EXHIBIT

DSS-2017

AIF Inter DSP Communication

Using C6474 Antenna Interface for Inter-DSP Communication

ABSTRACT

The TMS320C6474 Antenna Interface (AIF) is a CPRI and OBSAI-compliant peripheral whose primary purpose is to transfer baseband antenna samples, via a high- speed SerDes interface, between a radio sub-system and a baseband sub-system in cellular infrastructure applications. In this document, we describe how the AIF peripheral can be used for generic inter-DSP communication. We start by introducing relevant concepts of the OBSAI protocol. Next, we define a software protocol, built on top of OBSAI, for transmission of variable size packets, at a variable (bursty) transfer rate, between two DSP devices connected via the AIF. This software protocol has been implemented and is available as IAClib (Inter-DSP AIF Communication Library), which can be considered a low-level driver for inter-DSP AIF communication. The driver, as well as a sample application demonstrating the use of the driver is explained in this document, and the corresponding source code is made available as a starting point for further development and customization.

Note: Example code for AIF Inter DSP Communication is available on TI's GForge: https://gforge.ti.com/gf/project/aifinterdsp/

1. Introduction

Consider the standard AIF operation. After all required components (AIF, FSYNC, EDMA and interrupts) have been configured at initialization time, the interface is continuously transmitting and receiving data and the timing is strictly controlled by hardware. The nature of inter-DSP data traffic, however, is often asynchronous and bursty, rather than continuous. This type of traffic is best suited for a packet interface like gigabit Ethernet or sRIO, but in some applications, these interfaces are already used for other purposes. Given the continuous nature of the typical AIF traffic, the easiest approach to implement bursty traffic would be to have the AIF transmitter continuously send dummy data, and insert useful data when there is actually something to send. The receiver would have to periodically check if there is any data available. Given the data rates involved, the polling at the receiver could represent a very high load on the DSP (both the CPU and the internal buses). In this document, we introduce a protocol designed to remove the above mentioned polling overhead at the receiver. Rather than having to poll periodically looking for useful data, the receiver is notified by the transmitter, via a special message (called a Start Packet), that a data burst is about to be sent, giving it information about its location (within the OBSAI frame) and size. The receiver then performs necessary setup to be able to extract the data burst from the incoming bit stream.

2. Some elements of the OBSAI protocol

The Antenna Interface (AIF) peripheral provides an interface between a DSP and another device via multiple high-speed SerDes links running at up to 3.072Gbaud/sec. The peripheral was designed for communication between radio equipment and the DSP (which is found on the baseband module). There are two standardized communication protocols that can be used in this application – OBSAI and CPRI. In this section, we will focus on explaining the relevant aspects of OBSAI which are needed to understand the non-standard DSP-to-DSP protocol which will be described in subsequent sections. Detailed explanation of the operation of the AIF in the OBSAI mode is available in [1].

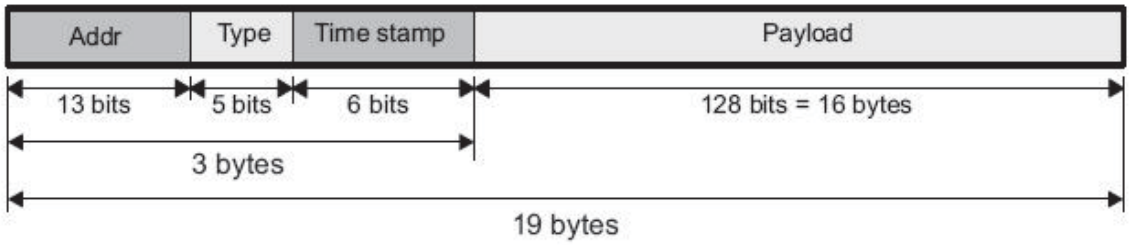
2.1 Overview

Once the antenna interface is up and running, data samples are continuously transmitted and received. The AIF peripheral mainly provides buffering capabilities so that no data would be lost. These buffers need to be filled and emptied at a certain rate in order to prevent overflow (losing data) and underflow (reading stale data). The data (typically antenna samples) can consist of multiple streams (or channels, or antenna containers) which are combined on a single physical link in a TDM (time-division multiplexing) manner. In addition to antenna samples, control channels are also supported in the OBSAI interface and are time-division multiplexed with antenna streams in

specific reserved locations. The OBSAI protocol defines important timing information which guarantees that the transmitter and the receiver are synchronized. Various timing events which are needed for the AIF are generated by the FSYNC module (see [2]) based on an external clock.

2.2 Frame structure

OBSAI frame structure is described in detail in Section 3.1 in [1]. Our intent here is to highlight some of the relevant concepts. The frame structure in OBSAI is based on a 10 msec master frame which consists of nx768000 bytes of payload, where n is the link rate with n=1,2 or 4. The master frame is split into message groups (where the number of message groups per master frame also depends on the physical link rate). Each message group consists of 21 messages, denoted M1, M2, …, M21. Each message consists of 16 bytes of payload and 3 bytes of header. The message is constructed as an RP3 packet and its format is shown in Figure 1.



2.2.1 Time Stamp

The time stamp is embedded into each OBSAI message, as shown in Figure 1, and it is used to make sure that the transmitter and the receiver are synchronized. This is important for antenna data because each antenna sample needs to be traced back to a particular location in the UMTS frame hierarchy. We will take advantage of this feature in our application in an entirely different way, as will be described in Section 3. Time stamp is set to 0 at each frame boundary. In UMTS systems, the frame duration is 10msec, and the smallest unit of time is 1 chip period, which is 1/3.84Mhz = 260.4nsec. The time stamp increments once every 4 chip periods, which translates to ~1.04usec. The incrementing is done based on counting external clock pulses. The time stamp is generated and inserted into the message header by AIF transmit hardware. It is also accessible to software via a read-only register. The time stamp is verified by AIF receive hardware. If the time stamp value contained in the message received from the SerDes link differs from the local time stamp (which is based on the local FSYNC counter), the message is discarded. If the received and the local time stamp are equal, the AIF hardware places the message in the location in the AIF receive buffer (AIF RX RAM) which corresponds to the time stamp. The AIF buffer organization will be discussed in Section 2.3.2.

2.3 Packet Switched vs Circuit Switched Messages

The AIF can transfer packet-switched (PS) messages typically used for control, or circuit switched (CS) messages typically used for antenna samples.

The OBSAI messages are organized into message groups of 21 times slots. 20 of those time slots are used for data messages, and one is used for a control message. Packet-switched messages can be sent both through data slots and control slots. This is configured by software via a look-up table. The two types of messages are therefore time-division multiplexed on the SerDes bus, and the above mentioned look up table is used by hardware to decide at which point in time it needs to insert a packet switched message from the packet buffer, vs. a circuit-switched message which comes from the data buffer (AIF TX RAM).

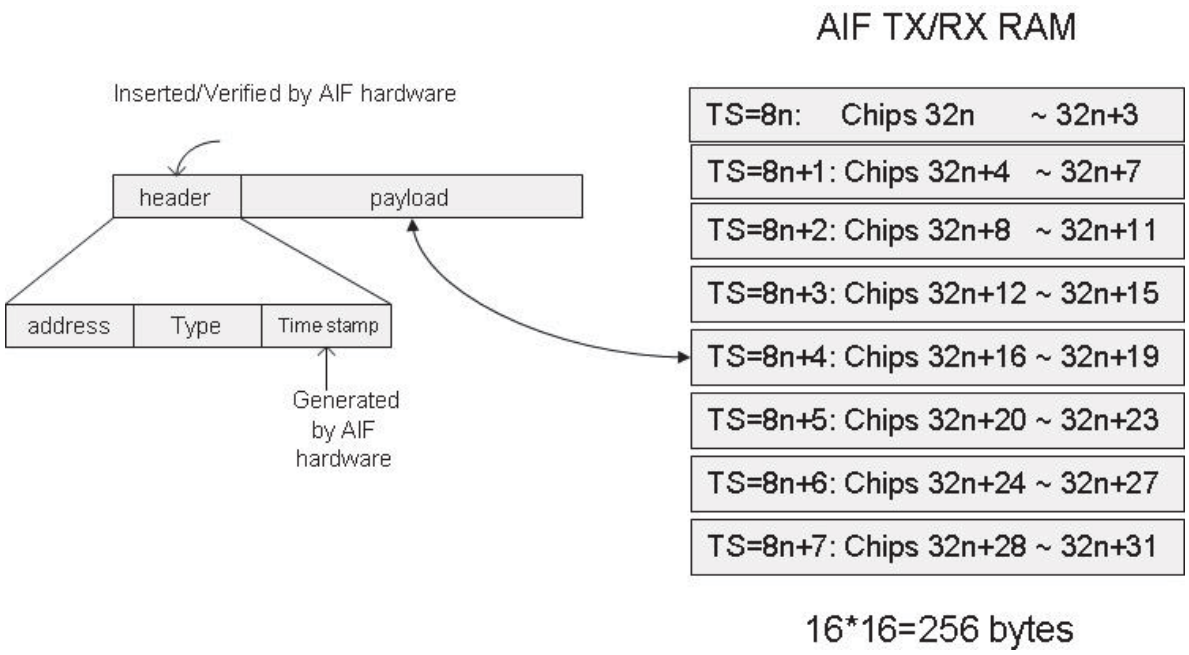
2.3.1 PS FIFOs

There are multiple FIFOs which are used for buffering of packet-switched messages: 3 inbound FIFOs for incoming messages, and 30 outbound FIFOs for outgoing messages. The FIFOs can be programmed to generate an event (CPU or EDMA sync) when a certain number of packets has left/entered the FIFO. On transmit, the EDMA or CPU places

the control message in the transmit FIFO. It is sent out on the next opportunity (i.e. the next slot configured for a PS message). On receive, message (payload and header) are placed into RX FIFO and after a (programmable) number of messages have been received, an event is generated which can interrupt the CPU or trigger an EDMA transfer.

2.3.2 CS RAM

AIF TX and RX RAM are used for circuit switched messages. The size of the RAM is 2Kbytes per link. Typically, these memories are accessed via EDMA. On transmit, the AIF takes data from the TX RAM, inserts header (including the time stamp) and transmits it over the SerDes. On the receive, the AIF checks the received time stamp and then places the message at the appropriate place in the AIF RX RAM. Therefore, the location of data in the AIF RAMs is closely related to the timestamp at which the data is to be sent or has been received. From the point of view of AIF and the SerDes link, the AIF RAMs operate like circular buffers. On the receive side, the data is continuously being written into the RX RAM based on the time stamp, and on the transmit side, it is continuously being pulled out of TX RAM based on the time stamp. This is shown in Figure 2. Therefore, the DSP side needs to be able to stay synchronized with the SerDes operation, i.e. the data needs to be written to the TX RAM at the same rate at which it is pulled out for SerDes transmission, and it needs to be pulled out of the RX RAM at the same rate at which it is being written to by SerDes. This is accomplished via synchronized EDMA transfers. For each synchronization event, 1 time stamp worth of data (16 AxC containers, 16 bytes each) is transferred. Therefore, a synchronization event is used which is generated once per TS increment, or once every 4 chips. The role of the CPU is merely to setup the FSYNC and EDMA prior to activating the AIF link, and (optionally) to respond to EDMA transfer completion interrupts.



2.4 Synchronization (FSYNC)

The synchronization events used by EDMA to synchronize accesses to the AIF RAMs between the DSP/EDMA side and AIF/SerDes side are generated by the FSYNC module (see [2]). In simplified terms, the FSYNC generates its events based on counting the FSYNC clock input pulses. The smallest transmission interval in the UMTS systems is one chip period, or 1/3.84MHz. To allow for timing alignment and offset compensation between the transmitter and th the receiver, the FSYNC module actually counts sub-chips (1/8 of chip duration). Due to the relationship between the time stamps and AIF RAM storage described in previous sections, the FSYNC event which is of interest to our application is the 4-chip event (generated once every time stamp increment, or approximately once every 1usec).

3 Inter-DSP AIF Communication (IAC) Protocol

3.1 Concept : Hybrid PS/CS mode

As seen in the previous section, OBSAI PS mode would be the natural candidate for inter-DSP communication, due to the availability of FIFO receive interrupts, but the bandwidth limitation to about 1-2Gbps means that the AIF can only be utilized at a fraction of its capabilities. On the other hand, OBSAI CS mode is exactly the opposite of "bursty" and "asynchronous" communication that we are looking for: it is running continuously, and inserting bursty data into a continuously running stream of dummy data can potentially present high overhead: (1) The EDMA needs to be emptying receive buffers (AIF RX RAM) continuously, and (2) the CPU needs to be checking the contents of received buffers continuously, and potentially discarding them most of the time.

The solution which we describe in this document is to use a "hybrid PS/CS" mode, in which the transmitter first sends a special message which we call the Start Packet (transmitted as a PS message) with the information about the location of the upcoming data burst, followed by the actual data burst (transmitted as a contiguous block of CS messages). The advantage of sending the Start Packet as a PS message in this context is that, on the receiving side, an event can be generated as soon as the packet arrives in the receive FIFO, which can interrupt the CPU. This is one of the few asynchronous mechanisms provided by the AIF.

3.2 Message Slot Assignments

The location of PS messages within a message group for our application is shown in Figure 3. It is the first message slot in a message group. This slot is configured as a control slot. Each of these control slots represents an opportunity for sending the Start Packet, and each of the CS slots at which the time stamp increments, denoted "CS+ in Figure 3, represents an opportunity for starting the transmission of the data burst.

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20	M21	
Message Group m	PS	CS+	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS+	CS	CS	CS	I
Message Group m+1	PS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS+	CS	CS	CS	CS	CS	CS	CS	I
Message Group m+2	PS	CS	CS	CS	CS	CS	CS	CS	CS	CS+	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	I
Message Group m+3	PS	CS	CS	CS	CS	CS+	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	I

PS	Packet-Switched message sent in a control slot
CS	Circuit-Switched message sent in a data slot
CS+	First Circuit-Switched data slot with a new (incremented) time stamp value

While Figure 3 graphically represents the configuration of the AIF lookup tables and the packet types assigned to each slot, Figure 4 shows an example of a data burst transmission. The Start Packet is sent in the first slot of Message Group (m+1), and the data burst transmission starts in slot 14 of Message Group (m+1), which is when the time stamp increments. The data transmission finishes in message group (m+i), where i is a function of the burst length and can be variable.

Message Group m	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	I
Message Group m+1	SP	E	E	E	E	E	E	E	E	E	E	E	E	CS+	CS	CS	CS	CS	CS	CS	CS	I
Message Group m+3	E	CS	CS	CS	CS	CS	CS	CS	CS	CS+	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	I
Message Group m+4	E	CS	CS	CS	CS	CS+	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	CS	I
Message Group m+i	E	CS	CS	CS	CS	CS+	CS	CS	CS	CS	E	E	E	E	E	E	E	E	E	E	E	I

In practice, it is also possible that the transmission of the data burst starts before the Start Packet is transmitted, depending on when the next configured control slot falls relative to the time stamp increment slot. This will not prevent the data to be received correctly because the AIF RX RAM stores 8 time stamps worth of information and the EDMA should have enough time to pull the data out of the AIF RX RAM prior to it being overwritten by new

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.