ATLAS Muon MDT

MROD Module:

MROD-X Design, Changes with respect to the MROD-1 design

Document Version: 1
Document Issue: 0
Document ID: ATLAS-TDAQ-2005-XXX or EDMS Id XXXXXX
Document Date: 26-Oct-2005
Document Status: Draft

Abstract

This document contains an overview of the MROD-X design. The MROD-1 design, the predecessor of the MROD-X design, will serve as a reference.

Institutes and Authors:

NIKHEF Amsterdam: Peter Jansweijer

Radboud University Nijmegen: Thei Wijnen

Table 0 Document Change Record

Title:	ATLAS MROD Module: MROD-X Design, Changes with respect to the MROD-1 design		
ID:	ATLAS-TDAQ-2005-XXX or EDMS Id XXXXXX		
Version	**Issue**	**Date**	**Comment**
1	0	26-Oct-2005	First version (draft)

Table of Contents

1 INTRODUCTION 4

2 MROD-X DESIGN DESCRIPTION 5

2.1 ROCKETIO BETWEEN MROD-IN AND MROD-OUT FPGAS 5

2.2 TDC LIMIT REGISTER 6

2.3 BACKPRESSURE AND ROD-BUSY 7

2.4 TTC FIFOS 7

2.5 EVENT BUILDER 8

2.6 SPY 8

2.7 ZERO SUPPRESS OVERRIDE 9

2.8 REMOTE FPGA CONFIGURATION 9

2.9 DATE AND REVISION REGISTERS 10

2.10 MROD-X IDENTIFIER 10

2.11 CSM LINKS 10

2.12 TEST GENERATOR 11

2.13 TEMPERATURE READOUT 11

2.14 S-LINK FLUSH MODE 12

3 REFERENCES 13

1 Introduction

This document contains an overview of the MROD-X design. The MROD-1 design, the predecessor of the MROD-X design, will serve as a reference. The MROD-1 design details (Tetris register, buffer memory etc.) will not be described here since the functionality is already described extensively in other documents and presentations [1]. A major difference between the MROD-1 and the MROD-X is the use of RocketIO [2] links in stead of SHARC-links between the SHARC processors [3] for event data transport. Studies on the performance of the MROD-1 showed that the links between the SHARC processors (40 MB/s) formed a bottleneck. The MROD-X design still has these SHARC processor links intact but adds RocketIO links (160 MB/s) between the eight input FPGAs (called MROD-In FPGA) and the output FPGA (called MROD-Out FPGA) so there is no longer a data bottleneck in the MROD-X module. Because the original MROD-1 data path is still intact, the MROD-X can run in “MROD-1 mode” for test purposes. More important however is that the MROD-1 mode data path opens the possibility to spy on the data for debugging and monitoring purposes. The SHARC processors are no longer used for data taking so they can now be completely exploited for these tasks. Like in the MROD-1 design the SHARCs take care of configuration, control and handling of error conditions. In the MROD-X design there is processor power available to handle these conditions.

2 MROD-X design description

2.1 RocketIO between MROD-In and MROD-Out FPGAs

The RocketIO links are described in reference [2]. One RocketIO consists of a high speed serial transmitter and a receiver. The bit stream that is sent over these serial connections is 8B/10B encoded as described in [4]. This paragraph will focus on the RocketIO links between the FPGAs.

The bandwidth of each RocketIO is set to 160 MB/s (1.6 Gb/s). Theoretically, the RocketIOs could run on 3.125 Gb/s, however in this case 1.6 Gb/s is sufficient since the output S-Link (also 160 MB/s) could be saturated with one RocketIO. Running on a lower bit rate consumes less power.

Figure 1 gives a schematic diagram of the connection between a MROD-In FPGA and MROD-Out FPGA based on a RocketIO connection. The nature of 8B/10B coding opens the possibility to send “extended data”. This has been used to implement two input data streams.

One data stream carries the event fragments built out of the TDC data received over the Chamber Service Module (CSM) link. The other data stream carries 12 bit event lengths that are sent together with associated 12 bit Event-IDs.

The event builder in the MROD-Out FPGA needs to have the event length information (LWC: Link Word Count) prior to the event data as described in the data format document [5]. The MROD-In FPGA however only “knows” the event length when it transferred the last word from its buffer memory to the output FIFO. The RocketIO link between the MROD-In and MROD-Out FPGA, together with its buffers solves this “reverse” problem.

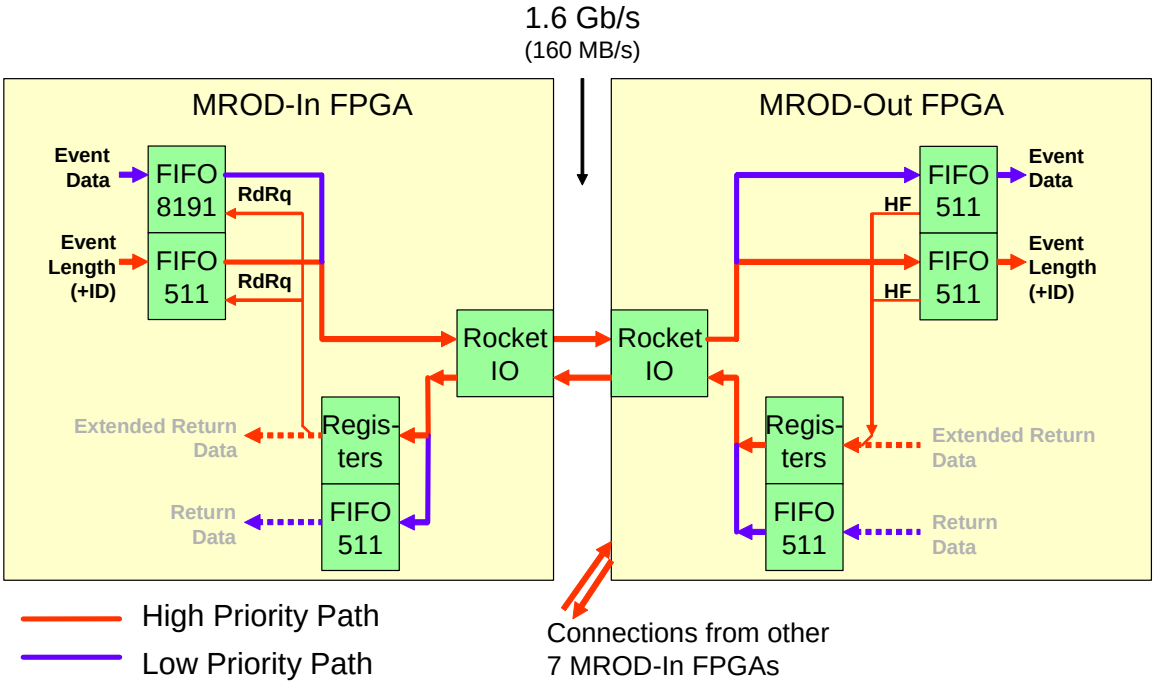

Figure 1: Schematic diagram of the connection between the MROD-In FPGA and MROD-Out FPGA based on a RocketIO connection.

The event length data path is made a high priority path. If an event length word needs to be transferred then it will be inserted in the RocketIO bit stream preceded by a Carrier Extend (ordered set K23.7 see reference [4]). The receiving RocketIO in the MROD-Out FPGA recognizes the Carrier Extend and stores the following word in the Event Length FIFO. The buffer size for the event data is always at least 8K words. If one can assure that a complete event always fits in this buffer size then the event length word will always be available for the event builder in the MROD-Out FPGA before the event data is read out of the RocketIO buffer. The TDC-Limit mechanism guarantees that this condition is always met (see TDC Limit register).

The RocketIO is bidirectional so the half full flags of the receiver FIFOs in the MROD-Out FPGA can be transmitted back to the transmitting MROD-In FPGA. These flags are also sent via a high priority data path. When the half full flag is received in the MROD-In FPGA then the corresponding FIFO is no longer read and the data transfer over the RocketIO link stops. In this way full backpressure can be accomplished up to the MROD-In buffer memory (see also “Backpressure”).

The FIFO flags do not take the full 32 bit word sent. There are bits left that could be used to transmit data over the high priority return data path from the MROD-Out to the MROD-In FPGA. At the moment these bits are not used. Next to the high priority return data path there is a low priority return data path from MROD-Out to MROD-In FPGA. At the moment this data path is not used.

On the MROD-In side, as well as on the MROD-Out side, the status (Link-Up/Down) of each RocketIO link can be read. After power-up, the RocketIO links are “Up” by default, but should it be necessary then each link can be reset at each side of the link. When a link on the MROD-Out goes “Down” then this can be signalled via an interrupt to the MROD-Out SHARC processor.

2.2 TDC Limit Register

As described in the previous section, a mechanism is needed to ensure that an event fragment always fits in the 8K buffer size. The MROD-In FPGA incorporates a TDC Limit register for this purpose. The value of this 12 bit register determines how many TDC data words (TDC-Header (=BOT) and TDC-Trailer (=EOT) not included, see also [5]) are to be transferred at maximum to the output FIFO. After the transfer of 1 to 4096 (programmable) data words, the surplus will be read from the buffer memory partition until the EOT is found. These data words are not sent to the output data stream. As the EOT will be sent, one can see that the limiter came into action from the word count in the EOT word and the number of data words received.

The default value for the TDC Limit Register is 0x60 which means that 99 words (97 data words, BOT and EOT) per TDC is the maximum that any TDC can transmit into the output FIFO. Thus for 18 TDCs this is 18 * 99 = 1782 words which is well below the 8K buffer depth. Even so, the event builder in the MROD-Out FPGA could receive 8 * 1786 (1782 + 4 extra “MROD-In event envelope” words). This amount of words can be accommodated in the 16K MROD-Out Spy data buffer as described later.

The TDC-Limit register is new with respect to the MROD-1. The software used on the MROD-1 also needed a mechanism to prevent buffers from overflowing. For this purpose the Maximum Event Size register was used, although it was not implemented to be used as such. This register still exists in the MROD-X. Its purpose is to prevent a faulty input data stream from blocking the output bandwidth. When an input TDC keeps sending garbage data without an EOT then the maximum event size is exceeded and causes a shutdown of the readout for that particular TDC. During normal data taking this situation should never occur.

2.3 Backpressure and ROD-Busy

The MROD-X module provides full backpressure up to the MROD-In buffer memory. When, for some reason, the Read Out Link (ROL) signals a Link FIFO Full (LFF) condition then the event builder in the MROD-Out FPGA can no longer transfer data from the RocketIO links to the ROL. Subsequently the buffers in the RocketIO link will fill and eventually they will signal an almost full condition to the MROD-In FPGA. When this occurs, the output controller in the MROD-In FPGA can no longer transfer data from its buffer memory to the RocketIO link. This will cause the buffer memory to fill. For the MROD-X module there is no possibility to signal to the CSM that it can not handle more data since the CSM links are unidirectional. Thus the only possibility for the MROD-X is to signal a “ROD-Busy” condition to the Central Trigger Processor (CTP) and a yellow LED (“B” for Busy) on the front panel is lit. This is done as soon as the MROD-In buffer memory associated with the CSM link gets half full.

For each of the 18 TDCs on each input CSM link there are 8K words of buffer memory available. Consider the situation where a lot of small events are received. In such a situation the buffer memory could easily store all the data without getting half full. However, the Input to Output (I2O) FIFO in the MROD-In FPGA could fill up rapidly. Recall that the I2O FIFO (512 words deep) stores an entry each time a complete event is received from each TDC on the CSM input link. To cover this situation, the half full status of the I2O FIFO will also cause ROD-Busy.

Finally, ROD-Busy can also be generated by the half full condition of the TTC FIFOs (taking care of receiving Event-ID, Bunch-ID and Trigger-Type) in the MROD-Out FPGA. Details about these FIFOs will be described below.

The status of all individual ROD-Busy sources can be read by the SHARC processors and each source can be masked. By default all sources are enabled. The ROD-Busy is signalled with a red front panel LED (“ROD-Busy”). The ROD-Busy signal is available via a NIM output. ROD-Busy is also transferred via the P3 backplane to the TIM [6].

2.4 TTC FIFOs

The MROD-X module receives and stores Event-ID, Bunch-ID and Trigger-Type information from the TTC system. This information is distributed in the MROD crate via the TIM [6], and is stored in two sets of TTC FIFOs working in parallel and independently. The first set serves the event builder in the MROD-Out FPGA, the second set of FIFOs is connected to the MROD-Out SHARC processor. This creates the possibility for the SHARC processor to gather information from the TTC system for debug, test and monitor purposes. The Event/Bunch-ID FIFO and Trigger-Type FIFO that are connected to the SHARC processor do not interfere with the normal data taking procedure.

The TIM sends Event-ID and Bunch-ID in one serial stream to the MROD-X module, the Trigger-Type is sent as another serial stream. Both streams are de-serialized in the MROD-Out FPGA and stored in the Event/Bunch-ID FIFOs and the Trigger-Type FIFOs (each 511 words deep), respectively. When there is an entry available in both Event/Bunch-ID and Trigger-Type FIFOs (by definition there should always be an entry in the Event/Bunch-ID FIFOs for each entry in the Trigger-Type FIFOs and vice versa) then this is a signal to the event builder in the MROD-Out FPGA that data are to be expected from the participating RocketIO links connected to the MROD-In FPGAs. When the FIFOs connected to the event builder get half full then this will result in a ROD-Busy.

2.5 Event Builder

For each CSM link the data is received via a RocketIO link connecting MROD-In and MROD-Out FPGA. The event builder in the MROD-Out FPGA waits for an entry in the TTC FIFOs. When an Event/Bunch-ID and Trigger-Type is received then this signals the event builder that data is to be expected from the participating RocketIO links. RocketIO links can be enabled and disabled in the event building process via the MROD-Out "Channel Enable Register".

The event builder reads the TTC information and transfers the header of the event to the ROL. Several programmable registers are available to store some important words (Format Version, Module-ID, Run Number, and Detector Event Type) that are part of the event header. Then the event builder reads the event length information from the first RocketIO link participating in the event building. Next the event data of that link is transferred to the ROL and the event length information is placed into the stream at the appropriate place (Link Word Count, LWC [5]). Event-IDs and Bunch-IDs in the data stream are checked against the Event/Bunch-ID that was received via the TTC FIFOs. In case of discrepancies an error flag is raised in the MROD Status Word (MSE1). This process is repeated for each participating RocketIO Link. The total number of data words (from BOB up to EOB) transferred to the ROL is counted and sent as the Number of Data Elements (NDE) in the trailer.

For test purposes the event builder can be set to a mode where it does not wait for TTC information (and thus does not check Event/Bunch-IDs) in cases where there is no TIM available. By default this test mode is disabled.

2.6 Spy

The SHARC links, used in the MROD-1 for event data transfer, are still intact in the MROD-X. These SHARC links can now be used to spy on the data. Spying can be done on the level of the four MROD-Ins or on the level of the MROD-Out. In either case there are several modes for the transfer of the data: *No*, *All* or *One in "n"* events (where n = [0..65535]). The normal data flow from CSM links via the RocketIO links to the event builder and to the ROL is never blocked by the spy channel. When the data from the spy channel is not processed fast enough, thus causing the spy channel to fill up, then events that ought to be spied are simply ignored. Should this be a problem then one should downscale the amount of events that must be spied on.

The spy data is sent via a FIFO to the SHARC where the event data is stored. In parallel the event length is stored in an event length FIFO. The event data FIFO is to be transferred using an "endless DMA" on the SHARC processor. The SHARC processor can moderate this stream of data by reading the event length FIFO. If there is no space left for another event length word in the event length FIFO, then the spy mechanism in the FPGA stops writing the event data.

For the MROD-In the event data FIFO is 512 words deep and the event length FIFO is 16 words deep. For the MROD-Out the event data FIFO is 16K words deep and the event length FIFO is 4 words deep. Note that the TDC Limit Registers of the MROD-Ins take care of the maximum size of an event (see TDC Limit Register).

For debugging purposes only, the spy mechanism can be put in a mode where it is blocking the main data stream whenever the spy channel is getting full. In this case no spy event data gets lost and the ROD-Busy mechanism gets into action when too many data is fed into the MROD-X. By default the debug mode is off.

Note that the MROD-X can run in “MROD-1 Mode” when the spy channel is set to “Spy All” in “Debug Mode” (thus all data is transferred and nothing gets lost).

Data for each CSM link are transferred to the SHARC processor (one SHARC serves two CSM links) when spying in the MROD-In. Data of a complete event are transferred to the output SHARC when spying at the MROD-Out level. These data are exactly the same as the data that are sent over the ROL.

2.7 Zero Suppress Override

The event data can be zero suppressed. The MROD-X module however has a zero suppress override mode which can be enabled or disabled. When this mode is enabled then non-zero suppressed data is sent every n-th event (where n = [0..65535]) into the main data stream. The first event of a run is always passed non-zero suppressed. This mechanism may be very helpful for debugging purposes. The default is set to override zero suppress once every 1024 events.

2.8 Remote FPGA Configuration

There are two configuration PROMs, one for eight MROD-In FPGAs and one for the MROD-Out FPGA on the MROD-X board. The FPGA JTAG chain runs through all FPGAs and Configuration PROMS (see figure 2).

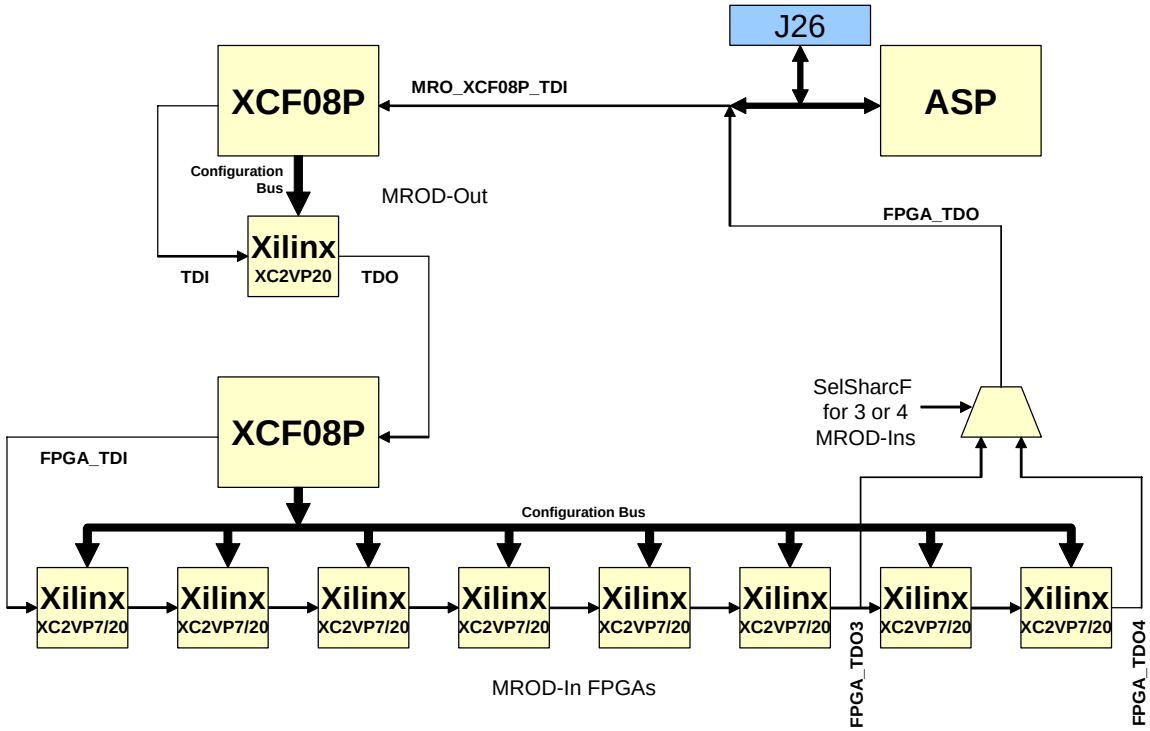

Figure 2: FPGA JTAG Chain

This JTAG chain can be connected to a Xilinx download cable via the front of the MROD-X module. A sense on one of the pins of this standard Xilinx connector (J26) detects that a download cable is connected.

When no download cable is connected then the JTAG chain is automatically routed to an on board JTAG Addressable Scan Port (SN74LVT8996 [8]) which in turn is connected to the Module Test and Maintenance (MTM) bus [7] that is implemented on the VME64x Backplane. This MTM bus can be driven by a Xilinx download cable that is plugged into the USB port of the Crate Controller. Xilinx provides special software, called IMPACT, to configure the FPGAs on the JTAG chain. IMPACT can run in batch mode which creates the possibility to send a so called "shadow protocol" [8]. Each MROD-X module contains one Addressable Scan Port which compares the address encoded in the shadow protocol with the address from the geographical address pins on the VME64x backplane. When the address matches then the Addressable Scan Port is set to transparent mode (the green front panel LED "ASP Conn" is lit) thus creating a connection from the IMPACT software to the JTAG chain of the selected module. Now IMPACT has full control over the FPGAs and the configuration PROMs. Since the Crate Controller is connected to the network, it is possible to do a remote update of the FPGA firmware. This circuit could be referred to as the "easy-jet ticket saver".

Note that the Addressable Scan Port is overruled by a Xilinx Download Cable that is plugged into the front connector. When the FPGAs load their configuration from the configuration PROMs, a red LED ("FPGA Config") on the front panel is lit. The FPGAs are configured after power-up or after a VME System Reset.

2.9 Date and Revision Registers

Because Remote FPGA Configuration is simple, it is very easy to get lost in various revisions of the FPGA firmware. Therefore each FPGA has a Date and Revision Register implemented from which the date and the revision of the FPGA firmware can be read directly. The value of this register is determined during synthesis of the VHDL code. The operating system date and a revision number are automatically passed to the synthesis tool using a TCL script.

2.10 MROD-X Identifier

Each MROD-X module has an identifier chip (Dallas DS2401 [9]) on board with a 48 bit unique number. This identifier, together with an 8-bit CRC checksum can be read out through the registers of the MROD-Out FPGA. This makes each MROD-X module unique and traceable.

2.11 CSM links

The Xilinx RocketIO links are not only used for the data transfer between the MROD-In and MROD-Out FPGAs, they are also used to connect the optical transceiver of the CSM link directly to the MROD-In FPGA. The input link interfaces are thus integrated on the MROD-X board whereas the MROD-1 used modified S-Link daughter boards as interfaces to the input links.

Each CSM input link has three LEDs where the status of the input link can be seen. A green LED signals "Link Up" ("U"), a red LED signals "Link Error" ("E") and a yellow LED signals a "Busy" ("B") condition (see "Backpressure"). The link is considered "Up" when the receiver is bit- and word-synchronized with the transmitter (initially transmitting IDLE ordered sets [4]). An error condition occurs when an Error Propagate ordered set is received.

2.12 Test Generator

The CSM links that are input to the MROD-X module are unidirectional. This leaves the transmitter of the optical transceiver on the inputs of the MROD-X unused. The MROD-In FPGA has a build in test generator that drives this transmitter. Test data can be fed into the CSM link by connecting a loop back fibre.

The test generator is in fact a FIFO that can be filled by the SHARC processor. After loading this test data the SHARC turns on the controller that sends the data out over the loop back fibre. Data read out of the test FIFO are immediately stored in the FIFO again. The Event-ID contained in the test data is automatically updated, so the test data can be generated at high speed (red arrows in figure 3).

The transfer of test events from the test FIFO can be done either free running or triggered. When triggered mode is selected, an event is sent each time a Level 1 Accept (L1A) from the TIM is received. In this way a controlled test can be done using several or all CSM links in parallel. The trigger rate can be controlled through the ROD-Busy.

Note that the test generator did not exist in the MROD-1, however the MROD-1 has an internal loop back mode where test data can be fed into the pipeline of the FPGA. This internal test mode is still available on the MROD-X (yellow arrow in figure 3).

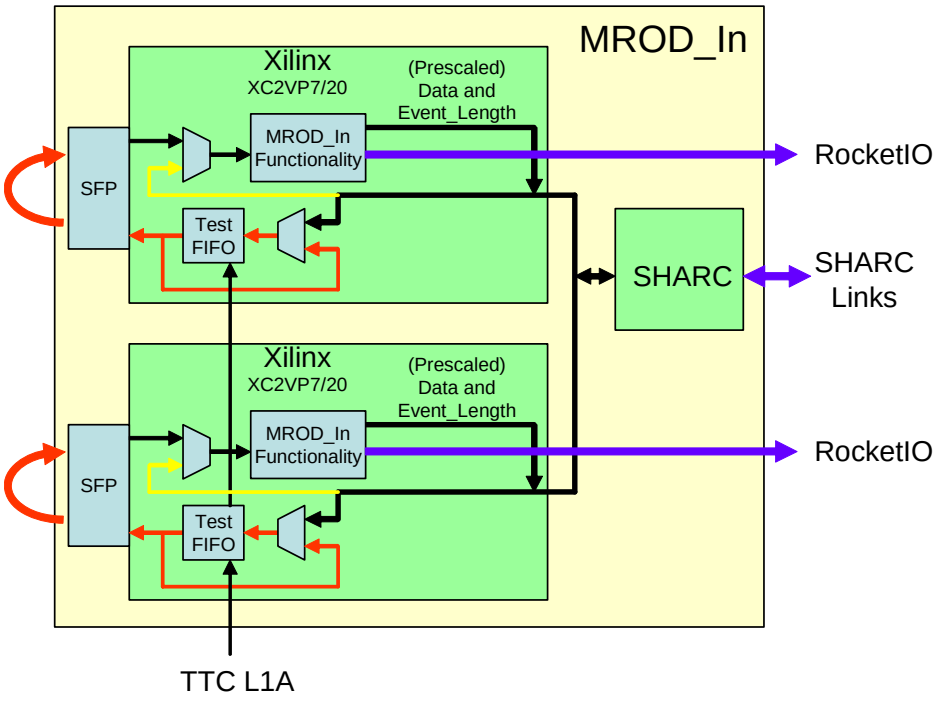

Figure 3: Test generator and internal test mode

2.13 Temperature readout

The temperature of each FPGA on the MROD-X board can be readout. The Virtex2Pro FPGAs have an integrated temperature sensing diode on the silicon. This diode is connected to a “Remote Temperature Sensor” device (MAX1618 [10]) next to each FPGA. This device can be controlled through a register in the FPGA which on its turn can be accessed by the SHARC processor.

2.14 S-Link flush mode

For trigger rate testing it proves very useful to have a mode where the ROL can be put into a state were it absorbs all data, always. For the MROD-1 usually a dummy connector was put onto the S-Link output with a pull-up resistor on LDOWN_n and LFF_n (link always up and never full). For the MROD-X this mode can now be controlled by software by setting a bit in the MROD-Out “S-Link Status and Interrupt Register”. When this mode is set then data from the event builder are flushed but not transferred to the ROL. By default this mode is off.

3 References

1 http://www.nikhef.nl/~peterj/MROD/MROD-1.html

2 RocketIO Transceiver User Guide: http://direct.xilinx.com/bvdocs/userguides/ug024.pdf

3 ADSP-21160 SHARC DSP Hardware Reference: http://www.analog.com/Processors/Processors/sharc/technicalLibrary/manuals/32BitIndex.html

4 IEEE std. 802.3. (Chapter 36)

5 The MROD data format and the tower partitioning of the MDT chambers, T. Wijnen, Radboud University of Nijmegen:

http://doc.cern.ch/archive/electronic/cern/others/atlnot/Note/daq/daq-2003-023.pdf

6 http://www.hep.ucl.ac.uk/atlas/sct/tim/

7 IEEE 1149.5 MTM bus

8 SN74LVT8996 datasheet, http://focus.ti.com/docs/prod/folders/print/sn74lvt8996.html

9 DS2401 datasheet, http://www.maxim-ic.com/DS2401

10 MAX1618 datasheet, http://www.maxim-ic.com/MAX1618