VadaTech AMC511

User's Manual

May 14, 2010

Version 1.2.0

vadatech inc

THE POWER OF VISION

Copyright

Notice

While reasonable efforts have been made to assure the accuracy of this document, VadaTech, Inc. assumes no liability resulting from any omissions in this document or from the use of the information obtained herein. VadaTech reserves the right to revise this document and to make changes periodically and the content hereof without obligation of VadaTech to notify any person of such revision or changes.

Electronic versions of this material may be read online, downloaded for personal use, or referenced in another document as a URL to the VadaTech Incorporated Web site. The text itself may not be published commercially in print or electronic form, edited, translated, or otherwise altered without the permission of VadaTech, Inc.

It is possible that this publication may contain reference to or information about VadaTech products (machines and programs), programming, or services that are not available in your country. Such references or information must not be construed to mean that VadaTech intends to announce such products, programming, or services in your country.

Trademarks

The VadaTech, Inc name and logo are registered trademarks of VadaTech Incorporated in the U.S.A. All other product or service names mentioned in this document are the property of their respective owners.

Revision History

Doc Rev	Description of Change	Revision Date
1.0.0	Document Created	6/8/2009
1.1.0	Updated with latest board and reference design information	2/10/2010
1.1.1	Changed the reference design's PCIe width to x4 instead of x8 to more closely match the initial customer's configuration.	3/10/2010
1.2.0	Added discussion of how to set up the reference design FPGA and software code for different available VCXO / Ref Clock frequencies.	5/14/2010

Table of Contents

Figures

Tables

1 Overview

This document describes the AMC511 board and the A/D FPGA Reference Design including the associated software tool and driver. It also describes how to go about customizing the FPGA design for customer specific needs such as adding additional DSP into the FPGA. The FPGA and software are suitable for use as-is for some customer applications or may be customized by the customer to meet their own performance and functional requirements as needed.

1.1 Applicable Products

- VadaTech AMC511
- Related product: VadaTech AMC510

1.2 Document References

- Linear Technology LTC2209 16-bit, 160Msps ADC Datasheet
- Texas Instruments CDCM7005 3.3-V High Performance Clock Synchronizer and Jitter Cleaner Datasheet
- Xilinx Virtex-5 User's Guide (UG190)
- Xilinx Solutions Guide for PCI Express User Guide (UG493)
- Xilinx LogiCORE™ Endpoint Block Plus v1.9 for PCI Express (DS551)
- Xilinx LogiCORE™ IP Endpoint Block Plus v1.9 for PCI Express® Getting Started Guide (UG343)
- Xilinx LogiCORE™ IP Endpoint Block Plus v1.9 for PCI Express® User Guide (UG341)
- Xilinx Virtex-5 Integrated Endpoint Block for PCI Express Designs User Guide (UG197)
- Xilinx Virtex-5 Embedded Tri-Mode Ethernet MAC Wrapper 1.6 (DS550)
- Xilinx Virtex-5 Embedded Tri-Mode Ethernet MAC User Guide (UG194)
- PICMG® AMC.0 R2.0 Advanced Mezzanine Card Base Specification

NOTE: When reading the Linear Tech datasheet keep in mind that the AMC511 uses screened parts from Linear Tech running at 180 Msps so the stated maximum of 160 Msps in the datasheet should be read as 180 Msps. The increased clock rate does not otherwise change the performance specifications of the chips, but provides improved oversampling performance in downstream DSP algorithms.

1.3 Acronyms Used in this Document

Acronym	Description
A/D or ADC	Analog to Digital Converter
AMC	Advanced Mezzanine Card
BAR	Base Address Register
BE	Big Endian
BPI	Byte Peripheral Interface
CPU	Central Processing Unit
DMA	Direct Memory Access
DSP	Digital Signal Processing
DWORD	Double Word (32-bits)
FIFO	First-In First-Out (Memory structure)
FIR	Finite Impulse Response (Filter)
FPGA	Field Programmable Gate Array
Gbps	Giga-bits Per Second
GBps	Giga-Bytes Per Second
HWM	High Water Mark
ioctl	Input/Output/Control
IP	Intellectual Property
LE	Little Endian
LED	Light Emitting Diode
LPF	Low Pass Filter
LSB	Least Significant Byte
LVCMOS	Low-Voltage Complementary Metal Oxide Semiconductor
MAC	Media Access Controller
M-LVDS	Multi-point Low Voltage Differential Signaling
mmap	Memory Map
MMC	Module Management Controller
MSB	Most Significant Byte
Msps	Mega-Samples Per Second
MUX	Multiplexer
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PHY	Physical Layer Device
PICMG	PCI Industrial Computer Manufacturer's Group
PIO	Programmed Input/Output
PLL	Phase Locked Loop
SERDES	Serializer/Deserializer
SMB	SubMiniature version B (connector)
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCLK	Telephony Clock
VCXO	Voltage Controlled Crystal Oscillator

Table 1: Acronyms

2 Hardware Overview

The AMC511 card includes the following primary components (your ordering option may vary slightly):

- AMC MMC controller
- Xilinx Virtex-5 XC5VLX110T-1 FF1136 FPGA w/ BPI flash (Intel JS28F256P30T95)
- Texas Instruments CDCM7005 Clock Synchronizer
- Linear Tech LTC2209 180 Msps 16-bit A/D x4
- QDRII SRAM (2 x CY1515V18-250BZXC)
- Secondary Reference In, Trigger In, Sync Out
- Four channel M-LVDS clock transceiver

The top-side layout of the card is shown below:

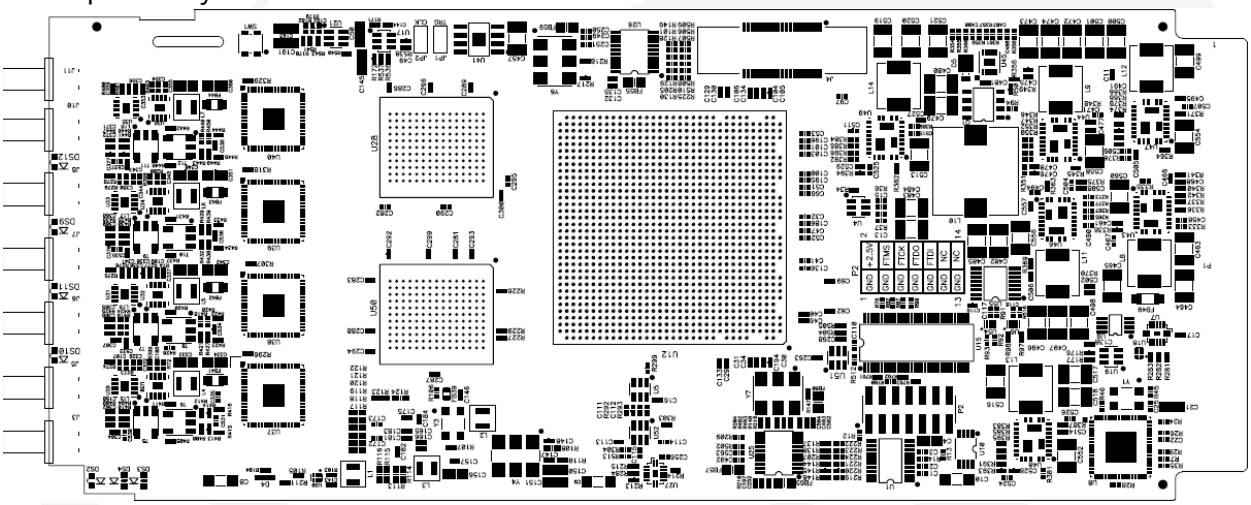

Figure 1: AMC511 top-side layout

A simplified block diagram of the card is shown below:

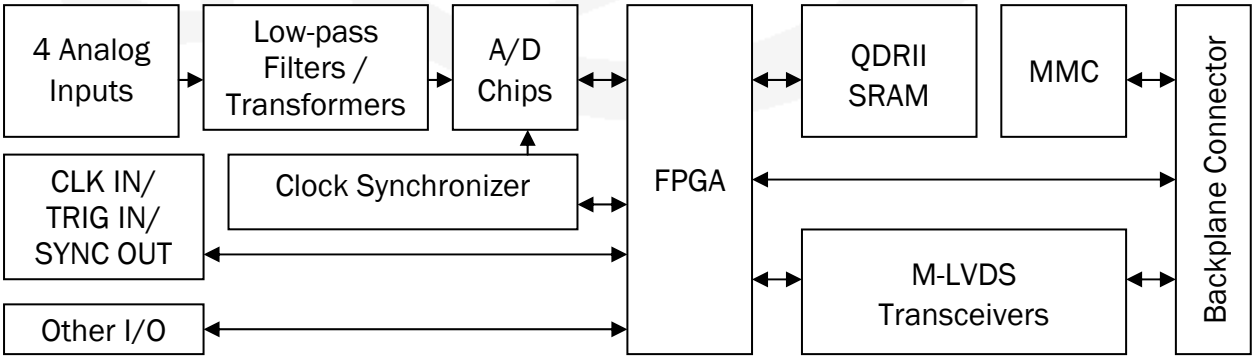

Figure 2: AMC511 block diagram

2.1 Backplane SERDES Interfaces

The card is designed to support flexible system interfacing to the backplane via the reprogrammable FPGA. Interfaces such as PCIe x1/x2/x4/x8, 1000Base-X, XAUI, SRIO, Aurora, and others are realizable. The FPGA reference design demonstrates PCIe x8 and 1000Base-X interfaces. The primary interface for the reference design is PCIe x4 while the 1000Base-X base channels are simply there to show a SYNC indication to validate the hardware.

The possible backplane interfaces are shown below:

GTP Tile	AMC Port	Reference Design	XAUI Options	SRIO Options	Aurora Options	GbE Options
X0Y5	0	GbE0 (hard core) [125MHz]				
X0Y5	1	GbE1 (hard core) [125MHz]				
X0Y4	4	PCIe x4/x8 (hard core) [250MHz]	XAUI0 [156.25MHz]	SRIO0A x1-x4 [156.25MHz]	AURORA0A x1-x8 [156.25MHz]	GbE2 [125MHz]
X0Y4	5	PCIe x4/x8 (hard core) [250MHz]	XAUI0 [156.25MHz]	SRIO0B x1-x4 [156.25MHz]	AURORA0A x1-x8 [156.25MHz]	GbE3 [125MHz]
X0Y3	6	PCIe x4/x8 (hard core) [250MHz]	XAUI0 [156.25MHz]	SRIO0C x1-x4 [156.25MHz]	AURORA0A x1-x8 [156.25MHz]	GbE4 [125MHz]
X0Y3	7	PCIe x4/x8 (hard core) [250MHz]	XAUI0 [156.25MHz]	SRIO0D x1-x4 [156.25MHz]	AURORA0A x1-x8 [156.25MHz]	GbE5 [125MHz]
X0Y2	8	PCIe x8 (hard core) [250MHz]	XAUI1 [156.25MHz]	SRIO1A x1-x4 [156.25MHz]	AURORA0B x1-x8 [156.25MHz]	GbE6 [125MHz]
X0Y2	9	PCIe x8 (hard core) [250MHz]	XAUI1 [156.25MHz]	SRIO1B x1-x4 [156.25MHz]	AURORA0B x1-x8 [156.25MHz]	GbE7 [125MHz]
X0Y1	10	PCIe x8 (hard core) [250MHz]	XAUI1 [156.25MHz]	SRIO1C x1-x4 [156.25MHz]	AURORA0B x1-x8 [156.25MHz]	GbE8 [125MHz]
X0Y1	11	PCIe x8 (hard core) [250MHz]	XAUI1 [156.25MHz]	SRIO1D x1-x4 [156.25MHz]	AURORA0B x1-x8 [156.25MHz]	GbE9 [125MHz]
X0Y7	12		XAUI2 [156.25MHz]	SRIO2A x1-x4 [156.25MHz]	AURORA1 x1-x4 [156.25MHz]	GbE10 [125MHz]
X0Y7	13		XAUI2 [156.25MHz]	SRIO2B x1-x4 [156.25MHz]	AURORA1 x1-x4 [156.25MHz]	GbE11 [125MHz]
X0Y6	14		XAUI2 [156.25MHz]	SRIO2C x1-x4 [156.25MHz]	AURORA1 x1-x4 [156.25MHz]	GbE12 [125MHz]
X0Y6	15		XAUI2 [156.25MHz]	SRIO2D x1-x4 [156.25MHz]	AURORA1 x1-x4 [156.25MHz]	GbE13 [125MHz]
X0Y0	17			SRIO3 x1 [156.25MHz]	AURORA2 x1-x2 [156.25MHz]	GbE14 [125MHz]
X0Y0	18			SRIO4 x1 [156.25MHz]	AURORA2 x1-x2 [156.25MHz]	GbE15 [125MHz]

Key:
Green = AMC.X standard port options (as supported for customer FPGA designs)
Purple = AMC.X non-standard port options (as supported for customer FPGA designs)
Yellow = AMC.X standard port mappings (as supported by VadaTech FPGA sample design)

Table 2: Possible packplane SERDES interfaces

NOTES: Different protocol types can be mixed and matched within reason (pairs of lanes on the same GTX Tile must generally use the same protocol). If you change the backplane interfaces via a custom FPGA design please contact VadaTech for custom E-Keying records for the MMC so that your backplane E-Keying will match your FPGA implementation. Refer to the Virtex-5 documentation and IP documentation for limitations of the GTP tiles including clock forwarding, placement restrictions, resource granularity, etc.

The AMC511 design attempts to provide the most flexible options for GTP tile clock forwarding. The following diagram illustrates the approach that was taken to minimize the limitations of the clock forwarding in the Virtex 5 FPGA:

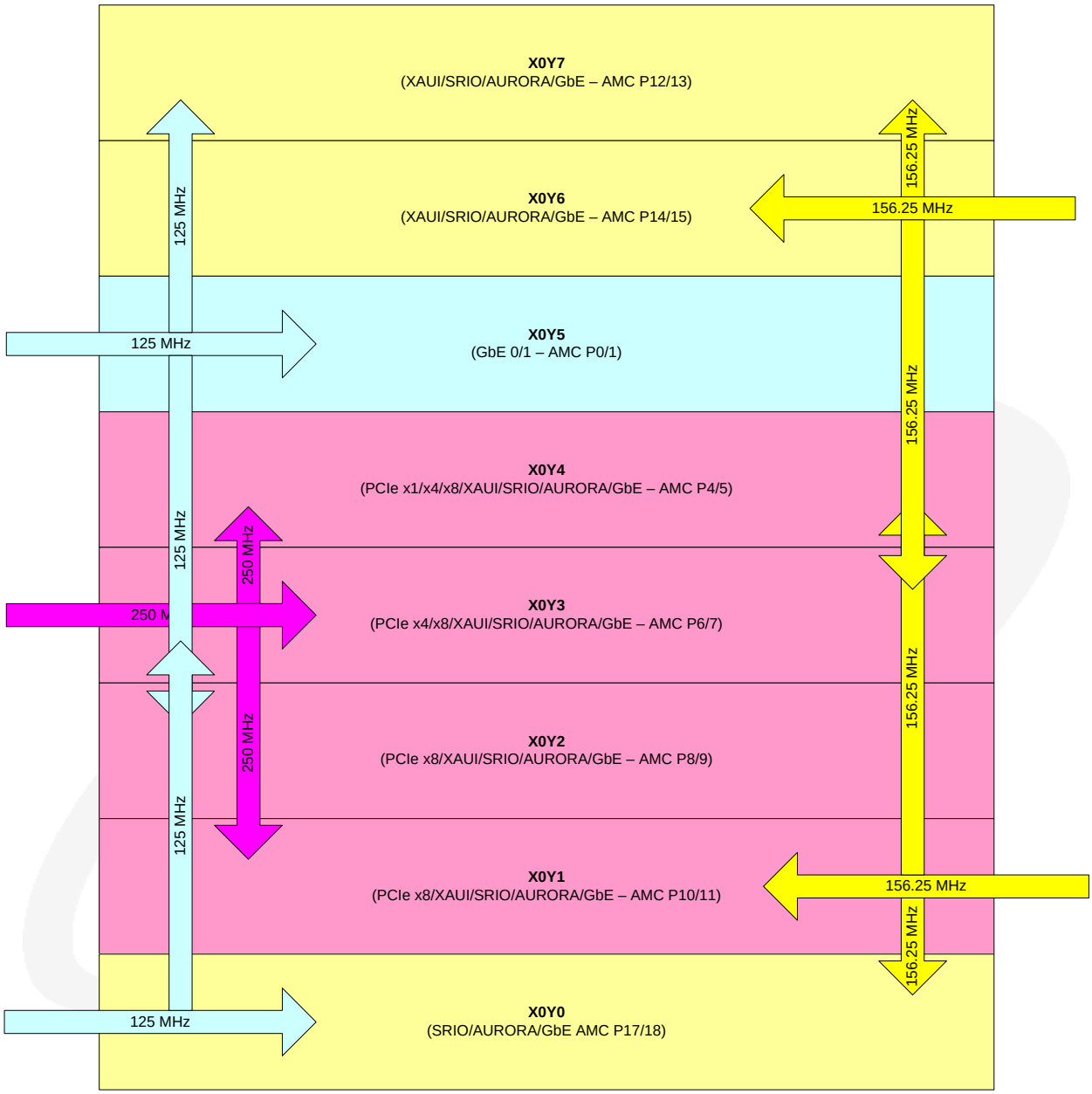

Figure 3: Virtex 5 GTP clock forwarding for AMC511

2.2 AMC Card-edge Pin-out

AMC Finger	Net	AMC Finger	Net	AMC Finger	Net	AMC Finger	Net	AMC Finger	Net
1	GND	35	n.c.	69	AMC/RX7-	103	AMC/TX10+	137	GND
2	AMC+12V	36	n.c.	70	GND	104	GND	138	CLKD-
3	*AMCPS1	37	GND	71	AMCSDA	105	AMC/RX11-	139	CLKD+
4	AMCMP	38	n.c.	72	AMC+12V	106	AMC/RX11+	140	GND
5	AMCGA0	39	n.c.	73	GND	107	GND	141	AMC/RX17-
6	n.c.	40	GND	74	CLKA+	108	AMC/TX11-	142	AMC/RX17+
7	GND	41	*AMCENABLE	75	CLKA-	109	AMC/TX11+	143	GND
8	n.c.	42	AMC+12V	76	GND	110	GND	144	AMC/TX17-
9	AMC+12V	43	GND	77	CLKB+	111	AMC/RX12-	145	AMC/TX17+
10	GND	44	AMC/TX4+	78	CLKB-	112	AMC/RX12+	146	GND
11	AMC/TX0+	45	AMC/TX4-	79	GND	113	GND	147	AMC/RX18-
12	AMC/TX0-	46	GND	80	PCI-E/CLK+	114	AMC/TX12-	148	AMC/RX18+
13	GND	47	AMC/RX4+	81	PCI-E/CLK-	115	AMC/TX12+	149	GND
14	AMC/RX0+	48	AMC/RX4-	82	GND	116	GND	150	AMC/TX18-
15	AMC/RX0-	49	GND	83	*AMCPS0	117	AMC/RX13-	151	AMC/TX18+
16	GND	50	AMC/TX5+	84	AMC+12V	118	AMC/RX13+	152	GND
17	AMCGA1	51	AMC/TX5-	85	GND	119	GND	153	n.c.
18	AMC+12V	52	GND	86	GND	120	AMC/TX13-	154	n.c.
19	GND	53	AMC/RX5+	87	AMC/RX8-	121	AMC/TX13+	155	GND
20	AMC/TX1+	54	AMC/RX5-	88	AMC/RX8+	122	GND	156	n.c.
21	AMC/TX1-	55	GND	89	GND	123	AMC/RX14-	157	n.c.
22	GND	56	AMCSCL	90	AMC/TX8-	124	AMC/RX14+	158	GND
23	AMC/RX1+	57	AMC+12V	91	AMC/TX8+	125	GND	159	n.c.
24	AMC/RX1-	58	GND	92	GND	126	AMC/TX14-	160	n.c.
25	GND	59	AMC/TX6+	93	AMC/RX9-	127	AMC/TX14+	161	GND
26	AMCGA2	60	AMC/TX6-	94	AMC/RX9+	128	GND	162	n.c.
27	AMC+12V	61	GND	95	GND	129	AMC/RX15-	163	n.c.
28	GND	62	AMC/RX6+	96	AMC/TX9-	130	AMC/RX15+	164	GND
29	n.c.	63	AMC/RX6-	97	AMC/TX9+	131	GND	165	AMCTCLK
30	n.c.	64	GND	98	GND	132	AMC/TX15-	166	AMCTMS
31	GND	65	AMC/TX7+	99	AMC/RX10-	133	AMC/TX15+	167	*AMCTRST
32	n.c.	66	AMC/TX7-	100	AMC/RX10+	134	GND	168	AMCTDO
33	n.c.	67	GND	101	GND	135	CLKC-	169	AMCTDI
34	GND	68	AMC/RX7+	102	AMC/TX10-	136	CLKC+	170	GND

Table 3: AMC card-edge pin-out

2.3 Front Panel Interfaces

The front panel of the AMC511 is shown below:

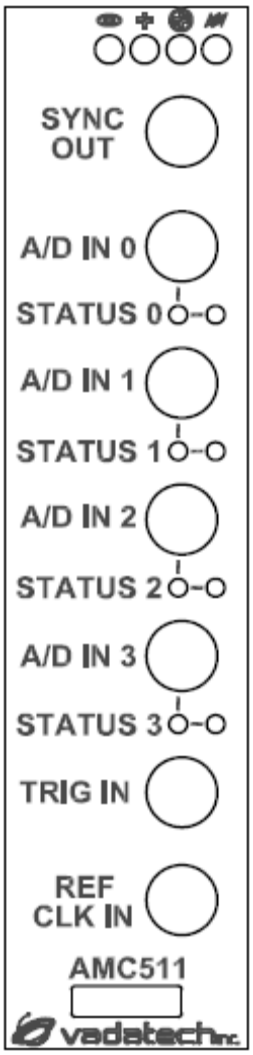

Figure 4: AMC511 front panel

2.3.1 Front Panel Management Interfaces

The front panel includes the standard AMC LEDs showing hot-swap status and general card health. The LEDs behave as follows:

LED	Off	On	Blink
Blue	Card active	OK to remove	Hot-swap/power transitioning
Red	No Fault	Persistent fault	Transient fault
Green	No management power	Management power OK Payload power OK	Management power OK Payload power not expected
Amber	Normal	Firmware upgrade	N/A

Table 4: AMC LED behavior

NOTE: The card should only be removed from a running carrier when the AMC Blue LED is solid ON.

To insert the card, pull out the hot-swap handle until it stops. Insert the card into the carrier's guide rails and push on the front panel firmly until it is fully seated into the connector. If the card does not go fully in, do not force it and instead remove it and check for proper orientation or obstructions. Once fully inserted the Blue LED should go to solid ON while the Green LED should start blinking. Then push in the handle to latch the card into the carrier, the Blue LED should blink for a time and then go solid OFF while the Green LED goes solid ON.

To remove the card, pull out the hot-swap handle until it stops to unlatch the card from the carrier (but do not pull hard enough to remove the card itself yet). The Blue LED should blink for a time and then go solid ON. Once it does, pull the hot-swap handle straight out firmly to remove the card from the carrier.

2.3.2 Front Panel A/D Inputs

The four front panel A/D inputs are provided via SMB connectors which go into a low-pass filter structure, are transformer coupled, and then go into the A/D chips. Please refer to the A/D chip datasheet for signal characteristics.

2.3.3 Front Panel Synchronization Interfaces

The front panel provides REF CLK IN and TRIG IN SMB connectors which are 3.3v LVCMOS compatible (after the optional 50 ohm termination). A SYNC OUT SMB output comes from the FPGA which drives as 3.3v LVCMOS. Refer to the following sections for details on how to enable/disable the 50 ohm input terminations.

2.3.4 Front Panel Channel Status LEDs

The front panel includes one set of green and amber LEDs per analog input. These LEDs default to showing various diagnostic status indicators when the FPGA is first configured. The default power-on behavior is shown below:

LED	Indication
STATUS 0 Green	1000Base-X 0 SYNC
STATUS 0 Amber	Blinking
STATUS 1 Green	1000Base-X 1 SYNC
STATUS 1 Amber	OFF
STATUS 2 Green	PCIe link up
STATUS 2 Amber	PCIe x4 width
STATUS 3 Green	SRAM Passing
STATUS 3 Amber	SRAM Failing

Table 5: STATUS LEDs power-on behavior

NOTE: If both the SRAM Passing and SRAM Failing LEDs are lit this means that the SRAM controller is doing calibration. This calibration will never end if the AMC511 was ordered without the SRAM ordering option; and would therefore be considered a normal condition. Otherwise, with SRAM mounted the calibration should complete very quickly and then only show the SRAM Passing indication.

The reference device driver changes these LEDs to their normal run-time indications once it loads. The normal run-time behavior is shown below:

LED	Indication
STATUS 0 Green	A/D Channel 0 Running
STATUS 0 Amber	A/D Channel 0 Fault
STATUS 1 Green	A/D Channel 1 Running
STATUS 1 Amber	A/D Channel 1 Fault
STATUS 2 Green	A/D Channel 2 Running
STATUS 2 Amber	A/D Channel 2 Fault
STATUS 3 Green	A/D Channel 3 Running
STATUS 3 Amber	A/D Channel 3 Fault

Table 6: STATUS LEDs normal run-time behavior

The run-time behavior can be changed via software if desired. Please refer to the register specification later in this document.

NOTE: The 'Fault' indication is a combination of various statuses from the channel such as A/D conversion overflow, FIFO overflow/underflow, etc. Please refer to the register specification later in the document for details.

2.4 Switches / Jumpers / Headers

The card includes a JTAG header at P2. This header is for programming the FPGA and/or BPI flash and is compatible with the Xilinx Platform Cable USB II.

The card includes a set of DIP switches at SW2. The first two switches on SW2 (1 and 2) are for factory use only and should be set to the OFF position for normal operation. SW2-3 controls the routing of the JTAG lines to the FPGA as follows:

SW2-3	Description
OFF	Connect the JTAG header (P2) to the FPGA's JTAG port
ON	Connect the AMC fingers to the FPGA's JTAG port

Table 7: SW2-3 JTAG routing selection

SW2-4 controls the write protection for the BPI Flash chip as follows:

SW2-4	Description
OFF	Do not write protect the BPI Flash
ON	Write protect the BPI Flash

Table 8: SW2-4 BPI Flash write protection

JP1 and JP2 select the 50 ohm termination for the TRIG IN and REF CLK IN inputs respectively as follows:

JP1/JP2	Description
OPEN	No input termination for TRIG IN/REF CLK IN
SHUNT	50 ohm input termination for TRIG IN/REF CLK IN

Table 9: JP1 and JP2 optional input termination

NOTE: The board expects a 3.3V compatible resulting signal AFTER the optional input termination. That means that if you use the input termination you may need to use a higher voltage so that the result after termination is 3.3V compatible.

The card includes an expansion/debug header at J4. This header provides expansion/debug ports for the FPGA. The header is a Tyco Mictor 38-pin receptacle (part number 2-767004-2). All of the pins use 2.5v LVCMOS signaling. The header is pinned out as follows (with the center blades also as Ground):

Pin	Signal	Pin	Signal
1	(open)	2	(open)
3	Ground	4	(open)
5	CLKE	6	CLKO
7	D15E	8	D15O
9	D14E	10	D14O
11	D13E	12	D13O
13	D12E	14	D12O
15	D11E	16	D11O
17	D10E	18	D10O
19	D09E	20	D09O
21	D08E	22	D08O
23	D07E	24	D07O
25	D06E	26	D06O
27	D05E	28	D05O
29	D04E	30	D04O
31	D03E	32	D03O
33	D02E	34	D02O
35	D01E	36	D01O
37	D00E	38	D00O

Table 10: J4 pin-out

3 Reference FPGA Design Overview

The FPGA is fully customizable and it is expected that customers will want to provide their own custom DSP solution. However, a reference design is also provided to demonstrate the basic board functionality and to provide a complete solution which may be sufficient for some customer's needs. The remainder of this manual discusses the reference design and culminates in a discussion of how the customer may either customize or replace the reference design to add new DSP functionality.

The reference design provides four A/D channels running at 180 Msps under the control of an external host CPU via a PCIe x4 interface. It also provides two 1000Base-X base channel ports which are only present in the reference design to verify the hardware by attaining SYNC. There is no reference application logic attached to the 1000Base-X port. Finally, the reference design includes an SRAM controller with BIST. Again this controller is only in the reference design to validate the hardware and provide a pass/fail indication. The reference design does not make use of the SRAM directly.

The PCIe configuration space information is as follows:

Item	Value
Vendor ID	0xABCD (VadaTech Incorporated)
Device ID	0x4511 (AMC511)
Subsystem Vendor ID	0xABCD (VadaTech Incorporated)
Subsystem Device ID	0x4511 (AMC511)
BAR 0	64KB (32-bit memory mapped, non-prefetchable)
Max Payload	512 bytes

Table 11: PCIe configuration

The design supports a control/status register interface via 32-bit PCIe PIO as well as a four channel DMA controller which uses PCIe posted writes with burst length based on the negotiated largest burst size. The DMA channels are arbitrated by a round-robin arbiter to ensure fairness. The design includes an interrupt controller which signals interrupts using either Legacy INTA or an MSI vector whichever is supported by the system.

The clock synchronizer chip is controlled via a SPI master in the FPGA and its status is monitored with interrupt notification of changes. A flexible clock/signal router allows many different signals to be passed out the front panel output or the backplane M-LVDS outputs. An LED controller is provided which allows for various statuses to be monitored visually on the front panel.

Each A/D chip's output data is pushed through a pipeline in the FPGA which includes the following stages (many of them optional):

- Capture (clocks in the data as close to the pins as possible)
- Optional Formatting (removes effects of the A/D chip's RAND function)
- Optional Pattern Generation (for debugging)
- Optional Half-Band FIR low-pass filter (reduces the data rate by a factor of 2)
- Optional Decimation (reduces the data rate by a factor of 2 to 256 - without filtering)
- Packing (Positions four 16-bit samples into one 64-bit FIFO entry)
- 64KB FIFO (Gets data from A/D clock domain to PCIe clock domain/buffers the data)

The pipeline is structured as shown in the following diagram:

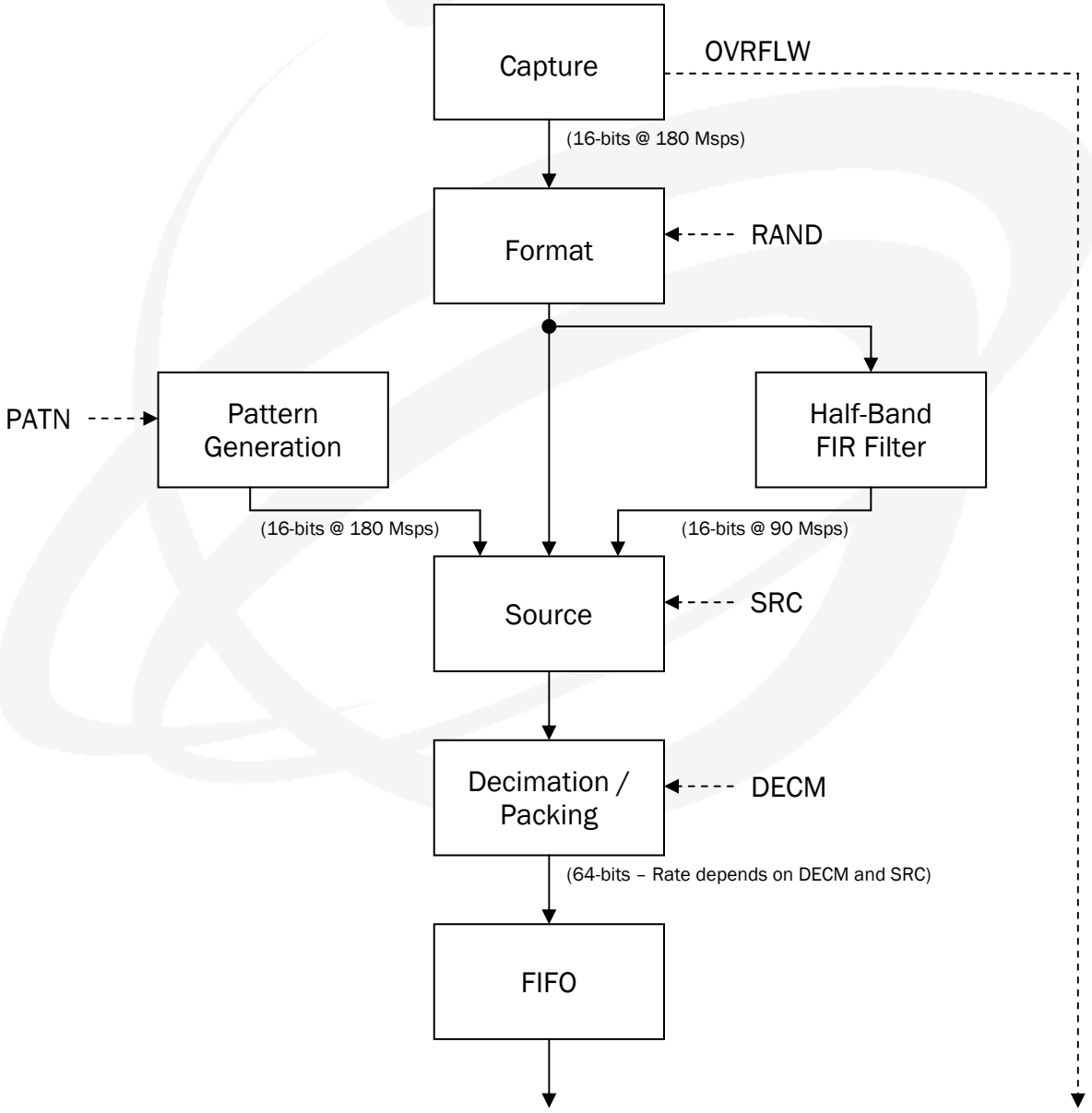

Figure 5: ADC pipeline

Each pipeline stage signals the next when it has data available, so that the data reaching the user application only contains valid samples.

Note that when the Half-band filter is used the data rate is already decimated by a factor of 2 prior to hitting the Decimation stage. The Decimation stage should only be used to evaluate system throughput/loading or if the input signal is already appropriately band-limited prior to entry to the board, otherwise aliasing may occur. In the latter case, a high-quality external low-pass filter with a cut-off at half the decimated sample rate is strongly recommended.

3.1 PCIe Bridge w/ DMA Burst Engine

The PCIe Bridge implements a state machine which converts PCIe Transaction layer packets into internal local bus transactions. It supports up to 32-bit PIO reads and up to 32-bit PIO writes. It also generates out-bound DMA bursts as requested by the DMA Burst Arbiter. The PIO Reads/Writes take priority over the DMA bursts so that the host CPU does not stall needlessly.

3.2 DMA Burst Arbiter

The DMA Burst Arbiter supports four ADC clients and arbitrates between them on a round-robin basis to ensure fair access to the DMA Burst Engine.

3.3 ADC Controller w/ DMA Burst Scheduler

Each ADC controller implements control/status registers, a data pipeline, and a DMA Burst Scheduler. The control/status registers enable control over the A/D chip as well as the processing pipeline and DMA. Data can be gathered from the end of the pipeline either by PIO or DMA accesses; however, DMA access is strongly advised due to the extremely high data rates involved. The A/D chip is automatically put into power-saving mode when the channel is not enabled to reduce power usage and heat generation.

3.3.1 Capture Stage

The capture stage simply clocks the data into the FPGA. This stage exists solely to ensure that the flip-flops involved can be placed directly at the input pad to minimize timing skew. The data from the chips is 16-bit signed (two's compliment) sample data plus one bit of overflow indication. The overflow indication is stripped off after this stage and is captured by a status register instead of following the remaining data flow. This maximizes the bandwidth utilization of the PCIe bus since 17-bit data doesn't pack very efficiently.

3.3.2 Formatting Stage

The formatting stage will automatically remove the XOR randomization that is applied by the A/D chip when the RAND feature is used. It is recommended that the RAND feature always be used to spread the spectrum of the noise caused by the A/D chip outputs.

3.3.3 Pattern Generation Stage

A pattern generator can be switched in as the data source for the decimation stage rather than the formatted A/D data or FIR data. The pattern generator is useful in verifying the dataflow through the hardware and software to the end-user application. It is capable of generating the following patterns:

- All zero bits
- All one bits
- Most negative signed 16-bit value
- Most positive signed 16-bit value
- Repeating ramp from most negative to most positive 16-bit values

3.3.4 Half-Band FIR Low-Pass Filter Stage

The FIR filter stage implements an 83-tap half-band low-pass filter with decimation by 2. It utilizes Distributed Arithmetic architecture to allow it to operate at the full incoming sample rate. Samples go into the filter at 180 Msps and come out at 90 Msps. The FIR filter takes care of band-limiting the data prior to decimation to avoid aliasing. Note that this decimation is fixed and independent of the Decimation Stage as described in the next section. The FIR filter has approximately 80dB stop-band attenuation as shown on the next page:

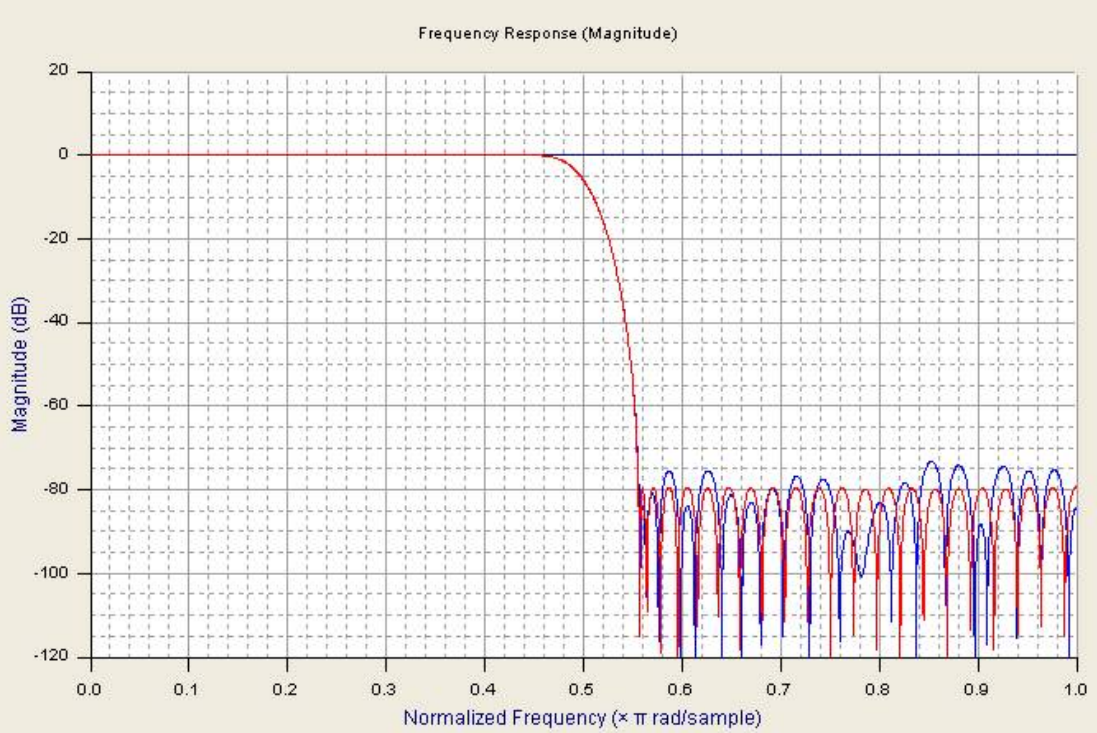

Figure 6: Half-band FIR filter response

3.3.5 Decimation Stage

The Decimation Stage optionally drops samples (without low-pass filtering). It can reduce the sample rate by a factor of 2 to 256. A setting of 0 performs no decimation (equivalent to decimation by a factor of 1), while a setting of 1 performs decimation by a factor of 2, etc.

3.3.6 Packing Stage

The A/D chips and the data pipeline are 16-bit and the PCIe core is 64-bit, so this stage simply packs four samples into one FIFO entry.

3.3.7 FIFO Stage

After the samples are packed they are pushed into a 64KB FIFO structure. This structure serves two purposes. First it enables seamless crossing from the A/D clock domain into the PCIe clock domain. Second it absorbs the latencies which occur while buffering up a DMA burst, waiting for other bus activity to complete, software latencies, etc. The pipeline design ensures that by the time the samples are taken out of the FIFO the host software will know if there was any A/D converter overflow, FIR overflow, or FIFO overflow/underflow so that a high degree of confidence can be established in the captured data.

3.3.8 DMA Burst Scheduler

Each ADC channel has its own DMA Burst Scheduler. The scheduler uses two scatter-gather DMA descriptor chains to facilitate very low-overhead streaming transfers to the main system memory. It reads the DMA descriptors and schedules burst transfers to complete the chunks described there. It waits for the FIFO to fill enough to satisfy a burst then it requests service from the DMA Burst Arbiter. The DMA Burst Arbiter forwards the request to the DMA Burst Engine which will then complete the burst by reading the FIFO using special 64-bit internal reads and writing the data out across the PCIe bus into system memory as instructed. The DMA Burst Scheduler notifies the software whenever a chain completes via a status register/interrupt. Each descriptor can point to up to 512KB of physically contiguous memory with a size/alignment granularity of 4KB. There are 128 descriptors per chain.

The DMA Burst Scheduler has a special mode which can be used for sourcing point-to-point (card-to-card) PCIe DMA transfers rather than doing card-to-memory transfers. This mode may be used for example if the AMC511 card is designated for capturing data while a second card is designated for doing additional DSP of the data. The software reference design doesn't support this mode of operation. However, it is provided in the FPGA as a starting point of what could be done in more sophisticated configurations designed to fully exploit the point-to-point architecture of PCIe (where data does not always need to flow through the memory/CPU).

3.4 Interrupt Controller

The central interrupt controller consolidates the interrupt lines coming from the four ADC cores and the Clock Synchronizer core to provide one single master interrupt line to the PCIe Bridge. The interrupt controller also supports a Bit Change Interrupt mechanism which is used for alerting the software any time the Clock Synchronizer status lines change.

3.5 Clock Synchronizer Controller

The Clock Synchronizer Controller implements a control/status register for basic control of the TI Clock Synchronizer chip. It also implements a SPI Master which can be used to set all of the registers within the TI chip using a serial protocol. The SPI Master handshakes with the host software to ensure that it remains synchronized with the serial data transfer so the software can know when the transfer is complete.

3.5.1 SPI Master

The TI chip includes four 32-bit control registers which are set via a SPI bus interface. The SPI Master function in the Clock Synchronizer Controller provides four shadow registers

which are available via PCIe PIO read/write which it shifts out to the TI chip on command from the host software.

Each time the host software instructs the SPI Master to start; it shifts all four data words to the chip using a 500 kHz clock and then indicates completion to the software via a status register/interrupt.

The SPI Master's default register values are customized to match the AMC511 hardware and differ from the TI chip defaults. These new defaults are shifted out immediately upon FPGA configuration without software intervention. The SPI Master enforces read-only semantics on some of the control fields that are otherwise read/write for the TI chip so that parameters specific to the AMC511 circuits are not violated inadvertently.

3.6 Clock/Signal Router

The Clock/Signal Router enables flexible routing of off-board and on-board signals such as clocks and status indicators, etc. It also enables/disables various input/output buffers. See the register specification that follows for details of routing sources and targets.

3.7 LED Controller

The LED Controller allows various statuses to be reflected onto the front panel User LEDs. See the register specification for details on the available LED sources.

3.8 BPI Flash Controller

The FPGA reference design includes a BPI Flash Controller which enables the FPGA's BPI Flash to be reprogrammed via PCIe in the field and then trigger the FPGA reconfiguration. This mechanism can be used in place of a JTAG probe for FPGA upgrades.

3.9 Utility Functions

The FPGA includes various utility functions such as reporting its version number and signature, as well as providing a scratch register for bus testing.

4 Reference Software Overview

The software provided with the FPGA reference design targets the Linux operating system with Linux Kernel 2.6. It includes the following:

- amc511.ko: Device driver module for Linux 2.6 kernel
- amc511tool: Tool for controlling the driver/card
- fcf2coe: Filter coefficient conversion utility

The device driver's makefile should be customized to point at the kernel sources on the build machine prior to building it. The other tools should build without customization; however the amc511tool does require that development support for 'pcilib' be present. This dependency can be stripped out if support for the debugging commands is not needed.

The coefficient conversion utility aids in designing new filters for inclusion in a customized FPGA image. It converts Matlab FCF file format (single/double precision floating point) to Xilinx FIR Filter Compiler COE file format (floating point decimal). It can be helpful if Matlab was purchased with the Signal Processing/Filter Design toolkits but not the Fixed Point toolkit (which includes a COE export functionality). The Xilinx FIR Filter Compiler can automatically quantize floating point decimal COE files.

4.1 AMC511 Device Driver

The device driver is capable of supporting AMC511 cards attached to the PCIe bus of the host CPU. Each card will be assigned four minor device numbers in the order of probing. These minor device numbers represent A/D channels on the card. The global functionality of the card is accessible via any of the four channels and it is the responsibility of the application to ensure that the global settings are suitable for all four channels. The global settings include the Clock Synchronizer, Clock Routing, and LEDs.

The device driver supports PIO and DMA transfer of data. DMA is the default and recommended method since it is dramatically more efficent, but PIO can be used for low data rate applications or for debugging. A module parameter is available to force PIO mode. For example:

```
insmod amc511.ko force_pio=1
```

In order to reduce the system load as much as possible during data transfer, the device driver pre-allocates the DMA/PIO buffer chains when each card is probed and re-uses the descriptors/buffers over and over again. The allocation is done using single pages (scatter/gather) to reduce pressure on the kernel memory allocator.

The number of pages to use for each transfer can be specified using a module parameter. Examples:

```
insmod amc511.ko xfer_pages=128
insmod amc511.ko force_pio=1 xfer_pages=15
```

The maximum number of pages per transfer for DMA is 128 (512KB) while the maximum for PIO is 15 (60KB). Since the driver and card perform streaming using two descriptor chains (i.e. the card can be writing to memory in one chain while the application is reading from memory in the other chain), twice the specified amount of memory is allocated by the driver. So for the DMA case up to 1MB of memory can be dedicated to each A/D channel.

Generally the larger the value of xfer_pages, the higher the latency of the samples from A/D to application but the lower the system load. Conversely specifying a smaller value for xfer_pages reduces the latency but increases the system load. It is up to the end-user application requirements to determine the best trade-off.

The device driver optimizes data transfer to the user application by allowing the application to memory map the driver's buffer chains directly into the application's virtual memory map. Once this is done the two buffer chains appear as one contiguous region of application virtual memory. When used with DMA, this makes the transfer of A/D data from the card all the way into the user application a zero-copy operation from the CPU's point of view and results in extremely low CPU overhead. This leaves almost all of the CPU's time available for user application processing of the A/D data or other application behavior.

The device driver supports open, close, mmap, poll/select, and ioctl operations from the application. Generally the application should **`open()`** one or more channels as desired, then make an ioctl call to get information about the channel. This information includes the size of the mmap region to be used. Once the information is obtained, the application should **`mmap()`** the buffer region and then proceed to enable the A/D channel.

After the channel is enabled the application can call the **`poll()`** or **`select()`** system call to wait for data or status to become available. The application can use **`POLLPRI`** for status changes and/or **`POLLIN`** for new A/D data. If the driver indicates that the status changed then the application should read the status using the appropriate ioctl. If the driver indicates that data is available then the application should get information about the data using the appropriate ioctl. As mentioned before, the data is actually already in the application's virtual memory space, so this ioctl simply provides an offset and length within the mmaped region so that the application knows where to find it.

After this call is made, the indicated buffer memory is marked as reserved for application use by the driver and neither the driver nor the card will touch it. Once the application has read or processed the data, the buffer space must be returned to the driver using the appropriate ioctl. This once again makes the buffer space accessible by the card for the next transfer. Keep in mind that there are two descriptor chains, so the driver/card will be transferring data into the next chain while the application is reading the current chain, so the application should not access the buffer memory after the memory has been returned to

the driver. Depending on the host CPU type, if the application accesses the data buffers at the wrong time (even if it is only reading them) cache coherency may be lost resulting in data corruption.

The application should return buffer memory to the driver as soon as possible to avoid stalling the streaming transfer and risking overflow of the channel's FIFO. This is another tradeoff however, in that it is probably more efficient overall to process the data directly in the memory buffer and then return it to the driver rather than making a copy of it prior to processing. But each application is different and some experimentation may be necessary to tune the overall system performance. In either case, the card/driver provides all of the necessary status information to let the application know when/if the FIFO overflows so that performance tuning can be accomplished.

Once the capture is complete the application should disable the channel and/or **`close()`** the file handle. The driver will automatically disable the channel when the file handle is closed if it was not already disabled.

The **`ioctl()`** interfaces available for the device driver are documented in an appendix at the end of this document.

4.2 AMC511 Tool Application

The amc511tool provides basic support for controlling the card and gathering status as well as capturing A/D data. It can capture data to memory and discard it (to test the PCIe and memory bandwidth of the system) or it can capture it to a file.

The tool supports two distinct modes of operation. One mode uses the device driver interfaces (as would be recommended during typical activities). The other mode bypasses the device driver by mapping the PCIe BAR into user-space using 'pcilib' and reading/writing registers directly. This mode is for debugging only as it can potentially interfere with the device driver.

The usage information for the tool is shown below:

```
usage: amc511tool <cmd> [<opts>]

NORMAL CMDs: (use device driver)
        detect <dev>                            - Detect driver/card and report version info
        leds <dev> [<green0> <green1> <green2> <green3>
                    <amber0> <amber1> <amber2> <amber3>] - Show/set LED selectors
        routing <dev> [<sync_out> <tclka_out> <tclkb_out> <tclkc_out> <tclkd_out>
                <adc0_trigin> <adc1_trigin> <adc2_trigin> <adc3_trigin>
                <pri_ref> <sec_ref> <sync_out_en> <trig_in_en> <ref_clk_in_en>
                <tclkx_in_en> <tclka_out_en> <tclkb_out_en>
                <tclkc_out_en> <tclkd_out_en>] - Show/set clock routing
        clksync <dev> [primary|secondary]       - Show/set clock synchronizer settings
        status <dev> [monitor]                  - Show/monitor channel status flags
        capture <dev> <decm> <pat> <src> <pga> <rand> <dith>
                <limit> [<file>]                - Capture ADC data from channel
                                                  (limit in bytes, limit=0 for unlimited)
        rampverify <file>                       - Verify that samples in file are a ramp
        convert <binfile> <csvfile>             - Convert capture file to CSV file (for Matlab)
         bpi_program <dev> <bin_file>           - Program the FPGA's BPI flash
         bpi_id <dev>                           - Report BPI flash mfg/dev IDs
         bpi_dump <dev> [all]                   - Dump flash contents

DEBUG CMDs: (direct memory mapping of device/bypass device driver)
        adc <0-3> <decm> <pat> <src> [file]     - Start/Dump ADC channel 0-3 FIFO data
        ovrflw <0-3> <pga>                      - Monitor ADC channel 0-3 OVRFLW status
        dump [all|desc]                         - Dump the register contents
        write <addr> <val>                      - Write value at address
        profile [read|write]                    - Test PIO read/write speed
        test [scratch|sig]                      - Repeated Read/Write or Read tests
```

The options for the tool generally follow the field definitions in the register specification. So for instance the values which can be provided for **pat** match what is described in the CHCTRLx register's PAT field.

Note that the ADC channels always have their trigger logic enabled and will not start capturing data after the channel is enabled until they see a logical '1' on their internal trigger input. Therefore, if an external trigger signal is not being used, the **adcX_trigin** options for the **routing** command should be set to '1' to create an auto-trigger functionality.

The amc511tool source code can be used as a reference for the development of end-user applications.

5 System Performance Considerations

Capturing A/D data is a continuous real-time streaming operation that involves many aspects of the system and not just the AMC511 card. The AMC511 FPGA can actually process data rates faster than the rest of the system can handle. Four channel signal acquisition at 180 Msps produces about 1.44 GB/s worth of data in addition to necessary status/control register reads/writes.

Factors such as FPGA overhead, memory bandwidth, CPU speed/loading, data path, and application behavior will impact the sustainable streaming throughput. As an example it may be possible to write full rate data to a file if the capture is limited to a few megabytes, but if unconstrained the streaming will quickly stall and overflow the FIFO when the filesystem buffers and hard drive cache fill up due to the inability of the hard drive platters to sustain that rate. Therefore it is important to understand the performance characteristics of every piece of hardware/software in the complete data path to see where a bottleneck might be occurring. It is also important to keep in mind that sometimes the effects of the bottleneck might not be immediately apparent and will only show up after some amount of time such as the hard drive cache effect mentioned here.

For high sample rate applications the best approach is to customize the FPGA to do the DSP operations in an environment where cycle-accurate, real-time performance can be guaranteed and then transfer the resulting data at a greatly reduced data rate to the host memory/CPU via PCIe or other link. The reference design may not be fully optimized for your particular application; it is simply provided as a working example from which to branch off from with your own development.

5.1 Software Optimization Tips

The application should strive to put the data buffer back to the driver before it is actually needed by the card for the next DMA transfer. The driver/card will deal with either case seamlessly as long as the FIFO does not overflow as a result of the stall. The driver will post both buffer chains to the card whenever possible so that the DMA Burst Scheduler on the card does not stall. Whenever the DMA Burst Scheduler sees that the next chain is available upon completion of the current one it proceeds to process the next chain without direct software intervention. This greatly helps to avoid the cumulative and variable effects of software/interrupt latency on the on-going streaming transfer.

It is possible to route the FPGA's internal interrupt request line out to the front panel Sync output and view it on an oscilloscope or send it to a frequency counter. This technique works best for single channel acquisition but it can also be used under full load to get a feeling for the overall interrupt frequency. When the line is low the card is not requesting interrupt service and when it is high it is requesting interrupt service. By watching this line it is possible to interpret how well the system is responding to the card's needs.

By observing the width of the low-period while the tool application is acquiring and discarding data is it is possible to see how much time the user's own application can be afforded between transfers to process the data. This single-channel information can then be scaled according to how many channels the user's own application will actually be processing.

If the user's own application starts overflowing the FIFO then the extended low-period of the interrupt request line should be detectable as a result of the delay caused when the application didn't put the data buffer back in time to prevent a stall of the DMA engine. The difference between this and the previously established baseline should reveal how much the application is exceeding its allocated time to process the data.

By observing the width of the high-period it is possible to see if other device drivers or kernel activities are contributing to interrupt latency (by locking out interrupts). If the high-period shows a lot of variability or occasional very long high pulses this can have an adverse effect on the streaming since it will delay the notification of new data availability from the driver to the application and therefore reduce the amount of time the application has to process the data before the card needs that buffer back again. If undesirable interrupt latency is occurring, the solution is not to look at the amc511.ko driver (which has a fairly consistent interrupt processing and lockout time), but instead to look to other sources of interrupt latency which would actually be in the kernel-layer but could possibly be stimulated by user-space activity. Typical sources might be hard drive access, networking drivers, USB, serial ports, etc. Try to eliminate these sources of latency by either disabling the devices or not using them during signal acquisition to see if the interrupt latency shown by the AMC511 card is improved.

By customizing the FPGA it is also possible to add performance counters to measure such things as bus utilization, stall time, idle time, etc in a more direct manner.

6 FPGA Customization

The reference FPGA design combines IP cores from Xilinx with VadaTech custom VHDL code. This design can be customized or replaced by the customer to allow for high-end custom DSP solutions that are tailor-made to take full advantage of the considerable signal processing capabilities of the Xilinx Virtex-5 FPGA. In order to ease this effort, this section of the document describes some of the building blocks used in the FPGA reference design as well as some development techniques particular to the AMC511 board.

Having the VHDL sources available while reading this section is recommended to see the correspondence of various files or instances of cores mentioned here.

The following cores from Xilinx are used:

- PCIe x4 Endpoint Plus (ep)
  - Wraps the Virtex 5 embedded PCIe core
  - Provides the basic control/status/data interface for the reference design

- Tri-mode Ethernet Core in 1000Base-X mode (dual_enet_block)
  - Wraps the Virtex 5 embedded MAC core
  - Simply tests link establishment in the reference design

- Rocket I/O GTP Core (GTP_DUAL_1000X_inst)
  - Wraps the Virtex 5 GTP tile
  - Used as the 1000Base-X SERDES PHY in the reference design

- Block Memory Generator (chain0/1_ram)
  - Used for holding the ADC DMA descriptors

- FIFO Generator (adc_dcfifo_inst, main_to_adc_mailbox)
  - Used for holding the ADC sample data (clock-domain crossing/buffering)
  - Used for the ADC control mailbox (clock-domain crossing)

- FIR Compiler (fir_halfband_inst)
  - Used for the ADC half-band low-pass decimating filter

- Memory Interface Generator (sram_ctrl)
  - Used to generate the SRAM controller

It is expected that if the customer wishes to synthesize a new FPGA image, that they will have access to both the Xilinx ISE tool v11.3 or later (full version required to support the Virtex-5 chip on the board) as well as the necessary IP cores. Most of the IP cores are generic or else they are wrappers for Virtex-5 embedded blocks and therefore are included

with the ISE license at no extra charge from Xilinx. The Xilinx ISE tool license is not included as part of the purchase of the AMC511 card.

Refer to the applicable datasheets, user guides, and app notes from Xilinx for more details.

The VadaTech custom cores used in the reference design include:

- Top level wrapper (amc511_fpga.vhd)
  - Glues everything together at the top level
- ADC Controller/Pipeline w/ DMA Burst Scheduler (adc_core.vhd)
  - Controls each ADC and processes the dataflow from it
- PCIe to Local Bridge w/ DMA Burst Engine (pcie_local_bridge.vhd)
  - Provides PIO Read/Write and DMA Write capabilities
- DMA Burst Arbiter (dma_burst_arbiter.vhd)
  - Round-robin arbitration of the DMA Burst Engine between the four ADCs
- Clock Synchronizer Controller (clksync_core.vhd)
  - Controls the synchronizer chip via SPI and reports status
- Clock/Signal Router (clkrouter_core.vhd)
  - Flexibly routes clock/status/debug signals between places in the design
- LED Controller (led_core.vhd)
  - Controls the user LEDs
- BPI Flash Controller (bpi_flash.vhd)
  - Provides a simple interface to the BPI Flash bus for reprogramming
- General Purpose Utility Core (vt_utility_x32.vhd)
  - Provides version information and general utility functionality
- General Purpose Arbitrary Clock Enable Divider (vt_clocken_div_arbitrary.vhd)
  - Provides a re-usable way to slow down portions of the design while still using the same clock
- General Purpose Interrupt Controller (vt_interrupt_controller_x32.vhd)
  - Provides a re-usable way to implement an interrupt controller for the design
- General Purpose Reset Synchronizer (vt_reset_sync.vhd)
  - Provides a re-usable asynchronous assertion/synchronous de-assertion reset

- General Purpose Clock Synchronizer (vt_multi_sync.vhd)
    - Provides a re-usable way to get individual signals from one clock domain to another to mitigate meta-stability

LEGAL NOTICE: The VadaTech custom VHDL code included in this reference design is the intellectual property of VadaTech Incorporated. Permission is granted to use the VadaTech custom VHDL code royalty-free in customer designs targeting the VadaTech AMC511 card only. Redistribution to third parties or use of this code for any other purpose is strictly prohibited.

It is possible to make slight changes to the design such as replacing the provided half-band FIR filter with your own custom FIR filter, etc or to completely replace the design. At a minimum the UCF file will be a useful starting point since it captures the pin-out of the FPGA chip on the AMC511 card. The provided source files were used to create the flash image that is shipped on the board from VadaTech. The MCS file used is also included so that the original BPI flash contents can be replaced after experimentation to restore the original board functionality.

A Xilinx Platform Cable USB II (or other compatible JTAG probe) is required to program the FPGA and/or BPI flash. During development it is possible to save time by only programming the FPGA. However, with a PCIe-based design it is required to soft-reset the system in order for the PCIe core to get re-initialized by the CPU. Doing a hard-reset will load the contents of the BPI flash and will wipe out any previous FPGA configuration that may have been done via JTAG. There is no special switch to select BPI vs. JTAG mode, the board is always in BPI mode but the JTAG can reconfigure the FPGA at any time. So the usual development sequences would be:

Developing using BPI Flash to load the FPGA:

1) Power the system (FPGA will load whatever is already in BPI flash)
2) Program the BPI flash with the MCS file
3) Power-cycle the system

Developing using FPGA only:

1) Power the system (FPGA will load whatever is already in BPI flash)
2) Program the FPGA with the BIT file (previous configuration will be overwritten for current power-cycle only)
3) Soft-reset the system (i.e. CTRL-ALT-DELETE or 'reboot' Linux command) to cause the PCIe BARs to be re-detected/re-initialized.

VadaTech is not responsible for damage or loss caused by reprogramming of the FPGA or BPI Flash by the customer. Use caution when changing the reference design or creating your own design as it is possible to damage the board or components.

7 Customizing the VCXO / Ref Clock Configuration

The AMC511 reference design code ships out with a default VCXO (A/D sample clock) rate of 180 MHz and a default Ref Clock rate of 10 MHz. However, these parts (Y4 and Y3 respectively on the schematic) may be changed out by Vadatech at the customer's request to better suit their target application.

To help facilitate this board-level change there are conditional compile options in both the reference FPGA code and the device driver code. These conditional compile options change the N/M divider configuration in the clock synchronizer PLL (refer to the SYNCSPI0 register in the following section). Using incorrect settings will prevent the PLL from locking and may result in an unstable A/D sample clock.

The following configurations are currently supported by the reference design. If your application requires a different combination of oscillators please contact VadaTech Sales.

VCXO (Y4)	Ref Clock (Y3)	N Divider	M Divider
180 MHz	10 MHz	990	55
107.52 MHz	10 MHz	1344	125

Table 12: Supported VCXO / Ref Clock frequencies

To change the hardware default N/M configuration which the FPGA applies immediately upon configuration, look for the following compile options in the FPGA VHDL reference design's clksync_core.vhd file and set exactly one of them to 1 and the other to 0 and then re-generate the programming file:

```
constant VCXO180_REF10     : integer range 0 to 1  := 1;
constant VCXO107_52_REF10  : integer range 0 to 1  := 0;
```

To change the device driver's card initialization values, edit the device driver's amc511.c file and look for the following compile options and set exactly one of them to 1 and the other to 0 then rebuild the driver:

```
#define VCXO180_REF10       1
#define VCXO107_52_REF10    0
```

It is recommended that both the FPGA and device driver defaults be changed to match the oscillator configuration of your particular board. However, if changing the FPGA is not practical then just changing the device driver should be sufficient since it overrides the FPGA defaults prior to any actual data acquisition.

8 Appendix A: FPGA Register Specification

The AMC511 Reference Design FPGA's registers are available via a PCIe 64KB non-prefetchable memory-mapped region (BAR 0). This region responds to byte-enabled 32-bit memory reads and writes; meaning that BYTE, WORD, and DWORD accesses are permitted. The layout of this region is shown below:

Core	BAR Offset	Mnemonic	Description
Interrupt Controller	0x0000	GIMSR	Global Interrupt Mask Set Register
	0x0004	GIMCR	Global Interrupt Mask Clear Register
	0x0008	GISR	Global Interrupt Status Register
	0x000C	BCSR	Bit-Change Status Register
	0x0010 – 0x0FFF	N/A	Reserved
A/D Channel 0	0x1000	CHCTRL0	Channel 0 Control Register
	0x1004	CHSTAT0	Channel 0 Status Register
	0x1008	CHMASK0	Channel 0 Mask Register
	0x100C	CHHWM0	Channel 0 High Water Mark Register
	0x1010	CHLVL0	Channel 0 Level Register
	0x1014	CHDMACTRL0	Channel 0 DMA Control/Status Register
	0x1018	CHDMACOUNT0	Channel 0 DMA Count Register
	0x101C	CHDEBUGA0	Channel 0 Debug Register A
	0x1020	CHDEBUGB0	Channel 0 Debug Register B
	0x1024	CHDEBUGC0	Channel 0 Debug Register C
	0x1028 – 0x13FF	N/A	Reserved
	0x1400 – 0x17FF	CHFIFO0	Channel 0 FIFO Read Port
	0x1800 – 0x1FFF	CHDMADESC0	Channel 0 DMA Descriptors Port

(continued)

Core	BAR Offset	Mnemonic	Description
A/D Channel 1	0x2000	CHCTRL1	Channel 1 Control Register
	0x2004	CHSTAT1	Channel 1 Status Register
	0x2008	CHMASK1	Channel 1 Mask Register
	0x200C	CHHWM1	Channel 1 High Water Mark Register
	0x2010	CHLVL1	Channel 1 Level Register
	0x2014	CHDMACTRL1	Channel 1 DMA Control/Status Register
	0x2018	CHDMACOUNT1	Channel 1 DMA Count Register
	0x201C	CHDEBUGA1	Channel 1 Debug Register A
	0x2020	CHDEBUGB1	Channel 1 Debug Register B
	0x2024	CHDEBUGC1	Channel 1 Debug Register C
	0x2028 – 0x23FF	N/A	Reserved
	0x2400 – 0x27FF	CHFIFO1	Channel 1 FIFO Read Port
	0x2800 – 0x2FFF	CHDMADESC1	Channel 1 DMA Descriptors Port
A/D Channel 2	0x3000	CHCTRL2	Channel 2 Control Register
	0x3004	CHSTAT2	Channel 2 Status Register
	0x3008	CHMASK2	Channel 2 Mask Register
	0x300C	CHHWM2	Channel 2 High Water Mark Register
	0x3010	CHLVL2	Channel 2 Level Register
	0x3014	CHDMACTRL2	Channel 2 DMA Control/Status Register
	0x3018	CHDMACOUNT2	Channel 2 DMA Count Register
	0x301C	CHDEBUGA2	Channel 2 Debug Register A
	0x3020	CHDEBUGB2	Channel 2 Debug Register B
	0x3024	CHDEBUGC2	Channel 2 Debug Register C
	0x3028 – 0x33FF	N/A	Reserved
	0x3400 – 0x37FF	CHFIFO2	Channel 2 FIFO Read Port
	0x3800 – 0x3FFF	CHDMADESC2	Channel 2 DMA Descriptors Port
A/D Channel 3	0x4000	CHCTRL3	Channel 3 Control Register
	0x4004	CHSTAT3	Channel 3 Status Register
	0x4008	CHMASK3	Channel 3 Mask Register
	0x400C	CHHWM3	Channel 3 High Water Mark Register
	0x4010	CHLVL3	Channel 3 Level Register
	0x4014	CHDMACTRL3	Channel 3 DMA Control/Status Register
	0x4018	CHDMACOUNT3	Channel 3 DMA Count Register
	0x401C	CHDEBUGA3	Channel 3 Debug Register A
	0x4020	CHDEBUGB3	Channel 3 Debug Register B
	0x4024	CHDEBUGC3	Channel 3 Debug Register C
	0x4028 – 0x43FF	N/A	Reserved
	0x4400 – 0x47FF	CHFIFO3	Channel 3 FIFO Read Port
	0x4800 – 0x4FFF	CHDMADESC3	Channel 3 DMA Descriptors Port
Clock Synchronizer	0x5000	SYNCCTRL	Synchronizer Control Register
	0x5004	SYNCSPI0	Synchronizer SPI Data 0
	0x5008	SYNCSPI1	Synchronizer SPI Data 1
	0x500C	SYNCSPI2	Synchronizer SPI Data 2
	0x5010	SYNCSPI3	Synchronizer SPI Data 3
	0x5014 – 0x5FFF	N/A	Reserved

(continued)

Core	BAR Offset	Mnemonic	Description
Clock Router	0x6000	CRSYNC	Clock Router Sync Out Source Register
	0x6004	CRTCLKA	Clock Router TCLKA Source Register
	0x6008	CRTCLKB	Clock Router TCLKB Source Register
	0x600C	CRTCLKC	Clock Router TCLKC Source Register
	0x6010	CRTCLKD	Clock Router TCLKD Source Register
	0x6014	CRTRIGIN1	Clock Router ADC 0 Trig Input Source Register
	0x6018	CRTRIGIN2	Clock Router ADC 1 Trig Input Source Register
	0x601C	CRTRIGIN3	Clock Router ADC 2 Trig Input Source Register
	0x6020	CRTRIGIN4	Clock Router ADC 3 Trig Input Source Register
	0x6024	CRPRIREF	Clock Router Primary Reference Source Register
	0x6028	CRSECREF	Clock Router Secondary Reference Source Register
	0x602C	CREN	Clock Router Enables Register
	0x6030 – 0x6FFF	N/A	Reserved
LED	0x7000	GREENLED0	Channel 0 Green Status LED Register
	0x7004	GREENLED1	Channel 1 Green Status LED Register
	0x7008	GREENLED2	Channel 2 Green Status LED Register
	0x700C	GREENLED3	Channel 3 Green Status LED Register
	0x7010	AMBERLED0	Channel 0 Amber Status LED Register
	0x7014	AMBERLED1	Channel 1 Amber Status LED Register
	0x7018	AMBERLED2	Channel 2 Amber Status LED Register
	0x701C	AMBERLED3	Channel 3 Amber Status LED Register
	0x7020 – 0x7FFF	N/A	Reserved
DMA Arbiter	0x8000	ARBSTAT	DMA Arbiter Status Register
	0x8004 – 0x8FFF	N/A	Reserved
BPI Flash	0x9000	BPICTRL	BPI Flash Control Register
	0x9004	BPIADDR	BPI Flash Address Register
	0x9008	BPIDATA	BPI Flash Data Register
	0x900C – 0x9FFF	N/A	Reserved
N/A	0xA000 – 0xEFFF	N/A	Reserved
Utility	0xF000 – 0xFFF3	N/A	Reserved
	0xFFF4	SCRATCH	Scratch Register
	0xFFF8	VER	Version Register
	0xFFFC	SIG	Signature Register

Table 13: FPGA register map

The following pages describe the registers in detail. In these register descriptions certain field behavior tags are used. The following is a list of those behaviors:

RO - Read Only
R/W - Read/Write
R/W1C - Read/Write 1 to Clear
R/W1S - Read/Write 1 to Set
R/W1SC - Read/Write 1 Self-Clearing

8.1 GIMSR – Global Interrupt Mask Set Register

Address: 0x0000

	31	30	29	28	27	26	25	24
Field	BCI	CSI	Rsvd					
Access	R/W1S	R/W1S	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd				ADC3	ADC2	ADC1	ADC0
Access	RO				R/W1S	R/W1S	R/W1S	R/W1S
Reset	X				0	0	0	0

Bit(s)	Field	Description
31	BCI	**Read:** Bit Change Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the Bit Change Interrupt enabled. Writing '0' has no effect.
30	CSI	**Read:** Clock Synchronizer Interrupt (SPI_DONE=1) is enabled when '1' else it is disabled. **Write:** Writing '1' sets the Clock Synchronizer Interrupt Enabled. Writing '0' has no effect.
3	ADC3	**Read:** ADC3 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the ADC3 Interrupt enabled. Writing '0' has no effect.
2	ADC2	**Read:** ADC2 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the ADC2 Interrupt enabled. Writing '0' has no effect.
1	ADC1	**Read:** ADC1 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the ADC1 Interrupt enabled. Writing '0' has no effect.
0	ADC0	**Read:** ADC0 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' sets the ADC0 Interrupt enabled. Writing '0' has no effect.

8.2 GIMCR – Global Interrupt Mask Clear Register

Address: 0x0004

	31	30	29	28	27	26	25	24
Field	BCI	CSI	Rsvd					
Access	R/W1C	R/W1C	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd				ADC3	ADC2	ADC1	ADC0
Access	RO				R/W1C	R/W1C	R/W1C	R/W1C
Reset	X				0	0	0	0

Bit(s)	Field	Description
31	BCI	**Read:** Bit Change Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' disables the Bit Change Interrupt. Writing '0' has no effect.
30	CSI	**Read:** Clock Synchronizer Interrupt (SPI_DONE=1) is enabled when '1' else it is disabled. **Write:** Writing '1' disables the Clock Synchronizer Interrupt. Writing '0' has no effect.
3	ADC3	**Read:** ADC3 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' disables the ADC3 Interrupt. Writing '0' has no effect.
2	ADC2	**Read:** ADC2 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' disables the ADC2 Interrupt. Writing '0' has no effect.
1	ADC1	**Read:** ADC1 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' disables the ADC1 Interrupt. Writing '0' has no effect.
0	ADC0	**Read:** ADC0 Interrupt is enabled when '1' else it is disabled. **Write:** Writing '1' disables the ADC0 Interrupt. Writing '0' has no effect.

8.3 GISR – Global Interrupt Status Register

Address: 0x0008

	31	30	29	28	27	26	25	24
Field	BCI	CSI	Rsvd					
Access	RO	RO	RO					
Reset	0	0	X					

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd				ADC3	ADC2	ADC1	ADC0
Access	RO				RO	RO	RO	RO
Reset	X				0	0	0	0

Bit(s)	Field	Description
31	BCI	Bit Change Interrupt is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the BCI is no longer asserted by the BCSR register.
30	CSI	Clock Synchronizer Interrupt (SPI_DONE=1) is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the CSI is no longer asserted by the Clock Synchronizer core.
3	ADC3	ADC3 Interrupt is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the ADC3 Interrupt is no longer asserted by the A/D 3 core.
2	ADC2	ADC2 Interrupt is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the ADC2 Interrupt is no longer asserted by the A/D 2 core.
1	ADC1	ADC1 Interrupt is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the ADC1 Interrupt is no longer asserted by the A/D 1 core.
0	ADC0	ADC0 Interrupt is pending when '1', and is therefore interrupting the CPU. This bit will clear automatically when the ADC0 Interrupt is no longer asserted by the A/D 0 core.

8.4 BCSR – Bit Change Status Register

Address: 0x000C

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd					PLL_LOCK	STA_VCXO	STA_REF
Access	RO					R/W1C	R/W1C	R/W1C
Reset	X					0	0	0

Bit(s)	Field	Description
2	PLL_LOCK	**Read:** The Clock Synchronizer's PLL_LOCK indication changed state when this bit is '1', else it did not change state. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
1	STA_VCXO	**Read:** The Clock Synchronizer's STA_VCXO indication changed state when this bit is '1', else it did not change state. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
0	STA_REF	**Read:** The Clock Synchronizer's STA_REF indication changed state when this bit is '1', else it did not change state. **Write:** Writing '1' clears this bit. Writing '0' has no effect.

8.5 CHCTRLx – Channel X Control Register

Addresses: 0x1000, 0x2000, 0x3000, 0x4000

	31	30	29	28	27	26	25	24
Field	Rsvd					PAT		
Access	RO					R/W		
Reset	X					0		

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	DECM							
Access	R/W							
Reset	0							

	7	6	5	4	3	2	1	0
Field	Rsvd		SRC		PGA	RAND	DITH	ENABLE
Access	RO		R/W		R/W	R/W	R/W	R/W
Reset	X		0		0	1	1	0

Bit(s)	Field	Description
26-24	PAT	Specifies the pattern generator's output pattern from the following list: 0x0 – Generate all '0' bits. 0x1 – Generate all '1' bits. 0x2 – Generate most negative signed 16-bit value. 0x3 – Generate most positive signed 16-bit value. 0x4 – Generate repeating ramp from most negative signed 16-bit value to most positive signed 16-bit value, incrementing by one. If the decimator is turned on the ramp will appear to increment by DECM+1 due to the decimation. 0x5-0x7 – Reserved
15-8	DECM	Specifies the decimation 'M-1' value. A value of 0x00 in this field decimates by a factor of 1 (no decimation), while a value of 0xFF decimates by a factor of 256.
5-4	SRC	Specifies the source of data for the channel FIFO from the following list: 0x0 – Raw A/D samples 0x1 – FIR filtered A/D samples 0x2 – Pattern Generator samples 0x3 – Reserved
3	PGA	When set to '1' turns on the A/D chip's Programmable Gain Amplifier (1.5x Gain), else keeps it off.
2	RAND	When set to '1' turns on the A/D chip's Randomizer function which can reduce EMI emissions. This also turns on the FPGA's De-randomizer pipeline stage so that the effect is transparent to the data set.
1	DITH	When set to '1' turns on the A/D chip's Dither function.
0	ENABLE	When set to '1' enables the A/D chip as well as the FPGA's A/D pipeline for this channel, else it is disabled which can save power.

8.6 CHSTATx – Channel X Status Register

Addresses: 0x1004, 0x2004, 0x3004, 0x4004

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd						DMA1DONE	DMA0DONE
Access	RO						R/W1C	R/W1C
Reset	X						0	0

	7	6	5	4	3	2	1	0
Field	Rsvd		FIROVR	AOVR	FOVR	FUND	FHWM	RUNNING
Access	RO		R/W1C	R/W1C	R/W1C	R/W1C	RO	RO
Reset	X		0	0	0	0	0	0

Bit(s)	Field	Description
9	DMA1DONE	**Read:** The DMA controller has finished transferring chain 1. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
8	DMA0DONE	**Read:** The DMA controller has finished transferring chain 0. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
5	FIROVR	**Read:** FIR filter overflow occurred when '1' else it didn't occur. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
4	AOVR	**Read:** A/D Chip OVRFLW occurred when '1' else it didn't occur. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
3	FOVR	**Read:** FIFO overflow occurred when '1' else it didn't occur. FIFO overflow occurs when the A/D pipeline drops one or more samples because the FIFO was already full. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
2	FUND	**Read:** FIFO underflow occurred when '1' else it didn't occur. FIFO underflow occurs when the host or DMA engine reads the FIFO when it is already empty. **Write:** Writing '1' clears this bit. Writing '0' has no effect.
1	FHWM	The FIFO High Water Mark indication is asserted when '1' else it is not.
0	RUNNING	The A/D pipeline is processing data when '1' else it is not.

NOTE: An ADC core interrupt is asserted to the GISR register whenever any bit is set in this register that is also set in the CHMASK register.

A channel 'fault' output is created internally for use by the LED controller which is the logical 'or' of FIROVR, AOVR, FOVR, and FUND. The LED controller stretches this signal to ensure that it is always visible for a minimum period of time to the human eye.

8.7 CHMASKx – Channel X Mask Register

Addresses: 0x1008, 0x2008, 0x3008, 0x4008

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd						DMA1DONE	DMA0DONE
Access	RO						R/W	R/W
Reset	X						0	0

	7	6	5	4	3	2	1	0
Field	Rsvd		FIROVR	AOVR	FOVR	FUND	FHWM	RUNNING
Access	RO		R/W	R/W	R/W	R/W	R/W	R/W
Reset	X		0	0	0	0	0	0

Bit(s)	Field	Description
9	DMA1DONE	When set to '1' allows the DMA1DONE status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
8	DMA0DONE	When set to '1' allows the DMA0DONE status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
4	FIROVR	When set to '1' allows the FIROVR status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
4	AOVR	When set to '1' allows the AOVR status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
3	FOVR	When set to '1' allows the FOVR status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
2	FUND	When set to '1' allows the FUND status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
1	FHWM	When set to '1' allows the FHWM status bit to assert an interrupt to the GISR register, else the interrupt is blocked.
0	RUNNING	When set to '1' allows the RUNNING status bit to assert an interrupt to the GISR register, else the interrupt is blocked.

8.8 CHHWMx – Channel X High Water Mark Register

Addresses: 0x100C, 0x200C, 0x300C, 0x400C

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd			HWM[12:8]				
Access	RO			R/W				
Reset	X			(see below)				

	7	6	5	4	3	2	1	0
Field	HWM[7:0]							
Access	R/W							
Reset	0x200							

Bit(s)	Field	Description
12-0	HWM	Sets the level above which the FIFO High Water Mark indication should be asserted in the range 0x000 – 0x1FFF.

8.9 CHLVLx – Channel X Level Register

Addresses: 0x1010, 0x2010, 0x3010, 0x4010

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd		LEVEL[13:8]					
Access	RO		RO					
Reset	X		(see below)					

	7	6	5	4	3	2	1	0
Field	LEVEL[7:0]							
Access	RO							
Reset	0x0000							

Bit(s)	Field	Description
13-0	LEVEL	Shows the current level of the FIFO (each FIFO entry contains 4 samples) in the range of 0x0000-0x2001.

8.10 CHDMACTRLx – Channel X DMA Control/Status Register

Addresses: 0x1014, 0x2014, 0x3014, 0x4014

	31	30	29	28	27	26	25	24
Field	DMAPTP	Rsvd	STATE		Rsvd		BURST_LEN[9:8]	
Access	R/W	RO	RO		RO		RO	
Reset	0	X	0x0		X		X	

	23	22	21	20	19	18	17	16
Field	BURST_LEN[7:0]							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	CURCHAIN	CURDESC						
Access	RO	RO						
Reset	0	0x00						

	7	6	5	4	3	2	1	0
Field	Rsvd						CHAIN1RDY	CHAIN0RDY
Access	RO						R/W1SC	R/W1SC
Reset	X						0	0

Bit(s)	Field	Description
31	DMAPTP	Specifies whether Point-To-Point DMA mode should be enabled. When this bit is '1' the DMA controller will assume that it is pushing samples to another PCIe card and not memory. It will therefore not clear the CHAINxRDY bits and will continue to cycle through both chains endlessly. Otherwise when this bit is '0' it performs normal DMA to memory behavior where the CHAINxRDY bits are cleared so that ownership of the buffer chain transfers back to the device driver after the DMA completes.
29-28	STATE	Shows the current DMA Burst Scheduler state for diagnostic purposes as follows: 0x0: Idle 0x1: Descriptor Fetch 0x2: Descriptor Processing 0x3: Waiting for end of chain burst completion
25-16	BURST_LEN	Shows the number of bytes that will be transferred in each DMA burst for diagnostic purposes. This value is decoded from the PCIe Configuration Device Control Register's Max Payload Size field. The possible values are 128, 256, or 512.
15	CURCHAIN	Shows whether chain 0 or chain 1 is the current chain that the DMA engine will process. This bit resets back to '0' whenever the channel is disabled.
14-8	CURDESC	Shows the current descriptor within the current chain that the DMA engine will process. This field resets back to 0x00 whenever the channel is disabled.

1	CHAIN1RDY	A '1' bit is written to this field whenever the device driver wants to enable the DMA channel to process chain 1. Effectively it passes ownership from the software to the hardware for the given chain. The DMA engine will process it immediately only if it is also the CURCHAIN, otherwise it will wait in a ready state until the CURCHAIN is done. In normal DMA mode the controller will clear this bit when the chain has been transferred, but in point-to-point DMA mode the controller will not clear this bit and will continue to re-use the chains. In either case, the DMA1DONE bit in the status register will be set upon transfer completion. Writing a '0' to this field has no effect. This bit resets back to '0' whenever the channel is disabled.
0	CHAIN0RDY	A '1' bit is written to this field whenever the device driver wants to enable the DMA channel to process chain 0. Effectively it passes ownership from the software to the hardware for the given chain. The DMA engine will process it immediately only if it is also the CURCHAIN, otherwise it will wait in a ready state until the CURCHAIN is done. In normal DMA mode the controller will clear this bit when the chain has been transferred, but in point-to-point DMA mode the controller will not clear this bit and will continue to re-use the chains. In either case, the DMA0DONE bit in the status register will be set upon transfer completion. Writing a '0' to this field has no effect. This bit resets back to '0' whenever the channel is disabled.

8.11 CHDMACOUNTx – Channel X DMA Count Register

Addresses: 0x1018, 0x2018, 0x3018, 0x4018

	31	30	29	28	27	26	25	24
Field	BURSTS[11:4]							
Access	RO							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	BURSTS[3:0]				BYTES[19:16]			
Access	RO				RO			
Reset	0x000				(see below)			

	15	14	13	12	11	10	9	8
Field	BYTES[15:8]							
Access	RO							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	BYTES[7:0]							
Access	RO							
Reset	0x00000							

Bit(s)	Field	Description
31-20	BURSTS	This field shows the total number of burst transfers (128, 256, or 512 bytes) completed by the DMA engine while processing the current descriptor in the range of 0x000 – 0xFFF. It is for diagnostic use only.
19-0	BYTES	This field shows the total number of bytes transferred by the DMA engine while processing the current descriptor in the range of 0x00000 – 0x80000. It is for diagnostic use only.

8.12 CHDEBUGAx – Channel X Debug A Register

Addresses: 0x101C, 0x201C, 0x301C, 0x401C

	31	30	29	28	27	26	25	24
Field	Rsvd					PAT_A		
Access	RO					RO		
Reset	X					0		

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	DECM_A							
Access	RO							
Reset	0							

	7	6	5	4	3	2	1	0
Field	Rsvd		SRC_A		PGA_A	RAND_A	DITH_A	RUNNING_A
Access	RO		RO		RO	RO	RO	RO
Reset	X		0		0	1	1	0

Bit(s)	Field	Description
26-24	PAT_A	Shows the value of PAT in the A/D clock domain for diagnostic purpose. *(Async)*
15-8	DECM_A	Shows the value of DECM in the A/D clock domain for diagnostic purpose. *(Async)*
5-4	SRC_A	Shows the value of SRC in the A/D clock domain for diagnostic purpose. *(Async)*
3	PGA_A	Shows the value of PGA in the A/D clock domain for diagnostic purpose. *(Async)*
2	RAND_A	Shows the value of RAND in the A/D clock domain for diagnostic purpose. *(Async)*
1	DITH_A	Shows the value of DITH in the A/D clock domain for diagnostic purpose. *(Async)*
0	RUNNING_A	Shows the value of RUNNING in the A/D clock domain for diagnostic purpose. *(Async)*

NOTE: Debug fields flagged as '*Async*' may display meta-stability effects resulting in unpredictable read values. They are mostly of use during debugging of possible stall conditions where meta-stability isn't a factor since the values would not be changing asynchronously. They should be viewed with a very skeptical eye if the pipeline is actively running and the values are changing rapidly.

8.13 CHDEBUGBx – Channel X Debug B Register

Addresses: 0x1020, 0x2020, 0x3020, 0x4020

	31	30	29	28	27	26	25	24
Field	PIPERST	SHDN	Rsvd	MBOX_FULL	Rsvd		FIR_RFD	FIR_RDY
Access	RO	RO	RO	RO	RO		RO	RO
Reset	1	1	X	0	X		0	0

	23	22	21	20	19	18	17	16
Field	PIPERST_A	MBOXRST	RSTSYNCRST	MBOX_EMPTY	CAPRDY	FMTRDY	SRCRDY	CHCLK
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	0	0	0	X

	15	14	13	12	11	10	9	8
Field	OVRFLW	PACKLANE		HWM_A[12:8]				
Access	RO	RO		RO				
Reset	0	0		(see below)				

	7	6	5	4	3	2	1	0
Field	HWM_A[12:8]							
Access	RO							
Reset	0x200							

Bit(s)	Field	Description
31	PIPERST	Shows the value of the pipeline reset line in the Main clock domain for diagnostic purpose.
30	SHDN	Shows the value of the SHDN line for diagnostic purpose.
28	MBOX_FULL	Shows the value of the mailbox full line for diagnostic purpose.
25	FIR_RFD	Shows the value of the FIR filter's Ready For Data line for diagnostic purpose. *(Async)*
24	FIR_RDY	Shows the value of the FIR filter's output data Ready line for diagnostic purpose. *(Async)*
23	PIPERST_A	Shows the value of the pipeline reset line in the A/D clock domain for diagnostic purpose. *(Async)*
22	MBOXRST	Shows the value of the mailbox reset line for diagnostic purpose. *(Async)*
21	RSTSYNCRST	Shows the value of the reset synchronizer reset (effectively holds mailbox/pipeline in reset until DCM is locked to A/D channel clock) for diagnostic purpose. *(Async)*
20	MBOX_EMPTY	Shows the value of the mailbox empty line for diagnostic purpose. *(Async)*
19	CAPRDY	Shows the value of the pipeline Captured stage ready line for diagnostic purpose. *(Async)*
18	FMTRDY	Shows the value of the pipeline Formatted stage ready line for diagnostic purpose. *(Async)*
17	SRCRDY	Shows the value of the pipeline Sourced stage ready line for diagnostic purpose. *(Async)*
16	CHCLK	Shows the value of the channel clock line for diagnostic purpose. *(Async)*
15	OVRFLW	Shows the value of the captured overflow line for diagnostic purpose. *(Async)*
14-13	PACKLANE	Shows the value of the pipeline Packed stage lane selector for diagnostic purpose. *(Async)*
12-0	HWM_A	Shows the value of HWM in the A/D clock domain for diagnostic purpose. *(Async)*

NOTE: Debug fields flagged as '*Async*' may display meta-stability effects resulting in unpredictable read values. They are mostly of use during debugging of possible stall conditions where meta-stability isn't a factor since the values would not be changing asynchronously. They should be viewed with a very skeptical eye if the pipeline is actively running and the values are changing rapidly.

8.14 CHDEBUGCx – Channel X Debug C Register

Addresses: 0x1024, 0x2024, 0x3024, 0x4024

	31	30	29	28	27	26	25	24
Field	CH_DATA[15:8]							
Access	RO							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	CH_DATA[7:0]							
Access	RO							
Reset	0x0000							

	15	14	13	12	11	10	9	8
Field	PAT_DATA[15:8]							
Access	RO							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	PAT_DATA[7:0]							
Access	RO							
Reset	0x0000							

Bit(s)	Field	Description
31-16	CH_DATA	Shows the value of the captured A/D data lines for diagnostic purpose. Note that this is before the Formatted stage so the data will be randomized if the RAND bit is '1'. *(Async)*
15-0	PAT_DATA	Shows the value of the pattern generator data for diagnostic purpose. *(Async)*

NOTE: Debug fields flagged as '*Async'* may display meta-stability effects resulting in unpredictable read values. They are mostly of use during debugging of possible stall conditions where meta-stability isn't a factor since the values would not be changing asynchronously. They should be viewed with a very skeptical eye if the pipeline is actively running and the values are changing rapidly.

8.15 CHFIFOx – Channel X FIFO Read Port

Addresses: 0x1400-17FF, 0x2400-27FF, 0x3400-37FF, 0x4400-47FF

	63	62	61	60	59	58	57	56
Field	SAMPLE1[15:8]							
Access	RO							
Reset	X							

	55	54	53	52	51	50	49	48
Field	SAMPLE1[7:0]							
Access	RO							
Reset	X							

	47	46	45	44	43	42	41	40
Field	SAMPLE0[15:8]							
Access	RO							
Reset	X							

	39	38	37	36	35	34	33	32
Field	SAMPLE0[7:0]							
Access	RO							
Reset	X							

	31	30	29	28	27	26	25	24
Field	SAMPLE3[15:8]							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	SAMPLE3[7:0]							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	SAMPLE2[15:8]							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	SAMPLE2[7:0]							
Access	RO [SIDE-EFFECT]							
Reset	X							

Bit(s)	Field	Description
63-48	SAMPLE1	Sample index 2 within the current FIFO entry.
47-32	SAMPLE0	Sample index 2 within the current FIFO entry.
31-16	SAMPLE3	Sample index 3 within the current FIFO entry.
15-0	SAMPLE2	Sample index 2 within the current FIFO entry. Reading the least significant byte of this sample has the side-effect of popping the FIFO so that the next FIFO entry is reflected in this register.

FIFO READ PORT NOTES:

- The FIFO read port is 64-bits wide unlike the other registers which are 32-bits wide. This is so that the DMA engine can process the sample data at the full bandwidth of the 64-bit PCIe core in the FPGA.
- The FIFO read port register is mirrored multiple times over the specified range. This allows for optimized memcpy style access from the CPU (the internal DMA engine also bursts from the FIFO using this technique).
- PIO reads can only read 32-bits of this port at a time.
- Even DWORD addresses read the upper 32-bits while odd DWORD addresses read the lower 32-bits.
- Reading the LSB of an odd DWORD causes the FIFO to de-queue and present the next entry. Therefore the expected read pattern is to read the even word then the odd word [FIFO then presents the next entry], then read the next even word and next odd word, etc.
- The DMA engine reads this port 64-bits at a time.

The samples are positioned into the FIFO entry such that by the time they reach the CPU's main memory they are in Little-endian 16-bit byte order. This occurs as follows:

1) The samples are staged into the entry as shown in the register description above.
2) The PCIe bus bridging logic converts the BE32 representation in the FPGA to LE32 representation.
3) The CPU interprets the data as LE16 (signed short) samples.

The following table shows the sample/byte order transformations that occur as data flows from the FPGA to the CPU:

Sample/Byte Order	0	1	2	3	4	5	6	7
FIFO order (native 64)	1A	1B	0A	0B	3A	3B	2A	2B
Bridge order (LE32)	0B	0A	1B	1A	2B	2A	3B	3A
CPU (native 16)	0A	0B	1A	1B	2A	2B	3A	3B

8.16 CHDMADESCx – Channel X DMA Descriptors Port

Addresses: 0x1800-1FFF, 0x2800-2FFF, 0x3800-3FFF, 0x4800-4FFF

	63	62	61	60	59	58	57	56
Field	ADDR[31:24]							
Access	R/W							
Reset	(see below)							

	55	54	53	52	51	50	49	48
Field	ADDR[23:16]							
Access	R/W							
Reset	(see below)							

	47	46	45	44	43	42	41	40
Field	ADDR[15:9]							Rsvd
Access	R/W							RO
Reset	0x000000							X

	39	38	37	36	35	34	33	32
Field	Rsvd							
Access	RO							
Reset	X							

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd				LENGTH[19:16]			
Access	RO				R/W			
Reset	X				(see below)			

	15	14	13	12	11	10	9	8
Field	LENGTH[15:9]							Rsvd
Access	R/W							RO
Reset	0x000							X

	7	6	5	4	3	2	1	0
Field	Rsvd							LAST
Access	RO							R/W
Reset	X							X

Bit(s)	Field	Description
63-41	ADDR	The upper bits of the destination bus address for the descriptor. This address must be at least 512 byte aligned since this is the largest possible burst size.
19-9	LENGTH	The upper bits of the length in bytes for the descriptor. This length must be a multiple of 512 bytes (since this is the largest possible burst size) and non-zero. The actual burst size is determined by the PCIe Configuration: Max Payload as set by the host and may be 128, 256, or 512 bytes.
0	LAST	When this flag is set to '1' it indicates to the DMA engine that this is the last descriptor in the chain. It should be cleared to '0' on any descriptors leading up to the last one. This flag must be set on the last entry in each chain in order for the DMA engine to operate properly.

DMA DESCRIPTOR PORT NOTES:

- The DMA descriptor is 64-bits wide unlike the other registers which are 32-bits wide. This is simply to group the related fields together into a structure.
- PIO reads/writes can only read 32-bits of this port at a time.
- Even DWORD addresses read/write the upper 32-bits while odd DWORD addresses read/write the lower 32-bits.
- There are 128 64-bit descriptors for each chain with 256 descriptors total. Chain 0 uses the first 128 and chain 1 uses the second 128.
- Not all descriptors in a chain have to be used by the software, but the last valid descriptor in a chain MUST have the LAST bit set.
- The initial contents of the descriptors are undefined.
- The content of the descriptors is persistent and may be re-used on multiple transfers if desired for improved efficiency.
- The descriptors of a given chain should not be written to while a DMA operation on that chain is currently in progress.

8.17 SYNCCTRL – Clock Synchronizer Control/Status Register

Address: 0x5000

	31	30	29	28	27	26	25	24
Field	PLL_LOCK	STA_VCXO	STA_REF	Rsvd				
Access	RO	RO	RO	RO				
Reset	0	0	0	X				

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							REF_SEL
Access	RO							R/W
Reset	X							1

	7	6	5	4	3	2	1	0
Field	Rsvd						SPI_DONE	SPI_START
Access	RO						RO	R/W1SC
Reset	X						1	1

Bit(s)	Field	Description
31	PLL_LOCK	This bit reflects the current state of the Clock Synchronizer chip's PLL_LOCK line. When this bit is '1' the PLL is locked, otherwise it is not. This bit is monitored for changes and changes are reported in the BCSR register which can cause a CPU interrupt when enabled.
30	STA_VCXO	This bit reflects the current state of the Clock Synchronizer chip's STA_VCXO line. This line can mean different things depending on the SPI Data configuration. Please refer to the chip's datasheet. This bit is monitored for changes and changes are reported in the BCSR register which can cause a CPU interrupt when enabled.
29	STA_REF	This bit reflects the current state of the Clock Synchronizer chip's STA_REF line. This line can mean different things depending on the SPI Data configuration. Please refer to the chip's datasheet. This bit is monitored for changes and changes are reported in the BCSR register which can cause a CPU interrupt when enabled.
8	REF_SEL	Selects the primary reference of the Clock Synchronizer chip (10MHz on-board oscillator) when set to '1' else the secondary reference (front panel input).
1	SPI_DONE	The SPI Master controller is done (idle) when this bit is '1', else it is currently busy shifting data out of the SPI Data registers into the Clock Synchronizer chip.
0	SPI_START	Writing a '1' to this bit starts the SPI Master controller which will drive the values of the SYNCSPI0-3 registers out to the Clock Synchronizer chip. The values in these registers do not control the Clock Synchronizer chip until this bit is set to '1 and then the SPI_DONE bit transitions to '0' and then back to '1' indicating that the SPI transfer is complete. This bit defaults to '1' so that the FPGA's power-on default values specific to the AMC511 will be automatically transferred to the chip to override its more generic defaults.

8.18 SYNCSPI0 – Clock Synchronizer SPI Data 0 Register

Address: 0x5004

	31	30	29	28	27	26	25	24
Field	REFDEC	MANAUT	DLYN			DLYM		
Access	R/W	R/W	R/W			R/W		
Reset	0	0	0			0		

	23	22	21	20	19	18	17	16
Field	N[11:4]							
Access	R/W							
Reset	(see below)							

	15	14	13	12	11	10	9	8
Field	N[3:0]				M[9:6]			
Access	R/W				R/W			
Reset	989 *				(see below)			

	7	6	5	4	3	2	1	0
Field	M[5:0]						C	
Access	R/W						RO	
Reset	54 *						0x0	

Please refer to the CDCM7005 datasheet for the meaning of these fields. Some of the power-on defaults have been changed in the FPGA design compared to the chip defaults in order to better suit the AMC511 design. Also, for purposes of preventing mis-operation within the AMC511 design, some fields may be designated read-only in the FPGA design that were originally read/write in the chip design.

*** NOTE:** The N/M divider register defaults may be configured using conditional compile options in the FPGA reference design code to adapt them to different available VCXO / Ref Clock combinations. The default configuration is for a 180 MHz VCXO and 10 MHz Ref Clock. Refer to the section entitled **Customizing the VCXO / Ref Clock Configuration**. Also note that the M and N values in this register are specified as the desired value minus one. See the CDCM7005 datasheet.

8.19 SYNCSPI1 – Clock Synchronizer SPI Data 1 Register

Address: 0x5008

	31	30	29	28	27	26	25	24
Field	90DIV8	90DIV4	ADLOCK	SXIOREF	SREF	OUT4B		OUT4A[1]
Access	R/W	R/W	RO	R/W	R/W	RO		RO
Reset	0	0	0	0	0	0x0		(see below)

	23	22	21	20	19	18	17	16
Field	OUT4A[0]	OUT3B		OUT3A		OUT2B		OUT2A[1]
Access	RO	RO		RO		RO		RO
Reset	0x0	0x0		0x0		0x0		(see below)

	15	14	13	12	11	10	9	8
Field	OUT2A[0]	OUT1B		OUT1A		OUT0B		OUT0A[1]
Access	RO	RO		RO		RO		RO
Reset	0x0	0x0		0x0		0x0		(see below)

	7	6	5	4	3	2	1	0
Field	OUT0A[0]	OUTSEL4	OUTSEL3	OUTSEL2	OUTSEL1	OUTSEL0	C	
Access	RO	RO	RO	RO	RO	RO	RO	
Reset	0x0	1	1	1	1	1	0x1	

Please refer to the CDCM7005 datasheet for the meaning of these fields. Some of the power-on defaults have been changed in the FPGA design compared to the chip defaults in order to better suit the AMC511 design. Also, for purposes of preventing mis-operation within the AMC511 design, some fields may be designated read-only in the FPGA design that were originally read/write in the chip design.

8.20 SYNCSPI2 – Clock Synchronizer SPI Data 2 Register

Address: 0x500C

	31	30	29	28	27	26	25	24
Field	NHOLD	NRESET	RESHOL	NPD	Y4MUX			Y3MUX[2]
Access	R/W	R/W	RO	R/W	R/W			R/W
Reset	1	1	0	1	0x0			(see below)

	23	22	21	20	19	18	17	16
Field	Y3MUX[1:0]		Y2MUX			Y1MUX		
Access	R/W		R/W			R/W		
Reset	0x0		0x0			0x0		

	15	14	13	12	11	10	9	8
Field	Y0MUX			FBMUX			PFD	
Access	R/W			R/W			R/W	
Reset	0x0			0x0			0x0	

	7	6	5	4	3	2	1	0
Field	CP				PRECP	CP_DIR	C	
Access	R/W				R/W	RO	RO	
Reset	0xA				0	0	0x2	

Please refer to the CDCM7005 datasheet for the meaning of these fields. Some of the power-on defaults have been changed in the FPGA design compared to the chip defaults in order to better suit the AMC511 design. Also, for purposes of preventing mis-operation within the AMC511 design, some fields may be designated read-only in the FPGA design that were originally read/write in the chip design.

8.21 SYNCSPI3 – Clock Synchronizer SPI Data 3 Register

Address: 0x5010

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	0							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	0							

	15	14	13	12	11	10	9	8
Field	Rsvd				HOLDTR	Rsvd	HOLDF	Rsvd
Access	RO				R/W	RO	R/W	RO
Reset	0				0	0	0	0

	7	6	5	4	3	2	1	0
Field	Rsvd	CSLIP	LOCKC		LOCKW		C	
Access	RO	R/W	R/W		R/W		RO	
Reset	0	0	0x2		0x1		0x3	

Please refer to the CDCM7005 datasheet for the meaning of these fields. Some of the power-on defaults have been changed in the FPGA design compared to the chip defaults in order to better suit the AMC511 design. Also, for purposes of preventing mis-operation within the AMC511 design, some fields may be designated read-only in the FPGA design that were originally read/write in the chip design.

8.22 CRSYNC, CRTCLKA/B/C/D, CRTRIGINx, CRPRIREF, CRSECREF – Clock Routing Registers

Addresses: 0x6000 (CRSYNC), 0x6004 (CRTCLKA), 0x6008 (CRTCLKB), 0x600C (CRTCLKC), 0x6010 (CRTCLKD), 0x6014 (CRTRIGIN0), 0x6018 (CRTRIGIN1), 0x601C (CRTRIGIN2), 0x6020 (CRTRIGIN3), 0x6024 (CRPRIREF), 0x6028 (CRSECREF)

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd			SOURCE				
Access	RO			R/W				
Reset	X			(see description on next page)				

Bit(s)	Field	Description
4-0	SOURCE	This field selects the source signal to drive to the destination as follows: 0x00 – Fixed logic '0' value 0x01 – Fixed logic '1' value 0x02 – Front panel TRIG IN 0x03 – Front panel REF CLK IN 0x04 – Backplane TCLKA input 0x05 – Backplane TCLKB input 0x06 – Backplane TCLKC input 0x07 – Backplane TCLKD input 0x08 – A/D channel 0's trigger out (RUNNING bit) 0x09 – A/D channel 1's trigger out (RUNNING bit) 0x0A – A/D channel 2's trigger out (RUNNING bit) 0x0B – A/D channel 3's trigger out (RUNNING bit) 0x0C – STA_REF from Clock Synchronizer chip 0x0D – STA_VCXO from Clock Synchronizer chip 0x0E – PLL_LOCK from Clock Synchronizer chip 0x0F – A/D sample clock as driven by Clock Synchronizer chip (REFCLK1) 0x10 – A/D sample clock as driven by A/D chip for channel 0 (CHCLK0) 0x11 – A/D sample clock as driven by A/D chip for channel 1 (CHCLK1) 0x12 – A/D sample clock as driven by A/D chip for channel 2 (CHCLK2) 0x13 – A/D sample clock as driven by A/D chip for channel 3 (CHCLK3) 0x14 – 10MHZ on-board clock 0x15 – PCIE_CLK1 on-board clock 0x16 – CLK125MHZ2 on-board clock 0x17 – CLK156_25MHZ0 on-board clock 0x18 – A/D channel 0 sign bit (simple comparator function) 0x19 – A/D channel 1 sign bit (simple comparator function) 0x1A – A/D channel 2 sign bit (simple comparator function) 0x1B – A/D channel 3 sign bit (simple comparator function) 0x1C – Master interrupt signal

The Clock Routing Registers all follow the same layout. The registers correspond to MUX selectors in the FPGA and select the source signal for a given signal path. The target paths are as follows:

Register	Target signal path	Default	Meaning
CRSYNC (0x6000)	Front panel SYNC output	0x00	Off
CRTCLKA (0x6004)	Backplane TCLKA output	0x00	Off
CRTCLKB (0x6008)	Backplane TCLKB output	0x00	Off
CRTCLKC (0x600C)	Backplane TCLKC output	0x00	Off
CRTCLKD (0x6010)	Backplane TCLKD output	0x00	Off
CRTRIGIN0 (0x6014)	Trigger input for A/D channel 0	0x01	On
CRTRIGIN1 (0x6018)	Trigger input for A/D channel 1	0x01	On
CRTRIGIN2 (0x601C)	Trigger input for A/D channel 2	0x01	On
CRTRIGIN3 (0x6020)	Trigger input for A/D channel 3	0x01	On
CRPRIREF (0x6024)	Clock Synchronizer Primary Reference	0x03	Front REF CLK IN
CRSECREF (0x6028)	Clock Synchronizer Secondary Reference	0x14	On-board 10MHz

The CRTRIGINx registers default to 1 because the trigger logic in the ADC channels is always active which means that the routed trigger has to be forced to a logic '1' in order for an automatic trigger to occur when the channel is enabled. If something other than an automatic trigger is desired, then the value can be changed as desired.

8.23 CREN – Clock Routing Enable Register

Address: 0x602C

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	TCLKD_OUT	TCLKC_OUT	TCLKB_OUT	TCLKA_OUT	TCLKX_IN	REF_CLK_IN	TRIG_IN	SYNC_OUT
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	0	0

Bit(s)	Field	Description
0	SYNC_OUT	Enable the front panel SYNC OUT output when '1' else disable it.
1	TRIG_IN	Enable the front panel TRIG IN input when '1' else disable it.
2	REF_CLK_IN	Enable the front panel REF CLK IN input when '1' else disable it.
3	TCLKX_IN	Enable the backplane M-LVDS receivers when '1' else disable them.
4	TCLKA_OUT	Enable the TCLKA M-LVDS transmitter when '1' else disable it.
5	TCLKB_OUT	Enable the TCLKB M-LVDS transmitter when '1' else disable it.
6	TCLKC_OUT	Enable the TCLKC M-LVDS transmitter when '1' else disable it.
7	TCLKD_OUT	Enable the TCLKD M-LVDS transmitter when '1' else disable it.

8.24 GREENLEDx – Green Status LED Control Registers

Address: 0x7000, 0x7004, 0x7008, 0x700C

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd					LEDSEL		
Access	RO					R/W		
Reset	X					0x4		

Bit(s)	Field	Description
2-0	LEDSEL	Selects the source to drive to the User LED 0 output. The LED will be ON when the signal is '1' and off when the signal is '0'. The LED signal can be sourced from the following list: 0x0 – Off 0x1 – On 0x2 – Blink 0x3 – Corresponding A/D channel's trigger out (RUNNING bit) 0x4 – Power-on indication GREENLED0: 1000Base-X 0 SYNC GREENLED1: 1000Base-X 1 SYNC GREENLED2: PCIe link up GREENLED3: SRAM Passing 0x5 – STA_REF from Clock Synchronizer chip 0x6 – STA_VCXO from Clock Synchronizer chip 0x7 – PLL_LOCK from Clock Synchronizer chip

8.25 AMBERLEDx – Amber Status LED Control Registers

Address: 0x7010, 0x7014, 0x7018, 0x701C

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd					LEDSEL		
Access	RO					R/W		
Reset	X					(see description)		

Bit(s)	Field	Description
2-0	LEDSEL	Selects the source to drive to the User LED 1 output. The LED will be ON when the signal is '1' and off when the signal is '0'. The LED signal can be sourced from the following list: 0x00 – Off 0x01 – On 0x02 – Blink *(AMBERLED0's default register value)* 0x03 – Corresponding A/D channel's fault indication 0x04 – Power-on indication *(AMBERLED1/2/3's default register value)* AMBERLED0: Off *(unused)* AMBERLED1: PCIe x8 width AMBERLED2: PCIe x4 width AMBERLED3: SRAM Failing 0x05 – STA_REF from Clock Synchronizer chip 0x06 – STA_VCXO from Clock Synchronizer chip 0x07 – PLL_LOCK from Clock Synchronizer chip

8.26 ARBSTAT – DMA Arbiter Status Register

Address: 0x8000

	31	30	29	28	27	26	25	24
Field	STATE	REQM	DONEM	ACKM	REQ3	REQ2	REQ1	REQ0
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	X	X	X	X	X	X	X	X

	23	22	21	20	19	18	17	16
Field	DONE3	DONE2	DONE1	DONE0	ACK3	ACK2	ACK1	ACK0
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	X	X	X	X	X	X	X	X

	15	14	13	12	11	10	9	8
Field	Rsvd						NEXT	
Access	RO						RO	
Reset	X						X	

	7	6	5	4	3	2	1	0
Field	Rsvd		PREV		Rsvd		CURR	
Access	RO		RO		RO		RO	
Reset	X		0x0		X		0x0	

Bit(s)	Field	Description
31	STATE	This field shows the current state of the DMA Arbiter as follows: '0': Waiting for REQ from currently selected ADC core '1': Waiting for ACK from DMA Burst Engine
30	REQM	This field shows the current state of the master request line to the DMA Burst Engine.
29	DONEM	This field shows the current state of the master DONE line from the DMA Burst Engine.
28	ACKM	This field shows the current state of the master ACK line from the DMA Burst Engine.
27-24	REQx	These fields show the current state of the REQ lines from the ADC cores.
23-20	DONEx	These fields show the current state of the DONE lines to the ADC cores.
19-16	ACKx	These fields show the current state of the ACK lines to the ADC cores.
9-8	NEXT	This field shows which ADC core would be next to DMA if the master ACK line would have been pulsed at the instant of the read.
5-4	PREV	This field shows which ADC core will be the recipient of the next DONE pulse from the DMA Burst Engine.
1-0	CURR	This field shows which ADC core is currently allowed to post a request to the DMA Burst Engine.

NOTE: All of these fields are for diagnostic use only (i.e. to help debug the hardware in the event of a DMA stall, etc).

8.27 BPICTRL – BPI Flash Control Register

Address: 0x9000

	31	30	29	28	27	26	25	24
Field	BUSY	Rsvd						
Access	RO	RO						
Reset	0	X						

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	Rsvd					RECONFIG	WRITE	EXEC
Access	RO					R/W1SC	R/W	R/W1SC
Reset	X					0	0	0

Bit(s)	Field	Description
31	BUSY	This bit indicates when the BPI Flash controller is busy performing a read or a write transaction on the flash bus. The host CPU should poll this bit to know when to proceed with the next operation.
2	RECONFIG	When this bit is set to '1' the FPGA will initiate a reconfiguration of itself by resetting and loading the contents of the BPI Flash as if a power-cycle occurred. **NOTE:** The PCIe bus link will go down for a short time as a result of this action.
1	WRITE	When this bit is set to '1' it indicates that a write operation should be performed when the EXEC bit is set to '1'. Otherwise a '0' in this field indicates that a write operation should be performed.
0	EXEC	When this bet is set to '1' the flash controller performs either a read or a write operation to the BPI Flash as determined by the current state of the WRITE bit. The BPIADDR and BPIDATA registers are used/updated accordingly and the BUSY bit indicates the status of the operation.

8.28 BPIADDR – BPI Flash Address Register

Address: 0x9004

	31	30	29	28	27	26	25	24
Field	Rsvd							ADDR[24]
Access	RO							R/W
Reset	X							[see below]

	23	22	21	20	19	18	17	16
Field	ADDR[23:16]							
Access	R/W							
Reset	[see below]							

	15	14	13	12	11	10	9	8
Field	ADDR[15:8]							
Access	R/W							
Reset	[see below]							

	7	6	5	4	3	2	1	0
Field	ADDR[7:0]							
Access	R/W							
Reset	0x0000000							

Bit(s)	Field	Description
24-0	ADDR	This field contains the BPI Flash address to read from/write to.

8.29 BPIDATA – BPI Flash Data Register

Address: 0x9008

	31	30	29	28	27	26	25	24
Field	READ_DATA[15:8]							
Access	RO							
Reset	[see below]							

	23	22	21	20	19	18	17	16
Field	READ_DATA[7:0]							
Access	RO							
Reset	0x0000							

	15	14	13	12	11	10	9	8
Field	WRITE_DATA[15:8]							
Access	R/W							
Reset	[see below]							

	7	6	5	4	3	2	1	0
Field	WRITE_DATA[7:0]							
Access	R/W							
Reset	0x0000							

Bit(s)	Field	Description
31-16	READ_DATA	This field contains the data read from the BPI Flash at the end of a read cycle.
15-0	WRITE_DATA	This field should be filled in with the data to write to the BPI Flash prior to a write cycle.

8.30 SCRATCH – Scratch Register

Address: 0xFFF4

	31	30	29	28	27	26	25	24
Field	SCRATCH[31:24]							
Access	R/W							
Reset	(see below)							

	23	22	21	20	19	18	17	16
Field	SCRATCH[23:16]							
Access	R/W							
Reset	(see below)							

	15	14	13	12	11	10	9	8
Field	SCRATCH[15:8]							
Access	R/W							
Reset	(see below)							

	7	6	5	4	3	2	1	0
Field	SCRATCH[7:0]							
Access	R/W							
Reset	0x00000000							

Bit(s)	Field	Description
31-0	SCRATCH	Scratchpad area for PCIe bus testing or free for other software use.

8.31 VER – Version Register

Address: 0xFFF8

	31	30	29	28	27	26	25	24
Field	MAJOR							
Access	RO							
Reset	(varies based on FPGA release)							

	23	22	21	20	19	18	17	16
Field	MINOR							
Access	RO							
Reset	(varies based on FPGA release)							

	15	14	13	12	11	10	9	8
Field	PATCH							
Access	RO							
Reset	(varies based on FPGA release)							

	7	6	5	4	3	2	1	0
Field	REV							
Access	RO							
Reset	(varies based on FPGA release)							

Bit(s)	Field	Description
31-24	MAJOR	Major release number of the FPGA image.
23-16	MINOR	Minor release number of the FPGA image.
15-8	PATCH	Patch release number of the FPGA image.
7-0	REV	Revision release number of the FPGA image.

8.32 SIG – Signature Register

Address: 0xFFFC

	31	30	29	28	27	26	25	24
Field	Rsvd							
Access	RO							
Reset	X							

	23	22	21	20	19	18	17	16
Field	Rsvd							
Access	RO							
Reset	X							

	15	14	13	12	11	10	9	8
Field	Rsvd							
Access	RO							
Reset	X							

	7	6	5	4	3	2	1	0
Field	SIG							
Access	RO							
Reset	0xAD							

Bit(s)	Field	Description
7-0	SIG	Unchanging value which helps to verify that the proper FPGA image is programmed into the part.

9 Appendix B: Device Driver IOCTL Specification

The interfaces to the driver are exported in the amc511.h header file which should be included in any user application that uses the driver.

9.1 AMC511_IOC_GET_INFO

```
typedef struct {
        unsigned char major;
        unsigned char minor;
        unsigned char patch;
        unsigned char rev;
} amc511_version_t;

#define AMC511_CHAN_NAME_LEN    24      /* amc511-BB.DD.F-C\0 */

typedef struct {
        char                    chan_name[AMC511_CHAN_NAME_LEN];
        unsigned int            mmap_size;
        amc511_version_t        driver_version;
        amc511_version_t        fpga_version;
        unsigned                fpga_configured : 1;
} amc511_info_t;
```

Usage:

```
int fd;
amc511_info_t info;
ioctl( fd, AMC511_IOC_GET_INFO, &info );
```

This call returns information about a channel and the driver/FPGA. It also informs the application about the size of the memory map region to use for the **mmap**() call.

9.2 AMC511_IOC_GET/SET_LEDS

```
#define AMC511_LEDSEL_OFF               0x0
#define AMC511_LEDSEL_ON                0x1
#define AMC511_LEDSEL_BLINK             0x2
#define AMC511_LEDSEL_A2D_STATUS        0x3
#define AMC511_LEDSEL_POWERON_STATUS    0x4
#define AMC511_LEDSEL_STA_REF           0x5
#define AMC511_LEDSEL_STA_VCXO          0x6
#define AMC511_LEDSEL_PLL_LOCK          0x7

typedef struct {
        unsigned        greenled0 : 3;
        unsigned        greenled1 : 3;
        unsigned        greenled2 : 3;
```

```
        unsigned        greenled3 : 3;
        unsigned        amberled0 : 3;
        unsigned        amberled1 : 3;
        unsigned        amberled2 : 3;
        unsigned        amberled3 : 3;
} amc511_ledsel_t;
```

Usage:

```
      int fd;
      amc511_ledsel_t ledsel;
      ioctl( fd, AMC511_IOC_GET_LEDS, &ledsel );
      ioctl( fd, AMC511_IOC_SET_LEDS, &ledsel );
```

These calls get or set the LED Controller registers. Refer to the register specification for additional details on A/D controller statuses and power-on statuses.

9.3 AMC511_IOC_GET/SET_CLOCK_ROUTING

```
#define AMC511_SOURCE_ZERO              0x00
#define AMC511_SOURCE_ONE               0x01
#define AMC511_SOURCE_TRIG_IN           0x02
#define AMC511_SOURCE_REF_CLK_IN        0x03
#define AMC511_SOURCE_TCLKA_IN          0x04
#define AMC511_SOURCE_TCLKB_IN          0x05
#define AMC511_SOURCE_TCLKC_IN          0x06
#define AMC511_SOURCE_TCLKD_IN          0x07
#define AMC511_SOURCE_ADC0_TRIGOUT      0x08
#define AMC511_SOURCE_ADC1_TRIGOUT      0x09
#define AMC511_SOURCE_ADC2_TRIGOUT      0x0A
#define AMC511_SOURCE_ADC3_TRIGOUT      0x0B
#define AMC511_SOURCE_STA_REF           0x0C
#define AMC511_SOURCE_STA_VCXO          0x0D
#define AMC511_SOURCE_PLL_LOCK          0x0E
#define AMC511_SOURCE_REFCLK1           0x0F
#define AMC511_SOURCE_SMPCLK0           0x10
#define AMC511_SOURCE_SMPCLK1           0x11
#define AMC511_SOURCE_SMPCLK2           0x12
#define AMC511_SOURCE_SMPCLK3           0x13
#define AMC511_SOURCE_10MHZ             0x14
#define AMC511_SOURCE_PCIE_CLK1         0x15
#define AMC511_SOURCE_CLK125MHZ2        0x16
#define AMC511_SOURCE_CLK156_25MHZ0     0x17
#define AMC511_SOURCE_ADC0_COMP         0x18
#define AMC511_SOURCE_ADC1_COMP         0x19
#define AMC511_SOURCE_ADC2_COMP         0x1A
#define AMC511_SOURCE_ADC3_COMP         0x1B
#define AMC511_SOURCE_INTERRUPT         0x1C

#define AMC511_PATH_ENABLED             1
#define AMC511_PATH_DISABLED            0

typedef struct {
        /* clock routes */
        unsigned        sync_out        : 5;
```

```
        unsigned        tclka_out       : 5;
        unsigned        tclkb_out       : 5;
        unsigned        tclkc_out       : 5;
        unsigned        tclkd_out       : 5;
        unsigned        adc0_trigin     : 5;
        unsigned        adc1_trigin     : 5;
        unsigned        adc2_trigin     : 5;
        unsigned        adc3_trigin     : 5;
        unsigned        pri_ref         : 5;
        unsigned        sec_ref         : 5;

        /* path enables */
        unsigned        sync_out_en     : 1;
        unsigned        trig_in_en      : 1;
        unsigned        ref_clk_in_en   : 1;
        unsigned        tclkx_in_en     : 1;
        unsigned        tclka_out_en    : 1;
        unsigned        tclkb_out_en    : 1;
        unsigned        tclkc_out_en    : 1;
        unsigned        tclkd_out_en    : 1;
} amc511_clock_routing_t;
```

Usage:

```
int fd;
amc511_clock_routing_t routing;
ioctl( fd, AMC511_IOC_GET_CLOCK_ROUTING, &routing );
ioctl( fd, AMC511_IOC_SET_CLOCK_ROUTING, &routing );
```

These calls get/set the Clock Router registers.

9.4 AMC511_IOC_GET/SET_CLOCK_SYNC

```
#define AMC511_CS_REF_SEL_SECONDARY     0
#define AMC511_CS_REF_SEL_PRIMARY       1

typedef struct {
        /* Control */
        unsigned        ref_sel         : 1;

        /* SPI Word 0 */
        unsigned        refdec          : 1;
        unsigned        manaut          : 1;
        unsigned        dlyn            : 3;
        unsigned        dlym            : 3;
        unsigned        n               : 12;
        unsigned        m               : 10;

        /* SPI Word 1 */
        unsigned        _90div8         : 1;
        unsigned        _90div4         : 1;
        unsigned        adlock          : 1;    // Read-Only
        unsigned        sxioref         : 1;
        unsigned        sref            : 1;
        unsigned        out4b           : 2;    // Read-Only
```

```
        unsigned        out4a           : 2;    // Read-Only
        unsigned        out3b           : 2;    // Read-Only
        unsigned        out3a           : 2;    // Read-Only
        unsigned        out2b           : 2;    // Read-Only
        unsigned        out2a           : 2;    // Read-Only
        unsigned        out1b           : 2;    // Read-Only
        unsigned        out1a           : 2;    // Read-Only
        unsigned        out0b           : 2;    // Read-Only
        unsigned        out0a           : 2;    // Read-Only
        unsigned        outsel4         : 1;    // Read-Only
        unsigned        outsel3         : 1;    // Read-Only
        unsigned        outsel2         : 1;    // Read-Only
        unsigned        outsel1         : 1;    // Read-Only
        unsigned        outsel0         : 1;    // Read-Only

        /* SPI Word 2 */
        unsigned        nhold           : 1;
        unsigned        nreset          : 1;
        unsigned        reshol          : 1;    // Read-Only
        unsigned        npd             : 1;
        unsigned        y4mux           : 3;
        unsigned        y3mux           : 3;
        unsigned        y2mux           : 3;
        unsigned        y1mux           : 3;
        unsigned        y0mux           : 3;
        unsigned        fbmux           : 3;
        unsigned        pfd             : 2;
        unsigned        cp              : 4;
        unsigned        precp           : 1;
        unsigned        cp_dir          : 1;    // Read-Only

        /* SPI Word 3 */
        unsigned        holdtr          : 1;
        unsigned        holdf           : 1;
        unsigned        cslip           : 1;
        unsigned        lockc           : 2;
        unsigned        lockw           : 2;
} amc511_clock_sync_t;
```

Usage:

```
int fd;
amc511_clock_sync_t clksync;
ioctl( fd, AMC511_IOC_GET_CLOCK_SYNC, &clksync );
ioctl( fd, AMC511_IOC_SET_CLOCK_SYNC, &clksync );
```

These calls get/set the Clock Synchronizer registers. The driver will automatically trigger the SPI Master to transfer the new settings for the set command and wait for completion so that the application can be assured that the new settings have taken effect in the TI Clock Synchronizer chip prior to the calls return. Refer to the register specifications for further details.

The status of the clock synchronizer chip is available via the channel status ioctl to simplify status polling.

9.5 AMC511_IOC_GET/SET_CHAN_CTRL

```
typedef struct {
        unsigned        decm            : 8;
        unsigned        pat             : 3;
        unsigned        src             : 2;
        unsigned        pga             : 1;
        unsigned        rand            : 1;
        unsigned        dith            : 1;
        unsigned        enable          : 1;
} amc511_channel_control_t;
```

Usage:

```
int fd;
amc511_channel_control_t chanctrl;
ioctl( fd, AMC511_IOC_GET_CHAN_CTRL, &chanctrl );
ioctl( fd, AMC511_IOC_SET_CHAN_CTRL, &chanctrl );
```

These calls get/set the channel control register in the ADC channel. The set call also controls the device driver's data acquisition behavior. Refer to the register specifications for further details.

9.6 AMC511_IOC_GET_CHAN_STATUS

```
typedef struct {
        // Current channel status flags
        unsigned        running         : 1;

        // Current clock synchronizer status flags
        unsigned        pll_lock        : 1;
        unsigned        sta_vcxo        : 1;
        unsigned        sta_ref         : 1;
} amc511_channel_status_t;
```

Usage:

```
int fd;
amc511_channel_status_t chanstat;
ioctl( fd, AMC511_IOC_GET_CHAN_STATUS, &chanstat );
```

This call gets the current channel status. The application can use the poll(POLLPRI) call to detect changes to these status fields.

9.7 AMC511_IOC_GET_CHAN_DATA

```
typedef struct {
        unsigned        fir_overflow    : 1;
        unsigned        adc_overflow    : 1;
        unsigned        fifo_overflow   : 1;
        unsigned        fifo_underflow  : 1;

        // Data
        unsigned int    length;         // Length in bytes
        unsigned long   offset;         // Offset within mmapped region
} amc511_channel_data_t;
```

Usage:

```
int fd;
amc511_channel_data_t data;
ioctl( fd, AMC511_IOC_GET_CHAN_DATA, &data );
```

This call provides information about the streaming status of the A/D pipeline as well as information about where to find the new data within the memory mapped buffer region. The status flags in the card are reset at the start of each transfer and any overflow/underflow that occurs during the transfer will be captured along with the data buffers and forwarded to the application.

The application should wait for new data to become available by using the **poll(POLLIN)** call.

The application must not access the buffer space without first gaining access via this call. The application should put the buffer back to the driver as soon as possible to prevent stalling the data streaming.

9.8 AMC511_IOC_PUT_CHAN_DATA

Usage:

```
int fd;
ioctl( fd, AMC511_IOC_PUT_CHAN_DATA, 0 );
```

This call puts the last data buffer back to the driver so that it can be reused for the next streaming transfer. This should be done as soon as possible to prevent stalling the data streaming. However, once the buffer is put back to the driver the application must not access it again.

9.9 AMC511_IOC_FLASH_OP

```
typedef enum {
      AMC511_FLASH_READ = 0,
      AMC511_FLASH_WRITE,
      AMC511_FLASH_RECONFIG
} amc511_flash_action_t;

typedef struct {
        unsigned int          address;
        unsigned short        data;
        amc511_flash_action_t action;
} amc511_flash_op_t;
```

Usage:

```
      int fd;
      amc511_flash_op_t op;
      ioctl( fd, AMC511_IOC_FLASH_OP, &op );
```

This call performs a read/write or reconfiguration using the BPI Flash attached to the FPGA.