CDF/DOC/TRIGGER/CDFR/6259
Version 1.0
September 5, 2002

Run IIb Upgrade for CDF L2 Decision Crate

W. Ashmanskas,† R. Blair,‡ M. Bogdan,† J. Dawson,‡ R. Demaat,$^{\diamond}$ H. Frisch,† K. Hahn,* P. Keener,* J. Kroll,* S. Kwang,† J. Lewis,$^{\diamond}$ C. Lin,$^{\diamond}$ T. Liu,$^{\diamond}$ A. Meyer,$^{\diamond}$ J. Patrick,$^{\diamond}$ S. Pitkanen,$^{\diamond 1}$ J. Proudfoot,‡ B. Reisert,$^{\diamond}$ H. Sanders,† M. Shochet,† F. Spinella,* R. van Berg,* P. Wilson,$^{\diamond}$ and P. Wittich*

‡ *Argonne National Laboratory,*
$^{\diamond}$ *Fermi National Accelerator Laboratory,*
* *INFN,*
† *University of Chicago,*
* *University of Pennsylvania*

Abstract

We describe an upgrade for CDF's L2 decision crate for Run IIb. The system is based on the Pulsar board currently under construction for the L2 test stand. The goal of the system is to have a high data bandwidth and, at the same time, to be highly modular, maintainable and flexible, and to use off-the-shelf hardware whenever possible. The modularity is achieved by building a universal interface board that can be interconnected in a variety of ways. Maintainability is achieved with this interface card as the only custom component in the system. Off-the-shelf components include the CERN S-Link interface cards and commodity processors. This approach is designed with three goals. Foremost, the design must have sufficient safety margin and flexibility in performance to meet the challenges of the high-occupancy Run IIb triggering environment. Second, ease of maintenance by using a small amount of custom hardware with built-in self-test capabilities. Finally, the design has a simple upgrade path with externally supported links and commercial processors. This note was previously released as a chapter of the Run IIb TDR[1].

[1]Also associated with the University of Helsinki.

Contents

List of Figures

List of Tables

1 Introduction and Motivation

In this note, we describe an upgrade for CDF's L2 decision crate for Run IIb. We start with an overview of the CDF Run II trigger, and then describe the requirements for the L2 decision crate, including the various data paths, sizes and timings. We briefly describe the Run IIa system. Next, we explain what the requirements are for the next phase of CDF (Run IIb) and how we intend to meet those requirements, incorporating lessons learned from the Run IIa project. We conclude with a brief summary.

1.1 CDF Trigger architecture in Run II

Fig. 1 shows part of the CDF Run II three level trigger system. Level-1 (L1) and Level-2 (L2) use custom-designed hardware to find physics objects based on subsets of the detector information, as shown in the Figure. Level-3 uses the full detector resolution to reconstruct complete events in a processor farm. The goal of each stage in the trigger is to reject a sufficient fraction of the events to allow processing at the next stage with acceptable dead time.

The L1 system is a synchronous 40 stage pipeline. When an event is accepted by the L1 trigger, all data is moved to one of four L2 buffers in the front end electronics, and trigger data is sent to the asynchronous L2 system. Here, some limited event reconstruction is performed and a final L2 decision is performed by custom processors in what is called the "Global Level 2 Trigger" in the diagram. It is this subsystem of the L2 trigger that we propose upgrading for Run IIb.

1.2 Level 2 Trigger Functionality and Requirements

By way of a review, we will describe the requirements of the L2 decision crate in Run II. The L2 decision crate makes the final L2 trigger decision based on primitives that have been created in the L1 trigger, and those found in two other L2 sub-systems, the Silicon Vertex Tracker (SVT) and the L2 Calorimeter (L2Cal). Additionally, the ShowerMax information is first available to the trigger in L2,[2] and an improved ϕ measurement is available in the μ trigger system. L2 also has at its disposal all trigger objects used in L1, such as XFT tracks, ΣE_T and $\not{E}_T$information, and the full description of the L1 decision in the form of the 64 L1 trigger bits.

These primitives must be quickly acquired and merged into physics objects in order to allow the trigger to make its decision. For instance, electrons are found by matching tracks to calorimeter clusters and the ShowerMax system. Table 1 shows the connection between some of the physics objects used in L2 and which trigger primitives are required to make them. After the primitives are acquired, the trigger applies basic kinematic requirements on the newly created physics objects or correlated sets of physics objects, and counts how

[2]The ShowerMax system is labeled as XCES in Fig. 1. In L2, the XCES information is received by RECES cards. For the purpose of these notes, the two names will be used interchangeably.

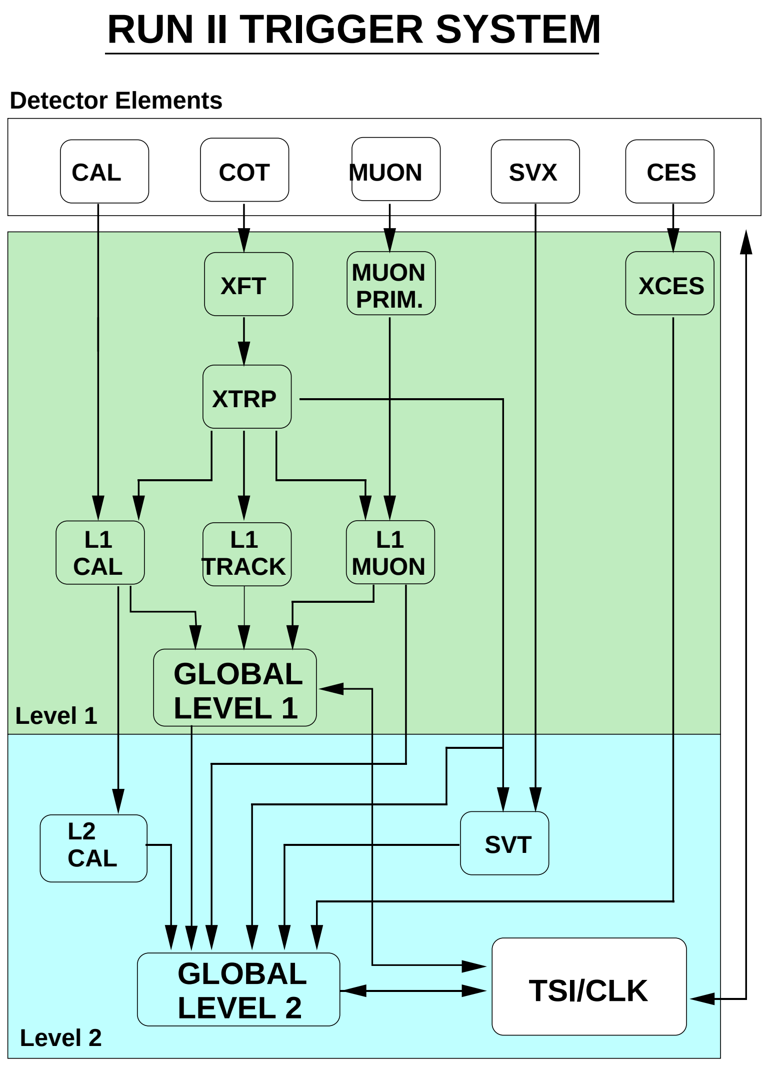

Figure 1: CDF Run II Trigger architecture, adapted from Reference [2]. Note that the L3 farm is not shown in the picture.

	L1	XTRP	SVT[†]	L2 Cal[†]	L2 Iso[†]	Shower Max[†]	$\mu^\dagger$	ΣE_T, $\not\!\!E_T$
Tracks	$\star$	$\star$	$\star$					
Jets	$\star$	$\star$	$\star$	$\star$				
e's	$\star$	$\star$	$\star$	$\star$	$\star$	$\star$		
γ's	$\star$			$\star$	$\star$	$\star$		
μ's	$\star$	$\star$	$\star$				$\star$	
τ's	$\star$	$\star$			$\star$			
ΣE_T	$\star$							$\star$
$\not\!\!E_T$	$\star$							$\star$

Table 1: Examples of the physics objects used in L2 decisions. Each row represents a physics object, and each column represents a trigger primitive. For a given row, the trigger primitives that are used to make the physics object are marked with a star ($\star$). Those primitives that are first available in L2 or where new information is available in L2 are marked with a dagger (†). This is not an exhaustive list.

many such objects or sets of objects exist to see if the required trigger criteria are met. Based on these requirements, the job of the L2 trigger decision system can be split into two parts: *loading* data and *processing* data. *Loading* describes how long it takes from a Level 1 accept (L1A) for the data to be available for processing such that a decision can be made. *Processing* describes how long it takes to unpack the data, form objects and make a decision based on simple kinematic cuts on objects or correlated sets of objects.

For Run IIa, the L2 trigger requirements are defined in [2, 3]. The L2 trigger is required to receive data at a L1A rate of ~ 40 kHz, and accepts as many as ~ 300 events per second, with a dead time of $\sim 5\%$. According to [3], attaining a mean loading time of $10\ \mu$s and a processing time of $10\ \mu$s plus an event-to-event exponential tail of mean $1.75\ \mu$s assures that the criteria will be met.

1.3 Overview of the Run IIa L2 Decision Crate and Data Paths

The Run IIa L2 decision crate is designed to operate as a *two stage pipeline*. The goal is to be simultaneously loading one event while processing the preceding event. During the loading stage, the dedicated L2 hardware subsystems find and send downstream calorimeter clusters (L2Cal) and tracks in the silicon detector (SVT). These data are then transferred into the memory of the L2 processor nodes via the interface boards. In the second stage, the data is unpacked and the trigger algorithms are evaluated to make the L2 decision. The final decision is then negotiated with the Trigger Supervisor (TS). Due to the nature of the pipeline implemented in the decision crate, all L2 decisions are made in L1 accept order.

The current L2 decision crate is based on a dual-bus architecture. The crate consists of two distinct classes of boards. The *interface cards* are the connection to the rest of the

Card	MB Type	System	Quantity	Link	VME
L1Int	master	L1 trigger bits	1	LVDS†	no
XTRPList	master	XFT tracks	1	LVDS*	yes
SVTList	master	SVT tracks	1	LVDS*	yes
Clist	master	L2 clustering	1	HotLink	no
IsoList	master	L2 isolation	1	Taxi	no
μlist	master and slave	L2 μ	1	HotLink	yes
RECES	slave	ShowerMax	4	Taxi	no

Table 2: L2 interface cards in the Run IIa L2 decision crate. The XTRPList and SVTList boards differ only in firmware. All told, the crate will ultimately hold ten interface cards of six different types. The link column indicates what type of connection exists for the inputs to these boards. As indicated by † and $\star$, the LVDS cables in the L1 board differ from the TrackList cables. The *VME* column indicates whether a system's data is available via a VME interface. All TL2D readout occurs via processor node's VME readout.

trigger system. They receive trigger primitives from other L1 and L2 trigger subsystems and pass this information to the other type of board, the *processor node*. Currently, there are six different custom interface cards. Table 2 shows a list of interface boards. The connection between the interface boards and the processors is via a custom 128-bit wide backplane referred to as the *Magic Bus* [4]. All trigger data flows from the interface cards to the processor nodes on the Magic Bus; while configuration and readout is via the processor nodes' VME interface. Interface boards can either be Magic Bus masters, Magic Bus slaves, or both. Magic Bus masters broadcast their data onto the Magic Bus, where it is received by all processor nodes simultaneously. Data from Magic Bus slaves can be retrieved by an individual processor on an as-needed basis. This means that data is both pushed into the processors (via direct memory access (DMA)) and pulled from the interface cards in this architecture (via programmed I/O (PIO)). This architecture choice (DMA or PIO) is made to minimize the Magic Bus bandwidth requirements for each interface type.

The processor nodes in Run IIa are custom computers based on 64 bit, 500 MHz DEC Alpha processors (which are obsolete) with both a VME interface and a Magic Bus interface. All processor interactions on the Alpha board occur via a Peripheral Component Interconnect (PCI) bus. VME access to the processor is available via a VME-to-PCI bridge; similarly, Magic Bus access is available via a Magic Bus to PCI interface. The baseline specification for Run IIa calls for four such nodes in the decision crate. However, currently, we have not run with more than one node.

Tab. 3 shows the event data size for various L2 trigger objects at the present luminosity, $\mathcal{L} = 1 \times 10^{31}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$. The numbers are based on [5]. n is an estimate of the mean (high tail) of the Run IIa occupancy based on data. The high tail is defined as the average plus $3 \times$ RMS. Muon and RECES input data are of fixed-length. Only the Region-of-Interest (RoI) data needs to be sent to CPU memory. In the Run IIa system, the muon and RECES RoI data is pulled by the processor node while processing the event.

System	Object	Object Size (bits)	mean(tail) n	Mean (Tail) Size @L2 inputs (bits)	Mean(Tail) Size into CPU mem (bits)
SVT	track	117	1.2(7)	140(819)	same
XTRP	track	21	8.4(31)	176(651)	same
L1	L1 bits	96/evt	fixed	96 (96)	same
CList	cluster	46	1.9 (7)	87(322)	same
Iso	cluster	145	1.9 (7)	275(1015)	same
μ	muon	11K/evt	fixed	11K(11K)	RoI only: < 1K
RECES	shower	1.5K/evt	fixed	1.5K(1.5K)	RoI only:~ 0.1K
Average total data size				13.3K(15.4K)	< 1.9K(4K)

Table 3: Data sizes for various L2 trigger objects at $\mathcal{L} = 1 \times 10^{31}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$. The numbers are based on [5]. n is an estimate of the mean (high tail) of the Run IIa occupancy based on data. The high tail is defined as the average plus 3 $\times$ RMS. Muon and RECES input data are of fixed-length. Only the Region-of-Interest (RoI) data needs to be sent to CPU memory. In the Run IIa system, the muon and RECES RoI data is pulled by the processor node while processing the event. The RoI numbers are a rough estimate.

The current loading time is dominated by the SVT/SVX processing time. While the mean total Magic Bus transfer time is $\sim 5\ \mu$s, the SVT tracks only arrive at the L2 decision crate's front panel $\sim 25\ \mu$s after a L1A, significantly later than the $10\ \mu$s goal. As the processor node waits for all data to arrive before analyzing the event, the total loading time is therefore defined by the arrival of the SVT tracks.[3] The processing time for a single Alpha node is $\sim 25\ \mu$s on average with significant tails; however, it is believed that aggressive optimization of the code and Alpha processor firmware could reduce this processing time to $\sim 15\ \mu$s [5]. This level of performance will allow this system to satisfy the Run IIa needs.

1.4 Requirements and Motivation for a L2 Upgrade

It is becoming clear that CDF needs to be ready for an unexpectedly complicated trigger environment for Run IIb. The Run IIb baseline bunch spacing is 396 ns rather than the previously expected 132 ns. At the expected peak Run IIb luminosity $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$, we will see ten interactions per crossing. This implies that the average data size will increase substantially, and, consequently, the combinatorics will grow in processing multi-object triggers. As a consequence, CDF will need to improve both the loading stage and the processing stage of the L2 decision system.

As the occupancy in the detectors increases with luminosity, the time it takes the L2 clustering and silicon tracking subsystems to find trigger primitives increases, as does the number of primitives found. To maintain the goal of a 10 μs loading stage, the increase

[3]There is a simultaneous effort to speed up the SVX readout and SVT processing time.

System	Object	Object Size (bits)	mean(tail) n	Mean (Tail) Size @L2 input(bits)	Mean (Tail) Size into CPU mem(bits)
SVT	track	117	3 (12)	351(1404)	351(1404)
XTRP	track	21	19(76)	399(1596)	399(1596)
L1	L1 bits	96/evt	fixed	96 (96)	96 (96)
CList	cluster	46	7 (28)	322(1288)	322(1288)
Iso	cluster	145	7 (28)	1015(4060)	1015(4060)
μ	muon	11K/evt	fixed	11K(11K)	RoI only: $<$ 1K
RECES	shower	1.5K/evt	fixed	1.5K(1.5K)	RoI only:~ 0.1
Average data size				14.7K(21K)	$<$ 3K(10K)

Table 4: Data sizes for various L2 trigger objects at $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$. n is an extrapolation of the mean (high tail) of the occupancy to the Run IIb luminosity at $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$ with 10 interactions per crossing. The extrapolation is done by taking the high p_T occupancy of Run IIa data and adding 10 minimum bias events. The high tail is defined as the average plus 3 $\times$ RMS of the high p_T occupancy. This extrapolation is taken from [5], where no attempt has been made to take into account any growth terms. Muon and RECES input data are of fixed-length. Only the RoI data needs to be sent to CPU memory. In the Run IIa system, the muon and RECES RoI data is pulled by the Alpha while processing the event.

in latency and data size of the upstream L2 subsystems must be offset by an increase in effective bandwidth of the L2 decision crate to transfer the trigger primitives into CPU memory.[4] For instance, in the case of L2 clustering, each addtional cluster adds 1 μs of latency to the arrival time of the cluster data into the L2 decision crate. According to Tables 3-4, in Run IIa, the mean arrival time is therefore 2 μs, while in Run IIb, the mean arrival time will be 7 μs.

With the increase in data size, the amount of processing increases for single-object and multi-object triggers. To maintain similar performance as the Run IIa system in the processing stage, the higher demand on the processor must be offset by increased CPU power.

Tab. 4 shows an extrapolation of trigger data size to the expected peak Run IIb luminosity $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$. The extrapolation is done by taking the high p_T occupancy of Run IIa data and adding 10 minimum bias events. The high tail is defined as the average plus 3 $\times$ RMS of the high p_T occupancy. This extrapolation is taken from [5], where no attempt has been made to take into account any growth terms.

In any case, the system must be able to accommodate the Run IIb task force's trigger specification for a high-p_T-only physics program; see Table 5 [6].

[4]Alternatively, one could upgrade the upstream systems to decrease their latency; however, this option is significantly more expensive.

Trigger	L1	L2	L3
	(Hz)		
e/μ	2300	250	22
$b\bar{b}/\tau\bar{\tau}$	4400	130	9
Cal.	2900	117	16
High p_T	15400	264	38
Total	25100	761	85

Table 5: Run IIb task force's estimate of trigger rates at $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$, based on a high p_T-only physics program [6]. This is the stated design goal for the Run IIb trigger.

1.4.1 Uncertainty of the trigger requirements

The process of making an estimate for the increase of the amount trigger data is difficult and is fraught with possibility for error. It is unclear how to scale the current Run IIa environment, with fewer than one additional interaction per crossing, to ten interactions. For instance, non-physics growth terms are as yet unknown and were not included in Table 4. The large uncertainty in the triggering requirements make it clear that a key component of any new L2 decision crate must be the ability to adapt to changing trigger environments. For instance, out-of-order event processing could be used to mitigate the effects of long tails in processing times for particularly busy events, and data could be suppressed early in the trigger chain if it is not needed for the decision to alleviate bandwidth needs.

1.4.2 Long-term maintenance

The long-term maintenance of the trigger crate must be kept in mind. The easiest way of providing long-term maintenance is to minimize the number and types of objects that must be maintained. Relying on commercial or externally supported components in as many places as possible allows us to reduce the load on CDF resources for long-term maintenance. The judicious use of such components also makes for clear upgrade paths of links and processors if the conditions warrant such upgrades.

1.4.3 Testability and Commissionability

As the Run IIa to Run IIb transition is expected to be short, any new system must be commissioned *before* the end of Run IIa and be able to come up with a high degree of certainty. Past experience has shown that for systems without adequate testing capabilities, the commissioning process can be long and require considerable beam time, thereby, impacting the physics running. In order to achieve the goal of a minimum length transition period, any new system should be built with self-testing capabilities.

1.4.4 Summary of Requirements and Motivation for Upgrade

In summary, the obsolescence of the Run IIa L2 processors, the increase in occupancy, the uncertainty of the Run IIb triggering environment and the need to provide long-term maintenance drives the need for an upgrade to the L2 decision crate. A new system must be flexible enough to handle Run IIb challenges and simple enough to assure long-term maintainability.

1.5 The Pulsar Approach

We propose to meet these requirements in a design based on the Pulsar board [7]. In this approach, all trigger fragments are converted and merged into a self-describing data format by a universal interface card. The common data stream is then transferred via a standard link into a commodity processor, where the decisions are made. The only custom element in this system is the universal interface board. The Pulsar is that interface board.

In the next Section, we will describe the Pulsar board-level design, the standard link used and how we combine these elements into the new system. We will show this design can not only meet the required Run IIb bandwidth, but that it is flexible enough to adapt to the evolving Run IIb trigger challenge. Finally, we will show that the system is designed in such a manner as to allow rapid prototyping, commissioning and integration, and require minimal long-term maintenance.

2 Pulsar Approach

The Pulsar project started as a way to provide some of the internal and external test functionality missing from the Run IIa L2 Decision crate. However, the Pulsar board also designed to be a universal interface card. The board has all the L2 decision crate's interfaces and can either sink or source data for each path. The general design philosophy of Pulsar is to use one type of motherboard (with a few powerful modern FPGA's and SRAM's) to interface any user data with a standard link through the use of custom mezzanine cards. This makes Pulsar a *universal interface card*. CERN S-Link [8] is chosen as the standard link for Pulsar. Using S-Link, Pulsar can communicate with commodity processors via commercially available, high bandwidth S-Link to PCI/PMC (**P**CI **M**ezzanine **C**ard) interface cards.

S-Link is a CERN specification for an easy-to-use FIFO-like data link which can be used to connect front-end to read-out at any stage in a data flow environment. It is a standard that defines interfaces of source and destination sides of a point-to-point High Energy Physics (HEP) oriented data link, and it has been used in quite a few HEP experiments (such as NA48 and COMPASS, to name but two) and will be used at future LHC experiments (ATLAS, CMS, LHCb) where high bandwidth is required.

In summary, in the Pulsar approach, we propose to use one motherboard, one custom mezzanine card for HotLink fiber data, one custom mezzanine card for Taxi fiber data, and two CERN S-Link mezzanine cards for sending and receiving S-Link.

2.1 Baseline Pulsar System Configuration

Fig. 2 shows the baseline configuration for a new Level 2 trigger decision crate. We use a total of eight Pulsar boards to receive event fragment trigger data from the L1 trigger, XTRP, SVT, Muon trigger, L2 Calorimeter, L2 Isolation and ShowerMax. These interface versions of the board gather data from each subsystem, package it, and send it downstream. Each Pulsar board has access to the full L1 decision information and the L1 trigger tracks. This data can be used to reduce the data volume, by keying on the L1 information to decide if the data will be needed in the L2 trigger decision, or by only sending region-of-interest data downstream based on, for example, matches between tracks and muon stubs.

It is important to note that the implementation of the L2 Pulsar scheme requires **no** modification of any o f the existing elements that provide data for Level 2. This means that all of the front-end and Level 1 trigger systems that provide data for Level 2 will be unmodified in going to the Run IIb system.

Each Pulsar merges event data fragments from multiple sources into one stream. The final Pulsar, the Global Processor Controller, streams the complete data packet into a CPU where L2 decision algorithms are run. The Global Processor Controller receives the trigger decision from the CPU and negotiates the L2 decision with the Trigger Supervisor(TS). Raw input data for each Pulsar can be saved in one of four DAQ buffers, and upon L2A, all Pulsar boards can make the data available to readout over VME.

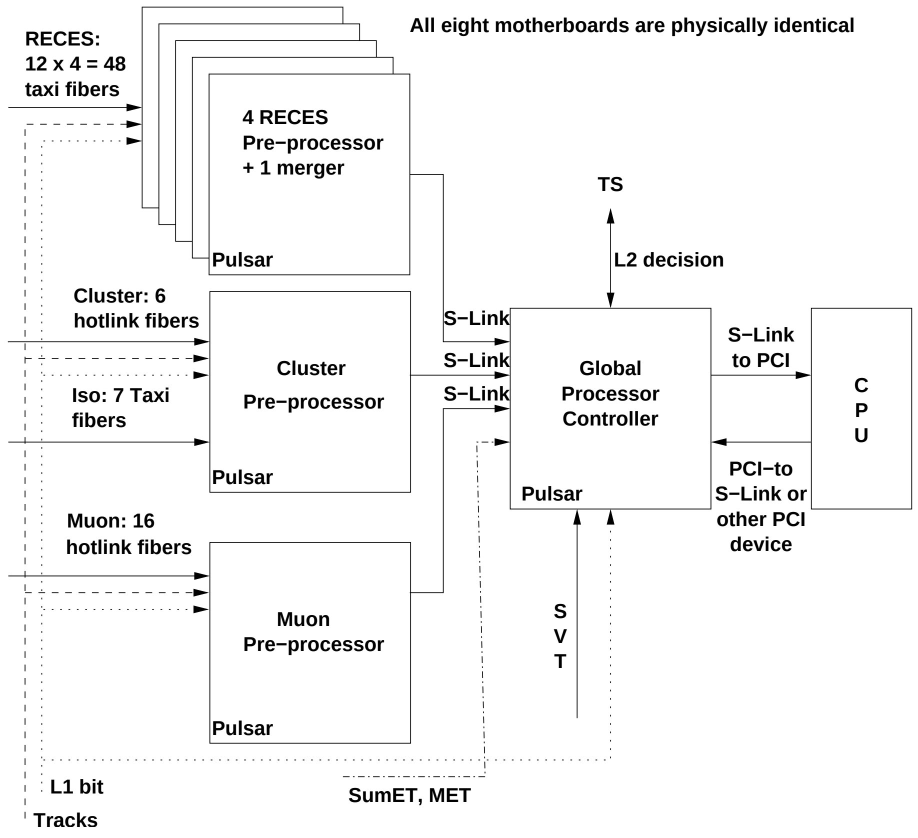

Figure 2: L2 Pulsar baseline configuration. The system lives in one CDF crate with a P2-style pass-through backplane in the P3 position. In total, eight physically identical Pulsar boards receive all L2 data. The final decision is made in a commodity CPU, which is connected to the crate via a S-Link to PCI card. The S-Link to PCI interface has a raw bandwidth of 260 MB/s per channel, while the S-Link bandwidth is 160 MB/s.

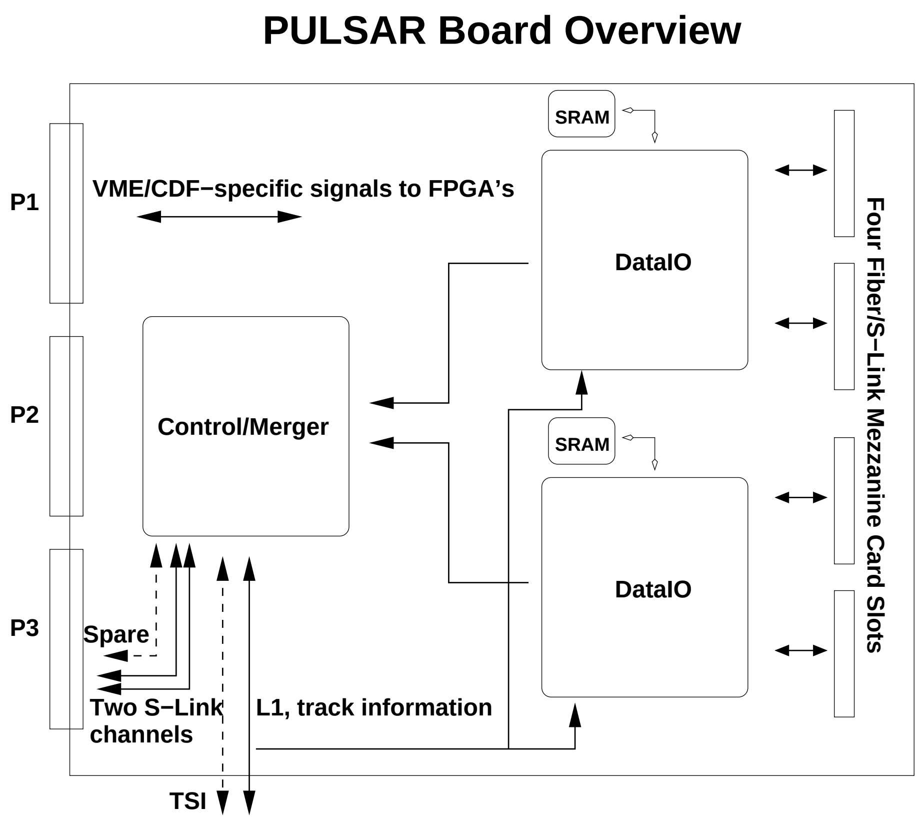

Figure 3: Pulsar board block diagram. The 9U double width board has four mezzanine connections that accept custom mezzanine cards for CDF-specific fiber data and CERN S-Link cards. Three FPGA's of two types receive and merge the data from the mezzanine cards. All FPGA's are visible to the VME backplane and all see the CDF-specific P2 signals. The board is compatible with a standard CDF VME crate. The board also has an interface to the CDF Trigger Supervisor (TS) and inputs for trigger track data and L1 trigger bit data.

The crate itself is a standard CDF VME crate with a pass-through backplane using P2-style (160 pin DIN) connectors in the P3 position. Since each Pulsar board is double width and takes 2 slots, eight Pulsar boards will take 16 slots in the baseline configuration.

2.2 Overview of Pulsar Board Design

Pulsar (as **Puls**er **a**nd **R**ecorder) is a 9U VME board. Fig. 3 shows the Pulsar block diagram. Each Pulsar has four mezzanine card slots. Data is received from these mezzanine card slots, processed and validated. Similarly, L1 and XFT or SVT track data is received via dedicated connections. Finally, data is sent downstream via one of two S-Link channels on the P3 connector to either another Pulsar board or to a commodity CPU. In addition, one connector is compatible with the TS-L2 protocol.

The Pulsar motherboard is dominated by three FPGA's: two *DataIO* FPGA's and one *Control/Merger* FPGA. Each DataIO FPGA provides the interface to two mezzanine cards. The DataIO FPGA's function is to receive data from the mezzanine cards, do any path-specific manipulation and error checking, and convert the data into S-Link format. While it is not strictly necessary, the use of a standard, self-describing data format is very useful. First of all, it makes the firmware design in both DataIO and Control FPGA very similar; secondly, the data will identify itself downstream and one can pack other information (such as buffer number, L1 trigger information and error conditions) in the header and trailer at each stage to allow robust error checking.

The Control FPGA merges data from the two DataIO FPGA's and sends the data to the P3 connector, again in S-Link format. In addition to the 9U VME board, a simple VME64x transition module is required. This card sits in the same slot as the Pulsar. It consists of two Common Mezzanine Card (CMC)[5] connectors for the two S-Link channels driven by the Control/Merger FPGA on the P3 connector. In addition, there are spare signal lines to the P3 connector from this FPGA.

The four mezzanine card slots on the Pulsar motherboard are also CMC compatible and are designed to hold either custom mezzanine cards or S-Link mezzanine cards. There are two types of custom mezzanine cards, Hotlink and Taxi, which are used to interface with other L2 systems.[6] Each custom mezzanine card accepts four fibers, so that a total of 16 fibers can be accommodated per Pulsar board. The ability to plug in S-Link mezzanine cards allows Pulsars to be chained: any Pulsar can accept the output of up to four Pulsars using CERN's Link Source Cards (LSC's) and Link Destination Cards (LDC's).

The L1 and track data is handled somewhat differently than the fiber data. A look at Table 1 reminds us that all trigger objects require L1 trigger decision information (as CDF's trigger is path-driven) and many require tracks. In order to distribute the task of making objects out of trigger primitives, L1 bits and tracks are distributed to each FPGA. The L1 bits can be used to suppress data that isn't required for a trigger decision at an early stage. Track data can be used to only pass data in a region of interest downstream, thereby again reducing the data size, or, in a more ambitious approach, to create physics objects.

We choose Altera 20K400 APEX FPGA's in a 652 pin BGA package for both the DataIO and Control/Merger FPGA. This choice is largely driven by IO requirements, as this chip has roughly 500 User-IO pins. In addition to an internal 26KB memory, each DataIO FPGA also is connected to a $128K \times 36$ fast SRAM (CY7C1350).

The DataIO FPGA and Control/Merger FPGA are very similar in their functionality. Each takes two data streams and merges them into a common stream. The DataIO FPGA also must accommodate the differences between various mezzanine cards and do any necessary pre-processing; however, this functionality can be cleanly separated from the overlapping functionality. The core firmware is similar in all FPGA's on all incarnations of the

[5]CMC is an IEEE draft standard for a family of mezzanine cards designed to be used interchangeably on VME, VME64, VME64x and CompactPCI cards. P1386/Draft 2.4a.

[6]See Table 2 for a listing of which systems use which links.

Pulsar in the new system. See Sec. 2.3 for more details.

We will briefly describe how a Pulsar based system will accommodate each L2 data path. Fig. 2 shows their position in a baseline Pulsar based system configuration.

2.2.1 Pulsar as a pre-processor for the muon data path

Level 2 Muon trigger data is transferred via 16 Hotlink fibers into the Level 2 decision crate. Each muon word is packed into 4 hotlink 8-bit words, though only 24 bits are significant. The baseline implementation packs this data into 32-bit words and sends it downstream for analysis.

Because the muon data is not zero-suppressed upstream, the data volume is large- about 1.3KB per event. There are several strategies for reducing the data volume. The simplest way to suppress muon data is to check Level 1 trigger bits and see if muon data is needed for the Level 2 decision, and, if not, send an empty packet downstream. Another option is to zero-suppress the data itself. Finally, one also has the option to use the on-board SRAM as a look-up table to match tracks with muon stubs and only send those stubs that have matches.

One Pulsar board can receive all 16 fibers with four Hotlink mezzanine cards.

2.2.2 Pulsar as a pre-processor for the Cluster and Isolation data paths

As with the muon case, the clustering data arrives on HotLink mezzanine cards. The cluster data is of variable length; information from one cluster is spread over several fibers. The cluster trigger data is transferred to the Level 2 decision crate via 6 hotlink fibers and one LVDS cable. One cluster's worth of information is encoded into 6 hotlink words per fiber over a total of 6 fibers. The information from the LVDS cable marks the end of the event. One Pulsar board can receive the data via two mezzanine cards and all CLIST data will be visible to one DataIO FPGA. The DataIO FPGA can then generate the cluster summary information and convert the data into S-Link format.

The Isolation data path case is similar to the cluster path. Here, six Taxi fibers carry the data, and one Taxi fiber carries the control information such as end-of-event bits. One DataIO FPGA can receive the Isolation trigger via two Taxi mezzanine cards.

One Pulsar board with two Hotlink mezzanine cards and two Taxi mezzanine cards can be used to receive both Cluster and Isolation data.

2.2.3 Pulsar as a pre-processor for the RECES data path

The RECES data path has 48 fibers, each fiber covers one wedge (east or west side) and the data is of fixed length. We plan to use four Pulsar boards each with 4 Taxi mezzanine cards to sink the RECES data and one additional Pulsar with 4 S-Link mezzanine cards to merge all the RECES data into one data stream. If the large RECES data volume is an issue, strategies similar to those described in the muon case can be used to reduce the data volume.

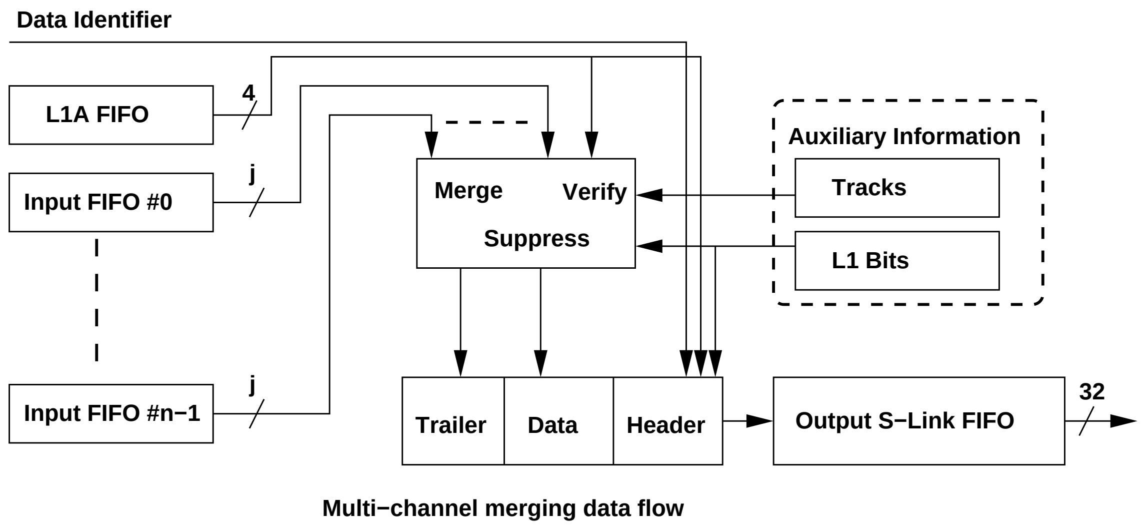

Figure 4: Basic merger firmware data flow. The data identifier is stamped into the header to show the data's origin. The input FIFO's can be S-Link channels, fiber data from the mezzanine cards, or the output of a previous merging stream, as long as it is in a FIFO. For DataIO merging, the input data streams are $j = 8$ bits wide. For S-Link input data streams, $j = 32$. The total number of streams n is two in the case of S-Link and the Controller/Merger FPGA and can be up to eight in the DataIO FPGA with fiber mezzanine cards.

2.2.4 Pulsar as a processor controller

The Processor Control Pulsar will act as a S-Link merger to merge all the data from S-Link LDC mezzanine cards. It will also merge the SVT data and pack data from the whole event into a single S-Link package. Once all data fragments are received from all subsystems for each event, the Processor Control Pulsar will send the data to a commodity CPU via S-Link. Using a separate PCI slot, the CPU running the Level 2 decision algorithm can return the trigger decision information back to the Pulsar board Control FPGA via either a PCI to S-Link card, or another commercial PCI card with a simple custom daughter card. The Pulsar Control FPGA will then handshake with the Trigger Supervisor to finish the L2 process.

2.3 Pulsar firmware

The firmware for the Pulsar is written in VHDL, the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language. The source code is maintained in a CVS repository. The task of the firmware is to take several data streams, merge them into a single common data stream in S-Link format, and send it downstream for further processing.

All Pulsar boards share the bulk of their firmware. Fig. 4 shows the basic merging data flow that is common to both the DataIO and Control/Merger FPGA in all boards.

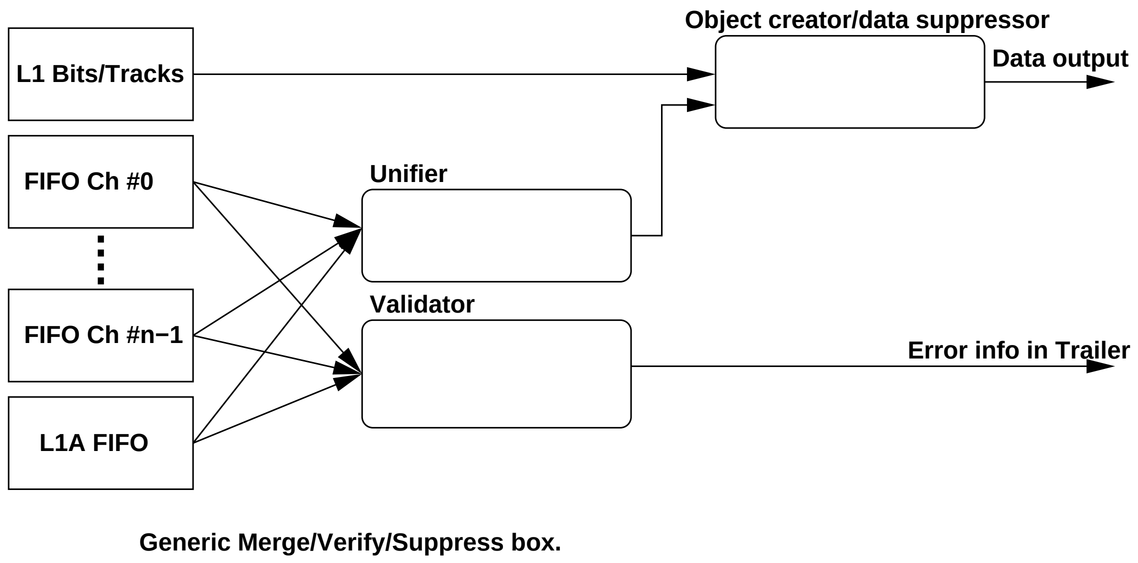

Figure 5: Generic view of merge/verify/suppress. Data from n FIFO channels is sent to both a unifier and a validator. The unifier does any necessary data manipulation such as final clustering, or packing into a 32-bit format. The resultant data from the unifier is sent to the object creator/data suppressor, which can be used to either generate physics objects or suppress un-needed data. In the baseline configuration, this box is a no-op and all data is passed downstream. The output of this process gets sent downstream. The FIFO data stream is also sent to a validator to check data integrity. The results of this integrity check are put in the data packet trailer. The unifier and merger are the only parts of the firmware that require customization for each subsystem. The creator/suppressor is optional and can be omitted completely.

Data is tagged by its L1A FIFO information, which brings with it the L1A, L1R, and buffer numbers for the event. This information is placed in the data header and sent to a merge/verify/suppress stage. Data from the input sources is sent to the same stage. This stage reformats the data as required and fills the data part of the L2 data bank. Finally, a trailer is filled with any error flags. Fig. 5 shows the merge/verify/suppress stage in some detail. The main features in this figure are the three boxes labeled *unifier, validator* and *object creator/ data suppressor*. The unifier merges the incoming data streams into a single output data stream, possibly manipulating the data in the process. The validator receives a copy of the incoming data stream and checks for data integrity. Any problems are tagged and put into the event trailer. The optional suppressor can be used to reduce data volume in cases where simply passing all data would challenge the bandwidth. This box will be a pass-through in most cases, as bandwidth is not expected to be an issue. As track data is available to the suppressor, matches between tracks and muon stubs, ShowerMax hits or calorimeter clusters can also be performed. In addition to the raw data, physics objects as in Table 1 could be created and passed downstream.

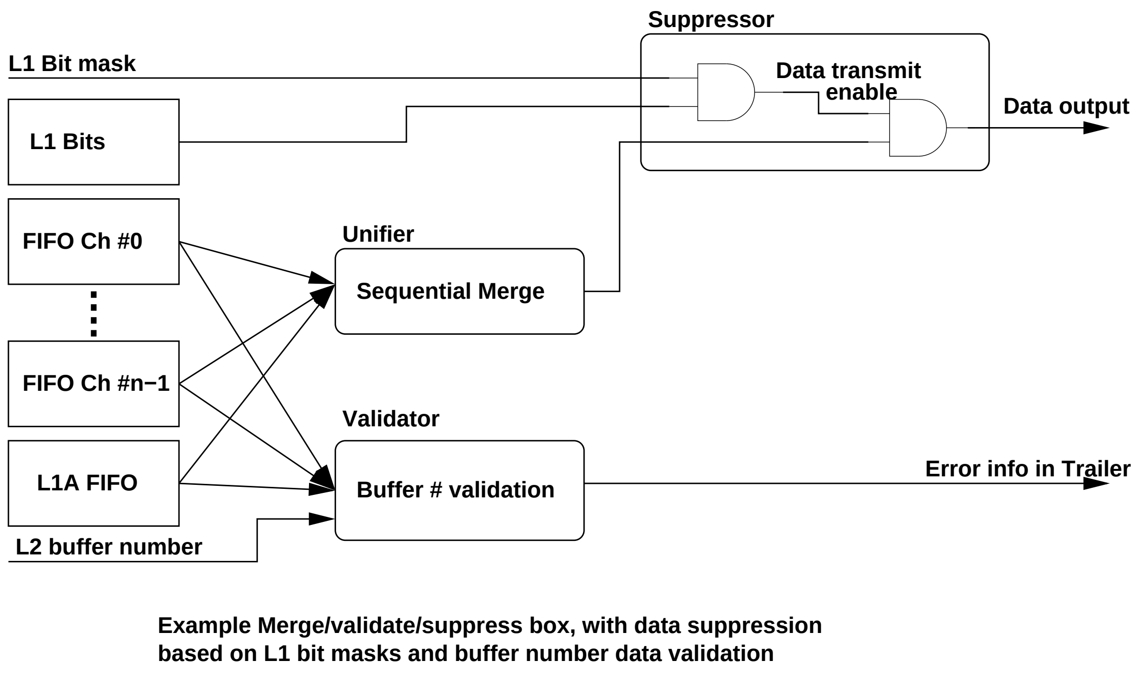

Figure 6: Concrete example of the merge/verify/suppress box in Fig. 4. In this case, data from the FIFO channels is merged sequentially. The data is suppressed from output to reduce bandwidth requirements if the L1 bits indicate that this data is not required for a decision. In addition, the buffer number from the L1A FIFO is compared to the buffer number in the data, and any mismatches are stamped into the trailer.

Fig. 6 shows an explicit example of the merge/verify/suppress box. The merger concatenates data coming from upstream sequentially, one FIFO stream at a time. This merged data stream is suppressed via a L1 bit mask. If none of the L1 bits in this event match the bit mask, we know that these data are not required for the trigger decision. An empty data packet will be sent downstream, and the data are only stored locally in one of the four DAQ buffers for the Level 2 data bank (TL2D) readout. The validator compares the buffer numbers in the incoming data stream to the event buffer number as received from the CDF backplane. Any errors are marked in the trailer.

A large amount of the firmware infrastructure is common to all Pulsar boards. This is possible as the fundamental process of moving and merging data is, in fact, identical. This makes the firmware requirements for the upgrade project manageable and maintainable.

2.4 Pulsar S-Link to PCI interface: bandwidth issues

Many S-Link cards, PCI/PMC interfaces and test tools are already commercially available, and new S-Link cards and S-Link to PCI interfaces are being developed at CERN to meet the bandwidth challenges in the LHC era. Each link itself can move data at 160 MB/s speed (40MHz clock with 32-bit data). A Simple S-Link to PCI interface card (SSPCI) has been available for many years and it has a raw bandwidth of 130 MB/s (achieved 100 MB/s on 33MHz/32-bit PCI bus). The newly available 32-bit S-Link to 64-bit PCI interface card (S32PCI64) is a high speed follow up of the SSPCI. It is designed for highly autonomous data reception and has a raw bandwidth of 260 MB/s [9].

A recent performance test in a laboratory environment indicates that one can achieve up to 150 MB/s data throughput with one S32PCI64 interface, and 300MB/s data throughput with two S32PCI64 interfaces on the fast PCI 66MHz/64-bit bus. The ultimate performance of the S32PCI64 will depend on the type of CPU architecture used. The performance tests described above were performed (at CERN) using PCs with a SuperMicro motherboard based on the ServerWorks LE chip-set. The memory of these PCs is fast enough to sustain the full speed of a 64bit / 66MHz PCI (520 MB/s) [10].

In summary, the bandwidth S-Link devices can provide is already comparable to or better than the S-Link bandwidth itself (160 MB/s). The bandwidth will continue to improve as CERN develops new S-Link to PCI interface cards . Of particular interest is the FILAR project, an S-Link to PCI interface which could handle four 160 MB/s links at once.

2.5 Final Decision-making: CPU choices

The baseline proposal for the final decision-making CPU is driven by two design requirements:

- high bandwidth into CPU memory
- low latency response time

We propose to address these issues via a server-class Intel-based CPU. We do not expect the processor to be CPU-bound, but instead will need high bandwidth, low-latency access to memory. While workstation or desktop systems may not fit these requirements, server-class systems do. In order to assure low-latency, we expect to require a real-time operating system. The freely available RTLinux, an open-source hard real-time Linux extension, is our proposed target OS [11].

2.6 Flexibility with Pulsar design

The Pulsar design is flexible in both the board configuration and in the ways in which the full system can be put together. While the baseline system is straightforward, more powerful scenarios are possible at the cost of somewhat more complex firmware programming. These improvements can be implemented as needed over time; thus the initial firmware requirements are modest.

2.6.1 Board Level Flexibility

The board-level flexibility is illustrated by Pulsar's ability to manipulate the data it receives from upstream. As discussed in Sec. 1.2, the L2 trigger makes decisions based on simple trigger requirements on trigger objects it has assembled from various primitives that originating from other trigger subsystems. The creation of trigger objects very often requires combining different trigger data from different data paths; in Run IIa, this work was done in the Alpha processor. The Pulsar design provides the option to combine different data paths (and thus create trigger objects) at earlier stages, as many data paths are already visible at the pre-processor stage.

For example, it is possible to define muon objects inside the muon pre-processor DataIO FPGA's since both the muon trigger data and track trigger data are available for track and muon stub matching. The fast SRAM's attached to the two DataIO FPGA's can be used as look-up tables for this purpose. The same can be said for electrons in the RECES (merger) pre-processor. By creating physics objects at an early stage and packing them into S-Link data package, the task for the trigger algorithm code running inside a commodity CPU will be less CPU intensive, thus reducing the processing time. This may only become significant if the trigger data sizes and fakes rate increase significantly beyond present projections.

2.6.2 System Level flexibility

Option to use multiple PCI slots

One could use more than one PCI slot for data transfer into a CPU's memory. For example, one can send a certain data fragment, such as ones with long latency, or large data size, on a separate PCI slot. This way, the CPU doesn't have to wait for all the data fragments to arrive before processing the event. This would improve the overall performance, since not all data fragments are needed for L2 trigger decision making for every single event. For

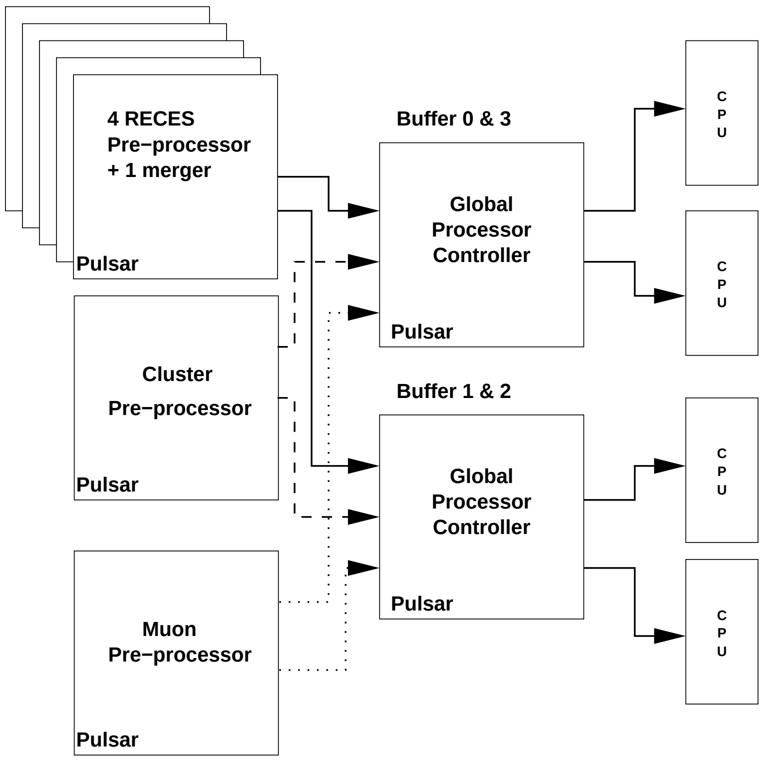

Figure 7: Possible Pulsar configuration with one CPU per L2 buffer.

example, if the SVT data arrives late, it can be sent on a separate PCI slot, allowing the CPU to process the other triggers before the SVT data arrives.

Option to use multiple CPU's for L2 decision processing

Pulsar is designed in such a way that it allows one to use more than one CPU as Level 2 processors. This is possible because Pulsar has two S-Link ports via P3. The simplest expansion to the baseline configuration is to use 2 CPU's for L2 decision. This can be easily done by letting Pulsar send data via two S-Link output ports instead of one. One could either let the two CPU's run different algorithms on the same event, or run the same algorithm on different events. Similarly, it is possible to configure the system to use 4 CPU's.

One interesting example is to use four CPU's running the same trigger algorithm on different events. Each pre-processor can send out its data on one of its S-Link output ports depending on the buffer number of the event, as is shown in Fig. 7. For example, one can have buffer 0 and 3 events sent to S-Link output one, and buffer 1 and 2 events to be sent to S-Link output two. There will be two Pulsar boards acting as Processor Controllers. One receives data for events with buffer 0 or 3, and the other receives data for events with buffer 1 and 2. Each Processor Controller Pulsar then sends each buffer event on one of its S-Link output ports to a CPU. This way, one can have four CPU's running L2 algorithms with

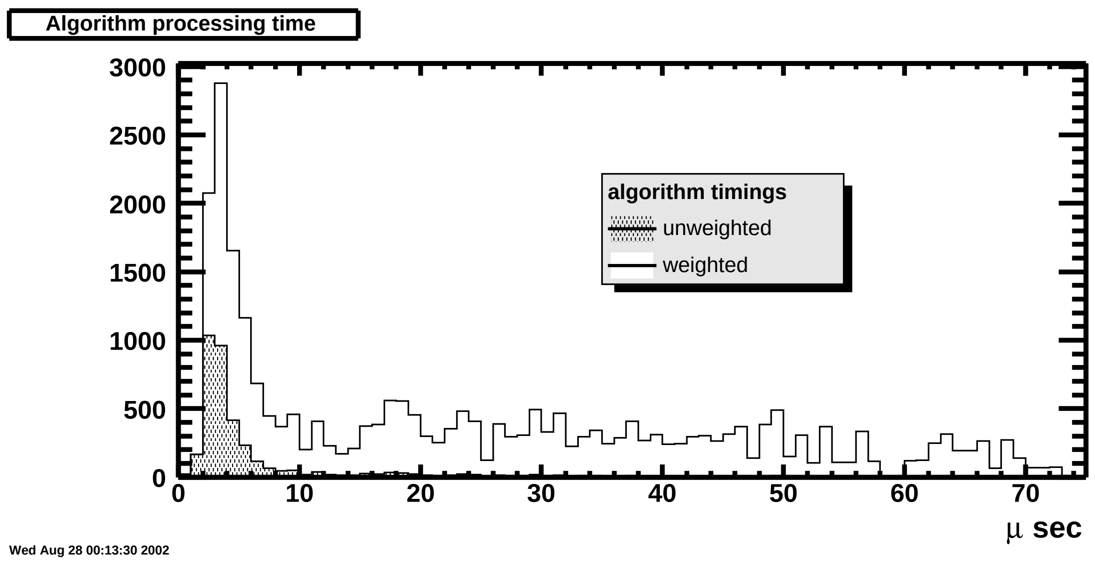

Figure 8: Contribution of tails to the L2 algorithm timings for the Run IIa system. The time shown is the measured algorithm processing time in μs for a beam collisions run. This time is a fraction of the total processing time in one of the two pipeline stages. The hatched time (labeled *unweighted*) shows the number of events which contribute to each time bin. A small tail is visible out to large times ($\sim 50\ \mu$s) and extends throughout the histogram. The unhatched histogram shows the weighted histogram, where each bin entry is weighted by the value of edge of the current bin. This weighting shows the equal-time contributions from the tails, and shows in this particular scenario, while the number of events in the tails is small, their impact is quite large.

each CPU dedicated for a given buffer.[7] In fact, this is a better way to take full advantage of having a four buffer system. The advantage of running the system this way is to avoid the situation where one long tail event can prevent the other three events from being processed. Our experience with the current system shows that this effect can be quite substantial. Fig. 8 shows the distribution of algorithm times as measured on an unbiased sample of L1A's in a Run IIa physics data sample. The algorithm time shown here is the time spent evaluating the trigger algorithms themselves, once data has been unpacked and ignoring all other overhead sources. The hatched histogram shows a distribution that peaks at $\sim 6\mu$s with a substantial tail (hatched histogram, labeled *unweighted* in the plot). The effect of these tails can be seen in the unhatched histogram (labeled *weighted*). Here, each entry is weighted by the time at the lower bin edge, such that equal areas in the plot represent equal time spent processing L2 algorithms. We see that the small number of events in the tail contribute as much to the total processing time as the events in the bulk of the distribution.

As in this case, contributions due to events with long tails could be significant in the Run IIb trigger environment, and the strategy of one CPU per L2 buffer can be used to

[7]Recall that CDF front-end electronics and trigger is a four buffer system at L2.

minimize this effect.[8]

2.7 Pulsar testability

As mentioned earlier, the Pulsar board was originally designed as a universal L2 test stand tool. As such, not only can it sink data but also it can source data for each L2 trigger data path. The source and sink Pulsars will be physically identical at hardware level, but they will have different firmware and mezzanine cards. The data source versions of Pulsar can be used to test the data sink versions of Pulsar (i.e., interface boards). In addition, a few data source Pulsars can be used to provide all the inputs from upstream subsystems and therefore to test the whole L2 decision crate in a stand-alone mode. The data source Pulsar allows the user to load a large number of events with different data size and latency. In this way it can mimic different luminosity running conditions. There is no new hardware involved, the data source and data sink Pulsars are identical at hardware level. In this sense, Pulsar is designed to have Built-In-Self-Test (BIST) capability. This modern practice for electronics with large FPGA's has proven to have positive effect on turn-on times. This test stand tool feature of Pulsar will not only speed up the commissioning of the new system, but also makes the long term maintenance much easier.

[8]An additional question is whether the decisions for L1 buffers have to be sent in L1 FIFO order. The design specification for the CDF DAQ was for each system to have addressable L2 buffers, so that just this type of out-of-order process would be possible. The SVX sub-detector is unique in that it does not have addressable buffers and instead behaves more like a FIFO in readout. The possibility of out-of-order trigger decisions requires further study.

3 Comparison between Run IIa system and proposed Run IIb system

From a functional point of view, both the Run IIa system and the proposed Pulsar system for Run IIb are quite similar. Both are designed to collect different trigger primitive data from different paths and create trigger objects to make L2 trigger decisions quickly. However, the architecture and actual implementation details of the proposed Pulsar system departs in a significant way from that of the existing L2 decision crate. The Pulsar system is designed not only to take full advantage of the technology and design techniques available to us now and that will be available in the near future, but the design is also based upon lessons learned from the current system. In order to highlight the differences between the two approaches, this section compares the two systems in some detail.

3.1 Data transfer

The Run IIa system is based on a mixed data push and pull architecture.[9] This approach was necessary because some of the data paths have large data size (notably the muon system with $\sim$1.3 KBytes per event), and simply pushing all the data downstream requires much higher bandwidth. Instead, as shown in Table 3, the current system was designed to only push the smaller data packets. The processor node pulls region-of-interest data on demand from the large data size systems while processing the event. This approach eases the bandwidth requirement at the cost of complicating the event transfer and processing. This inhomogenous environment has two effects:

1. In the case of multiple processor nodes, each processor node must individually pull the data that is not available via broadcast. If the data is only required by one node, this is irrelevant; however, if more than one node require the same non-broadcast information, it must be retrieved multiple times.

2. Pulling and pushing data ties together the two stages of the pipeline. In the pipeline mode, one event is being pushed into the processor memory while the previous event is simultaneously being processed. In a mixed push-pull architecture, the processing stage and the loading stage are vying for the same resource, the Magic and PCI buses.

The Pulsar system is a completely push-based system. All required data is sent to the processor. Due to the fundamentally higher bandwidth of S-Link, the need for data reduction is less severe. Any necessary reduction is done in the form of data suppression in the interface boards, as described in Section 2.1.

[9]In *Push* mode, the interface board is a Magic Bus master. In *Pull* mode, the board is a Magic Bus slave. See Table 2.

3.2 VME Readout of TL2D bank

In the Run IIa system, the TL2D bank for VME readout is formed by the processor node. The processor formats the bank on L2 accept, and the resources that are used to read out the data (PCI bus to memory) are the same as are used for the loading the next event. This contention for a limited resource, the PCI bus, effectively reduces the available bandwidth into the processor node's memory. Both the present and next event must, by design, compete for PCI bus bandwidth.

In the proposed Run IIb system, the readout is distributed across the universal interface cards. Each interface board has the CDF-standard four DAQ buffers for VME readout upon L2A. These readout buffers are filled on every event, regardless of the L2 decision. The final bank formation occurs in the VME crate controller. The processor node is not involved in the bank formation, and can solely concentrate on trigger decision processing. This approach completely decouples the readout task and L2 decision processing.

We also plan to make all raw data available to VME readout, in addition to the processed information that is sent to the processor. This will be an important debugging tool that is not uniformly present in the current system.

3.3 Implementation: Uniformity, Maintainablilty and Diagnostic Capability

In addition to the architectural differences, the system implementation is quite different in the two generations of systems. The number of custom boards and the emphasis of debugging capabilites is very different, which has wide-ranging effects.

3.3.1 Uniformity and Maintainability

The Run IIa design uses seven custom PCB's as interface boards,[10] a custom computer for the final decision, and a custom backplane to connect the two classes of boards. The large number of custom interface cards perform essentially identical roles. This non-uniformity was among the root causes of the long commissioning period of the Run IIa system. The primary problem encountered in the commissioning of the Run IIa system was the inability to reliably transfer data across the Magic Bus from the interface boards into the processor node. In particular, the arbitration mechanism was unreliable and suffered from low noise tolerance. During this time, lessons learned in the bus arbitration on one interface board did not automatically transfer to the next board, as each had implemented the same arbitration circuits differently. Each interface board's Magic Bus interface had to be debugged individually.

The non-uniformity also implies that a larger spares pool is required for the lifetime of this system: as each board requires two spares, a total of 14 spare L2 interface boards must be maintained.

[10]See Table 2.

In the proposed Run IIb system, there is *one* custom component: the Pulsar board. We take advantage of the fundamental similarity between the different data paths. As described in Sec. 2.3, each path is handled by a Pulsar board with slight variations in the firmware and two simple mezzanine cards. All the infrastructure, such as passing data downstream, is common, and therefore must only be understood once. This will allow the proposed system to be commissioned more rapidly and maintained more simply than the current system. Only one motherboard must be supported, and therefore we can maintain a smaller total pool of spares, again reducing the strain of long-term maintenance.

The interconnect among Pulsar boards is performed via externally supported S-Link hardware from CERN. The final decision is made in a commercial CPU. CPU's can be upgraded with minimal cost to take advantage of ever-increasing CPU power, and the system can take advantage of CERN's continuing research into S-Link for LHC experiments.

3.3.2 Diagnostic Capability

Another lesson learned from the Run IIa commissioning experience was the importance of building a system with testing in the design from the onset. The Run IIa system did not have a testing specification written into the system description, and, as a result, each interface board implented varying degrees of testing capabilities. For instance, many interface boards do not provide VME access to their data. System testing was performed on an ad-hoc basis, and no standard set of test suites was developed. As mentioned in the introduction to Sec. 2, the Pulsar project started as an attempt to provide some of the missing test capabilities to the Run IIa project by providing a realistic, uniform data source for all inputs to the L2 decision crate. The importance of this capability is evident by the fact that CDF chose to devote resources to a test stand so late in the process of developing the Run IIa system.

As a result, the Pulsar system was designed with diagnostic capability as a requirement. Both the Run IIa system and the Run IIb system will profit from the Pulsar in its test stand modus. With its data sourcing abilities, board- and system-level commissioning can occur away from the beam and before the nominal system replacement time. The universal interface card modus is also thoroughly testable. We provide access to the data flow at every stage as every FPGA is available via VME. The use of the S-Link data format provides a robust error checking capability. Each data fragment identifies itself downstream, and is stamped with error checking information from checks performed in the firmware. These tools allow errors to be quickly isolated to a particular part of the system, at which point the offending problem can be diagnosed. The testing strategy for the proposed Run IIb system is therefore dual: in-situ debugging and monitoring capabilities combined with an extensive ex-situ test stand in the form of physically identical Pulsar boards in their test stand configuration.

3.4 System Performance

The performance differences between the two systems are defined by three critera:

1. Raw bandwidth

2. Overhead

3. Architectural differences

Raw bandwidth is simply the ability to get data into the processor to allow it to make a L2 decision. This encompasses both the Magic Bus or S-Link bandwidth and the bandwith into the processor node's main memory for the appropriate system. The actual bandwidth differs from the raw bandwidth by the amount of overhead for transmitting data. Finally, architectural differences can amend these comparisons.

As described in Sec. 1.3, the current L2 system is a two-stage pipeline. This implies that two events can be in L2 at the same time. The bandwidth is to first order limited by the slower of the two stages. The baseline Run IIb system is architecturally the same. To achieve the canonical 40 KHz L1A rate, each stage needs to have an average latency of $\sim 10\ \mu$s. For this type of system, the smaller the latency of the slower pipeline stage, the higher the allowed L1 bandwidth. In what follows, we will compare these two stages in both approaches from system performance point of view.

3.4.1 Loading Stage Latency

The latency for the loading stage is the time it takes to transfer trigger primitives into memory of L2 processor. This latency depends on factors both outside (such as SVX-SVT processing time) and within the L2 decision crate. Here we assume that the SVX-SVT processing time will be improved for Run IIb, so we focus only on the decision crate performance. In what follows, we briefly describe the latency for the existing system and then compare that latency with the proposed design. For the sake of comparison, we will assume that the data size is the same for the Run IIa and Run IIb cases.

Run IIa Loading Stage Latency

For the Run IIa system, the list of items that affects the throughput into memory and hence the latency of the first stage are the raw Magic Bus bandwidth, the bandwidth into the processor node's PCI bus memory, and the overhead associated with the transfer.[11] The Magic Bus raw bandwidth is $\sim 80\mathrm{MB/s}\ (128b/200\mathrm{ns})$. The principal Magic Bus overhead is the need to distribute the bus mastership. The total per-event overhead for this is estimated to be $\sim 1\ \mu$s. The Alpha has a 64 bit PCI bus running at 33MHz which leads to an theoretical PCI DMA bandwidth into memory of 260 MB/s. The effective bandwidth is only about 20-80 MB/s. The actual transfer rate is limited by memory access time and other PCI activity (such as the read-out of TL2D bank and interference between the loading and processing pipeline stages due to the mixed push-pull architecture, as outlined in Sec. 3.1). As the bandwidth of a pipeline is determined by its slowest stage, the bandwidth of the Run IIa loading stage is the effective PCI DMA bandwidth, 20-80 MB/s.

As discussed in Sec. 3.1, the fraction of events which require data to be pulled (for RECES or muon systems) also increases the latency. RECES and Muon data is read over

[11]Most numbers here are taken from Stephen Miller's talk at the August 2002 L2 review. See [5].

Magic Bus by the processor node while processing the current event and at the same time the loading of next event over Magic Bus is on-going. This class of events contribute to the latency in a complicated way.

Proposed Run IIb Loading Stage Latency

In the new system, the latency for the first stage depends on the S-Link raw bandwidth, the time it takes to format and merge input data into S-Link format on the Pulsar boards, the bandwidth of the S-Link to PCI interface and the overhead associated with that. Since the new system has a data-push-only architecture, one cannot simply compare the raw bandwidth of S-Link with that of Magic Bus. Generally speaking, bandwidth demand will be higher in a data pushing only system. However, the Pulsar can suppress data at early stages. The S-Link raw bandwidth is 160 MB/s with 32-bit data words transferred by a 40 MHz clock. The time it takes to format and merge data into S-Link (i.e., the time it takes for data to go through each Pulsar board) can be short, as the task is done with fast modern FPGA's. This time is expected to be below 1 μs per Pulsar board based on initial VHDL simulation results. The latency for data to go through both the pre-processor Pulsar and the Processor Controller Pulsar is expected to be ~ 2 μs.

The raw bandwidth for the new S-Link to PCI interface (S32PCI64) is about 260MB/s. As mentioned in Section 2.4, recent performance tests at CERN indicate that one can achieve up to 150 MB/s data throughput with one S32PCI64 interface and 300 MB/s data throughput with two S32PCI64 interfaces. Unlike the Alpha case, memory in server-type processors is fast enough to sustain the full speed of a 66 MHz/64-bit PCI bus (520 MB/s). The test at CERN also shows that the overhead is quite small – about $1-2$ μs. Therefore, the S-Link to PCI bandwidth is well matched with the S-Link bandwidth.

With greater than a factor of two increase in effective bandwith compared to the Run IIa system, the only subsystem data one needs to suppress is the muon data path. As shown in Table 3, the raw data size of the muon path is 1.3KB per event. Without suppression, this path requires 52 MB/s bandwidth on average at 40 KHz L1A rate, which does not leave a lot of headroom. In Sec. 2.2.1, we showed how the muon data can be suppressed inside the pre-processor Pulsar based on the L1 trigger decision information or based on the data itself. In Run IIb, roughly 10% of the L1 accepted events require muon information for the L2 decision. We expect less than 10% of the data to remain after zero suppression. All other subsystems' data sizes are much smaller (more than one order of magnitude) and can be transferred downstream without modification.

In summary, bandwidth does not seem to be an issue for the new system with this data-push-only architecture.

3.4.2 Latency for the Processing Stage

As described in Sec. 1.3, in the processing stage, data is unpacked from memory, physics objects are created and trigger algorithms are evaluated to make the L2 decision. Besides the dependence on event data size and complexity, the latency for this stage also depends

CPU performance, memory access speed, L2 to TS handshake time, and any board configuration overhead.

Latency for the Processing stage for the Run IIa system

For comparison, we break out some of the times for the Run IIa system.[12] The algorithm processing time includes:

1. The time to unpack L1 trigger bits and scalers. With the current system this is about 2 μs.
2. Algorithm running and data unpacking, which is about 13 μs on average for the current system.
3. Error checking inside CPU, which takes about 1.1 μs.

The L2 to TS handshake time includes:

1. Starting TSI handshake which currently has 2 μs overhead (PCI writes).
2. Building the TL2D bank, which takes 1 μs for L2R and 71 μs for L2A.
3. Ending TSI handshake takes about 2 μs

The board configuration time mostly includes the Alpha DMA configuration setup time ($\sim 5\ \mu$s) and checking for an event in the L1 FIFO ($\sim 1\ \mu$s). Note that the board configuration time is not parallel with either processing time or Magic Bus transfer time.

Latency for the Processing stage for the proposed Run IIb system

Architecturally, the proposed Run IIb system is very similar to the Run IIa system for event processing. Therefore, the latency dependence for the second stage will be similar. There are a few differences. We use a faster, more modern CPU used instead of DEC Alphas, with faster access to memory and a larger on-chip cache for both data and instructions.[13] A high speed S-Link to PCI interface will be used instead of custom Magic Bus to PCI interface using the PCI bus in the processor node. The overhead is smaller for data transfer (including DMA setup time) since the S-Link to PCI interface (S32PCI64) is designed for highly autonomous data reception. L2 to TS handshake is expected to be around 1 μs. Unlike the Run IIa system, the decision-making processors are not involved in the formation of the TL2D bank (see Sec. 3.2).

As discussed above, with the two stage pipeline in the current system, the L2 decisions have to be made in L1A order. This means that all events are processed in the order that they are received. In Sec. 2.6.2, we discussed how the presence of long tails in the processing time introduces large dead time. The baseline proposal for the upgrade has similar architectural limitations, but due to the flexibility inherent in the design, the *Pulsar system can be reconfigured* to better handle this situation.

[12] Again, see [5] for the source of these numbers.

[13] In the current system, the on-chip cache is limited to the 48 KB L2 cache. The L3 cache has been disabled.

3.4.3 Summary Of the Performance Comparison

In summary, with the baseline design of the proposed system, the performance is expected to be much improved over the existing system. The combination of high S-Link bandwidth and data-reduction ability in the pre-processor gives the proposed system more than sufficient bandwidth to handle the increased throughput demand for Run IIb. By using faster, modern CPU's with high memory bandwidth and larger caches, we decrease the latency of the processing stage as well. As described in Section 2.6, with the extra flexibilities both at board level and at the system level, it is expected that the proposed new system can meet the trigger challenges in Run IIb with more than enough safety margin in terms of performance.

4 Summary

With 396 ns bunch spacing and ten interactions per crossing at $\mathcal{L} = 4 \times 10^{32}\ \mathrm{cm}^{-2}\ \mathrm{s}^{-1}$, the Run IIb trigger environment will be very challenging. There is a large uncertainty in the extrapolation of the trigger data size from the Run IIa environment. The current L2 decision crate was designed and built in the mid to late 1990's based on technology available at that time. The design is based on a custom bus (Magic Bus), several custom processors and many different custom interface boards. Although significant progress has been made in two years of a labor-intensive commissioning effort, the system has yet to reach its design goal. Due to the increase in occupancy and the uncertainty of conditions in Run IIb, we think it prudent to increase the bandwidth and processing power while simplifying the design and reducing the maintenance overhead.

The L2 decision crate is a critical subsystem for the entire experiment and our physics program depends crucially on its performance in Run IIb. We propose here to upgrade the L2 decision crate. The upgrade is based on a **single custom universal interface board** (Pulsar) design, plus CERN S-Link interface cards and commodity processors. The new design departs from the previous implementation for the existing L2 decision crate. A standard HEP supported link is used instead of a custom backplane to transfer trigger data into CPU memory. Level 1 trigger and track trigger information are made available to all FPGA's for each Pulsar board. This design feature is driven by physics requirements, providing flexibility in performance. In addition, by choosing S-Link as the standard link, the new system can be easily upgraded to better S-Link to PCI cards and more powerful processors as they become available. This new system is designed to have sufficient safety margin and flexibility in performance to meet the Run IIb trigger challenges, to have built-in self-test capabilities to speed up the commissioning process and to ease the long term maintenance effort all the way through Run IIb.

5 Acknowledgments

We would like to thank Nathan Eddy, Maurice Garcia-Sciveres, Eric James, Steve Kuhlman, Andrew Martin, Stephen Miller, Jane Nachtman, Kevin Pitts, David Saltzberg, Masa Tanaka and Monica Tecchio for useful discussions. We would like to thank Frank Chlebana, Bruce Knuteson, Takasumi Maruyama, Michael Schmidt, and Rick Tesarek for comments and criticism of drafts of this note. Thanks to Erik Van der Bij, Markus Joos, Christoph Schwick and Sidik Isani for discussions about S-Link.

References

[1] CDF IIb Collaboration, *The CDF IIb Detector Technical Design Report*, 9/2002. `http://www-cdf.fnal.gov/upgrades/run2b/Documents/Documents.html`. 1

[2] R. Blair *et al* (CDF II Collaboration), *The CDF Run II Detector Technical Design Report*, 1996. FERMILAB-Pub-96/390-E. 2, 3

[3] H. Kasha, M. Schmidt. *CDFII Trigger/DAQ Simulations Including the SVXII Readout*, 1997. CDF Note 4213. 3

[4] C. Murphy, M. Campbell, *The Level 2 Magic Bus*, 1997. See `http://umwnt1.physics.lsa.umich.edu/cdf/documents.htm\hyper@hashUpgrade`. 4

[5] Talk given by Stephen Miller at the Aug 2002 L2 trigger review. See `http://www-cdf.fnal.gov/internal/upgrades/daq_trig/trigger/reviews/l2_review_020801.html`. 4, 5, 6, 25, 27

[6] P. Azzi *et al*, *CDF Run IIb Report*, 2002. `http://www-cdf.fnal.gov/upgrades/run2b/run2b_task_force/` 6, 7

[7] `http://hep.uchicago.edu/~thliu/projects/Pulsar/` 8

[8] H. C. van der Bij *et al*, *S-Link, a Data Link Interface Specification for the LHC Era*, 1997. Published in the *Proceedings of the Beaune 97 Xth IEEE Real Time Conference*. See also `http://hsi.web.cern.ch/HSI/s-link/`. 9

[9] W. Iwanski *et al*, *Designing an S-LINK to PCI Interface using an IP core*. Article presented at the 12th IEEE-NPSS Real Time Conference, June 4-8, Valencia (Spain). See also `http://hsi.web.cern.ch/HSI/s-link/devices/s32pci64/`. 17

[10] Markus Joos, CERN, private communication. See also `http://documents.cern.ch/archive/electronic/other/agenda/a0281/a0281s1t21/transparencies/Sw_robin.pdf`. 17

[11] M. Barabanov, *A Linux-based Real-Time Operating System*. M.S. thesis, New Mexico Insitute of Mining and Technology, Socorro, New Mexico, 1997. See also `http://fsmlabs.com/community/`. 18