FMCADC100M14b4cha PCIe HDL specification

J.Serrano

November 2, 2010

Document History

DATE	CHANGES
26 April 2010	First release submitted for comments.
27 April 2010	Added ADCSTATR for state machine monitoring. Added TRIGPOSR for circular buffer support. Added endianness support in BANKSELR.
28 April 2010	Added power good, shutdown and current and voltage aqn for supplies. Added Vadj control. Added DMA. Added IQR sources list in IRQ controller subsection.
3 May 2010	Added TOC and Document History. More detailed figure 1. General conventions for unused bits in registers.
4 May 2010	Added space holders for VME case in SUPCTRLR. Added reset column to register map tables. Added two new interrupts: FMC input over-load and FMC over-heating.
9 May 2010	Re-wrote DMA chapter.
10 May 2010	Added address jump configuration in DMA engine. Added TRIGPOSR description.
11 May 2010	Added DMA Error interrupt.
25 July 2010	Added UTC and address converter in block diagram. Added Shot Counts in state machine. DMA stuff passed to GN4124 core. Temperature readout only every 10 seconds. DDR controller completely re-written. Added section on UTC block. Added trigger delay and hold-off. Multi-shot mode added. UTC time tags for trigger, START and STOP commands.
6 September 2010	Added RELTAGR 256-byte area. Split CONTROLR in two registers: one for frequency and one for VADJ. Filled current and voltage measurement table.
7 September 2010	Re-ordered subsections. Filled in tables with addresses and reset states.
9 September 2010	Added internal reset command to SUPCTRLR register (bit 31).
10 September 2010	Si570 interrupt taken out of IRQSRCR. Renamed ADCxGAINR to ADCxSWITCHR. Eliminated hold-off and ADCCNTR registers. Added Future Improvements section.
23 September 2010	Various cosmetic changes. Suppressed UTCSETR. Changed switch register default settings.

Contents

1 Introduction

This document gives information needed by HDL and driver/library developers to support the FMCADC100M14b4cha FPGA Mezzanine Cards[1] in the PCIe FMC carrier[2] designed by BE-CO-HT at CERN. The support for this module must be as generic as possible in order to benefit from this effort for other ADC FMC cards and carriers. In particular, the sample width, number of channels and sampling rate should all be configurable parameters in the design. Another important aspect is to preserve insofar as possible the re-usability of developments between PCIe and VME64x uses, knowing that the VME64x carrier can host two mezzanines whereas the PCIe carrier has only one FMC site.

The PCIe carrier board has a Spartan 6 XC6SLX150T FPGA at its heart, surrounded by a host of peripherals for different applications. In particular, there is a fair amount of DDR3 RAM, a PLL chip, a DDS and Flash ROM. The FPGA is connected to a VITA 57 FPGA Mezzanine Card (FMC) slot, covering all pins of the Low Pin Count (LPC) connector. The purpose of this document is to specify how this carrier and the FMCADC100M14b4cha 4-channel 100 MS/s ADC mezzanine card can be used to build a complete ADC solution through appropriate configuration of the FPGA in the carrier.

The internal structure of the FPGA design can be seen in figure 1. It consists of a set of Wishbone cores, namely one Wishbone master and a set of slaves. Each slave deals with one or more external peripherals, with the exception of the interrupt controller. The PLL, DDS and SRAM chips are not used in this design. In the following sections, we go through all blocks, specifying their function and their internal registers.

[1] See http://www.ohwr.org/projects/fmc-adc-100m14b4cha.
[2] See http://www.ohwr.org/projects/fmc-pci-carrier.

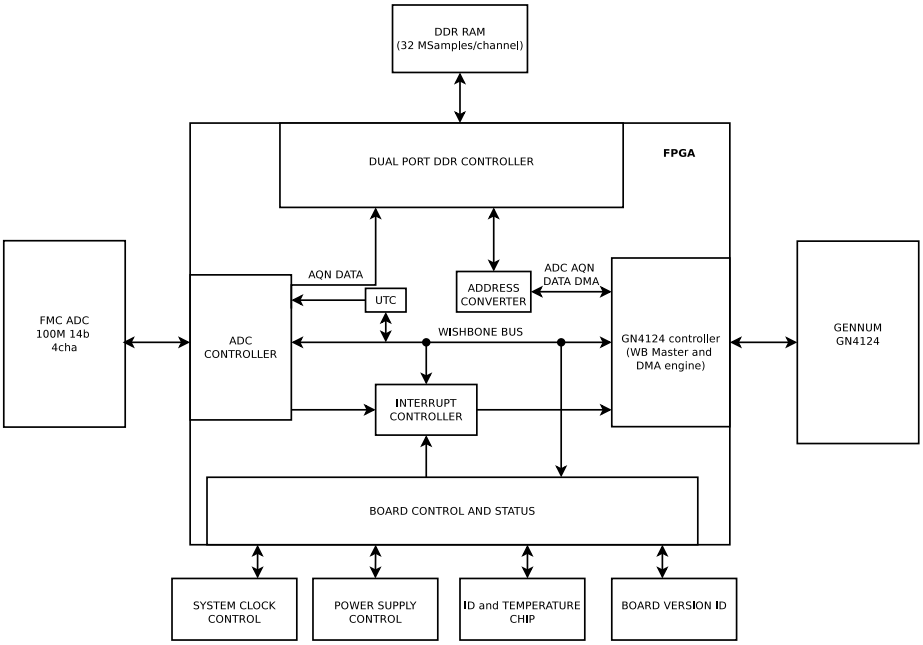

Figure 1: FPGA design internal structure.

2 FPGA blocks

For each internal block, we give a summary description of its function along with internal registers which can be read or written from the Wishbone master. Registers are presented in tables with their name, address offset (in 32-bit long words), access mode, value after reset and description. The address offset is the offset of a given register with respect to the beginning of the memory area pointed to by a given PCIe Base Address Register (BAR). This design only supports full 32-bit reads and writes for control and status registers. In general, unused bits should be ignored on read and written to with a '0'. For status registers, bits read as '0' represent a normal state of affairs, while those set to '1' signal some departure from nominal operation.

2.1 Board control and status

This block contains all control and status registers related to the carrier board independently of the application. Other applications can include it as is. Table 1 shows the list of registers in this block.

2.1.1 CARRTYPER

The CARRTYPER register uses bits [31..16] for a carrier type identifier and bits [7..0] for the PCB version. Bits [15..8] are reserved.

NAME	OFFSET	MODE	ON RESET	DESCRIPTION
CARRTYPER	0x0000	RO	0x00010001	Carrier Type and PCB version
SIIDLR	0x0001	RO	From SID chip	Carrier Silicon ID Low
SIIDHR	0x0002	RO	From SID chip	Carrier Silicon ID High
BSTREAMTR	0x0003	RO	0x00000001	Bit stream type
BSTREAMDR	0x0004	RO	From UTC	Bit stream date
CARRTEMPR	0x0005	RO	From DS18B20	Carrier temperature
STATUSR	0x0006	RO	From external signals	Carrier and power supply status
SUPCTRLR	0x0007	R/W	0x00000000	Power supply control
SUPAQNR	0x0008	RO	From external signals	Power supply voltage and current
FREQCTLR	0x0009	R/W	0x00000000	Carrier frequency control
VADJCTLR	0x000A	R/W	0x00000000	VADJ supply control
RELTAGR	0x0040 - 0x007F	RO	From versioning tool	Release tag

Table 1: Register set for the board control and status block.

2.1.2 SIIDLR and SIIDHR

The SIIDLR and SIIDHR registers contain respectively the low and high parts of the 64-bit Silicon ID read from the Maxim DS18B20 1-Wire digital thermometer after system reset (see 2.1.6).

2.1.3 BSTREAMTR and BSTREAMDR

BSTREAMTR uses an unsigned 32-bit number to define the bit stream type. BSTREAMDR contains the 32-bit unsigned UTC time when the bit stream was generated.

2.1.4 CARRTEMPR

CARRTEMPR contains the carrier temperature as read from the DS18B20 every ten seconds. The Board control and status block will set a bit to '1' for one clock tick after every reading if the temperature exceeds 60° Celsius. This bit will be connected to the interrupt controller so that the user can get a temperature interrupt if enabled.

2.1.5 STATUSR

STATUSR contains the carrier status, and in particular the status of power supplies and the detection of presence of a card in the FMC slot. Power supplies in the carrier are extensively monitored, and most of them provide a Power Good (G) signal. The internal structure of the STATUSR register is as follows:

- Bits [31..11] are unused.
- Bits [10..0] are (starting from bit 10): FMC_PRESENT, $\overline{3V3\_FMC\_G}$, $\overline{VADJ\_G}$, $\overline{1V8\_G}$, $\overline{1V5\_G}$, $\overline{CLEAN\_1V8\_G}$, $\overline{CLEAN\_3V3\_G}$, $\overline{5V\_G}$, $\overline{M2V\_G}$, $\overline{M5V2\_G}$ and $\overline{M12V\_G}$.

2.1.6 SUPCTRLR

The carrier has several power supplies that can be enabled or disabled individually from the FPGA. They are controlled from the SUPCTRLR register, which is a bit field made of individual Enable (E) bits. In addition, some bits in this register are used to select which power supply voltage and current are monitored through the SUPAQNR register.

- A write of '1' to bit 31 will generate a 1-tick-long internal system reset pulse for all registers in the FPGA.
- Bits [30..16] are unused.
- Bits [15..12] are unused.
- Bits [11..8] contain an unsigned 4-bit number which selects a power supply for voltage and current monitoring. Starting at 0: 12V_PCIe, 3V3_FMC, VADJ, 1V8.
- Bits [7..4] are unused.
- Bits [3..0] are (starting from bit 3): VADJ_E, M2V_E, M5V2_E, M12V_E.

2.1.7 SUPAQNR

The SUPAQNR register holds a voltage and current consumption value resulting from on-board measurements using 16-bit ADCs. The selection of power supply to monitor is done in SUPCTRLR. Bits [31..16] hold an unsigned word representing voltage magnitude, while bits [15..0] are used for current consumption. To get Volts and Amperes from the raw measurements, different conversion factors must be used for each supply, as described in table 2.

SUPPLY	Voltage factor (V/bit)	Current factor (A/bit)
12V_PCIe	0.000297546	9.53674E-05
3V3_FMC	7.62939E-05	9.53674E-05
VADJ	7.62939E-05	9.53674E-05
1V8	3.8147E-05	9.53674E-05

Table 2: Conversion factors for voltage and current consumption measurements.

2.1.8 FREQCTLR

FREQCTLR will allow setting the system clock frequency. The 16 lower bits of the frequency control word will drive a 16-bit DAC connected to a VCXO with 25 MHz center frequency and a span of 10 ppm.

2.1.9 VADJCTLR

VADJCTLR will allow setting the Vadj supply voltage for the FMC slot. Vadj will change from 1V (0x00000000) to 3.5V (0x0000FFFF).

2.1.10 RELTAGR

This is a 256-byte ASCII area for text automatically generated by versioning tools for a tagged HDL release. Its role is to facilitate correlation of the contents of the FPGA with a tagged release in a repository. It can also contain a URL of the repository.

2.2 UTC core

The UTC block counts the 125 MHz system clock (sysclk) in order to generate 64-bit UTC time for time-stampling purposes. Writing to UTCHR (see table 3) clears UTCLR and starts counting at 125 MHz in UTCLR. When UTCLR reaches 125 million, it is cleared and the value of UTCHR is incremented by one.

NAME	OFFSET	MODE	RESET	DESCRIPTION
UTCLR	0x0080	RO	0	UTC sysclk ticks within second
UTCHR	0x0081	R/W	0	UTC seconds

Table 3: Register set for the UTC block.

2.3 Interrupt controller

The interrupt controller receives interrupt requests from different blocks, combines them and sends an interrupt request to the GN4124/Wishbone bridge. For each interrupt input, it sets a bit in the IRQSRCR register upon synchronous detection of a rising edge. These bits are cleared on read. Care must be taken at design time to avoid race conditions in which a rising edge does not result in setting a bit because of the overriding effect of a concurrent read. Detection of a rising edge in any of the bits can result in the generation of an interrupt if the associated bit in the IRQENR register is set. The interrupt sent to the GN4124/Wishbone bridge is a one-tick-long positive pulse.

NAME	OFFSET	MODE	ON RESET	DESCRIPTION
IRQSRCR	0x0082	RO, clear on read	0	Interrupt sources
IRQENR	0x0083	R/W	0	Interrupt enable mask

Table 4: Register set for the interrupt controller block.

2.3.1 IRQSRCR and IRQENR

The IRQSRCR and IRQENR are both bit fields with the same internal structure. The upper bits (except bit 31 of IRQSRCR, see below) are reserved for future use. Bits [8..0] are: DMA completion, DMA error, carrier over-heating, FMC over-heating, FMC input over-load, ADC trigger, ADC end of acquisition, FMC ID I2C R/W completion and FMC ADC configuration R/W completion. If an interrupt fires more than once before IRQSRCR is read, bit 31 of IRQSRCR will be set, for debugging purposes. This bit is also cleared on read.

2.4 GN4124 to Wishbone bridge

This block is a slave of the external GN4124 local bus and a master of the internal Wishbone bus. The GN4124 from Gennum is a PCIe to local bus bridge, capable of using 4 PCIe lanes for fast communication with the host and with DMA capability as well. In addition, the GN4124 can be used to reprogram the on-board FPGA.

A one-tick-long positive pulse from the interrupt controller will trigger generation of a message-based PCIe interrupt. More details about this core can be found in http://www.ohwr.org/projects/gn4124-core/wiki. We reproduce in table 5 the registers associated to this core.

2.5 Dual port DDR RAM controller

This block handles access to the MT41J128M16HA-15E DDR3 RAM chip from Micron. This chip has a data width of 16 bits and can hold 32 MSamples per channel in our application. The fact that only one sample can be written at a time means that the RAM must work at least four times faster than the ADCs. The DDR controller handles access to the DDR RAM from two dedicated Wishbone busses, one writing from the ADC controller (port 1) and one reading

NAME	OFFSET	MODE	RESET	DESCRIPTION
DMACTRLR	0x0100	R/W	0	DMA engine control
DMASTATR	0x0101	RO	0	DMA engine status
DMACSTARTR	0x0102	R/W	0	DMA start address in the carrier
DMAHSTARTLR	0x0103	R/W	0	DMA start address (low) in the host
DMAHSTARTHR	0x0104	R/W	0	DMA start address (high) in the host
DMALENR	0x0105	R/W	0	DMA read length in bytes
DMANEXTLR	0x0106	R/W	0	Pointer (low) to next item in list
DMANEXTHR	0x0107	R/W	0	Pointer (high) to next item in list
DMAATTRIBR	0x0108	R/W	0	DMA endianness and control

Table 5: Register set for the GN4124 core.

from the DMA engine (port 2). Port 1 accesses have priority over port 2 accesses if there is a clash.

It is important to avoid collisions between Wishbone read requests and ADC write requests. Taking into account that continuous read applications are not typical for such high-speed sampling systems, we will just allow reading while the state machine of the ADC core (see figure 2) is in the Idle state. The software controlling the card must ensure that memory readout only happens in this state. Access at any other time will result in a bus error.

Memory will be organized internally as a circular buffer, with all 4 channels being logged in an interleaved way, starting with channel 1 at offset 0. The ADC controller block will group samples by two (per channel) in a little-endian way so as to have a 32-bit data access path to the DDR controller. Even if the DDR is a 16-bit device it will be seen as a 32-bit one, thanks to the grouping of accesses to DDR in the DDR controller.

A memory layout with two channel-1 samples at offset 0, two channel-2 samples at offset 1, etc. is not convenient for the host. Therefore, an address converter block between the DDR controller and the GN4124 core will ensure that the hosts sees this memory as four blocks, each dedicated to a channel. This block simply needs to take the two higher-order bits from the host side and place them instead in the lower side of the address bus for the DDR controller.

After an acquisition, the host can read the address in the DDR RAM (in non-interleaved space) of the last acquired sample for channel 1 in the LASTPOSR register of the ADC Controller block (see table 7). All addresses are byte addresses, i.e. DDR RAM addresses get incremented by 4 for each two-sample word. A consequence of the grouping in two-samples is that only an even sumber of samples should be requested. Applications where the user wants an odd number of samples can be dealt with by the driver requesting one more sample and discarding it.

2.6 ADC controller

The ADC controller handles all communication with the ADC FMC. For this application, a 125 MHz system clock (sysclk) is required; it is anticipated that any fixed value of the FREQCTLR register (see 2.1.8) will do, and that an internal PLL multiplying this 25 MHz by 5 will generate a 125 MHz sysclk which will guarantee an absence of FIFO overflows from the 100 MS/s ADC chip. The ADC controller block has a Wishbone slave for configuration registers and a dedicated output connection to the DDR RAM controller for samples. It can also drive interrupt requests into the interrupt controller. These interrupts are all one-tick-long positive pulses. The sampling state machine is depicted in figure 2.

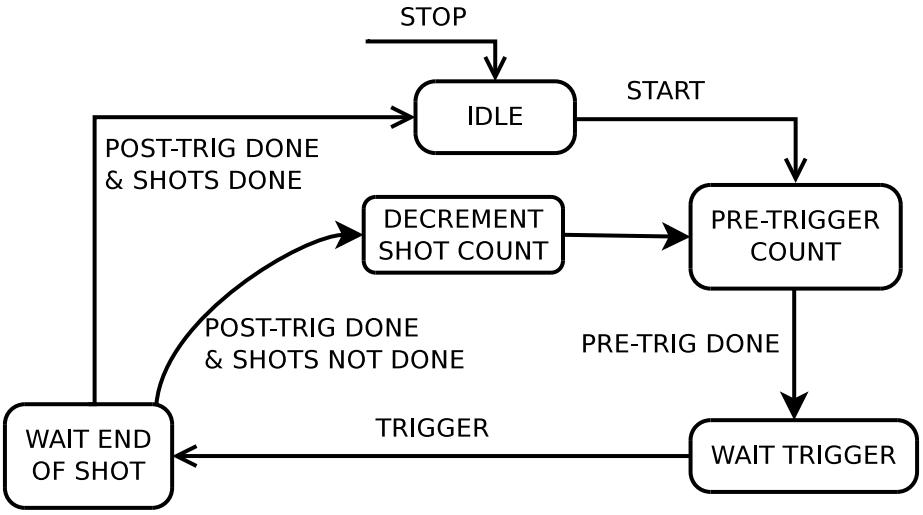

Figure 2: Sampling process state machine.

The state machine will drive two pulse-like signals into the IRQ controller. One will be a trigger interrupt and the other an end of acquisition interrupt, generated once the state machine reaches the Idle state after an acquisition. In addition, the ADC controller will produce interrupts in case of FMC overheating and input over-load (to protect the optional 50 Ohm termination). All of these interrupts can be enabled/disabled in the interrupt controller block. The ADC controller is also in charge of controlling two LEDs: power good and trigger (100 ms width).

2.6.1 ADCCTRLR

The START, STOP, and SOFT_TRIG commands can be used to provoke transitions in the state machine from the PCIe bus, by writing an appropriate value (1, 2 and 3 respectively) into the least significant byte of the ADCCTRLR register. A STOP command will always take the state machine to the Idle state, regardless of what its current state is.

NAME	OFFSET	MODE	RESET	DESCRIPTION
ADCCTRLR	0x0200	R/W	0	ADC state machine control
ADCSTATR	0x0201	RO	0	ADC status
TRIGCFGR	0x0202	R/W	0	Trigger configuration
TRIGDLYR	0x0203	R/W	0	Trigger delay
ADCSHOTSR	0x0204	R/W	0	Number of shots
TRIGUTCLR	0x0205	RO	0	UTC low of last trigger
TRIGUTCHR	0x0206	RO	0	UTC high of last trigger
STARTUTCLR	0x0207	RO	0	UTC low of last start
STARTUTCHR	0x0208	RO	0	UTC high of last start
STOPUTCLR	0x0209	RO	0	UTC low of last stop
STOPUTCHR	0x020A	RO	0	UTC high of last stop
ADC1OFFSR	0x020B	R/W	0	ADC1 offset
ADC1SWITCHR	0x020C	R/W	0x30	ADC1 switches
ADC1VALR	0x020D	RO	0	ADC 1 current value
ADC2OFFSR	0x020E	R/W	0	ADC2 offset
ADC2SWITCHR	0x020F	R/W	0x30	ADC2 switches
ADC2VALR	0x0210	RO	0	ADC 2 current value
ADC3OFFSR	0x0211	R/W	0	ADC3 offset
ADC3SWITCHR	0x0212	R/W	0x30	ADC3 switches
ADC3VALR	0x0213	RO	0	ADC 3 current value

Table 6: Register set for the ADC controller block (1/2).

NAME	OFFSET	MODE	RESET	DESCRIPTION
ADC4OFFSR	0x0214	R/W	0	ADC4 offset
ADC4SWITCHR	0x0215	R/W	0x30	ADC4 switches
ADC4VALR	0x0216	RO	0	ADC 4 current value
ADCIDADDR	0x0217	R/W	0	ADC ID I2C address
ADCIDDATR	0x0218	R/W	0	ADC ID I2C data
ADCCLKLR	0x0219	R/W	0	ADC Clock frequency low
ADCCLKHR	0x021A	R/W	0	ADC Clock frequency high
ADCADDR	0x021B	R/W	0	ADC config address
ADCDATR	0x021C	R/W	0	ADC config data
SRATER	0x021D	R/W	1	Sample rate decimation
ADCPRER	0x021E	R/W	0	Pre-trigger samples requested
ADCPOSTR	0x021F	R/W	0	Post-trigger samples requested
LASTPOSR	0x0220	RO	0	Last sample position in DDR RAM
ADCSHOTCNTR	0x0221	RO	0	ADC shot sample counter

Table 7: Register set for the ADC controller block (2/2).

2.6.2 ADCSTATR

The current state in the state machine can be read from the ADCSTATR register. Idle = 0, Pre-trigger Count = 1, Wait Trigger = 2, Wait End of Shot = 3 and Decrement Shot Count = 4.

2.6.3 TRIGCFGR

The TRIGGER condition in figure 2 is to be interpreted as an 'OR' of hardware and software (SOFT_TRIG) triggers, followed by a delay specified in the TRIGDLYR register. Trigger configuration is handled through the TRIGCFGR register. Bits [31:16] are used for a threshold (treated as 2's complement and compared to the raw ADC data) in case of internal trigger, bit 0 selects between internal ('0') and external ('1') trigger and bit 1 selects positive ('0') or negative ('1') slope. Hardware and software triggers can be enabled using bits 2 and 3 respectively. A trigger applies to all 4 channels. Bits 4 and 5 in TRIGCFGR select a channel to use for the case of internal hardware trigger.

2.6.4 TRIGDLYR

The TRIGDLY register contains the number of adcclk 100 MHz (see 2.6.10) ticks to count between the occurrence of the trigger condition as expressed in TRIGCFGR and its taking into account in the ADC state machine.

2.6.5 ADCSHOTSR

The lower 16 bits are used to set the number of shots required in multi-shot applications. The upper 16 bits are ignored on write, and when read return the current value of the shot down-counter.

2.6.6 TRIGUTCLR, TRIGUTCHR, STARTUTCLR, STARTUTCHR, STOPUTCLR and STOPUTCHR

UTC time tags of last trigger (including trigger delay), last start comand and last stop command.

2.6.7 ADCxOFFSR and ADCxSWITCHR

Controlling the offset and gain of each ADC we have the ADCxOFFSR and ADCxSWITCHR registers, where x ranges from 1 to 4. The ADCxOFFSR registers are used to load a 16-bit DAC in the mezzanine, so only bits [15..0] are used. ADCxSWITCHR are in fact bit field registers, with each bit controlling an independent switch. These switches are used in normal operation to set gains, but can also be used for disconnecting the input signal from the ADC for automatic calibration purposes, and to switch between low and high impedance modes. More information can be found in http://www.ohwr.org/projects/fmc-adc-100m14b4cha. For the purpose of this specification, it is enough to say that there are seven switches per analog input channel and they will be mapped to the least significant bits of the ADCxSWITCHR registers, starting with SW1 in bit 0 and ending with SW7 in bit 6. Switches get turned on by writing a '1' to their associated control bit. The default state after reset is 0x30, which means all input amplifiers disconnected and an input full scale range of 10V.

2.6.8 ADCxVALR

The current ADC output value can also be read from a dedicated register for each channel. This value is accessible in the ADCxVALR registers.

2.6.9 ADCIDADDR and ADCIDDATR

The ADC FMC card has two I2C busses connected to it from the carrier FPGA through the FMC connector. The first one grants access to an FMC identification EEPROM on the mezzanine, which can be used to read/write the type of mezzanine, in agreement with the FMC standard. It is read and written using registers ADCIDADDR and ADCIDDATR. The FPGA puts the address plus a R/W flag into the ADCIDADDR register and then reads or writes from/to the ADCIDDATR. The location used for the read ('0') or write ('1') flag is bit 31. A read to ADCIDDATR must only be performed after the I2C controller has had time to get the data from the EEPROM. A write also has to be performed carefully, only after the previous write has succeeded. The I2C controller inside the ADC controller signals read and write completion through an interrupt request to the IRQ controller block.

2.6.10 ADCCLKLR and ADCCLKHR

There is a separate I2C bus for controlling the Si570 clock generator in the mezzanine. Our application assumes it will be programmed to provide a constant 100 MHz frequency (adcclk). Registers ADCCLKLR and ADCCLKHR contain the low and high parts of the frequency to be programmed into the Si570. Only the lower 16 bits of ADCCLKHR are used, for a total of 48 bits of frequency setting. Bit 31 of ADCCLKHR can be set to go into PLL mode, where the Si570 tracks the system clock (which can itself be derived from a White Rabbit link). In this PLL tracking mode, the other frequency-setting bits are ignored.

2.6.11 ADCADDR and ADCDATR

The ADC chip itself can be controlled through an SPI bus granting read and write access to its internal configuration registers. SPI reads and writes use the same mechanism as the other two serial busses in the mezzanine. The ADCADDR register is used for addresses and the ADCDATR register holds the data.

2.6.12 SRATER

Effective sampling rate is obtained by dividing the nominal adcclk 100 MHz from the Si570 by a variable amount, i.e. the FPGA will get data constantly at 100MS/s and will decimate it before writing into RAM. The decimation factor will be stored in bits [15..0] of SRATER.

2.6.13 ADCPRER and ADCPOSTR

The number of samples to be acquired before and after the trigger are common to all channels and stored in the ADCPRER and ADCPOSTR registers respectively.

2.6.14 LASTPOSR

This registed holds a byte address inside the DDR RAM, pointing to the last acquired sample for channel 1. Since samples are 16-bit wide and grouped by 2 in 32-bit words, LASTPOSR is always a number of the type 4n-1 with n unsigned.

2.6.15 ADCSHOTCNTR

Another counter running at the sampling rate counts the number of samples for a given shot. It is reset on START, counts up to the number of pre-trigger samples, waits for a trigger and continues counting up to pre-trigger + post-trigger samples. Its value can be accessed in the ADCSHOTCNTR register.

3 Future improvements

- White Rabbit support in the UTC core.
- Time-tags for all shot triggers in a multi-shot acquisition, maybe embedded in the DDR data.