# A new interface technique for the acquisition of multiple multi-channel high speed ADCs

Denis Calvet

***Abstract*—Multi-channel high speed ADCs with a serial output interface operating at several hundred Mbps have been introduced several years ago. Interfacing to these high speed devices poses new challenges to the designer. Existing techniques usually rely on delay locked loops, require several milliseconds to reach stable operation, and do not guarantee a fixed latency making the accurate synchronization of several multi-channel ADCs difficult to achieve. A new interface technique is introduced to overcome these limitations. We detail the proposed method and show an implementation where a single field programmable gate array is used to collect data from twenty four 12-bit ADC channels clocked at 20 MHz.**

***Index Terms*—Field Programmable Gate Arrays, high speed analog to digital converters, source synchronous interfaces.**

## I. Introduction

MULTI-CHANNEL analog to digital converters (ADCs) operating at several tens of Msps have been introduced to the market several years ago. Common devices contain 4 to 8 channels and offer a resolution of 8-bit to 12-bit at a sampling rate of up to 100 Msps [1], [2], [3]. Pin count limitations, package cost and signal integrity issues make the traditional unipolar CMOS parallel output interface unpractical for these devices. The scheme unanimously adopted by manufacturers uses serial links and low voltage differential signaling (LVDS). Building an interface to readout multiple ADCs poses new challenges to the designer of digital electronics: the aggregate data rate quickly exceeds several Gbps and multiple flows of data have to be synchronized consistently. We propose a new technique for interfacing to multiple multi-channel ADCs which is able to synchronize multiple channels with guaranteed zero-skew. It features deterministic latency and fast locking time. The proposed method is meant for implementation in low cost field programmable gate arrays (FPGAs). Internal FPGA resource usage is minimized: no internal delay locked loop (DLL) or programmable delay taps in I/O pad (a feature only found in latest generation devices) are used.

After a description of traditional interface techniques to high-speed ADCs, we expose the specific constraints of the intended application that brought the need for a new method. We describe our proposal in details and demonstrate the operation of a system where a single Xilinx Virtex 2 Pro FPGA is interfaced to 6 quad-channel 12-bit ADCs operating at 20 Msps. The proposed technique can easily be adapted to other digitizer boards that require aggregation of data from several multi-channel high speed ADCs in the same FPGA with guaranteed clock cycle-accurate synchronization.

Manuscript received April 30, 2008; revised June 26, 2008.

D. Calvet is with Commissariat à l'Energie Atomique, Institut de Recherche sur les lois Fondamentales de l'Univers, Centre de Saclay, F-91191 Gif-sur-Yvette Cedex, France (phone: +33-1-69086909; e-mail: calvet@hep.saclay.cea.fr).

## II. Common Techniques for Interfacing to ADCs

### A. Parallel unipolar interface

Single channel, medium to high-speed ADCs almost always use a parallel output interface. The width of the parallel output bus is determined by the resolution of the converter. Unipolar signaling (low voltage CMOS) is the most common standard. The interface to such device is conceptually simple: a clock (derived from the sampling clock of the ADC) triggers a parallel register to latch the converted data. In order to reduce the impact of digital current switching on noise sensitive analog parts, current limiting resistors are often placed between the digital outputs of the ADC and the receiver logic. Although this scheme is adequate for single channel ADCs at up to ~100 Msps, it would not be effective for multi-channel devices: a quad-channel 12-bit ADC would require a 48-bit wide parallel bus. This would be inefficient in terms of pin count and becomes extremely difficult to operate as speed is increased: signal skew is hard to manage on wide busses due to unavoidable trace length mismatches, and ground bounce issues arise when large number of bus lines switch simultaneously. To overcome these limitations, silicon vendors have introduced two concepts: serial interfaces (to address the pin count limitation and signal skew issues) and differential signaling (to improve signal integrity).

### B. Conventional serial differential interface

We briefly recall the principle of operation of a typical high speed serial output ADC. Using the sampling clock as a reference, the ADC generates a high speed clock (DCO) to serialize data on its data output lines. Double data rate clocking is used and, for example, the DCO clock is 6 times faster than the sampling clock for a 12-bit ADC. The ADC also generates a framing signal (FCO), phase aligned with the serial data. The DCO clock is intended for data capture. The FCO signal can be used as a clock to latch data when it is

aligned, or it can be seen as a pseudo data channel that outputs a constant framing pattern (e.g. “111111000000”). This can be used to retrieve word boundaries in the serial stream. One of the simplest methods to interface to such ADC device is described in [4]. It is pictured in Fig. 1.

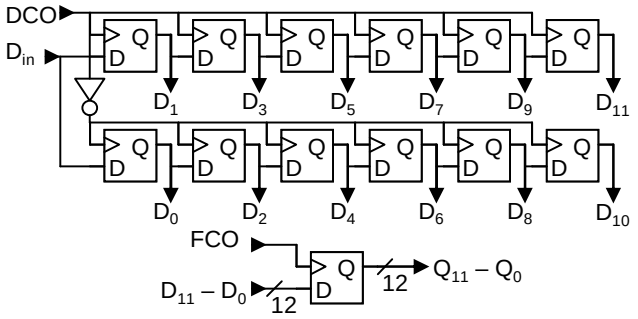


Fig. 1. Conventional de-serializer interface to a serial output 12-bit ADC.

The high speed DCO clock controls a dual 6-bit shift register to de-serialize the incoming serial data $D_{in}$ on both edges of the clock. It is assumed that the most significant bit, $D_{11}$ is shifted-in first. The framing clock FCO is used to latch parallel data after de-serialization when it is properly aligned on 12-bit word boundaries. Although simple, this scheme is not suitable for interfaces running at high speed (e.g. over ~30 Msps) because it cannot compensate for data and clock skews. A more robust scheme is described in [4]. It uses a delay locked-loop to produce 8 copies of the DCO clock shifted by 1:8$^{th}$ of a period. Eight shift registers skewed by 1:8$^{th}$ of a clock period capture data, and arbitration logic dynamically selects the optimal output. This scheme is effective but it consumes a substantial amount of logic, one DLL and at least 8 clock tree nets per interface. The quantity of DLLs and global clock networks available in low cost to mid-range FPGAs is typically limited to 8 and 16 respectively. Global clock trees and DLL blocks are generally needed in other parts of the design and cannot be entirely assigned to the interface to ADCs. Embedded DLLs that provide 8 outputs are also uncommon. Xilinx “Digital Clock Manager” blocks provide up to 4 outputs shifted by 45°. A design based on this technology is described in [5].

Another scheme for interfacing to a serial-output ADC consists in using some programmable delay on clock and data lines. Specific logic is used to determine the optimal delay setting to deskew each individual data line with respect to the fast clock (DCO) provided by the ADC. Delay settings can be static, i.e. they are calibrated once, or they can be monitored and adjusted in real-time. This method is generally used for high speed source synchronous interfaces such as memory interfaces [6]. Adjustable delay lines are only available in the most recent FPGAs. In Xilinx parts, the Virtex 2 Pro and Spartan 3 only allow one optional delay to be set in an input pad, Spartan 3E and Spartan 3A families have 6 and 8 input delay settings respectively; the Virtex 4 and Virtex 5 families have a fully dynamically controllable 64-tap delay line per input pad.

## III. Need for a New Interface Scheme

The motivation for a new scheme to interface to serial output ADCs comes from the requirements of the readout electronics for the time projection chambers in the T2K experiment [7]. In this application, a single digital front-end mezzanine card (FEM) receives data from 6 analog front-end cards (FECs). Each FEC contains 4 “AFTER” ASICs [8] that are readout by a quad-channel 12-bit ADC (Analog Devices AD9229 [9]) clocked at 20 MHz. The net throughput of each individual ADC channel is 240 Mbps (120 MHz double data rate) leading to an aggregate bandwidth of 5.76 Gbps for the 24 ADC channels connected to an FEM board.

The AFTER chip comprises 72 charge sense amplifiers coupled to a 72 x 511 time buckets switched capacitor array (SCA). During the sampling phase of detector signals in the SCA, sensitive front-end pre-amplifiers must be kept in the lowest possible noise ambiance. To achieve that goal, the clock of the ADCs is turned off and the receiver logic is placed in standby mode. Upon the reception of an external trigger signal, the content of the SCA in all AFTER devices is frozen. The clock is applied to each ADC, the receiver logic is woken up, synchronized and digitization takes place. Current leakage in the SCA imposes a minimum digitization rate of 20 Msps to perform the complete readout of all AFTER chips in less than 2.5 ms. The dynamic control of the clock applied to the ADCs imposes selecting a device that has a low wake-up time and requires the receiver logic to re-synchronize quickly. For our application, we require the total re-synchronization time be less than 200 µs, from the time the clock is applied to the ADC until valid data can be digitized. The dispersion of the re-synchronization time from one acquisition to the next must also be minimized. A peak-to-peak variation of 100 µs is acceptable in our case. PLL-based techniques or methods that require a delay calibration procedure usually suffer from potentially long (e.g. several milliseconds) and unpredictable settling times. These methods are inadequate for our needs.

The sequence of samples output by each AFTER chip has to be precisely retrieved by the FEM board: all 24 streams of data must be timely aligned to synchronized before further processing. Each of the 24 ADC and receiver logic paths must have equal, predictable, repeatable and deterministic latency. The required precision is one sampling clock period (i.e. 50 ns in our application).

Other constraints are more common: we require the logic be sufficiently simple to fit in a small/medium size FPGA target, optimizations should be made to conserve power and favor the use of a low speed grade device (for cost reasons). Because none of the interface techniques reported so far are able to satisfy simultaneously all the constraints previously listed, we propose a new scheme. It is based on a combination of several known concepts and original improvements.

## IV. Choice of the ADC Device

As previously stated, our application requires the clock applied to the ADC be periodically suppressed/re-applied and

correct operation of the ADC be restored in few tens of µs. Not all ADC devices support dynamic control of the sampling clock and fast recovery time from such idle state. This capability and the corresponding recovery time are usually not documented in component datasheets. This parameter is particularly critical for serial output ADCs because these devices have some intrinsic uncertainty on the wakeup time because an on-chip PLL is used in the output interface. We investigated 2 candidate devices: the AD9229 from Analog Devices and the ADS5240 from Texas Instruments. Both manufacturers were questioned and we used evaluation kits to test the parts ourselves. We found that the AD9229 is able to reach stable operation within ~35 µs on average from the time the sampling clock is applied. For the ADS5240, locking times in excess of several milliseconds were occasionally observed and investigations of that part were not pursued. We show in Fig. 2. the histogram of the wake-up time of the AD9229 alone and (in anticipation) the total synchronization time of the ADC interface that is described in this paper. The wakeup time of the ADC is defined from the time the sampling clock is applied to the time an arbitrary chosen number of clock cycles (16 in our case) are counted on the fast clock output DCO.

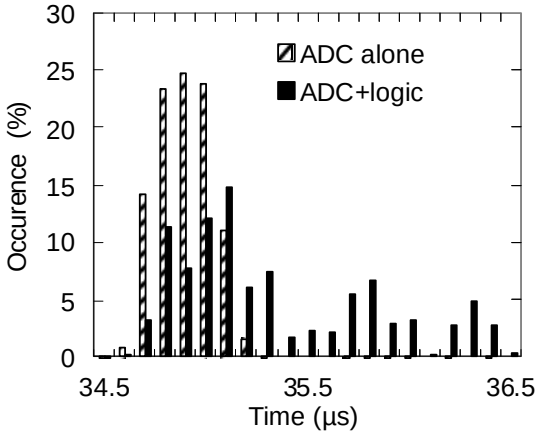


Fig. 2. AD9229 wakeup time and global synchronization time.

The total synchronization time is measured from the instant when the clock is applied to the ADC until the time the receiver logic detects the framing pattern on the FCO line more than twice in a row to avoid spurious detection. We checked in an independent series of measurements that the nominal performance of the ADC is reached within the total synchronization time. All the requirements on the wakeup time (mean, peak-to-peak and maximum value) for our application are comfortably met. In addition to a fast recovery time we measured that the AD9229 draws 10% less power when the sampling clock is idle. The improvement is appreciable, although marginal compared to the low power mode of the device (97% power reduction compared to the active state, but the recovery time from that state is 4 ms).

## V. Detailed Description of the Interface Scheme

### A. General concept

The block diagram of the proposed ADC interface logic is shown in Fig. 3. The logic is entirely implemented in an FPGA, although our application requires an external memory due to the large amount of data to be buffered (~1.2 MByte per event). The sampling clock of all ADCs is provided by the FPGA. The data capture logic block is responsible for capturing the data lines and the framing pattern line of each multi-channel ADC. A source-synchronous interface is used at that level: data capture is clocked by the fast clock (DCO) provided by each ADC. There is one data capture block per multi-channel ADC, each resides in its own clock domain.

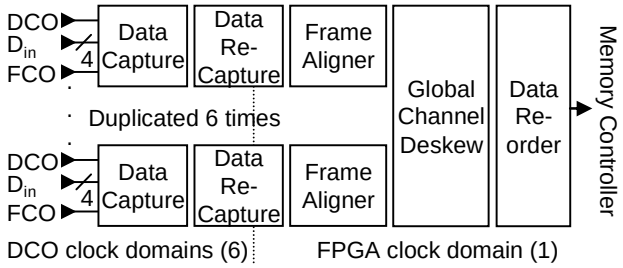


Fig. 3. Block diagram of the proposed multi-channel ADC interface.

The data re-capture logic is used to cross each DCO clock domain to bring all captured data in a common clock domain, called the FPGA clock domain ($F_{clk}$). The next blocks are the frame aligners. These delineate serial data to retrieve 12-bit word boundaries. Once this is done, global channel deskew is performed. This operation compensates for the differences in latency between the multi-channel ADCs. This block guarantees that all the digital samples at any given $F_{clk}$ clock cycle correspond to the same sampling clock period at the analog inputs of all ADCs. The next stage is data re-ordering. This block re-organizes data to facilitate storage and fast retrieval to/from the external memory.

### B. Data capture

The data capture logic is pictured in Fig. 4. Double data rate LVDS input pads (with the delay option enabled) are used to drive the two flip-flops embedded in an I/O pad. These are followed by 2 flip-flops located in the FPGA fabric, next to the I/O pad. To avoid the use of any DLL, the data capture logic is clocked directly by the DCO signal supplied by the ADC. An inverter is automatically inferred by the synthesis tool to clock data on the falling edge.

The location of all input pads must be carefully chosen to minimize internal routing delays. The DCO clock signal deserves specific attention. Pad location must be chosen for the optimal routing of the local clock tree net [10]. It is necessary to turn on the optional delay in the input pad of the DCO signal and apply some appropriate timing constraints. Manual placement of individual flip-flops in the FPGA fabric was not found necessary for operation at 120 MHz (20 Msps for the ADCs) in the slowest speed grade of a XC2VP4 FPGA (Xilinx Virtex 2 Pro family). On the other hand, imposing that the FIFO of the data re-capture logic (described later) is placed at the location closest to the data capture logic is required to meet timing constraints.

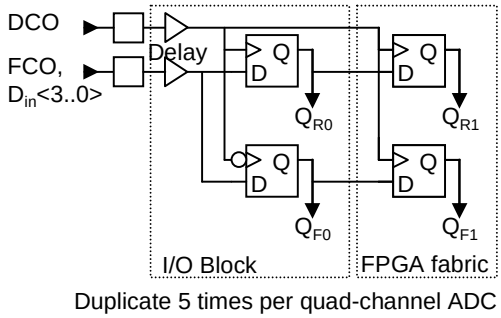


Fig. 4. Data capture logic.

The physical layout of a data capture logic block for an interface located on the left side of the FPGA die is shown in Fig. 5. Routing is cleaner on the left side of the die because logic resources inside the FPGA are built by translation of some design pattern while mirroring around a left/right symmetry axis would be required for optimal routing on the right side of the device. Higher speed designs would probably require manual placement of all the flip-flops of the data capture block, and should preferably connect to the external ADCs via the row of input pads located on the left side of the FPGA die.

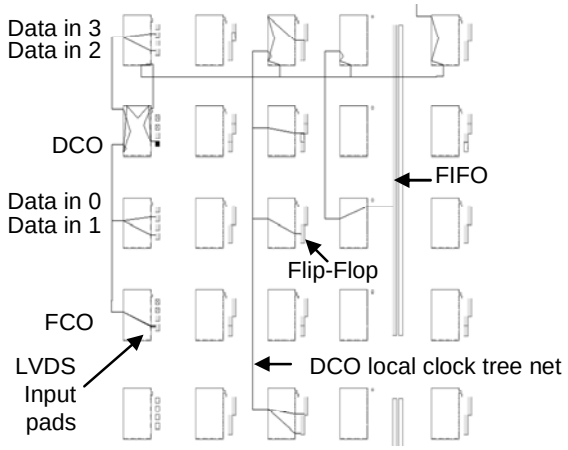


Fig. 5. Physical layout of a data capture block.

The data capture block is duplicated 5 times per multi-channel ADC. The output of that block consists of 5 four-bit wide busses clocked in the DCO domain ($Q_{R0}$, $Q_{R1}$, $Q_{F0}$ and $Q_{F1}$). Four busses correspond to the 4 data channels of the ADC; the fifth bus corresponds to the framing pattern line (FCO). Only three out of each series of six consecutive 4-bit words on any particular bus need to be accumulated to build complete 12-bit samples. The framing pattern "111111000000" is always output in a fixed sequence of six 4-bits words of the form: {"1111"; "1111"; "1100"; "0000"; "0000"; "0011"} and only one word out of two has to be retained to retrieve the correct pattern.

The data capture block performs a partial de-serialization: each 1-bit double data rate 120 MHz differential line is converted to a 4-bit wide bus that can be sampled at 60 MHz. This provides some optimal trade-off between internal speed and bus width. The global interface to 6 quad-channel ADC leads to a 120-bit wide internal bus clocked at 60 MHz. A complete de-serialization into 12-bit words as in [4] would need an unpractical 360-bit wide bus clocked at 20 MHz. Performing only double data rate to single data rate conversion at this stage would lead to a 60-bit wide bus clocked at 120 MHz which is not trivial to deal with inside an FPGA fabric.

### *C. Data re-capture*

The data re-capture logic is used to transport the data captured by the front-end logic from the 6 different DCO clock domains (one domain per multi-channel ADC) to a common clock domain ($F_{clk}$), inside the FPGA. The structure of this block is shown in Fig. 6.

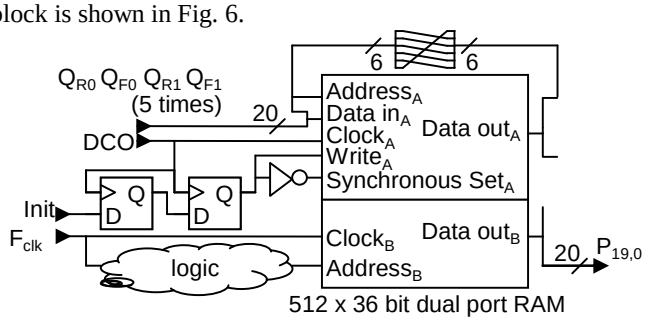


Fig. 6. Data re-capture logic.

We use a 512 word 36-bit synchronous dual port memory block configured as a self-addressing FIFO [5] to perform this operation. The depth of the FIFO is set to 6 words. This offers the guarantee (by design) that no more than one complete 12-bit sample is stored in a FIFO slice at any time. This aspect is important to ensure that no unpredictable delay greater than one ADC sampling clock period can possibly be introduced when crossing clock domains during data-recapture. The width of the input data bus of the FIFO is 20 bits in order to interface directly to the data capture logic of the quad-channel ADC. The feedback path of the write section of the FIFO takes 6 data lines and uses one-hot encoding for addresses. Consequently, no logic is needed in the feedback path: the next address is simply obtained by a hardwired cyclic shift of the current address by one bit. Avoiding the need for logic in the feedback path enables the highest possible operating speed and does not reduce the skew margin on the local DCO clock tree net because there isn't any flip-flop to clock at that level. The feedback path of the read section of the FIFO shifts the read address by 2 bits instead of one on every FPGA clock cycle: this enables reading out one word out of two being written in the FIFO, i.e. the read side of the FIFO operates at half the frequency of the write side. This is particularly important in high speed designs because only flips flops located inside (or close to) the I/O pads can be clocked at high frequency, while those distributed in the FPGA fabric are generally limited in speed by internal routing delays caused by the comparatively long distances to cover. Note that the feedback path of the read side of the FIFO contains flip-flops, but this is not critical because a global clock tree net is used on this side, and operating frequency is halved compared to the write side. The initial content of the block RAM used as a FIFO needs to be properly set and the initial value of the write pointer (DCO clock domain) and read pointer (FPGA clock domain) need to be initialized every time the logic needs re-synchronization.

This is achieved by a preset of the output registers of the dual port RAM via the Init signal. Note that two cascaded flip-flops are used to avoid metastability when transporting the Init signal from the FPGA clock domain to the DCO clock domain.

The output of the re-capture logic of each multi-channel ADC is a 20-bit wide bus (i.e. 120-bit wide in total) clocked at 60 MHz. Three consecutive 20-bit words are needed to assemble the four 12-bit samples that correspond to the conversion of the analog inputs and the 12-bit framing pattern. Depending on the (unknown) skew of the preset signal with respect to the sampling clock of each ADC, there are only 6 possible sequences being written into the FIFO for the framing pattern. These sequences are listed in Table I.

TABLE I. LIST OF ALL POSSIBLE SEQUENCES WRITTEN INTO THE DATA RE-CAPTURE FIFO FOR THE CONSTANT FRAMING PATTERN "111111000000".

| Address | case1 | case2 | case3 | case4 | case5 | case6 |
|---|---|---|---|---|---|---|
| 0x01 | 1111 | 0000 | 0011 | 1111 | 1100 | 0000 |
| 0x02 | 1100 | 0000 | 1111 | 1111 | 0000 | 0011 |
| 0x04 | 0000 | 0011 | 1111 | 1100 | 0000 | 1111 |
| 0x08 | 0000 | 1111 | 1100 | 0000 | 0011 | 1111 |
| 0x10 | 0011 | 1111 | 0000 | 0000 | 1111 | 1100 |
| 0x20 | 1111 | 1100 | 0000 | 0011 | 1111 | 0000 |

For each possible case, there is one and only one address read sequence that allows recovering the correct framing pattern. For cases 1, 2 and 3, the addresses to consider are 0x02, 0x08 and 0x20 while for cases 3, 4 and 5, only data at address 0x01, 0x04 and 0x10 shall be read. Proper alignment is obtained with the read address sequences: Seq_1={0x20; 0x02; 0x08}, Seq_2={0x08; 0x20; 0x02}, Seq_3={0x02; 0x08; 0x20}, Seq_4={0x01; 0x04; 0x10}, Seq_5={0x10; 0x01; 0x04} and Seq_6={0x04; 0x10; 0x01} for case 1, 2, 3, 4, 5 and 6 respectively.

### *D. Frame alignment*

The frame alignment logic is shown in Fig. 7. It consists in a finite state machine that tries to identify the framing pattern in 3 consecutive 4-bit words output by the data re-capture logic.

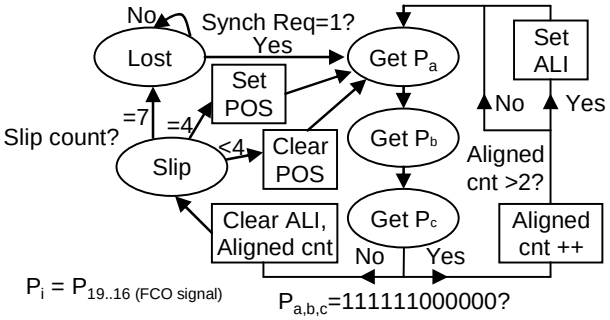


Fig. 7. Frame alignment logic.

Upon request for synchronization (signal Synch. Req), the initial value of the FIFO read pointer and write pointer are set to 0x02 and 0x01 respectively. The sequence of FIFO addresses being read is therefore Seq_3 = {0x02; 0x08; 0x20}. Three consecutive 4-bit words, $P_a$, $P_b$ and $P_c$, corresponding to the framing pattern lane are concatenated and compared against the expected 12-bit constant. If alignment is not found, a slip of one $F_{clk}$ clock cycle is introduced. If alignment is still not obtained after trying the 2 circular shifts of Seq_3 (i.e. Seq_1 and Seq_2); the alternate family of FIFO read address sequences, Seq_4, Seq_5 and Seq_5 are successively tried. In normal operating conditions there is always one and only one sequence of addresses that lead to correct frame alignment. Synchronization failure at this stage is a fatal error: the frame alignment logic stays in the "Lost" state until the next synchronization request. Note that the frame aligner block does not guarantee that the address on the write side of the FIFO always differ from that on the read side. This ill-defined situation is however extremely improbable because the clock on the write side (DCO) is frequency locked to that on the read side ($F_{clk}$): the two clocks do not drift with respect to each other and only a fixed (but unknown) phase shift between the 2 clock domains needs to be absorbed by the FIFO. Although this was not found necessary in our implementation, a more sophisticated scheme for clock synthesis could be devised to control the phase shift between the ADC sampling clocks (and consequently that of the DCO clocks) and the local FPGA clock to guarantee an optimal time margin.

The signal ALI is asserted when the framing pattern has been found more than twice in a row and is de-asserted as soon as one incorrect framing pattern is found. The lock time of the frame alignment logic is unpredictable but has fixed boundaries (only 6 possible sequences of read address have to be tried). The latency introduced by the data re-capture block reaches a different value depending on which of the 6 possible sequences of read addresses leads to frame alignment. In our implementation, we found that the delay mismatch between the 6 multi-channel ADCs is sufficiently low to ensure that only 2 different values of latency are ever reached: the 2 ADCs closest to the FPGA always exhibit one $F_{clk}$ clock cycle latency less than the 4 ADCs located further away. A single bit (POS signal) per ADC interface is used to make the distinction between the two possible latency values.

### *E. Global deskew*

As previously explained the data re-capture block skews data because the latency introduced by the FIFO used to cross clock domains differs from one multi-channel ADC to the other. This skew must be compensated to ensure that all ADC samples are aligned to the same $F_{clk}$ clock cycle. This is achieved on a per ADC basis by applying a delay of one $F_{clk}$ clock cycle to the re-captured data when the corresponding POS signal is asserted.

The ALI signals of all ADCs are AND'ed together to indicate the completion of the alignment procedure at the global level. Each ALI signal can be individually masked to allow correct operation when only a subset of the ADCs is present. Masking also allows operation in a degraded mode when one or several ADCs cause frequent or permanent synchronization failures.

### *F. Data formatting*

At this stage, data are re-organized for storage in the external memory buffer. The three 4-bit words that make each 12-bit sample are converted into two 6-bit words. This conversion is easily achieved in pipe-line logic. The output of this block consists of one 30-bit wide bus clocked at 60 MHz for each multi-channel ADC. These six busses are time multiplexed towards an external memory (2 zero-bus turnaround Cypress CY7C1354CV25 static RAM) over a 60-bit wide data bus operating at 120 MHz. The two 6-bit words that make a 12-bit sample are stored at consecutive addresses in the external memory (burst of 2 access during write and read). Because the frequency of the external RAM is exactly twice that of the internal FPGA logic, only one $F_{clk}$ clock cycle is needed to access 12-bit data samples stored in the external RAM.

In our application, the operations performed on the data after temporary storage comprise an optional pedestal equalization, zero-suppression with a programmable threshold value per channel, and transmission towards back-end electronic cards via a 2 Gbps optical link. The description is outside of the scope of this paper.

## VI. Implementation

The logic previously described has been fitted into a Xilinx XC2VP4 (672 pin BGA package; slowest speed grade) placed at the center of the FEM board. It interfaces to each quad-channel ADC of the 6 FECs connected to the FEM board via right-angled rear side connectors. This is shown in Fig. 8.

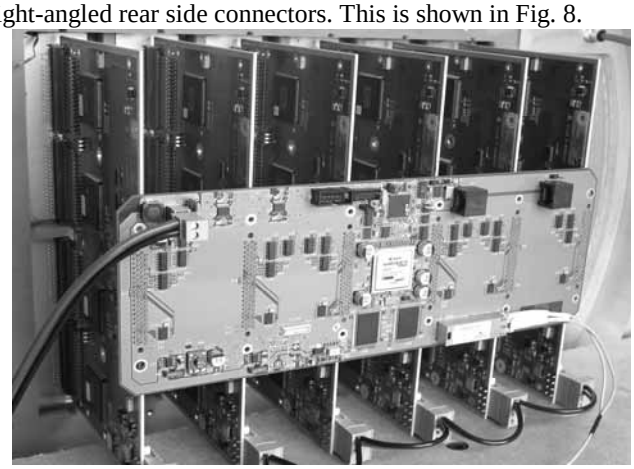

Fig. 8. View of the FEM board connected to 6 FECs.

The distance between the FPGA and the ADC is ~4 cm for the 2 cards located close to the center of the FEM board and reaches ~18 cm for the farthest cards. Extensive behavioral simulations have been made and tests of several FEM cards and FECs have shown very stable operation over many days.

## VII. Conclusion and Perspective

A new scheme for interfacing several multi-channel high speed ADCs to programmable logic has been described. The main advantages compared to designs reported so far are: a minimal use of internal PLL/DLL and global clock networks in the FPGA, guaranteed sampling clock accurate alignment of data on all channels, fast locking time (less than 100 µs), and internal rate conversion to run the core of the logic at one quarter of the speed of the external serial lines.

The source synchronous-side of the interface uses direct clocking. The main limitation of the proposed implementation is that fixed delays are used to compensate the skew of the data lines with respect to the clock used for capture. It is estimated that this would limit operation to 30 Msps ADCs. Using a more sophisticated FPGA device (Virtex 4 or Virtex 5) would allow higher speed because finer control on the input pad delay is necessary to bring data in the middle of the capture window. The limitation on the number of multi-channel ADCs that can be controlled by the same FPGA is primarily determined by the complexity of the printed circuit board. Connecting six quad-channel ADCs to the FPGA on the FEM board was very difficult using a 12-layer class 7 PCB (100 µm traces, 120 µm spacing). A design that comprises 8-10 quad-channel ADCs is probably close to the limits of what can be built at an affordable cost with today's industrial technology.

The proposed interface technique will be deployed on the 72 digitizer boards for the readout of the time projection chamber in the T2K experiment. The technique can be transposed to other designs where the aggregation of data delivered by multiple multi-channel high speed ADCs in a time coherent fashion is required.

## Acknowledgment

The author wishes to thank the main contributors to the design of the time projection chamber readout electronics for the T2K experiment: P. Baron, X. de la Broïse, C. Coquelet, E. Delagnes, F. Druillole, A. Le Coguie, E. Monmarthe, E. Zonca and particularly D. Besin who laid out the FEM board.

## References

[1] "ADS5273, 8-Channel, 12-Bit, 70MSPS Analog-to-Digital Converter with Serial LVDS Interface", component datasheet, Texas Instruments, September 2005. online: http://www.ti.com

[2] "AD9287 Quad, 8-Bit, 100 MSPS Serial LVDS 1.8 V A/D Converter", component datasheet, Analog Devices, July 2007. online: http://www.analog.com

[3] "ADC12EU050 Ultra-Low Power, Octal, 12-bit, 40-50 MSPS Analog-to-Digital Converter", component datasheet, National Semiconductor, January 2008. online: http://www.national.com

[4] G. Dutta, N. Viswanathan and A. Udupa, "Interfacing High Speed LVDS outputs of the ADS527x/ADS524x", Application report SBOA 104, Texas Instruments Inc., February 2005.

[5] M. Defossez, "Connecting Xilinx FPGAs to Texas Instruments ADS527x Series ADC", Application note XAPP774, Xilinx Inc. November 2004.

[6] M. George, "Memory Interfaces Data Capture Usign Direct Clocking Technique", Application note XAPP701, Xilinx Inc. November 2004.

[7] Y. Yamada, "The T2K program", Nuclear Physics B (Proc. Suppl.) vol. 155, pp. 28–32, 2006. online: http://www.sciencedirect.com

[8] P. Baron et al., "AFTER, an ASIC for the Readout of the Large T2K Time Projection Chambers", Nuclear Science Symposium Conference Record NSS 2007, Vol.3, pp.1865-1872.

[9] "AD9229 Quad 12-Bit, 50/65 MSPS, Serial LVDS A/D Converter", component datasheet, Analog Devices, September 2005.

[10] E. Eto and L. Lewis, "Local Clocking Resources in Virtex-II Devices", Application note XAPP609, Xilinx Inc. November 2004.