EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
CERN — AB DEPARTMENT

CERN-AB-2007-010 BI

An FPGA Based Implementation for Real-Time Processing of the LHC Beam Loss Monitoring System's Data

B. Dehning, E. Effinger, J. Emery, G. Ferioli, C. Zamantzas.

CERN – Geneva - Switzerland

Abstract

The strategy for machine protection and quench prevention of the Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) is mainly based on the Beam Loss Monitoring (BLM) system. At each turn, there will be several thousands of data to record and process in order to decide if the beams should be permitted to continue circulating or their safe extraction is necessary to be triggered. The processing involves a proper analysis of the loss pattern in time and for the decision the energy of the beam needs to be accounted. This complexity needs to be minimized by all means to maximize the reliability of the BLM system and allow a feasible implementation.

In this paper, a field programmable gate array (FPGA) based implementation is explored for the real-time processing of the LHC BLM data. It gives emphasis on the highly efficient Successive Running Sums (SRS) technique used that allows many and long integration periods to be maintained for each detector's data with relatively small length shift registers that can be built around the embedded memory blocks.

Presented at IEEE NSS 2006 – Oct. 29 / Nov. 4 2006 – San Diego/USA

Geneva, Switzerland
March, 2007

An FPGA Based Implementation for Real-Time Processing of the LHC Beam Loss Monitoring System's Data

Christos Zamantzas, Bernd Dehning, Ewald Effinger, Jonathan Emery, Gianfranco Ferioli

***Abstract*–The strategy for machine protection and quench prevention of the Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) is mainly based on the Beam Loss Monitoring (BLM) system. At each turn, there will be several thousands of data to record and process in order to decide if the beams should be permitted to continue circulating or their safe extraction is necessary to be triggered. The processing involves a proper analysis of the loss pattern in time and for the decision the energy of the beam needs to be accounted. This complexity needs to be minimized by all means to maximize the reliability of the BLM system and allow a feasible implementation.**

In this paper, a field programmable gate array (FPGA) based implementation is explored for the real-time processing of the LHC BLM data. It gives emphasis on the highly efficient Successive Running Sums (SRS) technique used that allows many and long integration periods to be maintained for each detector's data with relatively small length shift registers that can be built around the embedded memory blocks.

I. Introduction

THE strategy for machine protection and quench prevention of the Large Hadron Collider (LHC) at the European Organisation for Nuclear Research (CERN) is mainly based on the Beam Loss Monitoring (BLM) system. At each turn, there will be several thousands of data to record and process in order to decide if the beams should be permitted to continue circulating or their safe extraction is necessary to be triggered. The processing involves a proper analysis of the loss pattern in time and for the decision the energy of the beam needs to be accounted. This complexity needs to be minimized by all means to maximize the reliability of the BLM system and allow a feasible implementation.

Processing data in real-time requires dedicated hardware to meet demanding time or space requirements where performance is often limited by the processing capability of the chosen technology. To overcome such a limitation, as a first step, the BLM system is making use of modern field programmable gate arrays (FPGAs), which include the resources needed to design complex processing and can be reprogrammed making them ideal for future upgrades or system specification changes. Consecutively, a great effort has been committed to provide a highly efficient, reliable and feasible implementation of the real-time processing by employing various digital techniques and optimizing across all of its levels of abstraction.

Manuscript received November 19, 2006.

All authors are with CERN, CH-1211 Geneva 23, Switzerland (e-mails: firstname.lastname@cern.ch).

C. Zamantzas is the corresponding author (telephone: +41 22 767 3409, e-mail: christos.zamantzas@cern.ch).

II. BLM System Overview

Around 4000 *Ionization Chambers* are the detectors of the system. Tunnel cards, called *BLECFs* [1], acquire and digitize the data from the detectors and transmit those at the surface using *Gigabit Optical Links (GOL)* [2]. There, the data processing cards, named *BLETCs* [3], receive those data and decide whether or not the beam should be permitted to be injected or continue circulating.

Each surface card receives data from two tunnel cards, which means that it can treat up to 16 channels simultaneously. In addition, it provides data to the Logging, the Post Mortem and the Collimation systems that will be used to drive on-line displays in the control room, to allow off-line analysis of the losses and to setup automatically the collimators.

III. Surface FPGA's Processes

Between the blocks responsible for the correct reception and the comparison with the relevant for the channel and the beam energy threshold values lays the BLM's real-time data processing block. Fig. 1 shows a block diagram of the processes assigned in the FPGA and a more detailed explanation of its major parts follows.

A. Receive, Check and Compare (RCC)

The RCC process is part of the effort to provide very reliable implementations of the physical and data link layers for the BLM system. Because of the radiation environment in the tunnel, the evaluation of the detector signal has to be performed in the surface buildings. This leads to long transmission distances of up to 2 km between the front-end in the tunnel and the processing module on the surface. The link operates in the gigabit region to provide low system latency and it is using radiation tolerant devices for the parts residing in the tunnel installation.

This reception process, i.e. the RCC, is facilitated in the entry stage of the surface FPGA and its implementation has been done in a way that ensures the correct reception and detection of erroneous transmissions by redundancy in the

transmission and by using digital techniques like the Cyclic Redundancy Check (CRC) [4] and the 8B/10B [5] algorithms.

In addition, a significant portion of the transmitted packet is occupied with extra information which is used by this process to monitor constantly the correct operation of the tunnel installation.

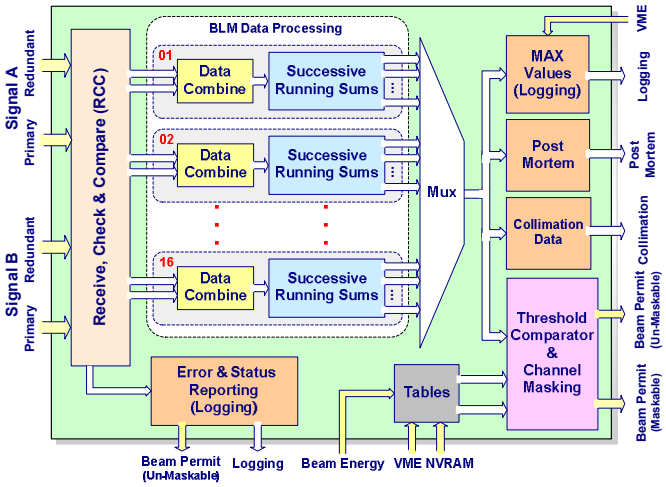

Fig. 1. Block Diagram of the processes assigned in the BLM system's surface installation FPGA.

B. *Data processing*

The proton loss initiated quench of magnets is depending on the loss duration and on the beam energy. Given the tolerance acceptable for quench prevention given by the specifications, the quench threshold versus loss time curve has been approximated with a minimum number of steps fulfilling the tolerance. That has resulted into reducing the number of sliding integration windows to twelve.

In the configuration chosen, each processing module of the system is able to treat 16 channels in parallel and maintain 12 integration periods for each of them spanning in various lengths with the smallest starting from 40μs and the longest reaching up to 84s.

Moreover, the system in order to achieve the nine orders of dynamic range requested by the specifications it is making use of both Current-to-Frequency Converter (CFC) and ADC circuitries for the acquisition and the processing module needs to merge those data subsequently.

Both of these parts are discussed in more detail on the following sections IV and V.

C. *Threshold Comparator and Masking*

The Running Sums, after every new calculation, need to be compared with their corresponding threshold values that were chosen by the beam energy reading given that moment. If on any of them the level is found to be higher, the comparator will initiate the necessary beam dump request.

All dump requests will initially be gathered by a Masking process with the main purpose of distinguishing between "*Maskable*", "*Un-Maskable*" and unconnected channels. Consequently they will be forwarded to the Beam Interlock System (BIS) [6], which will initiate the beam dump. The operators from control room will have the ability to inhibit some of the used channels, i.e. the "*Maskable*", under specific and strict conditions. At the same time, highly critical channels will not be possible under any circumstance to be disabled.

The proposed implementation of a quench level threshold comparator that allows also the possibility of channel masking is using unique tables for each detector to provide the threshold values depending on the beam energy reading.

In order to minimize that table and the memory needed to be stored, instead of using a global table the load will be spread between the processing modules. Thus, a unique block of values will be created for each of the cards of the complete system. The information included will be still for its moving windows, but for less beam energy levels and the specific detectors, each card is reading.

More specifically, on each card it was shown that 12 Running Sums for each of the 16 detector channels will be calculated. The beam energy information will be scaled into 32 levels (0.45 to 7 TeV) and each processing module will hold data only for those 16 detectors connected. That would give a total of 6,144 threshold values (i.e. 32 KB of data) needed to be held on each card.

D. *Logging and Post-Mortem*

In the LHC, storage of the loss measurements are needed to allow to trace back the loss signal developments as well as the origin of the beam losses in conjunction with other particle beam observation systems. Such data will be sent over the VME-bus for on-line viewing and storage by the Logging and Post-Mortem systems.

For supervision, the BLM system will drive an online event display and write extensive online logging at a rate of 1 Hz. The data available for this purpose will include the error and status information recorded by the tunnel electronics and the RCC process as well as the maximum loss rates seen by the running sums (in the last second) together with their corresponding quench level thresholds for the given beam energy. The Logging system will be able to normalize the loss rates with respect to their quench levels before displaying them so that abnormal or high local rates can thereby be spotted easily.

Additionally, there are two types of post-mortem data available from the system for more detailed offline analysis. Those will be, the acquired data (40 μs samples) from the last 20,000 turns, i.e. the last 1.75 seconds, and 82 ms summed values of the acquired data for the last 45 minutes.

E. *Collimation*

Finally, for the Collimation system and to support the correct alignment and setup of the collimators one more set of data is available. Those data contain whenever requested the losses seen on the last 81.28 ms organized in the form of 32 consecutive 2.54 ms sums of the acquired data for each detector.

IV. CFC & ADC Data Merging Algorithm

The Data Combine process will receive the two types of data, the counter and the ADC data, coming from the same detector and will merge them into one value, filtering at the same time noise passing through the ADC circuitry.

On the beginning stage, the ADC value is normalized by its effective range. The min and max of the ADC values received are continuously calculated. Their difference signifies the effective range of the ADC circuitry and is used to normalize each received value. (see Fig. 2)

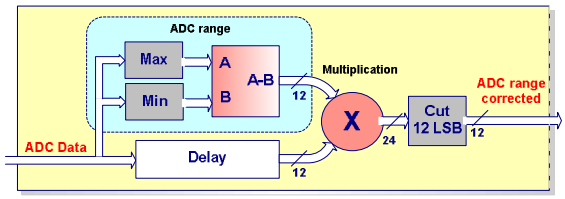

Fig. 2. Block diagram of the Data Combine process (first part). The ADC data normalizer part of the process operates by calculating the operating range of the ADC circuit and consecutively multiplies this as a normalization factor to the ADC value received. The multiplier is making use of the embedded DSP element in the FPGA device.

The two types of data acquired from each detector are of different type and a pre-processing is needed in order those to be combined seamlessly. The measurement of the frequency produced by the CFC with a counter relates to the current accumulated between the last acquisitions. On the other hand, the voltage measured by the ADC is the fraction remained between the last count and the first from the next acquisition.

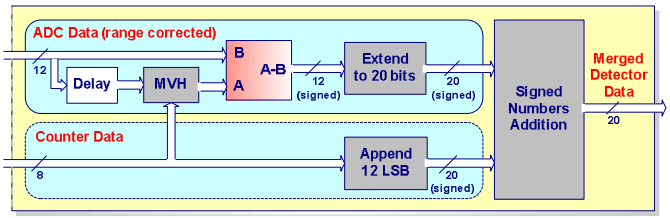

Fig. 3. Block diagram of the Data Combine process (second part). This part of the function outputs a 20 bit value comprising of the CFC and the ADC data. A Minimum-Value-Hold (MVH) block is also added in the ADC data input to filter out various types of noise coming from the acquisition circuit.

In order to merge those data the difference of the last two ADC measurements is needed. It corresponds to the counter fraction of the last 40 μs and thus could be added to the counter value. This could be described in an equation as:

$$V_{Merged}(n) = V_{Counter}(n) + \frac{V_{ADC}(n-1) - V_{ADC}(n)}{2^N} \qquad (1)$$

Where, $V_{Counter}$ and V_{ADC} are the recorded values from the counter and the ADC respectively, and N is the number of bits used from the ADC. The difference is divided by its full scale in order to be normalized as a fraction.

Of course in the implementation, since the difference could be a negative number, signed number arithmetic is used for the addition and in order not to loose in accuracy the values are transformed to 20 bit, which can be considered equal to the multiplication by 2^{12} both parts of Equation 1. (see Fig. 3)

V. Successive Running Sums

The procedure for the data processing, which was chosen to be followed, is based on the idea that a constantly updated moving window can be kept by adding to a register the incoming newest value and subtracting its oldest value. The number of values that are kept under the window, or differently, the difference in time between the newest and the oldest value used, defines the integration time it represents.

A. Basic principles used

A similar configuration for the production of the running sums, but more efficient, would be to delay each incoming new value with a fixed number of cycles by passing them through a shift register and add the difference of the new and the outputted from the shift register to an accumulator. As a result, the depth of the shift register will then signify the integration time of this running sum. (see Fig. 4)

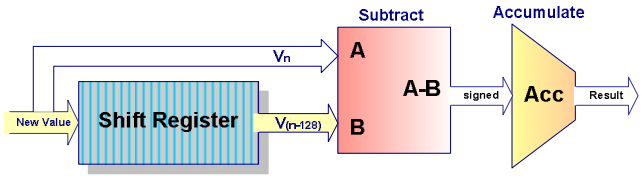

Fig. 4. Block diagram showing an efficient way (w.r.t. speed and resources) to produce and maintain a continuous running sum of arriving values.

Additionally, long histories of the acquired data are needed for the construction of long moving windows. The technique employed to reach long integration periods with relatively small in length shift registers is overcome by consecutive storage of partial sums of the received values.

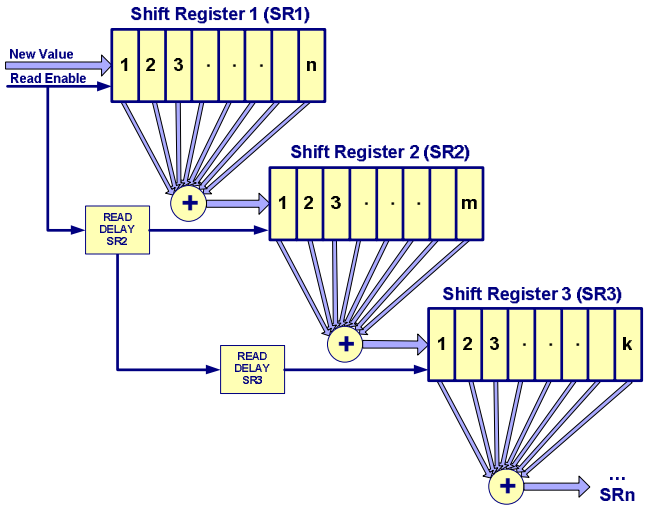

Fig. 5. Block diagram showing a configuration for efficient summation (w.r.t. resources) of many values. Instead of storing all the values needed for the sum, this technique stores successively parts of the total sum using only a fraction of the otherwise needed memory space.

In general, it works by feeding the sum of one shift register's contents, every time its contents become completely updated, to the input of another shift register. By cascading more of these elements it manages to construct very long moving sum windows that overcome the storage problem of preserving long histories of the acquired data. (see Fig. 5)

B. *Optimal Configuration for the BLM system*

Combining those two techniques alone, unfortunately it is not enough to solve all of the difficulties with the needed resources. Nevertheless, by following some more straight forward design rules for the construction the wanted result can be achieved.

For example, by doing such an operation it emerges that the already calculated running sums can be used in order to calculate bigger in length running sums without the need of extra summation points (as proposed in the example before) which translates to a huge data reduction and resource sharing. In the designs realization, the sum of the Shift Register contents is always kept and updated in the running sums. Thus, some of the running sums' outputs are also used directly to feed the following stage's inputs.

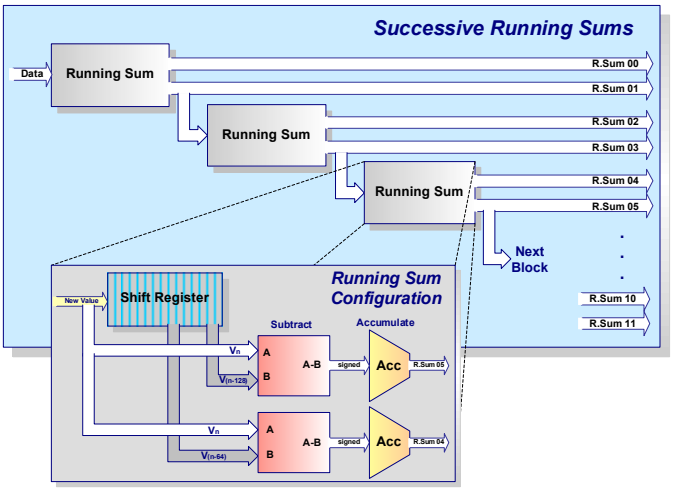

Fig. 6. Block diagram of the Successive Running Sums configuration in the BLM system. The process is making use of successive multipoint shift registers of 64 or 128 values to continuously update and maintain 12 sums with the longest providing a sum of more than 2 million acquired values or differently an integration time of 84 seconds.

One more step is the use of multipoint shift registers. That is, shift registers that are configured to give intermediate outputs, usually referred to as taps. The taps provide data outputs at certain point in the shift register chain. This feature can be effectively employed to combine overlapping memory contents, therefore minimizing even more the resource utilization. (see Fig. 6)

Finally, since the shift registers will be constructed by the FPGA's embedded memory blocks, where the width and depth of the memory block is fixed, any unused memory space will be wasted. If a longer or wider shift register is needed then two or more memory blocks will be combined but no other process can use the memory bits left unused by each shift register implemented. Fig. 7 illustrates an example where the contents needed for each detector are 32 x 8-bit values. If each detector is treated independently, its shift register will occupy one 512 bit memory block. For the same case, if the data from two detectors were pre-combined, the resource usage would drop to half.

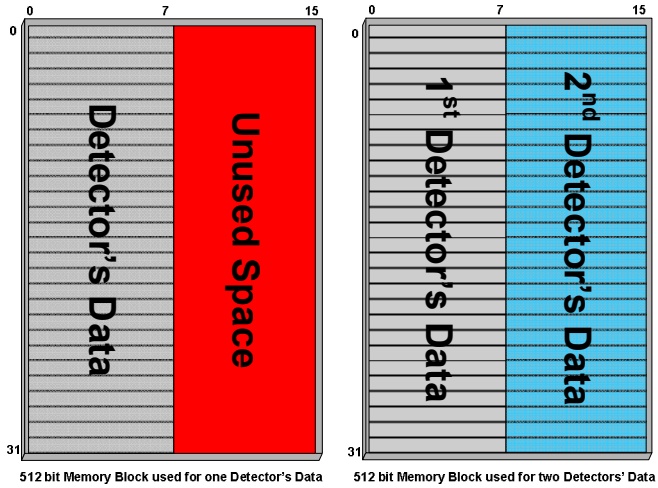

Fig. 7. Example showing the optimization that can be achieved in the usage of the FPGA's embedded memory blocks by the shift registers.

Of course, this example is not always the case and there is not a generic way to discover such optimizations. Probably this is also the reason why none of the synthesis tools available performs such resource sharing. Thus, it was found necessary an investigation to be made to find the optimal configuration and later the results were constrained into the synthesis tool.

VI. Evaluation of the SRS Technique

The optimal achievable latency in the response of each stage in such a system is equal to the refreshing time of the preceding shift register. That is, the time needed to completely update its contents. In Fig. 5, the supervision circuit, denoted as *"Read Delay"*, which is making sure that the sum is calculated every time with new values, holds a delay equal to this latency to guarantee the correct operation. Thus, the delay is every time equal to the preceding shift register's input clock period multiplied by the elements planned to be used in the sum.

For example (and using the notation of Fig. 5):

$$SR2_{DELAY} = f_{NewValue} * n \quad (2)$$

$$SR3_{DELAY} = SR2_{DELAY} * m \quad (3)$$

Where, $SR2_{DELAY}$, $SR3_{DELAY}$ are the read delays needed for the first and the second shift register respectively, $f_{NewValue}$ is the frequency of the input, and the n, m are the number of elements held in each of the shift registers.

Furthermore, as it can be seen in Table I where it is shown the configuration of the running sums optimized for the LHC's BLM system, the latency introduced has little effect to the optimal approximation accuracy. This is a result from the fact

that it varies between them. More specifically, the running sums that span to the low range (fast losses) have zero or very small additional latency. The latency gradually increases as the integration time increases, reaching up to 0.65 seconds for the 21 and 84 seconds integration time range.

TABLE I
SUCCESSIVE RUNNING SUMS CONFIGURATION FOR THE BLM SYSTEM.

Range		Refreshing		Shift Register Name	Running Sum Name
40 µs steps	Ms	40 µs steps	ms		
1	0.04	1	0.04		RS00
2	0.08	1	0.04		RS01
8	0.32	1	0.04	SR1	RS02
16	0.64	1	0.04		RS03
64	2.56	2	0.08	SR2	RS04
256	10.24	2	0.08		RS05
2048	81.92	64	2.56	SR3	RS06
16384	655.36	64	2.56		RS07
32768	1310.72	2048	81.92	SR4	RS08
131072	5242.88	2048	81.92		RS09
524288	20971.52	32768	655.36	SR5	RS10
2097152	83886.08	32768	655.36		RS11

The red colored running sums (*RSxx*) outputs, i.e. *RS01*, *RS04*, *RS06*, and *RS07*, represent their additional utilization as the inputs for adjacent shift registers (*SRxx*), i.e. *SR2*, *SR3*, *SR4*, and *SR5*.

Finally, by cascading just five of these elements, holding only 64 or 128 values each, it is enough to reach the 100-second upper integration limit requested by the specifications. This gained efficiency was necessary for this system to be applicable in a configuration with relatively very low memory available. In a different configuration of this system, where only Running Sums would be used, the shift registers would need to hold approximately 3 million values for each of the 16 detectors to achieve the same approximation error which translates to a total of approximately 150 MB of memory space. Instead, by using the Successive Running Sum technique the system is using only some of the FPGA's internal memory since it does not need more than 100 KB of memory space.

ACKNOWLEDGMENT

The authors would like to thank Stephen Jackson for implementing the server application and the Expert GUI for the visualization of the data collected by the processing modules, and Daryl Bishop and Graham Waters at TRIUMF for building and supporting the DAB64x module.

REFERENCES

[1] E. Effinger, B. Dehning, J. Emery, G. Ferioli, G. Gauglio, C. Zamantzas, *"The LHC Beam Loss Monitoring System's Data Acquisition Card"*, 12th Workshop on Electronics for LHC and future Experiments (LECC 06), Valencia, Spain.

[2] P. Moreira, G. Cervelli, J. Christiansen, F. Faccio, A. Kluge, A. Marchioro, T. Toifl, J. P. Cachemiche and M. Menouni *"A Radiation Tolerant Gigabit Serializer for LHC Data Transmission"*, Proceedings of the Seventh Workshop on Electronics for LHC Experiments (DIPAC), Stockholm, Sweden, 10-14 September 2001

[3] C. Zamantzas, E. Effinger, B. Dehning, J. Emery, G. Ferioli, *"The LHC Beam Loss Monitoring System's Surface Building Installation"*, 12th Workshop on Electronics for LHC and future Experiments (LECC 06), Valencia, Spain

[4] Rajesh Nair, Gerry Ryan and Farivar Farzaneh, "A Symbol Based Algorithm for Hardware Implementation of Cyclic Redundancy Check (CRC)", 1997 VHDL International User's Forum (VIUF '97), p. 82.

[5] A.X.Widmer, P.A.Franaszek, *"A DC-balanced, partitioned-block, 8B/10B transmission code"*, IBM Journal of Research and Development, vol 27, no 5, 1983, pp. 440-451.

[6] Engineering Specification, *"The Beam Interlock System For The LHC"*, LHC Project Document No. LHC-CIB-ES-0001-00-10, version 1.0, 17-02-2005.