ACQUISITION & CONTROL SYSTEM PROJECT FOR COSMIC RAY DETECTOR

Francesco Russo (INAF/IASF-Palermo) - Technical staff (francesco.russo@inaf.it)

Rev. Document 1.9 (FIRMWARE description) — 13/11/2025

Summary

INTRODUCTION 3
SYSTEM DESCRIPTION 4
COUNTERS MODULE 5
SAMPLER MODULE 7
PEAK READER MODULE 9
TRIGGER MANAGER MODULE 11
I/O COMMUNICATION MODULE 12
I/O INTERFACE MODULE 14
SERVICES MODULES 15
CONCLUSIONS 16

Introduction

This document describes the design of an acquisition and control system for cosmic ray detection instruments. Specifically, this document describes, from a firmware perspective, the various IP (Intellectual Property) Modules required to acquire rapid events with a duration of the order of nanoseconds, such as those generated by Cherenkov radiation produced by electromagnetic showers, and simultaneously acquire events with a longer duration, such as those generated by atmospheric fluorescence produced by hadronic showers. This document also describes the IP-Modules dedicated to reading analog peaks, managing triggers, reading housekeeping data, generating test pulses, managing service forms, and finally managing communication with a host computer.

The firmware IP-Modules described in this document were entirely developed, tested, and used by the technical staff of the Electronics Laboratory of the IASF-PA Institute of INAF. The functional description of the modules is provided in VHDL code, while the connection of all the modules was created in schematics, thus obtaining a detailed and easily usable graphical diagram of the entire project. The FPGA devices used are from the CYCLONE-V family by ALTERA (Intel), as is the Quartus development environment.

This project's development utilized extensive RTL structures and multiple dedicated state machines operating simultaneously in parallel. This allowed each IP-Module to be implemented and tested in isolation, allowing for performance-optimized development and the potential reuse of individual IP-Modules for projects other than this one. Furthermore, we avoided the use of virtual processors such as NIOS or embedded SoC physical processors, as the former would require a significant amount of resources for parallel use as required by the project, while the latter are present as a single unit within an FPGA chip, making them unusable in this context. In any case, using these processors in the project would compromise its portability.

A key role in achieving the described characteristics and performance is played by the complex and comprehensive definition, at the project level, of a whole series of constraints and assignments, which instruct the compiler to achieve the best routing and fitting for the entire project. Without these definitions, it is impossible to successfully compile such a dense project and obtain a system that functions at the required clock speeds.

With the above-mentioned FPGA devices, it was possible to achieve the following characteristics and performances for a total of 64 acquisition channels:

- 32-bit cosmic ray photon-counting (+ detector dark) up to 250 MHz for each acquisition channel
-
- Fast events lasting less than 10 ns:
- Direct and simultaneous sampling of all acquisition channel triggers at 1 GSMP/s, on ring-type free-run memory
- Pixel majority trigger algorithm, pipeline-type, at 1 GSMP/s, with low latency, on all acquisition channels
- Algorithm for generating read-out hits for multiple peak detectors
- Reading of peak detector analog values via a dedicated programmable controller
- Continuous saving of events (a portion of samples + the peak detector analog values) on a double-buffered memory stack
-
- Events lasting longer than 10 ns:
- Sampling of each acquisition channel counts at 6-8 bits up to 20 GSMP/s, on ring-type free-run memory
- Maximum threshold trigger algorithm with moving average, based on the sum of the acquisition channels
- Continuous saving of events (a portion of counting samples) on a double-buffered memory stack
-
- Acquisition management from triggers from the counting and sampling modules, and external input, via a dedicated controller
- Generation of a unique 64-bit timestamp for event marking, 32 bits for seconds and 32 bits for tenths of a microsecond
- Generation of a unique 32-bit counter ID for sequential event marking
- Management of PPS and 10MHz signal synchronization for timestamp generation
- Typical average trigger rate sustainable up to approximately 100KHz or higher (depending on the number of triggered channels and the number of samples to be read)
- Programmable coincidence window and reading of trigger rate meters
-
- Management of external service devices such as temperature sensors, voltage and current sensing, ASIC configuration, and High voltage modules, etc.
- Interfacing with these external devices via SPI, I2C, and UART buses
- Generation of test pulses programmable in width, period, amplitude, and finite or continuous number of pulses
-
- Management of the communication protocol with a host computer via a dedicated controller
- Interpretation of commands, responses, and payloads, flow control, and timeouts
- Sustainable average I/O throughput of up to 100 MB/s
-
- Interfacing with a host computer via parallel interfaces such as FT245-SYNC and ASYNC, or serial SPI HS, UART at 10 Mbps or higher

System description

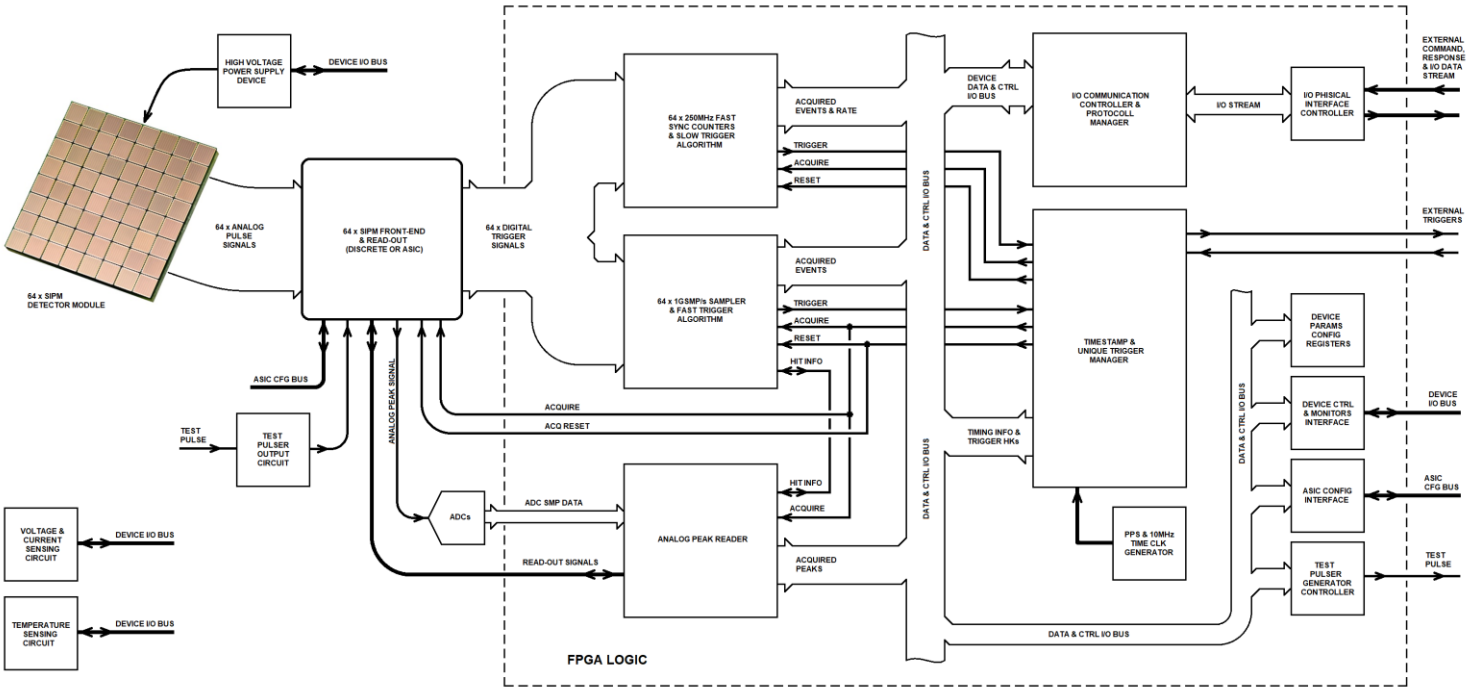

Fig 1: Typical cosmic ray detector acquisition and control system

A cosmic ray radiation detection instrument is primarily composed of a detector, interfaced with dedicated front-end electronic circuitry, typically an ASIC, which is in turn connected to back-end electronics for data acquisition, typically an FPGA. The FPGA is also used to interface with a host computer, allowing the instrument to be controlled and the acquired scientific data to be read. Finally, there is secondary service circuitry that allows for interfacing with temperature, voltage and current sensing sensors, HV modules, test pulses, etc. This service circuitry is also served by the FPGA.

It's clear that the FPGA plays a fundamental role in the design of a scientific instrument like a cosmic ray detector. Below, we describe the IP-Modules developed for this project and how these modules are interconnected within the FPGA to create a complete control and acquisition firmware.

As shown in Fig. 1, all the trigger signals output from the front-end circuitry of each acquisition channel are connected to the corresponding FPGA inputs. Internally, these digital signals simultaneously feed into two distinct IP-Modules: the "Counters" module and the "Sampler" module, which are the heart of the acquisition system.

The "Peak-Reader" IP-Module performs peak reading of analog signals, if this functionality is present in the front-end circuitry. This module manages an ADC and the control signals for selecting the channel to be read.

The Counters, Sampler, and Reader modules are interconnected to the "Trigger" IP-Module, which manages their acquisitions and event accumulation based on the formation of the relevant acquisition triggers. This module also creates a 64-bit timestamp, with a resolution of 100 ns, and a unique 32-bit sequential ID for timestamping events.

The project also includes a series of service IP-Modules, necessary for managing the secondary service circuitry described above.

All the modules described are interconnected to the "Communication Manager" IP-Module, which manages both the configuration of all the various modules and reads all the acquired scientific data. This module, in turn, communicates with a host computer via a proprietary protocol developed specifically for this purpose.

The various IP-Modules present in the project are described in detail below.

Counters module

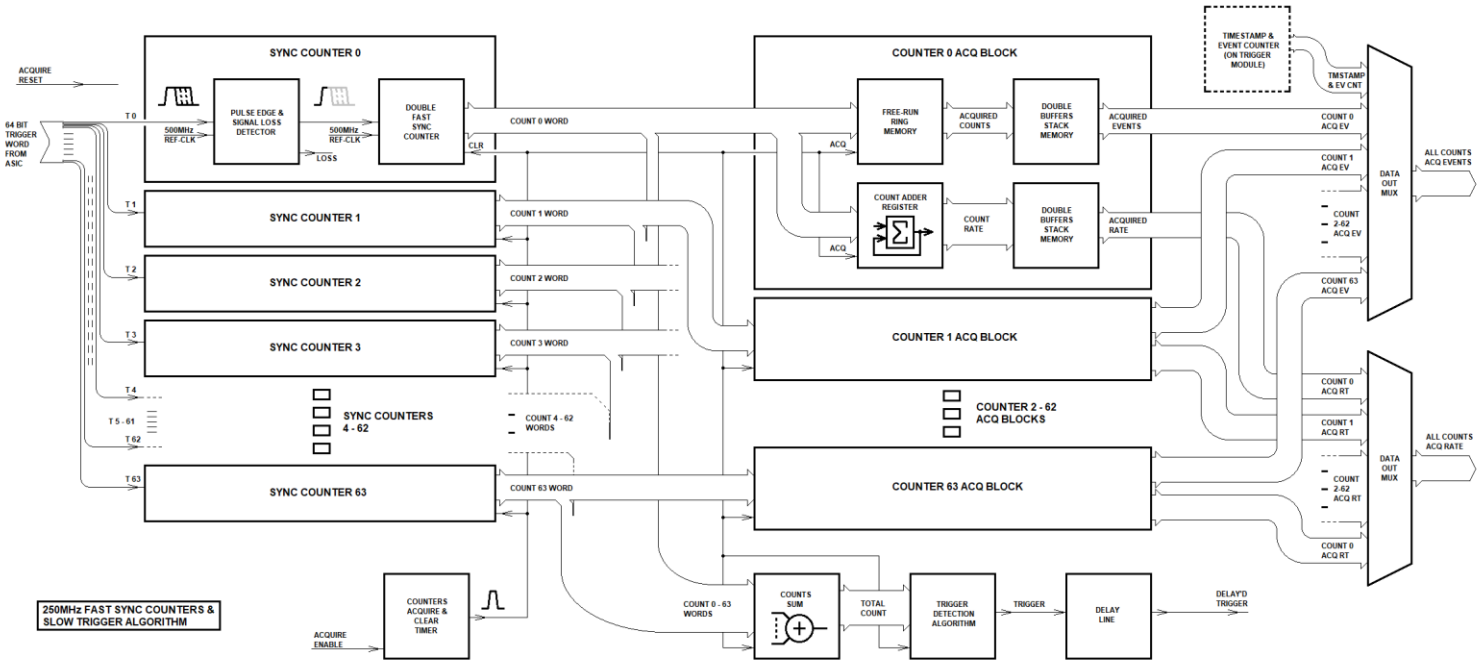

Fig 2: IP-Module with trigger algorithm for events >10ns and fast synchronous counters

The "Counters" IP-Module is designed to measure cosmic ray radiation and simultaneously acquire events with a duration of tens of nanoseconds or longer. These two functions are implemented for all 64 digital input signals from the front-end trigger outputs.

As shown in Fig. 2, each trigger pulse produced by the Front End enters a pulse shaper capable of detecting its leading edge. The pulse thus formed is counted by a synchronous counter whose value is sent to a stage consisting of two independent chains. One accumulates the count in a dedicated register that will be used to measure the cosmic background, while the other records each count in a free-run ring memory.

The rate at which count values are accumulated, recorded and finally reset is set by a programmable timer, controlled by the acquisition enable signal from the Trigger IP-Module.

In addition to being sent to the stage described above, the counter value contributes, together with the values of the counters of the other 63 acquisition channels, to the formation of the hadronic event acquisition trigger. The trigger algorithm is produced by the sum of the 64 count values, continuously processed through a programmable moving average, the result of which is finally compared with a threshold value. The comparator output will serve as input to the Trigger module. The latter will return the acquisition stop signal, which will stop the ring memory and start the process of transferring the event, that is a preset portion of the counts recorded in it, along with a time stamp, to a double-buffered memory stack, to then resume acquisition activity until the next trigger, and so on.

The double-buffered memory stack is appropriately sized to accommodate the accumulation of multiple events. The double buffer has the dual advantage of enabling event acquisition without introducing any dead time, and also allowing the host computer to unload these buffers in a concentrated manner, thus keeping the communication bus as free as possible, to the benefit of other I/O processes.

Notes:

Implementing as many as 64 32-bit counters within an FPGA is no simple task. First of all, the use of asynchronous counters must be ruled out. Asynchronous counters are sequential devices that have no relationship to a reference clock. Introducing the pulse to be counted directly into the clock input of a counter is inadvisable, if not completely incorrect. A 32-bit counter is composed of 32 flip-flops plus combinatorial logic to achieve the desired increment. There is no guarantee that the pulse to be counted will be distributed evenly within the FPGA with zero spread across all 32 FF clocks of a counter. This is even more true for 64 counters and especially when an FPGA is already dense with other logic circuitry to serve. The correct distribution of a clock signal within an FPGA must occur exclusively through the dedicated lines that each FPGA device provides for this purpose. But these lines are a small and very limited number.

Then there's the problem of reading and subsequently resetting the counter's value. This process always relies on a synchronous operation performed by a dedicated controller that transfers the count value to memory. There's no guarantee that this operation won't read an incorrect value from the counter, due to its asynchronous increment.

In summary, this development procedure is impractical. One must necessarily consider a synchronous counter, in which the counter's clock input is always subject to a reference clock, and the pulse to be counted, appropriately processed, acts only as an enable for the increment.

That said, it must be considered that the minimum trigger pulse width produced by some ASICs is around 2ns, and that the cosmic background (plus the detector dark) can reach a frequency of several tens of MHz for each channel. To meet all these requirements, a dual synchronous Fast Counter operating at a 500MHz clock was developed for all 64 channels of the Front End. To be able to operate at these frequencies, each of the two counters was designed with a maximum of 6-8 bits, and was as simple as possible from a circuit point of view, therefore with asynchronous reset and without recirculation control. The need for the dual counter is due precisely to the presence of the asynchronous reset, which prevents the counter from being reset within a clock cycle. The two counters alternate in counting consecutively, and while one is ready to count, the other is in a stop state with its value available for transfer to the next stage of the count acquisition and accumulation chain, to then be immediately reset with the asynchronous reset. All this within the acquisition timer period, which is calculated appropriately to avoid counter recirculation.

The Fast-Counter was designed to be compiled in two modes: as a standard counter or as a shift register plus encoder. This option was implemented to allow for greater flexibility during compilation, and therefore a greater probability of successfully completing the project. Many factors contribute to this success, such as the FPGA family used, the project density, the clock speed, etc.

As previously mentioned, it was necessary to implement several technical measures, as well as define various constraints and assignments, to ensure that 64 such counting chains function correctly. A complete description of these technical notes is reserved for another document.

Sampler module

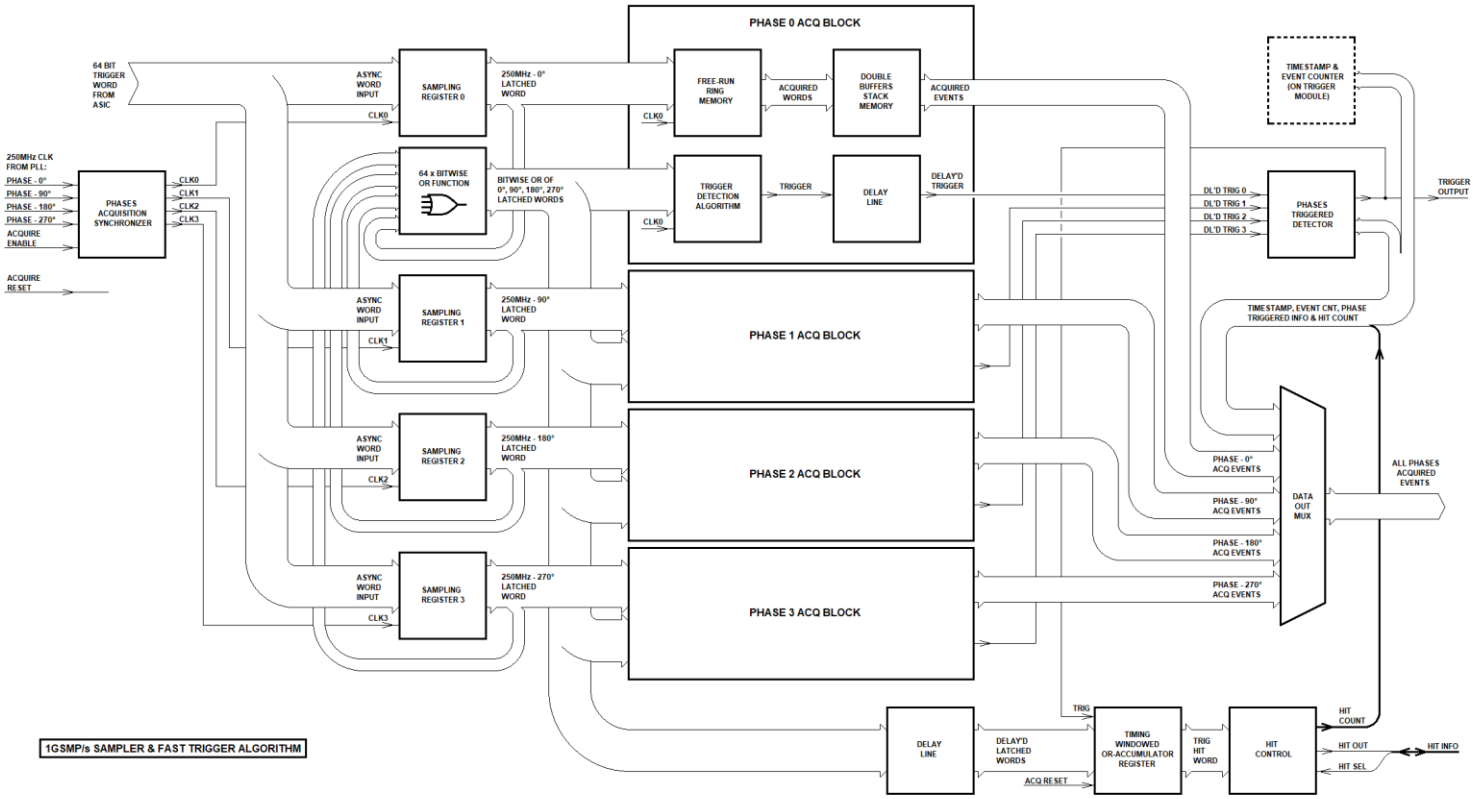

Fig 3: IP-Module with 1 GSMP/s sampling acquisition and fast trigger algorithm

The “Sampler” IP-Module takes care of the acquisition of rapid events characterized by a time duration in the order of nanoseconds.

The name of this module itself describes its behavior. In practice, this module simultaneously, continuously, and at high speed samples all 64 digital input signals from the front-end's trigger outputs. Specifically, in this project, sampling occurs at a rate of 1 GSMP/s.

The samples obtained from the sampling process are recorded to a free-run ring memory and simultaneously sent to the acquisition trigger logic. The trigger algorithm is the arithmetic sum of the logical "1s" present in each individual sample, this value is then compared with a majority threshold. The comparator output will serve as the trigger output to the Trigger module. The latter will return the acquisition stop signal, which will stop the ring memory and start the process of transferring the event, that is a preset portion of the samples recorded in it along with a timestamp, to a double-buffered memory stack. Acquisition activity will then resume until the next trigger, and so on.

As with the "Counter" module, the double-buffered memory stack is appropriately sized to allow for the accumulation of multiple events. The presence of the double buffer has the dual advantage of allowing event acquisition without introducing any dead time, and also of allowing the host computer to unload these buffers in a concentrated manner, thus occupying the communication bus as little as possible, to the benefit of other I/O processes.

Although the task of this IP-Module is simple in its function, its implementation is not so simple.

To achieve a sampling frequency of 1GHz on a low-end FPGA (but not only for this…), it is necessary to divide the sampling into multiple parts operating at lower frequencies, and then merge the samples of the individual parts together to obtain the desired initial result.

As shown in Fig. 3, the 64-bit word formed by the triggers is sampled by four identical input registers. Each register samples the same signal at a frequency of 250MHz, but with a 90° clock phase shift from each other. In practice, the four samples obtained are 1ns apart, so it is as if it were a single register operating at a frequency of 1GHz. Subsequently, the samples from each register are directed to four recording stages, each operating at 250MHz and with an incremental 90° clock shift.

To achieve the trigger function described above, the samples obtained from the four input registers are computed bitwise with a logical OR function. This operation is the functional equivalent of a 7ns time integration window, due to the overlap of four 4ns windows shifted by 1ns, which scrolls with a resolution of 1ns. Simply put, all the

logical "1s" of the words of four consecutive samples contribute to the formation of the acquisition trigger. This avoids the potential loss of triggers when an event is spread across multiple consecutive samples.

To operate at these frequencies, also the trigger logic is also quadrupled and made to operate at a quarter of the nominal frequency, appropriately phase-shifted. Furthermore, the sum of the logical "1s" is obtained through multiple sequential adding stages in a pipelined configuration. In this way, it is possible to obtain a 1ns acquisition trigger computation for each sample, but with an initial latency of a few tens of ns due to the depth of the pipe. This latency is then compensated by appropriately calculating the offset for transferring the samples from the ring memory to the stack.

In summary, 1GSMP/s sampling was achieved using four identical sampling stages operating at a quarter of the nominal frequency, appropriately phase-shifted. To achieve this goal, it is essential that these four stages be synchronized in acquiring samples with respect to the ACQUIRE signal that establishes the start and stop of acquisition. This essential and basic function was achieved using a phase synchronizer circuit specifically designed and developed for this purpose. In this synchronizer, the number of stages to be synchronized can be parameterized during the design phase.

Another essential circuit is the trigger phase detector. Given the nature of the acquisition trigger algorithm, which integrates the samples in bitwise-OR between the four sampling phases at the input of the four trigger stages, all four trigger stages will inevitably produce a trigger regardless of which sample of the four phases contains the information needed to exceed the majority threshold. A circuit is therefore needed that allows us to identify which of the four trigger stages exceeded the threshold first, thus providing the information that allows us, in hindsight, to exactly associate the trigger with the corresponding sample, included in the samples of the acquired event, with ns precision.

Finally, this Sampler IP-Module contains the circuitry needed to generate the Hit Info required for the "Peak Reader" IP module to identify the Front End channels from which to read the analog peak values, using a suitable ADC. This circuitry can operate in various modes:

1. Only the samples (bitwise-OR) of the event that produce the first trigger are considered
2. The samples from the first event plus all samples from subsequent events are considered
3. The samples of the first event plus all subsequent samples (regardless of whether they produce a trigger) are considered

Additionally, it is possible to enable a programmable time window during which to perform the OR integration of the words.

This circuit provides a solution to the lack of this functionality in some Front-Ends, or even replaces the same circuit where it is already present, as this digital version is much more precise and advanced in its function.

Notes:

An important aspect of the trigger chain is that it is affected by pipeline-related latency. To build a fast computation algorithm capable of evaluating whether the majority is exceeded for each 1ns sample, the use of a pipeline and the resulting latency are unavoidable. However, it is essential that this latency is always precise and constant. The latency, however short, is compensated for by adding a read offset during the transfer phase, but only if this offset never changes regardless of the shape of the sample being evaluated. The trigger logic of this IP-Module has been designed precisely to always have a predetermined latency.

One of the reasons limiting the sampling rate is the maximum write speed of the ring free-run memory. In this project, it was decided to limit the sampling rate to 1GSMP/s as it is adequate for obtaining the required scientific results. At this rate, each of the four stages operates at a frequency of 250MHz. This frequency is perfectly sustainable by the embedded memory present in an FPGA of the CYCLONE V family (C-6). If we wanted to extrapolate this sampling philosophy to the maximum, exploiting the maximum resources available in a CYCLONE V, namely the maximum write frequency of 315MHz, and eight 315MHz clocks shifted by 45°, a sampling rate of over 2.5GSMP/s could be achieved. If we calculate the same extrapolation for a much more performing and expensive high-end FPGA such as a STRATIX10 also from Altera, with a max write speed of 600MHz, and 16 clocks shifted by 22.5°, we would obtain a remarkable sampling speed of almost 10GSMP/s.

For this IP-Module, too, several technical measures had to be implemented, as well as various constraints and assignments defined, to ensure that the 1GSMP/s sampling rate functioned correctly as described. A complete description of these technical notes is provided in another document.

Peak Reader module

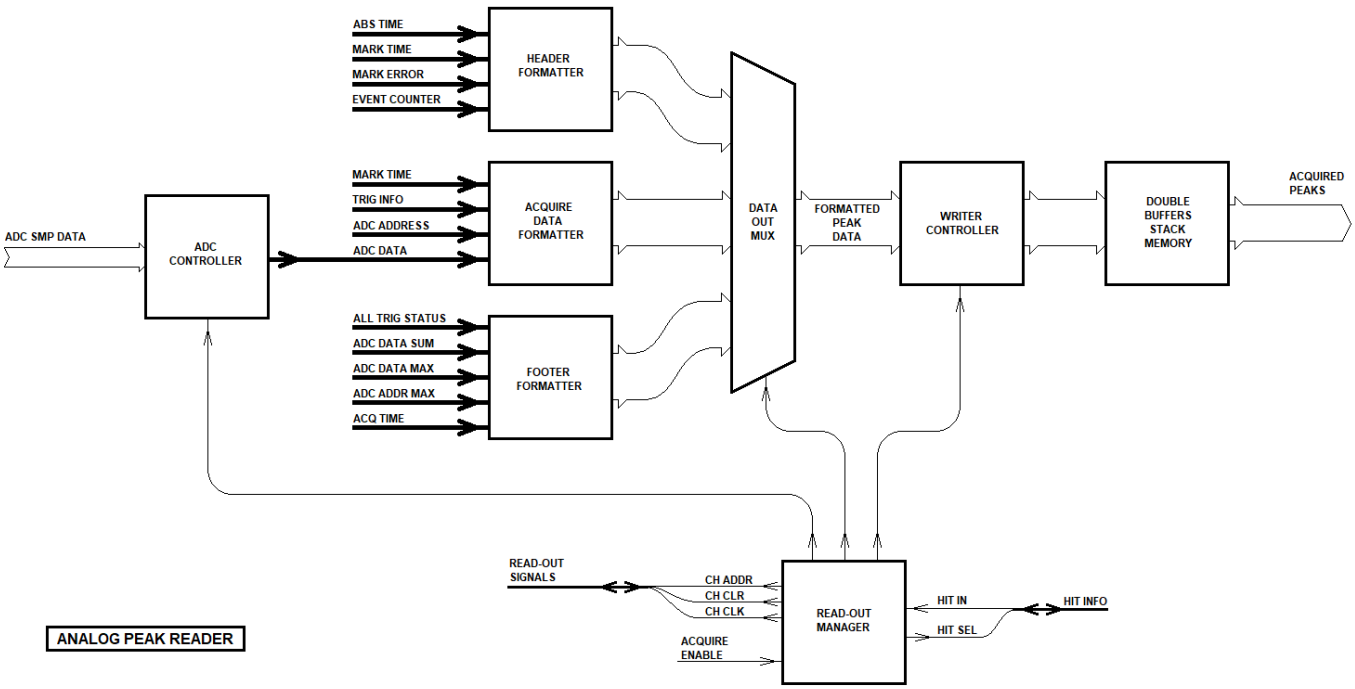

Fig 4: Analog peak reader IP-Module

The "Peak-Reader" IP-Module reads the analog values of the Peak-Detectors present in the Front-End electronic circuitry. This operation is composed of these main sequential phases:

1. Select the Peak Detector channel to be read via a mux
2. Wait for the mux output signal to settle
3. Start signal conversion via an ADC
4. Wait for the ADC to complete conversion
5. Transfer the converted value to memory
6. Return to step 1 to move to the next channel

From this list, it is clear that phases 2, 4, and 5 are time-consuming. This time depends on the electrical characteristics of the mux, the performance of the ADC used, and the write speed of the transfer memory, plus the amount of data to be transferred. Since all phases are necessarily sequential, it follows that the total time required to process all the Peak Detectors' channels can be considerable and become a critical point in the attempt to achieve the shortest possible acquisition dead time.

To overcome this problem, a pipelined read controller was designed that can interleave the "slow" phases of the various channels to be read, thus making the processing time uniform for all three phases. In practice, while waiting for the mux of channel n to settle, the ADC of channel n-1 is simultaneously awaited, and the converted value of channel n-2 is simultaneously transferred to the memory. And so on.

Furthermore, to minimize dead time, this controller can decide whether to proceed with reading the selected channel or move on to the next one, based on the status of the Hit signal coming from the dedicated circuitry of the IP Sampler module, or if available from the Front-End electronics itself. The Hit status is nothing more than the signal, on the selected channel, that the trigger has been exceeded, due to the input signal exceeding a certain threshold condition. This feature can, however, be bypassed if you want to read all channels regardless of the status described above.

With this controller it is also possible to perform a quick pre-scan of the channels to be read, monitoring only the Hits, and subsequently perform a direct addressing of the channels to be read previously memorized via the pre-scan.

Another feature is the ability to read a possible complementary channel together with the channel in the hit state. This is useful when multiple channels are paired with a detector array, but only one of the two exceeds the trigger threshold that signals the hit state.

Other features are available in this controller, as summarized below:

- 1 to 255 possible read channels
- Interleaved sequential reading via Hit-info
- Preliminary scanning of the channels to be read via Hit-info, and subsequent direct addressing
- Writing to memory with initial header
- Writing to memory with final footer
- Reading of the complementary channel (if present)
- Reading of all channels regardless of the Hit status
- Calculation of the maximum analog value of the channels (with the corresponding channel address)
- Calculation of the analog sum of all channels
- Reporting of the Hit status of all channels
- Reporting of the duration of the entire read process
- Access to the transfer memory in programmable mode

As shown in Fig. 4, the data written to the transfer memory can be formatted in a versatile manner, depending on the information required during the design phase. The data can be formatted with a HEADER to head the data, the body of the analog data itself, and finally a FOOTER to close it. Different parameters are available for each section, which can be used in any combination required.

The following parameters are available for the HEADER:

- 32-bit timestamp in seconds
- 32-bit timestamp in tenths of a microsecond
- Timestamp shift error
- Event sequential counter value

While for the analog data body these parameters are available:

- 32-bit timestamp in tenths of a microsecond
- Channel address
- Relative hit info
- Analog value read by the ADC

Finally, the following parameters are available for the FOOTER:

- Hit status of all channels
- Sum of all analog values read
- Maximum value of all analog values read
- Address of the channel with the highest value
- Total elapsed acquisition time

Again the transfer memory is a double buffered memory stack, and as with previous modules, this stack is sized appropriately to allow for the accumulation of multiple formatted data of the acquired analog peaks.

Notes:

The data formatting procedure for writing is described in VHDL code in a separate package from the rest of the body that defines the RTL structure and the controller's state machine. This simplifies the procedure for any user interested in simply defining the instrument's data formatting, without necessarily having to grapple with the entire controller's VHDL code.

This is useful for reusing this controller on projects other than the one in question.

Trigger Manager module

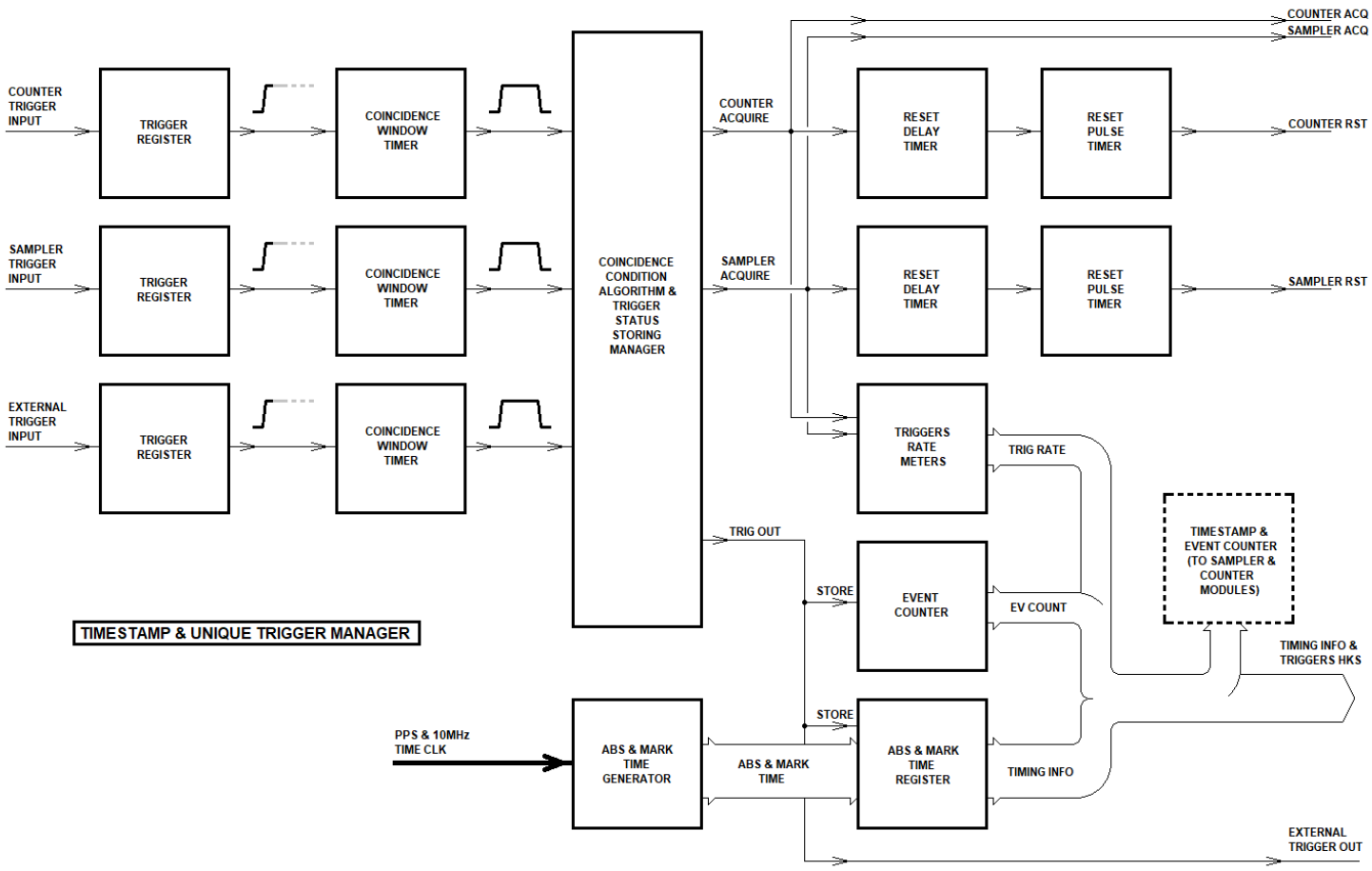

Fig 5: Trigger Management IP-Module

The "Trigger" IP-Module takes care for managing acquisitions in the Counters and Sampler modules.

As shown in Fig. 5, the acquisition trigger signal comes from these two modules, produced by their respective circuit algorithms. These trigger signals are initially processed to identify the leading edge and then fed into a timer to create a programmable coincidence window.

An external trigger signal is available, coming from other possible scientific instruments, which is treated in the same way as the previous two, in order to contribute to the management of the acquisitions.

The coincidence windows are evaluated by a programmable algorithm, after which the module produces the Trig-Out output signal. This signal stores the 64-bit Timestamp value and the sequential counter value in dedicated registers for timestamping events. Trig-Out also serves as an external trigger for use by other scientific instruments.

Simultaneously with the Trig-Out output, the module manages the ACQUIRE start and end acquisition signals to the Counters and Sampler modules. Finally, it produces the related programmable reset signal.

Notes:

This IP-Module also takes care of an essential function, that is, the synchronization, with a reference clock, of the external PPS and 10MHz signals, coming from a possible external CSAC or RTC module.

I/O Communication module

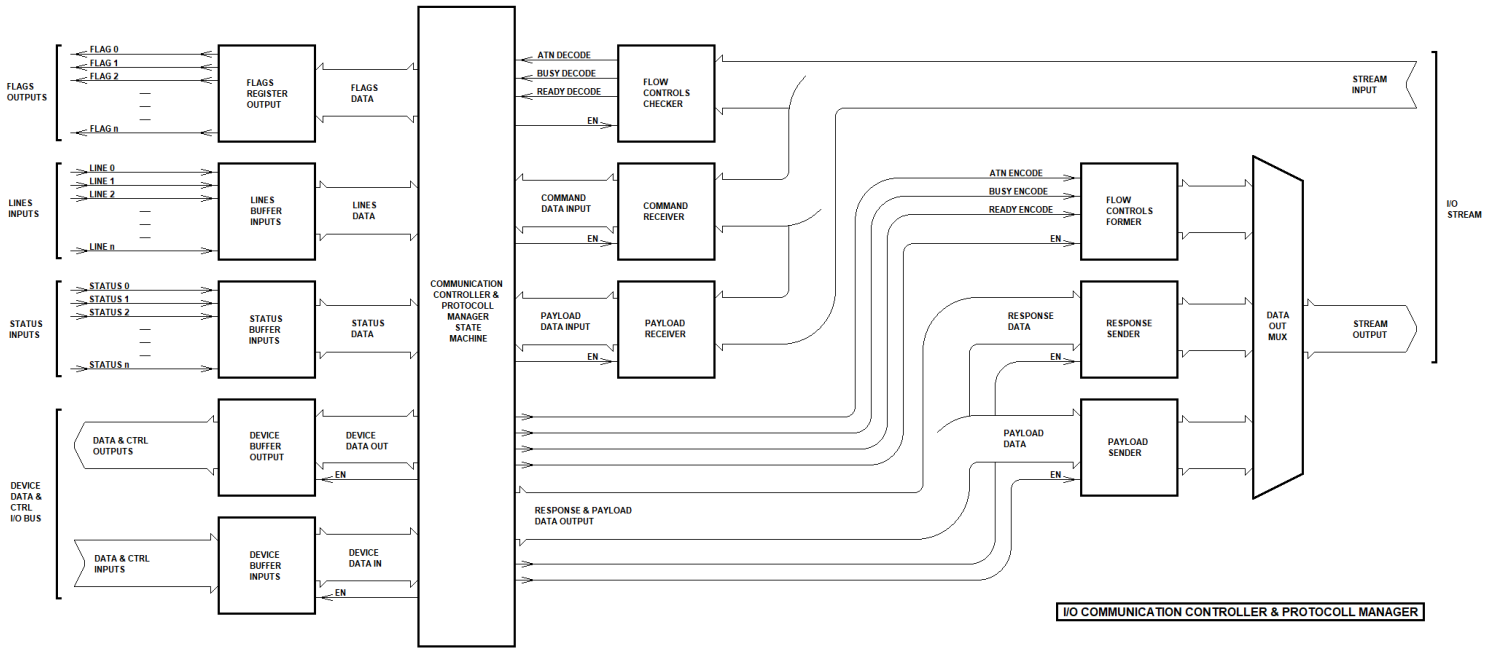

Fig 6: I/O communication management IP-Module

The "I/O Communication" IP-Module take care for managing internal communication between the project's IP-Modules, as well as managing communication with an external host computer, through the "I/O Interface" IP-Module.

Any I/O operation to and from the various IP-Modules connected to it is initiated by a related I/O transaction created by the host computer. In short, it is the computer that manages the entire instrument by writing and reading appropriately formatted data streams.

As shown in Fig. 6, the input stream of data from the host computer simultaneously enters three control stages. The first at the top is used to recognize and decode specific byte sequences in the stream, to control the I/O flow. The second is used to receive the command string to be executed, and the third is used to receive the data payload string. These three control stages are managed by a communications controller, which in turn manages three other output stages for composing the output stream to the host computer. The first encodes the byte sequence for flow control, the second sends the string in response to the command, and the third sends the requested payload string.

The communication controller is essentially a state machine that, through flow control signals, regulates the flow of transactions between the computer and the scientific instrument in use. Each transaction is subject to a proprietary protocol developed specifically for the purpose, which provides for three types of transactions: a TASK transaction without transferring PAYLOAD data, a WRITE transaction with transfer of the PAYLOAD sent by the Host and received by the controller, and a READ transaction with transfer of the PAYLOAD received by the Host and sent by the controller.

Each transaction is initiated by the Host and involves these steps:

- Sending ATN
- Receiving ATN_RET
- Sending COMMAND
- Waiting for COMMAND processing
- Receiving RESPONSE
- Waiting for PAYLOAD preparation
- Sending or receiving PAYLOAD, if applicable for the transaction type
- Waiting for transaction completion

The ATN is an attention signal, sent by the Host to the controller, to wake up the device and synchronize it with the incoming transaction stream. The controller in turn responds with an ATN_RET to indicate that it is ready to receive the stream. The COMMAND is composed of a predefined string of bytes and represents the transaction

command. The RESPONSE is also composed of a string of bytes representing the response to the received command. Finally, the PAYLOAD is the actual data block to be transferred.

Pauses must be inserted between the various phases to allow both the controller and the Host time to process the command and response, as well as prepare to send or receive the PAYLOAD data. During this interval, the Host and controller can send each other the BUSY control byte. At the end of this wait, the READY byte must be sent, indicating the resumption of the transaction flow.

The COMMAND string decoding, as well as the RESPONSE encoding, are fully definable during the design phase based on the specific needs of the control software developed by the instrument's user. The numerous available parameters allow for the flexible and efficient creation of the list of commands and responses required for complete instrument management.

The controller allows a transaction speed with the Host of over 100MB/s.

For internal communication between the project's IP-Modules, the controller features a proprietary, high-speed, full-duplex bus that allows for interaction with each project module in the shortest possible time. This is essential when managing multiple modules, in order to minimize overall acquisition downtime.

Furthermore, the controller provides multiple logic flags that can be used to quickly and easily configure the various operating modes of the project's IP-Modules. It is also possible to read the status of various static lines that can be used to monitor the conditions of various input signals to the FPGA. It is also possible to read the status of various internal logic statuses that are specifically used by the controller to monitor, and then act accordingly, in the event of a busy or error condition of the various IP-Modules being managed. All of these operations can be performed through simple TASK-type transactions without transferring payloads, and therefore very quickly in terms of communication time with the host computer.

In summary, a very versatile and efficient communication controller has been created between the interconnected IP-Modules of the project and the management host computer, which can be used for multiple projects, even those different from the one in question.

Notes:

The decoding and encoding procedure for the COMMAND and RESPONSE byte strings is described in VHDL code in a separate package from the rest of the body that defines the RTL structure and the controller's state machine. This is to simplify the procedure for any user interested in simply defining the instrument's command and response list, without necessarily having to grapple with the entire controller's VHDL code.

This is useful for reusing this controller on projects other than the one in question.

The complete description of the communication protocol, the proprietary bus for the interconnected IP-Modules, and the encoding and decoding parameters of the commands and responses is delegated to another document.

I/O Interface module

The “I/O Interface” IP-Module handles the interface between the “I/O Communication” IP-Module, which we recall is specifically designed to manage the communication protocol, and a host computer.

Various interface controllers have been designed to meet a variety of connection needs. These controllers, developed over the years by the IASF-PA Electronics Laboratory for the development of numerous scientific experiments, are capable of interfacing directly or through common commercial chips to standard communication buses with various transmission speeds, including:

- UART, up to 10MBaud
- SPI, up to 100Mbps
- USB2, via the FTDI FT232H chip in 245-Async mode, up to 10MByte/s, and 245-Sync mode, up to 40MByte/s
- eSATA + USB2, via the JM20330 and JM20338 chips, up to 100MByte/s (the system is seen by the OS as a standard HDD, without the need for drivers)
- USB3, via the FTDI FT60x chips, up to 400MByte/s

Additionally, a proprietary Master & Slave IP-Module has been developed that can provide a high-speed, high-efficiency serial point-to-point connection, designed to connect multiple scientific instrument units to a single master control unit. The controller has the following features:

- Dual low-latency I/O buffers
- Packet transmission with reconfigurable parameters
- CRC check
- Automatic retransmission in the presence of transmission errors
- Handshake with adaptive timing

This controller can manage various types of high-speed gigabit SERDES chips, such as the Texas TLK family chips, capable of delivering transmission speeds of 2.5Gbit/s and beyond. It can also manage the embedded SERDES of some FPGA families, such as the CYCLONE V-GX, capable of reaching speeds exceeding 6Gbit/s. These connections are typically made with just two differential, low-EMI transmission lines (TX and RX), using simple twisted, AC-coupled copper wires, thus also providing useful galvanic isolation between the motherboard and the various connected instrument units. This type of 2-line GIGABIT transmission also allows for the use of various commercial optical SFP modules, such as the AVAGO AFBR-57R5APZ or similar, to create a fiber optic connection, necessary when the length of the connections is considerable while simultaneously maintaining a high transmission speed.

Finally, an additional IP interface module was developed that emulates the behavior of Texas TLK family SERDES chips, allowing it to be paired with the SERDES controller described above. This controller allows for a high-speed serial connection between the main unit and various instrument units, though the maximum speed is limited to 500 Mbps, and it uses four transmission lines instead of two, all differential and AC-coupled.

This additional connection option can be useful as a compromise between speed and simplicity, when the transmission speed requirements are not very high, you do not want to add technical and space complications in the use of external SERDES chips, and you want to remain on low-cost FPGA families without integrated embedded SERDES.

Services modules

In this project, several IP-Modules dedicated to the management of electronic circuits external to the FPGA have also been implemented, such as:

- Communication with various sensors present in the instrument
- Management of ADCs and DACs to monitor the instrument's operating voltages and currents, and to control external high voltages and biases
- Communication with an ASIC (if available) to manage the configuration of its internal registers
- Generation of test pulses

For this purpose, several proprietary controllers have been developed for interfacing with standard transmission protocols such as SPI, I2C, and UART. These controllers, built without the use of virtual or embedded processors, are highly flexible in their use and are well-suited for reuse in various projects.

Furthermore, a controller for pulse generation has been created, programmable in width, amplitude and period, useful for functional verification of the Front-End.

Conclusions

As previously mentioned, this project was fully developed, tested, and successfully deployed by the technical staff of the Electronics Laboratory of the IASF-PA Institute of INAF, using Intel's Altera platform and the Quartus development environment. Both SIPM and MaPMT detectors were used for testing, connected to the relevant front-ends, either in the form of ASICs, such as Weeroc's RADIOROC, or with custom high-speed discrete electronics.

Although all IP-Modules have been described in standard VHDL language, making them easily compilable on other platforms, any porting of these modules must be carefully considered, as their development has been modeled around the resources available specifically on Altera FPGAs. Furthermore, the use of a graphical environment to create the Top-Level Entity and to connect the various IP-Modules to each other, an operation that simplifies and speeds up development, does not allow for immediate porting of the project. Furthermore, the definition of the temporal constraints, which are mandatory for the correct functioning of the modules, is intrinsically linked to the development platform and therefore cannot be directly reused in other contexts.

Remaining on the Altera platform, all the project's IP-Modules are easily and fully reused, individually or in full, on various types of projects, even those different from the one in question.

The supporting documentation mentioned in the description is currently being developed and completed. Additional documentation is widely available directly on the Altera.com website.