Application Notes
ZHCADF5 – Oct. 2023

TI AFE8092/AFE8030 JESD204C 配置及调试手册 Part C

Zhizhao Niu North & West China Team

摘要

AFE8092，AFE8030 是 TI 一代高性能、大带宽，多通道收发器。集成了 8 个射频采样发射通道、8 个射频采样接收通道和 2 个射频采样反馈接收通道（AFE8092 中的两个反馈通道是和接收通道复用 ADC 的，AFE8030 中含有两个独立的反馈通道 ADC）。AFE80 发射通道在 8 通道模式下最多能够支持 800MHz 带宽，使其非常适用于多频段的 4G 和 5G 基站。800MHz 大带宽需要 260Gbps 的串行数据总量在 ASIC/FPGA 和 AFE80 之间交互。为了能够支持如此高的数据总量，AFE80 集成了 8 对 Serdes 收发通道，并且支持 JESD204C 协议，每条 lane 的最大速率能够达到 32.5Gbps。 JESD204B/C 接口作为系统中速率最高的总线之一，往往在产品研发调试阶段需要花费大量的时间去进行配置和调试。本系列文章分为两个部分，Part A 将以 JESD204C 为例介绍 AFE80 JESD204C 模块的组成以及用户将如何进行系统参数配置。Part B 中将详细介绍 AFE80 JESD 建联流程，常见的告警以及解决方法，Part C 将介绍两种优化 JESD 链路稳定性的方法。

目录

图

1 引言

在本系列文章“TI AFE8092/AFE8030 JESD204C 配置及调试手册 Part A、Part B”中已经详细的介绍了 JESD204 模块的组成以及配置方法，建链流程，JESD204C 告警以及解决办法。在本文中将详细介绍两种优化 JESD204C 链路稳定性的方法。

2 Serdes 眼图优化

Serdes 眼图质量会直接影响整个 JESD204C 链路的稳定性和误码率（Bit Error Rate, BER）。用户在研发过程中常常需要对 Serdes 链路进行长时间全温范围的 PRBS 测试以确保 Serdes 物理层的 BER 能够满足系统要求。因此，AFE80 Serdes 模块内部也集成了以下功能方便用户对 Serdes 链路进行调试。

- SRX 含有自适应的 CTLE 和 DFE 模块，最大可以支持 20dB 的板上插损。STX 含有 3 tap FFE 模块。
- 支持 PRBS9，15，23，31 等测试码型。STX 集成了 PRBS 码型生成器。SRX 集成了 PRBS 误码检测模块。
- STX 集成了客户自定义码型的发送器，可以使 STX 发出未加扰的特定码型，方便使用示波器去检测 STX 发送的模型。
- SRX 支持 Lane 的眼高回读，输出眼图以及浴盆曲线测试等功能。

本节将介绍以上几个模块的使用方法以及如何使用上述工具测试系统的 BER。

2.1 Serdes 模块均衡参数配置

2.1.1 SRX 均衡模块

AFE80 SRX Lane 中包含了 CTLE 和 DFE 两个自适应均衡模块。在默认的情况下 CTLE 和 DFE 会根据 Serdes lane 上的数据进行自适应得到合适的参数。但同时 AFE80 也支持在调试阶段手动配置 CTLE 参数。可以用以下的 CAPI 函数进行配置。

CAPI Name	Description
reAdaptSerDesAllLanes()	重置所有的 SRX 的 CTLE, DFE，将其重新自适应
setSerdesLaneCtleMode()	设置 SRX CTLE 时自动自适应或者手动模式
setSerdesLaneCtle()	设置 SRX CTLE 的值为 0~7。0 为 CTLE 补偿最多，7 为最少。该函数只在 CTLE 手动模式下有效
getSerdesLaneCtle()	回读当前 SRX CTLE 参数

resetSerDesDfeAllLanes()	重置所有 SRX DFE
getSerdesLaneDfe()	回读当前 SRX DFE 参数

2.1.2 STX Lane 均衡模块

STX Lane 含有 3-tap FFE 模块可以用来补偿 PCB 的板上损耗。FFE 含有三个可调参数 Pre-cursor，Main-cursor 和 Post-cursor。这些值可以通过 CAPI SetSerdesTxCursor()进行设置。通过不同 cursor 的设置，STX 的输出输出摆幅 Vpp 在 500mV~1000mV 之间变化。需要注意的是三个 Cursor 参数之和需要小于 25。AFE80 的数据手册中列出了常用的 Cursor 组合，用户可以通过 Green Box 测试遍历所有不同的参数组合以得到最优的 cursor 参数。

2.2 Serdes 模块调试功能相关配置

STX Lane 支持产生 PRBS pattern 用来测试上行 JESD 方向的物理层性能。还支持手动的加入 PRBS Error 去检查测试的可靠性。同时也支持 STX 输出用户自定义的码型，方便用户使用示波器对高速链路的数据进行检测。

SRX Lane 含有 PRBS Checker 去测试下行 JESD 方向的物理层性能。并且还可以回读经过 CTLE 和 DFE 之后的 Eye margin 以及输出 Eye diagram 和浴盆曲线。常用的 CAPI 函数如下表所示。

CAPI Name	**Description**
sendSerdesTxPrbs()	使能某一条 STX Lane 发送某种码型的 PRBS 码
serdesTx1010Pattern()	使能某一条 STX Lane 发送 1010 码型的数据
serdesTxCustomPattern()	使能某一条 STX Lane 发送特定码型的数据
serdesTxSendData()	使能某一条 STX Lane 发送 JESD 业务数据
serdesTxPrbsInsertSingleError()	将单 bit 的 error 加入到某一条 STX Lane
enableSerdesRxPrbsCheck()	使能某一条 SRX Lane 的 PRBS Checker
getSerdesRxPrbsError()	回读一段时间的累计 PRBS Error 值
clearSerdesRxPrbsErrorCounter()	清除 PRBS Checker 中累计的 Error 值
getSerdesRxLaneEyeMarginValue()	回读某一条 SRX Lane 经过 CTLE 和 DFE 的眼高
getSerdesEye()	回读某一条 SRX Lane 的眼图，使用方法见 2.3
getSerdesBathTub()	选一条 SRX Lane 进行浴盆曲线测试，使用方法见 2.3

2.3 Serdes 模块常用调试工具配置

2.3.1 Serdes 眼图回读

AFE80 除了能够通过 CAPI *getSerdesRxLaneEyeMarginValue()* 回读 SRX Lane 上的眼高数据之外，还能够和 Python 脚本配合输出 SRX Lane 上的眼图。AFE80 SRX IP 能够创建一个数据矩阵，适合绘制 BER 相对于 x 轴眼宽 UI 和 y 轴眼高 mV 的眼图。由于要回读 BER 矩阵，因此运行此测试可能需要几分钟的时间。在眼图中纵轴表示的是眼高，并且会根据每一个次回读的眼高信息进行比例放大或者缩小。通常绘制 Serdes 眼图分为两步。第一步将 BER 矩阵和眼高的比例因子等数据回读回来。第二步将上述的数据带入 Python 脚本中进行绘制。具体操作流程如下：

- 运行函数 getSerdesEye() 函数的运行结果会返回一个 BER 的数据矩阵，ber。以及对应纵轴眼高的比例因子，extent。

- 之后将 ber 和 extent 传递给下图中的 Python 脚本得到如图 1 中的眼图。完整的眼图由数据矩阵组成，大小为 33x95 像素。 每个像素返回一个 0 到 255 之间的值，可以通过函数 BER = 10(-pixel/16) 计算 BER，下图中右侧的颜色图例表示的为不同像素点的 BER 水平，BER 水平越小表示颜色越蓝。蓝色部分表示张开的眼图，黄色部分表示眼图的边沿，红色部分表示眼图叠加后重叠的部分。下图中眼宽分辨率约为 0.0232UI，覆盖范围从 -0.37UI 到 +0.37UI。 眼高分辨率会根据 extent 的值进行缩放，单位为 mV。

```
import numpy
import matplotlib.pyplot
ber=[]
for i in range(95):
ber+=eyeReadVals[i::95][:]
# ber is the array printed and extent is the extent printed in the c function.
ber=numpy.array(ber).reshape((95,33))
f=matplotlib.pyplot.figure()
x=numpy.arange(-16.5, 17.5)×0.0232
y=numpy.arange(-47.5, 48.5)×extent/47
c=matplotlib.pyplot.pcolor(x, y, ber/-16, cmap='jet', vmin=-8, vmax=0, figure=f)
cax=f.add_axes([0.9, 0.1, 0.02, 0.8])
f.colorbar(mappable=c, cax=cax)
matplotlib.pyplot.show()
```

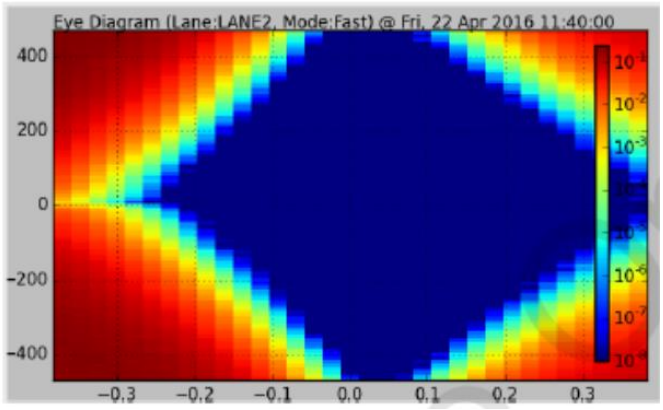

Figure 1. AFE80 getSerdesEye 回读眼图

2.3.2 Bath Tube 浴盆曲线测试

在研发过程中，如何保证 Serdes 链路在全温范围内的长时间稳定性是产品开发阶段的一个难题。因为 Serdes 的眼图质量会随着板温的升高而产生变化，比如在低温下眼图质量较好而高温下眼图质量会变差。而眼图的质量又会直接影响到 Serdes 链路的 BER。因此，通常对 Serdes 模块均衡模式产生最大挑战的场景是 Serdes 模块在低温环境下启动并进行自适应，随后将环境温度升高至高温。此时，由于低温时板上插损比较小，因此器件会适应出一个均衡能力较弱的 CTLE 参数。当环境温度升高至高温时，这一组均衡参数就有可能无法满足高温下板上插损，从而产生误码。

下文以 SERDES 方向为 ASIC 向 AFE80 传输为例进行说明，AFE80 向 ASIC 传输 SERDES 原理类似，不进行赘述。在项目研发的阶段，用户常常想要知道能够保证系统 BER 要求的最小 AFE80 回读眼高是多少，并以此为标准进行判断。这样的判断非常快捷，但是需要注意有两点误区：

- 眼高回读值在 AFE80 中是通过 CAPI 函数 getSerdesRxLaneEyeMarginValue() 实现的。眼高的回读值或者准确性是和 BER 密切相关的。getSerdesRxLaneEyeMarginValue() 只是用来作为调试使用，并且只能保证在 BER=10^{-9} 时眼高的准确性。当系统要求高于这个 BER 标准时，使用 AFE80 内部的眼高检测模块是不能完全准确地反应真实眼高数据的。

- 并不是只要眼高满足一定的标准，BER 就能满足系统的要求。BER 还取决于眼图的质量，即眼高和眼宽。

因此通过 CAPI 回读的眼高数据只能给出一个定性的判断，并不能定量。也就是说回读的眼高数据可以作为对不同均衡参数的效果进行判断。而不能够以此时单次回读到的具体眼高数据作为没有误码的依据。对于 AFE80 Serdes 链路能否支持系统 BER 的要求，通常有以下两种方法：

- 方法一：Serdes STX 的 FFE 通过 FIR 滤波器实现，需要通过实验对 FIR 滤波器的各个抽头参数进行遍历找到最优值。 遍历 STX FFE 参数的过程也被称为 Green Box 测试。通过 Green Box 测试选择 Serdes TX 均衡参数，随后将板子进行批量的全温范围循环测试。多次迭代测试后得到能够满足 BER 要求的 SRX 均衡参数以及眼图数据。该眼图数据将作为后续保证 BER 性能的眼图模板。对于 AFE80，由于没有眼宽的数据，可以通过进行一定时间的烤机测试，并在该时间对应的置信区间内没有误码上报时的眼高数据作为眼高的判断标准。这种方式得到的眼高数据仍然是 AFE80 的回读值，但是由于已经进行了小批量板级测试的验证，可以作为判断依据。这种方法需要多次迭代批量测试，耗时较长。

- 方法二：通过 Green Box 测试选择 Serdes TX 均衡参数，随后对每一条 Serdes lane 分别进行浴盆曲线测试。由于 Serdes lane 的眼图质量和温度相关，也可以选择其中一条在高温下插损最大，眼图质量最差的 lane 进行浴盆曲线测试以节省时间。如果该 Serdes lane 可以满足 BER 的要求，那么其他 Serdes lane 均可满足。对于 AFE80，可以将板子放在高温环境下，通过眼高回读找到眼高最差的 Serdes lane，随后运行浴盆曲线测试得到在系统 BER 要求小的最小眼高数据。只要保证其他 Serdes lane 的眼高大于该最小眼高数据，就可以保证系统能够满足 BER 的要求。之后再结合实际批量测试的结果进行验证。这种方法能够节省多次迭代参数进行批量验证的时间。在得到能够满足的值后再通过批量实验进行验证。

AFE80 能够支持通过 CAPI 的方式进行浴盆曲线测试减少用户的验证时间，并且能够得出多大的 Eye margin 能够满足系统 BER 的要求的结论。具体操作如下：

- ASIC 侧遍历 STX 的均衡参数，并通过 getSerdesRxLaneEyeMarginValue 回读 Eye Margin。确保找到的 STX 均衡参数是最优的。虽然 AFE 的 Eye magin 检测精度有限，但是仍然能够用来判断不同 lane 之间眼高的相对值。

- 由于链路插损在低温下较低而随着温度升高会恶化，因此对于 Serdes SRX lane 来说最差的情况是板子在低温启动得到一组均衡参数，保持 SRX lane 均衡参数不变并升温到高温环境。通常 PCB 的插损在低温下较小，高温下较大。因此低温下得到的均衡参数补偿能力较差，当板子升到高温时眼图的质量就会降低，反映到 AFE80 侧的状态是 Eye margin 降低。如果一条 lane 在低温下 Eye margin 相对较小时，高温时同样相对较小。为了找到眼高最差的 Serdes lane，用户可以在低温下启动 AFE80，通过回读每一条 lane 的 Eye margin 数据。随后在低温环境下多次重置 SRX lane 的自适应参数，回读所有 SRX lane 的 Eye margin。重复上述的步骤直到能够稳定看到其中一条 SRX lane 的相对 Eye margin 最小。

- 此时保持均衡参数不变将温度升至高温。在高温环境下对该 SRX lane 进行浴盆曲线测试判断该 Serdes lane 的 Eye margin 是否能够满足系统 BER 的要求。如果能够达到系统要求，记下当前 Eye Margin 的回读值。

- 由于浴盆曲线测试是一种预测手段，用户后续还可以进行小批量全温测试去验证浴盆曲线测试得到的能够满足系统 BER 的最小 Eye margin 值是否有效。如果两者实验结果相同，即验证了上一步得到 Eye margin 回读值可以作为后续判断 Serdes lane Eye margin 的相对标准，节省后续项目验证时间。

AFE80 运行浴盆曲线测试的具体操作流程如下：

- 通过上述步骤找到要进行浴盆曲线测试的 SRX lane。

- 运行 CAPI 函数 “getSerdesBathTub(afeInst2,LaneNo)” 得到某一条 SRX lane 的 BER 数组和范围，如下所示。

```
Eye Capture Progress(percentage): 63
Eye Capture Progress(percentage): 65
Eye Capture Progress(percentage): 100
SerDes BER for Lane 0 complete. Extent: 391
MPSSE mode set
([65, 65, 66, 70, 72, 73, 76, 77, 80, 82, 86, 85, 92, 94, 94, 99, 100, 103, 105, 107, 114, 113, 117, 118, 125, 123, 133, 129, 132, 142, 139, 151, 151, 255, 151, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 146, 151,
255, 255, 146, 140, 136, 137, 130, 126, 123, 119, 120, 113, 113, 109, 105, 105, 101, 101, 97, 98, 93, 92, 89, 85, 85, 81, 80, 78, 76, 76, 72, 70, 66, 64, 64], 391)
#-------
```

- 将上述得到的数组和范围传入附录中的 Python 脚本绘制出浴盆曲线，得到如下浴盆曲线图。

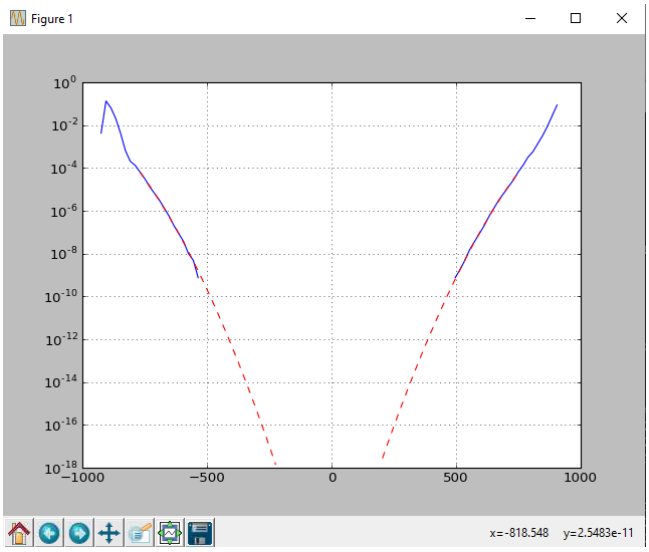

Figure 2. AFE80 浴盆曲线运行结果

AFE80 内的浴盆曲线中蓝色部分是实测得到的数据，AFE80 serdes 可以在一段时间内收集数据来基于一定置信区间统计出 BER=10^{-9}。随后经过 Python 脚本的对数据进行预测，红色部分是由算法预测得到的数据。需要注意的是 AFE80 的浴盆曲线测试和传统的 Serdes 浴盆曲线测试不同，只要当左右两侧的红色曲线在 Y 轴显示的 BER 处没有达到 X 轴的 0，就说明此时的 SRX lane 能够满足当前 Y 轴显示的 BER 的要求。

3 优化 JESD204C RBD

本章节重点讲述如何通过设置好的 RBD 值保证整个 JESD204C 链路的确定性延迟。本章节将以下行方向 AFE80 侧的 JESD204C 链路为例进行介绍，但是设定好的 RBD 的思路在上行方向 JESD204C 链路同样适用。

通常用户为了选择一个好的 RBD 值，需要去扫描整个 RBD 的范围得到一组链路延迟的结果，之后在不会发生链路延迟跳变的 RBD 值中，选择其中的中位数作为系统的 RBD 值。这样的做法能够保证 RBD 值有最大的余量，但是比较费时费力。AFE80 内部通过 SYSREF 去将所有的上行和下行方向 JESD204C link 的 LEMC 边沿进行锁存对齐。上行/下行 JESD204C 方向每一个 Link 含有一个 LEMC counter，将每一个 LEMC 的周期均匀的分成 32*E 段。在一个周期中 counter 值从 0 到 32*E-1 就像尺子一样对每一条抵达的 SRX/STX lane 的到达/发送时刻进行标记，并且可以从中计算出不同 Lane 之间的偏差。

对于下行方向，当同一个方向的 Lane 有了统一的标尺后，用户就可以通过回读不同 Lane 的到达信息从而计算出下行 JESD204C 方向的最优的 RBD 参数。对于上行方向，当用户无法调整接收侧 RBD 时，AFE80 也支持通过调节 LEMC 到数据发出时刻的偏差来实现确定性延迟。

3.1 优化 RBD 的原理

AFE80 中使用 LEMC 作为衡量不同 Serdes Lane 到达时刻的标尺。以 Lane rate=24.33Gbps，E=1 的 JESD204C 系统为例，LEMC 的频率为：

LEMC clock frequency=24.33/66/32/1 GHz=11.52MHz

为了补偿 JESD204C 中不同 Lane 之间的偏差，JESD204C 接收侧含有一个 Buffer 可以将这些偏差吸收掉，并在同一时刻释放所有 Lane，从而保证数据对齐，如下图 3 所示。Buffer 的释放点由 RBD 决定。选择一个合适的 RBD 值能够保证系统在多次上电周期和温度周期内保证 JESD204C 的确定性延迟不变，从而保证系统的下行和下行-反馈 DPD 环路的时延不变。

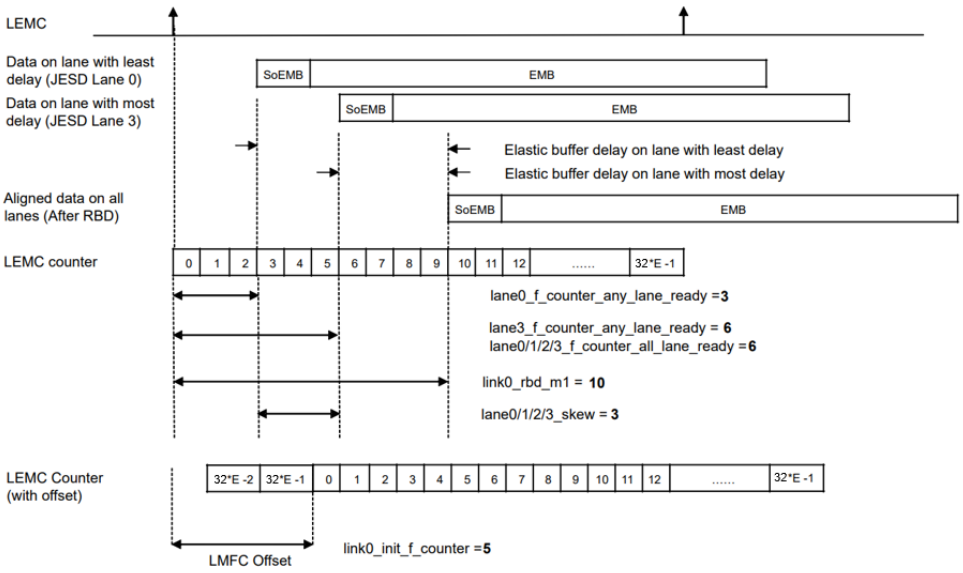

Figure 3. AFE80 JESD204C RBD 和 LEMC Counter 关系图

AFE80 中的 JESD204C 模块运行时序是基于 LEMC counter 的计数器架构。结合上面示例和图 3 帮助读者理解 counter 的工作原理：

1. LEMC counter 以 JESD204C Lanerate/66 的频率进行工作，并且在一个 LEMC 的周期内计数器从 0 计数到 31 工作。

2. 根据 JESD204C 的定义 LEMC 的工作频率是 Lanerate/(66*32*E), 所以一个 LEMC 的周期内 counter 的最大值是 32*E-1。

3. 默认情况下，LEMC counter 的起始点 0 和 LEMC 对齐。 但是 AFE80 同时也支持在 Counter 的起始计数值上加入一个偏置让最早到达和最晚到达的 Lane 处于一个 LEMC 中，这样可以防止产生 RBD 相关的告警。具体的方式是通过寄存器 link0/1_init_f_counter 设置偏置或者通过配置中的系统参数 jesdRxInitLmfcCounter 在器件配置文件中进行修改。例如，当 sysParams.jesdRxInitLmfcCounter = 5 时，整个 LEMC counter 的计数值会发生 5 个单位的偏移，即 LEMC 的边沿与 32*E-5 的 counter 值对齐，如图 3 所示。

下面以图 3 为例，解释在没有 counter 偏置时的情况下寻找最优 RBD 值的原理：

1. 回读寄存器 lane0/1/2/3_f_counter_any_lane_ready 得到每一条 lane 到达时对应的 counter 值。Counter 值返回的是每一条 Lane 中 SoEMB(Start of EMB)到达的时刻。
    - lane0_f_counter_any_lane_ready=3，lane3_f_counter_any_lane_ready=6。
2. 找出最晚到达的 Lane。这个值可以从 lane0/1/2/3_f_counter_all_lane_ready 回读得到。该值和上一步中寄存器回读值中的最大值相同。
    - lane3_f_counter_any_lane_ready= lane0/1/2/3_f_counter_all_lane_ready=6
3. lane0/1/2/3_skew 即为最早到达 Lane 和最晚到达 Lane 之间的差别。在本例中，Lane0 和 Lane3 的差别为 3。

- lane0/1/2/3_skew=3

4. 通过上面几步得到的数据，可以知道不同 Lane 的到达情况。通常为了补偿温度和工艺的偏差，RBD 值设置为最晚到达 Lane 的 counter 值加上 4 个 counter 值的裕量。同时由于 counter 值是在 0~32*E-1 之间循环的，所以引入了取余的符号，当最后一条 lane 到达的时刻接近最大值时，可以在下一个 LEMC 周期开始时同时释放。

   - RBD = (lane0/1/2/3_f_counter_all_lane_ready + 4)%(32*E)

5. 同时 RBD 还需要满足以下两个条件：

   - lane0/1/2/3_f_counter_all_lane_ready < RBD
   - RBD < min(lane0/1/2/3_f_counter_any_lane_ready) + BufferDepth – 10

   其中，第二个约束条件原因为：AFE80 Buffer depth 默认是 24 个 counter 单位。当两条 Lane 之间的时间偏差太大，有可能导致 Buffer 的储存深度不够，造成 Buffer 溢出错误。因此限定了 RBD 的最大范围。Buffer 的深度可以通过 link0/1_buffer_depth 寄存器进行配置，该寄存器可以设置为 0~23 的任意数，对应的 buffer depth 为 1~24。

6. 在写入新的 RBD 值后，需要回读 link0_sysref_cnt_on_release_opportunity 寄存器的值。如果回读值为 0，则说明 RBD 设置正确，否则需要带入下式计算出新的 RBD 值。

   - RBD=(link0_rbd_m1+link0_sysref_cnt_on_release_opportunity)%(32*E)

   这是因为在 AFE80 内部 JESD IP 和 DUC IP 是运行在两个时钟域上的，为了数据能够在两个时钟域上正确的交换，在两个 IP 核之间加入了一个 FIFO 模块。这个 FIFO 的读写指针会在 SYSREF 的上升沿重置，并且需要与 LEMC 对齐以实现确定性延迟。因此当寄存器 link0_sysref_cnt_on_release_opportunity 回读为 0，说明已经对齐，否则没有对齐，需要按照上面式子计算出新的 RBD 值。

3.2 设置 AFE80 JESD204C 的最优 RBD 值

3.2.1 寄存器设置方式

将上文所描述的寄存器和地址汇总如下表所示，请注意，下表中的内容都是 DAC JESD 页内寄存器：

Register Name	R/W	Address
DAC_JESD	R/W	0x16 [3:2]
lane0_f_counter_any_lane_ready	R	0x184[7:0], 0x185[7:0]
lane1_f_counter_any_lane_ready	R	0x186[7:0], 0x187[7:0]
lane2_f_counter_any_lane_ready	R	0x188[7:0], 0x189[7:0]
lane3_f_counter_any_lane_ready	R	0x18A[7:0], 0x18B[7:0]
lane0_f_counter_all_lane_ready	R	0x18C[7:0], 0x18D[7:0]
lane1_f_counter_all_lane_ready	R	0x18E[7:0], 0x18F[7:0]
lane2_f_counter_all_lane_ready	R	0x190[7:0], 0x191[7:0]
lane3_f_counter_all_lane_ready	R	0x192[7:0], 0x193[7:0]

lane0_skew	R	0x234[4:0]
lane1_skew	R	0x235[4:0]
lane2_skew	R	0x236[4:0]
lane3_skew	R	0x237[4:0]
link0_rbd_m1	R/W	0xDC[7:0], 0xDD[7:0]
link1_rbd_m1	R/W	0xDE[7:0], 0xDF[7:0]
link0_init_f_counter	R/W	0xE4[7:0], 0xE5[7:0]
link1_init_f_counter	R/W	0xE6[7:0], 0xE7[7:0]

将上文中选择最优 RBD 的步骤总结如下，以 link0 为例：

Step-1	首先选择一个 RBD 的值写入 link0_rbd_m1 寄存器或 Bring up 脚本中，初始化 AFE80。
Step-2	回读最后一条到达 Lane 的 counter 值 lane0_f_counter_all_lanes_ready
Step-3	将 Step-2 中的值带入下式进行计算，将先的 RBD 值写入 AFE80 并重新 Bring up。 ----modulo((lane0_f_counter_all_lanes_ready ' + 4), 32*E)
Step-4	回读 link0_sysref_cnt_on_release_opportunity 寄存器，如果回读为 0 则说明当前 RBD 设置合适。否则转到 Step-5。
Step-5	重新按照下面公式计算出先的 RBD 值并重新写入 AFE80 进行 Bring up。 ----modulo(('link0_rbd_m1 from Step-3' + link0_sysref_cnt_on_release_opportunity), 32*E)
Step-6	继续回读寄存器'link0_sysref_cnt_on_release_opportunity', 此时应当回读为 0。 将得到的新的 RBD 值写入到 AFE80 的 Bring up 脚本中。

3.2.2 CAPI 设置方式

为了用户方便，AFE80 还支持用运行 CAPI 的方式去计算出最优的 RBD 值。具体步骤如下：

Steps	CAPI	Description
Step-1	getAllLaneReady	回读最晚到达的 Lane 的 counter 值。
Step-2	checkIfRbdIsGood	检查当前配置的 RBD 值是否合适，如果不合适则跳到下一步寻找最优 RBD 范围。
Step-3	getGoodRbdRange	运行 CAPI 找到合适的 RBD 范围，得到最大和最小允许的 RBD 值。
Step-4	setManualRbd/ setGoodRDB	设置 Step2 中得到的 RBD 值。 或者直接运行 setGoodRBD 将自动设置最优 RBD 值。
Step-5	checkIfRbdIsGood	打开 SYSREF 进行 Relink，检查该 RBD 值是否合适。

下图为 CAPI 运行结果，可以看到这个场景下 RBD 应该设置为 16。

```
linkNo = 0, rxLatency = 11
linkNo = 0, rbdMin = 12, rbdMax = 24
linkNo = 0, rbdStatus = 1
```

Figure 4. 特定 case 的 CAPI 返回值

3.3 灵活使用 LEMC 偏置简化系统设计

如上文所述，AFE80 JESD204C 接收和发送模块都支持修改内部 counter=0 到 LEMC 边界的偏置大小。这个参数由系统参数 jesdRxInitLmfcCounter，rxJesdTxLmfcCounter 和 fbJesdTxLmfcCounter 决定。每一个偏置单位对应时间与 counter 的单位相同 Lanerate/66, 系统参数可以取值 0~32*E-1。

下面举例说明如何使用 LEMC 偏置优化系统的 RBD 设计.如图 5 所示在 JESD204C 的接收侧，当没有 LMEC 偏置时，第一条达到的 lane 对应 30，最后一条 lane 到达时刻对应 2，RBD 设定为 6 以保证确定性延迟。当板上最短和最长 trace 在全温范围内变化时，通常我们建议用户将不同 Lane 的数据到达时刻的 counter 值的数量级尽可能接近，否则可能会引起 JESD RBD 告警。 此时我们可以使用 sysParams.jesdRxInitLmfcCounter=7 或者 link0/1_init_f_counter=0x07，这样当 LEMC 上升沿到来的时候，counter 会从 7 开始计数，因此第一条 lane 到达时刻为 5 和最后一条 lane 到达时刻为 9。此时要保证和之前相同的确定性延迟，需要让 RBD 设定为 13。

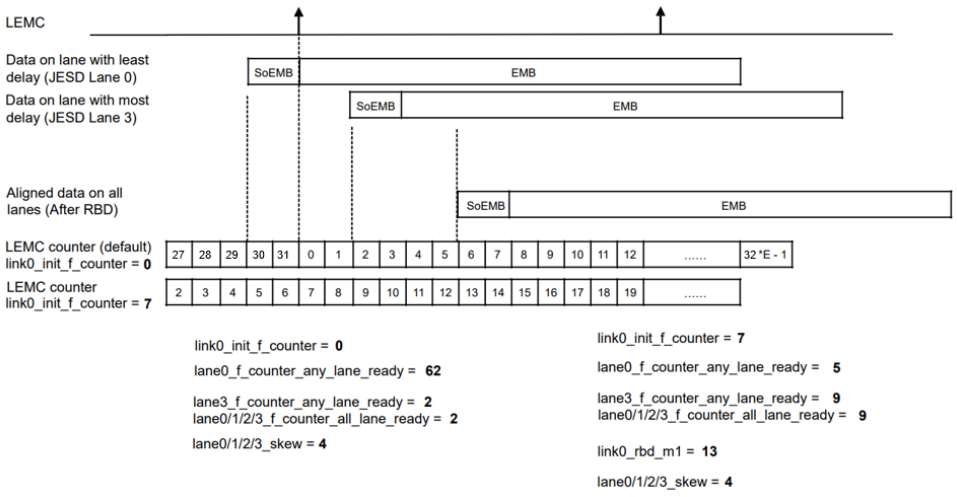

Figure 5. JESD204C 下行 LEMC 偏置优化链路稳定性

在 JESD204C 的发送方向，通常 RBD 的设置在 ASIC 端，但有的 ASIC RBD 的调整范围和分辨率有限，为了保证确定性延迟，可以通过调整数据的发送时间。AFE80 支持修改数据的发送时间。可以通过系统参数 sysParams.rxJesdTxLmfcCounter 和 sysParams.fbJesdTxLmfcCounter 去调整 RX 和 FB JESD204C Link 数据的发送时间以实现确定性延迟。该参数的含义是保持 LEMC 边沿不变，移动数据使得第设定值个 Multi-Block 和 LMEC 对齐。此时相当于 SoEMB 提前于 LEMC 设定值个偏置时间发送出去。

下图 6 为设置 rxJesdTxLmfcCounter=2 和 0 时不同 Lane 数据的对应情况。假设此时 E=1，当设置 rxJesdTxLmfcCounter=2 时，第 2 个 Block 会和 LEMC 对齐，也就是说 EoEMB 会提前 2 个 counter 时间发送出去。此时在 ASIC 侧就会看到数据提前到达，在 RBD 不变的情况下，RBD 仍然能够补偿到所有的延迟，此时数据在 ASIC 的释放时间不变。 但是由于数据是提前发出的，看整个 Link 的确定性延迟会增加。

如果 rxJesdTxLmfcCounter 向小的方向变化，数据将会更晚发出，此时如果 RBD 能保证最长和最短 Lane 的数据能够到达，那么整个 Link 的 latency 会缩短，如果 RBD 不能够保证，需要用户增加 ASIC 侧的 RBD 值以保证确定性延迟。

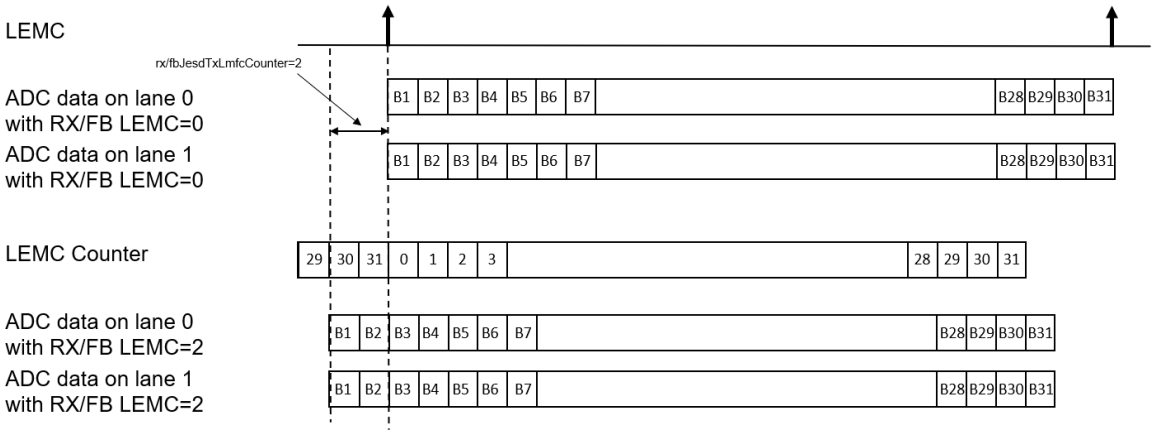

Figure 6. JESD204C 上行 LEMC 偏置优化链路稳定性

4 结论

本文介绍了优化 Serdes 眼图的方法，如何正确的看待 Eye margin 回读值的方法。随后介绍了 AFE80 浴盆曲线的使用方法，以帮助用户缩短验证 Serdes 链路稳定性的时间。确保了 Serdes 链路稳定性的前提下，选择合适的 RBD 值对保证 JESD204C 链路的稳定性至关重要。因此又介绍了如何选择 AFE80 合适的 RBD 值以及相关 CAPI 和寄存器。最后又介绍了巧用 LEMC 偏置设置简化 JESD204C 的系统设计。希望本系列文章能够帮助读者加深对 AFE80 JESD204C 部分的理解。

5 参考文献

1. User Guide “AFE80xx Configuration Guide”
2. 数据手册“AFE8030 Octal-Channel RF Transceiver with Feedback Paths”
3. TI 应用手册“Green-box testing: A method for optimizing high-speed serial links”

附录 1： 绘制浴盆曲线的 Python 脚本

```
import matplotlib.pyplot as plt
from math import sqrt
```

```
from scipy import special
import sys

rdArray=sys.argv[1]
dataTemp=rdArray.split('([')[1].split(']')[0]

dataTemp=dataTemp.split(',')

bathtub_data=[]
for item in dataTemp:
    bathtub_data.append(int(item.strip()))

extent = 0x393

# Convert the bathtub_data to real BER
y = [10**(-((d & 0xF) * (2**-4) + ((d >> 4) & 0xF))) if d != 255 else 0 for d in bathtub_data]

x = [round((d - 0.5) * extent / 47.0, 4) for d in range(-47, 48)]

MAX_SOURCE_ERR = 1e-4
MIN_SOURCE_ERR = 2e-11
MIN_TARGET_ERR = 1e-18

# This function is also implemented in bathtub.cpp
def bathtub_extrapolate(x, y):
    def bathtub_extrapolate_oneside(x0, y0):
        Y2 = 0; Y1 = 0; Y0 = 0
        XY1 = 0; XY0 = 0
        y = [0] * 48
        for i in range(48):
            xx0 = x0[i]
            yy0 = y0[i]

            if yy0 > MAX_SOURCE_ERR:
                continue
            if yy0 < MIN_SOURCE_ERR:
                continue
            yy0 = sqrt(2) * special.erfcinv(2 * yy0)
            Y0 += 1
            XY0 += xx0
            Y1 += yy0
            XY1 += xx0 * yy0
            Y2 += yy0 * yy0

        if Y0 == 0:
            return
        D = Y2 * Y0 - Y1 * Y1
        if D == 0:
            return
```

```
        sigma = (Y0*XY1 - Y1*XY0) / D
        m = (-Y1*XY1 + Y2*XY0) / D
        for i in range(48):
            xx0 = x0[i]
            yy = 0.5*special.erfc((xx0-m) / sqrt(2)/sigma)
            if yy > MAX_SOURCE_ERR:
                continue
            if yy < MIN_TARGET_ERR:
                continue
            y[i] = yy
        return y
    yy1 = bathtub_extrapolate_oneside(x[0:48], y[0:48])
    yy2 = bathtub_extrapolate_oneside(x[47:95], y[47:95])
    yy = yy1 + yy2[1:]
    return yy

# Remove the unreliable data for bathtub extrapolate
y2 = [0 if d<1e-9 else d for d in y]
yy = bathtub_extrapolate(x, y2)

# Plot
plt.semilogy(x, y, 'b-')
plt.semilogy(x, yy, 'r--')
plt.grid(True)
plt.show()
```

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265