CERN

ALICE Internal Note/DAQ
ALICE-INT-2002-01
Date of last change
January 29, 2002

Specification and Simulation of the ALICE DAQ System

Authors:
G. Di Marzo Serugendo, CERN-IT
P. Jovanovic, School of Physics and Astronomy, University of Birmingham
P. Vande Vyvre, CERN-EP
O. Villalobos Baillie, School of Physics and Astronomy, University of Birmingham

for the ALICE collaboration

Abstract:
The Trigger and Data Acquisition (DAQ) System of the ALICE experiment has been designed to support the high bandwidth expected during the LHC heavy ion run. A model of the ALICE DAQ has been developed. The goal of this model is twofold. First, it establishes the functional behaviour of the whole system and its sub-systems, the input/output interfaces among sub-systems, identifies the parameters and their values. Second, it allows to verify that the system-level design is consistent and behaves according to the requirements. In addition, it is used to evaluate the theoretical system performances using the measurements done on sub-systems prototypes. This report presents the formal specification of this model, realised using a commercial tool called Foresight, and the performances reached by the model during the simulation. In addition, this report gives some technical details on the efficiency of the simulation.

Contact Person: Giovanna Di Marzo Serugendo Giovanna.Di.Marzo@cern.ch
Phone: +41 22 767 41 37/Fax: + 41 22 767 71 55 Creation Date: January 2001

Distribution lists
For Approval:
For Information:
EDMS Number:

Contents

1 Introduction

The ALICE Trigger and Data Acquisition System (DAQ) is required to support an aggregate event building bandwidth of up to 4GB/s (Gbytes/s) and a storage capability of up to 1.25GB/s to mass storage. The system must also be able to combine different types of physics events: slow rates of Central and Minbias triggers generating the largest fraction of the total data volume, together with faster rates of Dielectron and Dimuon events.

The ALICE DAQ system has been decomposed in a set of hardware and software components. The detailed system design is going on in parallel with the development of prototypes of these components. We wish to verify this design in order to check that it can reach the expected behaviour and the target performances.

However, such a complex system happens to be difficult to verify manually, since there is no corresponding mathematical description. A tool that enables one to define a model of the whole system, and to perform its verification is therefore an extremely valuable help.

This report presents the formal specification and simulation of the DAQ of the ALICE experiment. A modelling and simulation tool, called Foresight, is used to specify a system in an abstract manner (at system-level) in order to focus on the functionality.

2 Foresight

A Foresight specification [3] is made of hierarchical data flow diagrams, finite state diagrams, and pieces of a procedural modelling language. The specification provides a unambiguous description of the system. The semantics of the specification provides a model of the system whose behaviour is very close to the behaviour of the system. The verification process is performed during the simulation. It demonstrates the functional correctness of the system.

The Foresight simulation consists of the real-time execution of the specification. It offers debugging functions like animation of diagrams, breakpoints, and monitor windows. The simulation is used to evaluate the performances of the specified system. It also makes it possible to perform some analysis such as the system sensitivity to some key parameters. One can also explore other algorithms, and new architectures. This is useful when the final architecture has not yet been defined (as is the case for ALICE), since it helps to compare architectures or choices of implementation.

3 Specification Overview

The current ALICE specification describes the *functionality* of the whole experiment and of the major sub-systems: Trigger, Trigger Detectors, Tracking Detectors, DAQ, Permanent Data Storage. The specification follows the description of ALICE DAQ given in [5], using up-to-date parameters values.

3.1 Overall System

The top-level Foresight specification of the ALICE DAQ system is made of the data flow diagram of Figure 1. Sub-systems are specified with Foresight processes. Links between sub-systems stand for the input/output interfaces among sub-systems.

The Interaction Source element generates events at a rate of 6000 Hz distributed according to a Poisson distribution. Trigger detectors receive the event signal emitted by the Interaction Source and inform the Trigger System. The latter emits L0, L1, L2 signals towards the Tracking Detectors. According to the received signals from the trigger system, the tracking detectors send data to the DAQ Sys (DAQ subsystem), and emit busy signals to the Trigger System. The DAQ Sys performs the sub-event building, then the event building and finally sends the events to the PDS Sys (Permanent Data Storage).

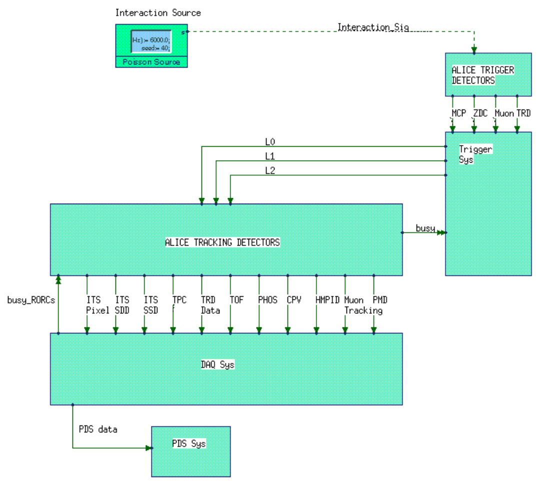


Figure 1: Overall Architecture

3.2 Trigger System

The ALICE trigger system has three levels: level 0 (L0), level 1 (L1), and level 2 (L2).

- L0 serves to strobe the detectors at $t_0 + 1.2\mu s$, where t_0 is the event time. L0 performs past/future (P/F) protection at t_0, i.e., no other interaction must have taken place during a given time interval before and after the occurrence of the current event. The time interval depends on the type of the event. At L0, we have simulated the case where an event can be either Central, Dimuon, Dielectron, Minbias, or a combination of two or more of these types. In the case of a combined event, two or more classes of detectors may be strobed. A class of detectors is the set of detectors involved in an event of a given type. L0 checks the busy flag of the detectors concerned by the current event. If one of them is busy, the classes of detectors to which this detector belongs are not strobed. If no class of detector can be strobed, then L0 does not send any signal to the detectors. Otherwise, the remaining classes of detectors receive the L0 signal;
- L1 performs a new P/F protection for the period up to $t_0 + 4.3\mu s$. If the event can be accepted as it is, the corresponding classes of detectors are informed at $t_0 + 5.5\mu s$. The trigger numbers are distributed by L1. In the case of a combined event, the P/F protection may act as a filter, so that only some classes of detectors are allowed to continue with the event. In this case, only some classes of detectors receive an L1 accept. If no L1 is generated, the event is rejected;
- Following the L1, level L2 makes the final trigger decision (no more combinations). It performs a P/F protection for the period up to $t_0 + 88\mu s$, and detectors are informed at $t_0 + 89.2\mu s$. The P/F protection may remove some classes of detectors. If after P/F protection, the final trigger outcome still includes more than one class, one of the participating classes is chosen proportionally to the re-

quired DAQ rates. The remaining class of detectors receives an L2accept signal, the others receive an L2reject signal;

- L0, L1, L2 signals always arrive in this order. It may happen that L0', L1', for a consecutive event, arrive before L2 (i.e., the sequence L0 L1 L0' L1' L2 is allowed). However L2 arrives always before L2', since level 2 signals reach the detectors at $t_0 + 89.2\mu s$.

3.2.1 P/F protection Intervals

For all trigger types, no other interaction can take place during the P/F protection interval $[t_0 - \Delta, t_0 + \Delta]$, where Δ is the P/F protection time. The parameters used for the P/F protection intervals are the following:

Dimuon:	$\Delta = 3\mu s$
Dielectron:	$\Delta = 7\mu s$
Combination(Dimuon,Dielectron):	$\Delta = 7\mu s$
Central, Minbias, Other combinations:	$\Delta = 88\mu s$.

The results of this report have all been computed with the above P/F protection intervals. Although TPC is involved in Dielectron events, at the time of establishment of the P/F protection interval, the intervals for Dielectron events, and the Combination(Dimuon,Dielectron) have been estimated to be $7\mu s$. More recently, it appears that an interval of $88\mu s$ could be more appropriate for Dielectron, and for the Combination(Dimuon,Dielectron). Therefore, it is worth noting that for intervals of $88\mu s$, results would vary from those presented here.

3.2.2 Event Rates at L0 Input

The event rates produced by the collisions and received at the L0 level have been estimated as follows. The total number of interactions that have to be taken into account for P/F protection is set to 6000 Hz. Only 4000 Hz are interesting at the physics level (physics events). All interactions participate in the P/F protection process, i.e., their occurrence may spoil another event.

Table 1 shows the distribution of the 4000 Hz of interesting physics events among the different classes of triggers. For instance, the rate of Central events (row "C") has been estimated to be 1000 Hz, distributed in the following manner: 403 Hz are "pure" Central events, 272 Hz are a combination of Central and Dimuon events, 272 Hz a combination of Central and Dielectron events, and 53 Hz are given by the combination of Central, Dimuon and Dielectron events. Minbias events do not combine with other events.

Table 1: Event Rates at L0 Input

	C	DM	DIEL	All	Total
C	**403**	272	272	53	1000 Hz
DM	**272**	**272**	53	53	650 Hz
DIEL	**272**	**53**	**272**	**53**	650 Hz
MB	**2403**				2403 Hz
Sub-Total					**4000** Hz
Others					**2000** Hz
Total					**6000** Hz

3.3 Tracking Detectors

The functional behaviour of the tracking detectors has been specified in the following manner:

- As soon as it receives a L0 signal, a detector becomes "busy". The trigger system does not send another L0 signal to a busy detector;
- The detectors wait for a L1 signal that has to arrive at $t_0 + 5.5\mu s$. If there is no L1 signal, then the detectors reset the collected data, and become available (not busy). If the L1 signal arrives, the detectors remain busy until the end of the reading period, unless the multi-event buffer is full (all event slots occupied);
- The data are stored in a multi-event buffer (one buffer for each DDL). In the specification, every detector has a multi-event buffer of 4 positions;
- Following the L1, the detectors wait for the L2 decision. If it is an L2reject, the data are discarded and a slot is freed in the multi-event buffer. If it is an L2accept, the data are transferred to the DAQ over the corresponding DDLs. Multi-event buffers are FIFO queues: data are sent or discarded in the order they were received - no "event overtaking" is allowed.

3.4 DAQ Sub-system

Figure 2 depicts the schematic view of the DAQ sub-system, showing the case of a detector having 20 DDLs. The elements in this figure are described below:

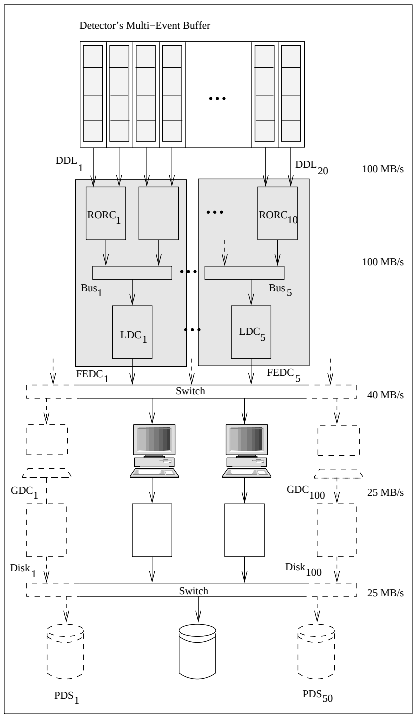


Figure 2: DAQ Sub-System

- A *Detector Data Link (DDL)* transfers the data from a buffer of the detector to the DAQ, at the rate of 100MB/s. The data arrive in a Read-out Receiver Card (RORC). Several DDLs may be connected to the same RORC. In the example, two DDLs are connected to each RORC. Data are sent independently and in parallel along every DDL. If the corresponding RORC is full, the buffer slot cannot be released;

- A *Read-Out Receiver Card (RORC)* stores the data received from the DDL. A bus of 100MB/s is used to transfer the data from a RORC to a Local Data Concentrator (LDC). Several RORCs can be plugged in the same LDC (two RORCs, in the example). Data are transferred to the LDC in the received order. The capacity of a RORC is set to 12MB. If a LDC is full, the corresponding RORCs are blocked;
- The *bus* transports the data from RORCs to an LDC. The bus is connected at most to one LDC. It sends data sequentially to the LDC. Therefore, a RORC, which wants to send data, has to wait upon the completion of the sending of the data of the preceding RORC;
- A *Local Data Concentrator (LDC)* is responsible for sub-event building. As soon as a LDC receives all the data corresponding to an event, it builds the sub-event and sends it to a given Global Data Collector (GDC). In the example, every LDC receives four pieces of data (two from each connected RORC) before sending it to the GDC. Sub-events are sent in chronological order to the GDC. The capacity of a LDC is set to 128MB. Every LDC sends the data of a given event to the same GDC. The choice of the GDC for a trigger event number n is: n MOD 100 (where 100 is the number of GDCs). Sub-events are sent to a Switch that forwards the sub-event to the GDC at a rate of 40MB/s. If the corresponding GDC is full, the LDC is blocked. All LDCs and GDCs are linked to the same Switch;
- The *Front-End Digital Crate (FEDC)* is the set made of one LDC, the bus and the connected RORCs;
- A *Global Data Collector (GDC)* is responsible for the event building. It waits for all LDCs to send the sub-events. Once the event is complete, the whole event is stored on a disk. Events are sent in chronological order. Each GDC has its own disk. The link between the GDC and the disk has a rate of 25MB/s. The GDC capacity is 512MB;
- There is one *Disk* for each GDC, used for storing events. As soon as a file of 1GB is built on the disk, the whole file is sent to an available Permanent Data Storage (PDS). Files are sent to a Switch that forwards the data to the chosen PDS, at a rate of 25MB/s. All disks and PDSs are linked to the same Switch;
- A *Permanent Data Storage (PDS)* receives files of 1GB from the DAQ. A PDS is considered to be an infinite buffer. There are 50 PDSs.

3.5 Parameters

In order to obtain a Foresight specification which can be easily adapted to requirements for changes of the ALICE DAQ, and which can be tested under different conditions, the Foresight specification has been heavily parameterized. The parameters values are taken from [4, 5].

3.5.1 Detectors Parameters

Table 2 lists the parameters specific to each detector.

The “Buffer” parameter stands for the size of the multi-event buffer. It is set to 4 event slots for every detectors. The “Read Time” is the time necessary to read an event. It starts at t_0, the time when the interaction occurs. The “Reset Time” is the time implied by a detector to reset the collected data and to become newly available after a L1 signal has not been received. This time has been set to $0.1\mu s$ for every detector. The “Throw Time” is the time necessary to free a slot in the multi-event buffer in case of a L2reject. It is set to $1.0\mu s$ for every detector.

Table 2: Detectors Parameters

	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer	4	4	4	4	4	4
Read Time	100μs	50μs	55μs	50μs	50μs	50μs
Reset Time	0.1μs	0.1μs	0.1μs	0.1μs	0.1μs	0.1μs
Throw Time	1μs	1μs	1μs	1μs	1μs	1μs
Trans Rate	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s
C (max)	75.9MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
C (min)	56.1MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
DM (max)	0.0MB	0.0MB	0.15MB	0.0MB	0.0MB	0.0MB
DM (min)	0.0MB	0.0MB	0.15MB	0.0MB	0.0MB	0.0MB
MB (max)	75.9MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
MB (min)	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB
DIEL (max)	2.11MB	0.18MB	0.15MB	0.0MB	0.0MB	0.0MB
DIEL (min)	1.56MB	0.18MB	0.15MB	0.0MB	0.0MB	0.0MB
DDL	180	18	28	20	20	20
RORC	180	18	14	10	10	10
LDC	180	18	5	3	3	3
FEDC	180	18	5	3	3	3
Min RORC Size	0.4216MB	0.4444MB	0.00535MB	0.009MB	0.006MB	0.006MB
Min LDC Size	0.4216MB	0.4444MB	0.0321MB	0.072MB	0.048MB	0.048MB

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	
Buffer	4	4	4	4	4	
Read Time	10μs	50μs	50μs	50μs	50μs	
Reset Time	0.1μs	0.1μs	0.1μs	0.1μs	0.1μs	
Throw Time	1μs	1μs	1μs	1μs	1μs	
Trans Rate	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	
C (max)	0.57MB	1.5MB	0.38MB	0.02MB	0.12MB	
C (min)	0.14MB	1.5MB	0.38MB	0.02MB	0.03MB	
DM (max)	0.57MB	0.0MB	0.0MB	0.02MB	0.12MB	
DM (min)	0.14MB	0.0MB	0.0MB	0.02MB	0.03MB	
MB (max)	0.57MB	1.5MB	0.38MB	0.02MB	0.12MB	
MB (min)	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB	
DIEL (max)	0.57MB	1.5MB	0.38MB	0.02MB	0.12MB	
DIEL (min)	0.14MB	1.5MB	0.38MB	0.02MB	0.03MB	
DDL	20	72	8	4	6	
RORC	10	36	4	2	3	
LDC	10	9	4	1	3	
FEDC	10	9	4	1	3	
Min RORC Size	0.0285MB	0.02083MB	0.0475MB	0.005MB	0.02MB	
Min LDC Size	0.057MB	0.166MB	0.095MB	0.02MB	0.04MB	

The “Trans Rate” stands for the maximum DDL bandwidth allocated to a detector. It is calculated as follows: DDL*100MB/s, where DDL is the number of DDLs associated to the detector, and 100MB/s (MBytes/sec.) is the transmission rate of one DDL.

The event size depends on the type of the event and it varies for each detector. The actual event size is not fixed as it changes for each event. The size is bound by a minimum and maximum value. Parameters “C (max)” and “C (min)” stand respectively for the maximum and minimum event size of an event of Central type. Similarly “DM (max)” and “DM (min)” bound the size of Dimuon events, “MB (max)” and “MB (min)” the size of Minbias events, and “DIEL (max)” and “DIEL (min)” the size of Dielectron events. Except for Minbias events, the size of an event is randomly chosen between the maximum and minimum value, according to a Gaussian whose average is the centre of the interval. In the case of Minbias events, the average of the

Gaussian is the first quarter of the interval (and the interval starts at 0.0MB).

The “DDL” parameter gives the number of DDLs allocated to the detector. Similarly, “RORC”, “LDC” and “FEDC” give respectively the number of RORCs, LDCs and FEDCs of the detector. The number of LDCs is always equal to the number of FEDCs, since there is one LDC in each FEDC.

Parameter “Min RORC Size” stands for the required minimum size that must remain available in a RORC buffer before it is considered full. The size is equal to C(max) / DDL. Indeed, it is equal to the maximum amount of data that a DDL can send to a RORC, i.e, the maximum size of a Central event divided by the number of DDLs. In the case of the Muon detector: Min RORC size = 0.15MB/28 = 0.00535MB.

Parameter “Min LDC Size” stands for the required minimum size that must remain available in a LDC buffer before it is considered full. The size is equal to the maximum size that a sub-event can have, i.e, (Min RORC size) * ([DDL/LDC]+1). Indeed, [DDL/LDC]+1 stands for the maximum number of DDLs “connected” to a LDC (through RORC). In the case of the Muon detector: 28 DDLs are linked to 14 RORCs (2 DDL for each RORCs), the 14 RORCs are then linked to 5 LDCs. This means that 4 LDCs are linked to 3 RORCs, and the last LDC is only linked to 2 RORCs. Since each RORC is linked to 2 DDLs, each LDC is linked to at most 6 DDLs. Therefore, we obtain [DDL/LDC] + 1 = [28/5] +1 = 6. Finally, Min LDC Size = Min RORC size * 6 = 0.00535MB * 6 = 0.0321MB.

3.5.2 DAQ Parameters

Table 3 lists the parameters related to the DAQ system itself.

The “Quantity” row gives the total amount of DDLs (sum of all DDLs of all detectors), RORCs, LDCs, GDCs, and PDSs that appear in the DAQ system.

The “Buffer Size” row shows the respective size of the buffers of one RORC, one LDC, and one GDC. PDSs are supposed to have an infinite buffer (their actual size does not cause any buffering problem). The Buffer Size parameter of RORCs is combined with the Min RORC Size in order to obtain the upper bound of a RORC: if the actual buffer size of a RORC is more than (Buffer Size - Min RORC size), then the RORC is considered full. It is similar for LDCs.

Parameter “Min GDC Size” stands for the required minimum size that must remain available in a GDC buffer before it is considered to be full. The size is equal to maximum event size, i.e, to the maximum size of a Central event, i.e., 87.06MB.

The ”Rate” parameter mentions the transfer rate of each element: DDLs and Buses transfer data at 100MB/s the Switch1 transfers sub-events at 40MB/s, and the storage on disk, as well as the storage on PDS through Switch2 is performed at 25MB/s.

“Sub-event qty” stands for the number of sub-event pieces necessary to have a complete event. It is actually equal to the number of LDCs involved in the event. It depends on the event type: in the case of a Central or Minbias event, all detectors are involved, thus all LDCs have to send a sub-event, then “Sub-event qty” for Central and Minbias events is 239; for Dimuon, only the LDCs of detectors ITSPixel, PHOS, Muon and PMD are necessary, thus “Sub-event qty” is equal to 19; and for Dielectron, this parameter is set to 230.

The “Size (max)” and “Size (min)” fields give the maximum, and the minimum size of the events, respectively. Finally, the “Detectors” row lists the detectors involved in each event according to the event type.

3.5.3 Trigger System Parameters

Parameters of the Trigger System consists of the P/F protection intervals given in subsection 3.2.1; and of the delays implied by levels L0, L1, L2 for performing P/F protection and for informing the detectors. In the case of L0, this level performs a

Table 3: DAQ Parameters

	DDL	RORC	LDC	GDC	Disk	PDS
Quantity	396	297	239	100	100	50
Buffer Size	-	12MB	128MB	512MB	50GB	∞
Min GDC Size	-	-	-	87.06MB	-	-

	DDL	Bus	Switch1	Disk	Switch2	
Rates	100MB/s	100MB/s	40MB/s	25MB/s	25MB/s	

	Central	Dimuon	Dielectron	MinBias		
Sub-event qty	239	19	230	239		
Size (max)	87.06MB	0.57MB	5.03MB	87.06MB		
Size (min)	66.74MB	0.34MB	3.96MB	0.0MB		
Detectors	All	ITSPixel	ITS Pixel	All		
		PHOS	ITSSDD			
		Muon	ITSSSD			
		PMD	TPC			
			TRD			
			PHOS			
			Muon			
			PMD			

P/F protection up to $t_0 + 0.0\mu s$, and informs the detectors at $t_0 + 1.2\mu s$. L1 performs a P/F protection up to $t_0 + 4.3\mu s$, and informs the detectors at $t_0 + 5.5\mu s$. Finally, L2 performs a P/F protection up to $t_0 + 88\mu s$, and informs the detectors at $t_0 + 89.2\mu s$.

Table 4: Trigger Parameters Delays

	L0	L1	L2
Perform P/F	$0.0\mu s$	$4.3\mu s$	$88\ \mu s$
Inform Det.	$1.2\mu s$	$5.5\mu s$	$89.2\mu s$

4 Detailed Specification

The previous section describes the functionality of the model. This section is dedicated to the detailed description of the Foresight model that has been realised. It explains the algorithms used in the Foresight model in order to attain the expected functionality.

4.1 Trigger System

Figure 1 is the top-level data flow diagram of the specification. The “Trigger Sys” process of this diagram is itself a data flow diagram given by Figure 3. It defines the trigger system of the ALICE experiment.

The Foresight “Trigger Sys” process receives input data from the Trigger Detectors (Muon, MCP, TRD, ZDC), and sends output data (L0, L1, L2) to the Tracking detectors. The input data incoming from the Trigger Detectors are Foresight signals sent to Trigger Sys each time an interaction occurs (6000 Hz). These signals cause the Trigger Sys process to compute the occurrence time (t_0) of the interaction, which is then sent to the "Enable Triggers" process. This process maintains the "event_list", a list of occurrence time of interactions used for P/F protection, and informs the L0 trigger that an interaction has occurred at t_0 (through data flow "goL0"). The Enable Triggers process updates the event_list, in order to let it contain only those interaction times that occurred in the interval $[t_0 - 200\mu s, t_0]$. This interval is sufficient to cover the P/F protection interval of the most recent interaction (occurred at t_0), but also the protection intervals of all the interactions that have not yet been processed by L2. Actually, $[t_0 - 2 * 88\mu s, t_0]$ would be sufficient.

In the actual ALICE experiment, the trigger system takes into account the data sent by the trigger detectors. In the specification we chose to avoid specifying physics data at the level of the Trigger detectors.

In order to represent physics data, the event type of the interaction is chosen according to the bold event rates of Table 1. Indeed, the "L0 Trigger" process chooses a random number in the interval [0,1] following a uniform law. The value of the number decides upon the type of the interaction: one of Central, Dimuon, Dielectron, Minbias, Miscellaneous(C,DM), Miscellaneous(C,DIEL), Miscellaneous(C,DM, DIEL), Others. Type Others stands for interactions that are not interesting at the physics level, but that have to be taken into account for P/F protection. Each of these types correspond to zero (Others), one or more (Miscellaneous) classes of detectors. The L0 trigger process performs first a P/F protection, which may change the event type (e.g. Miscellaneous(C,DM,DIEL) becomes Miscellaneous(DM,DIEL)). Second, the L0 trigger process verifies the busy status of the corresponding classes of detectors, which may also change the event type (e.g. Miscellaneous(DM,DIEL) becomes Dimuon).

Finally, if these two conditions are satisfied, the L0 trigger type is sent to all detectors (data flow "L0"). It reaches the detectors at $t_0 + 1.2\mu s$. In addition, the L0 trigger type and its occurrence time (t_0) is sent to the L1 Trigger (data flow "goL1"). They reach the L1 Trigger at $t_0 + 4.3\mu s$.

As soon as it receives data from the L0 Trigger, the L1 trigger performs a P/F protection (which can change the event type), computes the event number, and informs the detectors (unless the P/F protection fails). The L1 trigger information reaches the detectors at $t_0 + 5.5\mu s$ (data flow "L1"). In addition, the L2 trigger receives the L1 trigger type and the occurrence time at $t_0 + 88\mu s$ (data flow "goL2").

The L2 trigger immediately performs a P/F protection, decides upon a final trigger type, and sends an L2accept or L2reject to the detectors (data flow "L2"), which will reach them at $t_0 + 89.2\mu s$. The L2accept information contains the L2 trigger type.

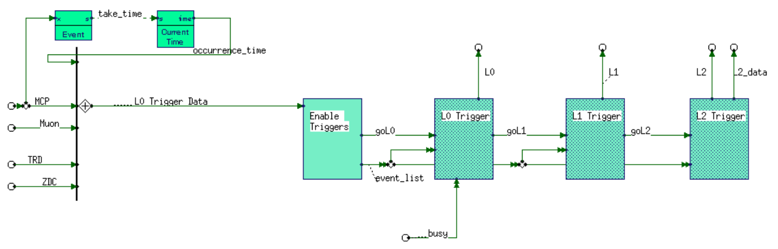


Figure 3: Trigger System

According to the delays used in the Trigger System specification, we see that L0, L1, L2 information reach the detectors in this order; and that, for a consecutive event occurred at t'_0 ($> t_0$), L2 arrives before L2'.

4.2 Tracking Detectors

The Foresight process “ALICE TRACKING DETECTORS” of Figure 1 is a data flow diagram made of 11 Foresight processes (one for each detector). It is given by Figure 4. Every detector receives input data from the Trigger System (“L0, L1, L2”), and from the DAQ sub-system “busy_RORCs”. L0, and L1 trigger information are sent to all detectors. Only those detectors concerned by the trigger type will perform some computation. L2 data are particular to each detectors. Indeed, some of them receive an L2accept, while some others receive an L2reject. The busy status of the RORCs is particular to each detector. Each detector receives its corresponding information through a particular data flow.

Every detector outputs its own busy status. They are all collected and sent to the Trigger system through the “busy” flow.

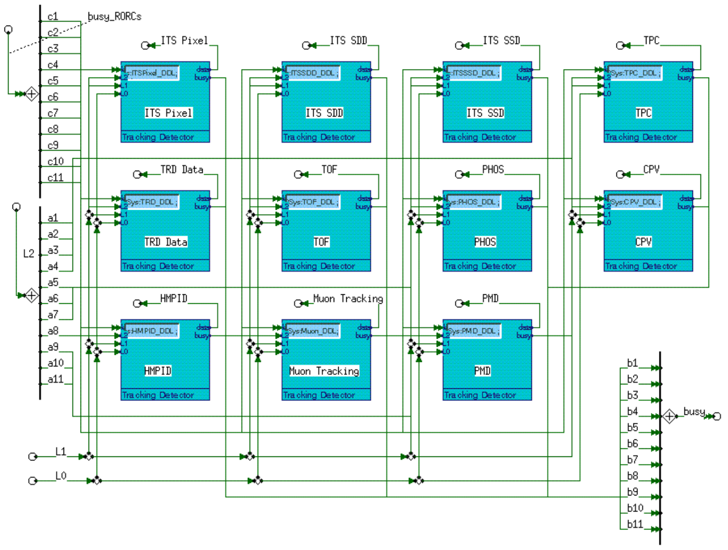


Figure 4: Tracking Detectors

Detectors have been specified as “reusable” Foresight processes. They are all defined by the same Foresight specification, but each process has its own set of parameters, which provides a particular behaviour to each of them.

Figure 5 shows the data flow diagram of a (generic) Tracking Detector.

L0/L1 signals are handled separately from L2 signals (in two different processes). This makes the input part (where data is collected and stored into the multi-event buffer) independent from the output part (where data is removed from the multi-event buffer and sent to the DAQ).

The “Handle L0/L1” process receives the L0, L1 trigger information and if it is the case sends the data to the “Insert into DDL Buffer”, which will actually store the data (data size, trigger type, and event number), corresponding to the event, into the multi-event buffer. The multi-event buffer is specified as a Foresight global variable “DDL Buffer”. It is accessible by several processes concurrently. The access to the DDL Buffer is regulated by a mutual exclusion semaphore (“Sem1”).

L2 data are stored into the “L2 Buffer”, a global variable which has the same structure as the DDL buffer variable. It is a multi-event buffer which stores L2accept or L2reject information. The “Insert into L2 Buffer” process simply inserts L2 information into the L2 Buffer variable, while the “Send Event to DAQ” process uses it for sending data to the DAQ. The semaphore "Sem2" manages the access to the L2 Buffer variable.

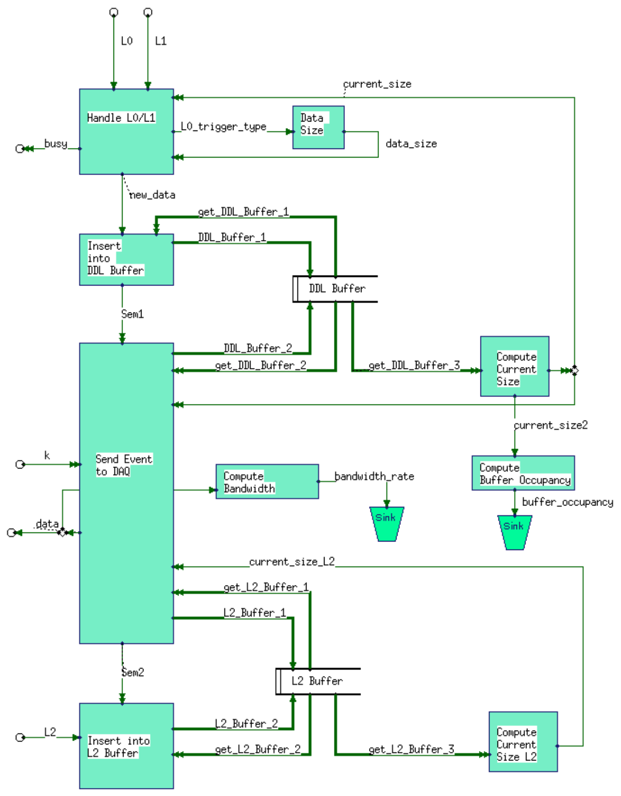


Figure 5: Tracking Detector

Figure 6 shows the State Transition Diagram (STD) corresponding to the "Handle L0/L1" process. In the initial state "Wait for L0", the process receives a L0 data. The data size of the event is computed, its value depends on the L0 trigger type. If the detector is not concerned by the event, a data size of 0.0 is provided, and the process returns in its initial state. Otherwise, the process becomes busy and enters a timeout period, waiting for the L1 trigger information (state "Wait for L1"). If the L1 trigger arrives before the timeout period elapses, the process computes the data size corresponding to the L1 trigger type. Since L1 may reduce the classes of detectors (due to P/F protection), a data size of 0.0 can be computed. In this case, the process resets the collected data, and then returns in the initial state. Otherwise, the process stores the event number, and enters a reading period, state "Busy Reading". At the end of the reading period, state "Test Full", the process fills the multi-event buffer with the new event, and checks if the multi-event buffer is full or not. In the first case, it remains busy. Otherwise, it becomes available for a new event. In both cases, it returns to the initial state. In the case of a full multi-event buffer, the busy status is updated once an event slot is freed in the multi-event buffer.

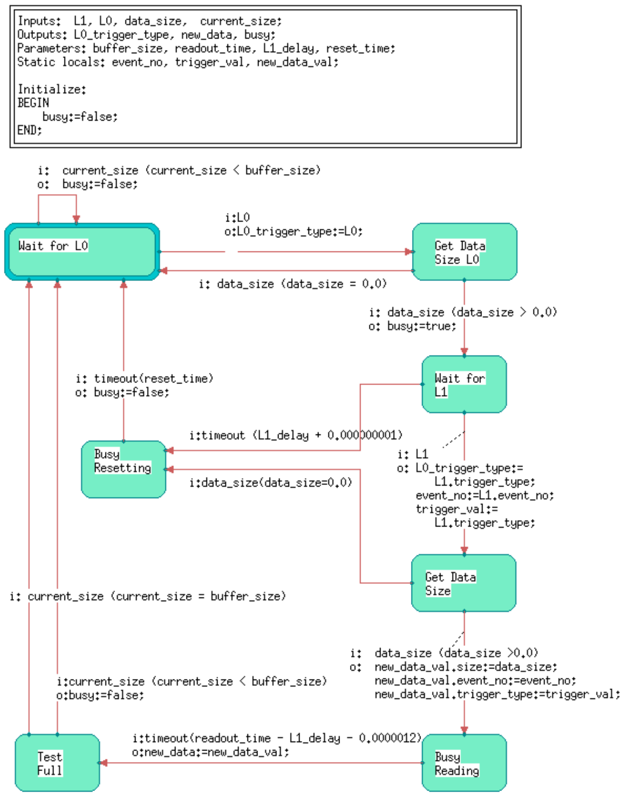

Inputs: L1, L0, data_size, current_size;
Outputs: L0_trigger_type, new_data, busy;
Parameters: buffer_size, readout_time, L1_delay, reset_time;
Static locals: event_no, trigger_val, new_data_val;
Initialize:
BEGIN
busy:=false;
END;
i: current_size (current_size < buffer_size)
o: busy:=false;
Wait for L0
i:L0
o:L0_trigger_type:=L0;
Get Data Size L0
i: data_size (data_size = 0.0)
i: data_size (data_size > 0.0)
o: busy:=true;
Wait for L1
i: timeout(reset_time)
o: busy:=false;
i:timeout (L1_delay + 0.000000001)
Busy Resetting
i:data_size(data_size=0.0)
i: L1
o: L0_trigger_type:=
L1.trigger_type;
event_no:=L1.event_no;
trigger_val:=
L1.trigger_type;
Get Data Size
i: current_size (current_size = buffer_size)
i: data_size (data_size >0.0)
o: new_data_val.size:=data_size;
new_data_val.event_no:=event_no;
new_data_val.trigger_type:=trigger_val;
i:current_size (current_size < buffer_size)
o:busy:=false;
i:timeout(readout_time - L1_delay - 0.0000012)
o:new_data:=new_data_val;
Test Full
Busy Reading

Figure 6: Handle L0/L1

Figure 7 shows the processing of the L2 information, and the sending of the data to the DAQ sub-system. It is a STD that is part of the “Send Event to DAQ” data flow diagram.

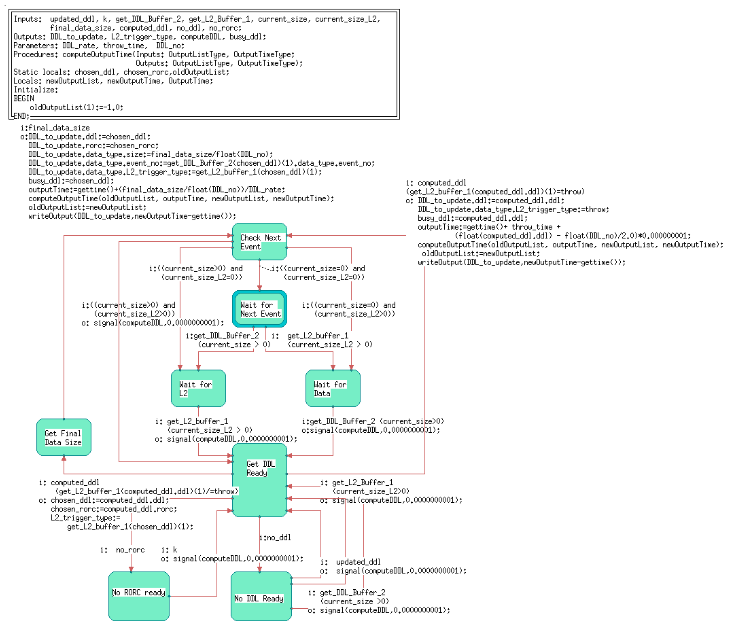
The role of this STD is to retrieve, from the DDL Buffer (multi-event buffer), the next data to send to the DAQ. The behaviour must be such that *each* LDC, RORC, and GDC buffer is fed *individually*. This STD checks the L2 buffer, in order to get the L2accept or L2reject information. It removes the data both from the DDL Buffer and from the L2 buffer, and sends the event data through the corresponding DDL.

In the initial state “Wait for Next Event”, both the DDL Buffer and the L2 Buffer are empty. Once information from both is available, the process looks for a “ready” DDL, i.e., a DDL for which the L2 buffer information is a L2reject, or a DDL which is not currently sending a data, for which the corresponding RORC is not full, and for which both the data in the DDL buffer and in the L2 buffer are available (state “Get DDL Ready”).

If no DDL is ready, the process enters the “No DDL Ready” state, waiting for a new data in the multi-event buffer, or in the L2 buffer, or for a DDL that becomes ready. If no DDL is ready because the corresponding RORCs are full, then the process enters state “No RORC ready”.

If it finds a DDL, either the L2 information is a L2reject, and the process goes to the “Check Next Event” state; or the L2 information is an L2accept, and it enters state “Get Final Data Size”. It computes the data size according to the event type decided by the L2 trigger, since the L2 trigger chooses a *final* event type. It outputs a value “DDL_to_update” containing the DDL number, the corresponding RORC number, the data size, the event number, and the L2 trigger type. The value is actually sent with a delay corresponding to the time necessary to send the computed data size along the DDL (given the DDL transmission rate). During this period, the corresponding DDL is marked as busy (it cannot be chosen to send another data). When the “DDL_to_update” value is actually output, the process that receives it in input (not shown here), will update the multi-event buffer, the L2 buffer, mark the DDL as ready, and output the data so that it is received by the corresponding RORC (flow “data” in Figure 5).

The use of the Foresight instruction: `writeOutput(data,delay)` enables to simulate parallelism within a sequential process. Indeed, the Send Event STD performs all its operations at the *same* simulation time. Therefore, if two DDLs are ready at the same time, and if there is a data in their corresponding buffer, the two pieces of data will be sent at the same time through the two DDLs. They will reach their corresponding RORC at two different moments depending on the value of the parameter `delay`. The STD suspends its work when no DDL is ready, or when the DDL buffer is empty, or when all RORCs are full (states “No RORC ready”, “No DDL ready”).



```
Inputs:  updated_ddl, k, get_DDL_Buffer_2, get_L2_Buffer_1, current_size, current_size_L2,
         final_data_size, computed_ddl, no_ddl, no_rorc;
Outputs: DDL_to_update, L2_trigger_type, computeDDL, busy_ddl;
Parameters: DDL_rate, throw_time,  DDL_no;
Procedures: computeOutputTime(Inputs: OutputListType, OutputTimeType;
                             Outputs: OutputListType, OutputTimeType);
Static locals: chosen_ddl, chosen_rorc,oldOutputList;
Locals: newOutputList, newOutputTime, OutputTime;
Initialize:
BEGIN
    oldOutputList(1):=-1.0;
END;
```

```
i:final_data_size
o:DDL_to_update.ddl:=chosen_ddl;
  DDL_to_update.rorc:=chosen_rorc;
  DDL_to_update.data_type.size:=final_data_size/float(DDL_no);
  DDL_to_update.data_type.event_no:=get_DDL_Buffer_2(chosen_ddl)(1).data_type.event_no;
  DDL_to_update.data_type.L2_trigger_type:=get_L2_buffer_1(chosen_ddl)(1);
  busy_ddl:=chosen_ddl;
  outputTime:=gettime()+(final_data_size/float(DDL_no))/DDL_rate;
  computeOutputTime(oldOutputList, outputTime, newOutputList, newOutputTime);
  oldOutputList:=newOutputList;
  writeOutput(DDL_to_update,newOutputTime-gettime());
```

```
i: computed_ddl
(get_L2_buffer_1(computed_ddl.ddl)(1)=throw)
o: DDL_to_update.ddl:=computed_ddl.ddl;
   DDL_to_update.data_type.L2_trigger_type:=throw;
   busy_ddl:=computed_ddl.ddl;
   outputTime:=gettime()+ throw_time +
             (float(computed_ddl.ddl) - float(DDL_no)/2.0)*0.000000001;
   computeOutputTime(oldOutputList, outputTime, newOutputList, newOutputTime);
    oldOutputList:=newOutputList;
   writeOutput(DDL_to_update,newOutputTime-gettime());
```

```
i:((current_size>0) and
  (current_size_L2=0))
```

```
i:((current_size=0) and
  (current_size_L2=0))
```

```
i:((current_size>0) and
  (current_size_L2>0))
o: signal(computeDDL,0.0000000001);
```

```
i:((current_size=0) and
  (current_size_L2>0))
```

```
i:get_DDL_Buffer_2
  (current_size > 0)
```

```
i:  get_L2_buffer_1
  (current_size_L2 > 0)
```

```
i: get_L2_buffer_1
   (current_size_L2 > 0)
o: signal(computeDDL,0.0000000001);
```

```
i:get_DDL_Buffer_2 (current_size>0)
o:signal(computeDDL,0.0000000001);
```

```
i: computed_ddl
   (get_L2_buffer_1(computed_ddl.ddl)(1)/=throw)
o: chosen_ddl:=computed_ddl.ddl;
   chosen_rorc:=computed_ddl.rorc;
   L2_trigger_type:=
       get_L2_buffer_1(chosen_ddl)(1);
```

```
i: get_L2_Buffer_1
   (current_size_L2>0)
o: signal(computeDDL,0.0000000001);
```

```
i:no_ddl
```

```
i:  no_rorc
```

```
i: k
o: signal(computeDDL,0.0000000001);
```

```
i:  updated_ddl
o:  signal(computeDDL,0.0000000001);
```

```
i: get_DDL_Buffer_2
   (current_size >0)
o: signal(computeDDL,0.0000000001);
```

Figure 7: Send Event STD

4.3 DAQ Sub-system

Figure 8 shows the Foresight process “DAQ Sys” of Figure 1. It is a data flow diagram made of 11 Foresight processes (one for each detector) corresponding to the FEDCs of the detector, one Foresight process “Switch and GDCs”, and one process for the EDM (Event Destination Manager). The FEDC processes receive in input the “data” (Figure 5) sent by the corresponding Tracking detector along the DDLs. They output the busy status of the RORCs, as well as the computed sub-events to “Switch and GDCs” process (flow “busy_RORCs”). The EDM process receives in input the list of available GDCs, i.e., those GDCs that can still receive sub-events, and simply forwards this list to the FEDCs. The EDM process is not necessary in this specification, however it corresponds to an existing element of the ALICE DAQ system [4], which provides this information to the detectors. The “Switch and GDCs” process outputs the full event to the PDS System (flow “PDS_data”).

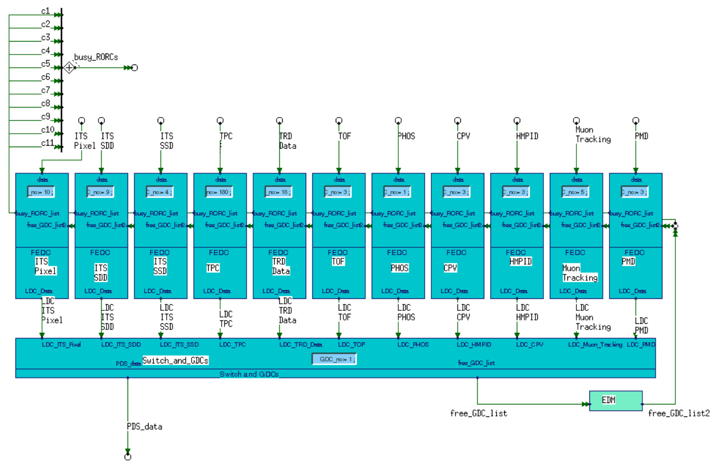


Figure 8: DAQSys

FEDCs processes of Figure 8 are Foresight parameterized reusable processes. Figure 9 shows the generic FEDC process. It is a data flow diagram that maintains two buffers as global variables: the “RORCs”, and the “LDCs”. The RORCs buffer is a multi-buffer (one for each RORC of the detector) which stores the data coming from the DDLs. The LDCs buffer is a multi-buffer (one for each LDC of the detector) which stores the sub-events. The “Insert into RORC” process simply stores, into the corresponding RORC, the data received through the DDL. It checks if the RORC is full or not, and notifies the detector. In addition, it informs the “Bus” process that a new data arrived in a RORC. The “Bus” process specifies the behaviour of *all* buses of the detector. It checks for a bus, which is not currently sending a data, and for which the corresponding RORCs have some data, and the corresponding LDC is not full. It marks the bus as busy, and after a period corresponding to the time necessary to send the data trough the bus, it updates the RORC, and the LDC (data is removed from the RORC and inserted into the LDC), and marks the bus as not busy. Similarly to what happens in the Tracking Detectors, the “Bus” process is specified in such a way that *one* bus sends data to a LDC in a *sequential* manner, while *all* buses perform their activities in *parallel*.

The Bus process updates the busy status of the RORCs, according to the RORC buffers, and the LDCs buffers.

The “SubEvent Building” process checks all LDCs for a complete sub-event. Then, it verifies if the corresponding GDC is free. If it is the case, the sub-event is output on the “LDC_data” flow, and after a period of time corresponding to the time necessary to send the sub-event through the Switch, the corresponding LDC is updated, and the Bus process is informed. Two LDCs may send sub-events in parallel to the Switch provided they do not send the sub-event to the same GDC.

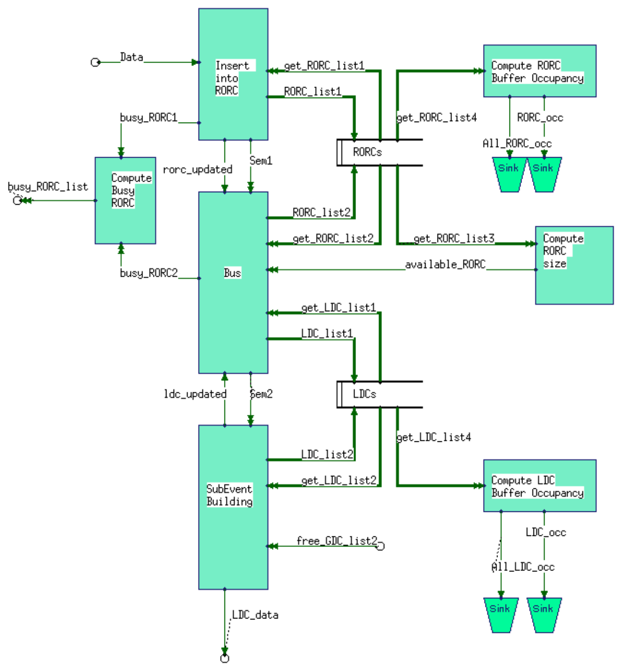


Figure 9: FEDC

The "Switch and GDCs" process of Figure 8 is a data flow diagram given by Figure 10. Sub-events sent by the LDCs of each detector are received by the "Send to Switch1" process. This process simply forwards the sub-events received along the different flows to the unique flow "subevent_in". As soon as a sub-event arrives to the "subevent_in" flow, the corresponding GDC is marked as busy, by process "Compute Busy GDCs". The busy status of GDCs is then forwarded to the FEDCs by the EDM.

The "Switch1" process simply outputs the sub-event received in input to the flow "subevent_out" with a delay corresponding to the transmission time of the sub-event through the Switch according to the transmission rate of the Switch. The behaviour of "Switch1" is such that data are sent in parallel to two different GDCs. It is important to notice that for a given event, the sub-events are sent sequentially, but in an interleaved way (coming from all the detectors). Moreover, two different events that must be collected by the same GDC can be sent in an interleaved way to the GDC, i.e., the sub-events of the second event may reach the GDC, while some sub-events of the first event are still in LDCs.

The "Insert into GDC" process updates the corresponding GDC once the sub-event arrives to the "subevent_out" flow. The GDCs are handled as a global variable called "GDCs". The GDCs global variable is a multi-buffer (one for each GDC). The "Event Building" process checks if a GDC has a complete event. If it is the case, it outputs the event to the Disk of the GDC, and updates the GDC after a delay corresponding to the transmission time of the event to the Disk. Once the Disk has built a file of 1GB, this file is output to the "event_out" flow, and the Disk is updated once the event has been fully transmitted through the Switch2.

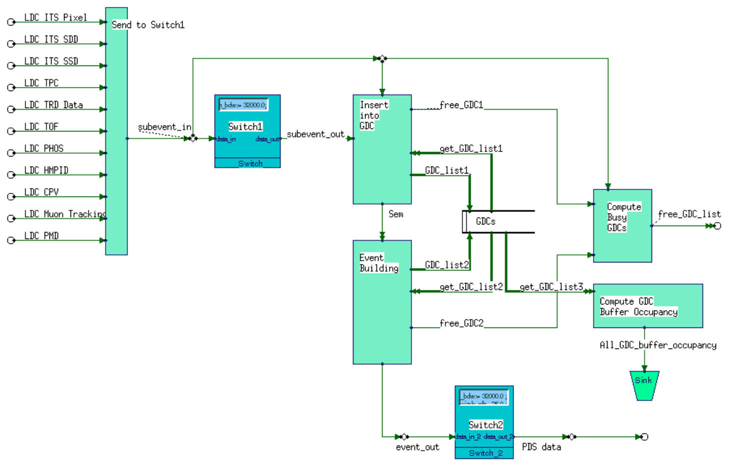


Figure 10: Switches and GDCs

Foresight does not allow process instantiation with a number of processes that are not known before run-time. Therefore, in order to (1) simulate parallelism within a Foresight model, and to (2) take into account the fact that the model must remain modifiable only by changing parameters, Tracking Detectors, FEDCs and GDCs are specified such that:

1. DDLs, RORCs, LDCs, GDCs are global variables specified as multi-buffers;
2. One Foresight process specifies the behaviour of:
    - *all* DDLs of a detector: process “Send Event to DAQ” in Figure 5;
    - *all* RORCs of a detector: process “Bus” in Figure 9;
    - *all* Buses of a detector: process “Bus” in Figure 9;
    - *all* LDCs of a detector: process “SubEvent Building” in Figure 9;
    - *all* GDCs of a detector: process “Event Building in Figure 10;
    - *all* Disks of a detector: process “Event Building in Figure 10;
3. When sending data along a link:
    (a) The link and the recipient are marked as being busy;
    (b) A timeout is started (in Foresight `writeOutput(data,delay)` instruction);
    (c) At the end of timeout: data are removed from the buffer of sender, buffer of recipient is updated, the link and recipient are marked as not busy.

5 A More Abstract Foresight Specification

The simulation of the detailed specification, described in Section 4, is too slow to produce meaningful results. Therefore, it has been modified in order to speed up simulation performance. The trigger system has not been modified. The Tracking Detectors and the DAQ sub-system have been modified such that their specification has a more abstract internal behaviour, while guaranteeing the same observable behaviour.

During execution the access to global variables slows down the simulation. Therefore, all global variables (L2 Buffer, DDL Buffer, RORC buffer, LDC buffer, GDCs, Disk) have been removed from the DAQ Sys and the Tracking Detector processes.

The processes accessing these variables have been modified in order to maintain the feeding of buffers identical to that provided by the initial specification.

This section reports the changes applied to the detailed specification above.

5.1 Tracking Detectors

Figure 11 shows the abstract version of the Tracking Detector given by Figure 5. The global variables "DDL Buffer" and "L2 Buffer" are removed. The process "Send Event to DAQ" handles these two variables locally.

In the initial specification, flow "data" of Figure 5 transfers one fragment of data. In the more abstract specification, it transfers *all available* fragments of data of the multi-event buffer (available on the first slot). Thus, in the case of the TPC detector, the flow may be updated once instead of 180 times. Of course, the receiving process must update the RORCs buffers accordingly, in order to guarantee the same behaviour as in the initial specification, i.e., each RORC is filled individually.

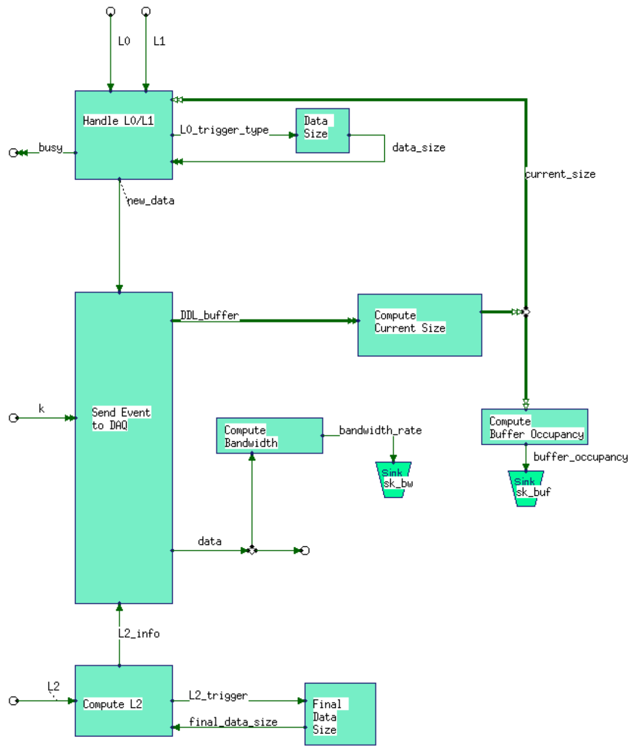


Figure 11: Abstract Tracking Detector

5.2 DAQ Sub-System

The way of sending sub-events to GDCs has been changed. In the initial specification LDCs compete for sending data to the Switch: they permanently wait on the availability of the link to the GDC. In the more abstract specification, the LDCs do not care about the availability of the link. They simply send the sub-event to the Switch, and wait for this sub-event to reach the GDC. The Switch regulates the way sub-events are sent to the GDC, and once the data of a given LDC has reached the GDC, then the Switch awakes the corresponding LDC. In this manner, LDCs processes do not react each time a new data reaches a GDC, they react only when their own data has reached a GDC.

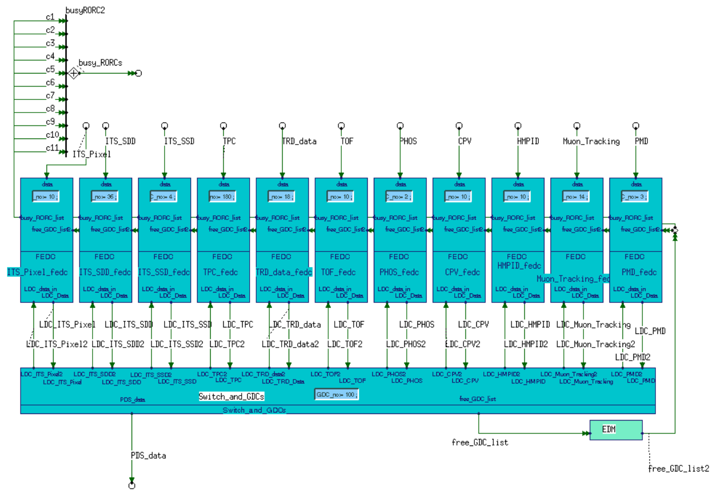


Figure 12: Abstract DAQSys

Figure 12 shows the abstract version of the DAQ sub-system. We observe that each FEDC receives an additional input from the "Switch and GDCs" process (e.g., FEDC process for the Pixel detector receives inputs from flow "LDC_Pixel2"). It is used by the Switch to awake the FEDCs once their sub-event has reached the GDC.

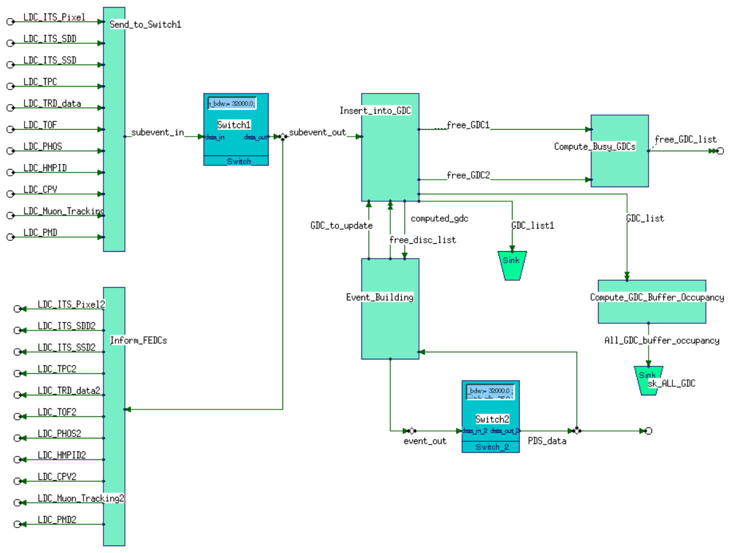


Figure 13: Abstract Switches and GDCs

Figure 13 shows the "Switch and GDCs" process. The "GDCs" global variable has been removed. It is handled locally by the "Insert_into_GDC" process. As mentioned above, as soon as a sub-event reaches a GDC, the "Inform_FEDCs" process outputs the information to the corresponding FEDC.

Figure 14 shows the abstract version of the FEDC of Figure 9. Global variables "RORCs" and "GDCs" have been removed. They are maintained locally by process "RORC and Bus" and "Subevent_Building_STD" respectively.

As mentioned before, flow "data" transfers multiple pieces of data. Process "RORC and Bus" fills simultaneously and individually the corresponding positions of the RORC multi-buffer. Therefore, the local variable RORC is updated once instead of multiple times.

The transfer of pieces of data from RORCs to LDCs follows the same idea as the transfer from the multi-event buffer to the RORCs. Flow "computed_rorc" indicates *all available* RORCs, i.e., those RORCs that are ready to send data to LDCs through an available bus. Thus, process "Subevent_Building_STD" updates the LDCs local variable once instead of multiple times.

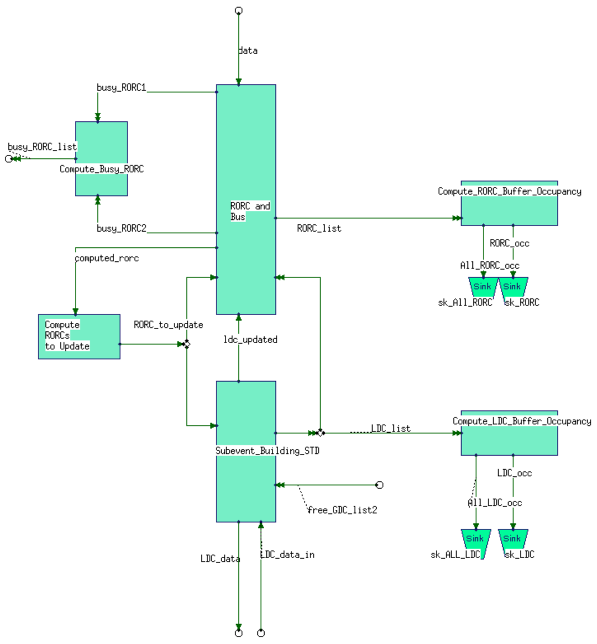


Figure 14: Abstract FEDC

Figure 15 shows the improvement obtained with the more abstract specification wrt the initial one. The execution of the initial specification needs 18000 min of CPU time to reach 0.2 sec of simulation time, while the more abstract specification reaches 3.8 sec of simulation for the same CPU time. The execution is performed on an UltraSparc10 with 384MB of RAM.

The more abstract specification described above has a more abstract internal behaviour. Nevertheless it guarantees the same observable behaviour as the initial specification: Trigger signals, Buses and Switches observable behaviours are the same; and the feeding of each *individual* LDCs, RORCs, GDCs buffers is maintained.

Defining a still more abstract specification (with a more efficient simulation performance) stops guaranteeing the same observable behaviour.

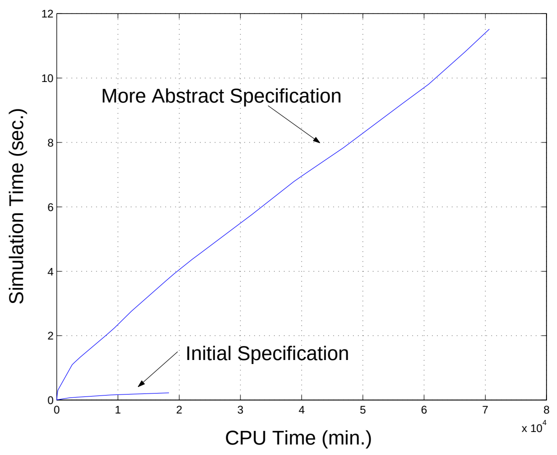


Figure 15: Simulation Performances

6 Results

The Foresight specification described above has served as a basis for several simulations. First, it has been simulated with the DAQ sub-system being able to absorb the full rate of the DDLs. Then, several other other simulations, where the DAQ sub-system offers a restricted bandwidth to the tracking detectors have also been performed: with 12 detectors and a new set of parameters; with a new trigger algorithm that cuts to 20Hz the number of L2 signals for Minbias and Central events; finally, with a trigger system signalling Central events only.

6.1 Full DDL Bandwidth

In the case with full DDL bandwidth, the DAQ sub-system actually works as an infinite buffer capable of absorbing all data coming from the tracking detectors. Detectors send data to the DAQ at a rate which is equal to the number of DDLs at their disposal times the DDL rate (100MB/s). For instance, in the case of the TPC detector, data are sent at a rate of 18000MB/s, since TPC has 180 DDLs. Parameters of Tables 2, and 4 have been used for this simulation.

6.1.1 Results

Table 5 summarizes the results of the trigger outputs observed during this simulation.

L0 rate depends upon the rate of the L0 trigger inputs (Table 1), P/F protection, and the busy status of the detectors. The high numbers of Central and Minbias events are the result of the high rates of Central and Minbias L0 trigger inputs. These events feed into the final Dimuon event rate. Indeed, while TPC and TRD are busy with Central and Minbias events, Dimuon events can still be accepted by other detectors (TPC, and TRD are not required for the Dimuon class). Row “Misc” reports the number of combined events signaled by L0 and L1 respectively. Row ”Others” shows the number of interactions that are not significant at the physics level but that have to be taken into account for P/F protection. The rate of such interactions has been estimated at 2000 Hz (see Table 1).

In this simulation, L1 outputs take into account P/F protection only. L0 and L1 rates are similar since it has been assumed that Dielectron events are identified at L0.

In practice this information is available at L1. The final L2 rates are not affected much by this simplification.

L2 rates for Dimuon and Dielectron events appear higher than L1 rates. As a result of re-classification, some of the combined events, in the row "Misc" of Table 5, become Dimuon or Dielectron events.

If we compare the column of L2 outputs obtained with the expected L2 rates for ALICE, given by Table 6, we notice that Dielectron and Dimuon events are relatively well represented. It is important to notice that the event rate at L0 input for Dimuon is 650 Hz combined with other events (see Table 1). Therefore, with such an input rate, it is impossible (due to P/F protection) that we obtain the expected rate of 650 Hz at L2 output.

Table 5: Event Rates with Full DDL Bandwidth

	L0 (Hz)	L1 (Hz)	L2 (Hz)
Central	137	133	92
Dimuon	462	457	585
Dielectron	159	152	197
Minbias	737	714	409
Misc	203	197	
Sub-total	1698		
Others	1997		
Total	6038		

Table 6: Expected Rates at L2

	C	MB	DM	DIEL	Total
L2	20	20	650	200	890 Hz

Figure 16 represents the distribution over time, of the L0, L1, and L2 signals. It shows that the three signals grow according to a linear function. Table 5 reports the values corresponding to 1.0s.

Table 7 summarizes the values of the buffer occupancy and of the bandwidth rate of each detector after 1.0s of simulation time.

The "Buffer Full" row shows the percentage of time during which the buffer of the corresponding detector has a full buffer. As seen before, every detector has a multi-event buffer of 4 positions. Except TPC, TRD and ITSSSD, none of the detectors employs the 4 event slots. Detector ITSSSD reaches the bound a very small percentage of time. On the other hand, the remaining two detectors TPC and TRD are heavily limited by the buffer: TPC reaches the bound of the buffer 23% of time, and TRD, 44% of time.

The "Bandwidth Rate" row shows the effective usage of the DDL bandwidth. It corresponds to the number of bytes per second that transit through the DDLs of the detector. Compared with the allocated bandwidth [DDL * 100MB/s], we see that TPC employs 79% (14279/18000) of its bandwidth, and TRD employs 88% of its bandwidth (1585/1800). The other detectors employ a very small fraction of their bandwidth. According to this table, TPC and TRD are the most busy detectors, and TRD is even more busy than TPC. In fact, if we consider the Detectors parameters of Table 2, we can already see that TRD needs more time than TPC for sending its data. Indeed, in the case of a Central event, TRD sends 8MB at a rate of 1800MB/s, thus it requires 4.4ms; TPC sends a Central event of 65MB (average) at 18000MB/s, thus it takes 3.6ms.

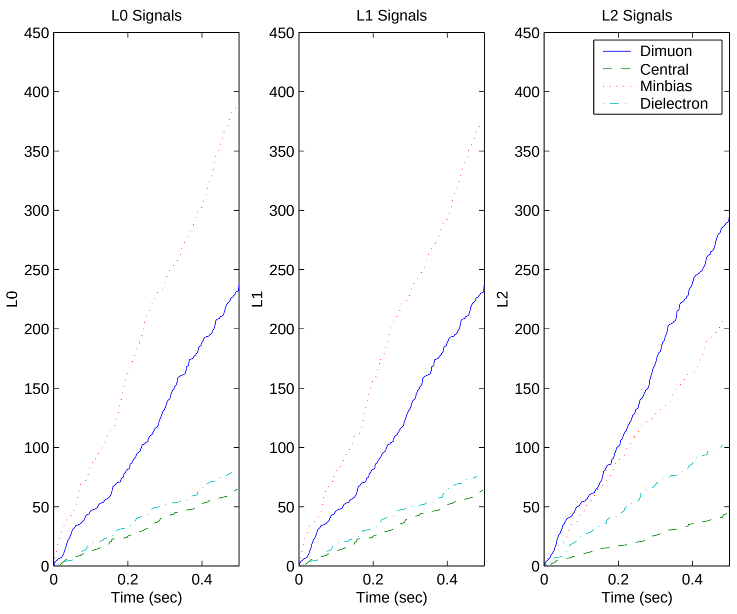


Figure 16: L0/L1/L2 Trigger Signals

Table 7: Buffer Full and Bandwidth Rate

	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	23 %	44 %	0 %	0 %	0 %	0 %
Bandwidth Rate	14279MB/s	1585MB/s	146MB/s	35MB/s	23MB/s	23MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	
Buffer Full	0 %	0 %	0.037 %	0 %	0 %	
Bandwidth Rate	374MB/s	588MB/s	149MB/s	19MB/s	79MB/s	
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	

Figure 17 shows the distribution over time of the “Buffer Full” and the “Bandwidth Rate” fields of the TPC and TRD detectors. After a short period of stabilization, the values of these fields reach a (more or less) constant value (reported in Table 7).

Figure 18 shows the behaviour of the L0 signal wrt the buffer occupancy of TPC and TRD. We can see that when TPC and TRD have occupied their 4 event slots, no Minbias event is accepted by L0, while during the same interval of time, Dimuon events are accepted instead. For instance, during the interval [0.248s, 0.251s], no Minbias have been accepted. During interval [0.254s, 0.255s], no Minbias have been accepted, but 3 Dimuon have been signalized.

6.1.2 Foresight Performances

The verification of a system specified using Foresight is performed through simulation. Therefore, the efficiency of the simulation, i.e., the evolution of the simulation time wrt the implied CPU time, is of particular interest.

In the case with full DDL bandwidth, the simulation of the Foresight specification runs relatively fast. After less than 5 minutes, the simulation time reaches 1.0s. In the case with full DDL bandwidth the system stabilizes after few interactions, and then the whole system behaves in a constant (linear) manner. The simulation ran on an Sun workstation UltraSparc10 with 128MB of RAM.

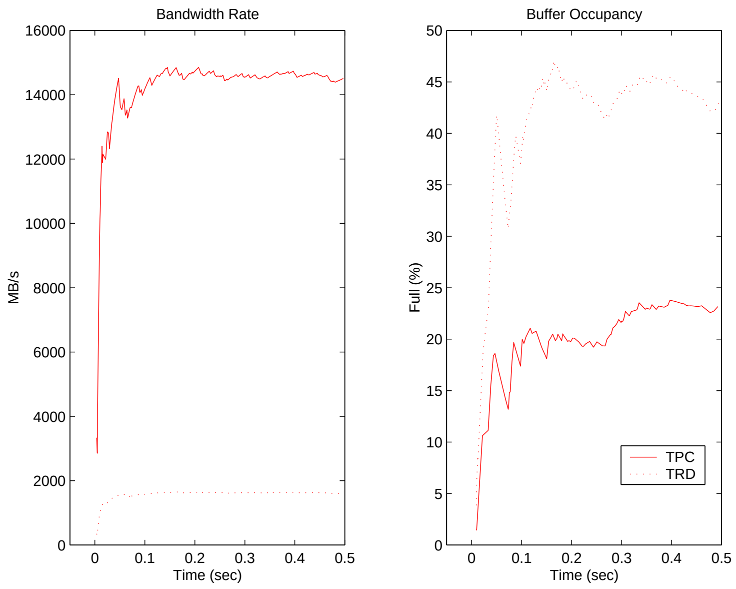


Figure 17: Bandwidth Usage and Buffer Occupancy

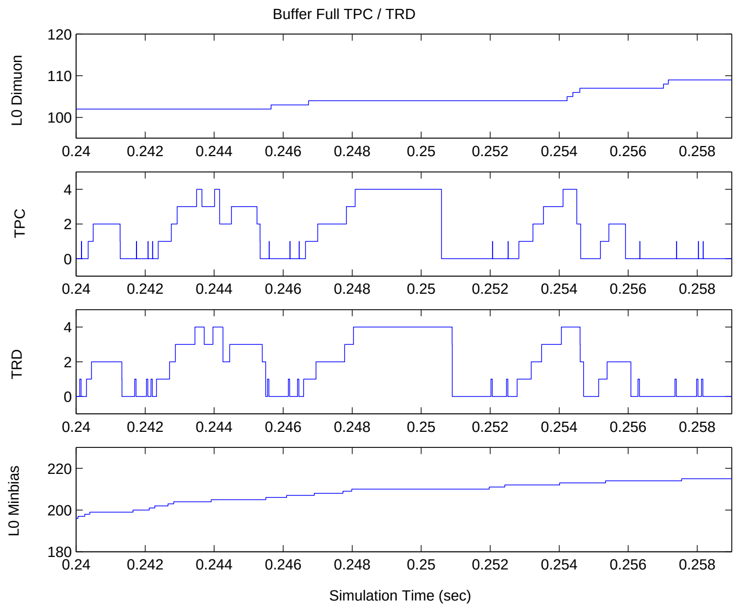


Figure 18: Buffer Full and L0 signals

6.2 Restricted DDL Bandwidth - 12 Detectors/New Parameters

The case with restricted DDL bandwidth corresponds to the specification described in Sections 3 and 5. This section describes a specification involving 12 (instead of 11) detectors, and takes into account up-to-date parameters that were not available at the time when the specification described in Section 3 has been defined. The 12th detector, called TRG, does not correspond to an additional tracking detector. It represents data incoming from the detectors involved in trigger decisions.

Contrarily to the simulation described in Section 6.1, the DAQ actually works and exerts a pressure on the DDL bandwidth allocated to each detector. The DAQ consumes data incoming from the detectors less faster than the detectors produce it for the DAQ. Therefore, the LDCs of some detectors become full, causing their corresponding RORCs to become full. Then, these detectors can no longer send data to the RORCs, causing in turn their multi-event buffers to become full.

When some detectors involved in Central and Minbias events are busy, then Dimuon events are given priority. Dielectron events do not gain from this effect, because detectors such as TRD, which limit Central and Minbias events, take part in Dielectron class too.

6.2.1 New Specification

Figure 19 shows the 12 detectors receiving signals from the trigger system, and sending data to the DAQ sub-system.

The specifications of the trigger system and the DAQ sub-system have been updated in order to take into account the 12th detector. However, they are similar to those described in Sections 3 and 5. Therefore, the same algorithms for the Switches, the sub-event and event building as well as for the GDC choice (modulo) are applied.

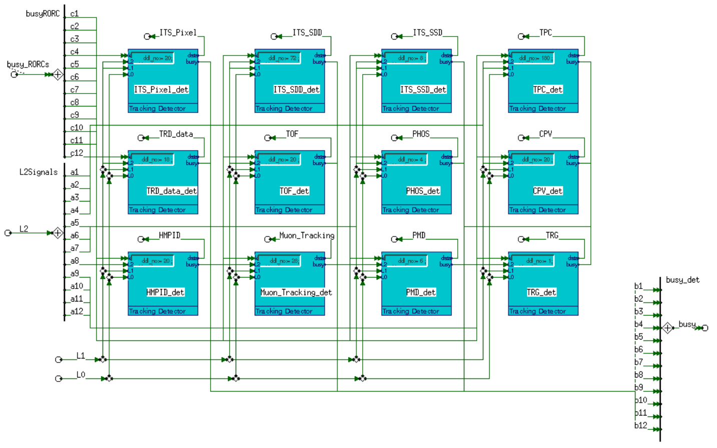


Figure 19: Tracking Detectors

6.2.2 Detectors and DAQ Parameters

Detectors and DAQ Parameters have been updated; values are given by Tables 8 and 9 respectively.

The main differences between detector parameters of Table 2 (used for the simulation described in the previous section) and Table 8 are the following. In Table 8, some detectors have an increased multi-event buffer capacity: 50 slots for TRD, 40 for Muon, and 10 for HMPID. Others have a reduced multi-event buffer capacity: 2 for TOF, and 1 for ITSSSD. Read-out times of TRD, TOF, CPV, HMPID, Pixel, PHOS, and PMD have been reduced; while those of ITSSDD and ITSSSD have been increased. The reset time and throw time of the Muon detector have been set to $1\mu s$. Those of all other detectors have been set to $0\mu s$. Finally, the data size for Pixel and ITSSSD have been reduced.

DAQ parameters, given by Table 9 have been updated accordingly to Detectors parameters of Table 8. We observe that the 12th detector, TRG, participates to all events.

Table 8: Detectors Parameters

	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer	4	50	40	2	4	10
Read Time	$88\mu s$	$4.5\mu s$	$55\mu s$	$10\ \mu s$	$4.5\mu s$	$11.9\mu s$
Reset Time	$0\mu s$	$0\mu s$	$1\mu s$	$0\mu s$	$0\mu s$	$0\mu s$
Throw Time	$0\mu s$	$0\mu s$	$1\mu s$	$0\mu s$	$0\mu s$	$0\mu s$
Trans Rate	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s
C (max)	75.9MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
C (min)	56.1MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
DM (max)	0.0MB	0.0MB	0.15MB	0.0MB	0.0MB	0.0MB
DM (min)	0.0MB	0.0MB	0.15MB	0.0MB	0.0MB	0.0MB
MB (max)	75.9MB	8.0MB	0.15MB	0.18MB	0.12MB	0.12MB
MB (min)	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB
DIEL (max)	6.33MB	0.27MB	0.15MB	0.0MB	0.0MB	0.0MB
DIEL (min)	4.5MB	0.27MB	0.15MB	0.0MB	0.0MB	0.0MB
DDL	180	18	28	20	20	20
RORC	180	18	14	10	10	10
LDC	180	18	5	3	3	3
FEDC	180	18	5	3	3	3
Min RORC Size	0.4216MB	0.4444MB	0.00535MB	0.009MB	0.006MB	0.006MB
Min LDC Size	0.4216MB	0.4444MB	0.0321MB	0.072MB	0.048MB	0.048MB

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer	4	4	1	4	4	50
Read Time	$0.1\mu s$	$230\mu s$	$154\mu s$	$10\mu s$	$10\mu s$	$0\mu s$
Reset Time	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$
Throw Time	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$	$0\mu s$
Trans Rate	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s
C (max)	0.24MB	1.5MB	0.22MB	0.02MB	0.12MB	0.12MB
C (min)	0.14MB	1.5MB	0.22MB	0.02MB	0.03MB	0.12MB
DM (max)	0.14MB	0.0MB	0.0MB	0.02MB	0.12MB	0.02MB
DM (min)	0.06MB	0.0MB	0.0MB	0.02MB	0.03MB	0.02MB
MB (max)	0.24MB	1.5MB	0.22MB	0.02MB	0.12MB	0.12MB
MB (min)	0.0MB	0.0MB	0.0MB	0.0MB	0.0MB	0.12MB
DIEL (max)	0.24MB	1.5MB	0.22MB	0.02MB	0.12MB	0.12MB
DIEL (min)	0.14MB	1.5MB	0.22MB	0.02MB	0.03MB	0.12MB
DDL	20	72	8	4	6	1
RORC	10	36	4	2	3	1
LDC	10	9	4	1	3	1
FEDC	10	9	4	1	3	1
Min RORC Size	0.012MB	0.02083MB	0.0275MB	0.005MB	0.02MB	0.12MB
Min LDC Size	0.024MB	0.166MB	0.055MB	0.02MB	0.04MB	0.12MB

Table 9: DAQ Parameters

	DDL	RORC	LDC	GDC	Disk	PDS
Quantity	397	298	240	100	100	25
Buffer Size	-	12MB	128MB	512MB	50GB	∞
Min GDC Size	-	-	-	86.69MB	-	-

	DDL	Bus	Switch1	Disk	Switch2	
Rates	100MB/s	100MB/s	40MB/s	25MB/s	25MB/s	

	Central	Dimuon	Dielectron	MinBias		
Sub-event qty	240	20	231	240		
Size (max)	86.69MB	0.45MB	8.88MB	86.69MB		
Size (min)	66.7MB	0.28MB	6.95MB	0.12MB		
Detectors	All	ITSPixel	ITS Pixel	All		
		PHOS	ITSSDD			
		Muon	ITSSSD			
		PMD	TPC			
		TRG	TRD			
			PHOS			
			Muon			
			PMD			
			TRG			

6.2.3 Results

Table 10 shows the trigger outputs rates observed during this simulation. In this case, signals do not grow linearly. Therefore, rates are taken at three relevant moments: 1.5s, 6s, and 10s. Figure 20 shows the corresponding progression over time of the L0, L1 and L2 information.

During the interval [0s, 1.5s], the three signals grow linearly. During this period, the rates are equivalent to those obtained with a full DDL bandwidth (figures of Table 10 at 1.5s are similar to those of Table 5). This means that during the interval [0s, 1.5s], the DAQ is able to absorb the full DDL bandwidth incoming from the detectors.

In the interval [1.5s, 6s], there is a clear restriction of Central, Minbias, and Dielectron events. We see that Dimuon events benefit from this fact: the rates of L0, L1, and L2 signals for Dimuon events increase, while the others decrease. After 6s, we observe that Dimuon events are limited too, and the corresponding signals decrease accordingly.

The limitation of these rates is clearly related to the LDCs and RORCs buffer occupancies. At 1.5s, the LDCs, and RORCs of some of the detectors involved in Central, Minbias and Dielectron events are full, causing their multi-event buffers to become full.

Table 10: Event Rates - 12 Detectors (up to 1.5s, 6s, and 10s)

1.5s	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk
Central	125	121	94	0
Dimuon	452	447	570	4
Dielectron	140	137	191	2
Minbias	709	688	405	2
Misc	203	200		
Sub-total	1629			
Others	1997			
Total	5991			

6s	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk
Central	36	35	29	2
Dimuon	570	558	596	80
Dielectron	39	38	53	26
Minbias	214	208	124	26
Misc	62	61		
Sub-total	948			
Others	1997			
Total	5959			

10s	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk
Central	25	24	20	20
Dimuon	390	383	409	208
Dielectron	26	26	36	66
Minbias	145	141	85	92
Misc	42	42		
Sub-total	628			
Others	1997			
Total	5977			

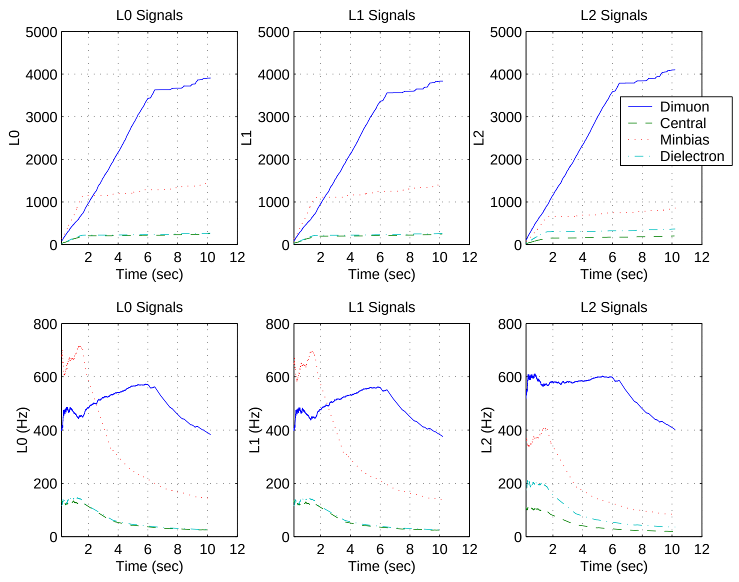


Figure 20: L0/L1/L2 Triggers: top shows cumulative signals, bottom shows signal rates

Figure 21 shows the LDCs and RORCs buffer occupancies of each detector. We see that at 1.5s, the TRD and TPC detectors have their LDCs buffers full. TRD has its RORCs buffers full shortly before 2s, causing the limitation of Central, Minbias and Dielectron events. Due to the limitation imposed by TRD, RORCs buffers of TPC do not get full before 8.5s.

During the interval [1.5s, 6s], we see that detectors involved in Dimuon events continue to fill their LDCs buffers. The TRG detector fills its LDCs at 5s, and Muon at 6s. TRG has its RORCs full, causing Dimuon events to be rejected. We observe the decrease of RORCs buffer of the TRD detector around 7.5s. It corresponds to the moment where some GDCs store events on disks (see Figure 23). As a consequence, more Central, Minbias, and Dielectron events can be accepted. This causes both detectors TPC and TRD to fill (once more) their RORCs.

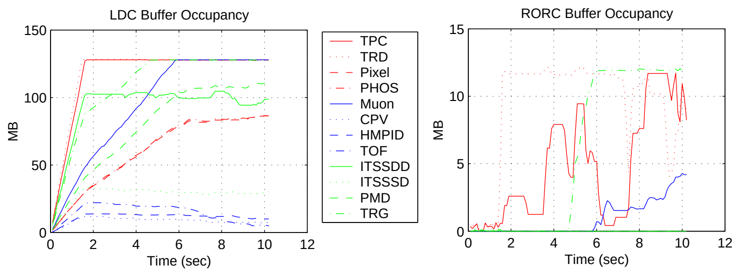


Figure 21: LDC and RORC Buffer Occupancies

Figure 22 shows the distribution over time of the Bandwidth rate and multi-event buffer occupancy of each detector. Table 11 reports the values at 1.5s, 6s, and 10s.

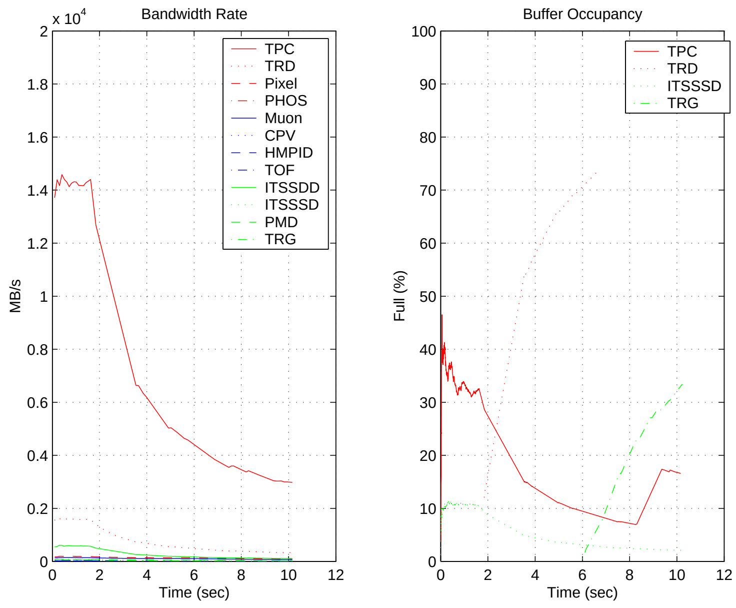


Figure 22: Bandwidth Rates and Multi-event Buffer Occupancy

Bandwidth rates and multi-event buffer occupancies of the detectors are constant in interval [0s, 1.5s]. During interval [1.5s, 6s], bandwidth rates of all detectors decrease.

This is due to the restriction of Central, Minbias, and Dielectron events. We observe that the multi-event buffer occupancy of TRD detector increases after 2s, and reaches 69.4% at 6s. The TRD detector has a buffer of 50 positions, which becomes full only after the RORCs buffer of this detector are full, i.e., after 2s (see Figure 21). Indeed, during the interval [0s, 1.5s], the big size of the TRD multi-event buffer ensures that it gets not filled. During the same interval, the multi-event buffer occupancy of TPC decreases (10%), because the number of Central and Minbias events are sufficiently reduced enabling TPC to send them to the DAQ without filling its RORCs buffers, and without filling the multi-event buffer as frequently as in the interval [0s, 1.5s].

Finally, during interval [6s,10s], Dimuon events are reduced too, due to full RORCs buffers of the TRG detector. Consequently, the multi-event buffer occupancy of the TRG detector increases after 6s, and reaches 31.9% at 10s. The ITSSSD detector has only 1 event slot; we observe that the corresponding occupancy varies from 10% to 2%.

Table 11: Buffer Full and Bandwidth Rate (up to 1.5s, 6s, and 10s)

1.5s	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	32 %	0 %	0 %	0 %	0 %	0 %
Bandwidth Rate	14277MB/s	1588MB/s	143MB/s	35MB/s	23MB/s	23MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer Full	0%	0 %	10 %	0 %	0%	0 %
Bandwidth Rate	187MB/s	583MB/s	85MB/s	19MB/s	76MB/s	57MB/s
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s

6s	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	10 %	69.4 %	0 %	0 %	0 %	0 %
Bandwidth Rate	4584MB/s	509MB/s	106MB/s	11MB/s	7MB/s	7MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer Full	0%	0 %	3.27 %	0 %	0 %	0.45 %
Bandwidth Rate	136MB/s	178MB/s	26MB/s	14MB/s	54MB/s	25MB/s
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s

10s	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	16.8 %	73.9 %	0 %	0 %	0 %	0 %
Bandwidth Rate	2998MB/s	335MB/s	73MB/s	7MB/s	4MB/s	4MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer Full	0%	0 %	2.14 %	0 %	0%	31.9%
Bandwidth Rate	93MB/s	115MB/s	16MB/s	9MB/s	37MB/s	17MB/s
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s

Figure 23 shows the filling of the GDCs buffers. The choice of a GDC for event building is computed on a modulo basis (wrt the event number). The left upper part of the figure shows the filling of the 100 GDCs separately. The right upper part of the figure shows the mean value of the 100 GDCs buffer occupancy, and its evolution in the interval [0s, 10s]. The lower part of the figure shows the variance and the standard deviation (square root of the variance) of the GDCs buffer occupancies. GDCs buffer are unequally filled. Up to 10s, some GDCs have half of their buffer filled, while others have less than 25MB of data in their buffer. At 10s, the mean is of 100MB, the variance is of 3600MB^2, and the deviation from the mean is of 60MB, thus confirming the fact that GDCs buffer are unequally filled. In addition, we see that the deviation increases from 0s to 7s, then it remains constant in the interval [7s,8.5s]. This corresponds to the moment when most GDCs buffer sizes decrease because of the storage of events on disks.

This figure is not complete, since the moment, when GDCs buffers get filled (512MB), and slow the whole system, is lacking. This point could not be reached, due to the fact that the simulation is very slow.

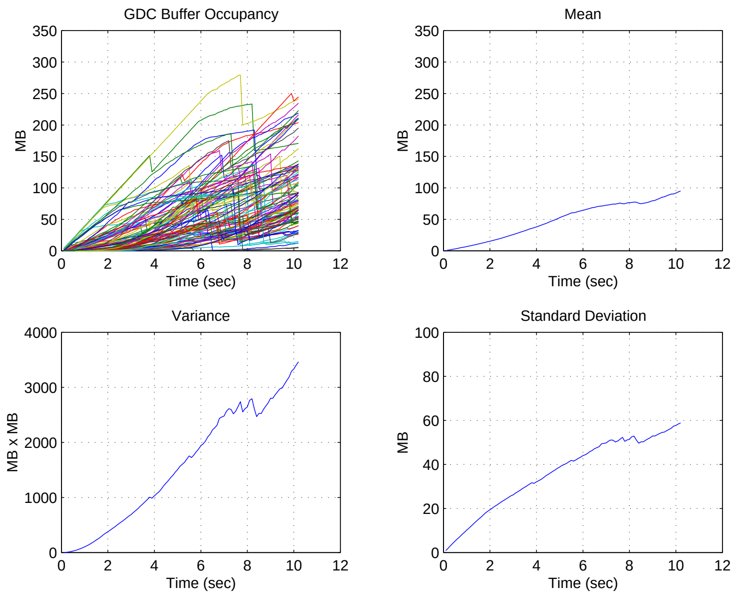


Figure 23: GDC Buffer Occupancies

The upper part of Figure 24 shows the number of events of each type stored on disks during the interval [0s, 10s]. The lower part shows the number of events built in GDCs during the same time interval. The difference between the two graphs is given by the delay needed to transfer the data from the GDCs to the Disks. The last columns of Table 10 gives the corresponding values of the upper part of this figure at 1.5s, 6s, and 10s.

We observe that the completion of Dimuon in GDCs is delayed because of Central and Minbias events being completed simultaneously. This is particulary visible at 4s, 6.5s.

The interval [2.5s,4s] is interesting to observe. Indeed, during this time interval, we see that no Dimuon events have been completed in GDCs, and hence stored on Disks. However, many Dimuon events are being sent to GDCs for event building. The event building of Dimuon events is delayed because events of a bigger size are being sent to GDCs at the same moment. If we consider a given GDC n, it will receive events numbered n, $100+n$, $200+n$, etc. (because of the modulo algorithm). Sub-events corresponding to event $100+n$ and $200+n$ are being sent to GDC n, even though all sub-events corresponding to event n have not completely reached the GDC. Since only one sub-event can be sent to the GDC at once, event n is delayed.

Detectors that succeed in sending sub-events corresponding to event n are then free to send sub-events corresponding to subsequent events ($n+1$, $n+2$, ...). However, in the case of detectors waiting to send event n, sub-events stored in LDCs buffer after event n (corresponding to events $n+1, n+2, ...$) are delayed. These sub-events have to wait for sub-event n to be sent to GDC, before being sent themselves to their corresponding GDC. In addition, bandwidth rate and sub-event sizes vary greatly from one detector to another. Therefore, some detectors are very rapid in sending sub-events to GDCs, while others take more time. Therefore, we observe that some detectors succeed in sending sub-events corresponding to events ranging from n to $200+n$, while others are still sending (or waiting to send) event n.

When events of a big size, such as Central or Minbias events, are completed or almost completed in GDCs, then a lot of Dimuon events succeed in being completed in GDCs. Indeed, the link to the GDC becomes available, and sub-events corresponding to event n are finally sent to GDC n. This frees the subsequent sub-events ($n+1, n+2, ...$) enabling them to be sent to their corresponding GDC. This is what happens for Dimuon events at 3.8s, and 6.5s.

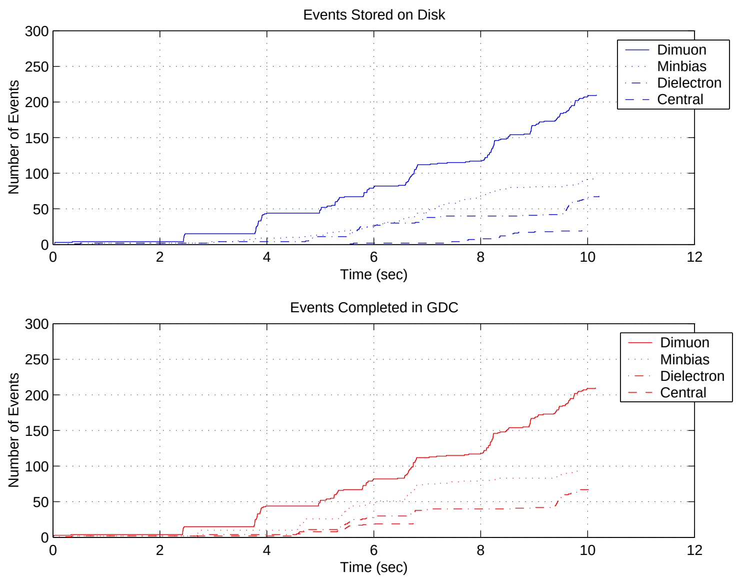


Figure 24: Events Stored on Disk

Foreseen performances for the event building rate are of 4GB/s. Indeed, 100 GDCs collect sub-events from a Switch that transfers them at 40MB/sec. Performances expected for permanent data storage are of 1.25GB/s (50 PDS linked at 25MB/sec to a Switch). The left part of Figure 25 shows the event building rate (upper part) and the corresponding absolute data size (lower part). The right part shows the rate and absolute data size stored on disks. Suprisingly, we see that the expected event building rate of 4GB/s is not reached by the DAQ. It does not go beyond 1.5GB/sec. This poor rate is the logical consequence of the way sub-events are being sent to GDCs, through the Switch (see discussion above regarding Figure 24). The Switch has been specified according to the original ALICE DAQ requirement: sub-events corresponding to different events are sent in parallel to two or more different GDCs, while sub-events corresponding to the same event (and thus being sent to the same GDC), are being sent in sequence. In addition, sub-event $n+1$, stored in a given LDC, has to wait for sub-event n, stored in the same LDC, being sent to GDC n, before being sent itself to GDC $n+1$. We clearly see that such a Switch has a great impact on the event building rate of ALICE. Other simulations, where the Switch has been specified in other way, show that the event building rate can be greatly improved [2]. Allowing for each LDC that sub-events $n+1, n+2, \ldots$ can be sent to their corresponding GDC, even though sub-event n has still to be sent to GDC n, increases drastically the performance. Careful specification of realistic Switches have to be undertaken and their respective performances measured.

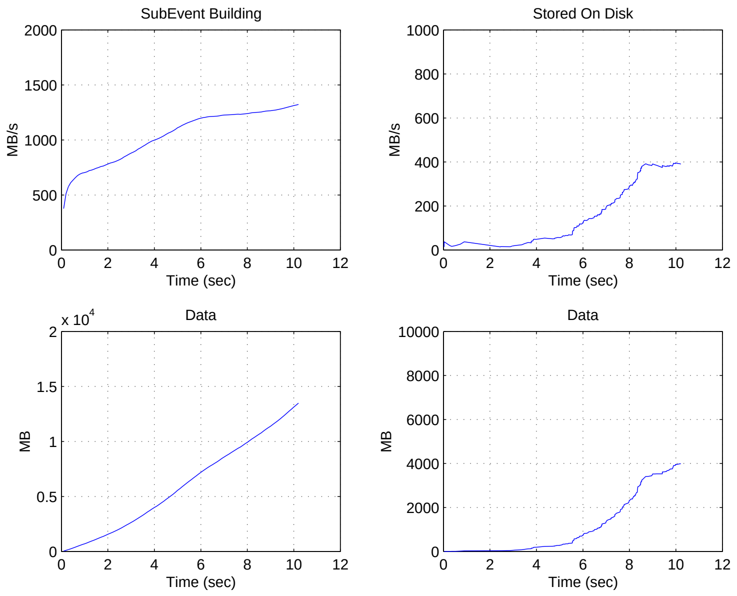


Figure 25: Event Building

6.2.4 Foresight Performances

During the simulation, the system progressively fills the RORCs, LDCs, and GDCs buffers, and checks these buffers for completed sub-events, and completed events. Up to 1.5s of simulation time, both the RAM and CPU time grow linearly. After the limitation of Central and Minbias events (1.5s), the RAM grows less faster, while the simulation speed decreases: the CPU time grows linearly but with a lower gradient.

Even though the simulation of the more abstract specification of Section 5 runs faster than the original one (described in Section 4), the simulation speed is too slow. Therefore, as mentioned before, the moment when the system reaches its slowest point (GDCs full), and then runs constantly has not been reached.

Foresight proposes a tool, called “codercpp”, which transforms a Foresight specification into a C++ executable. The simulation should then run faster than in the interpreted case. However, in the case of our specification, the compiled version is *slower* than the interpreted one.

The simulation described in this section ran on a SunFire 280R server, with 2 processors at 750MHz, with 1GB of RAM. It used one processor, and consumed 603 hours of CPU time, 277MB of RAM to reach 7.7s of simulation time.

6.3 Restricted DDL Bandwidth - 12 Detectors/New Trigger Algorithm

The trigger system defined in the simulations described so far produced signals depending on P/F protection, busy status of the detectors, and on the L0 trigger inputs given by Table 1. This table shows a high rate of Minbias events, explaining the high rate of L2 Minbias signals observed during the simulations, which is far from the expected 20Hz of Minbias events at L2. In order to reduce the rate of L2 Minbias signals, an algorithm that cuts Minbias events at the L1 level has been investigated.

The specification of the trigger system of Section 6.2 has been modified in the following way: the trigger limits L1 signals for Central and Minbias events, if the L2 signal is above 20 Hz. Parameters are those of Table 8 and 9, with L0 trigger inputs given as before by Table 1.

6.3.1 Results

Table 12 shows the trigger outputs rates observed at 2.5s, and 3.8s. Figure 26 shows the corresponding progression over time of the trigger signals.

Table 12: Event Rates - 12 Detectors/New trigger (up to 2.5s and 3.8s)

2.5s	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk
Central	186	27	20	0
Dimuon	353	451	540	9
Dielectron	270	363	369	5
Minbias	1112	32	20	1
Misc	314	104		
Sub-total	2133			
Others	1997			
Total	5962			

3.8s	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk
Central	129	23	18	0
Dimuon	244	308	374	82
Dielectron	187	250	256	45
Minbias	769	29	20	2
Misc	218	78		
Sub-total	1547			
Others	1997			
Total	6000			

During the interval [0s, 2.5s], the L0 signals grow linearly. The L1 signals for Central and Minbias events are strongly reduced, and the corresponding L2 signals remain under 20 Hz (thanks to the cut at L1). During this period, L1 and L2 signals for Dielectron and Dimuon events grow linearly. We observe a higher rate of Dielectron events (369Hz); these events clearly gain from the reduction of Central and Minbias events.

After 2.5s, we observe that L0 signal of all event types are limited. L1 and L2 signals for Dimuon events and Dielectron events are seriously limited, while L2 signals for Central and Minbias events fall under 20 Hz.

As shown in Figure 27, during interval [0s, 2.5s], detectors that are not involved in Dimuon events do not have their respective RORCs and LDCs buffer full. This is due to the 20 Hz limitation. At 2.5s, the TRG and ITSSDD detector have their LDCs buffer full. However, TRG only has its RORCs buffer full, causing the limitation of all event types. Since the TRG detector represents input from trigger detectors, and it creates the bottleneck in this simulation, it should be checked if the estimated parameters for TRG are pertinent.

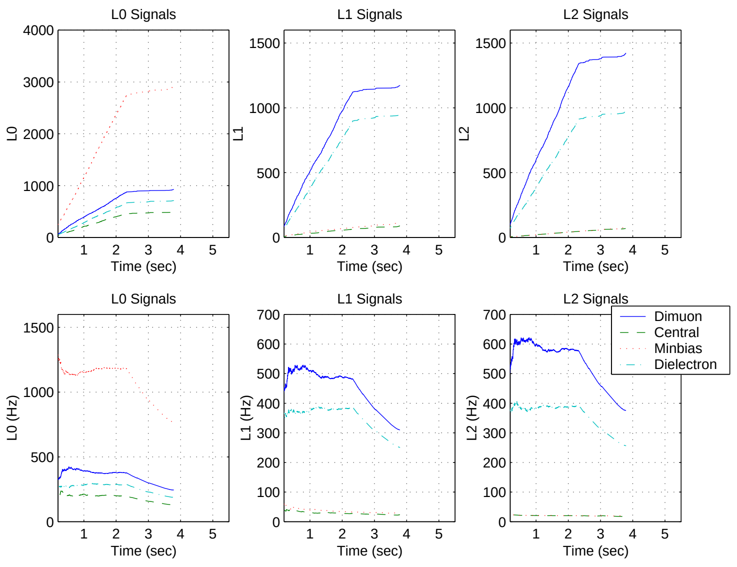


Figure 26: L0/L1/L2 Triggers: top shows cumulative diagrams, bottom shows signal rates

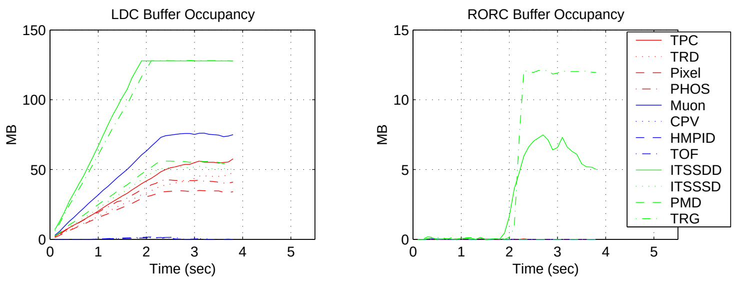


Figure 27: LDC and RORC Buffer Occupancies

Figure 28 and Table 13 report the evolution and values of the bandwidth usage and multi-event buffer occupancies of each detector.

Bandwidth rates of the detectors are constants in interval [0s,2.5s]. For TPC and TRD they are much lower than those observed without cutting Central and Minbias at L1: 3755MB/s for TPC (i.e., 21% of bandwidth), and 304MB/s for TRD (i.e., 17% of bandwidth). Consequently, we observe a lower occupancy of the multi-event buffers (0.4% for TPC, 0% for TRD). During this period, the limitation of Dimuon and Dielectron events is due to the ITSSSD detector. Indeed, we observe that the multi-event buffer occupancy of ITSSSD detector is of 10.8%. This is due to the fact that this detector has a multi-event buffer size of one slot only.

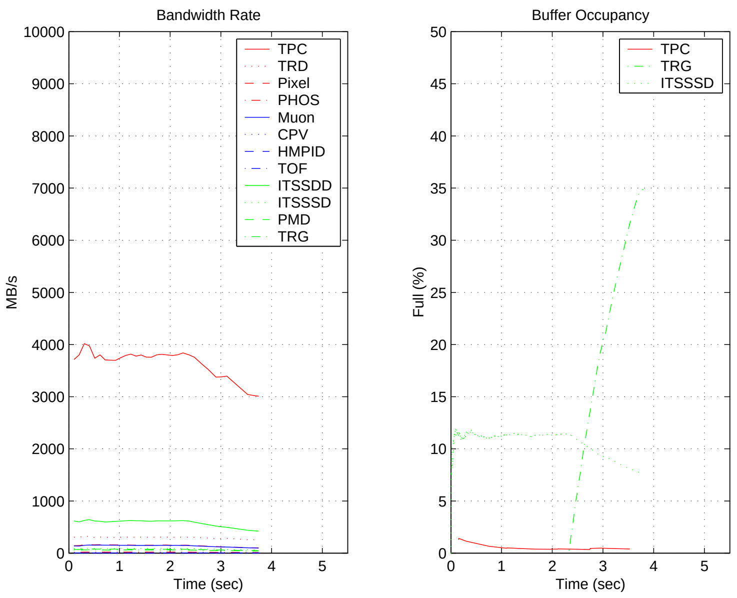


Figure 28: Bandwidth Rates and Multi-Event Buffer Occupancy

After 2.5s, bandwidth rates decrease. TPC, TRG and ITSSSD only fill their multi-event buffer. TRG fills its RORCs buffer after 2.5s, we observe an increase of its multi-event buffer occupancy (34.8%) causing the reduction of Dimuon events.

Table 13: Buffer Full and Bandwidth Rate (up to 2.5s, and 3.8s)

2.5s	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	0.4 %	0 %	0 %	0 %	0 %	0 %
Bandwidth Rate	3755MB/s	304MB/s	141MB/s	4MB/s	3MB/s	3MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer Full	0%	0 %	10.8 %	0 %	0%	0 %
Bandwidth Rate	146MB/s	595MB/s	87MB/s	19MB/s	70MB/s	58MB/s
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s

3.8s	TPC	TRD	Muon	TOF	CPV	HMPID
Buffer Full	0.4 %	0 %	0 %	0 %	0 %	0 %
Bandwidth Rate	3009MB/s	256MB/s	98MB/s	4MB/s	2MB/s	2MB/s
Allocated	18000MB/s	1800MB/s	2800MB/s	2000MB/s	2000MB/s	2000MB/s

	ITSPixel	ITSSDD	ITSSSD	PHOS	PMD	TRG
Buffer Full	0%	0 %	7.7 %	0 %	0%	34.8%
Bandwidth Rate	101MB/s	422MB/s	62MB/s	13MB/s	49MB/s	41MB/s
Allocated	2000MB/s	7200MB/s	800MB/s	400MB/s	600MB/s	100MB/s

Figure 29 shows the GDCs buffers occupancy. We observe the same high standard deviation as in the other simulations. Its value is lower (30MB instead of 60MB).

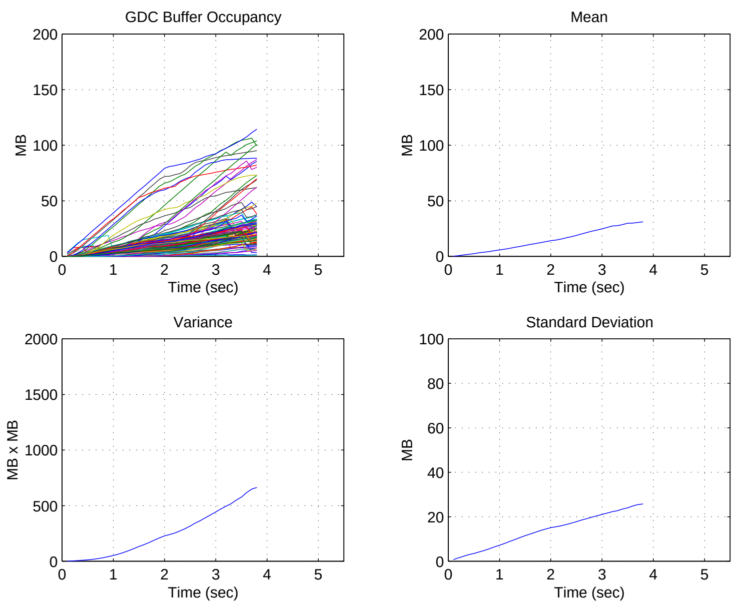


Figure 29: GDC Buffer Occupancies

Figure 30 shows the event storage on disks during the interval [0s, 3.8s]. Corresponding values at 2.5s, and 3.8s are reported in Table 12.

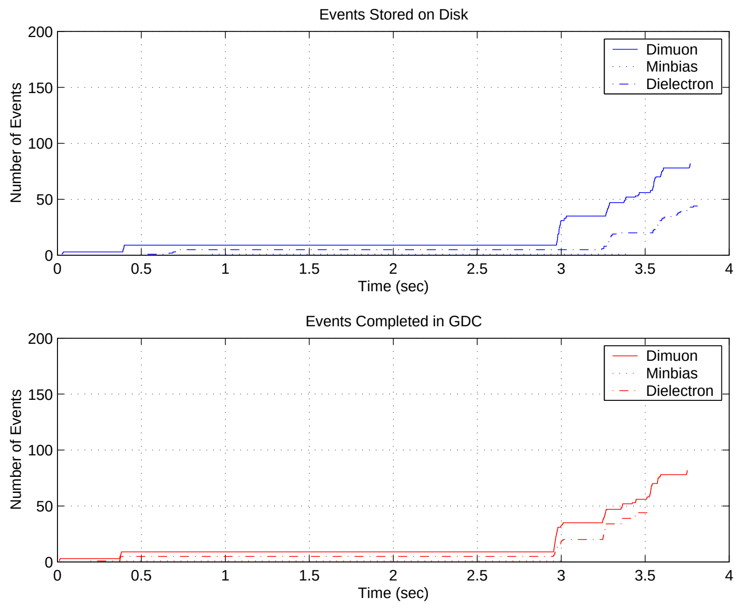


Figure 30: Events Stored on Disk

For the same reasons as those explained in the previous section, event building in GDCs is under 1GB/s.

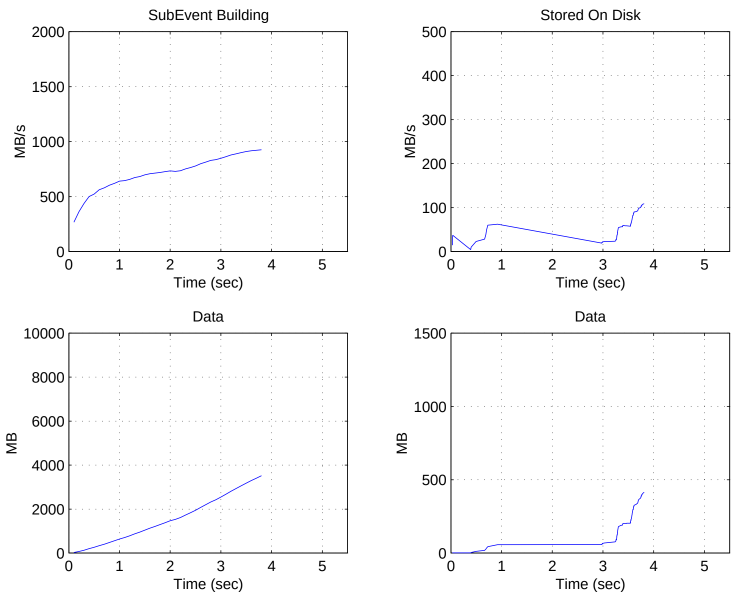


Figure 31: Event Building

6.3.2 Foresight Performances

This simulation ran on a Sun workstation UltraSparc10 with 384MB of RAM. It consumed 410 hours of CPU time, needed 225MB of RAM, to reach 3.8s.

6.4 Central events only - no limit for LDCs

The simulation explained in this section is derived from the specification including 12 detectors (see Section 6.2). It has been undertaken in order to observe the progression of the event building rate, when a high rate of data enters into LDCs. As mentioned before, we suspect that the chosen algorithm, for sending sub-events in chronological order to GDCs, combined with the choice for GDCs (modulo) creates a bottleneck causing the event building rate to be much lower than expected.

Therefore, the specification of Section 6.2 has been put under the following conditions: (1) the trigger system receives input for Central events only (at 6000 Hz). It signals Central events on the basis of past/future protection and the busy status of the detectors; (2) LDCs have a buffer size of 1280MB ensuring that LDCs will not be filled during the simulation. In that manner, we ensure that a high rate of data is available on the input side of the Switch. The event building rate that will be measured results then exclusively from the way data is sent from LDCs to GDCs.

6.4.1 Results

Since no limitation is imposed by the DAQ (LDCs are never full), all signals grow linearly, as shown by Figure 32. The corresponding rates are given by Table 14.

Table 14: Event Rates - Central events only (up to 7.6s)

	L0 (Hz)	L1 (Hz)	L2 (Hz)	Disk	in GDC
Central	383	374	228	1	31

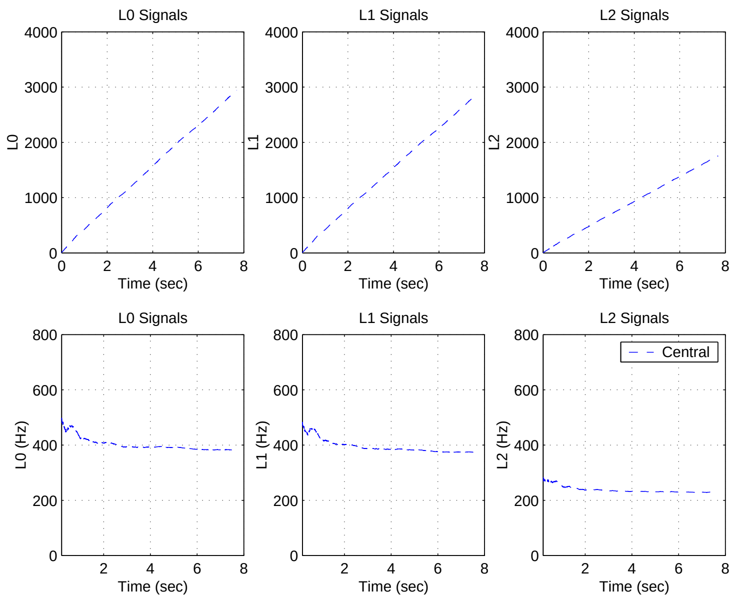


Figure 32: L0/L1/L2 Triggers

As shown by Figure 33, after a period of stabilization, bandwidth rates remain constant. We see that the TPC multi-event buffer occupancy decreases until it reaches 35% at 6s, then it remains constant. During the same period, the TRD multi-event buffer occupancy increases and reaches 65% at 6s, before remaining constant, while the multi-event buffer occupancy of the ITSSSD detector remains around 8%.

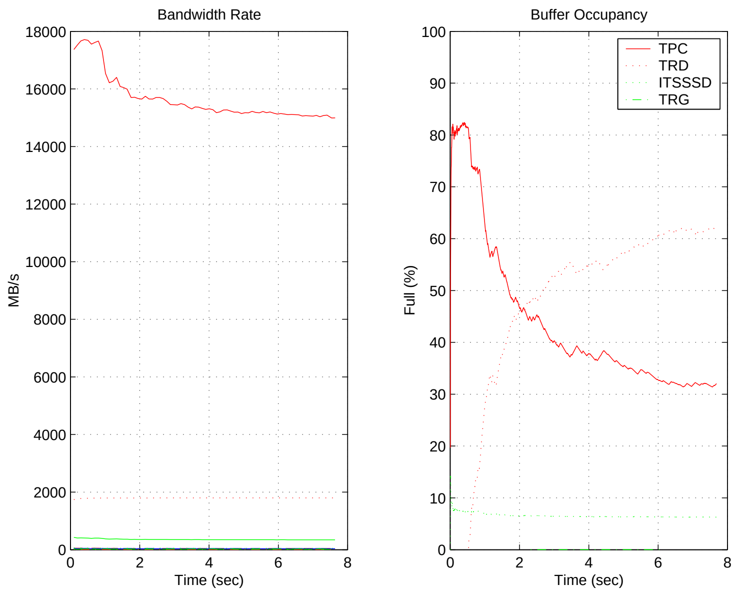


Figure 33: Bandwidth Rates and Multi-Event Buffer Occupancy

Figure 34 shows that LDCs buffer are filled linearly, while RORCs are never filled.

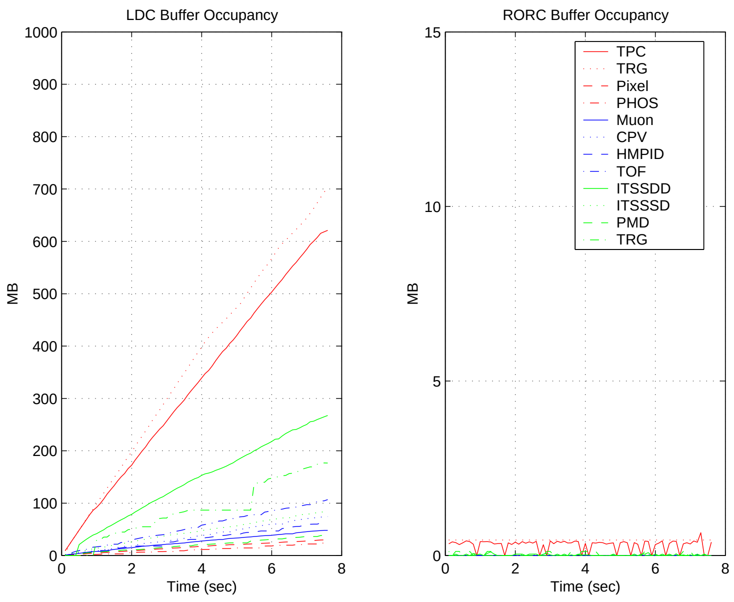


Figure 34: LDC and RORC Buffer Occupancies

On Figure 35, we see the first event being stored on disk at 5s, since one of the GDCs buffer decreases at that time. The standard deviation is of 65MB at 7.6s, and it continues to grow.

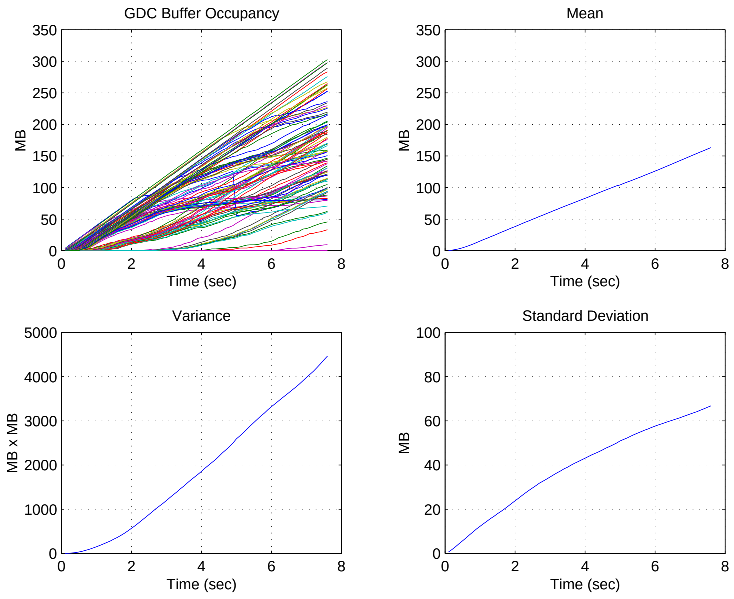


Figure 35: GDC Buffer Occupancies

As mentioned before, the aim of this simulation is to verify if the event building of 4GB/s can be reached when LDCs are not causing any bottleneck. We observe that similarly to the other simulations, the event building is far from 4GB/s, it remains around 2GB/s. This fact confirms the idea that the limitation of the event building rate is due to the way how sub-events are sent from LDCs to GDCs.

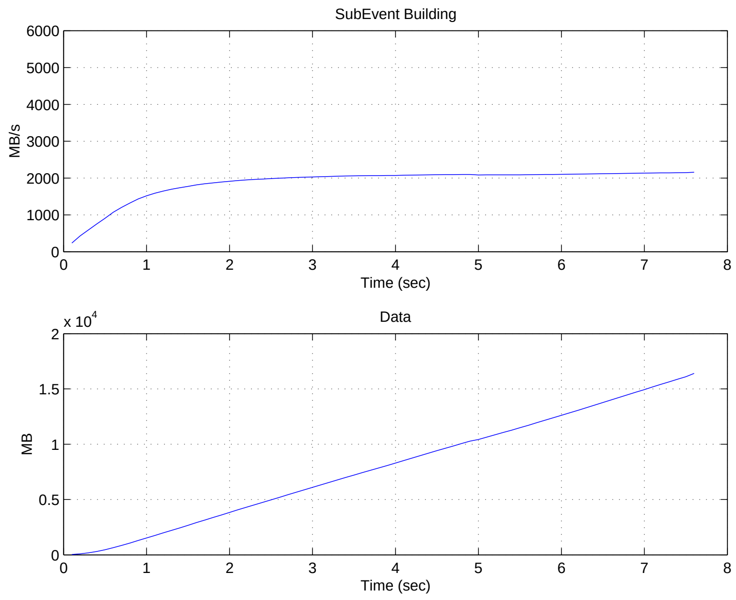


Figure 36: Event Building

6.4.2 Foresight Performances

This simulation ran on a Sun server Ultra Enterprise 3500, with 8 processors, and 3.8GB of RAM. The simulation used one processor only. It consumed 343H of CPU time, 291MB of RAM, to reach 6.6s.

6.5 Discussion

In the case of a simulation with a restricted DDL bandwidth (Section 6.2), we observe first a linear progression of the L0/L1/L2 signals with rates equivalent to those with full DDL bandwidth. At a certain point (around 1.5s), LDCs buffer of some detectors get full (TRD and TPC), causing RORCs and multi-event buffers to get full (TRD). As a consequence, Central, Minbias, and Dielectron events are seriously reduced. Dimuon events gain from this reduction during a certain period, i.e. until LDCs buffers of detectors of the Dimuon class get full too (TRG). This causes a reduction of all event classes.

In the simulation with a cut at L1 (Section 6.3), Central and Minbias events rates are reduced to 20Hz by the trigger algorithm. Dielectron events gain from this reduction. Dimuon events are reduced (around 2.5s) by TRG detector having RORCs full.

More generally, there are two main reasons why detectors involved in Central and Minbias events are busy (have a multi-event buffer full). First, their DDL bandwidth is not sufficient (RORCs are not full, but data cannot be sent more rapidly to RORCs). Second, there is a backpressure of the DAQ. Two cases occur: (1) RORCs and LDCs buffers are full. The bottleneck is caused by the Switch that transfers sub-events to GDCs; (2) some GDCs are full. Sub-events cannot leave LDCs.

LDCs buffer occupancy depends on the event size and on the number of DDLs allocated to a detector. According to detector parameters of Table 8, TRD needs 4.44ms

(8MB/(1800MB/s)) for transferring a Central event from the multi-event buffer to the RORCs, while TPC needs 3.6ms (66MB/(18000MB/s)). Pixel employs 0.0095ms (0.19MB/(2000MB/s)) for a Central event, and 0.005ms for a Dimuon event; the Muon detectors needs 0.0053ms (0.15MB/(2800MB/s)) for Central and Dimuon events; and the TRG detector 1.2ms for Central events and 0.2ms for Dimuon events. The detectors causing event limitation are the detectors which employ more time for transferring data along the DDLs: TRD in the case of Central events, and TRG for Dimuon events (with 12 detectors).

In all these simulations, the expected event building rate of 4GB/s is far from being reached. This is due to the GDC choice (modulo algorithm), and to the original ALICE requirement about the way sub-events are sent from LDCs to GDCs (chronological order). This fact is confirmed by simulation of Section 6.4. In this simulation, no backpressure is exerted on the detectors (high LDCs buffer size), LDCs receive a high rate of data. However, the event building rate does not go beyond 2GB/s in GDCs. Preliminary results show improvement when some algorithms are changed (see Ptolemy results [2]). However, Switches are complex, and need careful attention. Therefore, simulations involving realistic specification of the Gigabit Ethernet Switch, as well as other algorithms for GDC choice have to be undertaken, in order to establish how the event building rate can be enhanced.

7 Method

The Foresight specification has served to define the functional behaviour of the DAQ system and of its sub-systems. The use of this tool, which requires to "implement" algorithms, and not only define interfaces, has helped to clearly establish:

- the functional behaviour of the whole system;
- the functional behaviour of the sub-systems;
- the interfaces (input/output flows) among sub-systems;
- the parameters and their values.

Foresight provides the possibility to "execute" a specification. This is useful for showing how the system works, and to perform verification of the specification.

However, in the case of the DAQ, the execution must run a certain time (∼100 seconds of simulation time) before reaching interesting results. The Foresight tool does not scale in long term simulations, e.g, the simulation takes too much time before GDCs are full. Therefore, some results could not be reached.

In order to overcome this problem, the initial specification has been made more abstract: the same observable behaviour has been kept, while the internal behaviour has been made more abstract. A big improvement in the simulation has been realised. But, it is still not sufficient.

According to the initial specification a compiled simulation has been obtained by implementing, in C++, the specification into the Ptolemy hierarchical environment, which is an open and free software tool developed at Berkeley. The results obtained by the Ptolemy and the Foresight executions (on the first 12 seconds) are similar. Therefore, we can consider that there is a behavioural equivalence between the two executions, and that the Ptolemy version is a correct implementation of the Foresight specification.

The compiled Ptolemy implementation runs faster than the interpreted Foresight execution. Simulation times over 100 seconds can be easily reached, and several executions have been conducted using different parameter sets. Implementation details and results of the Ptolemy implementation can be found in [1].

Figure 37 summarizes the above described approach.

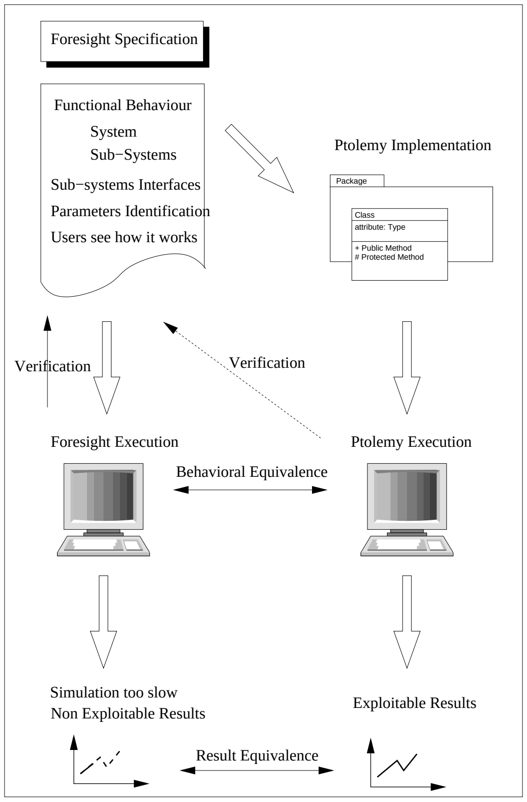


Figure 37: Method Adopted

8 Conclusion

The use of a formal specification and its related simulation has some advantages, but also limitations. In the case of the ALICE DAQ, the establishment of the formal specification helped in clearly defining the interfaces among sub-systems, the functional behaviour of the sub-systems, and identifying the parameters and their values. The execution of the specification enabled to verify the specification and to see how the whole system works.

In the case of complex systems with thousands of messages exchanged per second, long term simulations become impossible to envisage. The unambiguous definition provided by the specification constitutes a valuable document that can be used as a basis for other works or simulations. Indeed, it enabled the Ptolemy implementation to be rapidly realised, in order to reach the expected results.

Architecture Alternatives

In the current model, the L2 trigger outcome reaches the detectors at $t_0 + 89.2\mu s$. It should be interesting to see, if sending the L2 outcome as soon as it is available, changes significantly the Dimuon rate.

The current algorithm for the GDC choice takes into account the trigger number, but not the availability of the GDCs. This algorithm is currently used for the DAQ sub-system in ALICE. As observed during the simulations, GDCs get unequally filled. With such an algorithm, it happens that LDCs are blocked, waiting for a specific GDC, which is receiving another event, while there is another GDC completely free. A more interesting algorithm would be to take into account the availability of the GDCs.

Investigation of new algorithms for sending sub-events from LDCs to GDCs, using faster Switches need to be performed in order to measure the performance of the event building rate.

More Detailed Model

Till now, we have focused on the functionality of the different sub-systems of the ALICE DAQ and trigger systems. The model could be enriched with a more detailed specification of some existing DAQ components such as the ALICE Detector Data Link (DDL) or the DAQ software framework (DATE).

References

[1] T. Anticic. User Guide for the ALICE DAQ Simulation. Technical Report ALICE-INT-2001-0xx to appear, CERN, 2001.

[2] T. Anticic, G. Di Marzo Serugendo, , P. Vande Vyvre, O. Villalobos Baillie, and P. Jovanovic. Specification and simulation of the ALICE Trigger and DAQ system. In *Proceedings of Computing in High-Energy and Nuclear Physics (CHEP'01)*. IHEP, Beijing, China, September 2001.

[3] Foresight-Systems, inc., Austin, TX 78759. *Foresight User's Guide*. http://www.nuthena.com.

[4] P. Vande Vyvre. Physics Requirements for the ALICE DAQ System. ALICE Internal Note/DAQ ALICE-INT-2000-30, CERN, 2000.

[5] O. Villalobos Baillie, D. Swoboda, and P. Vande Vyvre. Data Acquisition, Control and Trigger: Common Report for the preparation of the ALICE Technical Design Reports. Internal Note DAQ, DCS, Trigger, ALICE/98-23, CERN, 1999.