 作为 EP，配置 IP 时将 Controller 的内部寄存器映射到了 BAR0，那么 Host 访问 Controller 寄存器中的 Capability 寄存器时是用 Cfg 还是 Mem 呢？

理论上两种均可：

- ✓ 如果该寄存器位于 EP Config Limit 之内，可以通过 Cfg 请问来访问。
- ✓ 如果位于 BAR 范围内，可以通过 Mem 请求来访问，iATU 及 DMA 等地址 unroll 的寄存器可能会放在 BAR 内。

一般来讲，4 KB Config Space 不会位于 BAR 范围内。BAR 基地址是 RC 通过 Cfg 分配的，RC 不会把基地址 0 分给 EP。当然，如果 EP 某个 BAR 的输出直接硬连接到 4 KB 配置空间模块并做了地址相关逻辑处理，随便这个 BAR 怎么配置都是能够通过 Mem 请求访问到的；此外，如果 EP 开启了 ATU 及 BAR Match 且 Bridge 做了相关支持，可以走 Mem 请求访问先出去，在外边 AXI Bridge 上根据地址又绕进来访问到相关寄存器。

956. 1 G 速率下把 PCIe 的 Timing 过约 30%，即只有 0.7 ns 可用，这样是否过于苛刻？

不算太苛刻，放松点的话也可以约到 0.8 ns，20%留给后端正常情况下是够了的。

957. Synopsys PCIe IP 给 MSI-X Table 留了 8 K DW 空间，为何经 DBI 只能访问到 1 K DW？

MSI-X Table 是给 Host（或者链路对端）用的，正常是通过 Wire 来访问。根据 Synopsys 公司 PCIe Databook 描述，DBI 访问 MSI-X Table 仅做 Debug 用。MSI-X Table 是挂在 ELBI 上了吗？如果 DBI 进 LBC 又转 ELBI 的话，那最大只能访问 1 K DW，这是 DBI 访问 ELBI 的限制。

958. Synopsys 公司的 PCIe IP 支持在 IP 里注错吗？

支持，RAS_DES 相关寄存器内可以注错。

959. AXI 在 PCIe 里是做什么的，为什么很多 PCI IP 用 AXI？

AXI 是 SoC 片内总线，在 SoC 内比较通用，跟 PCIe 没有强绑定关系，也可以用 CHI 等其他总线协议。简单点说，TL 层再往上是接 AXI 总线，即 MAC 对外接 AXI，方便使用。

960. PCIe 对 AxPROT 有什么要求吗？

AxPROT 是 AXI 的安全、非安全访问属性，跟 PCIe 没有直接关系。

961. ECAM 是必选功能吗？

不是，ECAM 需要额外 License 的。如果没有实现 ECAM，需要自行实现 Cfg 访问。

962. Synopsys 公司的 PCIe EP 发出 MRd，RC VIP 反馈 Cpl.UR 后 EP 把这笔 Cpl 丢掉了，发起 MRd 的 AXI Master 等不到 Rsp，只能等 Cpl 超时吗？

Port Logic 寄存器里可以配置 AMBA 相关寄存器把这笔异常 Cpl 传到 Rrsp 上。

963. Synopsys 公司有 x1 的 PHY 吗？只需要实现 x1，用 x4 的 PHY 好浪费面积。

65nm 制程有 x1 的 PHY，速率支持 Gen1、Gen2。

964. PCIe VF 中的寄存器可以用 DBI 访问吗？

可以。SNPS PCIe Databook 讲 DBI Addr Bus Layout 章节中给出了访问方法，CDM 地址中有几 bit 指示 VF Active 和 VF ID。

965. Xilinx 的 PCIe PHY 连标准的 PCIe Controller，Xilinx 的 PCIe PHY EQ 接口非 PIPE 标准，需要自己设计逻辑转换吗？

是的，它们均衡算法不一样，需要做个适配设计，不然只能 Gen2。

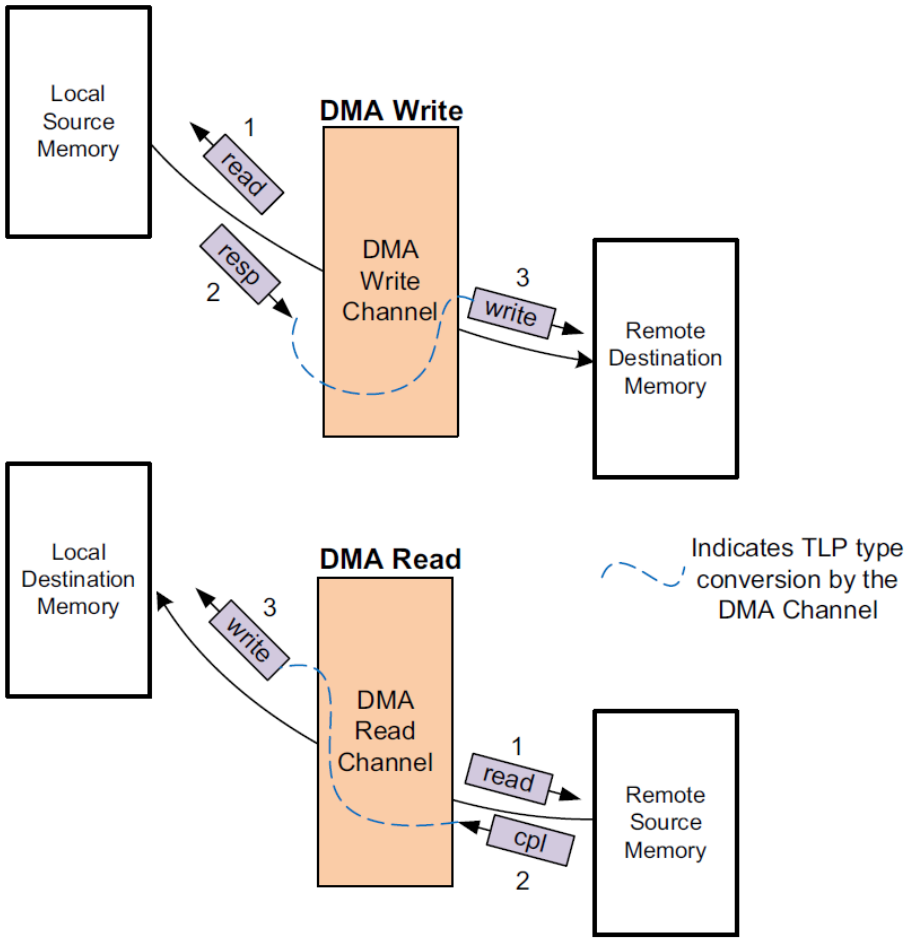
966. Synopsys 公司 PCIe IP 的 Port Logic 寄存器中到底包含 iATU 和 DMA 寄存器吗？

补充描述：根据其 Databook 第 3 章，EP Mode Controller Configuration Space Layout 图中写明了 Port Logic Registers 不包含 iATU 及 DMA 寄存器，在 USP Mode DBI/Wire Access 表格 Port Logic 寄存器中又包含了 iATU 及 DMA。

包含。Port Logic 寄存器是 SNPS 自定义的寄存器，一部分在 PCIe 4 KB 配置空间内，链路对端通过正常的 Cfg 请求能访问到；DMA 和 IATU 寄存器属于 Port Logic，但不在 4 KB 配置空间内，因此未体现在 Cfg Space Layout 图中，这部分寄存器只能通过 Mem 请求路由到 Trgt0 进行访问。

967. Synopsys 公司 PCIe DMA Descriptor 中的 SAR 目标地址是 RC 端的 PCIe Bus Address 还是 EP 端的 Bus Address？

PCIe DMA 是两个地址域之间的桥梁，SAR 和 DAR 分属于是两个不同的地址域。本地 DMA 作为 AXI Master 是挂在本地的总线上的，DDR 是作为 AXI Slave 挂在本地总线上的，因此 DMA 看 DRAM 地址为本地地址。如果需要将本地数据搬到对端，SAR 为本地 AXI Address，DAR 为 PCIe Address；反之，如果需要将从对端搬数据到本地，DAR 为本地 AXI Address，SAR 为 PCIe Address。可参考下图。



968. Synopsys 公司 PCIe Controller 中，Dedicated 和 Shared AXI DBI 模式有什么区别？

两者在 Misc Info、Data Width、Wstrb 上用法不太一样，Shared DBI 限制更大一些，更多详查 SNPS PCIe Databook。

969. 一般什么时候用 PCIe 的 Dual Mode 模式？

在芯片需要支持 RC、EP 多形态时可以用 Dual Mode 模式。比如芯片既可以做 PCIe 卡（EP），又可以做单芯片方案（RC）接 SSD 等外设。即使 DM 模式，RC、EP 工作模式也不是动态切换的，而是取决于芯片在板子上摆放位置，根据场景来自动配置或采用固定模式。

在通信行业里面 Dual Mode PCIe 用的也比较多，做 RC/RP 主要是为了控制下游的 WIFI 等 PCIe 接口芯片，做 EP 主要是跟 AP 芯片通信。

在多算力芯片互联场景中 PCIe Dual Mode 也比较多，互联的两颗芯片通过 Crosslink 的方式协商出 RC 和 EP。

仿真问题

970. 对于 Synopsys PCIe IP，Driver 发起 Mem 请求时 TLP Header 中的信息可以直接从 AXI Bus 拿到吗？该如何提供？

对于 Synopsys PCIe IP，TLP Header 中的信息可通过 Controller 的 Sideband（axi_misc_info）信号进行提供，这些 Sideband 信号一般拉到寄存器由软件配置，或者直接从 AXI User 上接过来，采用 User 的方式需要 AXI Master 支持发出数值可配的 User 信息。除了上述方式，还可以通过 Controller 内部的 ATU 进行转换，由 ATU 提供所需的 TLP Header 相关字段。

971. Synopsys 的 PCIe VIP 中 Unified 的 VIP 有没有一些说明文档呀？

pcie_svt_uvm_user_guide 中有一节专门讲了，Unified PCIe VIP Component。

972. Synopsys PCIe VIP 的 Unified TB 中，选择 Serdes 连接时参考时钟应接到哪个接口上？

VIP 默认不需要接，DUT 需要接参考时钟，VIP 和 DUT 连接 Tx/Rx 就好了。

973. PCIe VIP Unified TB 中，如果想要连接 DUT 去替换 Root 的话，是不是仍然要实例化 Root 和 Endpoint，只是将 Root 的 connect_active_vip 置 0，Env 中仍然要例化 Root 的 Agent 和 Cfg？

可连可不连。如果使用了 VIP Example 环境或者 Test Suite 的环境，那还是连上吧。自己从头写且不打算复用 Test Suite 提供的 API，多例化一个 SPD 的必要性不是很大。

974. Synopsys 的 PCIe VIP 可以支持多 Port 吗？就是一共 8 Lane 分为两个 x4 Port。

svt_pcie 的 spd 和 device_agent 数目都是可指定的，新版 VIP 宣称也支持 1spd+2agent 的 Bifurcation 用法。

975. 按照 Synopsys 公司的 VIP 手册写了 Topology File 并例化、连接了 DUT，编译时报下述[XMRE]错误是何原因？

错误信息：

```
Error-[XMRE] Cross-module reference resolution error
svt_cxl_subsystem_topology_file.svi 93
Error found while trying to resolve cross-module reference.
token ‘m_ser’. Originating module ‘test_top’.
```

```
Source info: defparam spd_0.m_ser.port0.SER_GEN_1.serdes.ADJUST_RX_CLK_MODE = 4;
Instance stack trace:
 test_top#(….
```

直接原因是跨模块引用出错，即引用了不存在的 Hierarchy 路径。根据 Log 具体来看，可排除以下两点：

- ✓ m_ser 是 serial connection，确认相关 macro/parameter 是否误定义为了 PIPE connection；
- ✓ SER_GEN_1 报错，先确认 SER_GEN_0 有没有 Compile Error，如果没有 SER_GEN_0 的 Error 进一步确认不同 Instance 中 Physical Lane Number 相关 Parameter 是否正确。

976. Synopsys VIP 中 PIPE 接口 pclkreq_n/pclkack_n 的这两个信号是什么？

Synopsys 自定义的，L1 过程中控制 spd pclk，具体使用方法可参考 VIP Example 中的 Interconnect 文件。

977. PCIe 仿真 Link Up 后仿真日志报“Detected EIOS, Receiver entering L0s”，是什么原因？

开了 ASPM L0s，长时间未发包自动进 L0s 了。

978. Synopsys 公司 PCIe Controller + Xilinx PHY 接 Synopsys PCIe VIP 做 EDA 仿真，LTSSM 走到了 Polling.Active 但 Xilinx PHY 的输出仍然是 High-X，可能是什么原因？

可排查 Xilinx PHY 是否在大概 2.4 ms 处释放了复位，如果没有问题可尝试把 VIP 释放复位的时间设置到 2.4 ms 之后，或者设置 Polling 超时时间。（反馈：phy_status_rst 在 2.478 ms 处释放了）

979. 如何用 PCIe VIP 发起 Stash RNF 访问？

Stash 跟 PCIe VIP 没有直接关系，是由 SoC 实现的。不过一般都是发 TPH，由 SoC 做映射。

980. 请问 Synopsys 的 VIP 可以发 Cpl 包吗？

可以。自动、手动都可以回。

981. VIP 报 ECRC 错误，但数据是对的，这个错误可恢复吗？

ECRC 错误不可恢复。数据没错但 TLP 中的其他字段可能错了。

982. Synopsys PCIe VIP 如何使用 Shadow Memory 来检查写读相同地址的数据是否一致？

可参考 VIP 手册进行配置，比如打开 Shadow Memory，参数中设置 SVT_PCIE_UI_ENABLE_SHADOW_MEMORY_CHECKING Parameter、cfg 中设置 enable_shadow_memory_checking，例化 Global Shadow Memory 时 shadow_mem.AddMemRange 等。

983. 用 SNPS PCIe VIP 仿真，有办法从 Transaction Log 上算出收发数据量吗？

没有直接方法，需要写脚本自己分析。

984. Synopsys PCIe VIP 如何让 EP 发出 Cfg 报文？

EP 发不了，只有 RC 能发 Cfg。

985. Synopsys PCIe VIP，发 MRd 后 Shadow Memory 报错是什么原因？

问题详述：读之前先后门写了 DDR 和 Shadow Memory，也设置了 VIP Address No Translation，从 DDR 读出来的值是预先写入的，但从 Shadow Memory 读出的仍然为 0。

同一地址没发过 MWR 直接读就会报错，需要提前通过前门或后门写进去期望值，且需要调用 VIP 相关 Sequence 进行写才能记录在 Shadow Memory 中。如果仍未成功，可参考 PCIe VIP Databook 上的示例。

986. Synopsys PCIe VIP，如何关掉收到 CA 后的 appl_driver_missing_good_status 报错？

问题补充：异常测试中，DUT 回复了 CA，这是正常的。如何把 VIP 收到的这个报错关闭？

可参考 VIP 中 svt_pcie_driver_app_mem_request_exception_sequence 这个发错包的 Sequence，发包时设置 transaction_error_kink=CA。或者采用 svt_err_catcher 这个类统一管理，把这个 Error Demote 掉。

987. Synopsys PCIe VIP，在添加频偏时 Serdes PLL Unlock，该如何解决？

可以把 CLK_TOLERANCE 这个 Parameter 改大点。

988. Synopsys 公司的 PCIe VIP 可以用 Cadence 公司的仿真工具吗？

如果 VIP 没加密，那么可以用 irun 跑；如果加密了，那不行。

PIPE

989. PIPE 接口上的 rx valid 和 rx data valid 的区别和关系是什么？

rxvalid 管 rx，rxdatavalid 管 rxdata；rxvalid=1 时 rxdartavalid 不一定=1。rxvalid 代表接收端 Symbol lock 或者 block alignment 的获取成功，rxvalid=1 时其他的 rx 数据才开始有意义。

990. PIPE 接口 Datak 指示 Datad 哪 8 bit 有效，但对 8b/10b 编码后的 10 bit 该如何表示？

Datak 是 original pipe 独有的信号，为发送数据符号的控制位，仅用于 Gen1/2，每一 bit 对应 8 bit 数据。在 Original PIPE 中，Data 位宽为 8/16/32，是 8b/10b 编码前的数据。

PHY Interface for PCI Express, SATA, USB 3.1, DisplayPort, and USB4, ver 6.0

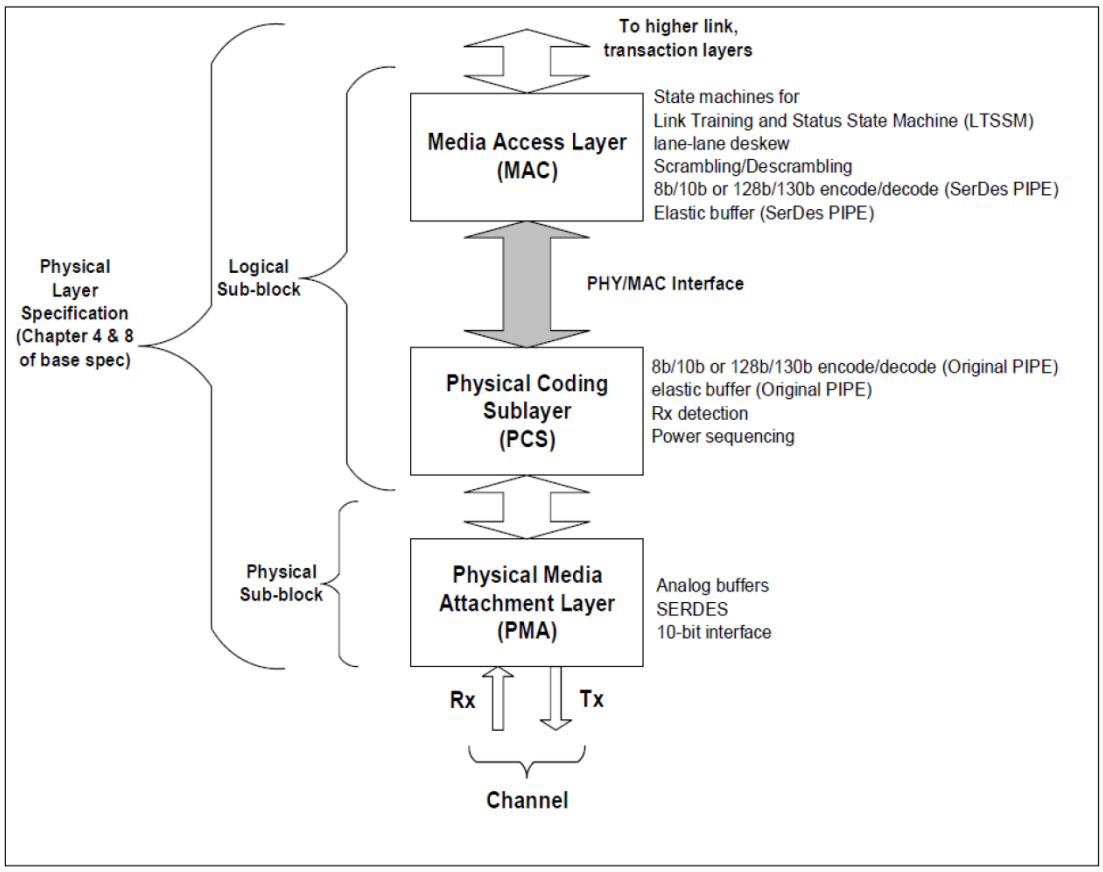


Figure 2-1: Partitioning PHY Layer for PCI Express

991. Synopsys 的 IP 是有 Gen5 速率的，为什么 PIPE 协议在 5.1 和 6.1 版本都没有定义 Gen5 速率 FS/LF 的寄存器？

有的，仔细找找，在 PIPE 协议 7.1 节 PHY Registers。

992. Synopsys 公司控制器接 Xilinx GTH PHY 只能用 16b 的 PIPE 吗？能支持 32b 吗？

切换速率时，Xilinx 一般是调 PIPE 宽度，Synopsys 公司一般配置成调 PIPE 频率，两个要匹配的话也就支持在 Gen2/3 左右。具体还要看选的频率还是 Symbol 模式，由用户自己选。

993. FPGA 平台上 Synopsys 公司 Controller 搭配 Xilinx PHY，PIPE 两边均衡算法不一致的问题是如何解决的？

FPGA 平台上一般在 Gen1、Gen2 速率下跑跑功能，不需要调均衡参数。

994. 使用 Synopsys PCIe 控制器和 X 家 PHY 做 FPGA 版本，两者之间需要做哪些处理？

问题补充：S2C 的 FPGA，PCIe Gen2 x8。Gen1 的话直接替换 PIPE 接口处理好时钟复位就好，Gen2 的话需要做那些处理？

Gen2 不需要特殊处理，Gen3 需要有一个 PIPE EQ 转换桥。

995. PIPE 口上的 TxDetectRx 已经起来了，为什么 LTSSM 没有退出 Detect？

TxDetectRx 是 MAC 告知 PHY 启动接收端检测，而非检测结果。检测结果要看 RxStatus。

996. PIPE 6.1.1 协议新增 Recalibration 允许 PCIe 控制器进入 Recovery 对 PHY 重新进行校准，但是 PCIe 规范中没有涉及这个行为，这块该怎么理解？

引文：The PIPE specification provides mechanisms for either the controller or the PHY to initiate recalibration. Recalibration must occur during Recovery, so if the PHY determines that a recalibration is necessary, it notifies the controller that it should enter Recovery and request a recalibration.

PCIe Base Spec 第八章讲了 Calibration 的内容，Recalibration 的流程看 PIPE Spec 就好了。Recalibration 之前需要 MAC 和 PHY 通过 Message Bus 发消息进行握手，两者均可主动发起；Recalibration 的执行者是 PHY。

997. Gen3 及以上速率时 SerDes Arch PIPE 接口信号中的 TxData 中每 10b 只用 8b 吗？

是的。Gen1/2时候需要做8b/10b编码，每个Symbol需要10b；Gen3/4/5时做128b/130b编码，每个 Symbol 只需要 8b。

应用

998. 手机用 PCIe 连接什么外设？

外接基带。

999. 如果芯片 DDR 有问题，可以使用 PCIe 接的 SSD 临时充当 DDR 用吗？

补充说明：主要是担心芯片回来 DDR 短时间内不通，想让芯片用 SSD 先跑起来进行后续工作。把 Code 放在 SSD 上跑，慢一点无所谓。

不可以。SSD 是 NAND，不能跑程序，即便程序放在 SSD 上，仍然需要拷贝到 RAM 上才能运行。也就是说，从 SSD 上读回数据仍然是要放在 DDR、RAM、iRAM 等地方，各级 Cache 最终还是要来 DDR 拿数据。即便把到上边这些接口都改道 SSD 也不行，SSD 不是内部总线，CPU 访问不到，且芯片架构就定死了走 DDR。再即便把软件接口也改到 SSD 上通过 PCIe 访问，PCIe 驱动、软件访问 SSD 的代码也没地方放。

芯片回片后，DDR 不一定立马调试稳定，一般都是先小 Code 在 IRAM 上调试各个模块，或者先在几 M 的 Cache 上跑 BIOS，等 DDR 稳定后才去跑大系统。一般 IRAM 或者其它内部 Share Mem 也不会很大，应该可以满足简单调试需求。

致谢

感谢参与讨论的：。、`骑士精神、80s、admin858、adrian、Alan、Aldan、Alice Eva、Amelie、Arwen、Astronaut、ball.冯 海 、Bazinga、Beibei、Beta~、Better、BH、bhyou、Brim.Lee、bingo、BugsBunny🐰 、Calcifer、camel、cathy、celia_xie、Cheney.Ren、Constantine、ck、CR 小队长🐿️、David、Debode、Debug 秋秋秋辰、DeYii、DMR、DongXie、DOUYUBA、DS、DwQ、dyc、Edouard、Einstein、Elvain、Eric-Xianyu、Ever、ewxxk、Felix、frank、furish、Gavin、Gloria、guoxu、H.Zhou、hans、Hanzong、Hao.、Harry 宋 、hazy、Hello、HH、HS、IC 小鸽、investu 王杰、Isabelle、jacker、Jianglfl、Jun、Kandri、kangling、kevin 宋 、kick、Kida、Leave、Lee、Lenv、lhl、Liam、linkyi li**yi、Ln、LR、lucissit、lv、Maggie、MangoPapa、Mobei_Peng、monkey~xiao、Mr 丶 w、Opus、PCIe 技术交流 、Picouth、rainy、random、resemble、richard、Roger、Satellite、segmentation fault、shawn 星、shyu、SKY、sound、SQ、Stan、Star、syt、T、thoralex、tian、uio、Venom、Vilmar.、Vin Diesel、Vincent、Wens Matthew、whitesnark、wszhong、wyf_taiga、XIAOSIR、xl、XLF、XXXXX、xy、XYX、YA、yh、YKK、yl.c、yu、Yue H、YYP、Z、zero、🙈、🐝、阿兵、阿锋、安庆、半猿半硅农、伴陈烟沙、冰狱明、曹杰、曹先杰、厂长🤠。、尘叶云渡、陈、陳建春、程明、程淑伟、程学习、吃花椒的喵醬、川、哒布溜 k😈、大白、丹青、邓仟、东木鱼人、豆儿、豆苗、杜文龙、范志强、飞、飞飞、飞天逐日、枫和雨、疯猫于是不知之、峰、福新、高二爷、咣当～、海、海不扬波、海绵宝宝【万一免五】、好久不见、好名字、虹太郎、花弄影、欢乐糗球、辉辉、极、坚强努力自律勇敢、角落里的梦、界贤、飓风、倦然～语默、俊、俊狗、克林、空帆船、夸克、来一口粟米条、老猫、老三、老姚、李兵、李胖、李群雄、李小强、里温、梁岩、两点之间、林、刘恒、刘琳琳、刘流鈺、刘旭辉、刘洋、陆小虎、鹿鸣、麦小客、曼施泰因、霉你不乐、每一刻都是崭新、米斯特-J、淼、民工松、木易凡、木子田、沐叶、南晏、匿名用户、牛明超、农夫山泉、暖暖的太阳、胖胖法式小面包、彭本华、皮皮锐、菩提清音、千、亲爱的、秦好强、秦木木、青简半卷、倾转旋翼、晴空一鹤排云上、趣多多、全频、戎国、若封、申越、神采逸逸、石静、时暮、史胜辉、世界之上、顺心、斯丢.皮特、松、苏 SCY、苏 、踏雪无痕、唐朝、唐古拉·老鸭、天明、天书奇谈、天天向上、听说者、土馒头、黫瑆萬、汪小明、王宁、王一昂、王一民、薇安、炜、文、问、我爱山水、我不是光光🚜⚪、我怕改了名字你们就不认识我了、我是一只特立独行的猪、无问西东、勿忘心安、雾中的大海、向生活比个 Ye、潇潇雨、小刚、小胡、小胡同学、小马小猪大兔子🐰 、小羊🙄 、小鱼儿、小张同学、谢智郎、心の音、心之所至、星洋、幸运草、徐汉权、徐强、炫、血轮眼、烟雨绘江南、闫莉莉、羊、杨柳、杨雪明、杨杨扬、尧烨、叶卡、叶子、一、一杯清补凉、一个依格、亦不可止、逸卓于群💕、勇决、鱼哥🐟、鱼之乐、宇、玉昂澈、早点睡、张凯、张清毅、张晓辉、长岛、赵升、櫂宸、折叠、箴言、执月、朱政、紫草石、自行车、坐看云起时。

感谢参与信息提取及文档整理的： MangoPapa、MangoPapa 小助理、折叠、先杰、CR 小队长、kangling、噫嘘唏、慕荷、JasonW 及皮塞工坊志愿者皮塞阿姨、席可儿。