



FLE Lancer: I2S Audio Core  
8/19/09

## Table of Contents

1. Change Log.....	3
2. Introduction.....	4
3. Architectural Overview.....	5
4. Register Interface .....	9
4.1. Interrupt Actions .....	9
5. Buffer Scheme .....	10
5.1. Buffer Organization .....	11
6. Typical Operation .....	12
7. Reference Designs .....	13
8. Implementation Guide .....	13
9. Utilization Statistics .....	14
10. Pricing, Availability, Customization.....	14

**1. Change Log**

Revision	Description	Editor
1.0	Updated Release to FLE Website	SMS

## 2. Introduction

This document is intended to provide the reader with an architectural overview and instantiation guide for the FLE Lancer I2S Audio Core.

The Lancer has been developed to provide configurable Audio Output capabilities to allow direct interfacing between the Xilinx Microblaze/PowerPC PLB Bus and an external I2S Dac such as the Texas Instruments PCM1742 or similar I2S DAC.

Through the use of this core, the reader can implement a near zero overhead bus mastering I2S Audio core capable of operating in mono or stereo mode using 16-bit or 24-bit sample output mode.

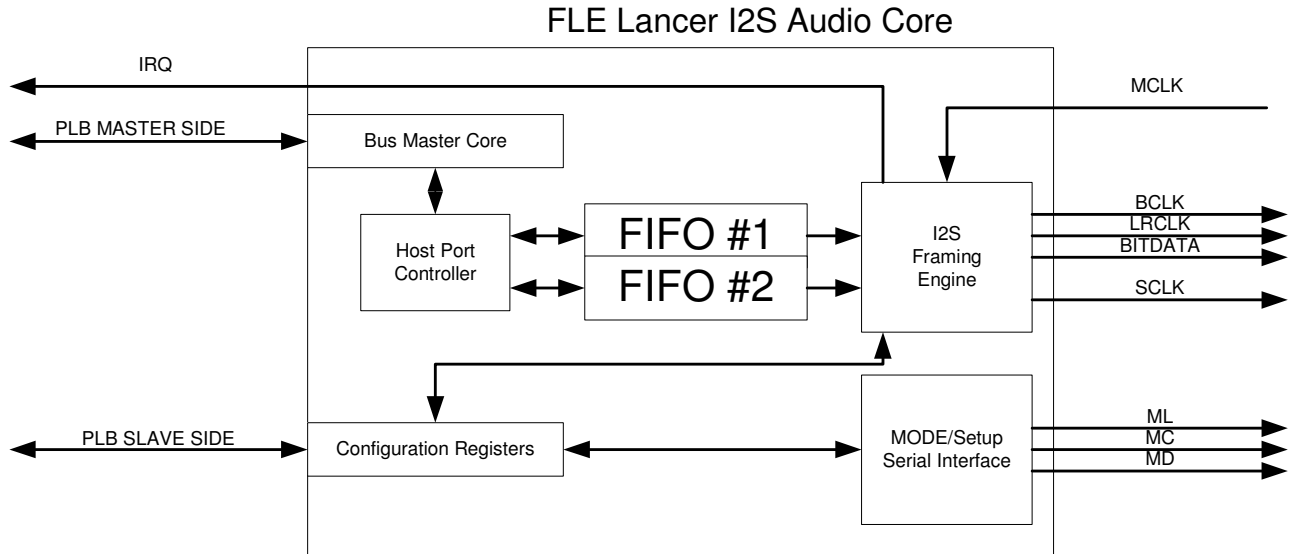
The core is designed to automatically transfer data from a memory pool (DDR/SDRAM) connected to the PLB bus, into a series of internal FIFOs, and then translate the data into I2S compatible serial bit frames.

A register/interrupt interface is provided for interacting with operating system or “main-loop-style” C-Code. The Lancer is also capable of operating in either an interrupt or polled operating mode.

Note: Some diagrams in this document has been reused from the Texas Instruments PCM1742 specification.

### 3. Architectural Overview

The FLE Lancer I2S Core is represented by the I/O Diagram shown below.



The FLE Lancer provides both PLB Master and PLB Slave host port connections.

The PLB Slave port is used to access the internal configuration registers of the Lancer and start/stop/configure both I2S Audio cycles and well as adjust an external I2S Dac settings through a serial interface engine.

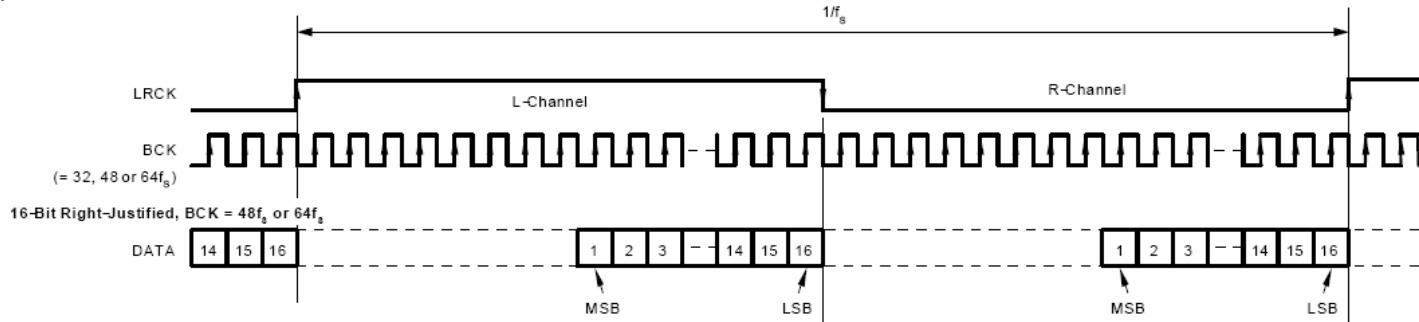
The PLB Master port is used by the internal logic to retrieve samples from an external memory pool (typically a large SDRAM/DDR type memory device) and transfer these samples into one or two internal FIFOs.

The internal FIFOs can be configured in several methods depending upon the customer requirements and the selected IP Core version. The variants are as follows:

- Mono Output 16 bit per sample → 16 bit data is applied to both “Left and Right” outputs
- Stereo Output 16 bit per sample → independent 16 bit data is applied to Left/Right outputs
- Mono Output 24 bit per sample → 24 bit data is applied to both “Left and Right” outputs
- Stereo Output 24 bit per sample → independent 24 bit data is applied to Left/Right outputs

Note that the Mono/Stereo 16-bit modes use the “16-bit Right Justified” output format. The timing of this format is shown below.

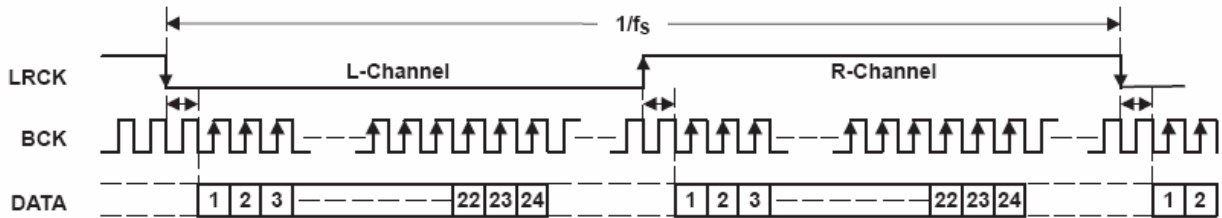
(1) Standard Data Format: L-Channel = HIGH, R-Channel = LOW



This format is selectable in the PCM1742 through a register configuration.

Note that the Mono/Stereo 24-bit mode uses the I2S Data format. Timing of this format is shown below.

(2) I<sup>2</sup>S Data Format; L-Channel = Low, R-Channel = High



This format is selectable in the PCM1742 through a register configuration.

An I2S Framing engine is provided to generate a glueless interface to an external DAC such as the PCM1742 or other industry standard I2S DAC devices. The framing engine provides the following I/O ports:

- SCLK → Output to the DAC for the master sample clock rate
- LRCLK → Left/Right MUX Signal for routing left and right channels
- BITDATA → Serialized sample data for the DAC
- BCLK → Sample data serial clock

The framing engine requires an external master clock input. This input (MCLK) is used to generate the SCLK, LRCLK, BCLK, and drive the internal logic of the I2S Framing engine. The MCLK is also used to directly drive the SCLK output. The LRCLK signal is toggled at a rate of MCLK / 512 allowing all standard sampling frequencies (shown in the table below from the PCM1742 specification) to be achieved with the appropriate MCLK frequency.

**Table 1. System Clock Rates for Common Audio Sampling Frequencies**

SAMPLING FREQUENCY	SYSTEM CLOCK FREQUENCY (f <sub>SCLK</sub> ) (MHz)					
	128 f <sub>s</sub>	192 f <sub>s</sub>	256 f <sub>s</sub>	384 f <sub>s</sub>	512 f <sub>s</sub>	768 f <sub>s</sub>
8 kHz	(1)	(1)	2.048	3.072	4.096	6.144
16 kHz	(1)	(1)	4.096	6.144	8.192	12.288
32 kHz	(1)	(1)	8.192	12.288	16.384	24.576
44.1 kHz	(1)	(1)	11.2896	16.9344	22.5792	33.8688
48 kHz	(1)	(1)	12.288	18.432	24.576	36.864
88.2 kHz	(1)	(1)	22.5792	33.8688	45.1584	(1)
96 kHz	(1)	(1)	24.576	36.864	49.152	(1)
192 kHz	24.576	36.864	(1)	(1)	(1)	(1)

Note: FLE can customize the Sample rate divide period to allow 128, 192, 256, 384, 512, 768 or other divisor rates to be achieved with the Lancer to meet various customer applications.

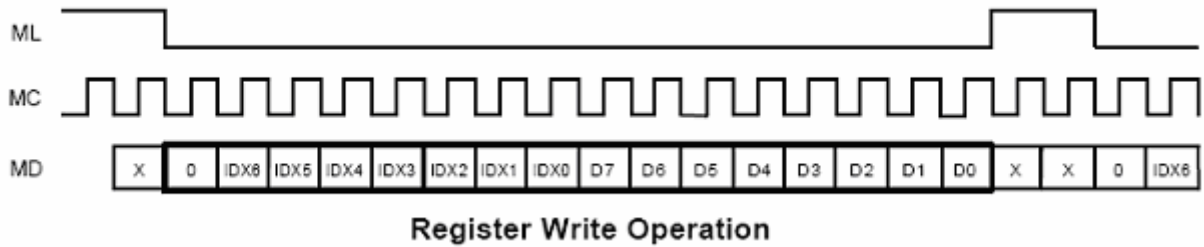
In general, the resultant audio sample Frequency (Fs) is  $F_s = MCLK/512$ . In this manner, any sample rate can be selected depending upon the maximum clock frequency constraint placed on MCLK during Xilinx Place and Route. Typically, this clock rate can be at least 75MHz is a Spartan3 device.

However, in order to produce exact and variable sample rates to match pre-recorded audio such as MP3 files, FLE recommends the use of a programmable silicon oscillator such as the LTC6904 device. This device allows an I2C interface to program its PLL to achieve a very wide range of output frequencies. It is possible (and FLE has a reference design that demonstrates this) to use the LTC6904 to provide the MCLK to the Lancer. This method allows a Microblaze system to exactly match the sample frequency playback rates to support pre-recorded audio such as MP3 files.

Note that the MCLK can be changed, started, stopped, “on-the-fly” during system operation.

The Lancer also provides a Mode/Serial interface engine. The engine is designed specifically for the PCM1742. The engine issues 16bit mode register write cycle operations to the PCM1742 device to configure its internal registers.

A typical write cycle specific to the PCM1742 is shown below.



Note that this serial interface engine is essentially a 16-bit serial shift device. It is not required to use this section of the Lancer core unless the target DAC supports internal register configuration via a similar serial interface bus. FLE can also customize this serial engine for specific customer applications.

## 4. Register Interface

The following table details the register interface of the Lancer.

Offset	Name	Mode	Bits/Value
0x00	BASE	WriteMode	32-Bit register indicating the starting address for the audio sample data
0x00	TXNT	ReadMode	32-bit register indicating the number of samples read by the Lancer
0x04	SAMPLECNT	WriteMode	32-bit register indicating the total number of samples to be transmitted on I2S Port. If this value is not known at startup, simple set it to a “large” number such as 0xAFFF:FFFF. The core can be manually stopped.
0x04	VERSION	ReadMode	Bit [31] indicates if the core is a “Trial” version Bit [30] indicates if the trial timeout has expired Bits [15:0] indicate the core hardware version
0x08	CONTROL	WriteMode	Bit [0] should be set to (1) to start I2S Core sample generation Bit [31] should be set 0→1→0 to issue a user reset of the core
0x08	STATUS	ReadMode	Bit [31] set to (1) when the Serial Interface Engine is ready to accept a new operation Bit [30] set to (1) when the I2S Core has transmitted all of its samples and can then be stopped.
0x0C	SERIAL	WriteMode	Bits [15:0] of this register are transmitted on the Serial Interface engine at each write cycle issued to this register.
0x0C	IRQ	ReadMode	Bit [31] is set to (1) when an interrupt action is required Bits [1:0] indicate the FIFO action required
0x10	N/A	N/A	N/A
0x10	N/A	N/A	N/A
0x14	N/A	N/A	N/A
0x14	N/A	N/A	N/A
0x18	N/A	N/A	N/A
0x18	IRQ_CLR	WriteMode	Pulse bit [0] 0→1→0 to clear a posted interrupt
0x1c	N/A	N/A	N/A
0x1c	BASELEN	WriteMode	32-bit register indicating the size of the first ½ and second ½ of the audio sample buffer.

### 4.1. Interrupt Actions

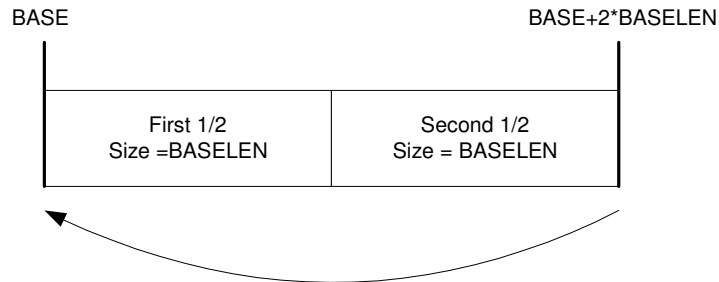
The interrupt assertion of the core signals the user to read the IRQ register and status bits [3:0]. These bits reflect the following states

Bit[0]→ Set to ‘1’ when the first ½ audio sample data has been processed

Bit[1]→ Set to ‘1’ when the second ½ of the audio sample data has been processed.

## 5. Buffer Scheme

The Lancer core uses a single sample buffer register starting at Address “BASE” to form a first ½ and second ½ sample buffer that functions as a ring buffer, this can be seen graphically below.



Prior to starting the core, the entire contents of the First and Second parts of the buffer should be loaded with value audio samples.

At the start of the core operation (ControlBit(0) is set to ‘1’), the Lancer core starts reading at address ‘BASE’. Once the core has read a number of samples equal to BASELEN, the core will post an interrupt request and **continue** reading data from the second part of the buffer.

At the first interrupt, the user program should see the IRQ FIFO bits [1:0] set to “01” indicating that the first part of the buffer has been emptied. The user program should then refill only the first part of the buffer with the next sample data. During this time, the Lancer is also reading from the second part of the buffer.

It is critical that the first part of the buffer is filled **BEFORE** the Lancer completes reading the second part of the buffer. If this condition is not met, a buffer underflow/overflow condition may occur and the audio output will become corrupted.

Once the Lancer finishes reading the second part of the buffer, an interrupt will be posted and the Lancer will reset back to address ‘BASE’ and continue reading from the first part of the buffer. The user should detect that the interrupt occurred and the IRQ FIFO bits [1:0] should be set to “10” indicating that the second part of the FIFO has been emptied. The user program should then refill only the second part of the buffer.

It is again critical that the second part of the buffer is filled **BEFORE** the Lancer completes reading the first part of the buffer. If this condition is not met, a buffer underflow/overflow condition may occur and the audio output will become corrupted.

It is also possible to pre-decode and pre-produce all relevant audio output samples and simply “start” the Lancer. In this manner, the BASELEN should be set to ½ of the number of desired audio samples and the SAMPLECNT register should be set to the total number of desired audio samples. The Lancer core can then be started though ControlBit(0) and the user can determine when all samples have been processed by monitoring StatusBit(30).

---

### 5.1. Buffer Organization

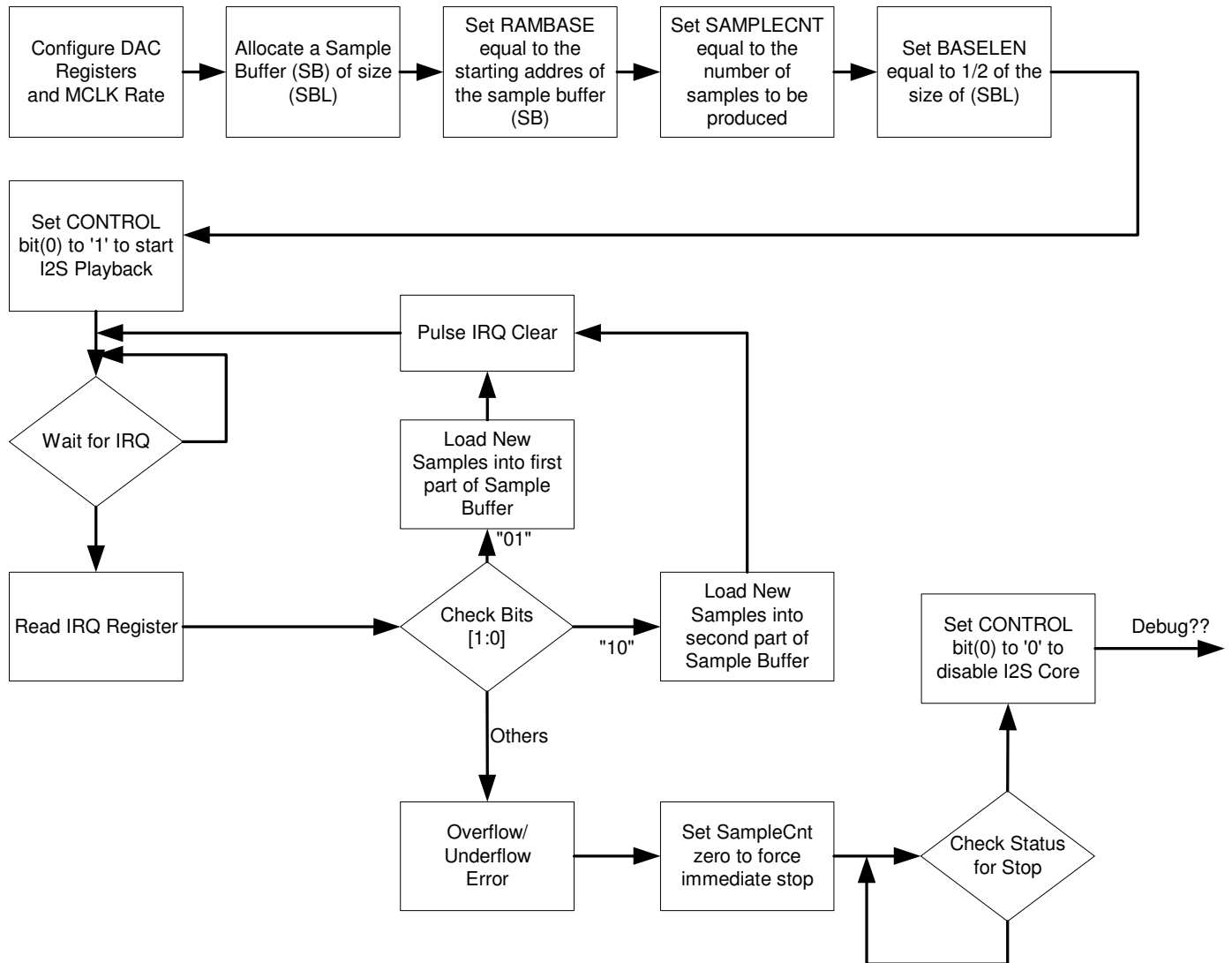
The Lancer always reads 32-bits at a time through the PLB Master interface. Depending on the operating mode, the Lancer will interpret the buffer data in the following manner.

- Mono Output 16 bit per sample
  - Bits [15:0] of the data will be routed to both Left/Right channels
  - Bits [31:16] will be discarded
- Stereo Output 16 bit per sample
  - Bits [15:0] of the data will be routed to the Left channels
  - Bits [31:16] will be routed to the Right channel
- Mono Output 24 bit per sample
  - Bits [23:0] of the data will be routed to both the Left/Right channels
- Stereo Output 24 bit per sample
  - Bits [23:0] of the first word will be routed to the Left channel
  - Bits [23:0] of the second word will be routed to the Right channel

Note that in Stereo 24-bit per sample mode, two 32-bit words are required to form a common Left/Right channel sample.

## 6. Typical Operation

The flowchart below shows the typical operation of the Lancer Core and its software/register interface.



Note: Overflow/Underflow conditions may occur if the processor/memory bandwidth cannot fill the appropriate section of the buffer before a readout of that buffer section occurs. Review the buffer scheme for more details.

---

## 7. Reference Designs

Finger Lakes Engineering provides two operational reference designs allowing the Lancer core to be demonstrated in its fundamental mode of 16-bit Mono output.

Reference #1 → Implementation under the FLE FUSION uCLinux O/S with MP3 Audio Playback

Reference #2 → Implementation directly in Xilinx Platform Studio as a “main loop” code with a simple tone-generator function.

Note that FLE has also produced a daughter card with the PCM1742 DAC that will directly connect to the Xilinx Spartan3-700A/700AN eval card. This daughter card is required for the reference designs to evaluate actual audio output.

The reference designs can both be found at [www.FL-ENG.com](http://www.FL-ENG.com) under the Fusion→Spartan3-700AN reference design.

Direct Link: [http://www.fl-eng.com/products/fusion/xilinx\\_sp3an700/](http://www.fl-eng.com/products/fusion/xilinx_sp3an700/)

## 8. Implementation Guide

During place and route, the Lancer I2S requires a CoreGen FIFO file called audio\_fifo.ngc. This FIFO is provided as part of the Lancer implementation.

A single timing constraint is required on the MCLK signal to allow proper logic functions at the target MCLK rate. A standard timing constraint such as the example below is recommended.

```
NET "AUDIO_CLOCK" LOC = "M22" | IOSTANDARD = LVCMOS33 ;  
Net audio_clock TNM_NET = audio_clock;  
TIMESPEC TS_audio_clock = PERIOD audio_clock 20000 ps;
```

### 9. Utilization Statistics

Device	Slices	LUTs	FlipFlops	BlockRams (depending on sample/channel configuration)
Virtex/Spartan Class	~567	~776	~707	1-4

### 10. Pricing, Availability, Customization

The Lancer IP Core is available in the following formats at the pricing detailed in the table below.

Lancer Configuration	Pricing	Format	Usage
16-Bit Mono Mode	\$1,250 (USD)	NGC NETLIST	Per Project
24-bit Mono Mode	\$1,900 (USD)	NGC NETLIST	Per Project
16-bit Stereo Mode	\$2,500 (USD)	NGC NETLIST	Per Project
24-bit Stereo Mode	\$3,250 (USD)	NGC NETLIST	Per Project

FLE can provide optional quotes for the VHDL Source code and “unlimited project” reuse of the Lancer Core.

FLE can also provide optional customization quotes to include specific MCLK divider rates, serial engine customization, or other requirements.

Contact Finger Lakes Engineering directly for more details on the Lancer Core.