An interface solution at 53.76 Gb/s input bandwidth to a single Xilinx Virtex-II Pro FPGA – a practical challenge

A.D. Oltean Karlsson* and G. Tröger**
* CERN Geneva Switzerland and "Politehnica" University Bucharest, Romania
** Kirchoff Institut für Physik, Universität Heidelberg, Heidelberg, Germany

CERN-OPEN-2006-032
01 / 02 / 2006

***Abstract*—When High Energy Physics meets high speed electronics, and state of the art methods fail to deliver the required performance, alternative methods have to be developed. This paper presents a novel solution for hardware trigger processing. Analog signals from 112 inputs are converted into high speed serial data with 12 bit resolution, representing a bandwidth of 53.76 Gb/s of trigger data streamed into a single Xilinx Virtex-II Pro FPGA node. The system automatically corrects for clock phase misalignments of 112 channels, each of those being received at 480 Mb/s. This solution has been implemented in the Trigger Region Unit (TRU) of the ALICE Photon Spectrometer (PHOS) detector, a highly integrated board for processing analogue signals received via intermediate Front-End Electronics cards from a large matrix of PWO crystals.**

I. PHOS TRIGGER REGION UNIT

The PHOS Trigger Region Unit is part of the trigger system of a high energy physics experiment. It is performing local signal processing on input data from one of the detectors, looking for specific trigger conditions. The results are then sent to the global trigger system, to be merged with other information from other subsystems. One of the key issues was receiving the amount of input data in the single FPGA used for the processing. In the following paragraphs we will introduce the full problem.

A. PHOS – ALICE's Calorimeter

ALICE (A Large Ion Collider Experiment) is one of the four experiments at the Large Hadron Collider (LHC) hosted by CERN in order to study the physics of strongly interacting matter at extreme energy densities. The PHOS (PHOton Spectrometer) of ALICE is an electromagnetic calorimeter which measures electromagnetic showers of up to 100 GeV, by using a large matrix of 17920 lead tungstate (PWO) scintillator crystals, grouped within five PHOS modules. One PHOS module consists of eight trigger region units (TRU), each of those regions covering a matrix of 16x28 crystals.

B. PHOS Front-End and Trigger Electronics

The first ideas of PHOS were described in [6]. Particle hits inside the PHOS detector generate light in the crystals which is converted into analogue signals by Avalanche Photo Diodes (APD) and amplified by charge sensitive pre-amplifiers (CSP) located on a small PCB glued to the end of each crystal. Inside of each TRU region, the analogue signals coming from the crystals are passed to PHOS' Front End Electronics (FEE), represented by 14 FEE cards which perform the shaping, digitization, buffering and summing of analog signals from 2x2 crystals. 112 analogue sums generated by fast summing shapers on the 14 FEE cards are interconnected and transmitted via equal-length differential cables to a central Trigger Region Unit (TRU) card. The connectivity between crystals, FEE and TRU is shown in Fig. 1.

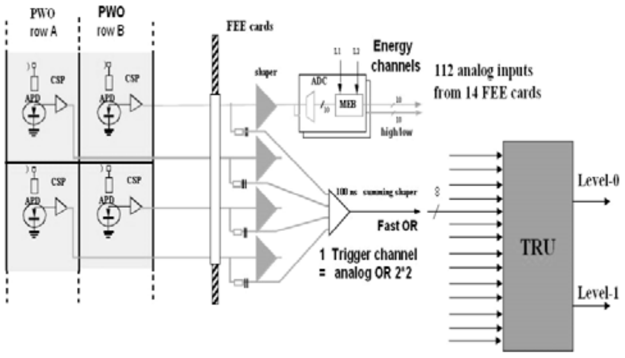

Figure 1. Connectivity overview: crystals-FEE-TRU

C. Trigger Region Unit (TRU)

The TRU board itself contains 14 analog-to-digital converters (ADCs) which digitize a total of 112 input analogue signals, received from the FEE cards, and sends them to a central Xilinx Virtex-II Pro (XC2VP50) FPGA. The particle hit information, corresponding to a TRU region, is processed using a parallel algorithm inside the FPGA, and as result, regional TRU level-0 and level-1 triggers must be generated within 300 ns (level-0), respectively 5.5 us (level-1) latency in the FPGA [7]. For a faster computation of the algorithm, the information received from the ADCs, has to be first deserialized inside the FPGA, as explained later in *II.A*.

D. Digitization Approach

Each of the 14 ADC devices on the TRU board is an eight channel 12-bit Texas Instruments ADC (ADS5270) working at a 40 MHz sampling rate [5]. The digitized outputs from each ADS5270 channel are sent to the XC2VP50 FPGA as serial 12-bit double data rate (DDR) bit streams, edge-aligned with a 90 degree shifted 240 MHz high-speed (lock) clock. In addition, the ADCs transmit a delayed 40 MHz frame clock used for detecting the start of the bit stream frame. Fig. 2 shows the timing diagram of an ADS5270 channel.

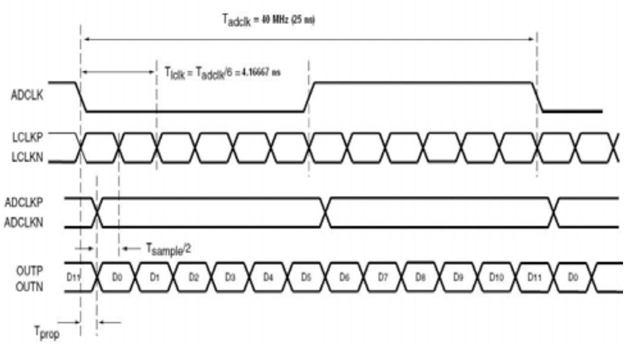

Figure 2. LVDS timing diagram per ADC channel

E. *TRU Board Layout*

112 analogue inputs from 14 ADCs, digitized to 12 bit precision in 25 ns intervals, represent a continuous input bandwidth for the Xilinx Virtex-II Pro of 53.76 Gb/s. This vast amount of binary information is transmitted via 112 serial LVDS lines which interface the ADCs to the XC2VP50 FPGA at a bit rate of 480 Mb/s.

The TRU board design and its 11-layer printed circuit board (PCB) were finalized in autumn 2005. Particularly difficult for the layout was the differential, matched impedance routing of the LVDS lines between the 14 ADCs up to the FPGA's dense ball grid array of 1152 pins. In order to preserve the original phase shift between the data and the clock up to the FPGA, these traces have been extremely carefully designed and routed for having the same length. A snapshot of the TRU board showing the top layer with 7 ADC devices placed around the Xilinx FPGA can be seen in Fig. 3. The other 7 ADCs are placed on the bottom layer and they are arranged in a similar way around the central FPGA.

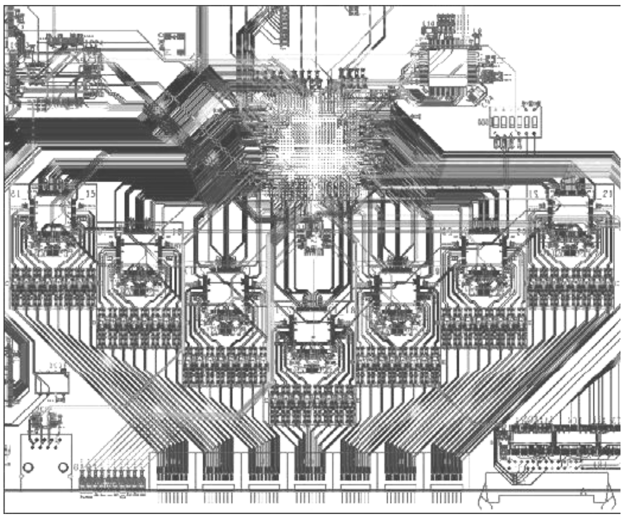

Figure 3. TRU board layout

F. *TRU – Aspects of the problem*

At the FPGA, the incoming data bits should be registered at the right moment, in the middle of the clock period. In practice, process variations such as temperature and voltage, in addition to modifications in the final wafers introduced by the PCB production process can easily determine a different phase alignment between the clock and the data at the arrival to the FPGA's inputs. Without a TRU board available yet, the real variation between clock and data, defined as skew, is not known and cannot be predicted a priori. It might be that clock and data arrive correctly aligned, but we don't have any definitive information available and no possibilities to test prove.

To compensate and eliminate any unwanted skew, Xilinx recommends the use of Digital Clock Managers (DCMs) when clocking an incoming stream of data at very high frequencies into their FPGAs. A DCM allows fine grained, fixed or dynamical, clock phase adjustments so that the incoming data can be registered correctly.

The XC2VP50 on the TRU board contains a total of 8 such DCMs. As all 14 ADCs are clocked with the same sampling frequency generated by the FPGA, we originally thought that it would be enough to use a single DCM to adjust only one of the 14 lock clocks and use this to register all the 112 LVDS data streams. A major issue against this idea arises from the ADC supplier, Texas Instruments, which could not guarantee a fixed phase alignment between any of the two ADC's PLLs, even though they are both originally clocked from the same reference clock. This meant that each bunch of 8 LVDS data streams from an ADC must be registered using its individual lock clock. This appeared to be an important problem and in the same time a practical challenge to us, as we could not use state of the art approaches presented in section *II*. However we found a solution, hereby presented in section *III.*, to properly connect all 112 high-speed 480 Mb/s channels to a single XC2VP50 FPGA.

II. STATE OF THE ART

A. *Single ADC Interface*

An Application Note describing the interface of a single Texas Instruments ADC from the family ADS527x to a Xilinx Virtex-II Pro FPGA has been provided by Xilinx [1] and it is summarized here.

To overcome any existing doubts on the data-to-clock alignment, a DCM is used in dynamic phase shift mode to register the serial data and the clock from the ADC into the FPGA [2]. Once the data is provided with a known phase shift relative to the 240 MHz lock clock, it is then input to the deserializer, whose architecture is presented in Fig. 4.

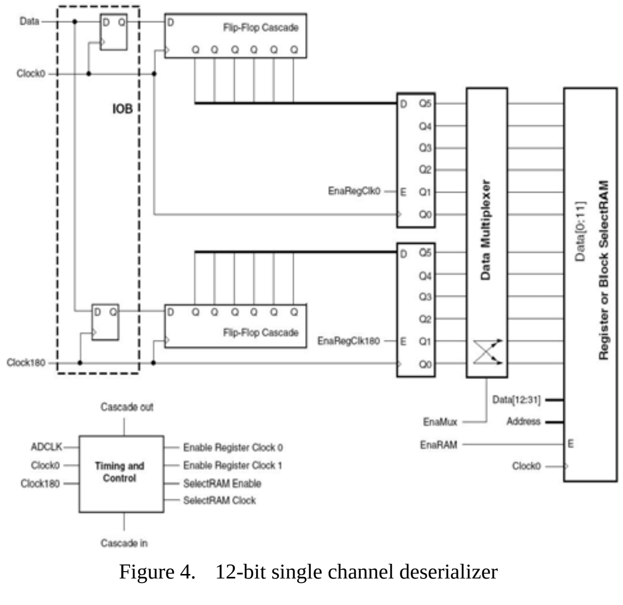

Figure 4. 12-bit single channel deserializer

The 12 incoming DDR bits are split into two sections – odd and even – and the 0 (positive clock edge) and 180 degree (negative edge) of the lock clock are used for their registration. When all the 6 bits corresponding to a frame have been registered in the two flip-flop cascades, an enabling signal is generated by the timing control module, allowing the storage of the odd and even bit streams in two separate parallel registers. To overcome any possible bit swapping, due to an earlier arrival of the negative clock edge prior the positive one (with respect to the frame clock), a multiplexer is used to correctly align in a 12-bit register the data. Once the bits are reordered, the output of the multiplexer can be fed to a set of registers, such as a Block SelectRAM FIFO or a memory.

All the enabling signals for the deserializer logic are generated inside a timing and control module, based on the edge detection of the 40 MHz frame clock, which is in phase with the first bit of the stream.

The major drawback of the deserializer solution, as described above, stands in the use of one DCM per ADC. As there are only 8 DCMs available in XC2VP50 and we would need 14 of them only to implement the ADC interfaces, the solution cannot be applied such as it is. It would exceed the available DCMs resources of the XC2VP50 FPGA on the TRU board by over 75%.

However, if we can provide the necessary clock and data inputs, synchronized and correctly aligned, the deserializer module can be possibly reused as explained later in section *III.C*.

To achieve the clock phase control for the whole set of ADC devices on the TRU board with a limited subset of DCMs and to make possibly use of the proposed single ADC interface, we analyzed several options.

One of these options was to dynamically switch them. While in principle, the global clock buffers can be switched between two different sources, we would require input selection from several pre-set, phase-shifted sampling clocks in addition to the calibration DCM which would be used to determine the phase of the input clock in the same way as in the existing solution. Even disregarding all other problems of this approach, the number of available global clock multiplexers (16) is far from the number that would be required.

Another possible solution would be the use of Dynamic Reconfiguration[1]. Since the clock multiplexers have two inputs, it would be possible to re-route any global clock to the currently unused input. This would limit the number of required global clocks buffers to one per ADC, giving 14 for the ADCs, plus one each for the 40 and 240 MHz system clocks, for a total of 16 – exactly the number which is available. Unfortunately, additional clock buffers are needed for the DCM feedback input. So again, this would exceed the available resources. In addition, using Dynamic Reconfiguration for switching clocks itself is not very well researched, and requires key information which is not publicly available.

As none of the previous options could be used, we had to come with a feasible solution to assure that the data inputs are provided with a known phase shift relative to the 240 MHz clock. We decided therefore to analyze the feasibility of using the over-sampling in order to achieve the alignment of the clock with the data, as described in the paragraph *II.B*.

[1] Dynamic Reconfiguration is a method for changing the programming of configurable architectures including FPGAs during runtime. A closer discussion of it is far beyond the scope of this paper.

B. Data to Clock Phase Alignment

Another Xilinx Application Note describes the approach to synchronize incoming data and clock signals with an internal clock [3]. This method bases on sampling the data on four phases of the same clock, each separated by 90 degrees, as shown in Fig. 5.

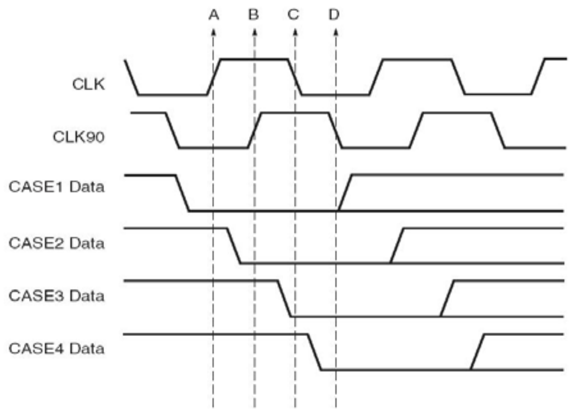

Figure 5. Timing diagram (XAPP225)

The four sample points are then clocked by 4 separate flip-flops (FFs) in 3 stages, as shown in Fig. 6. At the output of the FFs' columns, all four samples are synchronous with the same clock edge.

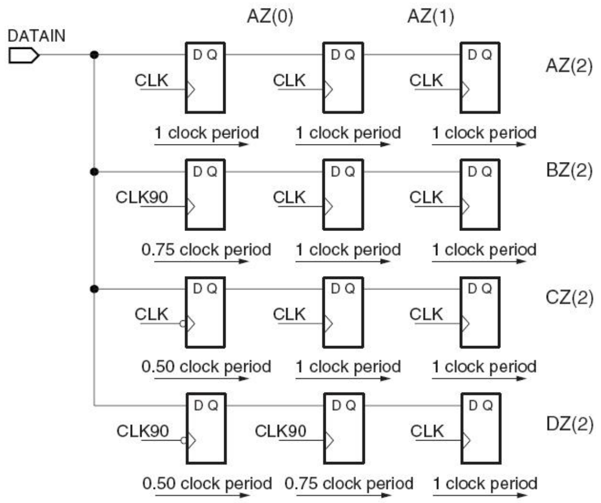

Figure 6. FFs input stage

A decision stage is then implemented to detect the clock phase which sees the first time a transition occurring in the data, to select the sample point and to forward it to the output of the over-sampling circuit.

The over-sampling design can be subject to instabilities, which are due to the fact that data is not fixed in time, moving slightly with temperature and voltage changes. It can happen due to instabilities that the circuit will synchronize with one time domain, but will resynchronize later to another. The over-sampling module however is designed to compensate for these issues. As the clock is oscillating and the data moves from one domain to another, a multiplexer is used to select the correct sample from one of the four available time domains. If a “wrap-around” occurs, i.e. the clock detection is oscillating such that the samples come alternatingly from fourth to first one (for example from clock domain D to A), the multiplexer automatically inserts one stage of delay prior to selecting the first (next) sample as valid.

The AppNote stipulates a speed limitation due to the maximum frequency that can be accepted by the FPGA’s Data Locked Loop (DLL) in a mode where it is capable of providing both a new clock and a clock shifted by 90 degrees. For XC2VP50 in speed grade -6, the maximum frequency is 210 MHz. Despite that, personal and public communication with Xilinx indicated that Xilinx Virtex-II Pro devices can actually achieve, with proper use of DCMs and careful design, sampling rates of up to 2 GS/s, respectively 500 Mb/s at 4x over-sampling. The method presented here is applicable for a single-data rate (SDR) signal, while in our case there are 112 DDR signals which have to be over-sampled. If the four points over-sampling would be applied for a DDR data running at 240 MHz, the final rate results at 1.92 GS/s, slightly below the 2 GS/s.

When analyzing the validity of the over-sampling method for the TRU design, we have to consider issues regarding available resources in the FPGA. It is mandatory that data and clock arrives at the over-sampling inputs flip-flops with minimum skew and delay. In addition, a tight timing control is required, inside the over-sampling circuit, from the input pins up to the first column of the input stage FFs. This is achieved by manually placing the first column of 4 FFs corresponding to a single pin, in the closest slices available next to the FPGA’s IOBs. For a DDR registration, there are 8 corresponding FFs which should be placed in the IOBs’ proximity.

Technologically, one slice can handle two inputs, as long as they are registered on the same clock edge, so that 8 slices would be enough for two DDR inputs. The relevant slices that can be use are maximum 8, located as shown in Fig. 7: 2 top right, 4 middle right and 2 bottom right (for an input located on the left side).

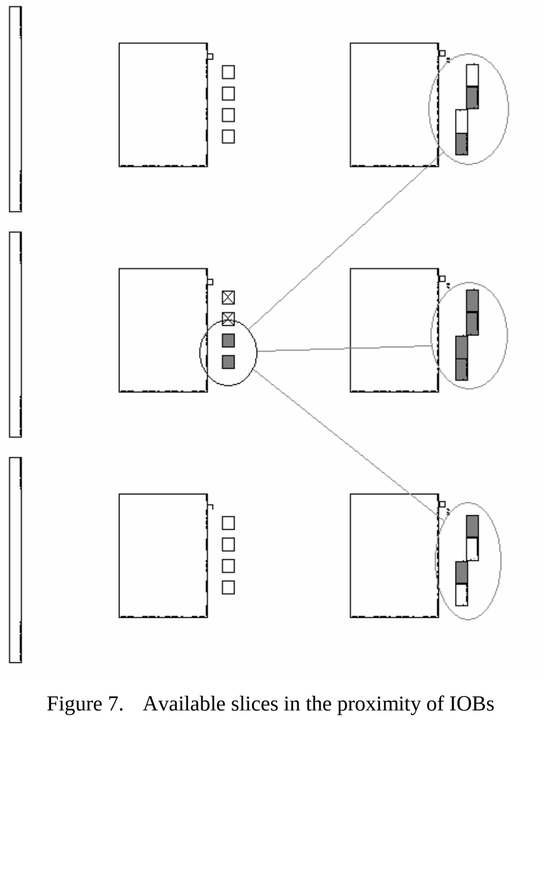
Figure 7. Available slices in the proximity of IOBs

However, due to the tight packing of the 112 LVDS lines which occupies a large number of IOBs in the XC2VP50, it can happen that more than two differential inputs are situated next to each other. In those areas, there would be simply more inputs than useful slices and sampling the data on four separate clock domains within the skew and delay requirements would be impossible.

C. Manual Place &Route

The flexible routing architecture of FPGAs provides a varied selection of different routes from any source to any destination. By carefully selecting the right route, one can achieve combinations of routing delays in such a way as to, for example, de-skew input buses or shift the phase of an input clock relative to the data inputs.

Any such solution is very inflexible itself, however re-adjusting one delay usually results in changed delays in other paths, basically requiring a completely new solution for at least part of the overall routing. And finding a routing solution in the first place is a Sisyphus job, as the available tools do not provide any support to automatically do this.

Therefore, while this is a possible approach for small problems, involving only a few pins and routes, it is not practical for our problem as the number of inputs and routes involved is too high to make human calibration practical, and any future change to the board would probably invalidate it.

D. Virtex-4

Even though the board design is already finished, this section would be incomplete without considering the possible benefits of a redesign using the new Virtex-4 devices. Considering the delivery deadline for the system, a preemptive redesign was not completely out of the question.

The main problem with using approach (*II.A*) is the number of available DCMs. From the Virtex-4 Family Overview [8] we can find that there is exactly one device that would have sufficient DCMs to accommodate all 14 ADC modules as per *II.A*: the V4FX140 offers 20 DCMs and also has enough IOBs. However, from a practical perspective, this is the top of the line, ‘dream’ device that might or might not get manufactured. It certainly will not be available any time soon, and not at a reasonable cost for the purpose of this project.

Beyond that, there are two new features specifically aimed for applications like ours – the IDELAY and ISERDES components. IDELAY is a configurable delay cell, which provides 64 steps of 78 ps delays for each individual input pin. This does not only simplify the PCB design, as it can be used to de-skew input buses, but it would also allow us to shift the inputs relative to the clock edge, either on a local (per-ADC) or global basis. The maximum achievable delay is 5 ns, or more than one full 240 MHz clock cycle.

If we could statically calibrate the delays, as one would do it for PCB de-skew, this would be the perfect solution. However lacking a priori knowledge of the behavior of the ADC PLLs (see *I.F*), a static calibration is not feasible, and some form of dynamic calibration would need to be done. The IDELAY cells do not offer any support for edge detection to facilitate this calibration, only the adjustable delay.

The second new and possibly useful component is the ISERDES cells. There is one for each input pin, or two for each differential pair. In master/slave mode, they can be used to provide 10-bit SDR or DDR capability, but we need 12 bits. They also provide some limited support for bit shifting for word alignment. Even without these additional features, it should be simple to build a 480 Mb/s deserializer using only one ISERDES cell in 6-bit mode[2] with two simple 6-bit registers (in user logic) attached to it.

This brings us back to the calibration problem: If the second ISERDES of each differential input pair remains unused for data input, it could be used to do edge detection on the input, which can then be used to dynamically adjust the IDELAY value. It is currently unknown whether this is feasible. Xilinx specifies the ISERDES to be useable as 1 Gb/s deserializer, based on 500 MHz f_{max} for the highest speed grade and DDR operation. This would leave us with only two samples per input bit, not enough for reliable edge detection.

III. OUR SOLUTION

A. Outline of the General Idea

In section *II* we presented existing concepts implemented by Xilinx which could be reused, we addressed a possible manual approach for place and route (P&R) on the FPGA in order to control any timing delays and finally, we discussed the advantages a Virtex-4 chip would have had if used on the TRU board instead the actual XC2VP50 FPGA. We have seen that the manual place and route cannot be successfully applied in our case, and that the use of a Virtex-4 would be an expensive, but feasible solution in the case of a respin of the board. For the moment however, we have to make the best use as possible of the existing TRU board design and its corresponding resources.

We concluded on not having enough DCM and slice resources available inside the FPGA for using the Xilinx solutions (presented in *II.A* and *II.B*). We highlighted however the possibility of reusing parts of those designs, for example the ADC interface, in the case when data and clock arrives phase aligned at the deserializer's inputs. In this order of ideas, we discussed to use over-sampling as an equivalent to employing DCMs to achieve the clock active phase alignment. Unfortunately, the over-sampling implemented by [3] required more than available slices into FPGA. We can adapt this method, by restricting it to 3 samples, each separated by 120 degrees, instead of 4 points sampled each at 90 degrees. Thus, fewer resources (only 3 slices) will be actually used on the FPGA compared to the previous over-sampling method. The over-sampling design implemented by us is presented in the section *III.B*.

The clock and data outputs from the over-sampling module are aligned, so that they can be deserialized, as described in section *III.C*. Once having all 112 data inputs (hit information) in parallel format, the trigger algorithm can be finally performed, as described in *IV.D*.

B. Over-sampling

The method we propose uses 3 samples instead of 4 for a SDR signal registration, respectively only 6 sample points (shifted by 60 degrees) instead of 8 for registering DDR data. Thus it occupies a total of 6 adjacent slices, one flip-flop used per input, or two slices less per pair compared to the previous method. In this case if 3 LVDS pairs of pins are situated next to each other, we still need an extra slice over the available 8, but we found that we could accommodate all inputs in the dense areas, as shown in *IV.C*.

The ADC lock clock cannot be used directly for over-sampling, as its phase relation to the incoming data is unknown, and therefore the over-sampling clocks are generated locally on the FPGA from the 240 MHz reference clock. A small disadvantage of the over-sampling on 6 different phases of the clock is that it requires extra 2 DCMs on the FPGA to generate the phase shifted 240 MHz clocks shown in Fig. 8.

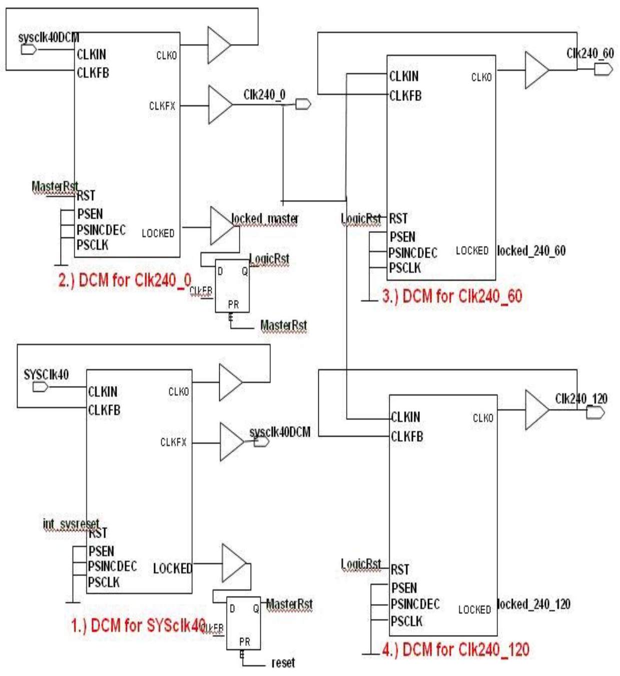

Figure 8. DCMs used for the clock generation

In addition to the DCMs needed to de-skew the 40 MHz clock in the FPGA (DCM1) and to generate the lock clock with a 0 degree phase (DCM2), a third DCM is used to get the lock clock shifted by 60 degrees and a fourth one is employed to get the phase shift of 120 degrees. The phases corresponding to 180, 240 and 300 degrees are obtained as negative edges of the 0, 60 and 120 degree clocks.

Using the locally generated 240 MHz clocks, all the incoming 112 data channels and theirs 40 MHz frame clock are over-sampled and thus synchronized to the 0 degree phase of the 240 MHz clock.

[2] 6-bit SDR would require the highest speed grade device, 6-bit DDR should be fine in any speed grade

C. Deserializer

As mentioned in our conclusion to *II.A*, if we could provide the necessary clock and data inputs, the deserializer module from [1] could be reused.

The over-sampling solution from the previous section (*III.B*) solved that problem and once having data information 90 degrees phase aligned with the lock clock, the 112-fold deserializer was implemented in XC2VP50 FPGA. We basically reused most of the existing solution presented in *III.A* after replacing the first flip-flop of the input shift register with the output from the over-sampling, as well as replacing the timing and control module with new logic. From the standard solution we did not use any set of registers, such as FIFO or memory, to output the data from the deserializer. A 12-bits parallel register is placed instead to synchronize the data to the 40 MHz LHC reference clock, on which is running the whole trigger algorithm from now on.

IV. IMPLEMENTATION ASPECTS

A. Outline of the hierarchy

As highlighted throughout the paper, there are in total 112 LVDS lines from the 14 ADCs on the TRU board which arrive with an individual data rate of 480 Mb/s to a single XC2VP50 FPGA. Our goal was to provide a practical interface solution for the deserialization of all those data which inputs the FPGA with 53.76 Gb/s bandwidth.

The interface has been written in Verilog and has been built as a hierarchy of 14 instances of a single ADC interface, each of these instances including an over-sampling and a deserializer module. The over-sampling module contains the generation of a local 240 MHz clock, whose 6 different phases are used to align the frame clock and the 8 LVDS data channels received by the FPGA from one ADS5270 device. The deserializer includes 4 identical receiver modules, which handle 2 channels each, and a timing module, which based on the edge detection of the aligned frame clock, generates the enabling and the validation signals for the deserializer logic. The 112 outputs of the ADC interfaces are updated each 25 ns, synchronously to 40 MHz LHC clock and they are further used in computing and generating of the TRU trigger, as explained in *IV.D.*

B. Single ADC interface implementation

The interface to a single ADC has been implemented[3] using Xilinx ISE (Integrated Software Environment) tools. In addition to the Verilog description of the ADC interface, there are some constraints or attributes which should be defined for the design and which are used by the tools during the implementation to achieve a better placement of the logic in the FPGA and a faster (tighter) trace routing between components.

The simplest constraints that must be applied are related to information about the frequency and the phase of the clocks involved in the design. There are however some other practical constraints which need to be considered in order to achieve a high-frequency for the running design on the FPGA. For the design on the TRU board, it is important to have a tight timing control i.e. equal minimum delay and minimum skew from input pads to the first column of 6 FFs in the over-sampling module. A minimum delay with equal propagation times is obtained by 'forcing' the 6 FFs placement in the closest slices available next to the inputs (as shown in Fig. 7). Regarding the skew values, the maximum allowed for a signal sampled to 1.44 GS/s should not be more than 350 ps, which is half of the corresponding period. We tried however to use more restrictive values than that and found out that a single interface design can be achieved within 250ps skew.

Among the 14 ADCs interconnected to the XC2VP50, some have the inputs in the FPGA located in the left, others in the right or in the bottom side of the chip. We implemented the single interface design for all of the individual ADCs instances and verified for all these cases, by checking the post P&R reports generated by the tool if the implementation was achieved, the design was working fine and the constraints have been all met.

C. Full system integration – 14 ADCs

After the successful implementation of the single ADC interface module was finished, full system integration was put forth. As to be expected, problems soon started showing up. Our carefully laid out constraints, set to ensure proper timing on all input pins, failed to be met when the interface to all 14 ADCs was implemented.

The reason for this was clearly to be searched within the tools, or rather the nature of the commonly used P&R algorithms. Since the exact solution of a given P&R problem is non-polynomial (NP)-strong, the regular tools use heuristical approaches, so for all but simple problems they can not be expected to find the optimum solution. This was made worse by the fact that the inputs from different ADCs were not clearly separated, but rather sometimes interleaved.

However, after looking through all the failed constraints, due to the tight placement of the IOBs, and knowing our previous good results for each individual ADC, we were certain that he timing was achievable. Therefore we first decided to manually place all the first stage input flip-flops for the critical regions, and later just locked them all to prevent any further mishaps.

The actual placement problem is made difficult by the fact that we have six clock phases, and only eight possible placement options (slices) for the first stage FFs for each input. With several inputs next to each other, both FFs of each relevant slice must be used, and an arrangement must be found such that all inputs can have their 6 FFs placed within their corresponding 8 locations, sharing space with the others.

We worked out an extendable pattern that would provide such a placement for all our areas. The solution we proposed is illustrated in Fig.9 for some of the IOBs situated in the left side of the FPGA. Rotated appropriately, the pattern illustrated in Fig. 9 can also be applied to the top, bottom and right side IOB banks.

[3] Design Implementation is the process of translating, mapping, placing, routing, and generating a BIT file for the design

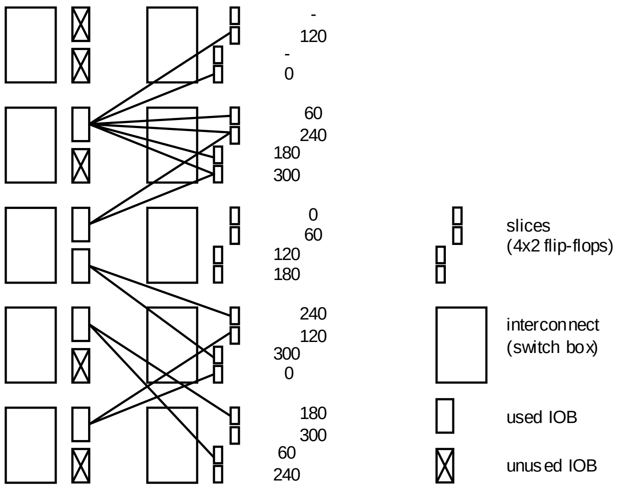

Figure 9. Placement pattern for the first stage input flip-flop and their corresponding clock phases

Each slice input uses the four slices available to the right next to it, as shown for the top most one. In addition, it needs two slice flip-flops above or below its own row – those are show for each IOB. The numbers indicate the clock (phase) used in each slice.

As already mentioned, we first placed all areas that failed the constraints. After running through a few iterations of locking down FFs and running the tools to place and route the rest of the design, resulting in new failures in other areas, we decided to fix the placement of all the first stage FFs. After we had done this, the tools successfully and reproducibly were able to place the remaining parts of the design, as well as route the inputs meeting the constraints, thus saving us the Sisyphus work mentioned in *II.C*, which would have been the last..

Once having successfully implemented the interface to all 14 ADCs, the primary problem of inputting the data into the FPGA was solved, and work could continue on the actual application which was the PHOS level-0 and level-1 trigger generation.

D. Trigger logic

A trigger decision processed by the TRU board for a TRU region of 16x28 crystals requires that the hit information to be summed up across 4x4 crystal windows and over 4 consecutive ADC samples [7]. The algorithm is implemented in parallel logic running at 40 MHz LHC machine rate and there are in total 91 simultaneously summing combinations corresponding to a 4x4 kernel size, which results every 25 ns. All 91 sum values are compared against individually programmable thresholds and the comparison's results are OR'ed such as that a single positive comparison counts as a trigger. All trigger information is output synchronous to the LHC clock: the level-0 trigger is output as a single non-return-to-zero (NRZ) signal, whilst the level-1 trigger has 3 NRZ outputs of different thresholds.

The trigger algorithm is implemented across 3 Verilog modules. First module performs the summing across a space of 4x4 ADCs input data. The second module pipelines the 4 consecutive previous sums from each of the 91 instances, sums them over time and compares the results against programmable thresholds. If any of those thresholds is exceeded, it means a trigger has been detected and finally, the third module outputs this synchronous to the LHC 40 MHz clock.

The entire trigger system is subject to external timing constraints. For example a level-0 trigger must be output from the FPGA with exactly 600 ns delay from the initial moment of the particle interaction. By subtracting the light time of flight, the conversion time in the APDs, the summing in the FEE cards (see *I.B*) and the digitization in the ADC on the TRU (see *I.D*), it remains not more than 300 ns for processing the level-0 trigger inside the XC2VP50.

V. PERFORMANCE

Without a TRU board available, we had to rely on the post P&R timing report generated by Xilinx ISE software for the design performance. In addition, we performed post P&R simulations in Cadence NClaunch to check if the implementation behaves as required.

After the initial good results meeting our timing constraints for all 14 ADC input modules, and the solution for the placement problems encountered during their integration into the full system, the final design again meets the required timing for the 1.44 GS/s over-sampling and the 480 Mb/s deserializers. Specifically, all skews matched the 250 ps target we had set, and all delays were roughly equal.

On the other side, we verified the correct functionality of the trigger algorithm with simulations and tested if the level-0 trigger can be implemented with 300ns delay in the FPGA. We first simulated the over-sampling module alone, to see if it works with different phase clock alignments and then, we added the deserializer logic to check if the enabling signals generated by its timing module are computed to output at the correct moment the parallel data.

Once the desired functionality was proven, timing information about the design performance have been obtained from the post P&R simulations. The over-sampling circuit was taking 3 cycles of the 240 MHz clock, which corresponds to 12.5 ns. The deserializer logic outputs parallel data, sync to the local 240 MHz clock, every 25 ns. To sync this data with the 40 MHz clock on which runs the trigger algorithm, a parallel register is used; this introduces between minimum 1 and maximum 6 cycles of delay, depending on the original alignment between the 240 MHz and the LHC clock. In total, the over-sampling approach together with the deserialization consumes in the XC2VP50 FPGA between 10 and 15 cycles of the 240 MHz clock (41.67 ns to 62.5 ns). So, there is still enough time left to process the trigger level-0 algorithm, which is the main timing constraint for PHOS. The timing sequence obtained for the level-0 trigger is shown below in Fig.10.

Following the deserialization of the 112 lines, the 4x4 sliding window algorithm is computed in one 40 MHz clock cycle. Pipelined 40 MHz sample sums are achieved in 5 clock cycles and between 3 and 6 extra clock cycles were used to detect a trigger and to output it. The computation of trigger level-0 was done in less than 300 ns, so that in order to output the level-0 trigger at exactly 600 ns from the initial moment when particles hit the crystals, we still needed to introduce some extra delays.

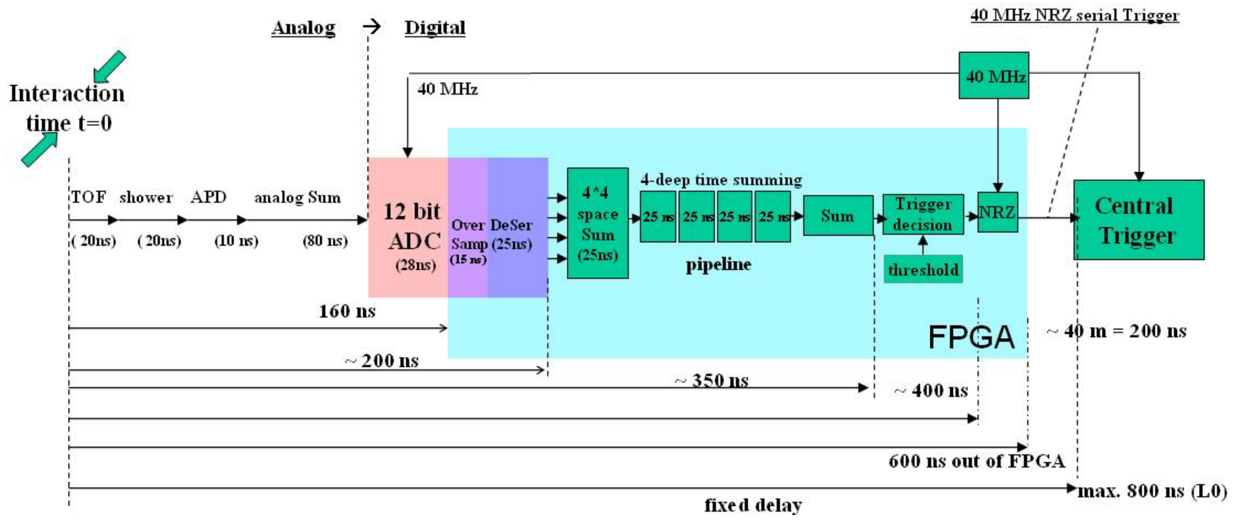

Figure 10. Timing sequence for the level-0 trigger

VI. CONCLUSIONS

In this paper we presented a practical solution for the interface of a continuous input bandwidth of 53.76Gb/s to a Xilinx Virtex-II Pro FPGA, realized on the TRU board on the TRU board used by PHOS electronics at CERN. In addition to presenting this solution, we reported practical implementation problems which appeared during the development. We continued with showing the manual placement approach undertaken for accommodating the interfaces to 14 ADCs in a single FPGA and we highlighted the implementation performances.

We also outlined in this paper the general context of the trigger algorithm (focus being put on level-0 trigger) within the PHOS, as this is the main purpose for which the TRU board is used and the main reason of deserializing the data inputs to the FPGA. Post P&R simulations have been performed for the implemented interfaces, but for the whole trigger level-0 design and they have shown that the required overall timing performance can be well achieved with the given resources.

VII. OUTLOOK

With the TRU board coming some time soon, we have to verify the implemented solution under real operating conditions, identify possible problems not shown by the tools and be ready to improve the existent development to make it work in practice.

ACKNOWLEDGMENT

The TRU project is financed by the Norwegian Research Council and INTAS project 03-5747. The TRU board design was designed at CERN by Hans Müller and Rui Pimenta and the layout of the board has been achieved at HUST Wuhan China.

REFERENCES

[1] M. Defossez, “Connecting Xilinx FPGAs to Texas Instruments ADS527x Series ADCs”, *Xilinx Application Note 774*, November 2004; see http://direct.xilinx.com/bvdocs/appnotes/xapp774.pdf.

[2] N. Sawyer, “Active Phase Alignment”, *Xilinx Application Note 268*, December 2002; see http://direct.xilinx.com/bvdocs/appnotes/xapp268.pdf.

[3] N. Sawyer, “Data to Clock Phase Alignment”, *Xilinx Application Note 225*, April 2002; see http://direct.xilinx.com/bvdocs/appnotes/xapp225.pdf.

[4] “ALICE - A Large Ion Collider Experiment at CERN LHC”, see http://aliceinfo.cern.ch.

[5] “8-Channel, 12-Bit, 40MSPS Analog-to-Digital Converter with Serial LVDS Interface”, Texas Instruments ADS5270 Data Sheet and Documentation, Rev. E, September 2005; see http://focus.ti.com/docs/prod/folders/print/ads5270.html.

[6] H. Müller et al., “Front End Electronics for the PHOS electromagnetic calorimeter”, *Alice Note*, 2004-2005; Alice 2005-XXX, Rev. 6a.

[7] H. Müller, A. Oltean et al., “Trigger Region Unit for the ALICE PHOS calorimeter”, *11th Workshop on electronics for LHC and future experiments*, Heidelberg, , pp. 384-387, September 12-16 2005.

[8] "Virtex-4 Family Overview", Xilinx Data Sheet 112, February 2006; see http://www.xilinx.com/bvdocs/publications/ds112.pdf