The final prototype of the Fast Merging Module (FMM) for readout status processing in CMS DAQ

Roberta Arcidiacono[5], Vuko Brigljevic[1,6], Giacomo Bruno[1], Eric Cano[1], Sergio Cittolin[1], Samim Erhan[7], Dominique Gigi[1], Frank Glege[1], Robert Gomez-Reino Garrido[1], Michele Gulmini[1], Johannes Gutleber[1], Claude Jacobs[1], Peter Kreuzer[9], Giuseppe Lo Presti[1], Ildefons Magrans De Abril[1], Nancy Marinelli[2], Gaetano Maron[3], Frans Meijers[1], Emilio Meschi[1], Steven Murray[1], Alexander Oh[1], Luciano Orsini[1], Marco Pieri[8], Lucien Pollet[1], Attila Racz[1], Peter Rosinsky[1], Christoph Schwick[1], Paris Sphicas[1,9], Joao Varela[1,4]

1. CERN, Switzerland
2. Institute of Accelerating Systems and Applications, Athens, Greece
3. Legnaro University, Italy
4. LIP, Portugal
5. MIT, USA
6. Rudjer Boskovic Inst., Croatia
7. UCLA, USA
8. UCSD, USA
9. University of Athens, Greece

Abstract

The Trigger Throttling System (TTS) adapts the trigger frequency to the DAQ readout capacity in order to avoid buffer overflows and data corruption. The states of all ~640 readout units in the CMS DAQ are read out and merged by hardware modules (FMMs) to obtain the status of each detector partition. The functionality and the design of the second and final prototype of the FMM are presented in this paper.

I. INTRODUCTION

The TTS [1,2] is an element of the CMS data acquisition system [3]. It regulates the Level 1 Accept trigger rate (LV1A) to prevent overloading of any electronic devices in charge of moving, processing, and storing the data from the very front-end electronic down to the storage media. The TTS consists of two parts: the synchronous TTS (sTTS) and the asynchronous TTS (aTTS). The sTTS deals with devices with small storage buffers (i.e. front-end electronic) and requires a fast reaction time. The aTTS deals with the readout and filtering computers for which the reaction time can be relaxed due to the availability of significant memory resources. The sTTS is implemented with hardware modules (FMMs) whereas the aTTS is implemented through network messages. A global view of the TTS is shown in Figure 1.

The Fast Merging Module receives and concentrates the states of Front-End Drivers (FED). The FMMs produce a single state per detector partition[1] (see Table 1). These elements form the sTTS system. The Trigger Control System (TCS) is able to handle 32 detector partitions. Based on the states provided by the FMMs, the TCS will adapt the LV1A rate if necessary.

[1]When more than 1 FMM is needed for a partition, a second layer of FMM is merging the first layer to form the partition state. The FMM can be splitted to act as 2 independent FMMs: in this case, a half FMM (1/2) is counted.

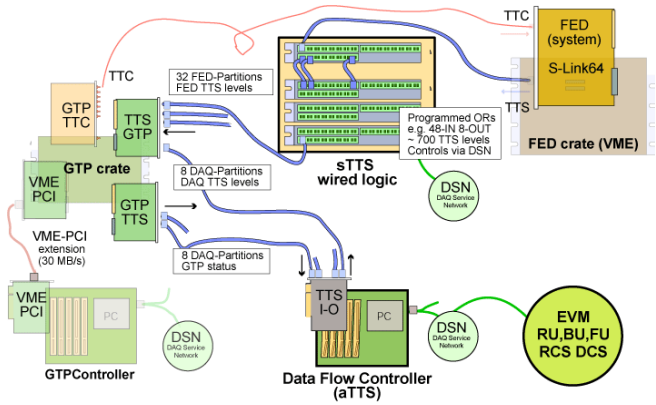

Figure 1: TTS global view

Table 1: Sub-systems and FMMs

Detector (# FEDs)	Partition (# FEDs)	# FMMs per partition	# FMMs per detector
Pixel (38)	Barrel (32)	2 +1/2	2
	Forward (6)	1/2	
Tracker (440)	Inner (114)	6+1/2	25
	Outer (134)	7+1/2	
	Endcap+ (96)	5+1/2	
	Endcap- (96)	5+1/2	
Preshower (~50)	SE+ (25)	2+1/2	5
	SE- (25)	2+1/2	
ECAL (54)	EB+ (18)	1	3
	EB- (18)	1	
	EE+ (9)	1/2	
	EE- (9)	1/2	
HCAL (32)	Slice_1 (6)	1/2	3
	Slice_2 (6)	1/2	

	Slice_3 (6) HO (8) HF (6)	1/2 1/2 1/2	
Muon DT (5)	Barrel + Barrel -	1/2 1/2	1
Muon RPC (6)	Barrel + Barrel – Endcap + Endcap -	1/2 1/2 1/2 1/2	2
Muon CSC (8)	Endcap + (4) Endcap – (4)	1/2 1/2	1
Calo-trig **Glob-muon** **Glob-trig**	na na na	1/2 1/2 1/2	2
Muon trig	CSC trig DT trig	1/2 1/2	1
Total	30 (636)	46	46

A data source can be in one of six possible states: Ready, Busy, Out-of-sync, Warning-Overflow, Failure, Disconnected (see figure 2). At startup, the device is in the Ready state. When the trigger rate is too high, after some time which depends on its buffering capacity, a device will go in the Warning-Overflow state, indicating to the Trigger Control System (TCS) that the rate is too high. If the trigger rate is not reduced, the device will enter the Busy state indicating that any further triggers will be lost leading to a loss of data and a possible loss of synchronization (i.e. event counters located in the device will no longer be synchronized with the event counters located in the global trigger logic). Some devices are designed such that they can drop the data for a specific event but still stay synchronized. A device that reaches the Out-of-sync state remains there until a resynchronization procedure is performed.

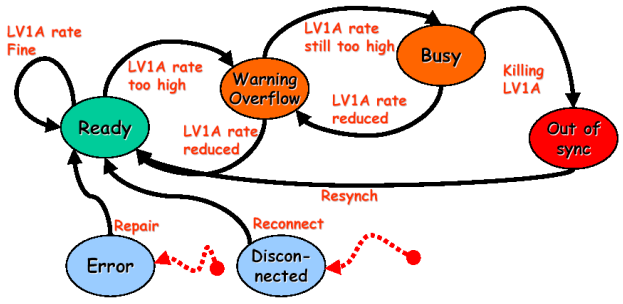

Figure 2: Data source state transition diagram

II. FMM REQUIREMENTS

The FMM requirements remain unchanged: the second prototype implements all of them whereas the first one could not make it due to unsufficient hardware resources (requirements in italic caracters were not implemented):

- the FMM elaborates the detector partition state from the device state
- *the FMM monitors, in real-time, the dead time generated by each device*
- the FMM stores, in a history memory, any state changes along with a time-tag for monitoring/debugging purposes
- each input can be masked in the computation of the partition state.
- *pattern-injection logic allows to test in-situ the FMM behaviour without data sources to be connected*
- *the FMM acts also as an output interface for the aTTS system*

The output state of an FMM is computed from the inputs and the associated merging function. The merging function depends on the state to which it is applied. Currently, two functions are used:

- a logical OR
- an arithmetic sum followed by a variable threshold

The logical OR is used when one device in a given state is enough to set the whole partition in the same state. For example, when a device of a partition is busy, the entire partition is declared to be busy.

The arithmetic sum combined with a threshold is used when an action is required (i.e resynchronization procedure) only when more than one device requires it. For example, it would be inappropriate to resynchronize a partition when a single device is out-of-sync. These merging functions can be changed on request since they are implemented in a Field Programmable Gate Array (FPGA).

III. FMM PROTOTYPE IMPLEMENTATION

It is mainly in the implementation that the second prototype differs from the first one:

- the FMM can now be configured as a single unit dealing with 20 inputs or as 2 independent units of 10 inputs each. With 32 inputs, the previous FMM was well adapted to large partitions (i.e. tracker) but under used for all other partitions (see table 1)
- the form factor is a Compact PCI 6U double width. A compact PCI crate can hold up to 8 FMMs.

The logic of the FMM (shown in Figure 3) is implemented in an FPGA from the Xilinx Virtex II pro family. External to the FPGA are only the Input/Output connectors, the history memory and the PCI interface chip.

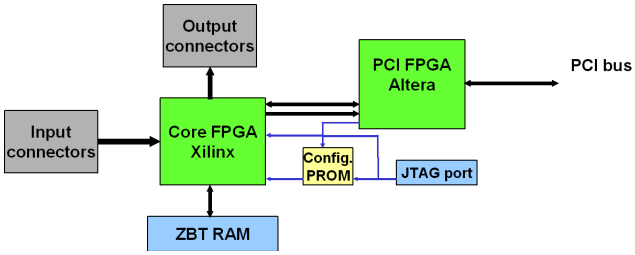

Figure 3: FMM block diagram

A. Input/Output connector

The connector is a standard RJ-45 network connector chosen for its low-cost and high reliability. The pinout of the connector is such that standard ethernet network cables can be used to connect a device with the FMM. The signaling level on the cable is LVDS. Built-in indicators allow to read directly the status of the attached device.(see Figure 4). The PCB is designed such that the connector is configured to be an output or input at soldering time.

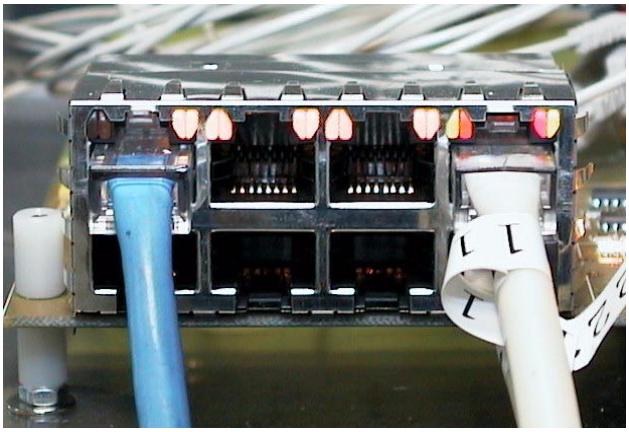

Figure 4: I/O connector with status indicators

B. History memory management

The history memory uses ZBTRAM components (Zero Bus Turnaround RAM). To optimize the usage of memory space, only state changes are written into the memory along with a time tag. A module in the FPGA continuously monitors the 80 input bits (20 devices giving each 4 bits) pre-sampled by a 80 MHz clock and re-sampled at 40 MHz. At a given point in time, if a 80-bit sample differs from the previous one, the current 80-bit status is written into the memory along with the 40-bit time tag. So for each state change, 4 32-bit words are written in burst mode into the history memory. The current memory on the FMM is 2 MB or 128.000 transitions. The time tag resolution is 25 ns: the delay before overwriting the time tag is ~ 7.6 hours.

In nominal running conditions, a typical transition rate of a few Hz per input is expected. Table 2 shows the generated bandwidth versus input transition rate.

Table 2: Bandwidth to the history memory

Transition rate (all inputs)	History length per MB	Data rate to memory
10 Hz	1.8 hour	160 bytes/sec
100 Hz	~11 minutes	1.5 kB/sec
1 kHz (worries…)	65 seconds	15 kB/sec
10 KHz (pathologic…)	6.5 seconds	156 kB/sec
100 kHz (Very pathologic!)	.65 seconds	1.5 MB/sec

Given the form factor of the FMM, a compact PCI crate will house 8 cards: with a pathologic transition rate of 100 kHz, a total bandwidth of 12 MB/sec is generated per crate. This amount of data is easily accommodated by the PCI bandwidth to the host PC.

An alternative working mode allows the data generated by a transition to be available directly to the PCI interface chip that will in turn initiate a DMA to the host PC.

C. Core FPGA logic

The FPGA block diagram is shown in Figure 5. From the input states, the FPGA computes the output state according to the merging functions. It detects any state changes and fills up the history memory. Before actually writing into the external ZBTRAM, the states and the time-tag are first written to a small internal FIFO queue of 15 events deep. The FIFO queue is emptied to the ZBTRAM if no concurrent access from the control interface is in progress. This FIFO is also useful as a buffer when bursty state transitions occur. Deadtime monitors are also implemented in the core FPGA. Deadtime is introduced when a data source is either in the Busy or Warning overflow state. 32-bit counters are monitoring these 2 states for every data source (i.e. 40 counters) with a resolution of 25 ns per count. For each data source, the two counted values indicate the duration spent in these two states. The monitoring software is using these values to compute the deadtime generated by each data source.

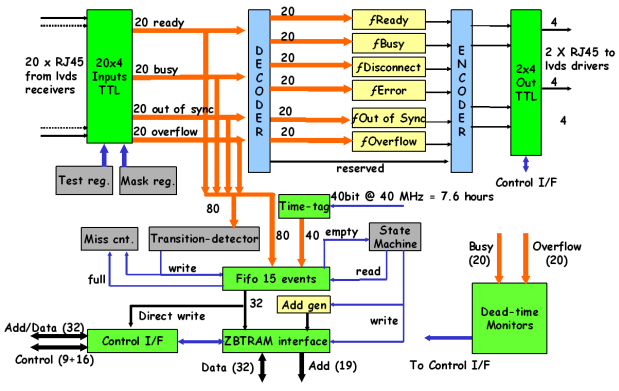

Figure 5: FPGA block diagram

D. *Control interface*

The control interface allows the user interaction with the FMM: configuration of the internal registers (mask, threshold, control/status), readout of the history memory, access to the deadtime monitors. The control interface is based on a design already in use for the Front-End Readout Link card (FRL) [3].

E. *Control and test software*

Control and test softwares are currently under development. As the FMM and the FRL boards appear very similar from the PCI point of view, the FMM profits from the work done for the FRL for its control during physics runs as well as for the production test.

F. *Performances*

The computation of the ouput state is pipelined at 40 MHz with an initial latency of 4 clock cycles (100 ns).

The PCI interface chip exploits the full PCI bandwidth (132 MB/sec). The local bus between the core FPGA and the PCI interface chip is running at 80 MHz: a single write is performed in 6 clock cycles (75 ns), a single read is performed in 3 clock cycles (37.5 ns).

IV. PLANS

After full validation of the prototype, a production of ~60 PCBs will be launched before the end of the year. Installation in the CMS experimental area will take place in the second half of 2005.

Figure 6: Prototype picture

V. SUMMARY

After summarizing the principle behind the operation of the TTS, the second and final prototype has been described in depth. This module is a CompactPCI 6U board. It can be configured to manage 20 data sources of the same detector partition or two groups of 10 data sources from two different detector partitions. The main functions of the FMM are implemented in a single FPGA whereas the PCI interface is implemented in a second FPGA. Details of the design have been presented.

VI. REFERENCES

[1] The Fast Merging Module (FMM) for readout status processing in CMS DAQ, A. Racz Proceedings of the ninth Workshop on electronics for LHC experiments, Amsterdam, 29 September-3 October 2003.

[2] Trigger Throttling System for CMS DAQ, A. Racz Proceedings of the sixth Workshop on electronics for LHC experiments, Cracow, 11-15 September 2000.

[3] CMS: The TriDAS project Technical Design Report Volume I, Volume II