Application Report
ZHCA590 – December, 2013

AIF2 编程

冯华亮，冯林

摘要

本文介绍了 AIF2 编程的一些信息，并提供了基于 TI EVM 的 AIF2 示例工程。

目录

1 介绍 4
2 AIF2 配置 7
2.1 KeyStone1 AIF2 SerDes 配置 7
2.2 AIF2 同步的配置 8
2.3 DB 配置 14
2.4 PE/PD 配置 15
2.4.1 通道间的带宽分配 15
2.4.2 无线帧计数器的配置 17
2.4.3 TDD 配置 (仅 Packet DMA 模式支持) 19
2.5 DIO 控制器 配置 19
2.5.1 AIF2 缓冲区的 DIO 配置 20
2.5.2 RAC 接口的 DIO 配置 21
2.5.3 TAC 接口的 DIO 配置 23
2.5.4 DSP 存储器缓冲区的 DIO 配置 24
3 AIF2 的通用数据传输 25
4 AIF2 调试 26
5 示例工程 28
5.1 基于 Packet DMA 的工程 32
5.2 基于 DIO 的工程 36
5.3 在其他板上运行工程 38
参考文献 40

图

图 1. LTE 基带方案中的天线接口 4
图 2. WCDMA 基带方案中的天线接口 5
图 3. AIF2 框图 5
图 4. KeyStone1 AIF2 SerDes 配置 7
图 5. AIF2 Timer 框图 8
图 6. 天线接口同步 10
图 7. 两片 C6670 的同步 11
图 8. AIF2 事件的典型时延 12
图 9. PE OBSAI 传输规则 15
图 10. Dual Bit Map 规则示例 15
图 11. 通道间平均分配带宽的示例 16
图 12. CPRI DBM 规则 17
图 13. 4x CPRI Link 封装 16 个天线流的示例 17
图 14. 无线帧计数器 18
图 15. AIF2 数据缓冲区的 DIO 配置 21
图 16. RAC 天线缓冲区数据结构 21
图 17. RAC DIO 配置 22
图 18. TAC 天线缓冲区数据结构 23

图 19. TAC DIO 配置 24
图 20. 存储器中的天线缓冲区的 DIO 配置 25
图 21. AIF2 EE (Error and Exception) 模块 27
图 22. 通过 BOC 板连接两片 DSP 的 AIF 链路 29
图 23. 内部环回测试 30
图 24. 外部转发环回测试 30
图 25. 示例工程目录结构 31
图 26. 切换到 OBSAI 模式 35

表

表 1. OBSAI v.s. CPRI 6
表 2. 每 link 支持的天线数 6
表 3. 各种 link 速率下的 SerDes 配置 7
表 4. AIF2 同步事件 8
表 5. 时延计算示例 13
表 6. 菊花链连接的时延计算示例 13
表 7. 各种带宽的 Packet DMA 通道建议的 FIFO 大小 14
表 8. 不同情况下规则的使用 16
表 9. 无线帧计数器的单位 18
表 10. DIO 配置参数 20
表 11. AIF2 数据缓冲区的 DIO 配置 21
表 12. RAC DIO 配置 22
表 13. TAC DIO 配置 23
表 14. DSP 存储器中的天线缓冲区的 DIO 配置 24
表 15. 各种情况下的通用数据传输 25
表 16. AIF2 常见问题和可能原因 28
表 17. 示例工程中的源文件 31

1 介绍

天线接口（AIF2）是 Keystone DSP 器件的一个外设，是用于支持基带 DSP 处理器和天线间的 IQ 数据传输的高速 SERDES 接口。同时，用于基站控制和维护的控制命令也可以通过 AIF2 传输。

Keystone DSP 可以支持多个无线通信标准，包括 LTE, WCDMA, TD-SCDMA, Wimax, GSM (仅 OBSAI 模式支持)。

AIF2 为用户提供了多种灵活的配置，在“KeyStone Architecture Antenna Interface 2 (AIF2) User Guide” 这篇文档中对这些配置有详细描述，而本文旨在提供一些 AIF2 编程的补充信息。

对于 WCDMA, AIF2 和 RAC/TAC 之间的数据传输采用 DIO 模式，这是一种类似于 EDMA 的传输模式。DIO 为 WCDMA 处理而做了特别的优化，相对于 Packet DMA，需要更少的存储器开销。

与 WCDMA 的“流数据”处理模式不同，对于像 LTE 和 TD-SCDMA 这样的无线标准，更倾向于“块数据”的处理模式, 所以用 Packet DMA 来传输 AIF2 和内部模块之间的数据更合适。为此，Packet DMA 和 QMSS 一起，被称为 Multicore Navigator ，用于控制和实现 DSP 内部块数据的高速传输，这是 Keystone DSP 的一个新特性。此特性减轻了 DSP 内部的通信负担，从总体上提高了系统性能。

本文也提供了用于 LTE 和 WCDMA 的测试工程。

用 AIF2 实现 LTE 和 WCDMA 方案的基本框图如下：

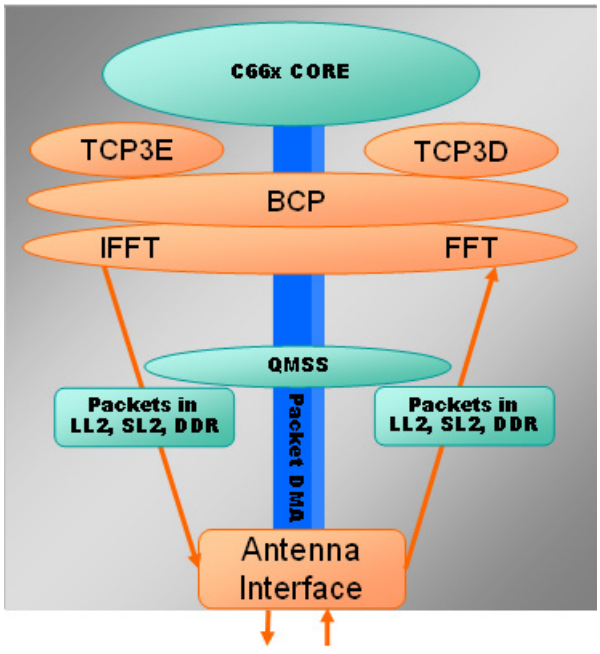

图1. LTE 基带方案中的天线接口

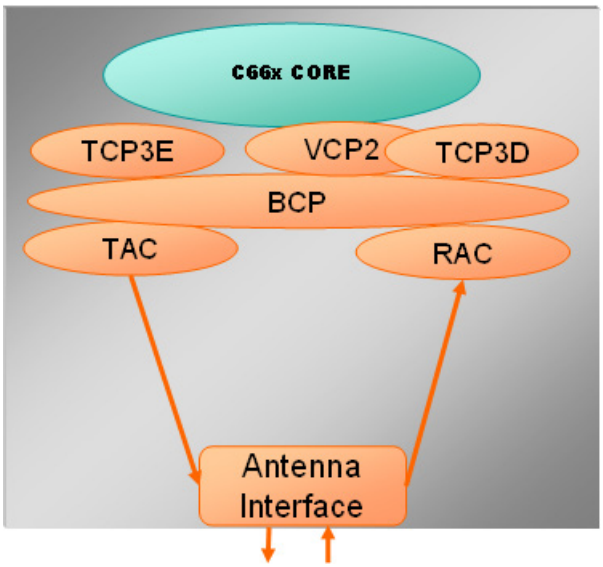

图2. WCDMA 基带方案中的天线接口

AIF2 可支持多条 Serdes 链路（link），例如，TCI6614 AIF2 有 6 条链路。图 3 为 AIF2 的功能框图，对于每个模块的详细信息，请参阅 AIF2 用户手册。

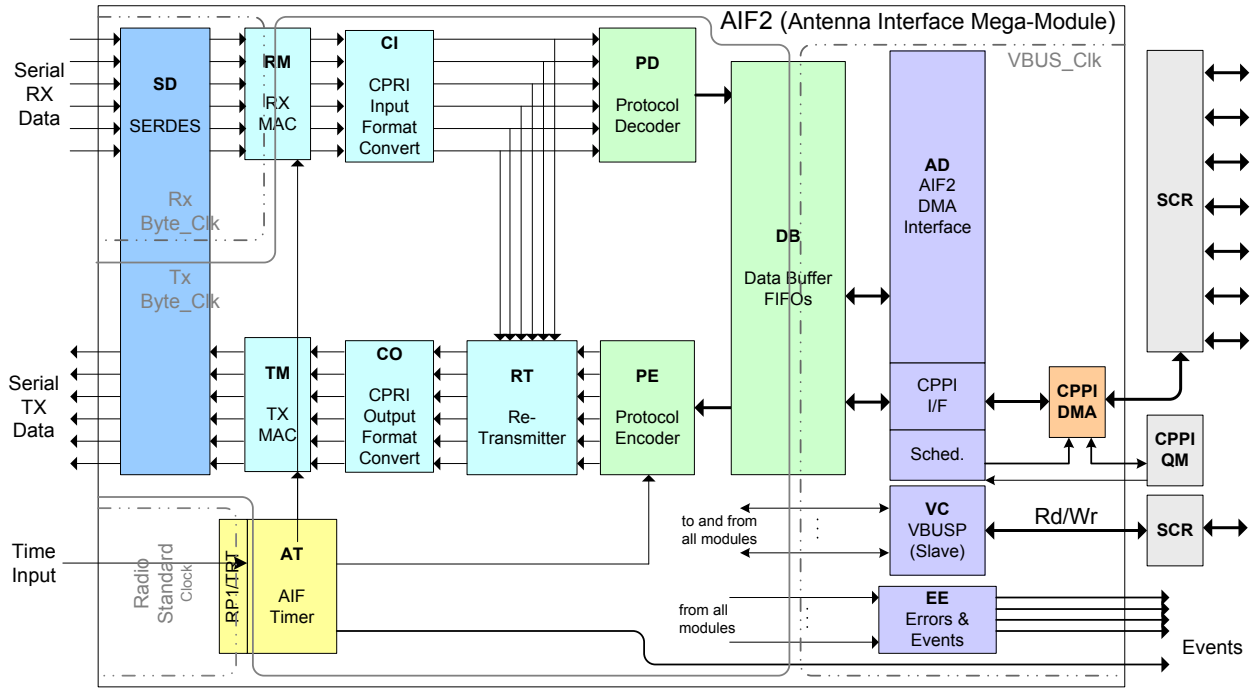

图3. AIF2 框图

AIF2 支持 OBSAI(Open Base Station Architecture Initiative) 和 CPRI (Common Public Radio Interface) 协议。表 1 比较了 OBSAI 和 CPRI 的主要区别。

表1. OBSAI v.s. CPRI

		OBSAI	CPRI
Link rate	2x	1.536Gbps	1.2288Gbps,
	4x	3.072Gbps	2.4576Gbps
	8x	6.144Gbps	4.9152Gbps
Data precision		8 bits, 16 bits	7 bits, 15 bits (support same number of streams as OBSAI) 8 bits, 16 bits (support less streams than OBSAI)
Packet Efficiency for antenna streams		(16/19)*(399/400)*(20/21)= 0.8	(15/16)= 0.9375

一般来说，OBSAI 的灵活性更好。例如，OBSAI 通过配置 DBMR（Dual Bit Map Rule）和 modulo 规则，可以提供比 CPRI 更好的灵活性。而 CPRI，每个链路（link）只有一个 DBMR，没有 modulo 规则。 但 OBSAI 需要更多的开销和更高的链路速度。

CPRI 和 OBSAI 最初都是为 WCDMA 设计的，所以链路速率都是 WCDMA 基带采样率的整数倍。

LTE（5M/10M/20M）的采样率是 WCDMA 的整数倍，而 WCDMA 和 TD-SCDMA 的采样率也是倍数关系，所以这些无线标准都能很好地匹配 OBSAI 和 CPRI（不需要配置 DBM (Dual Bit Map))。表 2 总结了在各种协议下，AIF2 每个链路能配置的天线流数目。

表2. 每 link 支持的天线数

Radio Standards	Sample Rate	AxC per Link (Typical case)		
		2x	4x	8x
TD-SCDMA	1.28MHz	24	48	96
WCDMA	3.84MHz	8	16	32
LTE 5MHz	7.68MHz	4	8	16
LTE 10MHz	15.36MHz	2	4	8
LTE 20MHz	30.72MHz	1	2	4

注意，AIF2 每个链路能支持的最大天线流数目为 128。

GSM, Wimax 和 15MHz LTE 的采样率不是 WCDMA 的整数倍，需要设置 DBM (Dual Bit Map) 进行速率匹配。

2 AIF2 配置

本节提供了一些配置 AIF2 模块的信息。

2.1 KeyStone1 AIF2 SerDes 配置

KeyStone1 系列 DSP 的 AIF2 SerDes 需要配置以达到期望的链路速率。图 4 为输入参考时钟和输出时钟的关系。

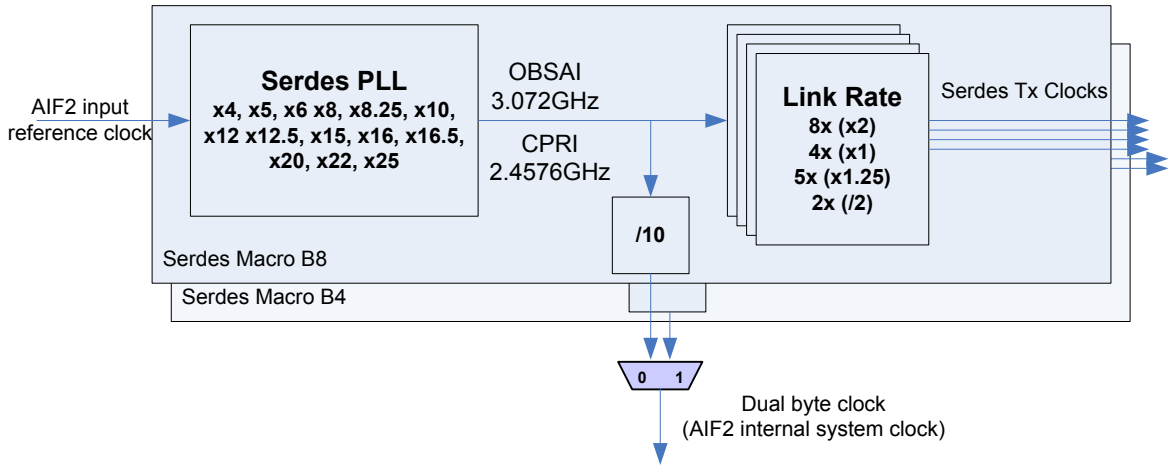

图4. KeyStone1 AIF2 SerDes 配置

建议的输入参考时钟为 122.88MHz ～ 800MHz, SerDes PLL 的倍频因子(multiply factor)用来产生 OBSAI 的 3.072GHz 或 CPRI 的 2.4576GHz。所以 PLL 倍频因子的计算公式为：

(PLL multiply factor) = (3.072GHz or 2.4576GHz) / (input clock speed)

最终的链路速率是按照链路的速率配置从 3.072GHz (OBSAI) 或 2.4576GHz (CPRI) 得到的。而且每个链路可以配置成不同的链路速率。表 3 总结了 TCI6614 EVM 评估板 (122.88MHz 输入参考时钟)PLL 倍频因子和链路速率的关系。

表3. 各种 link 速率下的 SerDes 配置

Standard	PLL x	Link Rate	SerDes Clock Speed
OBSAI	25	8x	122.88*25*2= 6.144Gbps
		4x	122.88*25*1= 3.072Gbps
		2x	122.88*25/2= 1.536Gbps
CPRI	20	8x	122.88*20*2= 4.9152Gbps
		4x	122.88*20*1= 2.4576Gbps

		2x	122.88*20/2= 1.2288Gbps

注意，双字节时钟(dual byte clock (TM, AT, PE, PD…的内部时钟))是 307.2MHz (OBSAI)或 245.76MHz(CPRI)。

KeyStone 2 系列 DSP 的 AIF2 SerDes 由现成的 CSL 库函数配置，不需要用户自己编程配置。

2.2 AIF2 同步的配置

天线数据需要严格的时间同步。AIF2 的天线帧同步由 AIF2 Timer(AT)模块控制。AT 模块运行在双字节时钟的频率上（307.2MHz (OBSAI)或 245.76MHz(CPRI)），并且与外部的系统帧同步信号对齐。 下图为 AT 模块的功能框图。

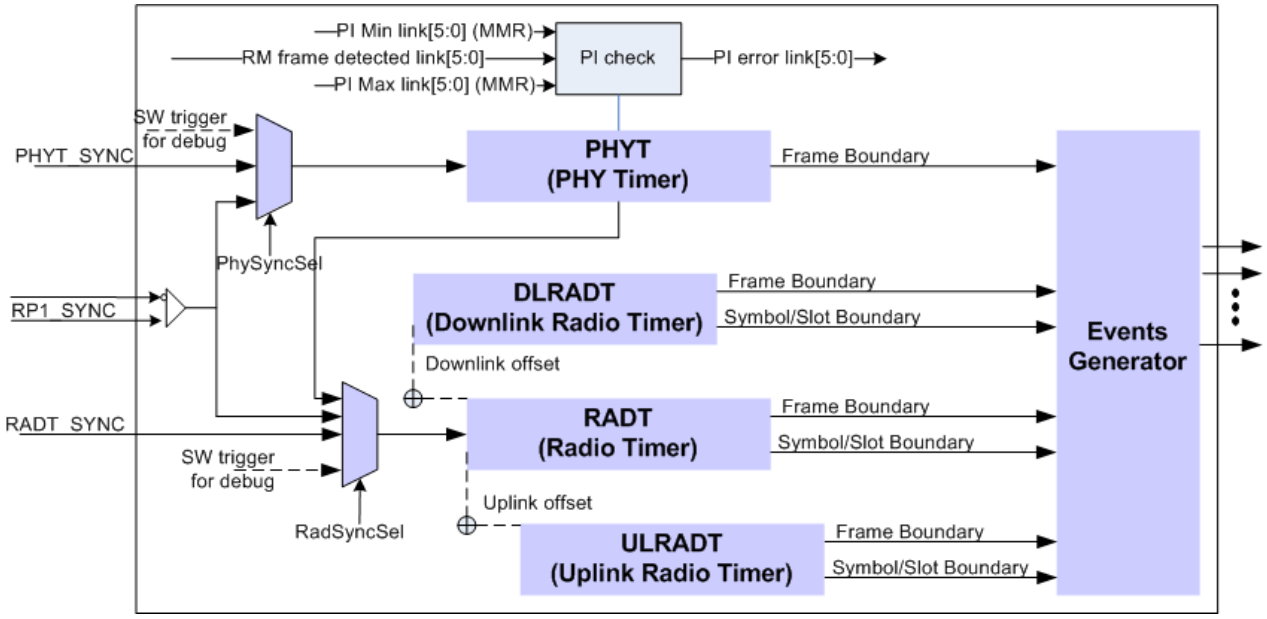

图5. AIF2 Timer 框图

PHYT 为 AIF2 的内部时序控制产生相应的事件，包括 TM delta， RM Pi， PE1 事件 和 PE2 事件。对应于 OBSAI 和 CPRI 帧结构，PHYT 应该设置为 10ms 周期，事件偏移以 PHYT 的帧边界为参考点。

RADT 支持各种协议的帧结构，为 AIF2 的其他模块产生各种事件以方便实现不同的标准和应用。RADT 也产生事件来触发 DIO 进行数据传输。这些事件的周期和偏移都是可编程的。

PHYT 和 RADT 在开始计数前都必须被软件“使能”（armed）并接收到同步脉冲。

表 4 总结了 KeyStone DSP 器件 AIF2 timer 产生的事件。

表4. AIF2 同步事件

Events	Typical Period

Event 0~7 to DSP cores and EDMA Event 13~23 (KeyStone 2 Only)	Frame, Slot or symbol according to application's requirements
Event 8 for TAC	4 chips
Event 9 for RAC_A Event 10 for RAC_B Event 11 for RAC_C (KeyStone 2 Only) Event 12 for RAC_D (KeyStone 2 Only)	32 chips
6 DIO events for AIF2	4~32 chips
Delta, PE1, PE2 and Pi for AIF2	10ms

图 6 为天线接口同步的框图。

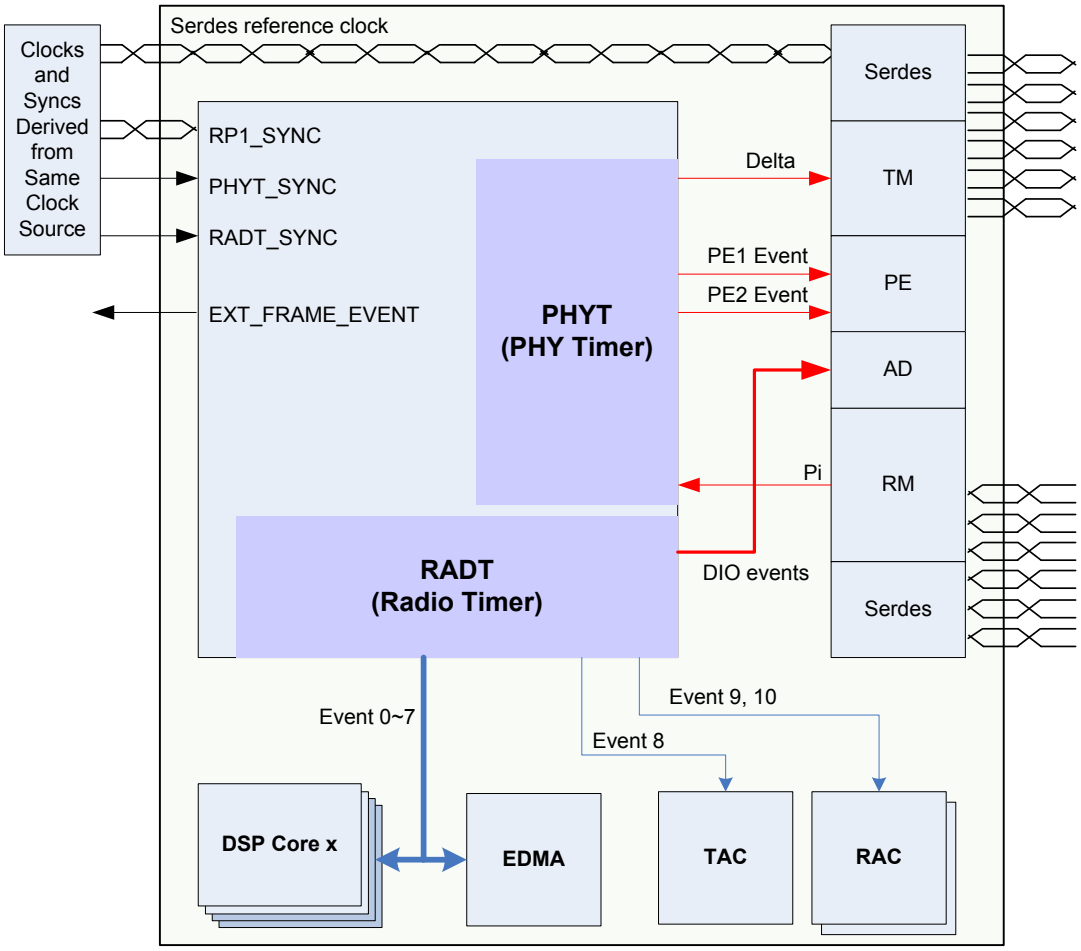

图6. 天线接口同步

若在两个器件之间传输天线数据，它们应采用相同的时钟源。下图为两片 TMS320C6670 DSP 实现同步的示例。

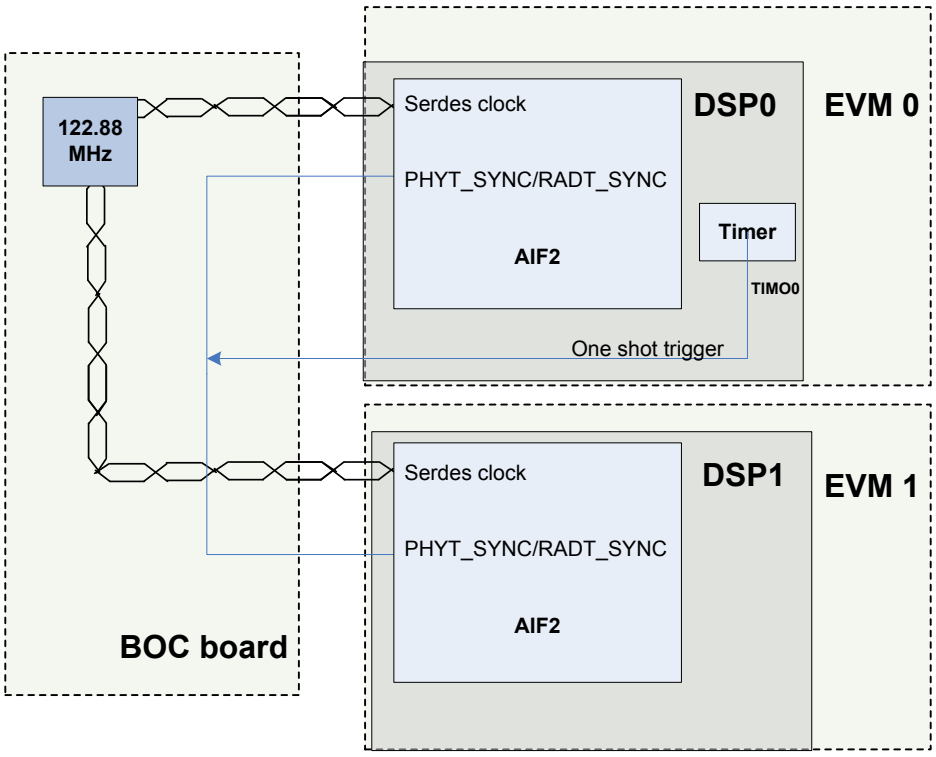

图7. 两片 C6670 的同步

在 6670 EVM0 上，DSP 的时钟输出引脚 TIMO0 通过 BOC (Break Out Card, http://processors.wiki.ti.com/images/f/fd/LC_DUAL_EVM_BoC_r0p4.pdf) 板被连接到两片 DSP 的帧同步引脚, (PHYT_SYNC and RADT_SYNC) 。

例子中，DSP0 timer 被配置为产生 shot pulse 来同时触发 DSP0 和 DSP1 的 AIF2。两片 DSP 的 AIF2 被配置为“run freely”模式，所以不需要外部的周期帧同步信号，这种配置方式只适用于在 EVM 板上的测试。实际系统中 ，应该为 AIF2 提供周期的帧同步信号，如果 AIF2 内部计数产生的帧边界信号与外部周期的帧边界不同步，则会产生错误中断，用户可以使能这个中断，并在中断服务程序中做相应的处理，比如重配 AIF2。

因为基站是一个时间敏感的系统，所以 AIF2 事件的时序配置非常重要。图 8 列出了 AIF2 典型事件的内部时延。

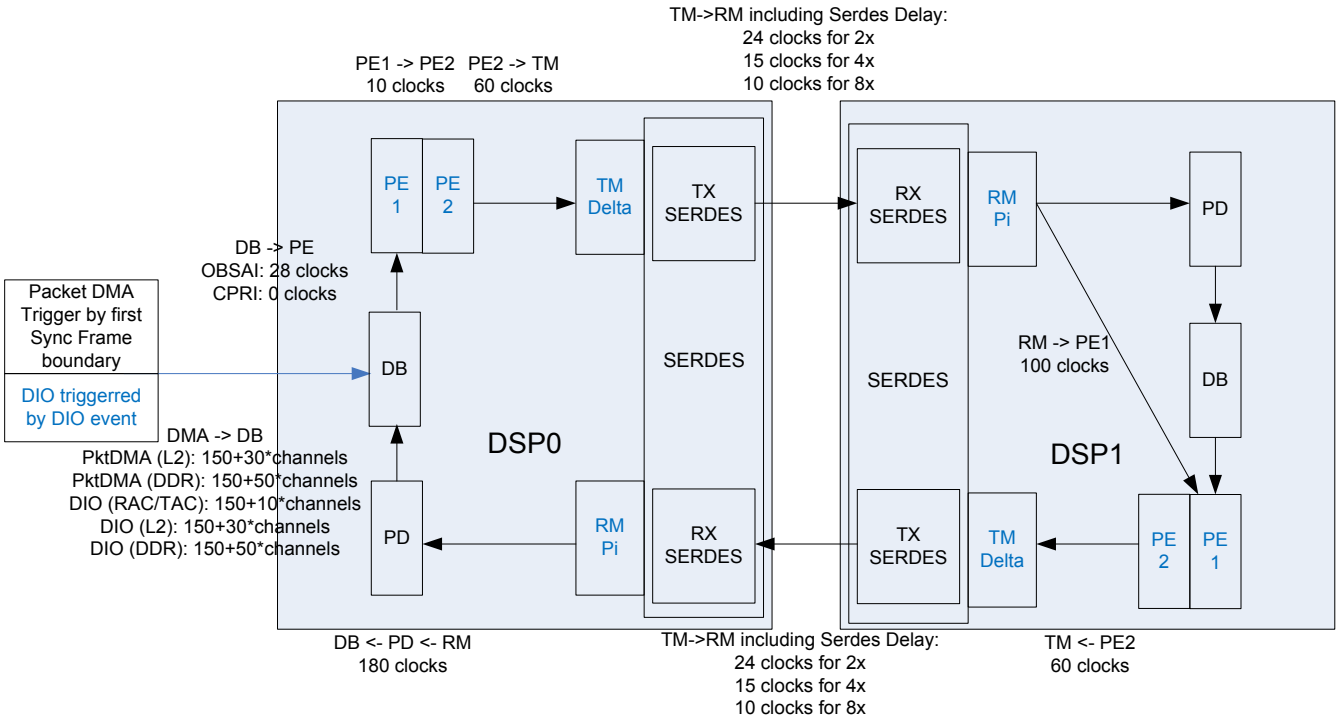

图8. AIF2 事件的典型时延

上图中的时延值，是以双字节时钟为单位。

注意，上图中 TM->RM 的时延没有考虑器件间的传播延迟。这对于内部环回测试（loopback）或者同一个板上的两个器件间的测试是可以的。而对于比较远距离的传输，额外的传播延迟需要被考虑。

对于天线数据，Packet DMA 在 TM/RM 进入同步状态后的第一个帧边界开始填充 DB。因此，天线数据应该在帧边界到来之前就已经被压入 Packet DMA 的发送队列。

如果是 DIO 模式，DIO 事件应该被正确地配置以触发 DIO 进行数据传输。

所有的天线数据应该在 PE1 事件被触发之前到达 DB，DMA 填充 DB FIFO 的延迟取决于通道数目。通道数越多，延迟越大。另外，如果有总线竞争的情况，延迟会更大。因为 AIF2 是时间敏感的，所以在 DSP 内部总线系统中，建议将 AIF2 的 Packet DMA 设置为最高优先级。在应用中，一般把 PE 触发事件的偏移设得稍大些，让 DMA 有更充裕的时间填充 DB。而 PE 的延迟越大，整个系统的延迟也越大，所以要结合系统的时间余量来综合考虑 PE 触发事件的偏移量设置。

如果下行 DB buffer 为“满”状态，DB 会有反馈信号通知 Packet DMA，因此我们不需要担心 Packet DMA 比 PE 启动得早而将数据覆盖。

然而，在 DIO 模式下，DB 没有反馈信号给 DIO，所以，DIO 不能启动得太早，否则，在 PE 读取之前，DB 中的数据会被覆盖。在下行方向，AIF2 事件通常被设置为 DB 在“半满”状态时启动 PE。上行方向当 DB 为“半满”状态时启动 DIO。

注意，无论天线数据是通过 DIO 还是 Packet DMA 传输，OBSAI 控制消息或者 CPRI 控制字传输的通用数据总是用 Packet DMA。通用数据通道不像天线数据流通道，当描述符被压入 TX 发送队列后 Packet DMA 会立即发送包数据。所以，包含控制信息的通用数据必须在 TM/RM 进入同步状态的第一个帧边界后，才压入 TX 发送队列。

事件的偏移值是该事件之前所有延迟的累加值。以图 8 为例，表 5 列出了如何计算各个事件偏移值的方法（假设 Serdes 的速率为 8x，DMA 填充 DB 的时间为 1500 clocks，CPRI 模式）。

表5. 时延计算示例

Event	Offset (dual-byte clocks)	
	Calculation	Result
PE1	1500	1500
PE2	1500+10	1510
TM Delta	1500+10+60	1570
RM Pi	1500+10+60+10	1580

如果多个 DSP 以菊花链的方式连接，事件偏移值应该随链结构累加。表 6 列出了在 C6670EVM 评估板上进行转发测试的事件偏移值。

表6. 菊花链连接的时延计算示例

Event	Offset (dual-byte clocks)	
	Calculation	Result
PE1 on first TX node	1500	1500
PE2 on first TX node	1500+10	1510
TM Delta on first TX node	1500+10+60	1570
RM Pi on first RX node	1500+10+60+10	1580
PE1 on second TX node	1500+10+60+10+100	1680
PE2 on second TX node	1500+10+60+10+100+10	1690
TM Delta on second TX node	1500+10+60+10+100+10+60	1750
RM Pi on second RX node	1500+10+60+10+100+10+60+10	1760
…	…	…

2.3 DB 配置

上行方向，所有通道的数据 buffer RAM 是 16KB，下行方向也有 16KB 的数据 buffer，并且这 16KB 的 buffer 可以灵活地分配给各个通道。每个通道的 FIFO/Buffer 按照下面的参数分配：

- 起始地址：实际起始地址=128*起始地址的配置值
- Packet DMA 的 FIFO 深度 (每个通道可以配置为不同的大小)： 实际大小=128 << 配置值
- DIO buffer 大小 (128 bytes 或 256 bytes, 所有 DIO 通道相同大小)

如果只有几个 Packet DMA 通道，则这些通道的 FIFO 应该尽可能地设置得大一些，否则每个通道应该按照带宽设置 FIFO 大小。表 7 给出了不同 Packet DMA 通道带宽下，建议的最小 FIFO 值。

表7. 各种带宽的 Packet DMA 通道建议的 FIFO 大小

Equivalent Bandwidth of channel	Value for FIFO size configuration	Actual FIFO size (Bytes)
8x link (e.g. 80MHz LTE stream)	4	2048
4x link (e.g. 40MHz LTE stream)	3	1024
2x link (e.g. 20MHz LTE stream)	2	512
1x link (e.g. 10MHz LTE stream)	1	256
Less than 1x link bandwidth (e.g. control slot/words of a link)	0	128

下面的伪代码给出了配置各个通道的示例。

```
dbFifoBaseAddress= 0;
for(i=0; i< (number of channels); i++)
{
 DbChannel[i].BaseAddress= dbFifoBaseAddress;
 DbChannel[i].BufDepth= (FIFO size according to bandwidth of this channel);
 dbFifoBaseAddress += (FIFO size);
}
```

2.4 PE/PD 配置

PE/PD 配置是 AIF2 相对复杂的部分，因为 PD/PE 被设计来满足多个无线标准的需求。实际应用中，典型的无线标准的配置则可以简化。

2.4.1 通道间的带宽分配

AIF2 用两种方法实现通道间的带宽分配： modulo 规则和 DBM (Dual Bit Map) 规则.

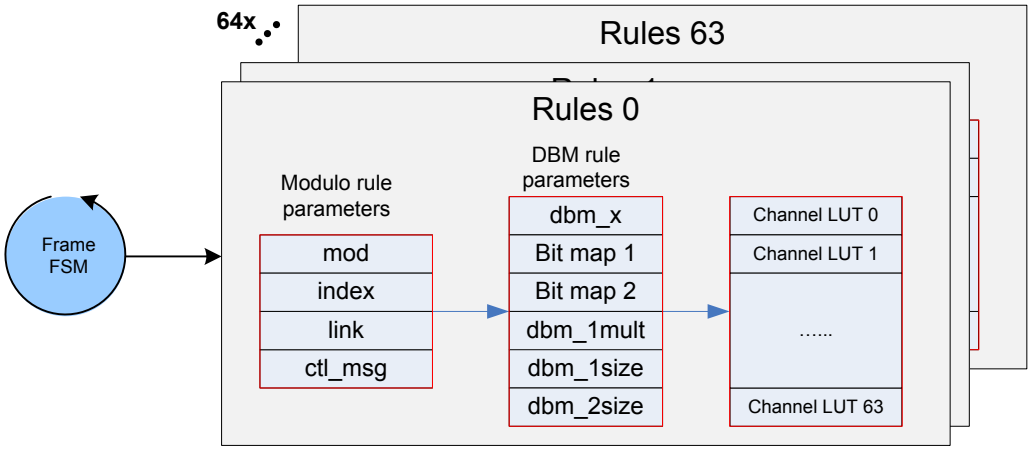

图9. PE OBSAI 传输规则

Module 规则能为多个通道分配相同大小的带宽。所以链路的带宽应该是天线流带宽的整数倍。这种情况适用于 WCDMA,TD-SCDMA,和 LTE(5MHz, 10MHz, 20MHz)这样的无线标准。参考表 2.

DBM 规则也能为多个通道分配相同大小的带宽。同时也支持链路带宽不是天线流带宽的整数倍这样的无线标准，比如 GSM, Wimax, 或者 15MHz LTE。DBM 规则用插入无用数据（bubble）的方法实现速率匹配。

例如，(链路带宽)/(天线流带宽) = 10/3= 3.33333… 图 10 显示了如何用 DBM 规则在 AIF2 链路上分配 3 个天线流。

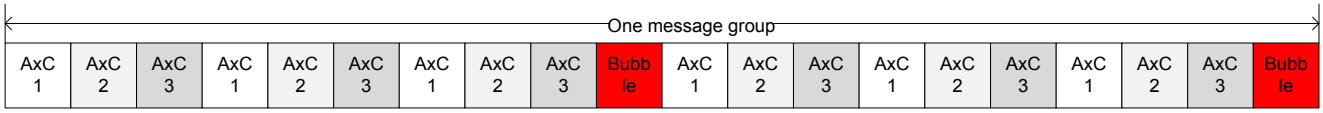

图10. Dual Bit Map 规则示例

上面的例子中，DBM 配置为: X=3, bit map= 001, bit map size =3

另一方面，如果链路带宽是天线流带宽的整数倍，如(链路带宽)/(天线流带宽) = 4，这种情况下，可以使用 modulo 规则或 DBM 规则。

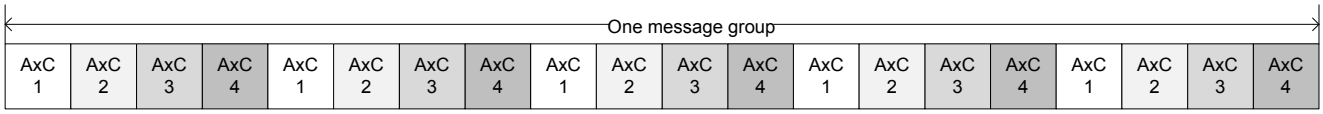

图11. 通道间平均分配带宽的示例

上面的例子中，如果使用 DBM 规则，配置为： X=4, bit map = 0, bit map size=1; 如果使用 modulo 规则，配置为: modulo= 4。

如果不需要使用 modulo 规则，设置 modulo = 0，本文附带的例子中采用此方法。 如果不使用 DBM 规则，设置 X=0;

如果两种方法都被使用，modulo rule 得到 1/mod 的带宽，然后此带宽被 DBM 规则进一步分成 1/X。

这两种方法的使用在 PD/PE 或 OBSAI/CPRI 模式下并不相同。表 8 作了一个简单总结。

表8. 不同情况下规则的使用

	PE (TX)	PD (RX)
OBSAI	Module, DBM	N/A
CPRI AxC slots	DBM	DBM

只有 PE 在 OBSAI 模式下才会同时使用两种方法。另外，OBSAI 的控制时隙和数据流 AxC 时隙必须使用不同的 modulo rule 配置单元，一共有 64 个配置单元。

对于 CPRI 模式，DBM 规则仅仅用于 CPRI AxC 时隙，CRPI 控制字和数据通道间的映射用不同的方法处理。详细信息请参考 AIF2 用户手册。

注意，PD 通道按照消息头来接收 OBSAI 消息。只要消息头包含的信息与通道的设置相匹配，通道就会接收该消息。这就要求发送方对不同的通道设置不同的头信息。本文附带的例子中，发送方 PE 会根据不同的通道设置不同的 OBSAI 消息地址。

而 CPRI 协议中的超帧/基本帧中，不包含通道的信息。所以，CPRI 用 DBM 规则来区分链路上的天线通道。对于某个特定的天线通道，发送方和接收方看到它在链路上的位置应该是一样的，即发送方和接收方应该采用相同的 DBM 规则。

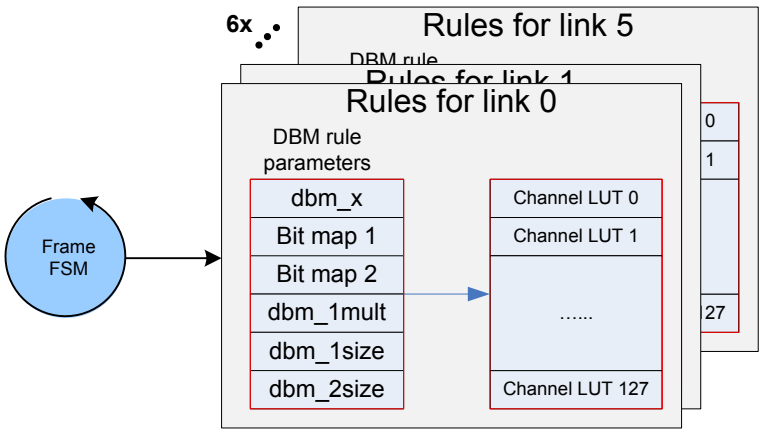

图12. CPRI DBM 规则

下面的 CPRI 例子中，DBM 配置为：X=16, bit map = 0, bit map size=1

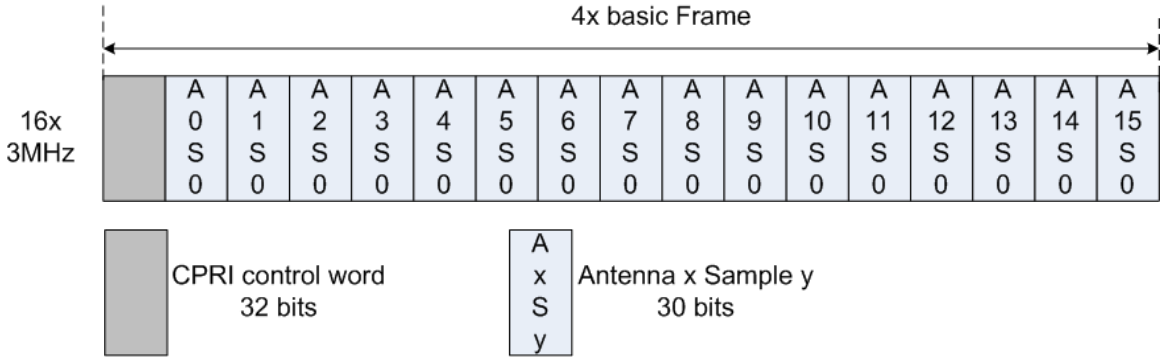

图13. 4x CPRI Link 封装 16 个天线流的示例

注意，对于 PE，有一个内部的 RAM 来存储 OBSAI 模式和 CPRI 模式的通道查找表(channels LUT)。RAM 的大小为 64*64LUT 单元，在 OBSAI 模式下它可以被看作一个二维的 LUT 数组：

```
ChannelLutStruct channelLuts[64][64];
```

这刚好与 OBSAI 规则的 LUT 数目匹配，所以对于某个 rule 的通道的 LUT，可以表示为：

```
channelLUT[ruleIndex][channelIndex];
```

而对于 CPRI，只有部分 LUT RAM 被使用，RAM 可以被定义为如下的二维 LUT 数组：

```
ChannelLutStruct channelLuts[8][512];
```

基于以上定义，对于一个链路上某个通道的 LUT，可以表示为：

```
channelLUT[linkIndex][channelIndex]; (linkIndex<6; channelIndex<64)
```

所以，第二维的序号 6，7 没有被使用，第一维的序号 64~511 没有被使用。

2.4.2 无线帧计数器的配置

如果 Packet DMA 被使用，则每个 packet 的大小被无线帧计数器配置。

PD/PE 通过无线帧计数器支持灵活的无线帧结构。帧结构被两级计数器定义：

1. 第一级指定了被发送/接收的 Packet DMA 包的大小。同一帧里的数据包，大小可以不同，所以有多个计数器来计数包大小。
2. 第二级指定了一帧里的包的数目。总共有 6 套计数器，在一个系统里的可以支持多个帧结构。

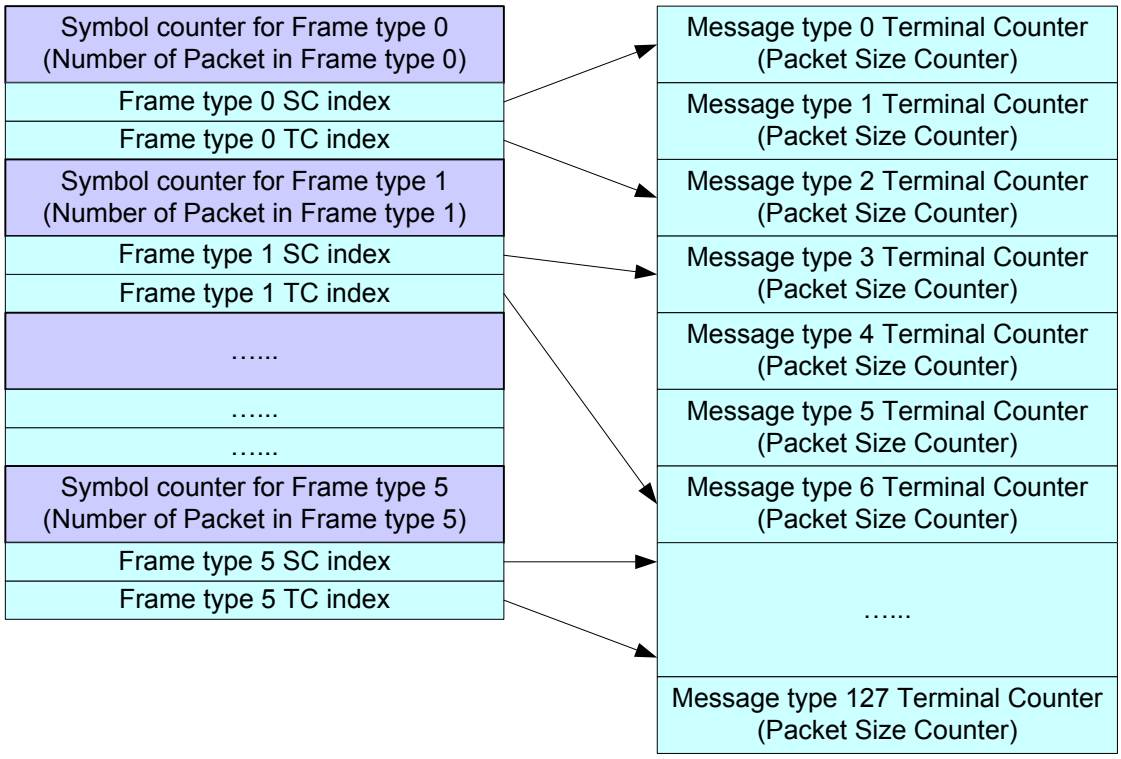

图14. 无线帧计数器

无线帧计数机制与 AT 相似，不同点在于：

1. PD/PE 同时支持 6 套不同的 terminal counts (6 种不同的帧结构)；而 AT 只支持一套 terminal counts (一种帧结构)。
2. AT 计数的是双字节时钟，而 PD/PE 计数的是数据字节(CPRI) 或者 messages (OBSAI)。下表总结了不同点。

表9. 无线帧计数器的单位

AT	Dual-byte clocks	
OBSAI	OBSAI messages (16 bytes payload)	
CPRI	Groups of **4** bytes (PE)	Groups of **16** bytes (PD)

应用程序可以用上面的计数器定义 6 种帧结构。每个通道可配置为采用其中的一种帧结构。对于某个通道，可以在帧边界进行帧结构的切换。

每个天线流（AxC stream）通道可以有一个 AxC offset，无线帧计数器从 AxC offset 开始计数，并且数据从 AxC offset 开始传输。注意，CPRI 模式下，PD 的 AxC Offset 以四个采样点为单位 (QW)，并且以接收的帧边界(sync byte K28.5)为参考点，通常被编程为 0。其他情况下，AxC offset 以 AT 帧边界为参考点。

2.4.3 TDD 配置 (仅 Packet DMA 模式支持)

SERDES 是全双工的，而实际的 TDD 是半双工的。其结果就是，在 TDD 模式下，AIF2 的 SERDES 带宽只有一半被使用。

Packet DMA 支持 TDD 模式，通过以下的方式实现：

- TX: 只在激活的 TDD 时隙发送数据包
- RX: 只在激活的 TDD 时隙接收数据包

对于发送方（TX）,PE 模块的 TDD_AXC 模式应该被使能，这种模式下，PE 只有在接收到 Packet DMA 的数据包后才会发送数据。这意味着，在未激活的 TDD 时隙，用户不能将数据包压入发送队列；而在激活的 TDD 时隙，用户应该注意发送的事件，只能提前一个符号将数据包压入发送队列。例如，LTE 符号 N 需要发送，那么数据包 N 应该在符号（N-1）期间被压入发送队列。

对于接收方（RX）, PD 为每个通道实现了一组可编程的 bit map，每个 bit 对应于无线帧的一个数据包，“1”表示相应的数据包应该被接收，“0”表示数据包被丢弃。注意，在 LTE 协议中，TDD 以子帧为单位来控制，而 AIF2 PD TDD 的一个 bit 对应于一个数据包（符号），而一个子帧中包含了多个数据包（符号），这意味着，如果定义了一个 LTE 子帧为有效的 TDD 子帧，则需要设置多个 PD TDD bits 来激活子帧中的多个符号。

2.5 DIO 控制器 配置

数据结构是软件编程的基础，理解天线数据结构对于 DIO 编程来说是非常重要的。

上行和下行方向分别有 3 个 DIO 控制器，每个控制器可以被配置来支持：

- 与 RAC 接口的上行 WCDMA 天线数据格式。在有些文档中为了和以前的 AIF1 兼容，也叫做“UL RSA”数据格式。
- 与 TAC 接口的下行 WCDMA 天线数据格式。
- 用户定义的天线数据结构。对于有些应用，WCDMA 天线数据可以用 DSP core 或者 RSA 来处理，而不是 RAC 或者 TAC。这种情况下，DSP 存储器内的天线数据可以被软件定义为任意的数据结构。一种典型的结构就是将每个天线流的数据放在一个单独的 buffer，而不是像 RAC 或者 TAC 那样，将几个天线流的数据交织在一起。

本文附带的 DIO 例子测试了上面所说的三种数据结构:

```
typedef enum
{
            /*test (simulate) antenna data from AIF2 to RAC*/
            WCDMA_AIF2_RAC_TEST = 0,
            /*test (simulate) antenna data from TAC to AIF2*/
            WCDMA_AIF2_TAC_TEST,
```

```
            /*test AIF2 antenna data to/from DSP core buffer*/
            WCDMA_AIF2_CORE_TEST
} WcdmaAif2TestType;
```

DIO 可以看作是 AIF2 的 EDMA。表 10 总结了 DIO 的配置参数，并与 EDMA 参数作了比较。

表10. DIO 配置参数

DIO parameter	Equivalent EDMA parameter
Number of quad words (at AIF2 DB buffer)	ACNT
Number of AxC (at AIF2 DB buffer)	BCNT
DBCN (Dio Buffer Channel Number) table (addressing inside AIF2 DB buffer)	N/A
Number of blocks	CCNT
burst size (1,2,4 QW) (at DSP memory)	ACNT
Burst address stride (at DSP memory)	BIDX
Block address stride (at DSP memory)	CIDX
DIO base address (at DSP memory)	SRC, DST

以下的章节描述了每种数据结构的配置。

2.5.1 AIF2 缓冲区的 DIO 配置

以下的图表列出了对于 AIF2 数据 buffer，重要的 DIO 配置。

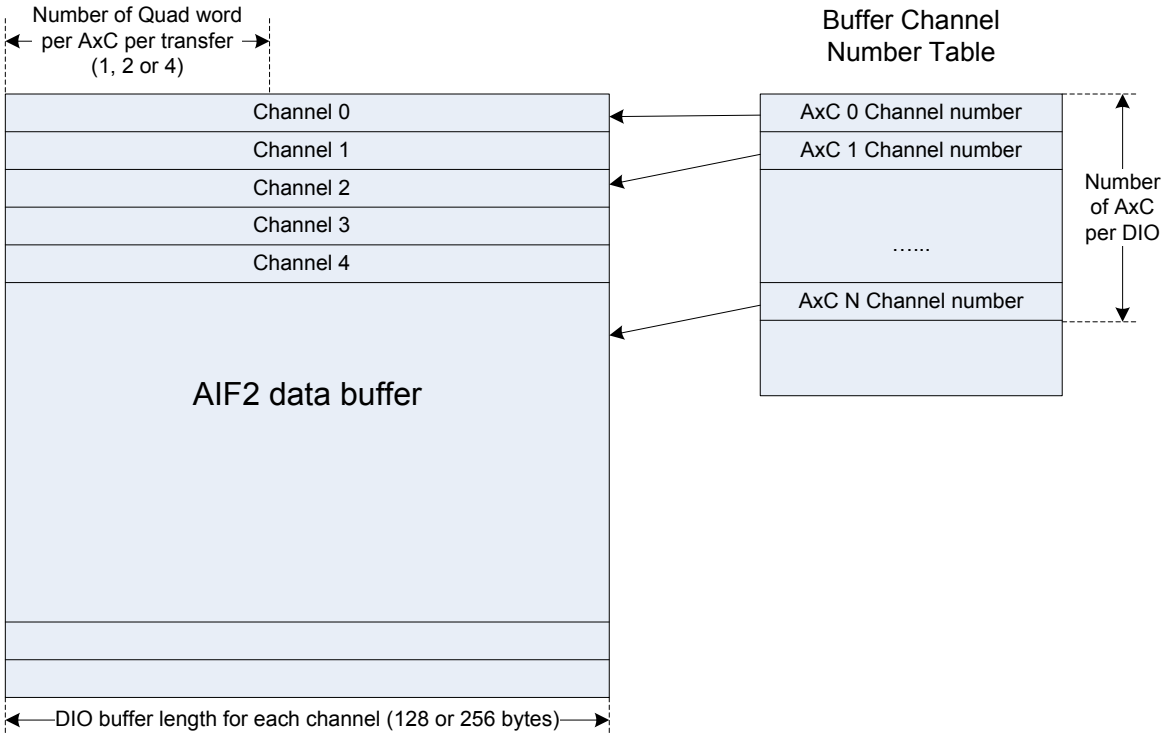

图15. AIF2 数据缓冲区的 DIO 配置

表11. AIF2 数据缓冲区的 DIO 配置

Parameters	Configuration	Comments
DIO buffer length	128 or 256 bytes	For WCDMA stream, 128 bytes is OK. For LTE stream, 256 bytes should be used.
Number of Quad word per AxC	1, 2 or 4	Number of quad word (16 bytes) transferred by one trigger.
Number of AxC	<= 64	Support maximum 64 antenna streams per DIO engine.

每个 DIO 控制器有一个 DMA Buffer 通道表(DBCNT)，用来选择被 DIO 控制器服务的通道。这种灵活的机制下，DIO 并不要求在 AIF2 数据 buffer 中的通道是连续的。

每个 DIO 控制器被 AIF2 AT 模块产生的 DIO 事件触发。每次事件都会触发 DIO 控制器传输 (number of quad word) x (number of AxC)大小的数据块。下一次事件产生时，DIO 控制器会自动增加每个通道的地址指针，如果地址到达了 buffer 的末端，DIO 会自动返转到 buffer 的起始处。所以，对于 DIO 来说，每个通道的 buffer 都被当作循环 buffer 来使用。

注意：有些寄存器字段应该配置为期望值减去 1。例如，如果有 4 个 AxC，那 num_axc 字段应该设置为 4-1=3。

2.5.2 RAC 接口的 DIO 配置

RAC 支持 54 个天线流，而 buffer 被设计为 64 个天线流的大小，所以天线流 54～63 的 buffer 是没有用到的。RAC 的数据 buffer 是一个 32chips 的循环 buffer 。图 16 显示了 RAC 输入 buffer 的结构。

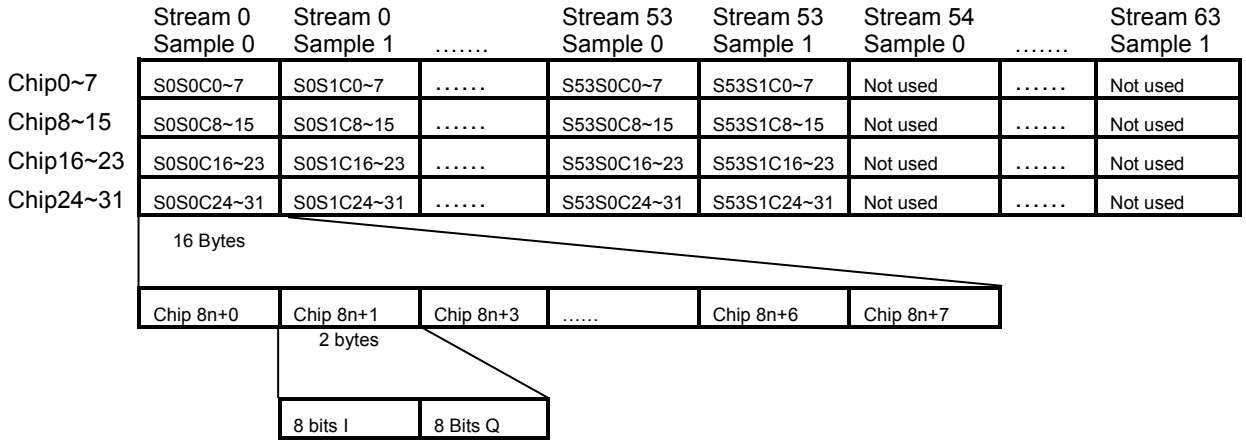

图16. RAC 天线缓冲区数据结构

DIO 每隔 8 个 chips 被触发一次。每次触发，DIO 都会将每个天线流的 8 个 chips 传输到 RAC。表 12 显示了这种情况下的关键 DIO 配置参数。

表12. RAC DIO 配置

Parameters	Configuration	Comments
Number of Quad word per AxC	2	Corresponding to 8 chips
Number of AxC	<= 54	Support maximum 54 antenna streams per RAC.
Number of blocks	4	4 x 8 = 32 chips, this is the size of RAC input buffer.
DMA burst length	4 quad words	One burst include data of two antennas
burst address stride	4 quad words	
block address stride	64 x 2 quad words	

图 17 表达了 DIO 的传输过程。

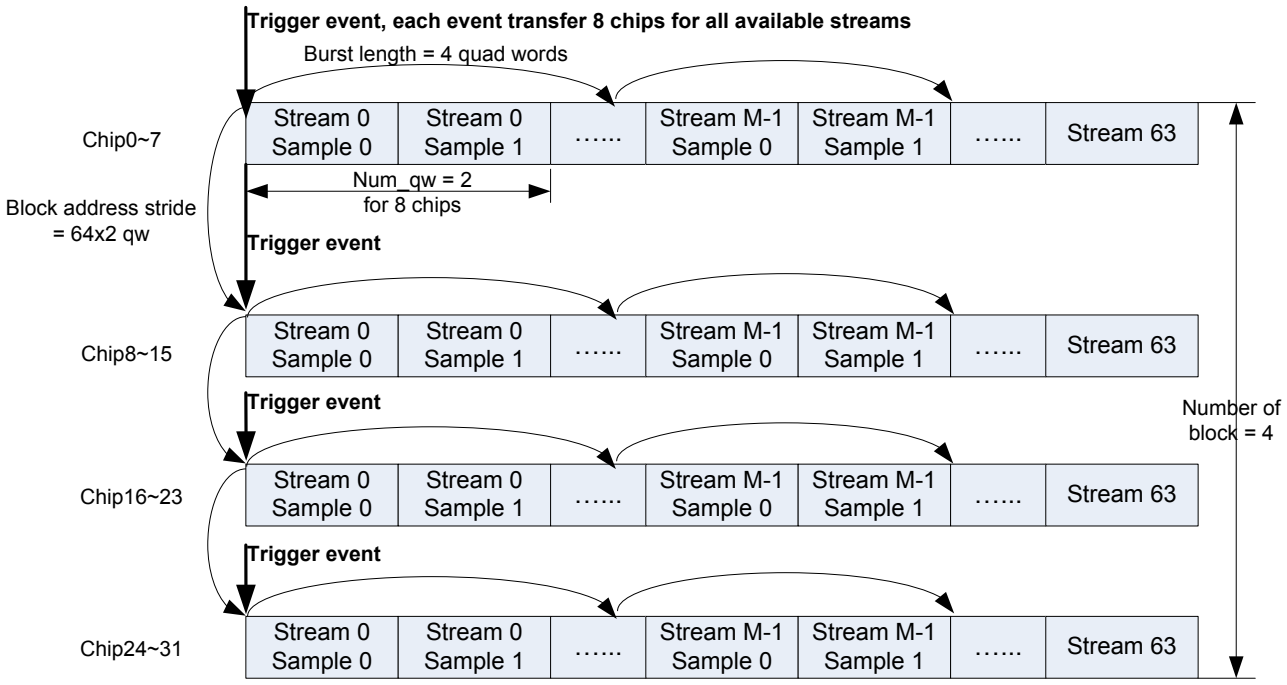

图17. RAC DIO 配置

注意，DIO 的读和写操作是独立的。实际上，num_qw 设置了 DIO 访问 AIF2 buffer 的最小值；burst length 设置了 DIO 访问 RAC buffer 的最小值。所以，上图的配置 num_qw = 2， burst length = 4，实际 DIO 的操作包括：

1. DIO 从第一个 AIF2 通道 buffer 读取 2 个 QW
2. DIO 从第二个 AIF2 通道 buffer 读取 2 个 QW
3. DIO 将这 4 个 QW 写到 RAC buffer 中。

2.5.3 TAC 接口的 DIO 配置

TAC 支持 64 个天线流。TAC buffer 是一个只有 4 个 chips 大小的循环 buffer。图 18 给出了 TAC 输出 buffer 的结构。

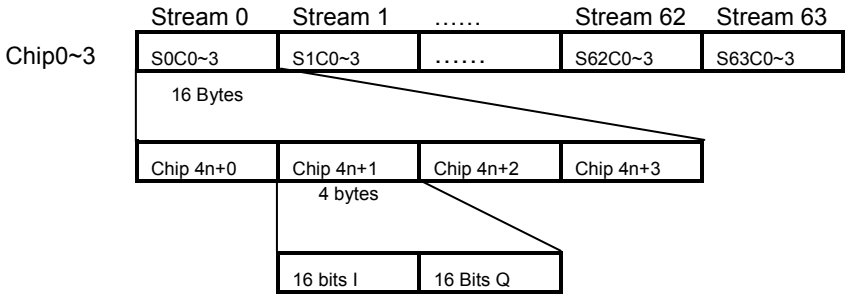

图18. TAC 天线缓冲区数据结构

DIO 每隔 4 个 chips 被触发一次。每次触发，DIO 都会将每个天线流的 4 个 chips 从 TAC 传走。表 13 显示了这种情况下的关键 DIO 配置参数。

表13. TAC DIO 配置

Parameters	Configuration	Comments
Number of Quad word per AxC	1	Corresponding to 4 chips
Number of AxC	<= 64	Support maximum 64 antenna streams per TAC.
DMA burst length	4 quad words	One burst include data of 4 antennas
burst address stride	4 quad words	
block address stride	0	Wrap around to the beginning after every transfer
Number of blocks	32	Since block stride =0, any value for this field should be OK. But a relative larger number may reduce average DIO overhead.

图 19 表达了 DIO 的传输过程。

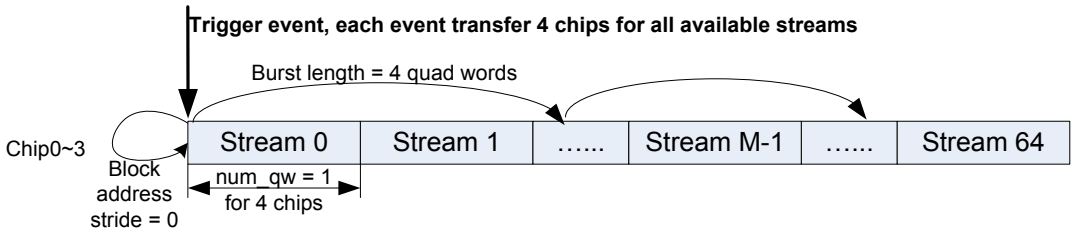

图19. TAC DIO 配置

注意：TAC buffer 是一个循环 buffer ，在一次传输完成后，下一次传输将从 buffer 的起始处重新传输。

2.5.4 DSP 存储器缓冲区的 DIO 配置

对于用 DSP core 或者 RSA 来处理的天线数据，其在 DSP 存储器内的数据结构可以被软件任意定义。当然，也可以用 RAC/TAC 接口的数据结构，但是那样的数据结构是将多个天线流的数据交织在一起，不利于 DSP core 有效地处理它们。一种典型的数据结构是将每个天线流的数据放到一个单独的 buffer。

这种情况下，DIO 的触发周期可以被任意配置，最大的周期是 16chips。周期越大，意味着突发长度越大，数据吞吐量越大。表 14 列出了 DIO 的关键配置参数。

表14. DSP 存储器中的天线缓冲区的 DIO 配置

Parameters	**Configuration**	**Comments**
Number of Quad word per AxC	4	Corresponding to 16 chips
Number of AxC	<= 64	Support maximum 64 antenna streams per DIO.
Number of blocks	<= 8192	num_blks is a 13-bit field. So, the maximum buffer size is limited to 8192*16= 131072 chips≈ 3.4 frames
DMA burst length	4 quad words	One burst include data of 4 antennas
burst address stride	Size of the buffer per AxC	brst_addr_stride is a 12-bit field. So, the maximum buffer size is limited to 4096*4= 16384 chips≈ 6.4 WCDMA slots≈ 0.43 frames
block address stride	4 quad words	

图 20 表达了 DIO 的传输过程

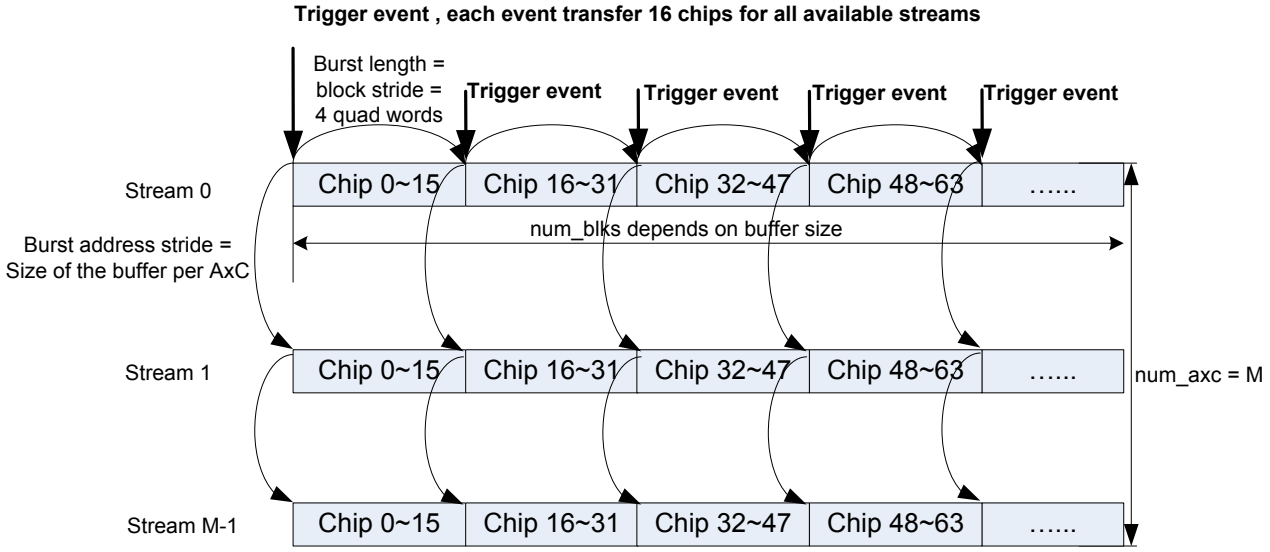

图20. 存储器中的天线缓冲区的 DIO 配置

3 AIF2 的通用数据传输

AIF2 支持用 Packet DMA 传输通用数据。通用数据可以通过 OBSAI 的控制消息或者 CPRI 的控制字和天线数据一起传输。表 15 总结了不同情况下通用数据的传输带宽。

表15. 各种情况下的通用数据传输

	OBSAI	**CPRI**
Over control message slots or control words	Bandwidth utilization is about 1/21 (one control message slot in every 21 message slots). For 8x link, bandwidth is about 6.144*0.8*(16/19)*(399/400)/21= 196Mbps (16/19 means 16 byte payload in a message with 19 bytes; 399/400 excludes the K28.5 in a message group with 400 bytes) Packet Size = n*16 bytes (n>0)	Can use 4B/5B or NULL delimiter. Bandwidth utilization is about 1/16*(64-P)/64 (1/16 means one control word in every 16 words. P is the start pointer of generic data in the control words array with 64 rows). For 8x link, assume NULL delimiter, P=20, bandwidth is about 4.9152*0.8/16*44/64= 168Mbps

注意：因为采用 8B/10B 编码，所以公式中要乘以 0.8。

OBSAI 协议本身就支持通用数据的传输。而 CPRI 为了支持通用数据传输，需要使用额外的分隔符：4B/5B 或 NULL 型分隔符。因为 4B/5B 的开销更大，所以在通用数据的传输中，一般用 NULL 型分隔符。

另外，CPRI 模式时，Packet DMA 必须及时传输数据，不能出现“饥饿”状态。如果位于数据包尾部的数据没有被 Packet DMA 及时传输，PE 将“截断”这个数据包，所以接收方接收到的是一个不完整的数据包，而且并不指示出错误状态。OBSAI 模式下有特殊机制处理这样的“饥饿”，所以不必担心这样的情况。

AIF2 下行方向可以配置 AxC 通道和通用数据通道间的优先级(AD_ESCH_CFG.PRI) 。CPRI 模式下，这个字段应该被设置为 1，表示 AxC 通道和通用数据通道有相同的优先级。而 OBSAI 模式下，这个字段应该被设置为 0，表示 AxC 通道有更高的优先级。

从以上可知，OBSAI 模式更好地支持通用数据的传输。

4 AIF2 调试

AIF2 是很复杂的，调试也是相对困难的。

一些普通的调试方法，比如断点调试，单步调试，或者在 AIF2 运行的时候使用 printf()函数等都会影响 AIF2 的正常工作，因为 AIF2 是时序敏感系统。一旦用了那些调试方法，AIF2 的时序就会被破坏而不能正常工作。

所以，通常在数据传输完成后，才检查 AIF2 状态以便于调试。另外，有些 AIF2 的状态寄存器是“易失性”的，当你“手动地”去检查时，这些状态寄存器的值可能已经改变了。所以，最好的方式是软件将 AIF2 状态寄存器的值存到某个数据 buffer，然后用户可以在传输完成后，查看这个数据 buffer，或者软件将这些信息打印出来。在附带的例子中，“aif_debug.c”文件就是用这种方法实现。

对 AIF2 的调试很有帮助的状态包括:

- AD EOP 计数器 (24-bit), 当达到最大值时，这个计数器将反转。
  - Packet DMA 方式下，它计数的是包的数目。
  - DIO 方式下，它计数的是上行的数据突发（data burst）数目（一般是 64 字节）。下行的 DIO 数据没有被计数。
- AT 帧, 符号/时隙，时钟数
- RM/TM 状态
- RM Pi 值
- 其他的错误或状态

AIF2 EE 模块监测 AIF2 内部 18 个模块的错误/状态。对每个模块都有一套记录错误/状态的寄存器。发生在任何模块的错误都可触发中断，并且设置 ERR_ALRM_ORGN 寄存器中的相应比特。

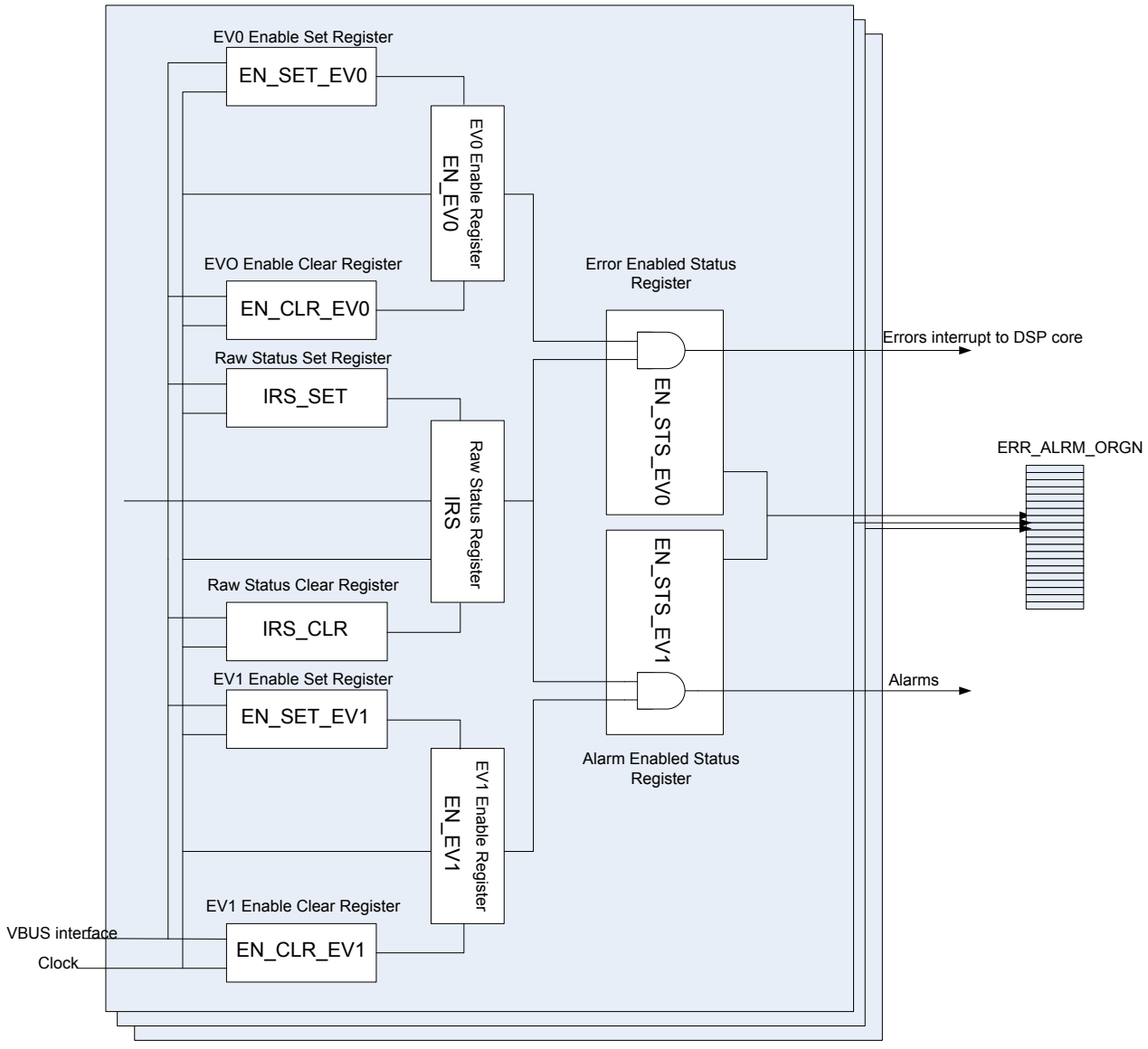

图21. AIF2 EE (Error and Exception) 模块

中断服务程序应该:

1. 检查 ERR_ALRM_ORGN 寄存器以确定是哪个模块发生错误
2. 检查该模块的 EN_STS 以确定是什么错误

3. 清除 IRS

表 16 总结了常见的错误及可能的原因。

表16. AIF2 常见问题和可能原因

Errors	Possible reasons	
RM Line Code Violations, or RM not in SYNC state.	Hardware signal integrity is not good, or the other side of the AIF link does not run properly.	
PI out of window	The AIF2 event timing is not configured properly.	
data shift	the AIF2 AxC offset is not configured properly	
PE DB did not have data for a channel	The Packet DMA does not transfer data in time.	
PE Symbol index in Navigator protocol specific header did not match for one or more symbol.	The packet is not pushed into the TX queue in correct time, or the symbol index in the protocol specific header is not set correctly.	
Packet DMA descriptor starvation	The descriptors are not returned to the FDQ in time, or RX data is generated faster than processing.	

5 示例工程

本文附带的工程是基于 TMS320C6670EVM 板开发。进行外部转发环回测试时，需要使用 BOC 板连接两块 TMS320C6670EVM 板。两片 DSP 的 link4 和 link5 连接到 AMC 接口上，并通过 BOC 板连接在一起。图 22 显示了两片 DSP 间 AIF 的连接。

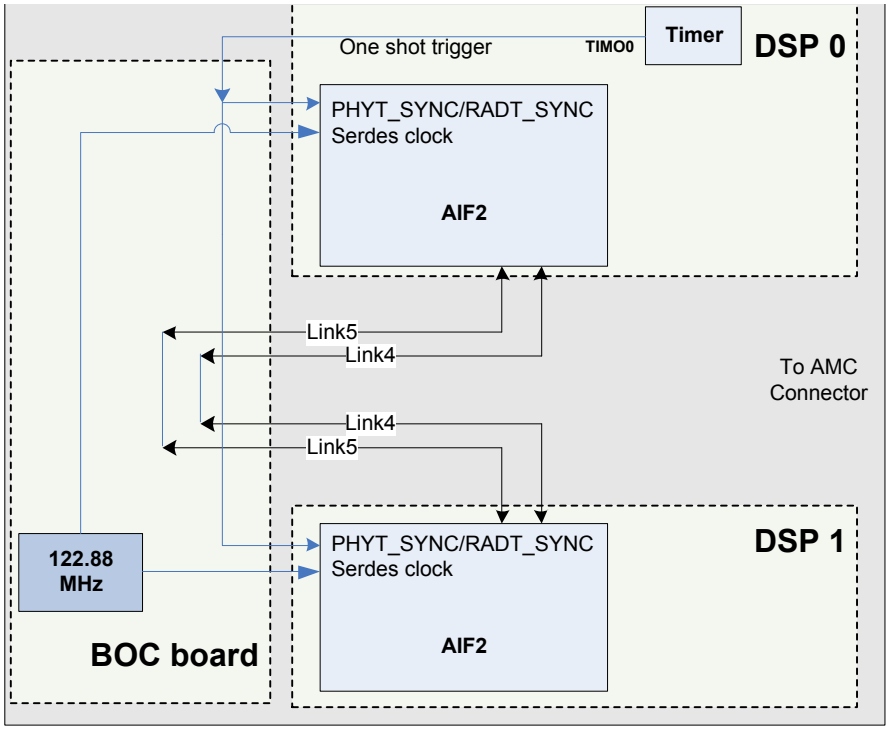

图22. 通过 BOC 板连接两片 DSP 的 AIF 链路

本文附带的例子中演示了 AIF2 的多种工作方式：

- 协议: OBSAI, CPRI
- 无线标准: LTE (FDD/TDD, normal/extended symbol), WCDMA
- 链路速率: 2x, 4x, 8x
- 天线数据存储位置：LL2, SL2, DDR
- 数据类型
  - 天线数据 (AxC 时隙)
  - 天线数据 (AxC 时隙) 和通用数据 (控制时隙)
- 数据路径
  - 内部环回
  - 外部转发环回(两片 DSP 之间)

图 23 显示了不同的数据路径。

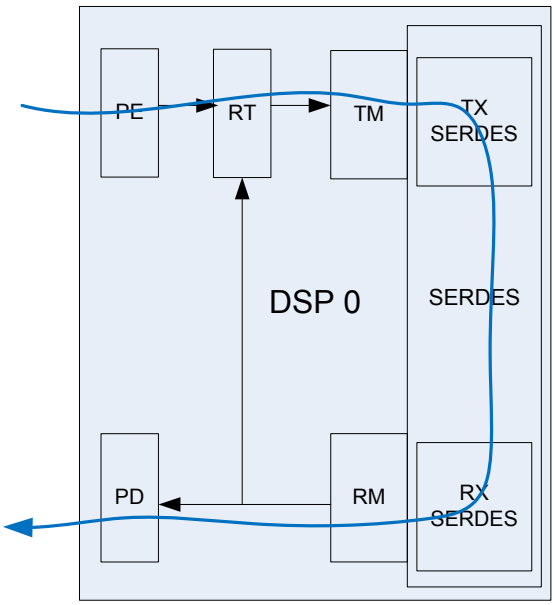

图23. 内部环回测试

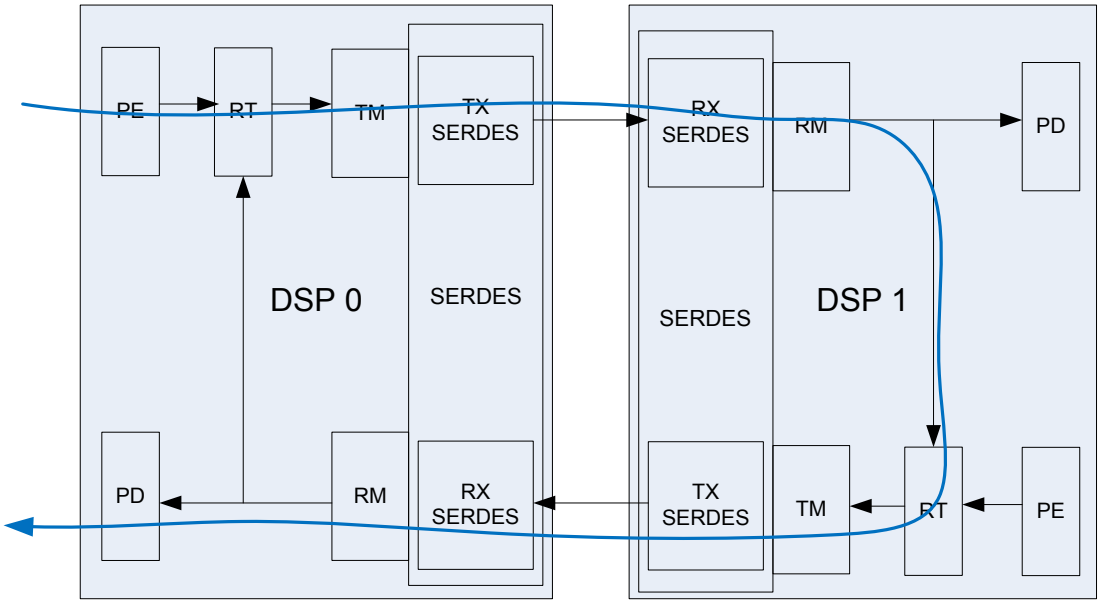

图24. 外部转发环回测试

注意，在 TMS320C6670EVM 板上的 DSP 链路 4，5 不是直接连接在一起的，需要 BOC 板进行连接。否则和链路 0～3 一样，只能进行内部环回测试。

工程的目录结构如下：

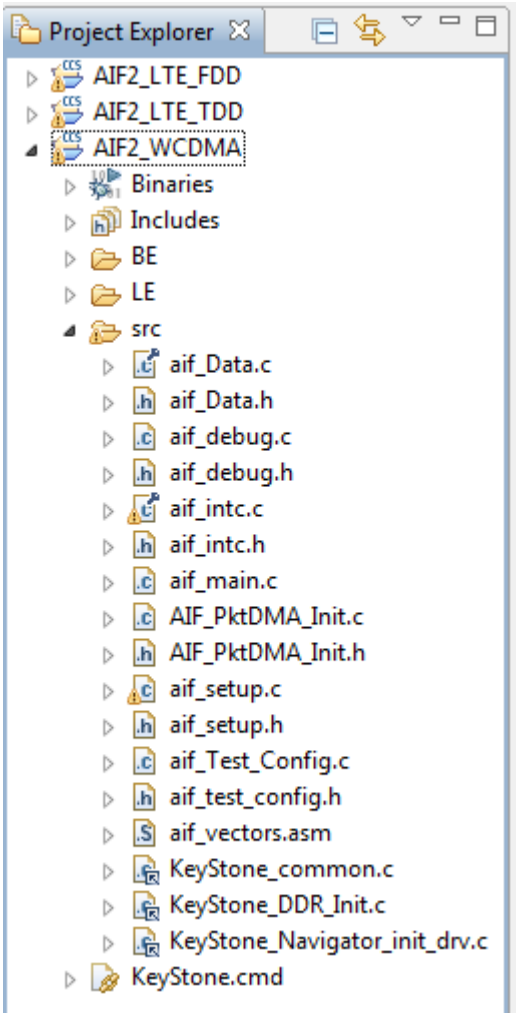

图25. 示例工程目录结构

下面的每个文件夹中都包含了一个特定标准的工程:

- AIF2_LTE_FDD
- AIF2_LTE_TDD
- AIF2_WCDMA

“common” 文件夹包括了一些通用代码，比如 DDR 初始化，Multicore navigator 的驱动等每个工程都有一个“src”文件夹。表 17 列出了代码的内容。

表17. 示例工程中的源文件

Files	Descriptions
aif_main	main() function and the top level control codes

aif_Test_Config	This file includes basic configuration structure for test. User can modify the basic parameters to change the test mode to verify most functionalities of AIF. The codes in this file calculate many other parameters for AIF based on the basic configuration.
aif_setup	Setup AIF registers according the configuration parameters from aif_Test_Config module.
aif_Data	This file includes code for sending data and processing RX data, TX/RX packets statistics information is collected and summarized
aif_PktDMA_Init	This file setup Packet DMA and QMSS for AIF2.
aif_intc	This file setup interrupt controller, and includes ISR for Frame/Slot/Symbol, data transfer are triggered by them. ISR of QMSS accumulation triggers RX packet processing. ISR for error log is also in this file.
aif_debug	This file include functions to log and print status and error for debug.

运行示例工程的步骤：

1, 如果测试两个 DSP 之间传输，要正确设置 BOC 板，通过 BOC 板连接两块 C6670EVM。

2, 连接 CCS 到 DSP。

3, 下载程序到 DSP0 的 core 0。如果测试两个 DSP 之间传输，下载程序到 DSP1 的 core 1

4, 如果测试两个 DSP 之间传输，首先运行 DSP1，然后运行 DSP0。如果只在一个 DSP 上进行内部环回测试，直接运行 DSP0

5, 在输出窗口检查每个 DSP 的测试结果。

5.1 基于 Packet DMA 的工程

基于 Packet DMA 的工程，其典型的输出信息如下：

```
==================AIF OBSAI mode test for 1000 ms (100 frames, LTE FDD 20MHz normal cyclic prefix)==================
link 0 runs at 8x rate, redirection test, antenna data on AxC slot, generic data on control slot, generic packet size = 10240
link 1 runs at 4x rate, redirection test, antenna data in AxC slot only
link 2 runs at 8x rate, internal loopback test, generic data in AxC slot only, generic packet size = 10240
link 3 runs at 4x rate, internal loopback test, antenna data on AxC slot, generic data on control slot, generic packet size = 10240
link 4 runs at 8x rate, internal loopback test, antenna data in AxC slot only
link 5 runs at 4x rate, internal loopback test, generic data in AxC slot only, generic packet size = 10240
Ingress End Of Packet count = 237425
Egress End Of Packet count = 238265
AT PHYT Frame= 100, Clock= 50788
AT RADT Frame= 100, Symbol= 0, Clock= 12773
AT RADT captures Frame= 3, Symbol= 19, Clock= 153599 at PHYT boundary
```

```
----------link 0 status----------
captured PI value = 3250
RM ST3 State FRAME_SYNC
RM captured scrambling code = 0x0
TM FSM in FRAME_SYNC state
----------link 1 status----------
captured PI value = 3264
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
----------link 2 status----------
captured PI value = 3080
RM ST3 State FRAME_SYNC
RM captured scrambling code = 0x2
TM FSM in FRAME_SYNC state
----------link 3 status----------
captured PI value = 3085
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
----------link 4 status----------
captured PI value = 3080
RM ST3 State FRAME_SYNC
RM captured scrambling code = 0x4
TM FSM in FRAME_SYNC state
----------link 5 status----------
captured PI value = 3085
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
SL2_FDQ entry count = 133 (initial value 192)
AIF TX queue 0 entry count = 4
AIF TX queue 7 entry count = 2
AIF TX queue 8 entry count = 4
AIF TX queue 15 entry count = 3
  generic Channel 0 transfer 1180 packets, receive 1175 packets (1175 good, 0 bad), 12032000 bytes in SL2, achieve 12 MB/s
  AxC Channel 1 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
  AxC Channel 2 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
  AxC Channel 3 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
  AxC Channel 4 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
Throughput of link 0 = 496 MB/s (56051 good packets, 0 bad packets)
  AxC Channel 5 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
  AxC Channel 6 transfer 13861 packets, receive 13719 packets (13719 good, 0 bad), 120413632 bytes in SL2, achieve 121 MB/s
Throughput of link 1 = 242 MB/s (27438 good packets, 0 bad packets)
  generic Channel 7 transfer 46678 packets, receive 46673 packets (46673 good, 0 bad), 477931520 bytes in SL2, achieve 482 MB/s
Throughput of link 2 = 482 MB/s (46673 good packets, 0 bad packets)
  generic Channel 8 transfer 592 packets, receive 587 packets (587 good, 0 bad), 6010880 bytes in SL2, achieve 6 MB/s
```

```
 AxC Channel 9 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
 AxC Channel 10 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
Throughput of link 3 = 250 MB/s (28305 good packets, 0 bad packets)
 AxC Channel 11 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
 AxC Channel 12 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
 AxC Channel 13 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
 AxC Channel 14 transfer 13861 packets, receive 13859 packets (13859 good, 0 bad), 121642432 bytes in SL2, achieve 122 MB/s
Throughput of link 4 = 488 MB/s (55436 good packets, 0 bad packets)
 generic Channel 15 transfer 23512 packets, receive 23507 packets (23507 good, 0 bad), 240711680 bytes in SL2, achieve 243 MB/s
Throughput of link 5 = 243 MB/s (23507 good packets, 0 bad packets)
Total throughput of AIF = 2201 MB/s (237410 good packets, 0 bad packets)
```

用户可以在“aif_Test_Config.c”文件中改变配置结构中的初始值，并重新编译工程来验证 AIF2 的大部分功能。

```
AifLinkConfig aifLinkCfg[6]=
{
            /*Configuration for link 0*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_8x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        AIF2_AXC_DATA_ONLY,                    /*testDataType*/
                        AIF_MONO_PACKET_SIZE-16,   /*genericPacketSize*/
                        0,          /*numberAxC: 0 means maximum number*/
                        LTE_20MHZ                              /*lteBandwidth*/
            },
            /*Configuration for link 1*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_4x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        AIF2_AXC_DATA_ONLY,                    /*testDataType*/
                        AIF_MONO_PACKET_SIZE-16,   /*genericPacketSize*/
                        2,          /*numberAxC: 0 means maximum number*/
                        LTE_10MHZ                              /*lteBandwidth*/
            },
            /*Configuration for link 2*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_2x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        AIF2_AXC_AND_GENERIC_DATA,             /*testDataType*/
                        AIF_MONO_PACKET_SIZE-16,   /*genericPacketSize*/
                        2,          /*numberAxC: 0 means maximum number*/
                        LTE_5MHZ                   /*lteBandwidth*/
            },
            /*Configuration for link 3*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_4x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        AIF2_GENERIC_DATA_ONLY,    /*testDataType*/
                        AIF_MONO_PACKET_SIZE-16,   /*genericPacketSize*/
                        0,          /*numberAxC: 0 means maximum number*/
                        LTE_20MHZ                              /*lteBandwidth*/
            },
```

工程默认为 CPRI 模式，可以预定义“AIF2_LINK_PROTOCOL_OBSAI”宏来选择 OBSAI 模式。 如图 26 所示。

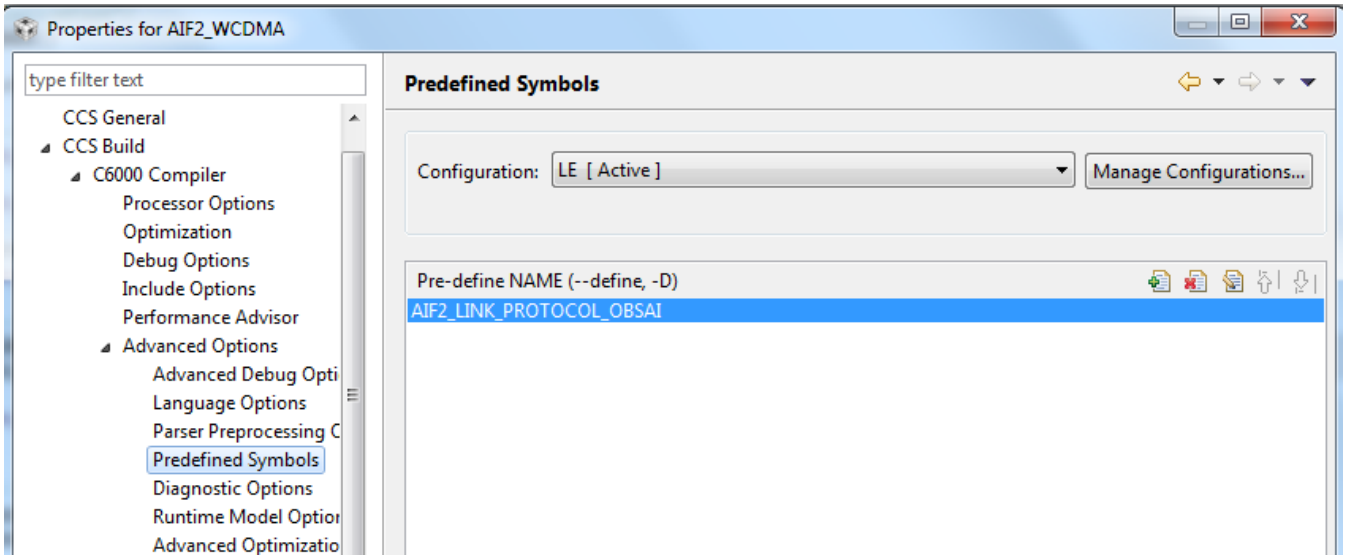

图26. 切换到 OBSAI 模式

AIF2 数据包的 buffer 可以在 “AIF_PktDMA_Init.c” 文件中修改，每一行为一个通道定义了一个数据包的 buffer。

```
/*flow look up table for each channels,
specify which flow be used for each channel*/
unsigned char flowTable[]=
{
            AIF_RX_FLOW_SL2,
            AIF_RX_FLOW_CORE0_LL2,
            AIF_RX_FLOW_CORE1_LL2,
            AIF_RX_FLOW_CORE2_LL2,
            AIF_RX_FLOW_CORE3_LL2,
            AIF_RX_FLOW_DDR,
            ……
};
```

注意，AIF2 是以“round-robin”的方式在每个数据通道上传输 64 字节的数据，所以，存储器不是连续被访问的，对 DDR 来说，这种访问非常低效，所以 DDR 最多支持 3 条 8x 链路。而 SL2 和 LL2 能同时支持 6 条 8x 链路。

以上工程运行在 CCS5.4 环境下，CSL 为 pdk_C6670_1_1_2_6。在其他 PC 上使用新的配置文件时，需要指定正确的 CSL 路径。

5.2 基于 DIO 的工程

基于 DIO 的工程，其典型的输出信息如下:

```
================AIF CPRI mode test for 1000 ms (100 WCDMA frames)================
link 0 runs at 4x rate, redirection test, RAC antenna data and control words
link 1 runs at 4x rate, redirection test, TAC antenna data data only
link 2 runs at 4x rate, internal loopback test, DSP core antenna data only
link 3 is disabled
link 4 is disabled
link 5 is disabled
Ingress End Of Packet count = 11419832
Egress End Of Packet count = 14700
AT PHYT Frame= 100, Clock= 35764; TSC= 1000085507
AT RADT Frame= 100, Symbol= 0, Clock= 9004
----------link 0 status----------
captured PI value = 1011
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
----------link 1 status----------
captured PI value = 1294
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
----------link 2 status----------
captured PI value = 1108
RM ST3 State FRAME_SYNC
TM FSM in FRAME_SYNC state
  AxC Channel 0 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 1 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 2 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 3 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 4 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 5 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 6 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 7 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 8 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 9 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 10 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 11 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 12 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 13 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 14 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 15 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
link 0 transfer 23744 good antenna slots, 0 bad antenna slots
```

```
  control Channel 48 transfer 14700 good packets, 0 bad packets, 3763200 bytes, achieve 3801 KB/s
 Throughput of link 0 = 249 MB/s
  AxC Channel 16 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 17 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 18 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 19 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 20 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 21 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 22 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 23 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 24 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 25 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 26 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 27 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 28 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 29 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 30 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 31 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
 link 1 transfer 23744 good antenna slots, 0 bad antenna slots
 Throughput of link 1 = 245 MB/s
  AxC Channel 32 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 33 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 34 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 35 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 36 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 37 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 38 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 39 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 40 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 41 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 42 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 43 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 44 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 45 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 46 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
  AxC Channel 47 transfer 1484 good packets, 0 bad packets, 15196160 bytes, achieve 15349 KB/s
 link 2 transfer 23744 good antenna slots, 0 bad antenna slots
 Throughput of link 2 = 245 MB/s
 Total throughput of AIF = 740 MB/s
```

用户可以在“aif_Test_Config.c”文件中改变配置结构中的初始值，并重新编译工程来验证 AIF2 的大部分功能。

```
AifLinkConfig aifLinkCfg[6]=
{
              /*Configuration for link 0*/
```

```
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_8x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        WCDMA_AIF2_RAC_TEST,                   /*testType*/
                        4,          /*numberAxC: 0 means maximum number*/
#ifdef AIF2_LINK_PROTOCOL_OBSAI
                        TRUE                                   /*bUseControlMsg*/
#else //AIF2_LINK_PROTOCOL_CPRI
                        32,                                    /*firstCtlWordUsed*/
                        64-32,                                 /*numCtlWord*/
                        CSL_AIF2_DATA_WIDTH_8_BIT  /*dataWidth */
#endif
            },
            /*Configuration for link 1*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_4x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        WCDMA_AIF2_TAC_TEST,                   /*testType*/
                        2,          /*numberAxC: 0 means maximum number*/
#ifdef AIF2_LINK_PROTOCOL_OBSAI
                        FALSE                                  /*bUseControlMsg*/
#else //AIF2_LINK_PROTOCOL_CPRI
                        32,                                    /*firstCtlWordUsed*/
                        32-32,                                 /*numCtlWord*/
                        CSL_AIF2_DATA_WIDTH_16_BIT             /*dataWidth */
#endif
            },
            /*Configuration for link 2*/
            {
                        1,          /*Link Enable: 1=enable, 0=disable*/
                        CSL_AIF2_LINK_RATE_2x,
                        TEST_PATH_INTERLNAL_LOOPBACK,          /*test data path*/
                        WCDMA_AIF2_CORE_TEST,                  /*testType*/
                        2,          /*numberAxC: 0 means maximum number*/
#ifdef AIF2_LINK_PROTOCOL_OBSAI
                        FALSE                                  /*bUseControlMsg*/
#else //AIF2_LINK_PROTOCOL_CPRI
                        32,                                    /*firstCtlWordUsed*/
                        32-32,                                 /*numCtlWord*/
                        CSL_AIF2_DATA_WIDTH_16_BIT             /*dataWidth*/
#endif
            },
```

工程默认为 CPRI 模式，可以预定义“AIF2_LINK_PROTOCOL_OBSAI”宏来选择 OBSAI 模式。 如图 26 所示。

以上工程运行在 CCS5.4 环境下，CSL 为 pdk_C6670_1_1_2_6。在其他 PC 上使用新的配置文件时，需要指定正确的 CSL 路径。

5.3 在其他板上运行工程

在其他板上运行工程，需要做以下的修改：

1. 在其他板上，如果没有用定时器来触发 AIF2，则需要用其他方法来触发。例如，对一些简单的测试，AIF2 可以用写寄存器的方式手动触发，这种方式可以在“aif_Test_Config.h”文件中使能。

```
//#define AIF2_TRIGGER_MANUALLY
```

2. 在其他板上，AIF2 的参考时钟也可能会不同，需要在“aif_setup.c”文件中重新配置。

```
#ifdef AIF2_LINK_PROTOCOL_OBSAI
    pllMpy= CSL_AIF2_PLL_MUL_FACTOR_20X; /*153.6*20=3072*/
#else
    pllMpy= CSL_AIF2_PLL_MUL_FACTOR_16X; /*153.6*16=2457.6*/
#endif
```

3. DSP core PLL 和 DDR 配置也有可能需要在 “aif_main.c”文件中修改，或者用 GEL，或其他方法来初始化。

```
//DSP core speed: 122.88*236/29= 999.9889655MHz
KeyStone_main_PLL_init(122.88, 236, 29);
//DDR init 66.667*20/1= 1333
KeyStone_DDR_init (66.667, 20, 1);
```

参考文献

1. KeyStone Architecture Antenna Interface 2 (AIF2) User Guide (sprugv7)
2. Multicore Navigator for KeyStone Devices User's Guide (sprugr9)
3. TMS320C66x CorePac User's Guide (sprugw0)
4. TMS320C6670 datasheet (SPRS689)

重要声明

德州仪器(TI) 及其下属子公司有权根据 JESD46 最新标准, 对所提供的产品和服务进行更正、修改、增强、改进或其它更改， 并有权根据 JESD48 最新标准中止提供任何产品和服务。客户在下订单前应获取最新的相关信息, 并验证这些信息是否完整且是最新的。所有产品的销售都遵循在订单确认时所提供的TI 销售条款与条件。

TI 保证其所销售的组件的性能符合产品销售时 TI 半导体产品销售条件与条款的适用规范。仅在 TI 保证的范围内，且 TI 认为 有必要时才会使用测试或其它质量控制技术。除非适用法律做出了硬性规定，否则没有必要对每种组件的所有参数进行测试。

TI 对应用帮助或客户产品设计不承担任何义务。客户应对其使用 TI 组件的产品和应用自行负责。为尽量减小与客户产品和应 用相关的风险，客户应提供充分的设计与操作安全措施。

TI 不对任何 TI 专利权、版权、屏蔽作品权或其它与使用了 TI 组件或服务的组合设备、机器或流程相关的 TI 知识产权中授予 的直接或隐含权限作出任何保证或解释。TI 所发布的与第三方产品或服务有关的信息，不能构成从 TI 获得使用这些产品或服 务的许可、授权、或认可。使用此类信息可能需要获得第三方的专利权或其它知识产权方面的许可，或是 TI 的专利权或其它 知识产权方面的许可。

对于 TI 的产品手册或数据表中 TI 信息的重要部分，仅在没有对内容进行任何篡改且带有相关授权、条件、限制和声明的情况 下才允许进行复制。TI 对此类篡改过的文件不承担任何责任或义务。复制第三方的信息可能需要服从额外的限制条件。

在转售 TI 组件或服务时，如果对该组件或服务参数的陈述与 TI 标明的参数相比存在差异或虚假成分，则会失去相关 TI 组件 或服务的所有明示或暗示授权，且这是不正当的、欺诈性商业行为。TI 对任何此类虚假陈述均不承担任何责任或义务。

客户认可并同意，尽管任何应用相关信息或支持仍可能由 TI 提供，但他们将独力负责满足与其产品及在其应用中使用 TI 产品 相关的所有法律、法规和安全相关要求。客户声明并同意，他们具备制定与实施安全措施所需的全部专业技术和知识，可预见 故障的危险后果、监测故障及其后果、降低有可能造成人身伤害的故障的发生机率并采取适当的补救措施。客户将全额赔偿因 在此类安全关键应用中使用任何 TI 组件而对 TI 及其代理造成的任何损失。

在某些场合中，为了推进安全相关应用有可能对 TI 组件进行特别的促销。TI 的目标是利用此类组件帮助客户设计和创立其特 有的可满足适用的功能安全性标准和要求的终端产品解决方案。尽管如此，此类组件仍然服从这些条款。

TI 组件未获得用于 FDA Class III（或类似的生命攸关医疗设备）的授权许可，除非各方授权官员已经达成了专门管控此类使 用的特别协议。

只有那些 TI 特别注明属于军用等级或“增强型塑料”的 TI 组件才是设计或专门用于军事/航空应用或环境的。购买者认可并同 意，对并非指定面向军事或航空航天用途的 TI 组件进行军事或航空航天方面的应用，其风险由客户单独承担，并且由客户独 力负责满足与此类使用相关的所有法律和法规要求。

TI 已明确指定符合 ISO/TS16949 要求的产品，这些产品主要用于汽车。在任何情况下，因使用非指定产品而无法达到 ISO/TS16949 要求，TI不承担任何责任。

	产品		应用
数字音频	www.ti.com.cn/audio	通信与电信	www.ti.com.cn/telecom
放大器和线性器件	www.ti.com.cn/amplifiers	计算机及周边	www.ti.com.cn/computer
数据转换器	www.ti.com.cn/dataconverters	消费电子	www.ti.com/consumer-apps
DLP® 产品	www.dlp.com	能源	www.ti.com/energy
DSP - 数字信号处理器	www.ti.com.cn/dsp	工业应用	www.ti.com.cn/industrial
时钟和计时器	www.ti.com.cn/clockandtimers	医疗电子	www.ti.com.cn/medical
接口	www.ti.com.cn/interface	安防应用	www.ti.com.cn/security
逻辑	www.ti.com.cn/logic	汽车电子	www.ti.com.cn/automotive
电源管理	www.ti.com.cn/power	视频和影像	www.ti.com.cn/video
微控制器 (MCU)	www.ti.com.cn/microcontrollers		
RFID 系统	www.ti.com.cn/rfidsys		
OMAP应用处理器	www.ti.com/omap		
无线连通性	www.ti.com.cn/wirelessconnectivity	德州仪器在线技术支持社区	www.deyisupport.com

邮寄地址： 上海市浦东新区世纪大道1568 号，中建大厦32 楼邮政编码： 200122