UNIFIED SYNCHRONIZED DATA ACQUISITION NETWORKS

Inauguraldissertation

zur Erlangung des akademischen Grades

eines Doktors der Naturwissenschaften

der Universität Mannheim

vorgelegt von

Frank Karl Wolfgang Gerhard Rolf Lemke
(Diplom-Informatiker im Integrierten Diplomstudiengang
Mathematik und Informatik)

aus Mutterstadt

Mannheim, 2012

Dekan:	Professor Dr. H. J. Müller, Universität Mannheim
Referent:	Professor Dr. U. Brüning, Universität Heidelberg
Korreferent:	Professor Dr. R. Männer, Universität Heidelberg

Tag der mündlichen Prüfung: 30.11.2012

Abstract

The permanently evolving technical area of communication technology and the presence of more and more precise sensors and detectors, enable options and solutions to challenges in science and industry. In high-energy physics, for example, it becomes possible with accurate measurements to observe particles almost at the speed of light in small-sized dimensions. Thereby, the enormous amounts of gathered data require modern high performance communication networks. Potential and efficient implementation of future readout chains will depend on new concepts and mechanisms.

The main goals of this dissertation are to create new efficient synchronization mechanisms and to evolve readout systems for optimization of future sensor and detector systems. This happens in the context of the Compressed Baryonic Matter experiment, which is a part of the Facility for Antiproton and Ion Research, an international accelerator facility. It extends an accelerator complex in Darmstadt at the GSI Helmholtzzentrum für Schwerionenforschung GmbH.

Initially, the challenges are specified and an analysis of the state of the art is presented. The resulting constraints and requirements influenced the design and development described within this dissertation. Subsequently, the different design and implementation tasks are discussed. Starting with the basic detector read system requirements and the definition of an efficient communication protocol. This protocol delivers all features needed for building of compact and efficient readout systems. Therefore, it is advantageous to use a single unified connection for processing all communication traffic. This means not only data, control, and synchronization messages, but also clock distribution is handled. Furthermore, all links in this system have a deterministic latency. The deterministic behavior enables establishing a synchronous network. Emerging problems were solved and the concept was successfully implemented and tested during several test beam times.

In addition, the implementation and integration of this communication methodology into different network devices is described. Therefore, a generic modular approach was created. This enhances ASIC development by supporting them with proven hardware IPs, reducing design time, and risk of failure. Furthermore, this approach delivers flexibility concerning data rate and structure for the network system. Additionally, the design and prototyping for a data aggregation and concentrator ASIC is described. In conjunction with a dense electrical to optical conversion, this ASIC enables communication with flexible readout structures for the experiment and delivers the planned capacities and bandwidth.

In the last part of the work, analysis and transfer of the created innovative synchronization mechanism into the area of high performance computing is discussed. Finally, a conclusion of all reached results and an outlook of possible future activities and research tasks within the Compressed Baryonic Matter experiment are presented.

Zusammenfassung

Mit der sich ständig beschleunigenden technischen Entwicklung im Bereich der Rechnerkommunikation sowie immer genauer werdenden Sensoren und Detektoren ergeben sich neue Möglichkeiten und Lösungsansätze für viele Herausforderungen in der Wissenschaft und Industrie. Im Bereich der Hochenergiephysik beispielsweise wird, es möglich, immer genauere Messungen von Teilchen mit annähernd Lichtgeschwindigkeit auf engstem Raum durchzuführen. Die hierbei anfallenden enormen Datenmengen benötigen moderne Hochleistungsnetzwerke. Die optimale Umsetzung zukünftiger Ausleseketten erfordert neue Konzepte und Mechanismen.

Diese Arbeit widmet sich dem Ziel, neue effiziente Synchronisationsmechanismen sowie Auslesesysteme zu entwickeln, um zukünftige Sensor- und Detektorsysteme zu optimieren. Dies geschieht im Kontext des Compressed-Baryonic-Matter-Experiments, welches ein Teil der Facility for Antiproton and Ion Research ist, einer internationalen Beschleunigeranlage zur Forschung mit Antiprotonen und Ionen. Es handelt sich hierbei um eine Erweiterung des in Darmstadt bestehenden Beschleunigers der GSI Helmholtzzentrum für Schwerionenforschung GmbH.

Zunächst befasst sich die Arbeit mit der Beschreibung und Analyse der Problemstellung sowie dem Stand der Technik. Hieraus ergeben sich Randbedingungen und Erkenntnisse, die alle folgenden Entwicklungen beeinflusst haben. Anschließend werden die einzelnen Entwurfs- und Entwicklungsschritte beschrieben. Begonnen wird mit der Analyse der Grundlagen für ein optimiertes Detektorauslesenetzwerk. Dem schließt sich die Beschreibung des Designs eines effizienten Kommunikationsprotokolls an, welches alle erforderlichen Fähigkeiten aufweisen muss. Um kompakte und effiziente Systeme bauen zu können, sollten alle benötigten Funktionen nur über einen Kommunikationskanal zur Verfügung stehen. Dieser Kanal dient nicht nur zum Austausch von Daten, sondern ermöglicht auch eine Taktverteilung. Ferner werden Links mit deterministischer Latenz verwendet, da es mit

einem deterministischen Verhalten möglich ist, eine Synchronisation über diesen Kanal aufzubauen. Auf Basis dieser Grundlagen wurde ein Protokoll definiert, das drei virtuelle Kommunikationsklassen zur Verfügung stellt: die Daten-, Kontroll- und Synchronisationskommunikation. Dieses Konzept wurde erfolgreich umgesetzt, aufkommende Probleme wurden gelöst, und die Tauglichkeit durch zahlreich Teststrahlzeiteinsätz bewiesen.

Darüber hinaus wird auf die Implementation und Integration in unterschiedliche Netzwerkkomponenten eingegangen. Zunächst wird das generische modulare Konzept erläutert. Dieses ermöglicht es, vor allem in ASIC-Entwicklungen bereits getestete, funktionierende Hardwarekomponenten zu verwenden, um Entwicklungszeit und Risiken zu minimieren. Außerdem erlaubt das Konzept einen flexiblen Einbau von Detektorauslese-ASICs mit verschieden Datenraten in einem Auslesenetzwerk. Ferner werden das Design und die Prototypentwicklung für einen Datenaggregations- und Konzentrator-ASIC diskutiert. Dieser ASIC ermöglicht in Kombination mit einer dichten räumlichen, elektrisch-optischen Kommunikationsumsetzung den Aufbau eines Auslesenetzwerks für das Compressed-Baryonic-Matter-Experiment, mit dem sich die volle geplante Kapazität erreichen lässt.

Am Ende wird der erschaffene innovative Synchronisationsmechanismus auf das Gebiet des Höchstleistungsrechnens übertragen, und seine Vorteile werden erläutert. Die Arbeit schließt mit einer Zusammenfassung der erreichten Entwicklungen sowie einem Ausblick auf mögliche zukünftige Forschungsaufgaben und Projekte im Kontext des Compressed-Baryonic-Matter-Experiments ab.

Acknowledgments

First, I would like to thank Professor Dr. Brüning for giving me the opportunity for doing this research work at the Computer Architecture Group. Without his support and advice, this work would not have been possible.

In addition, my thanks go to all of my colleagues at the Computer Architecture Group for creating a pleasant and productive working atmosphere and all groups we worked together from the CBM Collaboration for the good cooperation. Especially thanks go to Walter F.J. Mueller for his outstanding support and for the numerous fruitful discussions that we had together.

Finally, special thanks go to my family and friends for always supporting me.

Content

Chapter 1

Introduction

This chapter introduces the subject of sensor and detector data acquisition systems. Furthermore, the goals and tasks are described, which are achieved within this research work. The significance ot the data aquisition system within the Compressed Baryonic Matter experiment is clarified. Additionally, facts and challenges concerning required synchronization within modern sensor and detector data acquisition networks are presented. An overview of the Compressed Baryonic Matter Experiment is presented. Due to the huge amount of captured data and the accuracy needed, this high-energy physics experiment requires special solutions for its data acquisition network. This chapter ends with a structured overview.

1.1 Motivation

Sensor and detector readout networks have become more and more optimized networks. Due to scientific challenges, like detection of particles traveling near light speed in small-sized dimensions within high-energy physics, as well as demands in industrial sectors, the accuracy of measured data and performance concerning its data acquisition (DAQ) [1] system has to improve. Improvements can be found regarding sensor control, capability of sensors, DAQ network topologies, synchronization mechanisms, and communication bandwidth. However, continuous technology enhancements and new conceptual approaches are the driving factors enabling solutions for upcoming challenges.

One of the most challenging tasks in this context is synchronization. Synchronization is required to assure that all devices have the same relative time base. Later, it is possible to correlate measured data for further analysis. Thus, the finer the synchronization precision is, then the more accurate is the measured data. For standard applications, it is sufficient to use standard network and protocol systems combined with software synchronization or synchronization on protocol application layer. Thereby, synchronization precision in the order of µs is reachable. Some systems have been improved with high efforts to reach even better resolutions using separate special clock distribution and synchronization networks. However, in future environments, a precise and reliable synchronization in the order of ps is required. Achieving this magnitude, especially with respect to density and bandwidth intense future DAQ networks, will require special hardware support for synchronization. Therefore, new communication concepts using protocols with completely integrated functionalities, compact special DAQ hardware, and efficient synchronization mechanisms, need to be invented.

This thesis targets the creation of a new synchronization mechanism to fulfill upcoming requirements for sensor and detector networks. This synchronization mechanism needs to be capable of supporting different topologies and needs direct integration into communication streams avoiding additional hardware. Besides this mechanism, within the context of

the Compressed Baryonic Matter (CBM) Experiment, the goal is to create a complete CBM DAQ network solution. These solutions need to be optimized for the special requirement of CBM.

CBM requirements:

- Precise time synchronization mechanism < 200ps
- Compact hardware, due to limited space
- Dense interconnection solutions
- Radiation tolerance
- Self triggered front-end electronics
- Flexibility to support different DAQ structures
- Reusability of modules and flexible build-up variants
- Efficient data aggregation schemes and rate conversion
- High bandwidth handling up to several TB/s of data

These requirements dictate design constraints for protocol and hardware. The protocol needs to deliver a high link utilization, to provide efficient interfaces, and to handle traffic types as slow control messages, data messages, and synchronization. In addition, a clock distribution needs to be spanned. This provided clock has to be precise enough for avoiding bit slips. Deterministic latency connections must be established. A deterministic latency connection guarantees delivery of synchronization messages with a constant runtime. Therefore, it always takes an identical amount of hardware clock cycles to process them within devices. This needs to be valid even after power cycles. Due to a guarantee of no occurring bit slips and deterministic latency connections, device distances in relation to a master synchronization node can be measured. Then a synchronization mechanism can be used to adjust all devices to run in a synchronous mode. Additionally, the network devices should support the same protocol reusing almost identical DAQ network hardware. Thus, protocol conversion can be avoided. Compatible implementations for various FPGAs and ASICs need to be available.

Furthermore, this thesis focuses on concepts, plans, and the first prototyping of an ASIC for early data aggregation, rate conversion, synchronization, and control of front-end electronics (FEE) near the detector. This ASIC handles all traffic classes and controls numerous via electrical links attached FEE boards. It has to combine the FEE data streams and send them with increased speed to the next network hierarchy level. Due to potential separation and communication distance, optical fiber connections are required to connect this ASIC to higher hierarchies. These optical fiber connections need to be constructed with high density, because of space constraints and build up dependencies. Therefore, an innovative electrical to optical converter, in close proximity to the ASIC, is required to combine links into dense fiber connections for delivering sufficient bandwidth.

1.2 Design Challenges

There are different network classifications such as asynchronous, plesiochronous, and synchronous/mesochronous networks. Asynchronous networks do not have any timing relation. If they are used for data transfer, a separate network for providing synchronization is required. The plesiochronous networks are almost synchronous networks. They are timed with clocks having almost the same timing. Due to sophisticated adjustment mechanisms, they reach accuracies in the order of µs. Nevertheless, synchronous/mesochronous networks can achieve the highest precision, because their clocks are completely derived from one master clock. Thus, synchronization quality strongly depends on the clock distribution system. They are ideal for achieving the target accuracy requirements, but in order to meet these requirements the network has to take care of various challenges. In standard literature [2] jitter and wander, the most important challenges within synchronous systems, are described. Both jitter and wander are phase variations. Jitter is defined as short-term variation, including all variations above 10 Hz. It depends on clock quality, common noise, and noise introduced by circuits. Wander, on the other hand, is a long-term phase variation below 10 Hz. Wander can be caused by slight frequency variations between oscillators, transmission fluctuation, or bit stuffing mechanisms. In huge networks, wander can hardly

be avoided. Thus, from time to time, readjustment must be done. However, jitter and wander may lead to bit slips and should be avoided. A bit slip is caused if a clock cycle is skipped. This causes errors and affects the system accuracy.

1.3 The Compressed Baryonic Matter Experiment

The Compressed Baryonic Matter (CBM) experiment is a part of the Facility for Antiproton and Ion Research (FAIR) in Darmstadt at the GSI [3]. FAIR [4] works together with the GSI [5] for constructing and running the planed FAIR facility. FAIR extends the existing GSI accelerator and synchrotron. Construction has already begun and the first stage of expansion is currently planned to function in 2018. Figure 1-1 gives an overview of the planned FAIR facility. It shows the existing GSI accelerator in blue color on the left side and the FAIR synchrotron extending it in red color on the right side. FAIR focuses on five beam properties: highest beam intensities, brilliant beam quality, higher beam energies, highest beam power, and parallel operation. It contains eight different experiments with collaborations.

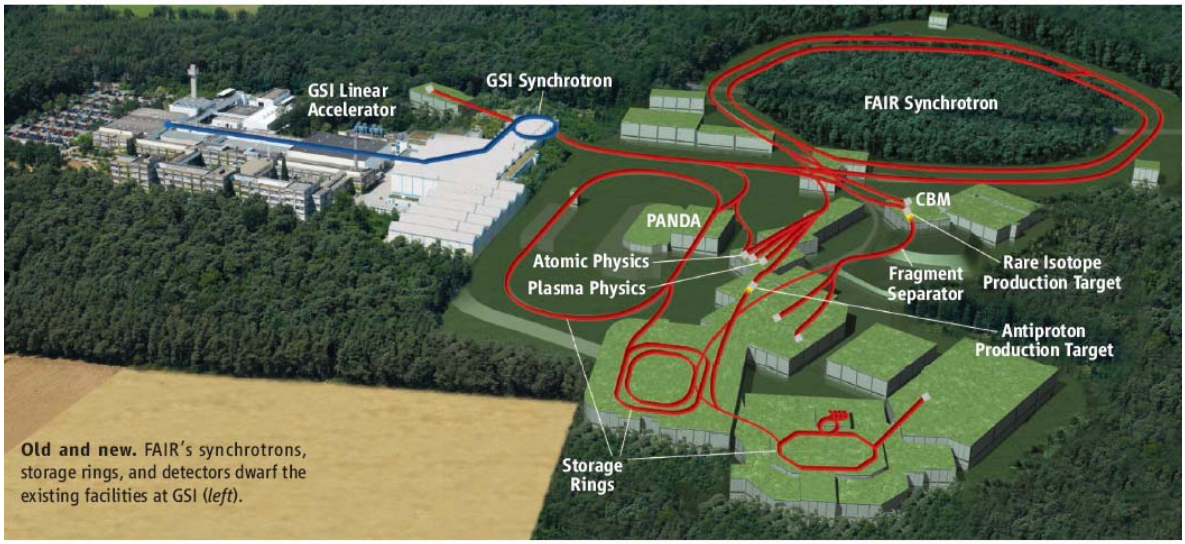


Figure 1-1: FAIR Area at the GSI in Darmstadt [6]

The planned extension consists of two high-energy superconducting synchrotrons, SIS100 and SIS300, built on top of each other in a subterranean tunnel, also consisting of five collector, cooler, and storage rings and numerous new detectors serving five fields of physics [6]. The first expansion stage, the SIS100 will deliver beams with energies up to 11A GeV for Au, 14A GeV for Ca, and 29 GeV for Protons. The next expansion stage, planned to be ready two years after initial operation, will deliver up to 35A GeV for Au and 89 GeV for Protons [7]. The core physics planned for CBM will require the capabilities of SIS300.

The CBM [8] experiment investigates the highly compressed nuclear matter using nucleus-nucleus collisions. This experiment will examine heavy-ion collisions in fixed target geometry and will be able to measure hadrons, electrons and muons. Key observables are detection of open charm, phase-space distributions, flow of protons, pions, kaons, hyperons, hadronic resonances, light vector mesons, charmonium, and heavy vector mesons. The unique discovery potential of the CBM experiment is the investigation of a very promising territory of the QCD phase diagram [7]. Detecting this set of particles requires fast, and radiation-hard detector systems, which are positioned in dense arrangements and readout by front-end electronics (FEE). In figure 1-2, the CBM detector system is depicted. The main detectors are:

- Silicon Tracking System (STS)
- Ring Imaging Cherenkov Detector (RICH)
- Transition Radiation Detector (TRD)
- Muon Chamber/Absorber System (MUCH)
- Resistive Plate Chambers (RPC)
- Electromagnetic Calorimeter (ECAL)
- Projectile Spectator Detector (PSD)

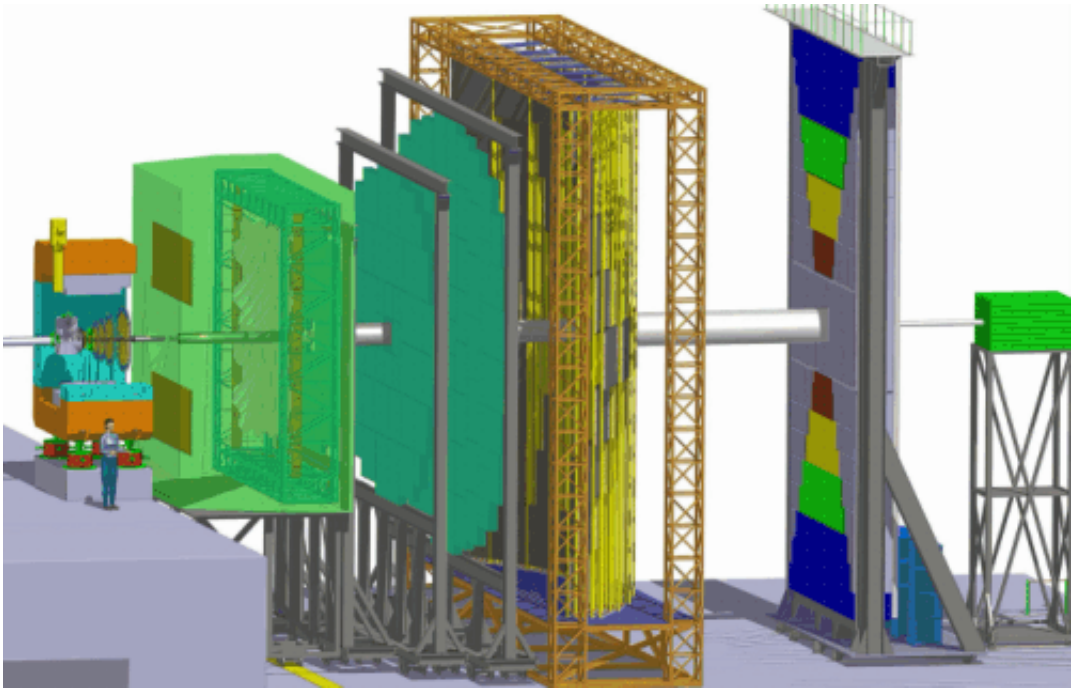


Figure 1-2: The CBM Detector System [5]

The FEE readout uses a self-triggered approach creating a continuous data stream. Thus, it requires synchronization for time stamping and a high-speed data acquisition network. The event selection observes up to 1000 charged particles for each collision at reaction rates of up to 10 MHz. This event selection is based on full track reconstruction and is done after event building in a processor farm.

The data acquisition (DAQ) network system is one of the challenging tasks of the ambitious CBM experiment. There are several special demands on the DAQ System. Due to its self-triggered approach, a data flow from the detector of up to several TB/s has to be handled. Additionally, the support of different types of detectors, placement constraints and other requirements for the DAQ network system lead to a set of features. These are required for the CBM network such as flexible build-up variants, efficient data aggregation schemes including link speedup, precise time synchronization, and dense interconnection solutions.

A special mechanism is needed for synchronization of the detectors. Radiation tolerance has to be provided by the frontend part of the network. The network protocol needs efficient support of four communication types as clock distribution, time synchronization, control messages, and data streams. The CBM network has to be capable of using fiber optics solutions to overcome length restrictions and problems concerning different voltage levels at the detectors. Due to limited budgeting, it is important to analyze and use commercial off-the-shelf (COTS) parts, like Field Programmable Gate Arrays (FPGAs), and try to integrate common solutions, for example, concerning control systems [9], [10]. Cost effective COTS parts usually have better availability, which is an additional important argument for long term experiments. Nevertheless, for most of the front-end detector parts special solutions, within FEE ASICs or for optical interconnects, are required. FEE ASICs are not only needed to readout detectors, but also for data aggregation and FEE controls. These ASICs need to be radiation tolerant and should combine several detector read-out ASICs. The communication protocol and the physical interface hardware within the network need to be reusable and has to deliver a modular conception. This reusability in the network creates the advantage of requiring no protocol conversion within the DAQ system.

1.4 Chapter outline

In chapter two, *Design Space*, different network topologies and implementation variants are discussed. Additionally, a state of the art part presents an overview and a detailed analysis of comparable research implementations and widely used synchronized communication systems, mainly within the telecommunications industry. The third chapter, *Development of a Synchronous Network for CBM*, describes the CBM data acquisition network structure and development of the CBMnet protocol. Additionally, it focuses on the synchronization mechanism used within CBMnet. Then in chapter four, *Generic Modules for CBM Device Development*, the CBMnet generic modules concept, providing reusable built-in module blocks for CBM network devices, is presented. Furthermore, it includes the example of a first ASIC prototype implementing CBMnet and additional modules. The following chapter five, *The HUB CBMnet ASIC Development*, a description of the HUB ASIC concepts focus-

ing on traffic handling and synchronization mechanism is described. In addition, prototype measurements, opto-converter development, and build-up scenarios are shown. Ideas for transferring the innovative CBMnet mechanism used for synchronization into modern high performance computing clusters are presented in chapter six, *Networks based on DLM Synchronization*. Finally, chapter seven, *Conclusion and Outlook*, concludes research results and achieved goals. It gives an outlook on future activities, especially in context of the CBM experiment.

Chapter 2

Design Space

This chapter analyzes the CBM DAQ network structure and provides an overview of synchronization and readout systems as state of the art examples. It includes general widely used commercial solutions for computing clusters. Additionally, specialized scientific solutions are shown, which fulfill similar requirements as the CBM FAIR project. It is elaborated, if these solutions can be reused or if some consequences can be drawn for the CBM specific developments. This is essential to be able to fulfill the high requirements for CBM concerning topics like area, synchronization, and bandwidth of the DAQ readout system.

2.1 Basic DAQ Structure and Abilities

Before a DAQ network structure can be designed and developed, the basic abilities required have to be specified. These abilities together with the basic DAQ structure have to be carefully tended during selection and design of a DAQ network structure. Figure 2-1 delivers an overview of the basic DAQ structure. It consists of a detector system acquiring analog measurement and a directly attached, front-end-electronics device. This device is responsible for control, for analog to digital conversion, and for data synchronization. Still contained in a near detector region, these front-end electronic requires an aggregation and control stage. A hierarchical DAQ network connects this stage to an area containing synchronization, control, data sink and processing, which is attached to the computing cluster. The basic abilities for selecting an appropriate DAQ network structure are:

- Physically constraining
- Cost and Performance specifications
- Ease of use
- Reusability
- Scalability
- Reliability

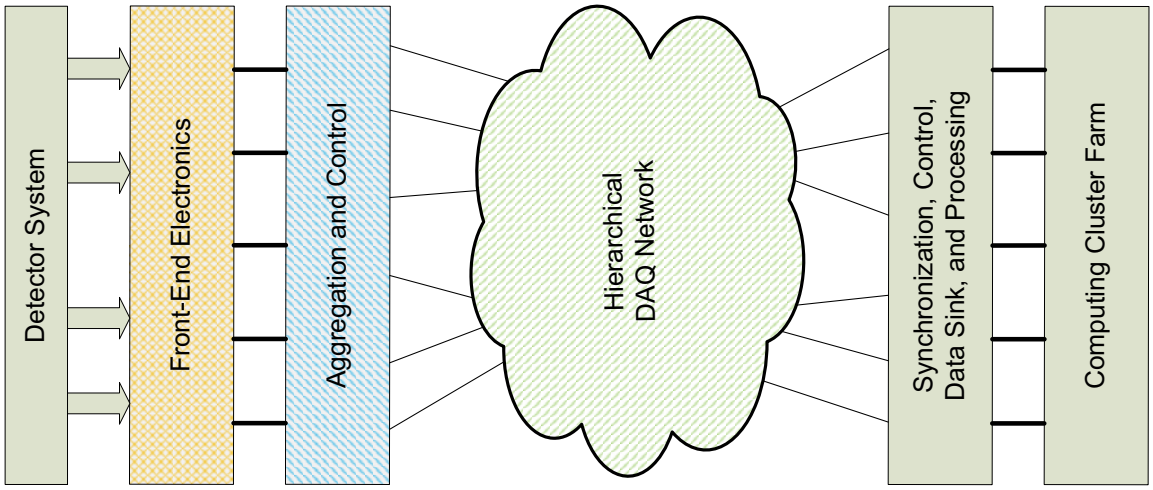


Figure 2-1: DAQ Network Structure

2.2 Network Topology

One of the first decisions to be made before starting creation of a DAQ network system is about its interconnection network. For interconnection networks, it is crucial to choose the right type of network and topology. This decision influences the maximum network performance and affects the design space for the network protocol and all special hardware.

In general, networks can be classified as shared-medium networks, direct networks, indirect networks, and hybrid networks [11]. A DAQ network, which has thousands of end-point devices such as the CBM network requiring a high constant bandwidth from all front-ends obviously can not use shared-medium networks like buses. Analyzing the required communication traffic results in two different communication patterns. Data is only streamed from front-end devices to the backend direction. Control and synchronization traffic is streamed between a front-end device and a control node in an intermediate network region. There is no pattern of communication, which benefits from a hybrid network. The only network types left are direct and indirect networks, but they both do not fit perfectly onto this application specific network. There are no external switching structures in the network and the connections are more or less from the FEE direct to a control node or data sink. They are only aggregated on their way between direct device connections. Though it is similar to a direct network, it really is not a direct network considering the common definition.

In first order, the topology selection depends on traffic patterns of a network. There are two patterns for communication in CBM, the data traffic and the synchronization/control traffic. The typically known structure fulfilling this behavior is a tree. Figure 2-2 depicts this CBM network tree structure. Due to data aggregation, within each stage starting at the front-end bandwidth and link speed increases. This principle is important for meeting the physical constraints of CBM, because it enables usage of dense interconnection solutions. In addition, reducing the number of links and a possible unified connection scheme helps to reduce cost with regards to performance requirements. However, this concept needs almost identical logic for network communication in multiple devices. Therefore, it is necessary to focus

on the ease of use and reusability during development. As long as there is space left, parallel readout trees can be used to extend this network. This fulfills the scalability needs of the system. The basic idea regarding reliability is to use as many standard components as possible in the readout system. Therefore, an analysis of widely used solutions was done. The results are presented in the following paragraphs.

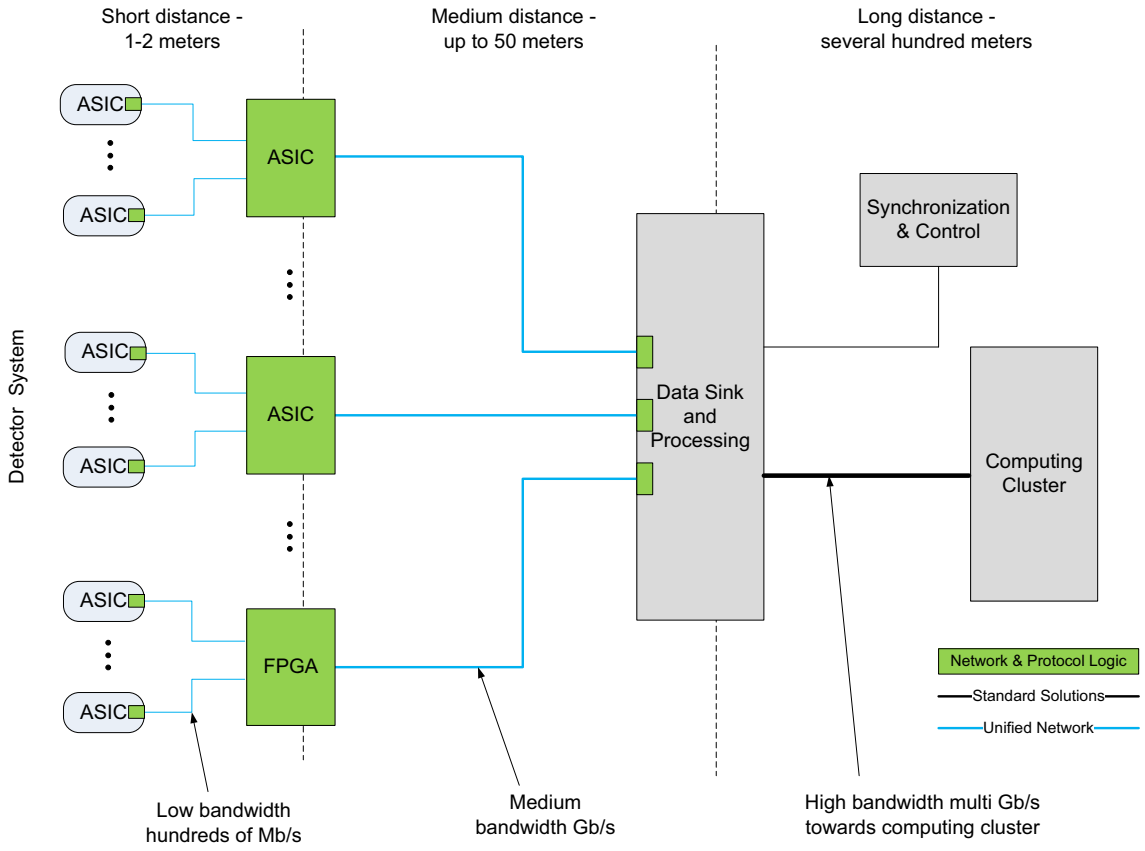


Figure 2-2: CBM Direct Tree Network Topology

2.3 State of the Art

2.3.1 SONET / SDH

This subchapter is focused on presenting widely used telecommunication standards. Analyzing these standards is of interest for the specific networks focused, because they are widely used and thereby cheap, reliable, flexible and modular concerning their network structure, and they are well standardized. Recently, the various proprietary telecommunication networks used were substituted by new standards called [12] the Synchronous Optical Network (SONET) defined by American National Standards Institute (ANSI) and the Synchronous Digital Hierarchy (SDH) defined by the Comité Consultatif International Téléphonique et Télégraphique (CCITT today migrated into International Telecommunication Union (ITU)). The differences between SONET and SDH are insignificant and they are largely compatible. Thus, communication between them works reliably. The most interesting benefit delivered by SONET/SDH is a multiplexing scheme that is efficient and supports high bandwidth optical communication. Additionally, it guarantees a certain level of performance and is designed to be reliable. It supports services like voice telephony over PSTN (Public Switched Telephone Networks), FDDI (Fiber Distributed Data Interface), Fibre Channel, ATM (Asynchronous Transfer Mode), PPP (Point-to-Point Protocol), Ethernet, and Gigabit Ethernet. SONET/SDH products are used all-around in the telecommunications industry. There are solutions supporting SONET/SDH as hard or soft intellectual property (IP) blocks within FPGAs [13], which are widely used for telecommunication purposes because of their flexibility. Additionally, various providers [14] offer multifunctional chips delivering integrated CDR functionalities, framing, and other SONET/SDH features.

SONET/SDH is a synchronous transport layer protocol having a linear or ring topology. A SONET/SDH network is structured in layers. The main layer structure consists of *Paths* containing multiple *Lines* or *Multiplex Section Layers* (for SDH) being aggregations of *Sections* or *Regenerator Sections* (for SDH). A *Path* is defined as an end-to-end connection. The *Line* layer combines *Sections* and is responsible for the synchronization network func-

tions. A *Section* is a connection between two network elements. A *Tandem Connection Layer* (TCM) was added later to this standard definition being between Line and Path layer. A TCM adds some overhead to ease performance monitoring and allows the grouping of signals on a logical level. Within these layers, fixed frames are transported and multiplexed through the network. One frame period is 125µsec. Thus, 8000 frames are transported within a second. The frame unit sizes were adapted to the given environment in North America and Europe. Thereby they differ in their definition, but in general, they are kept compatible. SONET uses Synchronous Transport Signal (STS) units as a frame definition. Here, the smallest frame unit is a STS-1 frame, which is 810 bytes. SDH transports Synchronous Transport Modules (STM). STMs have the triple size of STS frames, ergo 2430 bytes. To achieve higher bandwidth these basic elements are multiplexed into frames containing multiple basic frames keeping the frame period identical. Thus, STM-1 is the counterpart for a STS-3 frame. In table 2-1, the different defined frame formats are shown with their associated data rates. A STS-1 frame [15] has 90 bytes wide columns and 9 byte high

Electrical Level	Optical Level	Data Rate (Mbps)	Payload Rate (Mbps)	SDH equivalent
STS-1	OC-1	51.84	48.38	STM-0
STS-3	OC-3	155.52	149.76	STM-1
STS-12	OC-12	622.08	599.04	STM-4
STS-48	OC-48	2488.32	2396.16	STM-16
STS-192	OC-192	9953.28	9584.64	STM-64
STS-768	OC-768	39813.12	38486.02	STM-256

Table 2-1: Data Rate definitions for SONET/SDH

rows. The first three columns of the frame are used for the *Transport Overhead* (TOH) bytes consisting of a 3 by 3 bytes field for *Section Overhead* (SOH) and a 3 by 6 field for *Line Overhead* (LOH). The remaining 87 columns are the *STS Payload Envelope* (SPE), which contains the *Path Overhead* (POH) in the fourth column, while columns 30 and 59 are used as the fixed stuff columns. Thus, 84 x 9 bytes stay as raw data delivering a 93.33 % (84/90) data utilization for the transport layer. STS-N frames are generated by multiplexing the sub-frames providing an identical utilization. The multiplexing of three STS-1 frames

into a STS-3 frame is depicted in figure 2-3. The TOH fields are merged into the first 9 columns of this extended frame and SPE is now 261 bytes wide, delivering 3 x 84 byte-wide column payload. Howerver, in the SPE some services can be transported directly or included into the Virtual Tributary (VT). In case of SDH [16], the STM-1 is very similarly

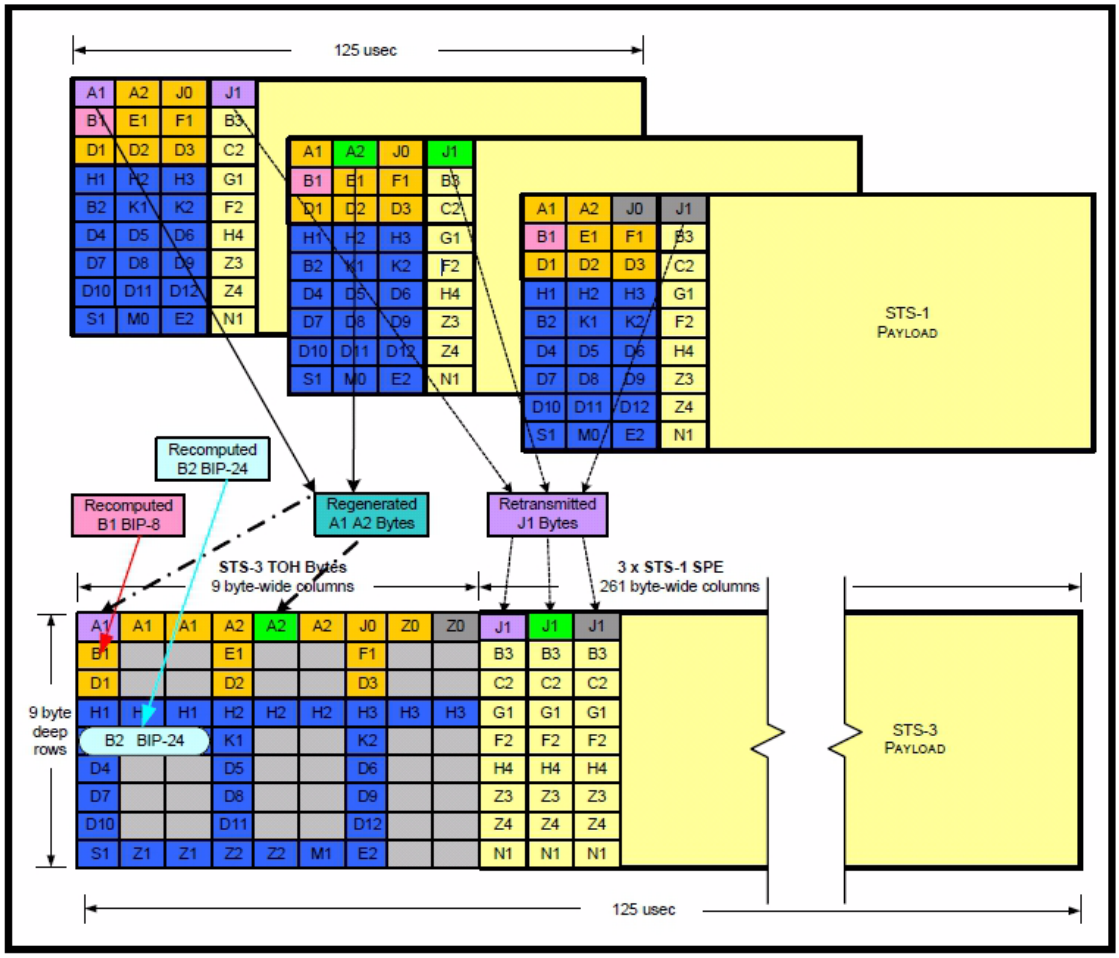


Figure 2-3: SONET/SDH Framing STS-1 and STS-3 [16]

defined as also having a 9 bytes TOH at the start of the frame, but with a 4th row reserved for *Administrative Unit* (AU) pointers and the SPE here is called a container. A container can either transport an AU or an *AU Group* (AUG), or *Virtual Containers* (VC). VCs contain a *Tributary Unit* (TU) or a *TU Group* (TUG). Within the TUs, the different services are transported. There are different time slots for sending AU and TU frames. The STM-1 not

only has a TU-3s mapping that is identical to the STS-3 structure, but also a TU-4 mapping that has only the 10th column as a POH using the rest as payload. This delivers a utilization of up to 96.29 % (260/270) for the transport layer. Like STS-N frames, STM-N frames are generated by multiplexing sub frames.

Synchronizing network links provides many advantages over an asynchronous transmission mode, especially when the line rate is the same at the sender and the receiver, so that the transmission of packets does not need any bit stuffing or synchronizing fifos at the data path. SDH/SONET uses such a scheme, where a high quality clock source is used as the frequency normal for the packet transmission. Links are started with a reference from a local clock oscillator and then after receiving a higher quality clock signal, they switch to the line-derived timing. Using high quality VCXOs in each router or concentrator, this synchronous transmission can be preserved. Table 2-2 shows the clock quality standards defined for SONET. An atomic clock source is required to achieve a quality level of Stratum 1 or 2, but for the others, a standard oscillator of high quality can be used. However, it is not always

Clock Quality Standard	Accuracy (in ppm)
Stratum 1	0,00001
Stratum 2	0,016
Stratum 3	4.6
Sonet Minimum Clock (SMC)	20

Table 2-2: SONET Clock Quality Standards

possible to run systems fully synchronous when they become bigger. It leads to having SONET islands, networks with asynchronous interfaces, connected by network elements having to handle two clock domains. When sending for example DS1 streams, a telecommunication signaling scheme, over SONET, the communication among several islands may cause slips. Here jitter and wander [17] become important, because their quality defines the

number of slips. A wander budget of 10.1 microseconds is defined for the SONET network element synchronization, which leads to national and local networks with less than an average of about 2.3 slips in 24 hours.

SONET/SDH is not a protocol, which should be used directly in a custom network to deliver flexible and high performance communication. However, it shows useful working principles that should be considered when defining a specific protocol. Analysis shows that synchronous solution for running timing critical networks seems to be a good solution to avoid special synchronization hardware and to eliminate the need for bit stuffing. The fixed framing format does not present the most flexible way to balance network communication and to consider specific user formats, but SONET/SDH provides an interesting scheme for data aggregation and increasing data flow speed. This is important for generating compact and efficient readout solutions.

2.3.2 Common network timing protocols

The standard protocol used for basic synchronization within the internet, local area networks, and even cluster systems is the Network Time Protocol Version 4 (NTPv4) [18]. NTP is an Application Layer protocol based for example on TCP/IP over Ethernet. The goal of NTP is to synchronize the Coordinated Universal Time (UTC) to all clients. UTC is not considering time zones or specific changes like daylight saving times. Servers and clients using NTP are synchronous within a few 10 microseconds. There are three operation modes it provides, a primary server, a secondary server or a client mode. A primary server uses a high quality clock source, usually a stratum 1 level clock, and delivers service for synchronization. The secondary server may provide several up-stream and down-stream links. It may not only provide synchronization for clients, but also for other secondary servers. Typically, the stratum level increases by one for each synchronization stage. Others are not capable of inheriting synchronization from a client. There are three synchronization schemes available within NTP: symmetric, client/server, and broadcast.

Three data types are defined to represent the time within NTP. The *date format* has 128 bits that uses 64 bits to represent seconds and 64 bit fractions to resolve in 0.05 attoseconds (as). 64 bits in seconds represent 584 billion years. Thus, they are divided in a 32 bit era field, starting at 0 hours 1 January 1900 UTC, and a timestamp field, counting 136 years. This timestamp field is also present in the *timestamp format* together with a 32 bit fraction, unitizing in 232 picoseconds. This format is, for example, used within packet headers. The third kind, the *short format*, has a seconds and fractions field reduced to 16 bit. It is used for delay and dispersion headers. The local time precision depends on the local clock precision. Quality of synchronization is influenced by jitter and wander.

During each synchronization sequence, the measured data values for offset, delay, dispersion, and jitter are calculated. A standard NTP synchronization sequence for an on-wire protocol delivering the measurement data is shown in figure 2-4 [19]. Therefore in different points in time t, time values T are transferred. Starting with the measurement algorithm, a client sends a time value T1, the send time t1, to a server where it is received at a time t2.

The server then sends all the values back together with its new send time value T3. A client can now calculate the required values for offset θ and delay δ by calculating the formulas (i) and (ii) with the stored times. However, since it has the basic set of values, all the other values can now be calculated.

(i) $$\theta = \frac{1}{2} \times ((T_2 - T_1) + (T_3 - T_4))$$

(ii) $$\delta = (T_4 - T_1) - (T_3 - T_2)$$

A set of samples of the quad-tuples (offset, delay, dispersion, and arrival time) is periodically stored. A clock filter algorithm analyzes them and calculates an ideal offset to discipline the system clock.

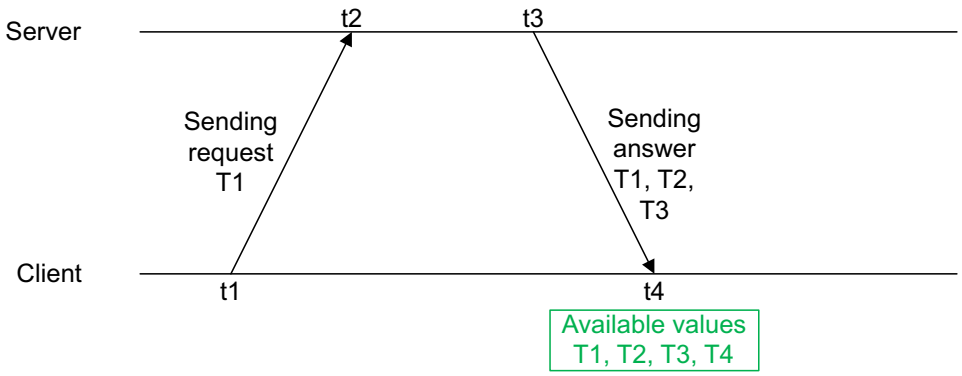


Figure 2-4: NTP - Synchronization Sequence - On-Wire Protocol

The Precision Time Protocol (PTP) [20] is also placed in the application layer and is designed for spatially localized systems achieving accuracy in the order of microseconds down to sub-microseconds. The time base for PTP is International Atomic Time (TAI). TAI is the average time derived from several hundred atomic clock sources. PTP is ideal for synchronizing control systems and may be used in larger research experiments [21].

While NTP uses a clearly defined server/client scheme, within PTP a flexible master/slave scheme is used. The PTP system has to fulfill some requirements like symmetric links and non-cyclic path forwarding. For each link it is negotiated which side is to provide a clock as

master and synchronizes with it as slave. Therefore, the best master clock (BMC) algorithm calculates the clock qualities. The master clock delivering the system reference clock is called the grandmaster clock. Thus, the protocol uses two phases. Phase one sets up the master/slave hierarchy and phase two synchronizes the clocks. For PTP, it is specified how to synchronize and syntonize different clocks. Clocks are syntonized, if they advance with the same rate. Clock are synchronized, if they even share the same epoch and are capable to sample values with a common tolerance.

The key for enhancing the synchronization quality for PTP is providing precise timestamps by taking the times as direct as possible on the ingress and egress port, removing all variable unnecessary elements out of the calculation. Therefore, it is not only defined to use the application layer for message timestamp generation, but also to use the kernel level or even direct hardware support to deliver precise time stamps. This scheme to focus on ingress and egress ports also works for active PTP switches. Here, the switch is treated as full link end-point being master or slave device.

Figure 2-5 depicts the PTP synchronization sequence. A master sends a *Sync* message to the slave. This message may directly include send time value T1. If this is not the case, a *Follow_Up* message is sent including T1. The slave stores the arrival time t2 together with the master send time value T1. Then a *Delay_Req* is sent by the slave device and its send

time value T3 is stored. The master calculates the *Delay_Req* receive time and sends it back within a *Delay_Resp* message. Finally, the slave has all four values required for its calculations.

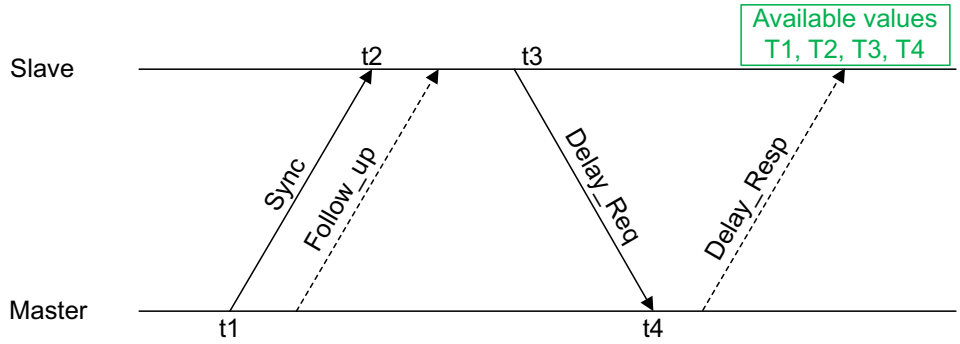


Figure 2-5: PTP - Synchronization Sequence

The standard timing protocols NTP and PTP deliver flexible solutions for application layer synchronization protocols that are able to run on top of numerous physical implementations. They are used to synchronize local times in computer networks, but the resolution is far too low for synchronizing detector electronics. The achievable resolution depends strongly on the hardware used and the fact that it is used mostly on the application layer increases flexible processing time and decreases accuracy. Nevertheless, for some specific detectors that have low timing constraints supported by special hardware, this might be a possible solution and the mechanisms used are worth keeping in mind when defining timing protocols.

2.3.3 White Rabbit

The White Rabbit (WR) project [22] was started with the support of several research groups and companies to enhance timing and control systems. The goal was to create a solution for a reliable, fast, and deterministic transmission of control information. This solution should be feasible for the needs of currently used and possibly tomorrows distributed network systems. WR mixes two standards: synchronous Ethernet and PTP. It is an open software and open hardware project hosted on the open hardware repository [23]. With its hardware support and the precise measured fixed link delays, this protocol is able to achieve sub-nanosecond accuracy. One of the central new designed elements is the WR switch. It is a fully compatible standard 802.1x switch, however it uses an extended feature set that is capable of serving as a active clock distribution and a synchronization stage. The WR extension features are only activated by passing a WR initialization sequence. The latest development version of the switch is WRS-3/18. It is a standalone switch with 18 SFP connectors, which is planned for availability at end of 2012.

Figure 2-6 depicts the extended synchronization sequence used to start-up WR [24] links. A PTP master periodically sends modified *announce* messages. A WR device recognizes the special suffix and uses the modified BMC algorithm to integrate into the network. Then the slave sends a *SLAVE_PRESENT* notification. The master sends a *LOCK* message to indicate the syntonization start for the slave hardware. When the slave syntonization is finished, it answers with the *LOCKED* message. After receiving the *LOCKED* message, the master then sends a *CALIBRATE* message to request calibration patterns. All fixed delays for transmission and reception are calculated. As soon as these hardware mechanisms end, the master sends the *CALIBRATED* message. Now, the slave can start the identical calibration for its side. After this WR initialization is finished, a *WR_MODE_ON* message is sent by the master to indicate its completion. Then the standard PTP sequences are performed. Thereby WR master and slave have become syntonized and synchronized.

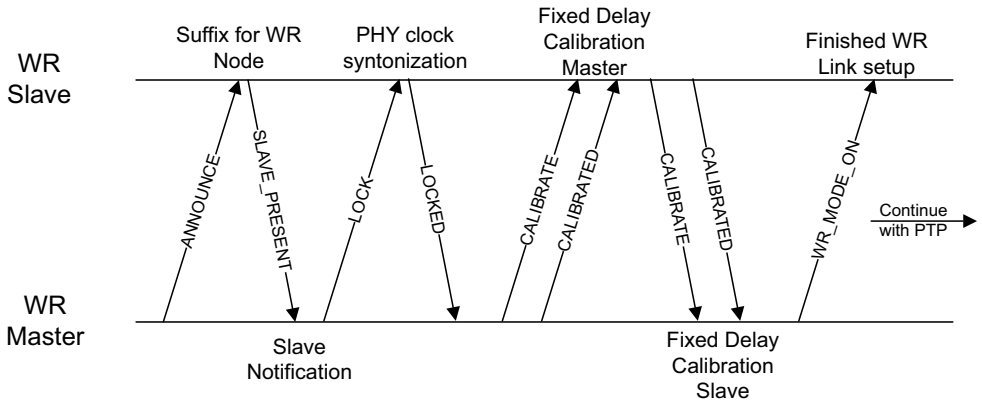


Figure 2-6: White Rabbit Synchronization Sequence

The White Rabbit implementation is a hardware supported PTP implementation based on Ethernet. One of its advantages is compatibility for Ethernet components, which decreases all costs. Its synchronization and synthonization features are useful and it is able to establish a synchronous network. However, mapping a well defined deterministic synchronization tree onto it to achieve very low precision is not possible. The bandwidth abilities are restricted to standard Ethernet speeds being reliable, but far away from high performance systems. Thus, WR is not a readout system for very timing precise or data intense detector systems, but it seems to be a low cost and easy to handle solution for most commonly used systems and for standard detector instrumentation.

2.3.4 Large Hadron Collider

The largest and most famous example for a research detector readout system is the Large Hadron Collider (LHC) experiment at Conseil Européen pour la Recherche Nucléaire (CERN), which is described in detail within design reports [25] and illustrated to the public [26]. It is comparable to FAIR at least in its challenging complexity as well as in some parts of the requirements. FAIR is a lot smaller versus LHC, as the largest accelerator in the world with a circumference of more than 26 km. LHC produces, especially with its leading experiments, huge amounts of data and frequent need for hardware updates. However, the general problems and tasks are similar even knowing that LHC is not having that strict placement, data aggregation, and timing constraints. LHC contains six different experiment setups such as Alice, Atlas, CMS, LHCb, TOTEM, and LHCf. They are all organized by international collaborations. All of these experiments need a specific developed control and synchronization system to guarantee a sustainable data acquisition system, because standard solutions do not fulfill their requirements.

The initial LHC setup implements a dedicated timing, trigger, and control (TTC) network [27] to distribute the clock, broadcast trigger, and control information over unidirectional fiber links. TTC has a slow and a fast timing system. The slow timing system delivers the UTC time in a resolution of 1ms. UTC time is used for data tagging and post mortem applications. The fast timing system distributes its signals through all experiments and the beam instrumentation of LHC. Their frequencies are synchronous with the circulating beam around the standard clock frequency of about 40.07897 MHz and an orbit frequency of about 11.2455 kHz

The TTC distribution topology is presented in figure 2-7. From the timing generators placed at the SR4, all signals are sent encoded using one 9.5 km long phase-stabilised singlemode fiber and standard TTC components to the Prevessin Control Room (PCR). PCR contains high power transmitters supplying a fanout tree through 1:32 optical tree couplers to manifold the orbit and clock signals. Via optical fibers from here, these signals are distributed to all experiments, the instrumentation, and other areas. In an experiment area, the TTC

machine interface crates (TTCmi) are responsible for further processing of signals and spans the local experiment distribution trees. At the tree leaves a low noise of 160.316 MHz VCXO in a PLL reduces RMS jitter of the 40.079 MHz clock to 7 ps and further distributes the clock electrically. To compensate time-of-flight delays in the detector as well as skew in the TTC network itself, programmable deskew taps with a minimum granularity of 100 ps are used. Data packets are broadcast using a time-division-multiplexing arbitration.

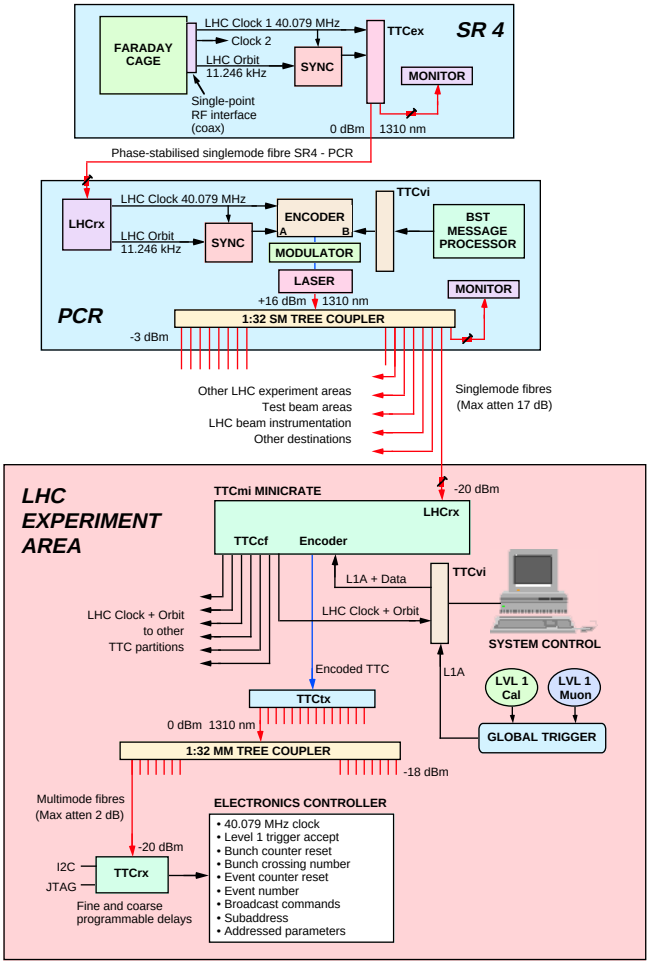


Figure 2-7: LHC TTC Structure [27]

Due to higher beam luminosities for the planned LHC upgrade, all electronic components are required to provide higher data rates and must be capable of sustaining high radiation doses. Therefore, the GigaBit Transceiver (GBT) project [28] was started at Cern. This

project is planned to deliver a complete front-end electronics (FEE) readout solution including electrical components of a radiation hard optical link enabling usage of a fast versatile link attached to standard component devices in an Off-Detector region. Figure 2-8 shows the GBT structure using the concept of a unified communication to transport a *Timing & Trigger*, a *DAQ*, and a *Slow Control* channel over one optical link. The GBT protocol [29] uses frames of 120 bits for communication at 40 MHz to reach the 4.8 Gb/s GBT link capability. The general framing structure includes four bits of header information and 32 Bit Reed-Solomon forward error correction (FEC) coding resulting in 84/120 = 70 % payload. The payload contains a four-bit slow control field and an 80 bit data field. It delivers a data utilization of up to 80/120 = 66.67%. Scrambling is used onto the links. Thus, a link can transport 3.2 Gb/s data. The TTC system uses a 16 bit field within the data for spanning a timing tree. These 16 bits are directly used for broadcasting a parallel trigger to the FEE.

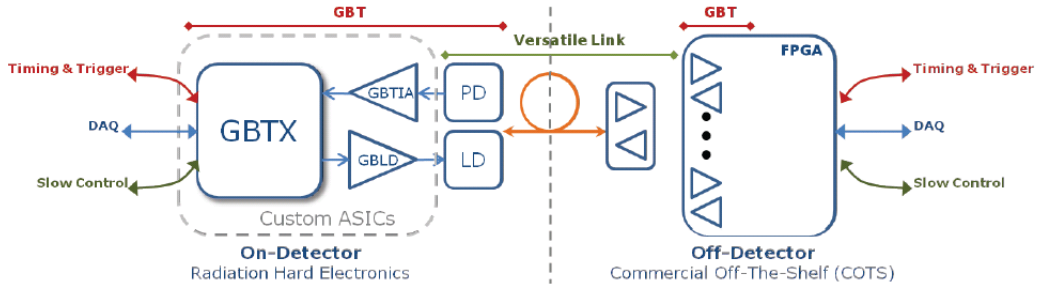


Figure 2-8: GBT Structure [30]

The GBT prototype custom ASICs [30] in 130nm technology are a transimpedance amplifier (GBTIA), a laser Driver (GBLD), and a serial transceiver (GBTX). They are capable of operating at 4.8 Gb/s using a custom GBT-SERDES. The serializer, as depicted in figure 2-9, first uses three 40 bit shift registers filled by 120 bit input registers. These 40 bits are then serialized by a factor of 40 before they are division multiplexed 3:1 into the 4.8 Gb/s stream. Special within the deserializer is the usage of a clock data recovery (CDR). This CDR uses a pre-calibrated VCO to assure that it can always be locked to the data. Front-end devices are attached to the GBTX by using two 40 bit double-data-rate (DDR) buses as input and output structure directly mapped into or from the 80 bit data field. The buses can

be configured into e-links providing 40 links each with 80 Mb/s, 20 links delivering 160 Mb/s, or 10 links delivering 320 Mb/s data bandwidth. Due to TTC, a maximum of 16 FEE devices can be attached to one GBT. Additionally to FEE devices, GBT-SCA devices can be attached to an e-link. A GBT-SCA is used as interface to be compatible to standards like JTAG or I2C. For easy integration into FEEs and other devices, an e-port macro is supplied.

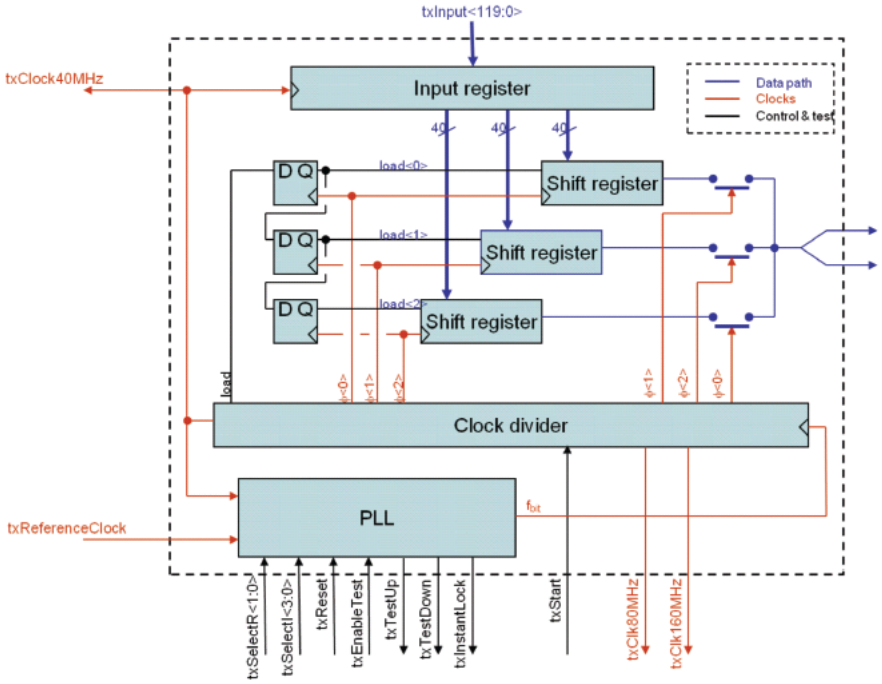


Figure 2-9: GBT Serializer Block Diagram [30]

Having a separate clock distribution is a disadvantage of the LHC solutions because of the additional space required. However, the GBT project seems to be a very interesting solution for challenging radiation requirements at a reasonable speed. It provides unified traffic over one link and delivers fault tolerance. Additionally, the e-port modules, which provide the e-link implementation in different technologies to other devices, show the interesting feature of reusability. The connection among FEE devices and the GBT having DDR multiple lanes of 80 MB bandwidth does not seem to be ideal for all applications. CBM needs more efficient data rates per connection to FEE devices, less radiation tolerance, different synchronization mechanisms, and additional data aggregation capabilities to achieve significantly

more bandwidth and data rates within inner hierarchy levels. Especially concerning bandwidth per area capability required within CBM, the ASICs have to concentrate on different parts to increase read-out bandwidth per FEE device and bandwidth towards the back end.

2.3.5 The HADES DAQ

The High Acceptance Di-Electron Spectrometer (HADES) experiment and collaboration [31] at the GSI in Darmstadt has developed their own DAQ system [32]. Figure 2-10 depicts the HADES DAQ structure. The two major components in the network are the Trigger and Readout Board (TRB) and the Hub board. The TRB is a FPGA-based board delivering 128 TDC channels for attaching all timing relevant detector types, a DSP processor, a 2 Gbit/s optical link, and a CPU with attached Ethernet for slow control purposes. A Hub is a FPGA board containing 20 different 2 Gb/s optical links, one of them used as a Gigabit Ethernet connected to a computer. It is responsible for network interconnection, not only to stream the data, but also to attach the central trigger system (CTS) and Slow Control. The server farm and the control room are integrated using Gigabit Ethernet.

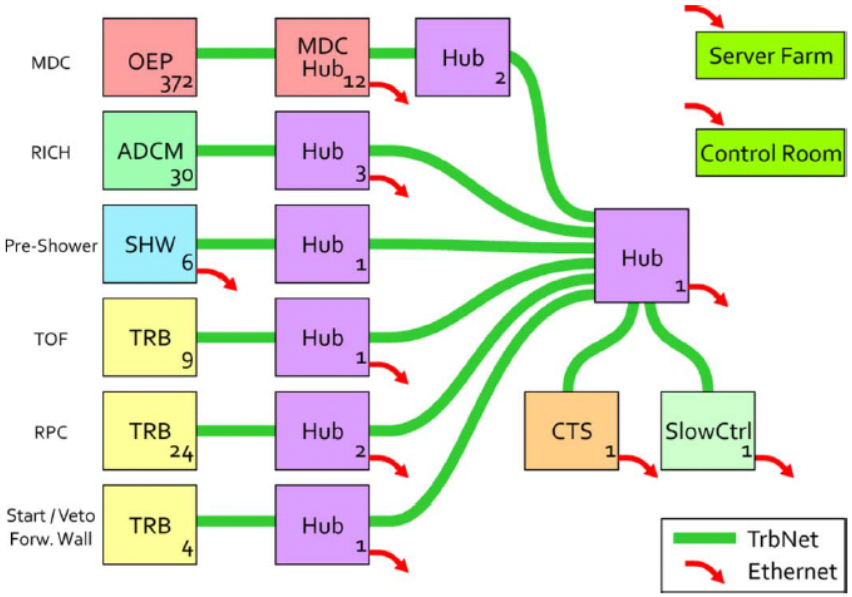


Figure 2-10: HADES DAQ Structure [32]

TRB has a separate Ethernet Slow Control, but placement constraints require using only the optical links for communication. Thus, the TRB Network (TrbNet) protocol was created. TrbNet integrates monitoring, and slow control features as well as readout and trigger distribution into a single network protocol. Only the clock distribution and synchronization is done through a separate clock tree. In addition, TrbNet is optimized to transfer trigger events with a latency of less than 5 µs through the hierarchy of the DAQ network. In order to guarantee low latency, packet length is restricted and arbitration prioritizes the fixed sized trigger packets.

TrbNet transports packets of 80 bits protected by an 8 bit CRC. In addition, 8b/10b coding is used within link level communication. A CRC check searches for erroneous data and an automatic hardware retransmission is initialized if it is required by sending a request back to the sender. One packet consists of a 16 bit header, a 64 bit payload for data or control, and an 8 bit CRC. Thus, TrbNet delivers a data utilization of 64/88 = 73%, considering 8b/10b coding up to 58% total utilization.

CTS is responsible for the trigger and readout process. Therefore, its trigger decision is first propagated to the frond-ends by using a dedicated differential signaling. It arrives 500ns after the event with less than 20 ps jitter and then the readout is started. Then a trigger packet is sent to inform the front-ends about the type and all required information it has to know to decide how to proceed with data processing. At the end of a trigger sequence, a *busy release* packet is transported back to the CTS and then the next trigger process can be performed.

All HADES DAQ system features are well-optimized concerning their density and trigger performance. An advantage of the system is using unified traffic for communication of control, data, and some trigger information. The disadvantages, compared to CBM, are its use of a separate clock distribution and trigger connections, and also its bandwidth is restricted to Gigabit Ethernet and 2 Gb/s optical links. Nevertheless, it is an exemplary custom build experiment readout system which shows solutions for all essential DAQ parts.

2.4 Conclusion

All the presented solutions show interesting features and abilities, though unfortunately no solution directly fits the requirements for CBM. While analyzing requirements and studying the state of the art technologies, it becomes clear that the final required CBM network protocol and PHY implementation needs to be modular, support for easy data aggregation, additional data transfer frequency acceleration features, synchronous network capabilities, clock distribution from one single source, and a unified traffic including synchronization, control, and data. Adapting these schemes for CBM detector read-out systems provides the clock distribution, together with the packet transport, in a well-defined environment over the same links. This reduces the amount of required lanes to a minimum. In addition to the frequency lock of all detector endpoints (leaves) in the network, a precise phase alignment and global time synchronization must be supported. Moreover, the detector electronics need to be phase synchronous and the latency of the clocks to each leaf must be deterministic and fixed during the measurement time. This completely synchronized system, with a bit-clock precise deterministic behavior in a well-defined environment, compensates problems with wander. All kown components have been analyzed and it is assured that, in respect to the length and technology constraints together with possible temperature variations, the wander remains within the tolerance budget. For example, calculating the time variation within the used fiber having 50 m with a temperature difference of $\Delta T = 10$, the fiber refraction index slightly changes and it results in a time variation of $\Delta t = 1,6678 ps$. The priority insertion of the deterministic latency messages described in the following chapter 3, *Development of a Synchronous Network for CBM*, allows it to precisely time all endpoints and measure link latencies with resolution on a bit clock accuracy level for the high speed SERDES of up to 200ps.

Chapter 3

Development of a Synchronous Network for CBM

This chapter gives an overview of the CBM DAQ structure and describes the CBMnet link protocol developed and used to interconnect devices within the CBM network. An analysis of the requirements and features concerning this protocol is presented. It provides high-speed SERDES functions. Further different communication traffic classes are supported. In addition, for purposes of synchronization, deterministic latency communication over each link is guaranteed. CBMnet delivers point-to-point interconnects using single SERDES lanes and internal 16bit interfaces. The features of this protocol are presented, such as system calibration and fault tolerance. The specialty of CBMnet is an innovative mechanism for synchronization, providing a precise synchronization on a bit clock level. The protocol is developed in the context of the CBM Experiment part of FAIR located at GSI.

3.1 The CBM Network Overview

Early analysis of the requirements and special demands of the CBM experiment Data Acquisition (DAQ) System resulted in designing a hierarchically structured CBM network [33]. This CBM network used a first version of the CBMnet generic link protocol [34] for demonstrators and detector readout systems running unified over a single bidirectional fiber link serving all functions and providing all different traffic classes. The unified links enable compact and efficient readout structures helping to achieve the CBM network requirements. When designing the CBMnet and a hierarchical readout structure for CBM, experiment requirements such as limited space for hardware, radiation tolerance, potential separation, flexibility for all different types of network traffic, support for various types of hardware, and delivery of synchronization mechanisms were considered. In addition, for protocol optimization, data flow behavior received special attention and the user-friendly modular concept was implemented to assure usability within different devices. The experiment data flow is unidirectional from detector front-end electronics towards a cluster farm at the back-end. While control messages are mostly initiated at the backend or a dedicated control node being exchanged in both directions. A precise clock and synchronization scheme provided to front-end detector readout systems supports their self-triggered hardware design. The CBM network guarantees a synchronous deterministic behavior during runtime, otherwise they could lose their synchronization and correlation of captured data could be lost. All administration packets are transmitted only on a link level. Thus, they are transparent to all network users.

The hierarchically structured network used for the first demonstrator and as a basic concept for test beam times is shown in figure 3-1. Within the readout system, different types of detectors are directly connected to the *Front-End Boards* (FEB). In the first versions, they were assembled with a varying amount of *n-XYTER* ASICs [35]. The n-XYTER chips are self-triggered and data driven detector readout ASICs. A *Readout Controller Board* (ROC) [36] located next to the FEBs is responsible for their initialization and control. Data measured by n-XYTERs is collected by a ROC and subsequently sent through the links of the

network. Therefore, each ROC interfaces network links. For the first tests, Ethernet links were used, but they were then exchanged by optical links using the CBMnet protocol to reach all necessary bandwidth requirements and synchronization needs. In the next hierarchy level, *Data Combiner Boards* (DCB) are used to aggregate data from several ROCs and deliver clock, synchronization, and control messages to the ROCs. As DCBs, a FPGA prototyping board [37] is used. It delivers six Small Form-factor Pluggable (SFP) connectors to attach several devices via an optical communication, a FPGA big enough to host a crossbar and all required hardware structures for CBM, and extension connectors for special purpose needs. Up to four ROCs can be connected to each DCB, while the other two SFP connections are used for either a separate clock and synchronization source and a data sink, or one unified link. *Active Buffer Boards* (ABB) [38] are then used to interface PCs or DAQ cluster farms. In addition, an ABB can serve as a clock and synchronization source. During the processing of data, an ABB sorts it according to temporal and special information. Immediately

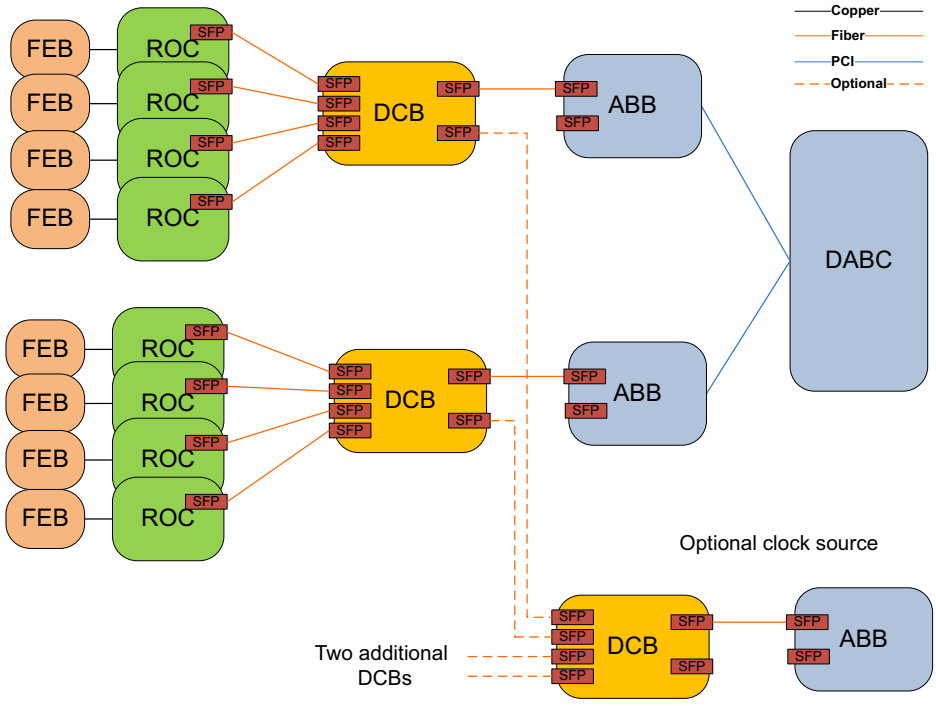


Figure 3-1: CBM DAQ Prototype Structures - Demonstrator I & Optional Extension

after processing, it is written into specified local buffers. The *Data Acquisition Backbone Core* (DABC) [39], [40] software framework is used as a general-purpose control and readout software for the DAQ. In addition, it serves as data flow engine in the DAQ backbone [41].

These prototype structures were the platforms used to develop the CBMnet protocol [42]. The CBMnet features are presented within this chapter with all its demands on the network and flexibility to support different CBM network readout hierarchies. In addition, prototype structures were used to verify CBMnet functionalities like precise time synchronization over a single unified bidirectional link [43]. The reliable, bit clock synchronous message-based synchronization scheme has proven its functionality and is ideal for synchronous counter reset in detector FEE. Towards the final experiment, protocol and DAQ network structure have been refined [44]. A second CBMnet version especially with extension on data reliability was created. Even if other groups have shown that there are techniques for using FPGAs in radiation areas [45] to readout some of the detectors, there are still the most demanding detectors requiring a large data bandwidth with early data aggregation and higher link speeds, which require a radiation tolerant ASIC capable of these challenging tasks. Additionally, the data preprocessing and sorting strategies changed. Therefore, a new DAQ structure is planned to be used in the final setup. Figure 3-2 presents an overview of it, designed by the project administrators.

The front-end near detector region now includes two variants to control and readout next generation of specific readout ASICs required for the final experiment setup. One is an improved readout of FEEs using FEBs directly attached to a new version of ROCs [46]. This new version uses an updated FPGA and can serve as ROC, DCB, or even ABB within the next prototypes. It has been designed considering all these demands, in addition to the features required for a final readout FPGA board. The other one used in higher radiation areas requires an ASIC to fulfill the density and radiation constraints for some of the detector types. Both variants are closely coupled with opto converters for electrical optical conversion to enable long distance communication. The subsequent CBM service building

region contains planned data processing boards (DPB) and the experiment control system (ECS) are placed. The DPB delivers the unified functions of a DCB providing data combining and preprocessing together with the detector control system (DCS) responsible for clock distribution and synchronization. The ECS is used to provide the clock distribution and a synchronization control network for the DPBs to enable all required features within the CBM network. The 'Green Cube' called region is the compute cluster area including a FLes Interface Board (FLIB) to receive the data and store it in an appropriate way for a first level event selector (FLES). The FLES is responsible for selection of interesting events and thus supports the computing cluster concerning data processing. First FLES analyses have already been done [47]. Additionally, some analysis concerning possible cluster structures and high performance event building [48] has been performed.

In the following sections, the CBMnet and its features are described, while its device integration and the HUB ASIC required for the demanding detector readout systems are presented in the subsequent chapters.

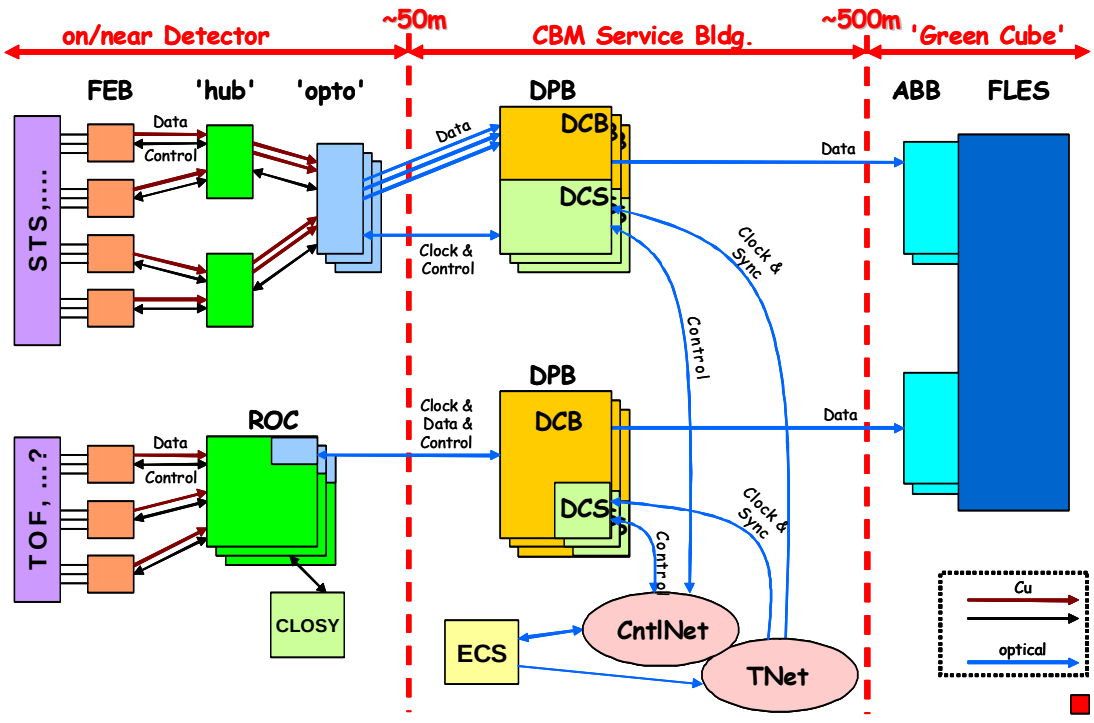


Figure 3-2: CBM DAQ network structure overview [Walter F.J. Mueller, GSI, 2011]

3.2 CBM Protocol Introduction

The CBM Network Protocol (CBMnet) works using point-to-point network connections. Point-to-point connections were selected, because they are ideal for CBM. They enable an efficient readout hierarchy and deliver the fastest possible connection between devices. The network switching is done within intermediate devices and point-to-point connections are usable for synchronization. In addition, a clean partitioning into communication layers within the CBM network simplifies reusability and eases CBMnet integration. Therefore, CBMnet modules implement the communication layer 2 (data link layer), layer 3 (network layer), and parts of layer 4 (transport layer) within the layer definition of the Open System Interconnect (OSI) Reference Model [49]. These CBMnet modules represent the generic link protocol (GLP) implementation used in all hierarchy stages and on different hardware platforms. It is always supplemented by a layer 1 (physical layer) implementation for the used hardware. In its first version, the protocol worked without routing, because specific communication is restricted and optimized to build-up a first test demonstrator. Therefore it is certain, that data will only be sent from the detector into the data acquisition (DAQ) System to a specific receiver and control packets are only sent in the opposite direction to all detectors. Routing information has been added later in the next layer, which is needed to enable message forwarding into bigger networks and to use a selectable control-node, either as DCS or as ECS. When the system is started, an automatic sophisticated link initialization is performed. After initialization is done, the network can be used to send messages. In case of a link failure, network links are automatically reinitialized. Initialization, packaging, CRC checking, and handling of special network functions are done within the CBMnet modules. Figure 3-3 shows a diagram of a network device, including the CBMnet modules connected to the network via SERDES or MGTs and to a core module. This core module represents custom modules different for all stages of the CBM network. All link ports in the network are running with the same CBM link protocol logic, but depending on the function, they may not use all parts of it.

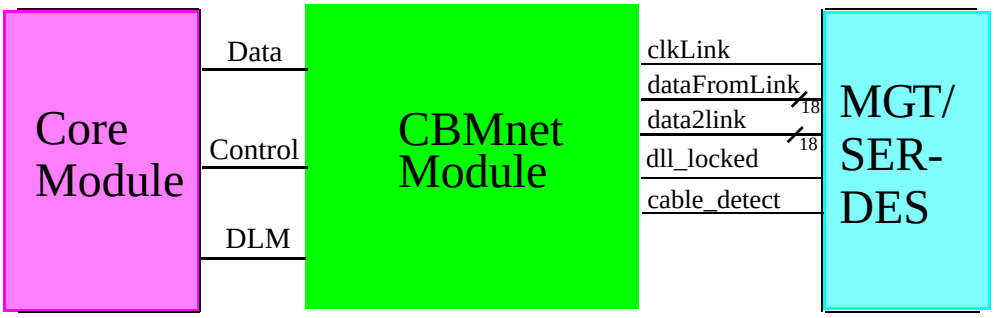


Figure 3-3: CBMnet Overview

In the following sections, the protocol structure including its functionality, the special characters definition, and the packet format are described. Then the implementation of the CBMnet, especially in a detail description for the main modules, such as the packet generator, link in, and link out is presented. Additionally, the definition of the optimized interface for communication and synchronization with an external Core module is described.

3.3 CBM Protocol

3.3.1 Traffic classes

Four different services have to be delivered by the CBM network in order to enable functionality of the FEEs. These services are a clock distribution, a control message channel, a data message channel, and a synchronization mechanism. Of course, all these services can be provided using different nets, but especially due to space and cost constraints, they must be delivered through the use of one unified bidirectional link connection. Clock data recovery (CDR) mechanisms are used to recover the clock out of a received data stream and supply the system. All other services could be handled using one message type. One way to implement this is using fixed frames containing fixed fields for all of them. This is not useful for CBM, because of its inflexibility and the waste of utilization. A second variant is

using one message type, which has to be processed afterwards to find the right service and makes it difficult to use specific message features. Both of these solutions do not seem to be efficient enough for CBM. The utilized solution is to use one message type for each service and run all required services in parallel over the link. Here again different possibilities are given, for example, fixed slots for each service or usage of virtual channels onto a link. However, utilization and flexibility are the reasons for preferring a virtual channel solution. Different traffic classes are provided in order to fit the requirements of the specific message types in the Network. Each traffic class is assigned to one virtual channel to make these traffic classes independent of each other. Moreover, the classes have different priorities to access the physical link. The following traffic classes are supported:

- Deterministic Latency Messages (DLM)
- Data Transport Messages (DTM)
- Detector Control Messages (DCM)

The link layer has a special support built-in for time synchronization of large networks using Deterministic Latency Messages (DLM). This type of message is well defined and has a fixed length with packet size of only 16 bits. This smallest possible fixed processing unit is ideal for synchronization. DLMs are more of a special control character than a message, because they carry no real payload. Instead, they come in 16 special coded variants, where one part consists of 8bit values that encode the function and a second set of 8 bit values to provide a couple of different DLMs. The coding assures fault tolerance at least against one-bit errors by using a hamming distance. It is described in section 3.3.2. DLMs, as their name may indicate, must always have a deterministic latency in the entire network. Therefore, the most important feature provided by the network must be the *priority request insertion.* A similar mechanism is part of the HyperTransport protocol [50], but not with the intention of providing deterministic latency guarantees. The *priority request insertion* guarantees an insertion of DLMs at any given time with a fixed latency into the link even during data or control packets are sent. This leads to the fact, that there is always a fixed amount of

clock cycles for processing of DLMs in the link port. In addition, the network physical layer has to provide a deterministic environment. This must not only be true for a single run, it must be reproducible at any time a restart is initialized, so that the accumulated round trip time for DLMs always produces an identical total number of clock cycles in the CBM network. All these requirements assure that DLMs are an ideal vehicle for system synchronization or time critical service purposes.

For the traffic class of Data Transport Messages (DTM), the most important ability is high bandwidth for streaming the data. The variation in the amounts of generated data requires supporting flexible packet sizes to keep the readout system efficient. These requirements are reached by usage of an efficient data processing and an optimized packet structure. This leads to a link utilization for data of about 91.428% (data / (data + CRC + SOP + EOP) = 64/70) leading to 73.142%, considering the 8b/10b coding on the network links. When an additional routing scheme is used, described within section 3.8, raw data utilization reduces to around 88.9% (71.1%). In the first version of the CBMnet protocol, it was debilitating to correct errors in data packets, only detecting errors and marking erroneous packets was required. This was realized with a CRC and a marking mechanism using special characters. During the enhancements within the complete CBM system, it became a requirement to guarantee reliability for data transmission. There are two mechanisms delivering the required proportion of fault tolerance. One is the usage of a retransmission mechanism and the other one is the implementation of a forward error correction (FEC) mechanism. Because of the way CBMnet is designed, both mechanisms have to be used on the link level. End-to-end approaches deliver too many disadvantages within the CBM network and would thereby not fulfill all the requirements. In Table 3-1, pro and contra arguments for both methods are depicted. The CBMnet consists of small messages, because a restricted message length is necessary for network traffic balancing and for keeping internal buffer sizes small. Thereby, additional buffer space required for retransmission will only slightly extend the design size. Since the CBM experiment has a self-triggered approach, it does not rely on latency and so latency is not of high importance for data capturing and transport. Thus, higher latency for a retransmitted packet and runtime for FEC calculation is tolerable.

	PRO	CONTRA
Retransmission	- only some new logic - normal data stream is unaffected	- needs more buffer space - increased latency for retransmit - additional flow control packets into backward direction
FEC	- corrected data stream arrives in real-time	- significantly more logic - higher latency during runtime operation - link utilization decreases because of message overhead

Table 3-1: Retransmission vs. FEC

A major requirement for the CBM network is optimized link utilization and thereby a higher effective bandwidth for data streams. This is due to late event selection in a self-triggered system, so the amount of data taken is higher as in other experiments. The FEC reduces utilization by requiring some extra space for additional information to enable error correction. Retransmission needs less information to detect errors, but it requires additional administration for retransmission control. However, the administration packets are mostly flowing into front-end detector direction on an almost empty link, which has no effect on maximum data bandwidth. Thus, the error free data stream stays almost unaffected. The general system assumption within the CBM network is that errors rarely occur. This gives retransmission mechanisms a significant advantage over FEC. Additionally, CBMnet is used within various hardware devices with different speed requirements and sizes. Thus, additional complex logic to calculate FEC needs to be tuned for several implementations. In addition, required logic has to be placed for each link. This leads to a lot of complex logic placed on data aggregation and collection dies, which easily contain more than 48 CBMnet module blocks. However, these arguments led to the decision to implement a link based retransmission, because it fulfills all CBM network requirements. In case of retransmis-

sions, data sending restarts at the first erroneous packet and retransmits complete data streams from this point in time. At the receiver side, all data is rejected until a retransmission arrives. Thereby, the in order message delivery from each sender is guaranteed.

Detector Control Messages (DCM) as third traffic class is responsible for reliable delivery of control, monitor, and configuration messages. It required a fault tolerant design even in the first demonstrators and test beam setups. Here, a retransmission mechanism was also implemented. However, the first protocol version was resending only erroneous messages after the detection of an error. Due to the final setup requirement of in order delivery for control messages, this scheme was changed. The retransmission scheme used for the data channel, resending everything after an error occurred, is used within CBMnet V2. There may be some implementation stages supporting viewer credits for control messages than for data messages, because of the smaller size and total amount of control messages used.

In addition to the traffic classes, service packets like acknowledgements or idles are sent through the unified links. They are completely handled by link ports and are not visible to core modules. Service packets are coded as special characters. Thus, they are minimum size messages. The credit based flow control processing service packets is handled internally in the link port modules.

3.3.2 Special character coding

Due to the usage of fiber and copper in the CBM network in order to connect different devices, not only data coding, but also special character coding must be DC free. Therefore, 8b/10b coding [51] is used. There are various special characters required for framing as message separation and their detection, such as administration packets for credit control for DCMs and DTMs, also service characters for initialization and link handling including idle notification, and DLM flagging as a special purpose feature. These characters must be easy to capture, therefore allowing efficient message handling, switching, and aggregation. Thus, they must be as small as possible, clearly defined, reliable, easy to decode, and must also have a well-controlled running disparity.

For general coding length, 16 bits are used for special characters. This is equal to the width of processing hardware units and enables efficient hardware structures. The character coding enables multiple fault detection and a one-bit error correction for all special characters. The mechanism used to provide these capabilities is the Hamming distance [52] in respect to 8b/10b coding. Character combinations with ideal hamming distance were chosen [53] and their coding is presented in Table 3-2. A one-bit error correction should be sufficient in most of the cases, because single event upsets (SEU) are the most likely errors to occur. Retransmission logic or higher control levels cover the handling of other errors.

There are framing characters, like start of packet (SOP) characters, for data messages and start of slow control (SOSC), which are both in multiple versions for credit control and end of packet (EOP) markers for both message types. Additionally, administration characters are defined as acknowledgement (ACK), non-acknowledgement (NACK), and retransmission (RETRANS) for credit flow control. Service characters like INIT, ACK and IDLE are responsible for link initialization. SLAVE characters assure unbalanced lane handling. In addition, deterministic latency (DLM) characters are specified as autonomous traffic class. Usage of all these characters is successively described in the following subchapters and chapters.

Name	Upper Byte	Lower Byte	Name	Upper Byte	Lower Byte
SOP0	K 28.3	D 14.1	DLM9	K 27.7	D 3.1
SOP1	K 28.3	D 20.1	DLM10	K 27.7	D 11.2
SOP2	K 28.3	D 20.6	DLM11	K 27.7	D 17.2
SOP3	K 28.3	D 22.3	DLM12	K 27.7	D 25.3
SOSC0	K 28.3	D 28.2	DLM13	K 27.7	D 17.5
SOSC1	K 28.3	D 28.5	DLM14	K 27.7	D 3.6
SOSC2	K 28.3	D 6.2	DLM15	K 27.7	D 5.3
SOSC3	K 28.3	D 14.6	NACK0	K 28.7	D 10.3
ACK0	K 28.3	D 3.1	NACK1	K 28.7	D 14.1
ACK1	K 28.3	D 11.2	NACK2	K 28.7	D 20.1
ACK2	K 28.3	D 17.2	NACK3	K 28.7	D 20.6
ACK3	K 28.3	D 25.3	NACK00	K 28.7	D 22.3
ACK00	K 28.3	D 17.5	NACK01	K 28.7	D 28.2
ACK01	K 28.3	D 3.6	NACK02	K 28.7	D 28.5
ACK02	K 28.3	D 5.3	NACK03	K 28.7	D 6.2
ACK03	K 28.3	D 10.3	(EOP_ERR)	K 28.7	D 3.1
DLM0	K 27.7	D 10.3	EOP_C	K 28.7	D 11.2
DLM1	K 27.7	D 14.1	EOP	K 28.7	D 17.2
DLM2	K 27.7	D 20.1	RETRANS	K 28.7	D 17.5
DLM3	K 27.7	D 20.6	IDLE/SYNC	K 28.7	D 3.6
DLM4	K 27.7	D 22.3	INIT	K 28.7	D 5.3
DLM5	K 27.7	D 28.2	ACK (INIT)	K 29.7	K 28.3
DLM6	K 27.7	D 28.5	SLAVE1	K 28.7	D 14.6
DLM7	K 27.7	D 6.2	SLAVE2	K 28.7	D 25.3
DLM8	K 27.7	D 14.6	SLAVE3	K 30.7	K 28.3

Table 3-2:CBM Link Protocol Characters using 8b/10b Coding

3.3.3 CBMnet Protocol Structure

A flexible framing has been defined as a packet structure fulfilling the CBM requirements using different message sizes. It consists of start and end delimiters, the payload, optional routing characters, and a CRC. The data packet structure is depicted in figure 3-4. As shown, a data packet starts with a special character for start of packet (SOP). It is followed by an optional routing character, which is defined as a 16 bit source address of the data. Then, a data payload secured by a CRC is attached. The end of a packet is marked by an end of packet character (EOP). The granularity used during processing is optimized for 16 bits, exactly the size of special characters. Payload size is defined as a value between 8 and 64 bytes in multiples of 2 bytes, the 16 bit processing width.

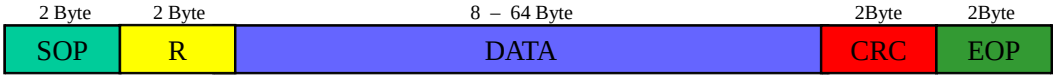


Figure 3-4: Data Packet Structure

The structure of control packets depicted in figure 3-5 is similar to the data packet structure. It differs in special character usage. Instead of SOP and EOP as delimiters, SOSC and EOP_C are used. In addition, the optional routing proposal is different. It consists of two routing characters, the destination and the source address. This has the advantage that by switching these characters, the generation of routing for answers is done.

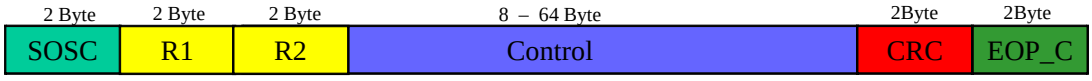


Figure 3-5: Control Packet Structure

The CBMnet modules responsible for processing network traffic are implemented within the CBM link port. The CBM link port consists of four parts such as *initialization control*, *packet generator*, *link port in* and *link port out*. The link *initialization control* module handles for automatically setting up the connections and detection for link failures, with self-

triggered integrated reconnect functionality. The next one is the *packet generator* module, which is responsible for converting data from the CBMnet interface to packaged messages and for unpacking data structures into the opposite direction. In addition, there is a *link port in* and a *link port out* module responsible for providing all the previously described special features for the different virtual channel and for controlling message transmission over the CBM network. In a different project, an *initialization control* module has been developed [54]. It is based on a standard initialization mechanism and its main part is an initialization control finite state machine (FSM). This *initialization control* was adapted to the CBM project needs and was used instead of rewriting it. The other three parts designed for CBM are described in detail in the following subchapters.

3.4 CBMnet Modules

The CBMnet protocol consists of three internal module blocks such as packet generator, LP_in and LP_out. Figure 3-6 presents an overview of the CBMnet module structure. As depicted, the DLM handling is done in separated logic with direct access to the LP_in and

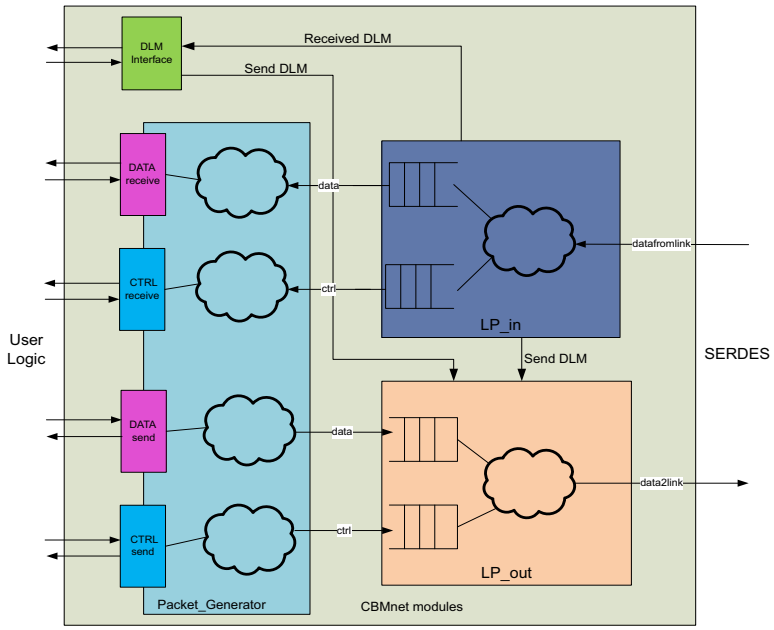


Figure 3-6: CBM Modules Overview

LP_out modules. Even within these modules, there is a prioritized extra DLM processing path. These terms are required to guarantee deterministic latency for DLMs. Between LP_in and LP_out, administration information is exchanged. If flow control permits, data and control messages are processed and packed or unpacked by the CBMnet modules. The CBM modules are described in the following sections.

3.4.1 Packet Generator

The packet generator is directly attached to the CBMnet interface. It is responsible for the conversion of DTM and DCM streams into the link packets used in the CBM network. This enables efficient message processing for following units. Because of its special performance, DLMs are bypassing this unit. They are transferred directly to and from the link innd link out port modules. Figure 3-7 presents the inner packet generator module structure. Each path consists of a processing logic and a FSM to control this logic. In the send direction, its main task is to insert the special framing characters and packetize the data or control streams transferred through the interface. Thereby, dummy placeholders are also inserted

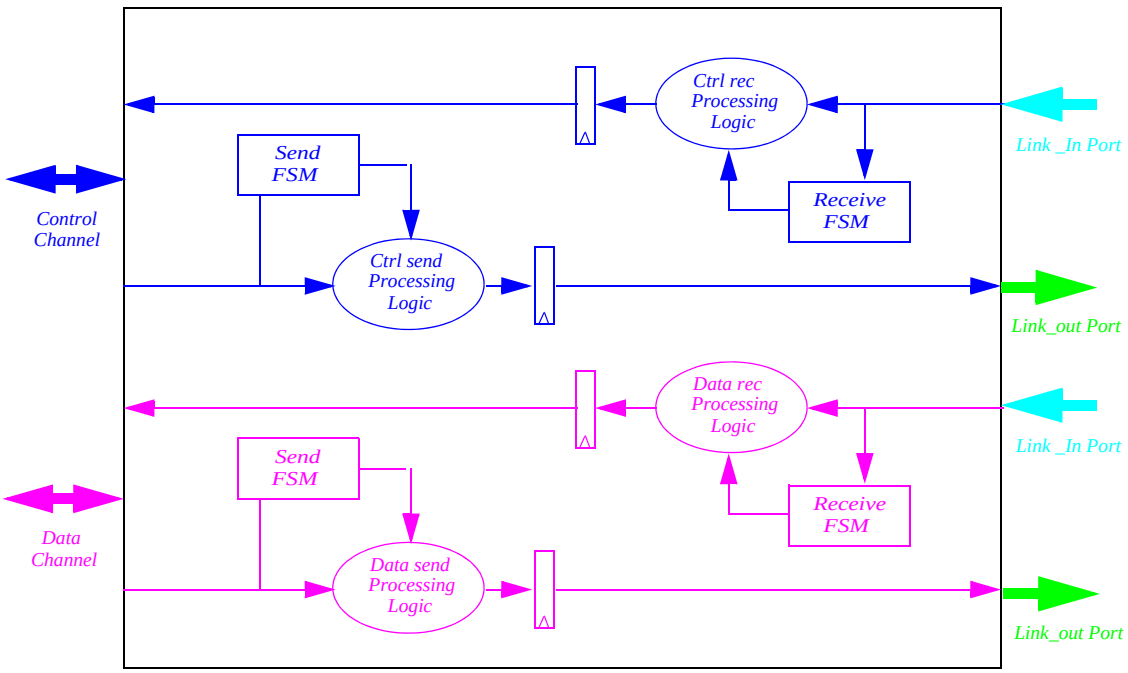


Figure 3-7: CBM Packet Generator

for the CRCs, which are calculated in a later stage and can then be exchanged at the right position. Incoming control flags make sure, that messages do have the right type and packaging information. For incoming messages, special characters and CRCs have to be removed and control flags for the core module have to be generated. Therefore processing logic blocks are also used, which are controlled by special FSMs. From this point on, it is guaranteed in the network that all processed messages can not be interrupted. This means when processing a message in either direction has been started, this message is always completely processed, even if a stop event occurs during that time. The hardware is built to ensure this behavior.

In figure 3-8, the control logic FSMs that process control path messages are shown. The send FSM depicted to the left, has a sequential flow of states building a message. As long as there is no stop asserted, data from the CBMnet interface is accepted by the CBM packet generator. It is internally delayed and processed in parallel to the FSM and controlled by this FSM. For each special character and the CRC dummy insertion, a one-clock cycle stop signal is asserted by the FSM for the CBMnet interface, because a wait is required to fill up the control stream. This is done at the end of each message. A started message is never interrupted and following modules have to take care that there is always enough buffer space left to store complete maximum length messages. Because of input stream delay and FSM behavior having three stop states, a message must have a length of at least four processing units.

Thus, a minimum message was defined with a granularity of 8 bytes, four times a 16 bit processing width. Message building is controlled by the control send FSM, while the next available credits are flagged by the link port out module to generate correct framing characters. The control receive FSM, shown at the left side of figure 3-8, simply reacts on a stop signal from the CBMnet interface to start and stop the message flow. It flags the payload as valid, while the 16 bit stream is handed over to the CBMnet interface. The data path FSMs work in an analogous manner.

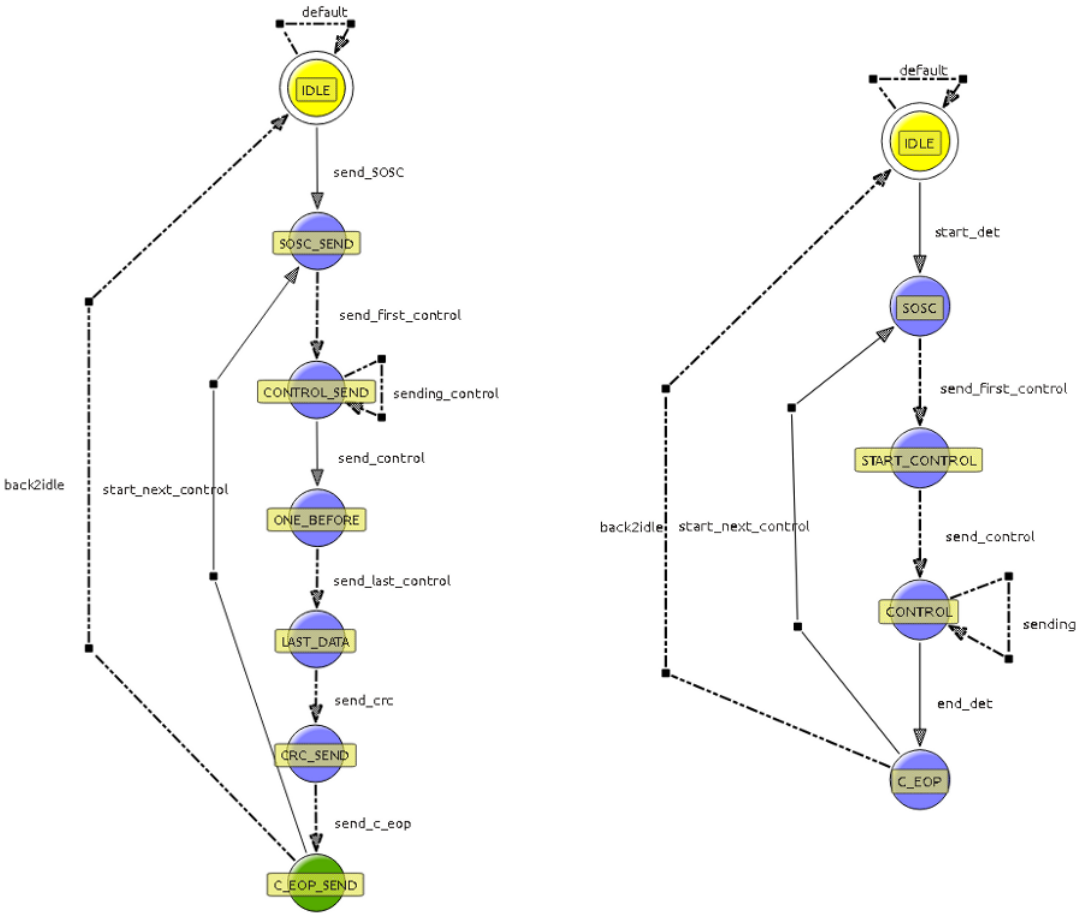


Figure 3-8: Packet Generator FSMs (control send FSM left, control receive FSM right)

3.4.2 LP_in

The *link port in* (LP_in) module is depicted in figure 3-9. The *decode* module is its main module. It decodes special characters to separate virtual channels and process all service characters. Arrival of service characters is flagged to appropriate units to process them as the link initialization or the link port out module for flow control. All these administration characters stay transparent to users and are completely handled within the CBMnet modules. Within the LP_in module, data messages are streamed into a *Data Receive Buffer* and control messages are received into a *Control Receive Buffer*. This is controlled by the signaling of the *decode* unit. A *CRC Checker* module processes package payloads in parallel and indicates its correctness to the *decode* module for further handling.

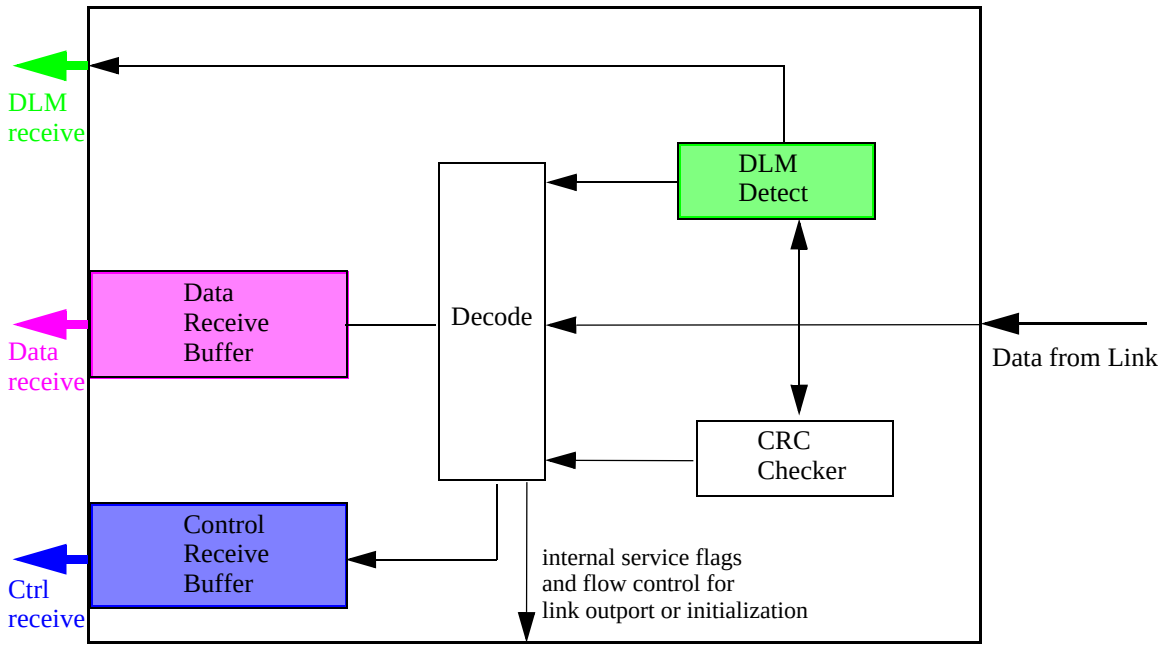


Figure 3-9: CBM Link Inport

Additionally, directly attached to the input stream is a *DLM Detect* module. It works in parallel, detecting DLMs and forwarding them directly towards the CBMnet interface. Due to the signalization of detected DLMs and a mechanism extracting them out of messages, they are ignored by the *decode* module and so messages stay consistent.

3.4.3 LP_out

An overview of the *link port out* module is shown in figure 3-10. It contains a multiplexer logic controlled by an *arbiter*, which inserts different virtual channel messages or service packets into the link. The *arbiter* control works via a request grant mechanism to guarantee correct packet insertion. Due to this special requirement for DLMs, the arbiter controls *DLM delay* stages which allows priority request insertion for DLMs. Therefore, all message channels have an optional path with a one-cycle delay. When a DLM appears, it is directly inserted into the output stream and afterwards the delayed message stream is sent. Then, the delay mechanism is reset, which causes a one-cycle pause in this channel. This mechanism

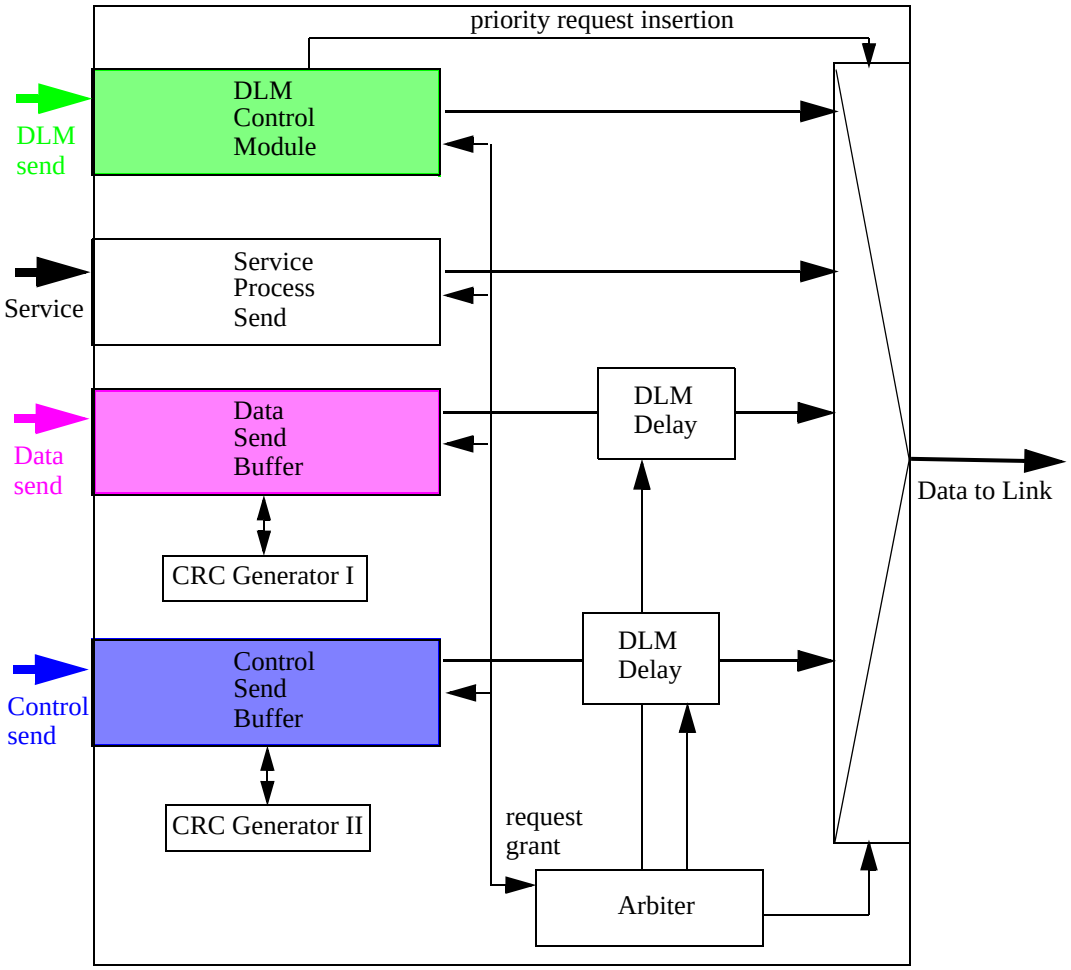


Figure 3-10: CBM Link Outport

restricts DLM usage to one per message. Two *CRC Generator* modules are attached to the *data send buffer* and *control send buffer* to exchange all placeholders for CRCs with new ones calculated on demand. The send buffers contains the complete flow control with retransmission functionality. All service packets such as link initialization character and administrative special characters are generated and then inserted by a *service* module. It receives the requests for character generation directly through flags either from the link port in module or the initialization module.

In the send buffer modules, the flow control and retransmission handling for the channels is done. It includes an administration unit, which keeps track of virtual credit status conditions for all credits. Messages are sent in order from the buffers as long as there are more credits available. Even if there are only four start characters used, a user-defined number of credits are available. This becomes clear considering the fact that a reliable network is used and the flow mechanism can reuse all four characters at any time. The assumption for the network reliability is that one-bit errors are corrected and complete characters do not get lost. However, if this is the case, the link is down and a reinitialization is required. As start characters for packets, four characters are used counting from for example SOP1 to SOP4, afterwards it continues again from SOP1. The messages are acknowledged by the appropriate acks, for example an ACK1 for SOP1 or an ACK01 for SOSC1. In case of an error a non-acknowledgement, for example a NACK1 for SOP1 is sent. A NACK is sent if the CRC check fails or an order inconsistency is given. Every following message until a retransmission, marked with a RETRANS char, occurs sends the same NACK. The link administration unit in the buffer module is responsible for freeing buffer space on ack appearance and initiating retransmissions starting at erroneous messages. Additionally, there is a timeout implemented for the retransmission mechanism in case of unforeseen errors.

3.5 Interface

The Interface for using the CBM network protocol represents the communication layers 1 to 4. The CBMnet is identical in all communication devices in the CBM network hierarchy. The physical layer implementation is hardware dependent and delivers all required features to transport the CBMnet. For optimization purposes, the CBMnet interface, as depicted in figure 3-11, represents all three virtual input and output channels for the traffic classes used as Deterministic Latency Message (DLM), Data Transfer Messages (DTM) and Detector Control Messages (DCM).

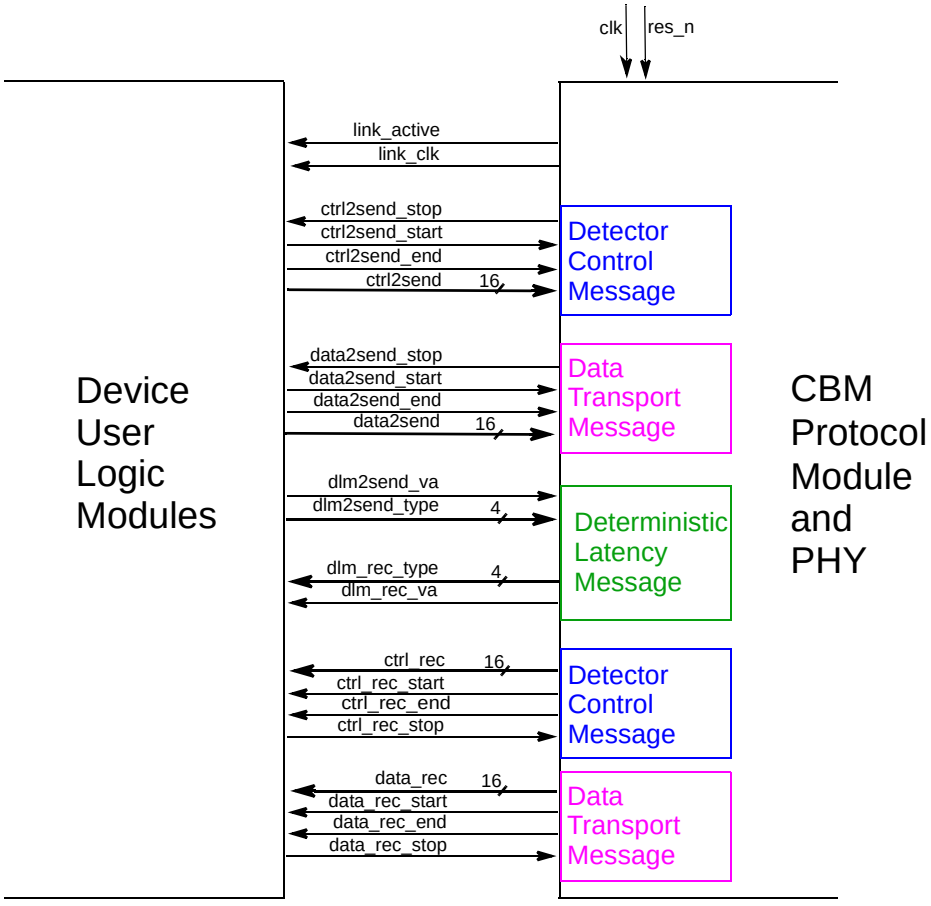


Figure 3-11: CBM Link Interface

The Interface synchronization for all three traffic classes is done with a common valid and stop implementation. The flow control between the CBM protocol module and the user logic module, a free definable user module for sending data streams, contains a start signal and a end signal for forward flow control and a stop signal for reverse flow control. The interface might be asynchronous and therefore could need a separate synchronization mechanism like a synchronizing FIFO between the source synchronous networks including the CBMnet protocol modules.

The signal description for the CBM network interface is presented in Figure 3-11. The *link_active* signal shows if the network is ready to send and receive data. When this signal is set, the initialization sequence has been successfully completed and from this point in time, it is possible to transport data through the network. The initialization sequence needs a clock signal *link_clk* of the corresponding receiver link. It starts after a link clock is stable. The source synchronous CBM protocol module runs with 125 MHz for usage of 2.5 Gbit/s link and scales linear for higher frequencies. If user logic is not running with the recovered receive clock delivered by the interface, the data and control messages must be synchronized before insertion into send streams.

All three virtual input channels are presented internally as complete separate streams and an arbitration mechanism assigns them onto the link. Due to its priority request insertion, DLMs always have the highest priority if inserted. For the virtual channel of DLMs, there are the signals *dlm2send_va* and a 4 bit wide *dlm2send_type* to show if there is a DLM, which has to be sent and has to depict its number in binary coding. The receive path signals for DLMs work analogously with the signals *dlm_rec_va* and *dlm_rec_type*.

Data flow synchronization is performed with a variant of the valid-stop synchronization on a granularity of packets. Information on general functionality and ideas of the valid-stop synchronization mechanism can be found in [55]. For a send path all special characters are inserted automatically in the CBM protocol module, for example start of packet (SOP), CRC, and end of packet (EOP). This is the reason for using a variant of the pure valid-stop synchronization mechanism for passing the 16 bit width data payload *data2send* or control

payload *ctrl2send* through the interface. It is realized by using the *data2send_stop* and the *ctrl2send_stop* signals as common stop signalizations and a flag *data2send_start* together with a *data2send_end* respectively, and a *ctrl2send_start* together with a *ctrl2send_end* to implement a more detailed valid signalization. These tags are required for optimal data delivery at the interface and for a more time efficient and proper data processing in following units. While the *data2send_end* and *ctrl2send_end* signals are exactly one clock cycle active to indicate the end of a packet, the *data2send_start* and *ctrl2send_start* signals stay active until the first data is accepted. If an error occurs within the CBM network, and a data or control stream is erroneous, a retransmission mechanism is activated and the erroneous stream is retransmitted exactly after the last received correct message. This is transparent to the user and the interface will continue delivering messages after successful retransmission. All received messages before the retransmission are discarded. This guarantees a correct order of messages for each virtual channel. Because of the use of different virtual channels, in case of a retransmission, all other channels stay unaffected. Thus, for data and control streams it is guaranteed that they are always valid and free of errors when received through the CBM interface. When a message is passed through the interface, as soon as the start flag is asserted and there is no stop, it is guaranteed that the complete data message is accepted. Thus, it must be delivered in a continuous stream until it ends. This must also be guaranteed by the receive part of the user logic module. Internally there must be enough time for one additional clock cycle before a packet to insert a SOP and two additional clock cycles after the end of a packet to give the CBM protocol module a chance to insert a CRC and an EOP. Therefore, after each packet there are three stop cycles, which are appropriately flagged to a user send logic. The user logic module must take care that data packets and control packets have at least 8 bytes and at most a 64 byte payload.

The signaling for data and control transfers at the receive path works analogously to the send path. The valid-stop synchronization at the receive path uses *data2rec_stop*, *ctrl2rec_stop*, *data2rec_start*, *data2rec_end*, *ctrl2rec_start*, and *ctrl2rec_end* to realize flow control. Because only complete packets can be streamed through the network, the *data2rec_stop* signal or the *ctrl2rec_stop* signal affect only following packets. Thus, it must

be assured that enough puffer space is left in a user logic module for receiving the rest of an already started packet, when a stop is performed. The next packet is always stalled in the last puffer stage respectively the complete stream is stalled within the CBM Network.

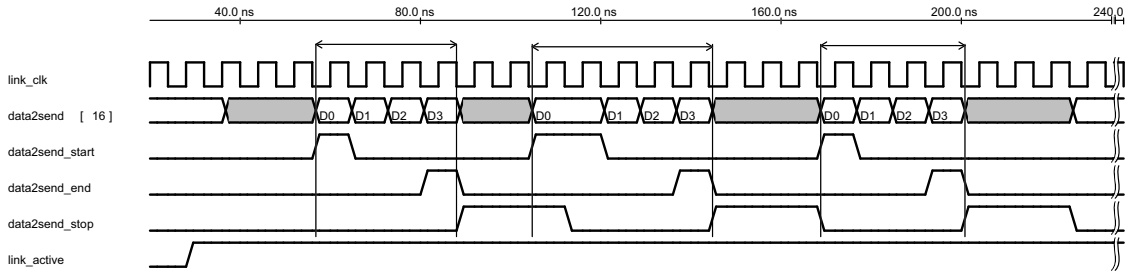


Figure 3-12: Interface Example for Send

A timing diagram of a data throughput example for a send stream at the interface is given in figure 3-12. It shows the hand over of several data packets with the length of 64 bits to the CBM Protocol Module. This example is simplified, because possible routing extensions are ignored and due to identical interface signaling, the usage of larger packets is not presented. All invalid data parts are marked gray in the *data2send* stream. When the *link_ative* signal is set and the *data2send_stop* signal is inactive, a packet can be sent. To send a packet, the valid signalization *data2send_start* has to be set to active. Then data can be sent over the *data2send* bus. Each clock cycle 16 bit of new data are accepted. During the last 16 bit of a packet, its end is indicated by setting the *data2send_end* signal one clock cycle to active. The CBM protocol module then sets the *data2send_stop* signal to active for receiving time slots to insert all required special characters for proper network communication.

The second packet of the example shows what happens, if the next packet is ready to be sent before the CBM protocol module is able to accept it. The *data2send_stop* signal is asserted and the next packet is available, indicated by the *data2send_start* in active state and the *D0* 16 bit value visible at the *data2send* bus. Then, the output of a user logic module remains in this state. As soon as the *data2send_stop* is released, data can be delivered and at the next

clock cycle, packet processing starts and continues until a *data2send_end* appears. With this method, it is possible that after a packet is completed, the next packet is already allowed to be prepared to send without being bound to any restrictions.

An example for a data throughput at the receive path is presented in Figure 3-13. It shows three received packets of minimum length. The start of a packet is indicated by a *data_rec_start* pulse and the end of it by a *data_rec_end* pulse. Each packet produces one NOP before and two NOPs after its delivery. This overhead is size independent and for the maximum packet size of 64 byte, it results in less than nine percent overhead. This amount of overhead is acceptable. During the second arriving packet within the example, a stop is flagged. It is depicted that this flag has no effect on the currently processed packet. Only the subsequent coming packets may be affected, if they occur.

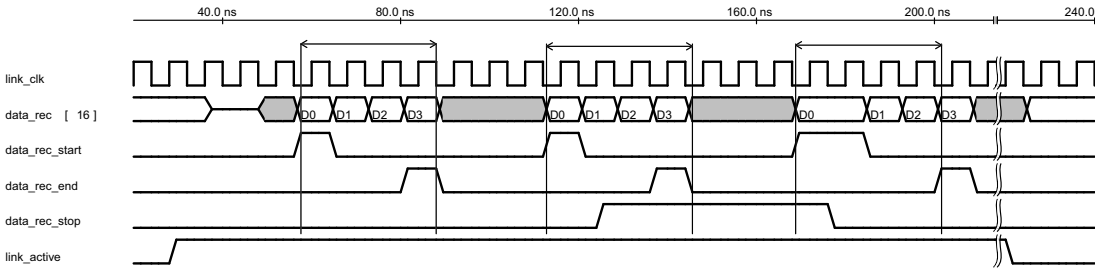


Figure 3-13: Interface Example for Receive

In general, there are only complete packets streamed through the interface and the network. It is not possible to interrupt a packet with a stop. If a stop is set as active during packet streaming, it only affects the next packet when it is still active. Therefore, it is necessary that all involved modules have at least enough buffer space left to accept a complete packet, which could be a maximum size of 64 bytes with additional routing. The complete error handling of the link is done by the CBM protocol module and is transparent to the user logic module. Erroneous data or control messages are automatically retransmitted. CRCs are automatically attached, detached, and checked. As long as there is no valid data available, IDLE characters are inserted. IDLE characters and all network flow control or management

characters are not visible at the interface. Therefore, a user logic module does not have to care about the lowest layers of communication. Its only concern should be to deliver enough data to utilize the bandwidth.

3.6 CBMnet Synchronization

3.6.1 Requirements

Following the main concept of the CBMnet protocol, all kinds of communication have to be transported over unified links. Thus, a synchronization mechanism working through the entire network must be derived by them. As a basis for providing synchronization, the deterministic behavior of the network must be guaranteed. It must be precise enough for experiment requirements. This leads to the need of having bit cycle accuracy. It does not work directly with different clock oscillators, because over time even if they were only slightly different they would drift apart. One way of solving this problem is to recalibrate occasionally. Another way is that all clocks used within the network must be derived from a master clock. The second variant is implemented. Therefore, the clock is propagated over the links by recovering it out of the data stream. After clock recovery, jitter can be too high at least for some parts of the system, especially for data acquisition or in particular for its usage as reference clock for sending. Then jitter cleaning is necessary. This was already successfully done in FPGAs for enabling MGTs/GTPs usage during beam time readout with the CBMnet. A reliable deterministic behavior then delivers the platform for synchronizing the CBM network using the special DLM messages.

3.6.2 Deterministic Latency Messages

DLMs are sent through the CBMnet from the synchronization source, the ECS, down to the leaves, the detector front-end. They appear exactly at the right clock cycle in the front-ends, for example to reset readout counter. This guarantees that the correlation of measurement data taken by front-end detectors are kept. However, it requires special working principles

to ensure its reliability. Special deterministic hardware had to be built or configured and its most important ability, the priority request insertion, had to be implemented. This feature makes sure that DLM characters are always inserted with highest priority onto a link. Not only for standard arbitration, but also for direct insertion into data streams at any time, no matter if there is a control or a data message currently in process. Implementing this feature, of course, requires extra hardware and can result in a huge complexity. Therefore, it is restricted to a maximum insertion of one DLM per packet. This value is clearly enough, because the frequency of DLM usage for all algorithms is not very high. To ensure proper behavior, only every 35 clock cycles a DLM can be inserted. Especially for FPGAs, deterministic configuration mechanisms are necessary to assure that after a new initialization, the deterministic latency is still provided.

In table 3-3, all DLMs with their functionality are listed. The DLM0 is used for the initialization procedure. This DLM must be directly reflected with a known clock cycle value for the reflection within devices. This enables the length measurement of all cables in the network and delivers a basis for start-up calibrations. The DLM0 might be used during service times within running experiments, however it is usually unused during actual data acquisition. After link calibration, the DLM2 is used for counter reset within the front-end by setting a system rest value for the most significant global epoch counter part. Then, the arrival of the first DLM1 typically resets the lower part of the counter. Normally, a global epoch counter defines a time for collecting detector data in the order of microseconds. This global counter has a most significant part, the epoch counter and a least significant part, the time-stamp. After the system is once setup and the epochs are synchronized due to its deterministic behavior, it should not lose its proper correlation. However, with every new epoch a DLM1 may arrive at the front-end for checking that it is still in sync. The granularity for sending DLM1 markers may differ among different systems and they might be skipped when another DLM is sent due to minimum sending distance of DLMs, but they should always arrive on wrap around of the counter. This event can cause a time-stamp reset when the counter is out of sync. In this case, usually an error is flagged in a register file or it is directly communicated to a control node. The control node, which is checking the status of

DLM Number	1.Char	2.Char	Comment
DLM0	K 27.7	D 10.3	Initialization and Measurement
DLM1	K 27.7	D 14.1	Periodic time counter reset
DLM2	K 27.7	D 20.1	Set new counter value
DLM3	K 27.7	D 20.6	Reserved
DLM4	K 27.7	D 22.3	Reserved
DLM5	K 27.7	D 28.2	Reserved
DLM6	K 27.7	D 28.5	Reserved
DLM7	K 27.7	D 6.2	Reserved
DLM8	K 27.7	D 14.6	Start DAQ
DLM9	K 27.7	D 3.1	Stop DAQ
DLM10	K 27.7	D 11.2	Command List 0
DLM11	K 27.7	D 17.2	Command List 1
DLM12	K 27.7	D 25.3	Command List 2
DLM13	K 27.7	D 17.5	Command List 3
DLM14	K 27.7	D 3.6	Command List 4
DLM15	K 27.7	D 5.3	Command List 5

Table 3-3:DLM Coding and Functionality

front-end devices periodically, will then be responsible for further actions. The radiation environment can of course lead to temporary failure of front-end chips or devices. In that case, a device has to be restarted and resynchronized. Otherwise, more and more devices could fail during a beam time and measured data would lose its value. Therefore, after link reinitialization and a recalibration of the restarted device, an additional mechanism is required. Its first step after a link reinitialization is to send a control message to the front-end delivering the value of an epoch, in which the device should resynchronize, to a local device register file. After this value has been stored, the DLM2 is sent to the device. It sets the next epoch into the device epoch counter. Simultaneously, all error flags and counters should be reset and if necessary, FIFOs should be cleared. This guarantees that the control

node can read all status information until a DLM2 is sent and afterwards the status registers may capture future error conditions. Then the next coming DLM1 validates this new epoch value as valid. The following data is then again in the right epoch and data collection can continue. Future DLM1 markers which arrive occasionally, check on the synchronicity. A mechanism to support data acquisition uses of a DLM8 as a *Start Data* taking command and a DLM9 as a *Stop Data* command. This delivers the ability to start and stop the complete system at a defined time. A variant for this feature is using specific group masks to enables starting or stopping of data streams for defined groups within the CBM network. The DLM3 - DLM7 are reserved for future use. The DLM10 - DLM15 are user defined DLMs. They enable usage of specific user command lists to be started at front-ends. A command list can be used to run a user defined sequence of commands to deliver features like initialization, calibration, or to run service routines. Command lists can not only be hardcoded, but also be filled with commands by sending DCMs to front-ends placing new commands within a list. Thereby, this gives the user a powerful instrument to run tasks at specially defined deterministic times.

3.6.3 DLM Synchronization Sequence Example

A complete initialization and synchronization sequence is presented in figure 3-14. It shows different initialization and calibration phases, which must be successfully passed before standard communication traffic can occur.

At the start of link initialization, a first phase low-level initialization is performed. It is implemented in the SERDES parts of the PHY and assures that barrel shifter positions are correct, received words are aligned, and communication is stable. Therefore, SERDES modules use two different ready characters READY0 and READY1. While sending and receiving READY0, barrel shifter adjustment and word alignment is done. Because of deterministic latency, the barrel shifter position within all devices must be well controlled. After successful alignment, each device sends READY1 char to signal its readiness. In case both sides receive READY1 within a defined time, the low-level initialization is finished. Otherwise, the sequence and calibration is restarted by sending READY0.

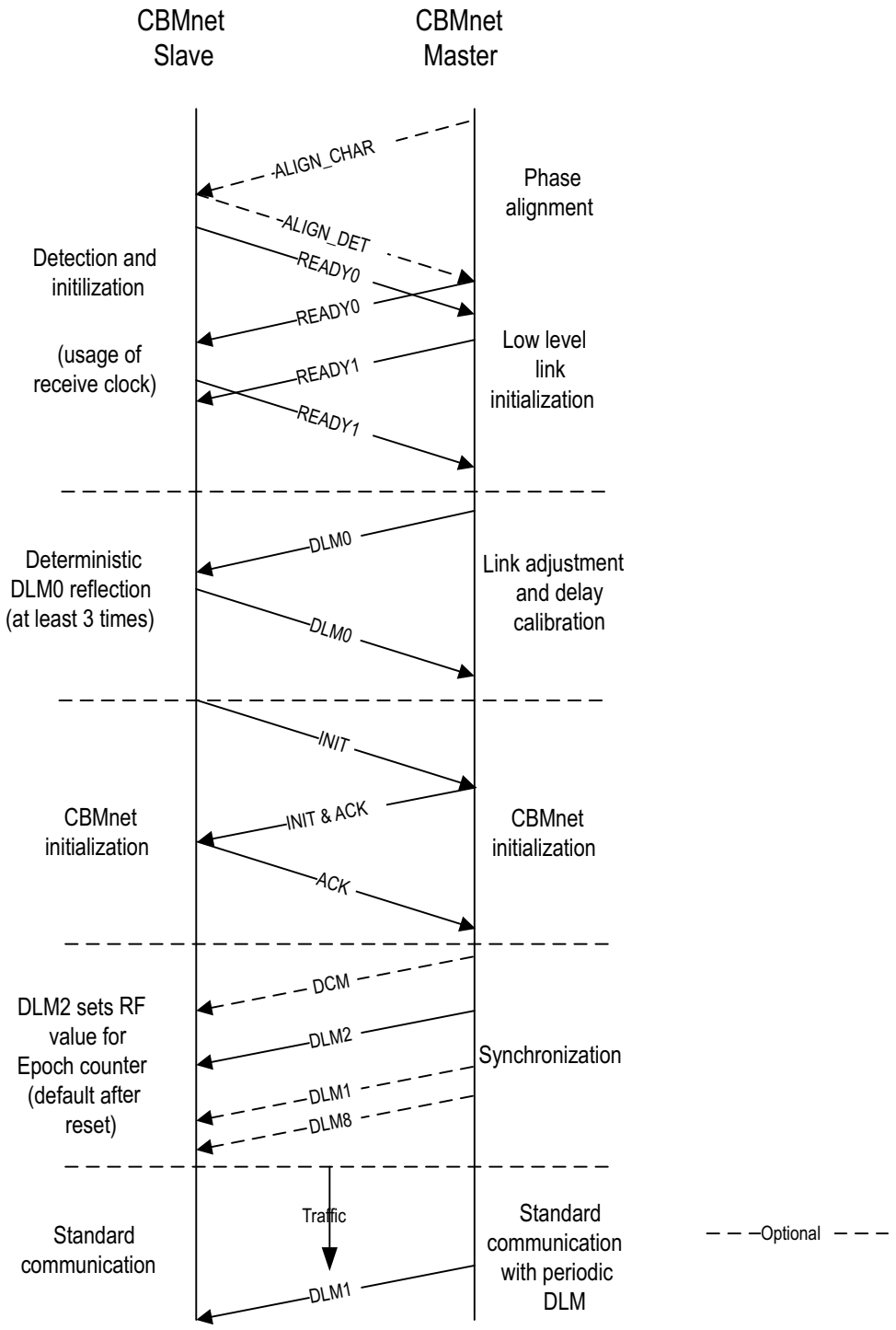


Figure 3-14: DLM Initial Synchronization and Resynchronization Sequence

In addition, an optional alignment mechanism for proper phase alignment within FEEs is provided. This mechanism is required for front-end device connections not running CDR. They are using data streams, which are sampled with the synchronous system clock, but not a recovered one from its data. Thus, received communication streams must not be phase aligned. The phase alignment is calibrated by sending an ALIGN_CHAR to the front-end and receiving an ALIGN_DET detection signaling. It flags if the char was successfully received or not. The algorithm runs through different delay tabs and remembers transitions while detecting a stream. Then it configures the alignment to use the delay tab, which is in the middle of a correct detection area. This guarantees a perfect detection within the signal eye.

The next initialization phase is responsible for measuring link length and defining distances of attached devices. Therefore, DLM0 messages are sent and become directly reflected by the devices, which have to be synchronized. This is repeated three times and the correct measured value is stored within the register file. In case of an error, the link is restarted. The synchronization unit analyzes all front-end distances and calculates an ideal delay values for each device. Then, the appropriate delays are set within all transmit modules. However, after running this initialization sequence, it is assured that all devices have their word clock and thereby, the DLM arriving within an identical link clock cycle. Finally, the PHY layer initialization and calibration is done. CBMnet initialization, which runs a standard handshake protocol, is performed and then the link is ready to be used and at the CBMnet interface, the *link_active* signal is asserted.

Now actual synchronization can take effect. A DLM2 is sent to the FEE devices and they set their epoch counters due to the current given value in the register file. Its default value after an initialization is zero. In case of a reinitialization, a DCM is sent to a device before the DLM2 sets a valid resynchronization epoch counter value for resynchronization. Then, the periodic send DLM1, usually from an ECS, arrives and validates the new epoch. From time to time, there are additional DLM1 messages sent within the communication streams to check front-end synchronization. However, DLM8 and DLM9 can be used to start and

stop data acquisition from the synchronized FEEs. For accessing only parts of the FEE devices, the masking mechanism is suggested. Here a masking bit for each FEE device is given in the register file. Via DCMs, the mask is set and only chosen FEEs are addressed by an appropriate DLM. In the CBM network, the occurrence of errors is flagged in the FEE register files and a DCS or ECS can decide whether to restart or recalibrate devices.

3.7 Jitter Cleaner Device

The system wide clock distribution over the links using CDR requires that recovered clocks have an appropriate quality. This is required to assure that no bit slips appear in the communication chain, which would breach the deterministic latency. Additionally, the recovered clock is used as reference clock for sending. FPGA transceivers have minimum jitter requirements of less than 40 ps RMS jitter for their reference clocks. The recovered clock jitter was between 80 - 100 ps RMS, so these requirements were not fulfilled for the recovered clocks received in the used FPGAs. Thus, a jitter cleaner device had to be used, preferably with COTS parts, which cleans the clock and feeds it back into the FPGAs as a usable reference clock for their transceivers.

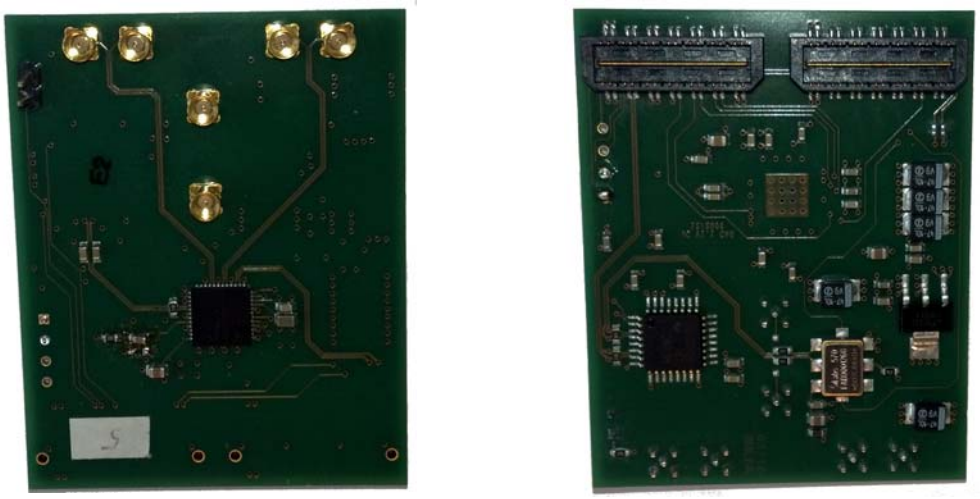


Figure 3-15: Jitter Cleaner Device (top side left, bottom side right)

Figure 3-15 shows the developed jitter cleaner board in its second version. It uses a LMK3000 Family [56] device for cleaning a recovered clock and can be attached to a DCB via a mezzanine connector. MMCX connectors are used for clock outputs. They are connected to a MMCX input on the DCB supplying a reference clock to its MGTs. Performance measurements with the first jitter cleaner board version are presented in figure 3-16. It

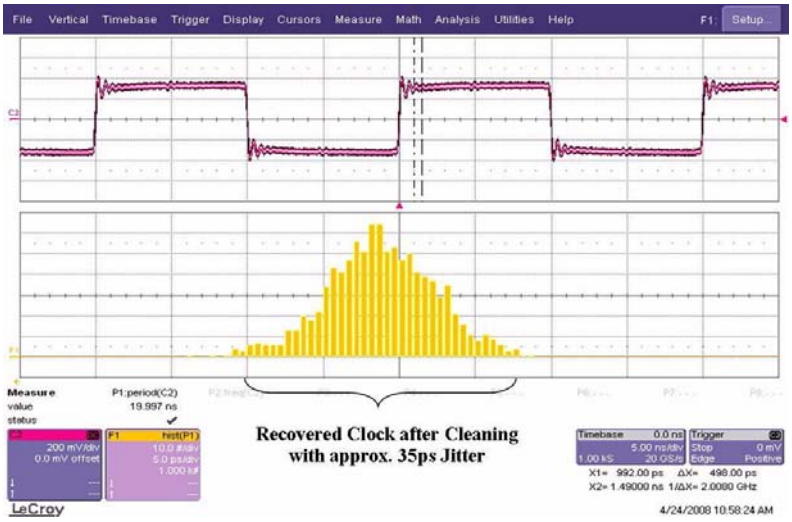


Figure 3-16: Jitter Histogram of Cleaned Clock

shows that using this device reaches a peak-to-peak jitter below 40 ps, which is required for MGTs. A RMS jitter at least below 10 ps can be reached. This enables reliable communication. Additionally, the jitter cleaner board is assembled with an oscillator, so it can be used as a clock source. A platform useful for clock testing, with several MMCX configured as clock inputs and clock output and additionally direct onboard jitter cleaner integration, was built [57]. This development was based on a student thesis [58] done in the context of a research project with Altera [59]. Prototyping and measurements using this platform helped in the understanding of possible solutions for clock distribution, onboard jitter cleaning, and separate synchronization distribution. Furthermore, designing with an Altera FPGA was beneficial, because comparing these experiences with the Xilinx [60] FPGAs, it became clear that Altera devices could be an adequate substitution. Due to usage of Xilinx devices

in various parts of CBM, it does not make sense to use a different FPGA. Nevertheless, the experiences gained helped in supporting the design of the ROC3 FPGA as future FEE readout controller, with onboard usage of LMK3200 Family device.

3.8 CBMnet Routing

A routing scheme for the CBM network has requirements like minimality, efficiency, and flexibility. Minimality is necessary for integration into numerous devices in the system, especially within large crossbars. Data utilization is important for achieving high data rates. Thus, efficient address coding, switching structure, and aggregation methods have to be considered while developing a CBMnet routing scheme. The various different detectors have their own readout structures and partitions. Therefore, flexibility is required to support them. All different parts within the protocol definition are separated and modular implemented. The processing width is 16 bits. Thus, an address can be either 16 bits or a multiple of 16 bits. However, Table 3-4 compares these two approaches. Analyzing the utilization of messages shows that for minimum message utilization for data is around 6 % better using the 16 bit addressing scheme and for control about 8 % it is even about better. Regarding maximum messages, difference is around 2.5 % for data and circa 4.5 % for control. These numbers considered together with the facts of faster routing calculations and smaller hardware seems to make 16 bit addressing superior to 32 bit addressing for the CBM network. The handicap of partitioning into different detectors or setups with more than 32K devices is of little significance, due to existing differences between detector types and the lack of clarity for huge systems.

Thus, CBMnet uses 16 bit routing addresses. Routing addresses are inserted before each payload. In case of data messages, only the FEE identifier is inserted to make sure that a data source is known and available at the beginning of each payload for fast processing. There is a fixed data sink for each data source and the FEE identifier delivers enough information for a source path routing. Later, this FEE identifier can additionally be used to sort data packets before it is removed. All control messages have two addresses, first a destina-

	PRO	Contra
16 bit address	• address processing in one word clock cycle after arrival • utilization of packets • efficient packet switching • small hardware	• partitioning into different detectors • needs partioning within a detector at the root node for more than 32K detector ASICs • restricted address space
32 bit address	• freehanded partioning for all device types • all detectors in the same address space	• larger table structures and hardware requirements • management of experiment address space • a lot of unused address space

Table 3-4: Small vs. Large Addressing

tion address followed by a source address. DTM and DCM addresses are transported in different virtual channels. Thus, they use a different routing interpreter, which simplifies their processing.

It is useful to partition the address space and use special routing fields to enable routing decisions based on parts to accelerate this process. Figure 3-17 presents the CBMnet address space. It is divided into two regions, the Endpoint device space and other network devices. Endpoint devices are partitioned into 1024 blocks of four device groups, containing up to eight endpoint devices. Other network devices are subdivided into intermediate network devices close to a detector, and processing and control devices in the service region.

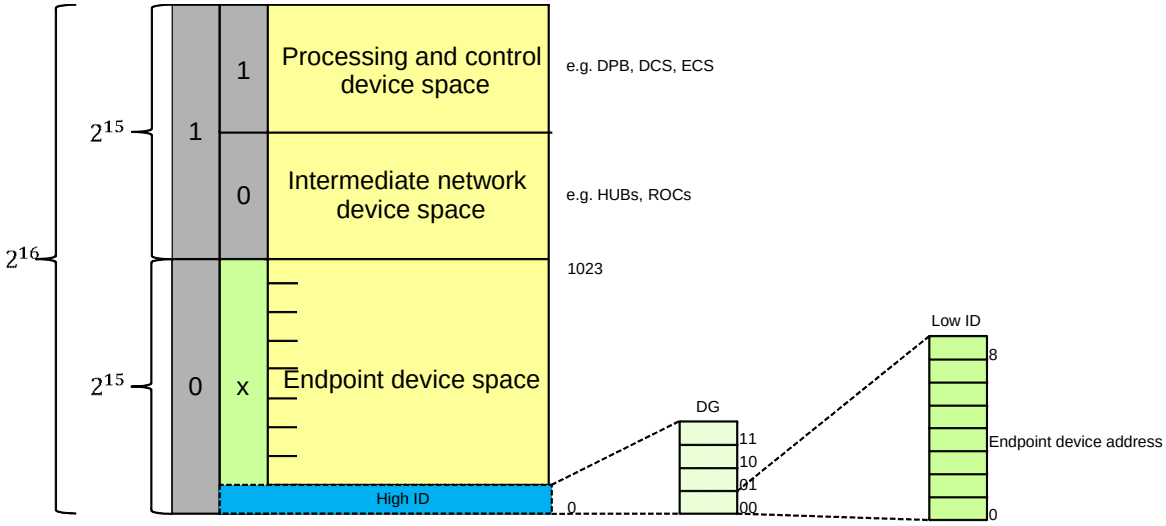


Figure 3-17: Address Space

Figure 3-18 shows the unique address of a detector endpoint like a readout ASIC or a ROC depending on the setup, called the Front-end Device Identifier. The Identifier has an address space ID in the most significant bit. A zero within this field defines that this is an endpoint address. The rest of the 15 address bits spans the FEE address space. Thus, up to $2^{15} = 32K$ endpoint can be reached. A high ID field is used for routing in upper hierarchies. The device group (DG) together with the low ID field is used to select attached endpoint devices. The partitioning may differ among detectors. Therefore, an intermediate device can contain either one, two, or four device groups, and DGs are used for selecting intermediate devices or endpoint devices.

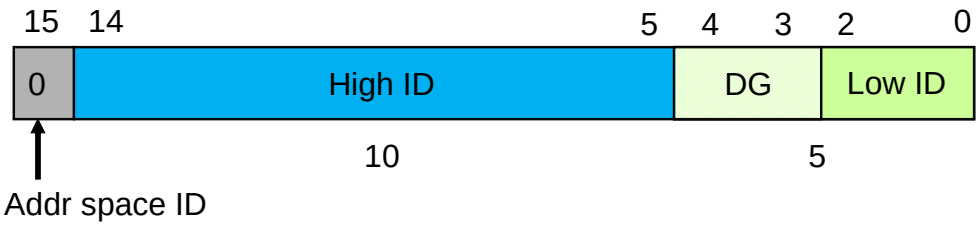


Figure 3-18: Front-end Device Identifier

The scheme of routing towards the endpoints is presented in figure 3-19. In some cases, checking the address space ID is enough to find a routing decision, which eases processing. If this is not the case, routing in intermediate hierarchies is done through using the high ID address field for a setup with four virtual device groups aggregated in an intermediate device. When less DGs are integrated into an intermediate device, the appropriate DG field information has to be considered for routing. The final routing destination decision is then done by checking the low ID field and so far unused DG field information.

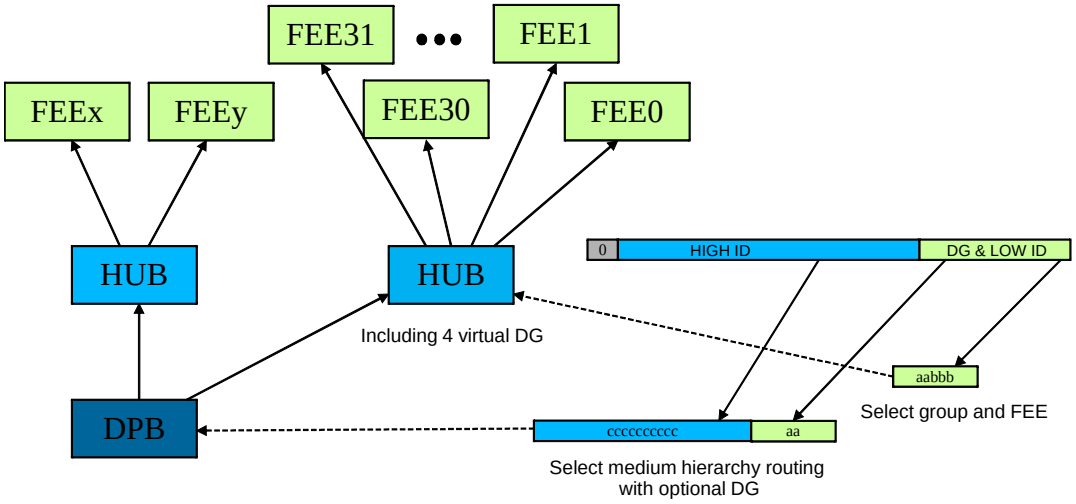


Figure 3-19: Routing Scheme

Figure 3-20 shows the Node Addressing Identifier necessary to address all non-FEE devices. A one in the address space ID field flags that a message is routed to non front-end devices. It is possible to address a maximum of 2^{15} = 32K inner nodes. The intermediate device ID marks if it is a detector area device or not. The devices directly attached to the FEEs, for example HUB ASICs, are flagged with a zero in this field. Then, they are addressed by using the network ID field, which is identical to the high ID part of its attached endpoint devices plus the DG field. All other network nodes as ECS or DPBs are reached

using a one as an intermediate device ID and may use a 14 bit address space. Routing in the inner network part might use interval routing due to small number of connections and the large size of a table based routing.

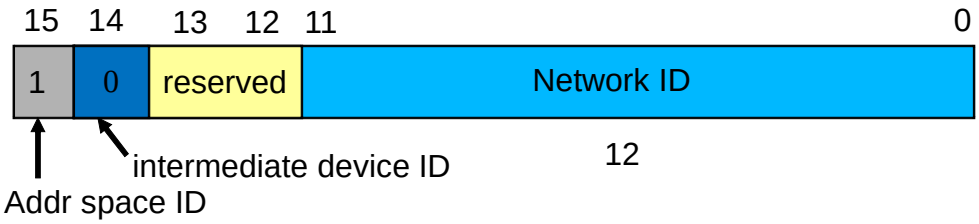


Figure 3-20: Node Addressing Identifier

3.9 Measurements

Directly within the first test environments [61] using prototype boards and a first version of the protocol, proof of concept for the CBMnet design and readout chain was done. Therefore, reduced test setups were used connecting an ABB to two ROCs or connecting a single chain ABB - DCB - ROC. The first tests were performed using test pattern generators creating random sized data and control packets with periodically included DLMs. This had the advantage of generating enough data to fill up the links and perform testing under stressful test conditions. These tests have been successful. Nevertheless, testing with a real detector readout chain has an additional quality. Thus, further testing was done together with other development groups [62] and integrated ROC and ABB user logic. In this setup, readout and control functions were successfully tested. The optical links were configured to use 2.5 Gb/s, which is sufficient for test beam readout systems. Some evaluations have been done using 5 Gb/s or 6.24 Gb/s and as soon as it becomes necessary, link bandwidth can be adapted. The theoretical peak bandwidth utilization of the CBMnet V2 protocol is about 228.56 MB/s (73.142 % of 312.5 MB/s) of the maximum bandwidth of 250 MB/s with 8b/10b coding for a 2.5Gb/s link. The test setup was using CBMnet V1 with slightly lower theoretical peak bandwidth and the DMA engine implemented in the ABB to transfer data from the event buffer to the host node. Measurements for data bandwidth utilization

resulted in about 224.55 MB/s. In addition to bandwidth testing, the system synchronization [43] was tested. The deterministic latency of links has been proven in several long time tests of up to 168 hours without errors. At random times, a peak-to-peak clock jitter measurement resulted in jitter below 40 ps. MGTs being sensitive to jitter did not lose their link locks during these tests. The CBMnet approach guarantees synchronization accuracy between two ROCs of one-bit clock cycle. This results for the used link speed of 2.5 GHz in at least 400ps accuracy. This means that time difference between two different ROCs, considering jitter between clock source and ROC, is never greater. However, there has been no loss of synchronization seen during the tests. The latest jitter measurement test of RMS jitter presented in figure 3-21 shows a mean RMS jitter of around 4.77 ps. The noise in the figure having a roundabout 3µs periodically peaks can be ignored. This is due to the assumption is that this noise comes from the DC/DC converters. Furthermore, it does not affect the transmission quality and it has not been measured in a similar setup, which is only capable of lower clock rates.

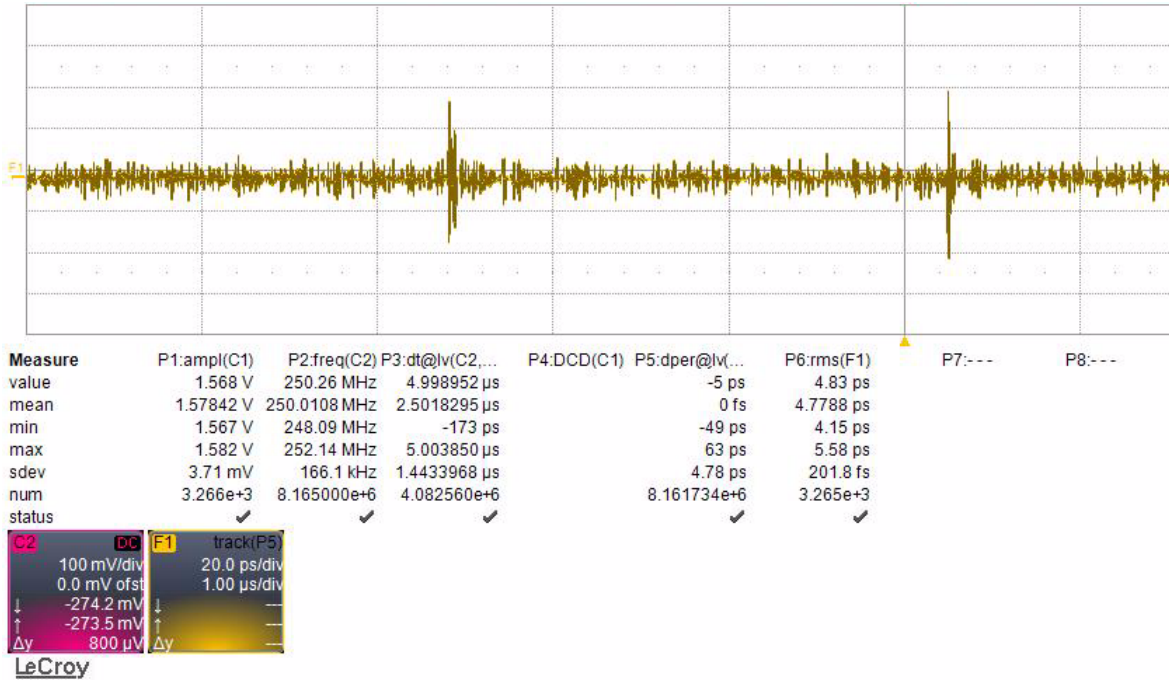


Figure 3-21: Jitter Measurement

Figure 3-22 shows a hierarchical setup successfully used during several beam times and it is used as prototyping platform [63]. It consists of up to eight ROCs, using blue fiber in the figure, collecting data from different detectors, attached in groups of 4 to 2 DCBs. In some setups, DCBs were used as DPB prototypes. The DCBs are connected to an ABB responsible for clock distribution and system synchronization using DLMs. Here, the ABB may serve as an emulation platform for an ECS. Additionally, each DCB used an ABB as a data sink. In this system, the DCBs aggregate the incoming data from ROCs and sends it to an ABB plugged into a workstation that runs DABC software for data collection. Directly the first beam setup using this hierarchical readout [64] ran problem-free. ECS emulation providing a control system using DCMs and clock distribution also worked reliably. All DTMs coming from four data-streams were correctly merged into a single stream within the DCB. No data transmission errors or data flow problems occurred. All epoch counters values were reset synchronous and synchronization using DLMs showed no problems.

CBMnet synchronization achieved one-bit clock synchronization and the concept of using unified links providing DLMs for synchronization is proven as a working solution for CBM. The existing prototype systems are a good basis for future development towards the final readout chain.

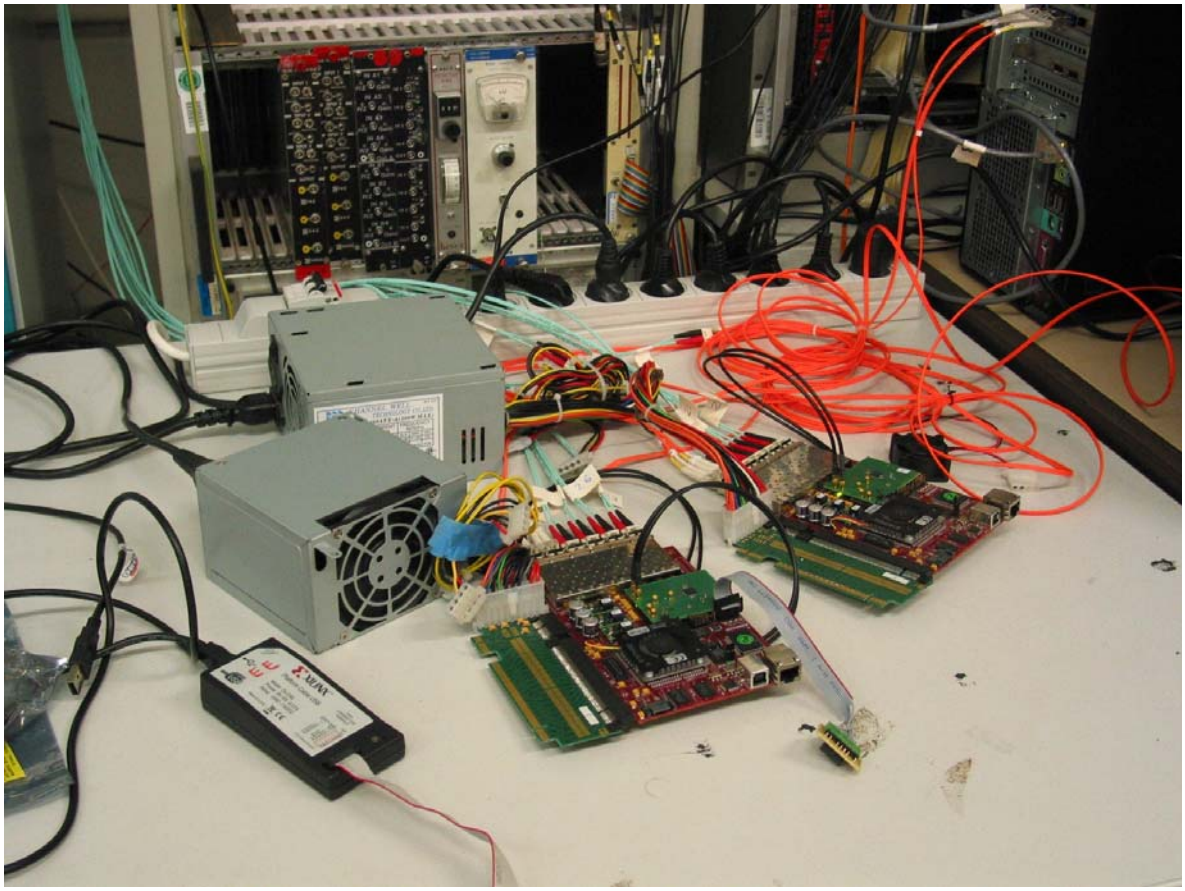


Figure 3-22: Beam Time Measurement Setup

3.10 Conclusion

Analysis of CBM requirements and prototyping setups led to the decision to development a new network protocol for the CBM DAQ. CBMnet combines three different traffic classes onto the same unified network. In addition, an integrated clock distribution using CDR supplemented the CBMnet concept. The CBMnet delivers all required network features and the unified link saves cost and space. It achieves one-bit clock cycle accuracy on link level for the synchronization of devices. The modular concept guarantees usability within different devices by enabling easy adaptions and updates. A summary of all CBMnet features is presented in the list below. CBMnet presents a working and scalable solution to readout self-triggered front-ends.

- Communication over one optical link supporting Data Transport Messages (DTM), Detector Control Messages (DCM) and Deterministic Latency Messages (DLM)
- Early stage data aggregation
- Optimized data utilization and efficient rate conversion
- Reusability assured by an easy to use interface and a modular structure
- User transparent link administration
- Fault tolerance using retransmission for DCMs and DTMs
- Different physical layer support
- Deterministic latency for bidirectional FPGA (Xilinx V4, V5, V6, S6) links
- Deterministic latency for unbalanced links
- System wide clock recovery and synchronization using DLMs
- Flexible high bandwidth front-end electronics attachment
- Network scalability

All FSMs within the design have been developed using an Electronic Design Automation (EDA) FSM designer tool that was previously developed [65]. This tool became an open source project [66] and has been successively supplemented with new features [67]. Due to the good experiences [68] using this EDA tool, it is and will be highly used to design all used FSMs.

Chapter 4

Generic Modules for CBM Device Development

An overview of the generic module concept for CBMnet devices is presented within this chapter. The goal is to supply users with generic code blocks that have efficient interfaces for easy integration. Therefore, it implements not only the CBMnet protocol, but also adds several functional blocks to create a generic built-in environment for users. It shows solutions for FEE detector devices and FPGA solutions. An advantage is gained by reusability of modules and usage of preexisting tools to improve the design. By using the generic built-in support blocks, it then becomes easier to integrate CBMnet into new designs, shortens design time, and reduces production risk.

4.1 Generic Modules

4.1.1 Concept

The CBMnet protocol will be used within FPGA and ASIC devices in the CBM network. The goal is to avoid protocol conversions in the readout chain, especially for the bandwidth intense readouts. Therefore, various users are supplied with CBMnet modules to integrate them into their designs. This integration is supported by the Computer Architecture Group (CAG) and was done for different FPGAs in the first CBM project phase. The devices have different tasks requiring special hardware or firmware, but there are always some module blocks, which are identical among devices. Thus, there is a potential to share module blocks. Furthermore, the concept of sharing modules avoids multiple implementations of identical functions. This not only saves design time and additional effort, but also avoids

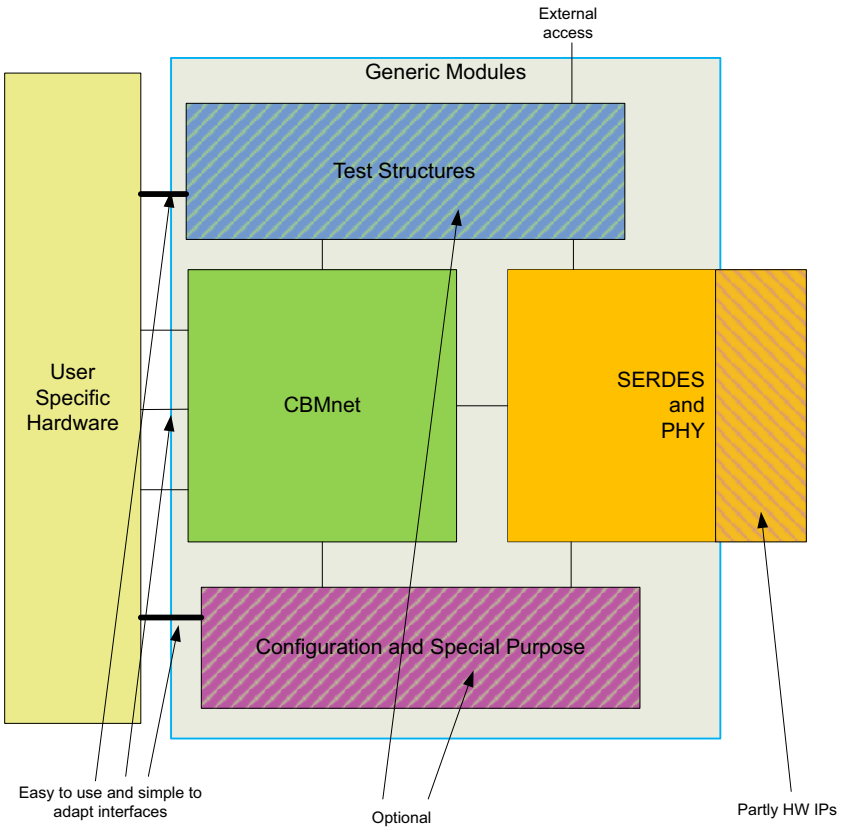


Figure 4-1: Generic Modules Diagram

having different implementations of similar functions. It helps software reusing code bases to control multiple kinds of readout hardware and delivers a homogeneous system. Thus, the generic module concept was created with the goal to supply all users with a set of generic modules, and not only the CBMnet. Figure 4-1 depicts the generic module concept delivering CBMnet and SERDES/PHY implementations together with optional parts such as test structures, configuration, and special purpose functions. Besides other advantages, the most valuable one was to reduce design time for generic module users

4.1.2 Generic Modules

The main modules within the generic module concept are the CBMnet V2 modules. These modules are responsible for all actions concerning the protocol as described in chapter 3, *Development of a Synchronous Network for CBM*. They deliver three virtual channels for communication and handle the link administration. Its link speed scales according to the reference frequency. These CBMnet V2 modules were extended for the generic module concept. Because of special usage for front-end devices, a master and slave concept was created. It supports unbalanced links. This means there is one bidirectional link and up to three additional unidirectional links. Thus, the data bandwidth for detector readout can be increased. This can, for example, be used with one up-stream link with 500MB/s for the current 180nm implementation and up to four down-stream links with 2Gb/s for endpoint devices. A CBMnet V2 master is completely compatible with a CBMnet V2 standard implementation, but it may support slave handling.

Besides CBMnet, there is one other essential module required for implementations using CBMnet. This module is the SERDES module. The most important ability this module has to guarantee is the deterministic latency over the link. This must be assured at any time, even after a power cycle or a link reinitialization. There are currently two different kinds of SERDES modules supported. One kind is based on FPGA SERDES modules [69]. Multi-Gigabit Transceivers (MGTs) as well as GTP transceivers were used for Xilinx FPGAs. Different control logic parts of MGTs and GTPs have to be configured to guarantee a reproducible, deterministic latency. The basic configuration is derived from the MGT user guide

[70]. The fabric interface had to be chosen with a width of 32 bits to guarantee the deterministic behavior at the interface. This is required because internal MGT width is 32 bits and multiplexing must be avoided. In addition, the wrapper generated by the RocketIO Wizard had to be modified to have access to the barrel shifter position signal RXLOSSOFSYNC of the alignment block to read out deserialized data. A derived parallel clock generated by clock data recovery (CDR) can lead to different barrel shifter positions. To obtain deterministic latency, the CDR can be reset for altering barrel shifter position of alignment until the predefined value is reached. Right alignment will always be achieved after a number of resets. Thus, this configuration method with a reproducible and deterministic latency is achieved after link initialization, while the same bit file is used for all FPGAs.

The second kind of SERDES is the standard cell based SERDES. The first implementation of this SERDES was done in the context of a research cooperation with the RWTH Aachen [71], combining innovative analog and digital design blocks. Different hardware versions have been successfully produced in a 65nm ASIC [72] and can therefore be labeled as ASIC proven. This SERDES includes input and output delay adjustment and an automatic link initialization. A second implementation has been done for the CBM SPADIC ASIC described in Section 4.3. Here a design without delay cells was chosen. Thus, link delay adjustment is completely done by the attached higher-level device. This SERDES is tested and can be labeled as ASIC proven. However, there are currently two standard cell based SERDES implementations available, one in a 65nm ASIC TSMC process and the other one in an 180nm UMC.

Special resets are required for the SERDES and a word clock must be provided. Thus, within the generic modules block, there are modules integrated which create reset signals and generate the word clock out of the received bit clock and system reset. These modules were extended to provide the complete set of clocks and resets to supply all ASIC blocks. Furthermore, a synchronization pulse generator was added which provide these pulses for the analog detector readout logic.

A module kind required in multiple designs is a register file (RF). It contains a general part and a user specific part. The general part is required in all devices. It includes the device address and other general data. The user specific part contains configuration, status, and error registers for a device. In addition, it must provide an external interface to attach either sub register files or special building blocks within the analog design. As a special building block, an analog shift chain support module is provided. However, similar RFs are used in different devices. The idea was to generate them with a tool to reuse already designed parts and take advantage of similar structures. An EDA tool to generate RFs was developed in a PhD thesis at CAG [73]. It uses human readable XML code for fast defining registers and structures. The XML is used for automatic generation of RFs. It uses a standardized access interface and supports sub register files. Thus, this tool fulfills all the requirements and is used to support the generic module concept.

A control decode unit was implemented to access the RF through the CBMnet protocol. In addition, an I2C slave development was completed. This I2C slave module can be attached to the RF. This widely used standard is assumed useful for most of the generic module users. It is ideal for first tests and bring up. When the design is running, it will stay in standby and can still be used for testing or service access.

The modules provided until now as built-in blocks are listed below.

- CBMnet modules with Master/Slave support
- Deterministic MGT/GTP implementation
- ASIC proven standard cell based deterministic SERDES in 65nm and 180nm
- Clock, reset, and synchronization pulse generators
- Register File
- Shift register chain support for analog designed ASIC parts
- CTRL decode for CBMnet ctrl messages interacting with RF
- I2C support for RF access

This appears to be a good starting set of generic modules and will enhance the CBM network device development. The modules can easily be adapted and integrated into new devices. The set will be further extended and multiple integration processes will be supported by the CAG.

4.2 FPGA modules

4.2.1 Common FPGA devices

Currently used common FPGA devices utilize MGT/GTP deterministic latency implementation attached to SFPs for fiber-based communication. The MGT/GTPs are connected to CBMnet modules which unpack the messages and transfer them through the CBMnet interface to user logic blocks. Some implementations use multiple MGTs in parallel supported by multiplexing and transformation structures. Current firmware versions using the CBMnet V2 and the special MGT/GTPs exhibit reliable and efficient communication abilities.

4.2.2 Data Combiner Board

The DCB is a special implementation, because it differs from other FPGA implementations concerning CBMnet message handling. It is a passive device that routes messages to the right receivers. Therefore, messages are not processed and unpacked by the complete CBMnet module block. They stay in the message format and are streamed through the DCB. In figure 4-2, the DCB structure is presented. It shows the five-port crossbar design. There is an additional six-port crossbar design implemented that supports a separate back-end connection for control and synchronization to enable an ECS. The figure shows the DCB input on the left where the incoming messages are transferred from the SERDES/MGT modules to CBM_lp_in modules. The CBM input modules have their own separated fifo structures. Directly attached crossbars pull the messages out if the routing channel is free. There are separated crossbars used for data and control to avoid deadlock situations. Due to link credit handling, output CBM_lp_out modules are used together with replacer units. These units

replace framing characters, which are assigned to credit numbers, with the next valid following framing character. DLMs are received from the back-end at CBM_lp_in_I4 and are forwarded to the FEE out ports with deterministic latency. This is a special implementation, but it is useful as a basis for future development of switching devices.

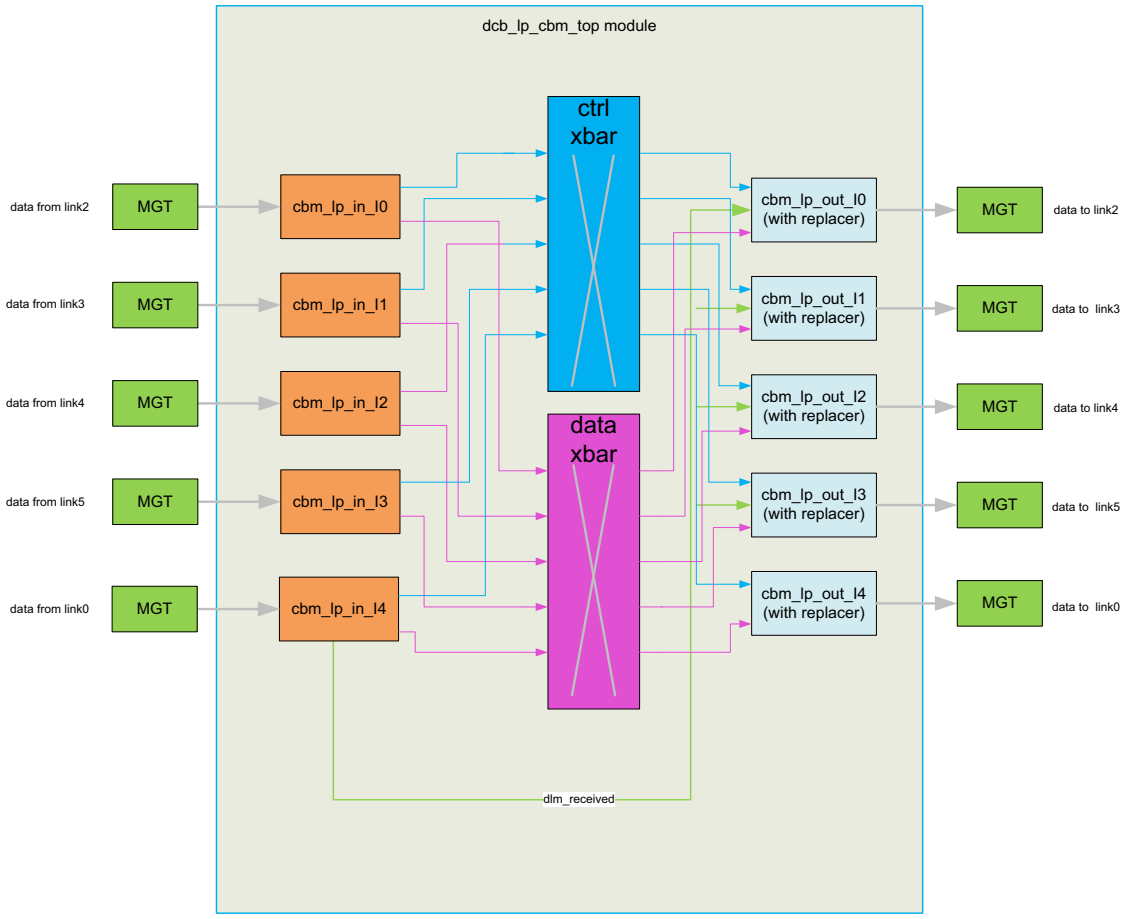


Figure 4-2: DCB Modules Block Diagram

4.3 SPADIC as Prototype

4.3.1 SPADIC Overview

Within the scope of the CBM, a research group of the Chair of Circuit Design at the University of Heidelberg developed a self-triggered amplification and digitization chip [74]. It was a candidate for different detectors to be used as readout ASICs. Tests worked fine and it was chosen as the TRD readout ASIC. Then the Self-triggered Pulse Amplification and Digitization asIC (SPADIC) project [75] was started. A first SPADIC prototype [76] was developed in 2010. This SPADIC Version 0.3 was successfully tested in 2011. After these successful tests, the next ASIC version SPADIC 1.0 [77] was designed and implemented together with the Computer Architecture Group (CAG) for direct integration into the CBMnet readout chain. This was necessary because of bandwidth requirements and the advantages of a protocol conversion free setup for the final experiment. Its integration and tape out is described in the following paragraphs.

4.3.2 Integration

As modules integrated into the SPADIC, a complete generic module block including as main parts the RF, CBMnet, SERDES, and I2C were used. A structure diagram of the SPADIC is presented in figure 4-3.

The RF was designed using the RF generator tool. It includes configuration registers, status registers, control registers, error registers, and a sub register interface. This sub register interface is used to access a shift chain of the analog SPADIC part. Therefore, a specially implemented module translates RF queries to the shift chain interface. There are two ways to access the RF from outside of the ASIC. The first one is an I2C module working as a I2C slave device to access the RF for bring up and first tests. The second access path is through the CBMnet protocol. Here, a control decode unit interprets control messages and operates the RF. In addition, it is responsible for generating control message answers. A simple multiplexer is sufficient to connect them to the RF.

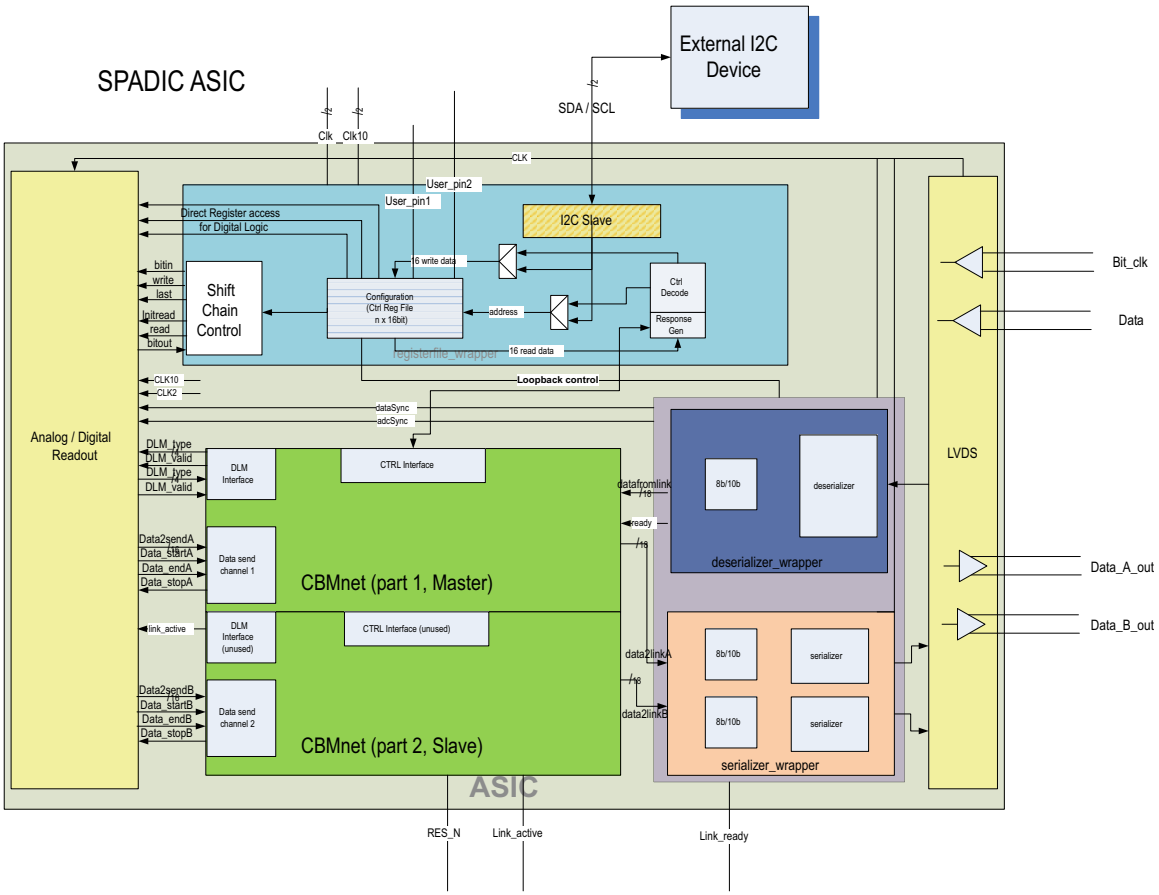


Figure 4-3: SPADIC Structure Diagram

The LVDS links connected to the CBM network are analog designed blocks that are directly attached to standard cell based SERDES modules with synchronization and link adjustment features. These SERDES modules and low-level initialization are working in conjunction with connected devices, e.g. a HUB ASIC, and assure physical communication, character alignment, and clock alignment. This guarantees deterministic latency through the link. In addition, within the SERDES modules, there is a clock generator and a synchronization pulse generator implemented. The clock generator derives a word clock with 25 MHz and a 125MHz reference clock for the SPADIC analog blocks from an incoming 250 MHz system clock. These clocks have a special relationship and start up sequence. The pulse generator is

required to generate various resets for the clock domains and special data synchronization (dataSync), and analog digital converter synchronization (adcSync) pulses required for proper reset and synchronization for the digital channels.

The CBMnet V2 module, which connects the SERDES with the RF and the analog SPADIC part, is the main logic block within the generic modules block. It handles the CBMnet protocol. After protocol initialization, it provides two data channels for sending captured data, a DLM interface for synchronization, and the control interface as the RF access path. A specialty of this block is the enabling of unbalanced communication. Therefore, a master slave concept has been developed. In this concept, the master is responsible for the link administration of slave channels. This means it decodes incoming ACKs and NACKs for a slave channel and flags them to the slave module. Thus, the slave module requires less modules and the number of incoming lanes can be reduced. This helps to overcome the space problems.

4.3.3 Tape-out

The SPADIC version 1.0 [77], the first detector ASIC in 180nm which implements the CBMnet with a 2x interface has been produced. On the left picture in figure 4-4, the floorplan of the SPADIC V1.0 before the tape out is shown and the right picture shows a received ASIC after this first tape out. This device will be used for the testing of integrated generic modules, proving synchronization, testing detectors in beam times, analyzing analog logic controlling detectors, and proving control mechanisms. However, the SPADIC device has been tested in conjunction with a Virtex 5 FPGA board which shows correct link initialization and working operations like RF access. There are newly developed readout boards, FEBs and adapter boards available to attach the SPADIC to existing Spartan 6

FPGA boards. This integrates the SPADIC into the CBMnet based, fast, optical beam time readout chains. The first results seem very promising and within the beam times of the next few years, the SPADIC and integrated CBMnet modules will show their potential.

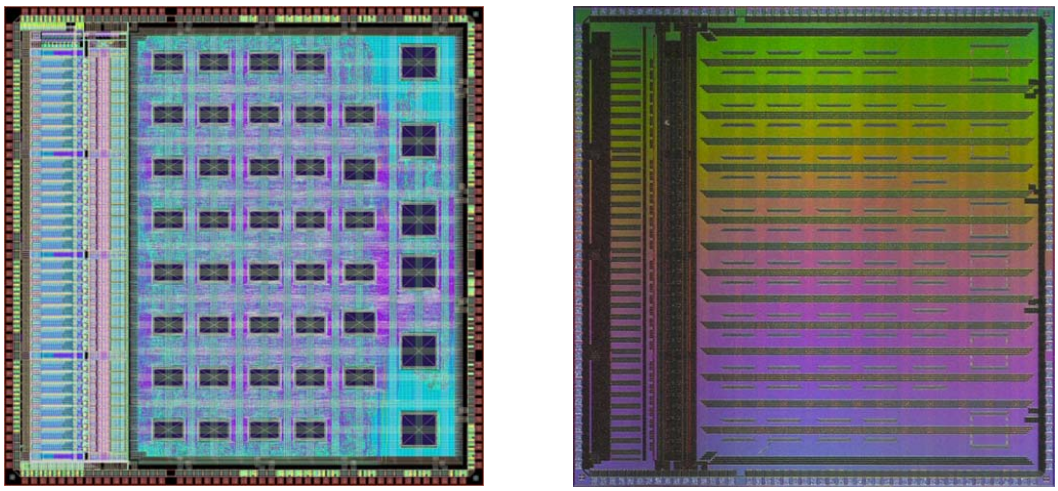


Figure 4-4: SPADIC Tapeout (left) and ASIC (right) pictures

4.4 STSXYTER concept

4.4.1 STSXYTER Overview

The front-end readout ASICs for the silicon strip detectors within the STS detector of CBM will be built by a collaboration group of the AGH University of Science and Technology in Krakow Poland. They are planning a time-over-threshold based readout chip. Their first prototype TOT01 [78] has been produced in an 180µm CMOS process as mini@sic with Europractice. After it was successfully tested, an improved version called the TOT02 [79] was produced. The final STS readout ASIC will produce a high amount of data and needs to be integrated into existing readout chains. Thus, the new version now called STSXYTER will have an integrated CBMnet protocol and will use the generic module concept. This ASIC can then use the standard readout chain within beam times and does not need a protocol conversion. Due to reusability of I2C, the control path, and the standard cell based SERDES this development will have enough readout bandwidth and can be tested in already existing test environments. In addition, using the generic module concept will save development time for the STSXYTER.

4.4.2 Integration

The STSXYTER integration concept, presented as a block diagram in figure 4-5 is similar to the SPADIC. STSXYTER includes the same and now hardware proven built-in blocks as I2C implementation, RF, CBMnet modules and standard cell based SERDES. There are mainly three differences. The STSXYTER requires more bandwidth. Thus, instead of two outgoing 500 Mb/s lanes, four lanes with a total bandwidth of 2 Gb/s are implemented. These new lanes are automatically managed and administrated by the hardware modules of the incoming data link. The delivered clocks for the STSXYTER differ. There is only the 250 MHz bit-clock and the 25 MHz word-clock required. The last difference is the analog control interface directly attached to the RF. Here sub register files in the analog part are used and connected via sub register file interfaces to the main RF. Due to the innovative

modular concept most of the source code was reused. New blocks were simply attached to the interfaces and thereby integrated into the design. Additionally, this reduces fabrication risk, because of the well-tested hardware blocks.

The CBMnet generic modules are already prepared and development of the analog ASIC part integrating it is ongoing. Concerning the LVDS cells, it is planned to use the same cells, which are used in the SPADIC. As for the SPADIC integration and tape out, this will be done in close cooperation between both development groups. After successful post place and route simulations, tape out is planned for the last quarter of 2012.

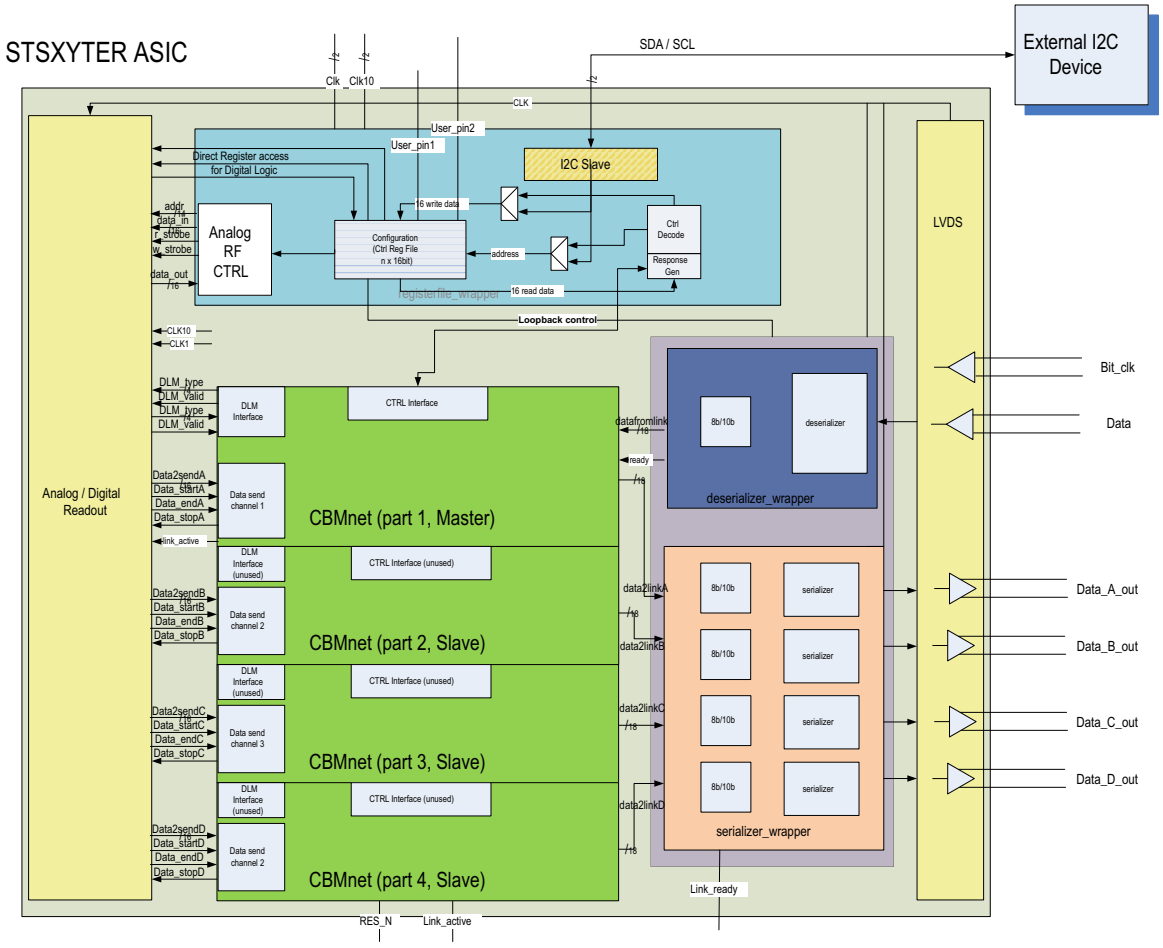


Figure 4-5: STSXYTER Structure Diagram

4.5 Conclusion

The generic module concept has been implemented and has shown its value to the CBM project. The goals have been achieved through generating a set of modules to supply CBMnet users. Besides delivering a synthesizable code base, it delivers complete simulation environments with testbenches and automated tool flow support. The generic modules are integrated into various supported FPGAs. Additionally, there are two FEE detector readout ASICs with direct integration of generic modules. One of them will be used during beam times at the end of 2012. It is assumable that additional groups will want to benefit from this approach and will join the user community. In addition, this concept will be useful for the HUB ASIC development described in chapter 5, *The HUB CBMnet ASIC Development*. It has shown that it saves design time and is useful in achieving the goals of the CBM network.

Chapter 5

The HUB CBMnet ASIC Development

This chapter describes the design analysis and required functionality for the CBM HUB ASIC. It is responsible for combining multiple front-end electronic detector ASICs and transfers their messages to a data processing board. Therefore, it contains CBMnet interface modules, a RF, a crossbar structure, buffers, and two types of serializers and deserializers. Clock distribution and synchronization are special features within this ASIC, because synchronous data detection for FEE detector ASICs must be assured. Implementation of the HUB ASIC is discussed together with corresponding tasks and challenges. In addition, a concept concerning the opto-converter transforming high-speed electrical signals of the HUB into optical signals for long distance communication is presented.

5.1 Toplevel Concepts and Design

The development and analysis of the hierarchical readout structures for the CBM network [80] based on FPGA technology and successfully used in several beam times together with requirements for the final experiment setup led to concepts for a HUB ASIC [81] to improve the CBM network structure. This HUB ASIC needs to efficiently provide early data aggregation, radiation tolerance features, clock distribution mechanisms, synchronization capabilities, and reasonable link bandwidth. The overview regarding a portion of the planned CBM data acquisition (DAQ) network structure, including the HUB ASIC, is shown in figure 5-1. The hierarchical CBM network structure is typically divided into three network regions.

A near detector region including front-end electronic (FEE) detector ASICs assembled onto front-end electronic boards (FEB). These front-ends use HUB ASICs for read-out because of bandwidth and link-speed requirements. Additionally, it is responsible for FEE initialization, control and data streaming, clock distribution, synchronization, FEE control, data aggregation and link speedup. Especially for the TRD and STS detectors, the HUB ASIC is required to enable the read-out, because they generate the highest amounts of data within the CBM experiment. Current statistics show that for TRD readout, up to 30000 TRD detector ASICs are needed, each having 2x lanes connected to around 1800 HUB ASICs. The STS requires readout for around 17000 detector ASICs connected to more than 2000 HUB ASICs. These numbers show the importance of an efficient read-out ASIC that fulfills all requirements. The HUB ASICs are closely coupled together with opto-converter boards for an electrical optical conversion to enable long distance communication. Due to density and radiation, the HUB implementation can not be done using FPGAs. The complexity of this ASIC depends a lot on the amount of supported FEE devices. Because crossbar structures become more and more complex and must thereby be hierarchical, thus especially timing and area become critical. In addition, synchronization effort for front-end devices increases in complexity and the start-up time of the network increases. Thus, the amount of FEE devices supported for each HUB must be carefully considered.

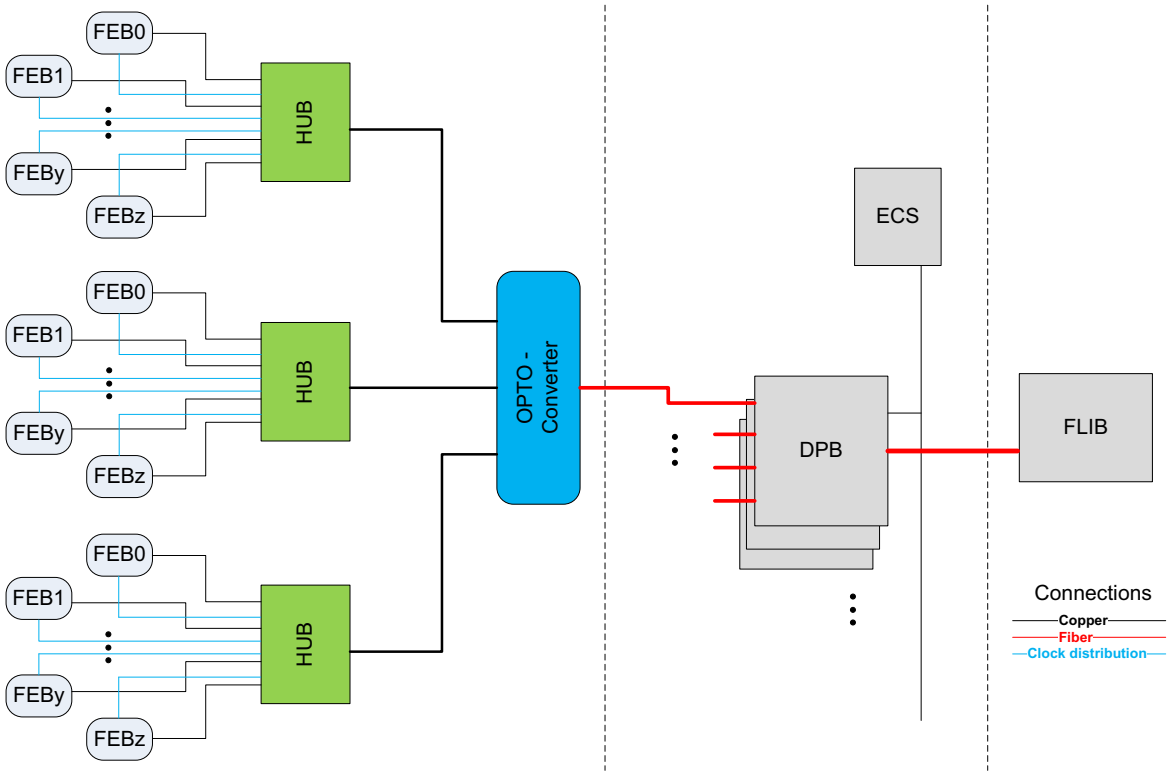


Figure 5-1: CBM DAQ Structure with HUB

The middle region of the CBM network includes a service area represented by the planned data processing boards (DPB) which presents the unified functions of the data combiner board (DCB), providing data combining and preprocessing features with the detector control system (DCS) responsible for clock distribution and synchronization. An experiment control system (ECS) is used to provide the clock distribution, synchronization, and network control to enable all required features within the CBM network.

The third region is the compute cluster area including a FLES Interface Board (FLIB) to receive the data and appropriately store it for a First Level Event Selector (FLES). The FLES is responsible for the selection of interesting events and thus supports the computing cluster concerning data processing.

In the following section, the planned usage of HUB ASICs will be described. It focuses on its responsibility for bundling multiple FEBs with relatively low data rates together to sustain higher bandwidth connections. This is done by supporting FEBs with an adaptive number of low data rate channels for sending its detector data. Due to technological constraints, the FEE ASICs use a data rate of 500Mbits/s. Coming from the CBM network using fast, unified, and balanced bidirectional communication links supporting clock data recovery (CDR), for front-end communication, the possibility for unbalanced communication must be provided with a separate connection for clock transmission to FEE detector ASICs. This simplifies the FEE detector ASICs, because they do not have to implement CDR. Then, the task of synchronization and clock distribution among FEBs is presented. The electrical to optical conversion is done in a separate converter board using as many commercial off the shelf (COTs) components as possible in order to reduce cost and guarantee availability. Active optical cables (AOC) are another interesting variant for delivering high efficiency and flexibility within this concept are also shown. Finally, flexible HUB ASICs build-up structures supporting different types of detector systems are presented.

5.2 Hub ASIC

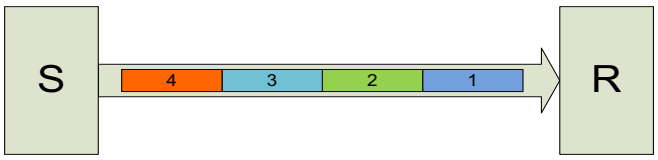
5.2.1 Interfaces

The interface of the Hub ASIC will consist of multiple 500 Mb/s electrical connections to the FEE detector ASICs and a small number of 5 Gb/s optical connections into the back-end direction. Depending on the technologies used, the speed of serial links may be increased up to 7.5 Gb/s, which would reduce the number of fibers and link inports at the concentrator and processing level (DPB). However, the 5 - 7.5 Gb/s link needs an electrical to optical conversion, because of communication distance. This is done on a separate opto-converter board.

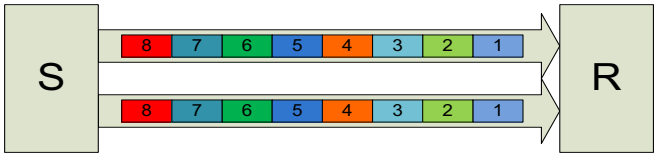
At the FEE detector side, the challenge is supporting FEBs with the flexibility to choose between an x1, x2 or x4 connection for detector data transport that provides enough bandwidth for different kind of FEE ASICs. This is necessary, because FEE ASICs currently use 180nm technology that already works close to the edge concerning their clock frequency per channel bandwidth. It can not increase any further. Due to the fact that there is only a small amount of traffic streaming into the detector, multiple lanes can be used in an unbalanced setup for communication. This means only one control channel with an additional clock lane to an FEE is required, while up to four lanes away from each front-end are possible. However, using balanced links, with their own clock lane or a CDR for each link with a bidirectional implementation, consists of too many unnecessary lanes and increases logic complexity within an ASIC. In case of multiple clock lanes, it is responsible for the synchronization of multiple jittering clocks. Thus, only one clock lane is used as the master clock. Using a separate link for the clock simplifies the FEE ASICs significantly, because there is no CDR in FEEs necessary. Of course, this increases complexity for link control and initialization within the HUB ASIC, but overall this proves to be less effort. Furthermore, a solution must be found for skew alignment among multiple lanes without any embedded clock. Phase shifter at an input path or probably even on an output path of each lane would be one solution. Such delay elements have been developed for TSMC's 65nm technology and have tested successfully. Nevertheless, always finding the right sampling points while guaranteeing deterministic behavior at all times and among parallel lanes is a challenging task. There need to be training patterns, automatic alignment mechanisms, and non-trivial mechanisms to setup deterministic latency. Another task is to analyze the configured links to find out if a static configuration is stable or if dynamic adjustments have to be done. In addition, the behavior concerning temperature variations, radiation, and power supply changes have to be checked and a proper operation has to be assured.

In addition, an important design decision to be considered is the methode of splitting messages over different lanes, when the usage of multiple lanes is required to increase bandwidth for FEE ASICs. The separation could be a 16-bitwise split over multiple lanes, or a message based kind of separating lanes using them autonomously. Flow diagrams showing

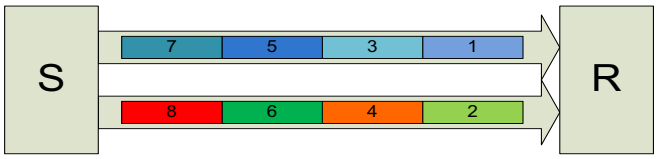
the difference between a split and a separated solution are depicted in figure 5-2. Figure (a) shows the common 1x data flow having one message after another onto a single link. In the following figure (b), a 2x split flow delivering each message on both lanes is presented. All arriving messages do not have a 16 bit width anymore, as the per lane deserialization width. For a 2x lane, they are now 32 bits and for a 4x lane, the width would even be 64 bits. Due to an increased bit width, all protocol processing modules have to be rewritten to enable processing of new data width in an efficient way. Below in figure (c), a separated solution using a 2x connection depicts data flow handling of complete messages on each of the lanes. Thus, increasing bandwidth using additional lanes leads to the parallel sending of multiple messages. This multiplies the amount of used hardware modules in the receive stage for message processing, but identical modules can be used. Furthermore, out of the data flow perspective for a lane, it makes no difference whether it is within a 1x, 2x, or 4x lane bundle.



(a) 1x Data Stream



(b) 2x Split Data Flow



(c) 2x Separated Data Flow

Figure 5-2: Spilt and Separated Data Flow

The support for 1x, 2x, and 4x would lead to three different kinds of input structures. In addition, the link initialization sequence must be enhanced to support multiple lanes in a split mode. A block diagram showing a data flow structure example for a separated solution within the HUB is shown in figure 5-3. It presents the structure for processing data and needs only two different clock domains using the same data width. This configuration type referring to a first assumption shows flexibility for free assignment of frontend devices using different lane counts. The required rate conversion increasing the frequency can be handled in the crossbar part.

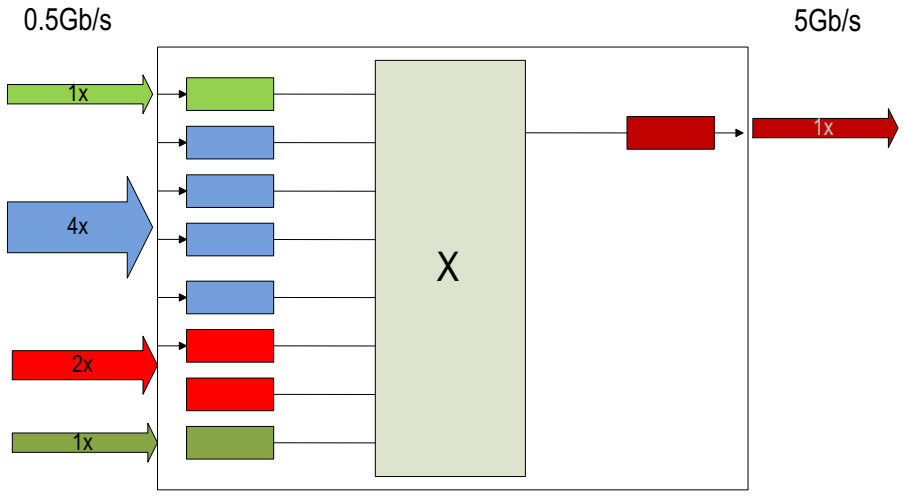


Figure 5-3: Separated Data Flow HUB Structure

On the other hand, split data flow requires completely different input structures for supporting all the possible input configurations. This leads to three input levels with different processing modules with different processing widths. Either within the next HUB part, a direct rate conversion is required or different crossbars have to be used to merge data streams. The direct rate conversion would lead to one additional module for each non 1x link. Additional changes in the arbiter to guarantee a fair arbitration are required. Thus, it needs some additional hardware and it is not a clean implementation. The most likely alternative using different crossbars and merging them afterwards is shown in figure 5-4. This solution is a clean straightforward solution, but requires three crossbars. The incoming data channels

with different data widths are marked in green for 1x, red for 2x, and blue for 4x. There is still a lot more hardware required, but there are only three rate conversion modules, one for each crossbar. A problem here is fair arbitration between different speeds.

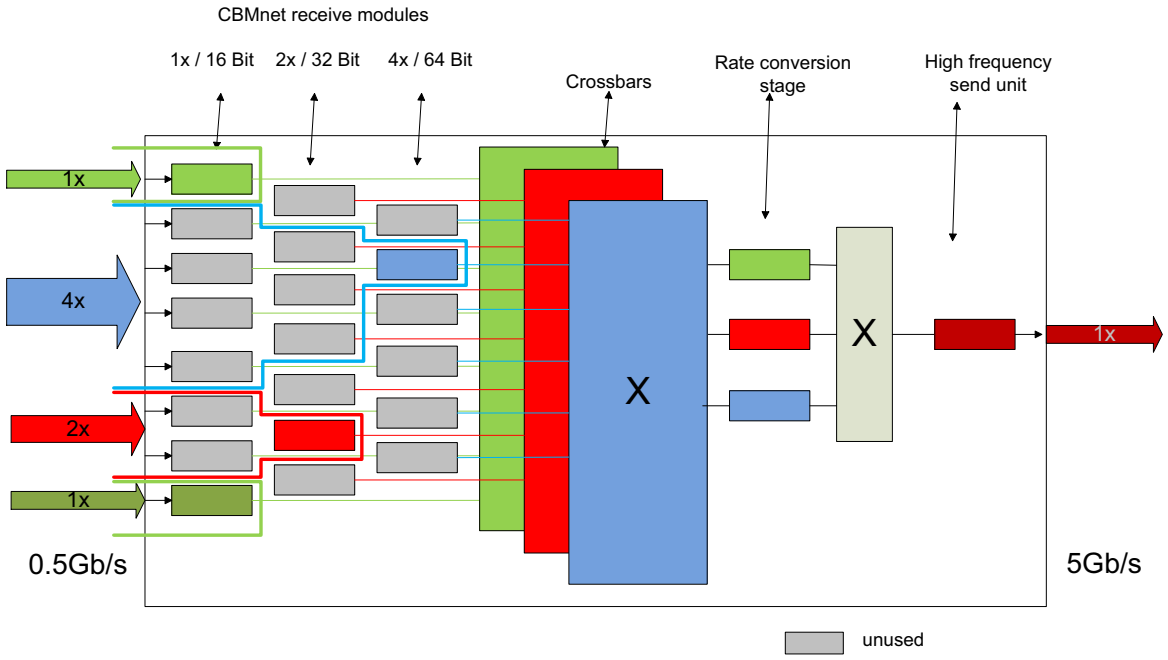


Figure 5-4: Split Data Flow HUB Structure

Table 5-1 compares pros and cons between both possible ways of implementation. The working assumption emerging out of the arguments is that supporting the separated message based methode results in the only feasible implementation, because of the enormous additional complexity and the high risk of optimization problems will lead to significantly more development effort. This leads to multiple lane connections being completely separate link connects. Then epoch markers must be sent on all lanes, because message order can only be guaranteed for separate lanes. Due to the very high complexity of a free configuration between 1x, 2x, and 4x links to FEBs, the additional required logic for implementing links sending each message on all four lanes of a 4x link seems to demand a lot more effort.

	PRO	CONTRA
Split	- Only one wide interface required	- Extra training pattern required - Complex lane synchronization concerning deterministic latency message detection - Needs three different receive structures - Two extra clock domains - Crossbar structure becomes hierarchical and needs two more clock domain crossings - Increased crossbar complexity will lead to optimization and speed problems - Two different clock domains for TX and RX on FEE side => Configuration options must be restricted
Separated	- Standard initialization for CBMnet V2.0 usable - Only one receive structure type required (replicated structures) - Efficient crossbar structure within HUB possible - High flexibility for user defined front-end configurations - Hub chip stays with two clock domains	- Epoch markers have to be sent on all lanes - Possible additional overhead in case of bad utilization - Requires the handling of up to four message interfaces within FEE ASICs

Table 5-1: Split over Multiple Lanes vs. Separated Messages

The other interface connecting DPBs consists of balanced bidirectional 5 Gb/s or 7.5 Gb/s links. The difficulty here is that fast serializers are required which include CDR and have to be radiation tolerant. The recovered clock has to be good enough to be used for sending and processing communication streams, because the complete system runs with deterministic latency which enables synchronization over common links. This deterministic latency is not only required during runtime, it must also be given over power cycles. The recovered clock must have a good enough jitter to be used for further clock distributing and it must be usable

as send reference clock for all sending structures. Otherwise, a cleaner device like the one used in the demonstrator I with DCBs has to be considered. The CBM protocol V2.0 will be used for both interfaces. The separated clock to the FEE does not affect the usage of the CBM protocol at all. Figure 5-5 shows the structure of a HUB ASIC having up to 4x lanes per link from the FEE detector devices, using only one clock and one control lane towards them. This configuration delivers sufficient bandwidth to the FEE of 500 Mb/s for control messages and a variable bandwidth for the FEE data between 0.5 Gb/s and 2 Gb/s, supporting different bandwidth requirements for the FEE ASICs.

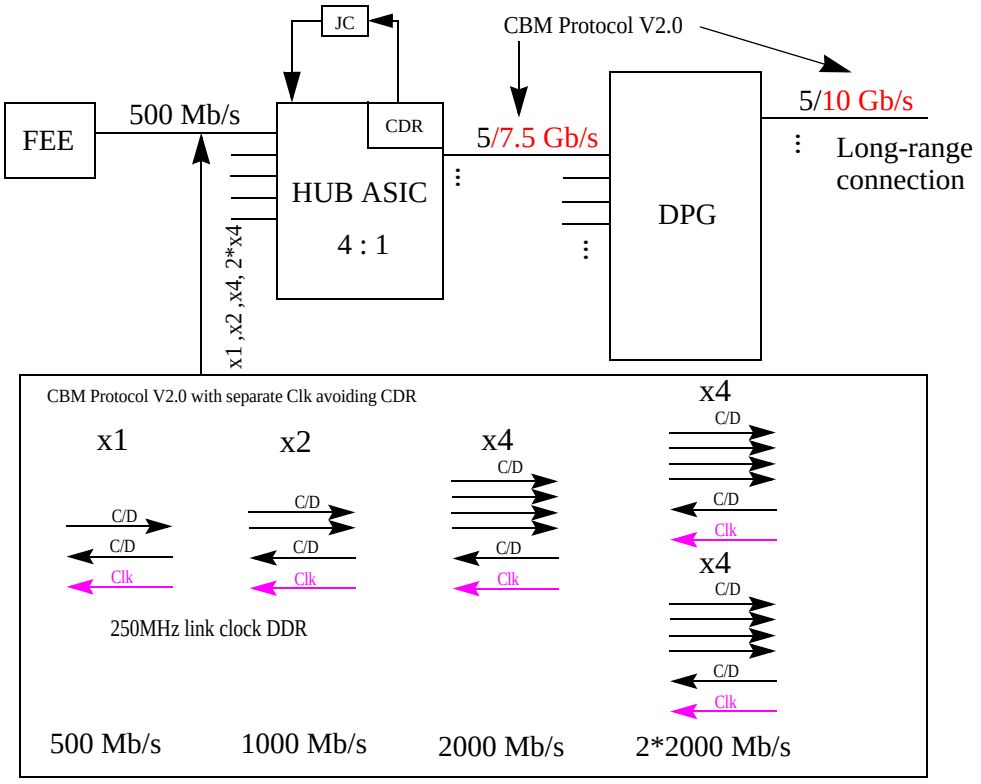


Figure 5-5: Hub ASIC Interfaces

5.2.2 Integration of CBM Protocol Modules

A toplevel block diagram of the HUB ASIC is presented in figure 5-6. Its structure shows three different logical blocks representing different possible clock domains. The front-end device interface includes 500Mb/s serializers/deserializers, logic blocks for initialization and bundling, a clock distribution mechanism, a synchronization and delay adjustment mechanism, and CBMnet protocol modules. The DPB interface block requires radiation tolerant high-speed serializers/deserializers, an integrated CDR, and CBMnet link modules.

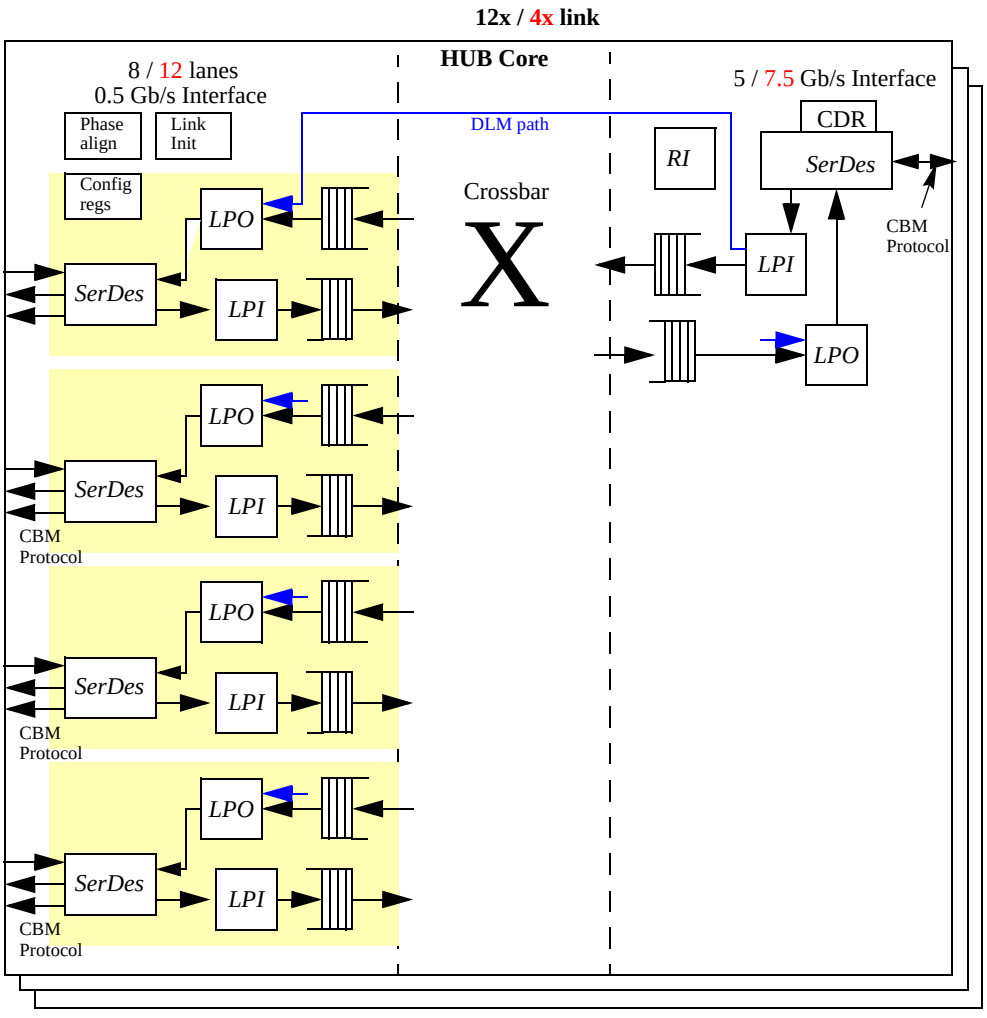


Figure 5-6: HUB ASIC Structure

The inner module consists of a hierarchical crossbar structure supporting different output link speeds, including synchronization FIFOs arbitrating communication between front-end devices and the DPB stage. The internal width for data and control streams should be 16 bits, as defined in the standard for the protocol. This results in a moderate internal clock speed of 250MHz at 5Gb/s or 375MHz at 7.5Gb/s. The crossbar design runs at the same frequency as the DPB interface generating enough utilization for faster links and ease synchronization at this point. The optimal design would use one clock domain for the complete chip, but due to the reduced speed for the front-end devices, this does not seem achievable

Analyzing the crossbar structure for the HUB ASIC in more detail leads to some basic requirements as a separate path for DLMs in order to guarantee determinism and priority request insertion. In addition, the fact that virtual channels for data and control can not be handled within one crossbar because of deadlocks and starvation has to be considered. Additionally, the different types of traffic have to be analyzed. Data is only sent towards backend stages. This eases crossbar requirements for data messages. Control messages come from arbitrary points within the network like a DCS or an ECS. These control nodes are permanently fixed nodes during runtime. Thus, a control crossbar must support messages in all directions. Fault tolerance features are necessary to guarantee permanent usability for a HUB ASIC design. There are different implementation variants for the crossbar message flow of data and control traffic classes connecting multiple slower FEE device links with faster 4x links. Figure 5-7 shows the most interesting ones. The first implementation, (a) direct mapping, merges 8 to 10 FEE links into one link towards the DPB. This variant has the advantage of being a compact modular design, which could be replicated in parallel within an ASIC to save design time. In this case it is clearly defined how each FEE link is statically connected, which might help handling data in the following processing stages. Nevertheless, there is no fault tolerance included in this system and a link failure leads to losing a complete read-out tree. Additionally, for control traffic connecting a DCS or an ECS it is less than optimal, because all control messages need to be routed on fixed lanes over the right DPB links. Thus, there is a lot of overhead and inflexibility in the following stages. The second variant, (b) partially connected, considers fault tolerance by add-

ing a crossbar structure. A DPB link failure would reduce bandwidth, but would not lead to losing a complete read-out tree of detector ASICs. It still does not solve the problem of control flexibility. The last implementation type (c) delivers a full connectivity. Ideal for control flow, because no matter where a control message comes from, it can just be routed everywhere using its address. In addition, data message flow shows an interesting feature, with

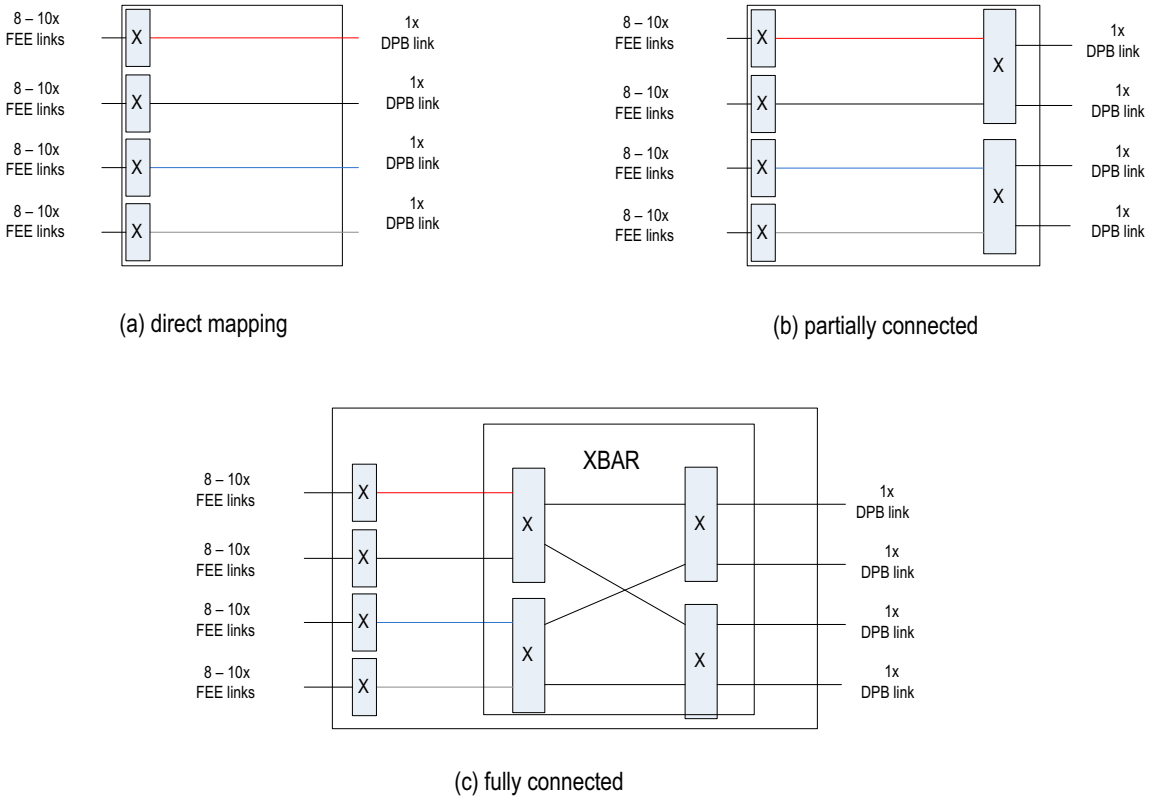


Figure 5-7: XBar Structures

the possibility of balancing the data flow over all outgoing links. This could lead to a still sufficient data capturing in case of link failure and non-full utilization of all basic links. This approach seems promising and needs further analysis to achieve an optimal solution which considers similar implementations like Clos [82], Benes, and Banyan [83] switch networks. Due to high complexity, using one huge crossbar for the design seems to be out of the question. Upon analysis, using a multi-hierarchical crossbar seems to be a good deci-

sion. Only for control messages, a full connectivity is required, so further analysis during implementation may lead to different crossbar variants used for the control and data traffic classes.

Figure 5-8 shows a block diagram depicting the data flow structures of the HUB ASIC. It presents one of the possible arrangements for FEE links and back-end connections. Even if the HUB receives and transmits unified traffic streams, internally they need to be handled separately. Of course, it would be easier to aggregate unified streams, but then deadlocks could occur and synchronization over the CBMnet would not be possible. DLMs need a separate deterministic path between the two deterministic link regions. Especially for the special synchronization mechanism used to adjust front-end device links and insert DLMs at the right time to guarantee detector device synchronism. The data and control traffic classes processed by crossbar structures need to be virtually separated at the crossbar stage or else two disjoint crossbars have to be used. Thereby it is assured that a data message can not block a control message or vice versa. The crossbar implementation needs to fulfill special requirements. There are many front-end links attached to the HUB ASIC. They require fair arbitration, because in case of a throttling effect on the back-end link, starvation of detector endpoints is not tolerable. Directly attached to the crossbar, synchronizing FIFOs need to assure that only complete messages received from the front-end are processed, because of the different clock speeds. In case of a back-end link failure, the crossbar must be able to reroute the messages. Thus, the required fault tolerance is achieved. There are a lot more issues which have to be handled. In other projects concepts for efficient data processing within crossbars [84] and a special networks-on-chip framework HTAX [85] have been developed. These concepts and all requirements on the final structure must be further analyzed and an efficient CBM crossbar solution must be found during the HUB ASIC implementation. This is important, because a significant part of the HUB ASIC performance depends on the crossbar structure used.

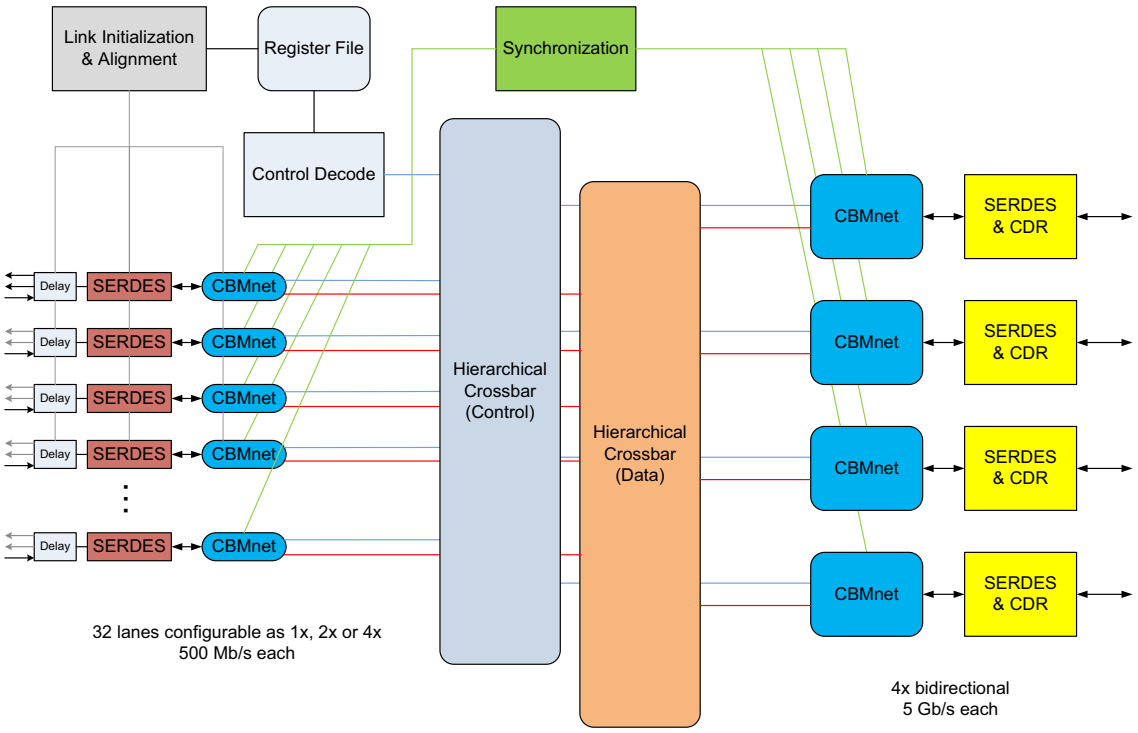


Figure 5-8: Block Diagram of HUB Structures with Traffic Flows

5.2.3 Dimensions and Partitioning

One design option of the HUB ASIC is to implement 12 high-speed SERDES per chip. This would match directly with a high-density AOC cable, which carries 12 lanes into each direction. It could be adapted to the needs of CBM by using x12 lanes downstream and only 1 lane upstream. It will be analyzed if this is necessary to reduce costs or if symmetrical AOCs can be used without significant cost increase. 12 SERDES can support 12x 5Gb/s which results in 12 x 2 x 4 x (4+2) (4 channel to FEEs or FEBs per lane, all together 96 channels per chip) differential LVDS pairs with 576 pins. Depending on the configuration, some of the LVDS lanes stay unused in the later setup, but the HUB ASIC requires the full setup flexibility. A 12x SERDES requires 135 pins, and control and test at least around 20

pins each. The estimation for the additional power pins needed is 240. All together, this results in 971 pins for a 5Gb/s 65nm implementation. Analyzing this pin amount leads to the fact that the ASIC will be pin limited. The package will be a BGA with flip chip attachment of the die. Flip-chipping is required due to the high frequency signals of the SERDES. The 971 pins require a 31x32 bump array. Assuming an 180µm pitch, chip size is 5580µm x 5760µm = 32.14 mm^2.

Due to the high density of 96 connected FEE/FEB channels and the calculated chip size, it makes sense to split the 12x SERDES into three parts. This size delivers more flexibility for build-ups and a better yield due to smaller ASIC size. SERDES IP is typically partitioned into 4x blocks anyways. This results in a HUB ASIC implementation having 32 channels connected to one x4 SERDES using 361 pins/bumps (19x19). Up to three chips will then be connected to one 12x AOC. Figure 5-9 shows a true to scale picture of this 65nm HUB ASIC including a 4x SERDES, a realistic amount of SRAMs, and LVDS IOs.

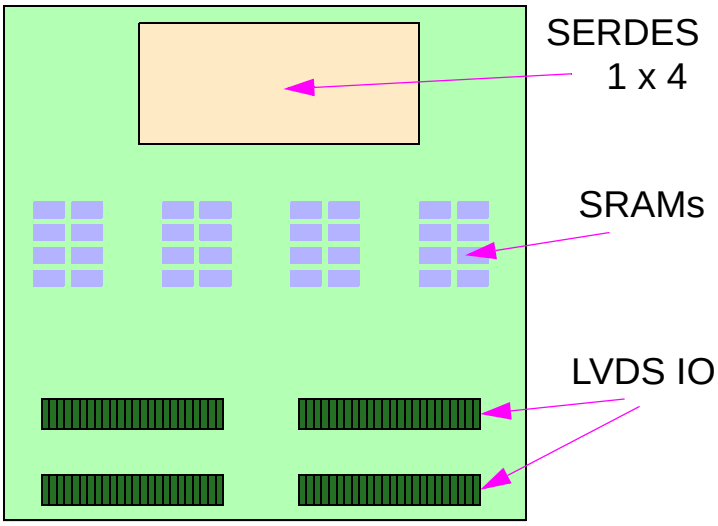


Figure 5-9: HUB ASIC 65nm version (3420 x 3420um)

However, assuming that the proposed chip size provides enough pins for a 32 link to 4x conversion, it is ideal for current production possibilities, and a 4x lane fits to the read-out concept. Further analysis can be done to find other solutions under these basic conditions. Thus, a different variant for an HUB ASIC Type2 could be the use of 20 times 2x connec-

tions instead of 32 times a 1x. This requires only 80 differential LVDS pairs (20 x {clk, data_out, data_in0, data_in1}) instead of 96 (32 x {clk, data_in, data_out}). It provides the same flexibility for 4x and 2x connections, but for each 1x connection one data_in LVDS remains unused and only 20 1x connects are possible. It also restricts the flexibility of attachment permutations. Nevertheless, more 2x or 4x connects are accessible, which increases the maximum bandwidth from detector chips. It leads to a utilization of the back-end links of up to 100% concerning the FEE data. However in the previous proposal, leaving some additional space in the back-end links had a purpose. There was space left for hardware fault tolerance and for retransmission. With the HUB Type2 concept, retransmission during data capturing with full utilization would lead to buffer overflow and there is no space left for hardware fault tolerance. Additionally, sending and receiving control messages between the HUB ASIC and a DCS or an ECS for controlling FEE endpoint devices and the HUB itself would lead immediately to backpressure and then to buffer overflow. Thus, using the HUB Type2 design requires clear data and control times. Furthermore, it needs frequent system dead times together with additional buffer space to absorb retransmission events and allow at least some fault tolerance. This might be reasonable for some detector systems, but will decrease flexibility and can lead to less useful solutions for others. Nevertheless, a final design decision requires further analysis, because both proposed variants define a realistic basis.

5.2.4 HUB Challenges

While designing the HUB ASICs there are many tasks to be handled for problems, which should be solved like dynamic configuration between 1x, 2x, 4x lane support for data streams from FEE endpoint devices. Therefore, low-level initialization and hub configuration need to support optional disabling of clock and data-out within each group of clock, data-out, and data-in. This enables implementation of 2x or x4 links. In addition, lanes from a FEE device should be mapped into the same high-speed SERDES lane towards back-end, for example, two 4x input streams mapped into one SERDES running with 5Gb/s mode. This must be supported to ease data handling in following stages. The clock distribution

will use a 250 MHz clock, because of the standard cell SERDES using a 500Mb/s LVDS interface with DDR between HUB and FEE devices. This clock recovered in the CDR part of the fast SERDES must achieve reasonable jitter values to be used as a reference clock for detector endpoints.

The CBMnet uses an 8B/10B coded data transfer and the receive data must be phase aligned at hub receive path, because there is no clock data recovery within FEE devices. Therefore, delay cells for phase adjustment at HUB ASIC chip are required and during initialization, an automatic alignment has to be performed. For data aggregation and correct distribution towards the back-end, a special crossbar structure must be built. This crossbar must retain package order and specific bypass structures must assure the DLM functionalities to enable the synchronization mechanisms using priority request insertion.

The high speed serializer deserializer with clock data recovery (CDR) provides at least 5Gbit/s @ 65nm technology, but it may be targeting 7.5 Gbit/s. It must either be designed and produced by CBM Collaboration partners from India who are specialists for SERDES analog designs or bought if available on the market. They must provide a deterministic behavior and concerning CDR, it must be analyzed, if a jitter cleaner integrated into the HUB is required. PLL supporting such features was designed together with RWTH Aachen. All this functionality must be heavily tested and verified.

All preexisting modules and building blocks have to be adapted and integrated into the design. Together with the integration of an external IP, as for example SRAM, PLL and I/O-cells, the high amount of new code in the design will exceed a critical size. Thus, a complete implementation that uses a professional verification environment using universal verification methodology (UVM) [86] is required to have a realistic chance of not requiring endless reruns for such a complex chip. In addition, a detailed analysis for a fault tolerant design, fault recovery features, and link failure recovery needs to be done. Fault tolerance extensions adding redundancies are required to provide a radiation tolerant design. Single event effects (SEE) [87] must be handled as single event upsets (SEU), single event transients (SET), and single event latchup (SEL). Therefore, FSMs could be implemented using ham-

ming coding [88], SRAMs with error-correcting code (ECC) correction and triple modular redundancy (TMR) structures must be used. This needs further evaluation and requires reliable solutions.

5.2.5 Initialization, Control and Synchronization

A complex device like the HUB ASIC requires a link initialization that starts automatically as soon as a physical connection is detected. This initialization procedure needs to establish a physical link connection. However, the most important task of the HUB ASIC, besides data transportation, is synchronization. Therefore, running on a deterministic network, CBMnet protocol provides the ability of synchronization through its unified traffic links by using DLMs. Thus, during automatic initialization, additionally the synchronization mechanism has to be calibrated and deterministic latency on each link must be guaranteed. It is not only required to implement deterministic hardware, but also necessary to derive all clocks from one master clock. Otherwise, deterministic behavior can not be assured at all times. There are two ways to distribute this clock. One way is using a clock tree that provides a clock to each stage within the network. This requires additional hardware and space for a clock net. The second way is to use an embedded clock within a data stream, recover it at a receiver and use it as reference clock. The HUB ASIC implements both methodologies. The high-speed lanes connected towards the detector control system (DCS) use CDR, delivering the reference clock for running local logic blocks and sending data. The initialization and synchronization of these lanes run with the standard CBMnet algorithms. On the other hand, for FEE device clock distribution a separate clock lane is used in parallel to the data stream. This clock is then used for sampling data, deriving clocks for internal logic, and sending data back to the HUB. This simplifies FEE endpoint links on both sides running without CDR, but received data at HUB ASICs has to be specially adjusted for reliable and deterministic sampling. Furthermore, clock edges of FEE endpoint link bit-clock and word clock need a well-defined relationship that leads to bit clock accuracy on the links. Figure 5-10 depicts the bit-clock synchronization, which has to be achieved at the front-ends. A synchronization sequence measures the FEE endpoint device distance and adjusts

all the devices to the same arrival point for DLMs. The mechanism used to adjust devices is to delay all links to the first joined synchronization point. Synchronization can then be achieved by sending DLMs to FEE devices which use the standard synchronization algorithms developed. DLMs arrive exactly at the same time. Then timing counters can be set to a common time or other special purpose actions can be performed. Thus, within the HUB hierarchy level, all HUBs assure a common synchronization view for their attached devices. The HUB clearly shows how to set up synchronization for the CBM network system. Each hierarchy level guarantees the synchronization for its sub trees, which leads to an exact arrival of DLMs, sent by the DCS. This runs automatically and is planned as basic initialization. All configuration values are stored within the RFs. However, this gives the user the ability to change them. This is required to support different detector types in one chain having different data preparation times for sampled data. It does not know about device-specific behavior, because time adjustment is only done on link-based network information. If e.g. due to a power cycle a sub tree loses synchronization, a resynchronization sequence is performed. Then after a well-defined epoch, the sub tree is synchronous again.

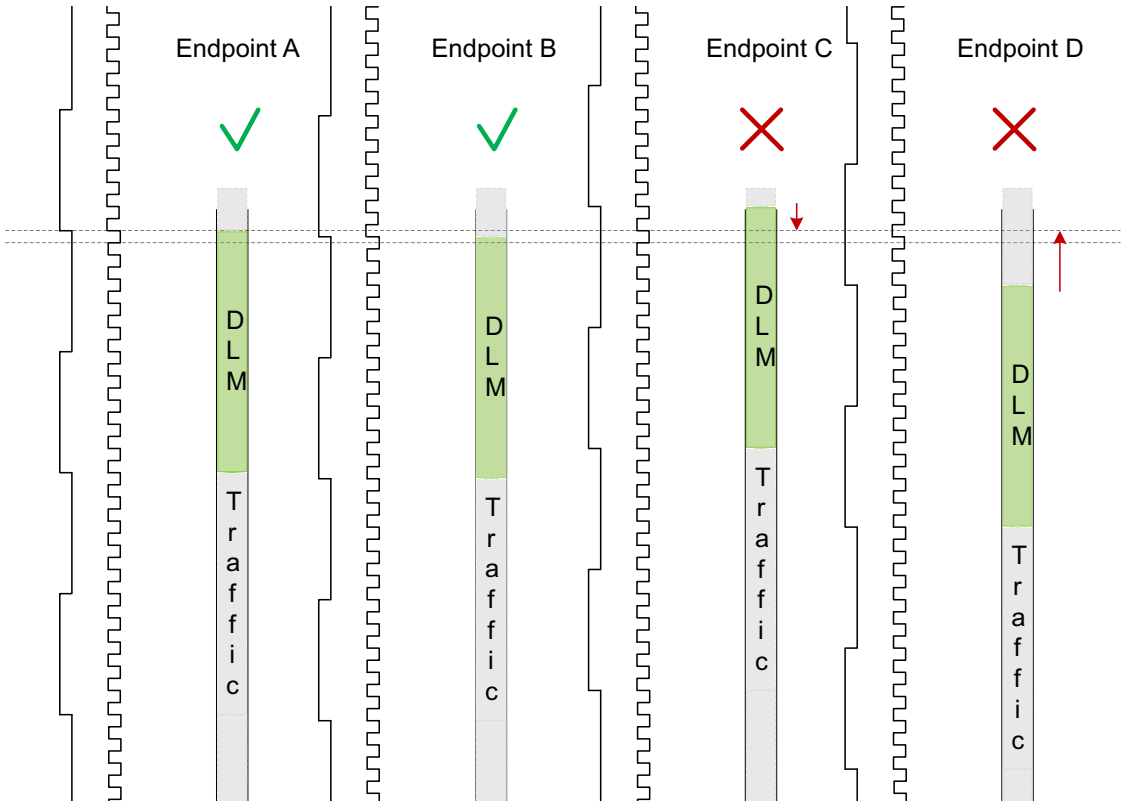


Figure 5-10: Synchronization Mechanism

The synchronization algorithm used for synchronizing devices attached to the HUB is subdivided into different phases.

Phase 1: Special characters are sent to FEE detector ASICs. These characters are detected and an acknowledgement (ack) or a non-acknowledgement (nack) character is sent back to the HUB. In case of a nack, the output delay is adjusted until a certain number of acks is received. This may require using different input tabs. Correct sampling for FEE detector ASICs is now assured.

Phase 2: The HUB uses its send clock to sample incoming bit-streams. Because of different cable lengths, changing input delay adjustment settings are required. An automatic mechanism runs through all delay tabs and an eye detection algorithm is performed. It samples the

ack character for a certain time on each tab and marks the tabs with non-stable values. Then a tab is chosen with the highest distance to marked tabs delivering a stable value. A reliable point for sampling input streams is found.

Phase 3: A DLM0 is sent to the FEE devices and clock cycles required for a roundtrip are measured. Then together with hardware clock cycle information, a distance is calculated. Afterwards, all FEE device connections are balanced out by delaying them to the level of the slowest device. Now all devices receive their DLMs on an equal arrival time.

Phase 4: Deterministic latency is guaranteed and the standard initialization and synchronization sequence can be performed.

In addition to all the standard synchronization with DLMs, a masking for FEEs to receive only specific DLMs is possible. Therefore, masking registers are available for each HUB configuring the FEE connections. These registers are read and write accessible by control messages. Thus, registers of delay values for DLMs to FEEs can be modified. This enables a manual adaption using some specific FEE hardware with different delay behavior and multiple chip types can be supported and synchronized within one setup.

5.2.6 Prototype Measurements

Before HUB ASIC prototypes become manufactured, a simulation environment for detailed simulations and a FPGA platform for firmware prototyping are required. These platforms will be used for developing, simulating, and testing of the different digitally designed parts. The HUB ASIC is structured into logical blocks for delivering automatic initialization for links, setting up the synchronization system, managing crossbar structures, and providing capabilities to stream data, control, and synchronization traffic over unified deterministic links. Crossbar structure, CBMnet modules, and SFP link control can be used from the DCB firmware for first prototyping. Additionally, all the link tester modules for simulation and traffic generators developed for the last generation of beam time readouts can be reused. Thus, prototyping starts focusing on the front-end link initialization and synchronization. Especially, the unbalanced communication using 500Mb/s standard cell based SERDES together with delay cells for link alignment and the master-slave CBMnet protocol modules for communication, need to be simulated and tested. In addition, Tests for the data flow designed to generate back pressure on a data path when data cannot be delivered properly have to be performed.

In the context of a Bachelor thesis [89], successful simulations running the front-end device synchronization with different emulated cable length values were performed. Figure 5-11 depicts a successful synchronization that receives DLMs in all four simulated FEE devices within a bit clock cycle. This simulation has been enhanced and extended by some more tasks of the HUB. Additionally, successful simulations concerning read and write access to front-end detector device RFs have been done. First simulations focus on required parts for beam time readout implementations, but will be extended until a complete working simulation environment for all HUB logic blocks is available.

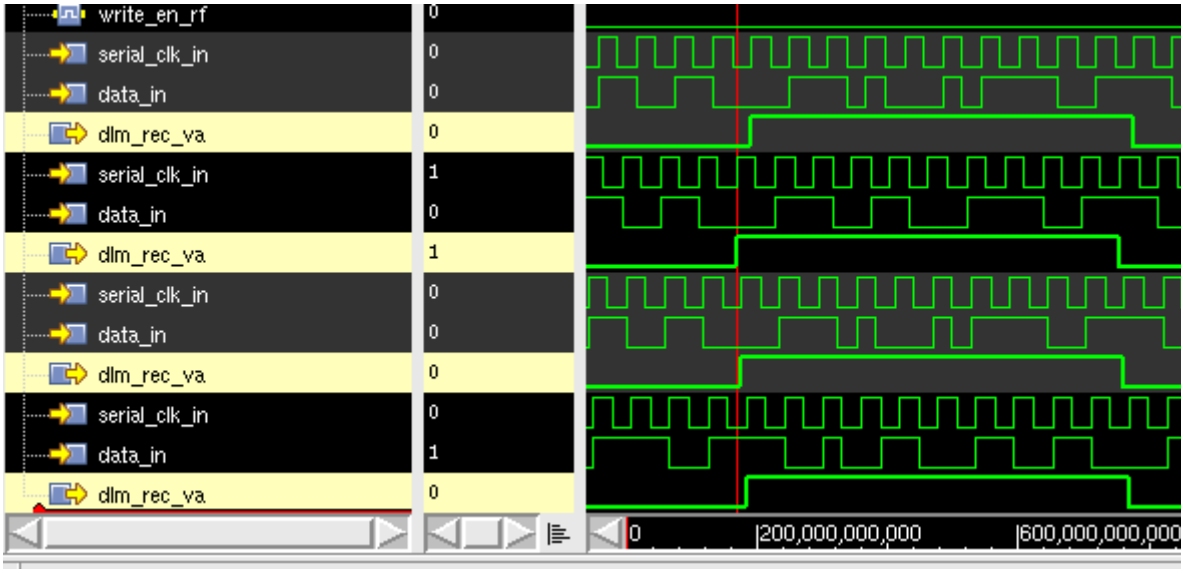


Figure 5-11: Simulation Results [89]

In addition to these simulations, FPGA prototyping was started first with emulations of front-end detector devices. Then prototyping setups for beam time readout usage was designed and tested. Therefore, Xilinx SP605 evaluation boards [90] were used, because the ROC3 boards planned for prototyping and as device for lab and test beam time readouts were not yet available. For connecting devices with the 500Mb/s links and as first cable to connect them within beam times, HDMI was used as the widely available standard. Therefore, FMC extender boards were built capable of connecting up to four HDMI cables for each SP605 device. The first front-end ASIC including the CBMnet, the SPADIC was initially tested while directly attached to the SUCIBO, a Virtex 5 based readout board. With this setup, link initialization and RF access tests were successfully performed. First Synchronization tests using two connected SP605 that emulate four front-end devices within one of the FPGAs shows that the synchronization mechanism should work technically. After hardware for attaching SPADICs to SP605 devices was available, tests showed that some adaptions had to be done to achieve the correct link lock. The algorithm detecting initial synchronization patterns sent by front-ends had to be slightly changed. The first imple-

mentation locking on stable signals had to be changed to lock on stable patterns [89]. This was the original idea, but it was adapted after successful simulation and emulation of the stable signal algorithm. Figure 5-12 shows a laboratory setup. This setup shows a SPADIC FEB attached to a SP605 test board. It runs successfully with an adapted implementation. The red lights at the SPADIC PCB depict that the SERDES modules are ready and the link is up and running. The third board, also a SP605, was used for emulation and was not part of this test.

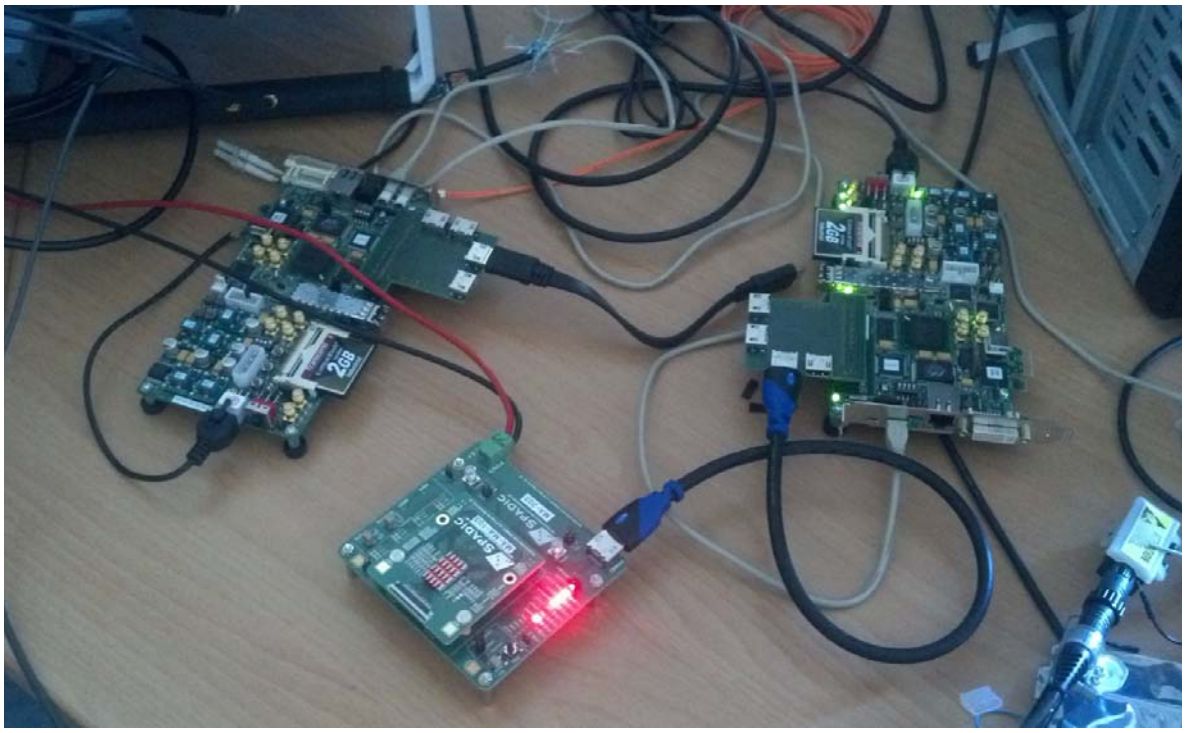


Figure 5-12: Laboratory Test Setup

In addition to this front-end part of a beam time read-out, the formerly used optical readout chain containing ABB and DCB devices was updated. The implementation for the SP605 fast SFP link connected to these boards was derived from a DCB implementation. This readout chain works reliable like in earlier used beam time setups.

This beam time prototype will be incrementally extended until it has all features of a HUB concerning basic functionalities. It will help to increase the value and success probability for test ASICs. Hopefully this will also reduce the amount of required test ASICs. These are the first steps within the progressive development of the HUB ASIC.

5.3 Opto-Converter Board

5.3.1 Design

The HUB ASIC must be complemented by an electrical to optical conversion stage to enable middle-range communication distance. This is necessary for transporting data in a concentrated way with high data rates. Therefore, an Opto-Converter has to be designed providing a possibility to gather multiple HUBs connected via copper together and perform conversion. The Opto-Converter can be on a separate Board or right next to HUB ASICs. The required optical components are more sensitive to radiation. Thus, the general assumption is that there will be a separate Opto-Converter Board. Achieving the high density and the possibility of adding a patch panel stage between build-up stages, a pigtail solution is one way to implement it. However, there are different implementations attaching fiber to VCSELs and photodiodes using pigtails, but the disadvantage of pigtails is that one has to replace the complete board with pigtails in case of damage. A different solution delivering a high density for fiber connections using multiple fiber lanes are Active Optical Cables (AOC). An AOC delivers an electrical connector performing the conversion within the cable. Thus it can easily be replaced by just changing the cable. Enabling a patch panel stage can be done by inserting optical connectors like MPOs/MTPs into the cable. Then, it is not a real AOC anymore, but due to efficient optical couplers, transmission quality will not significantly decrease. There are multiple variants of AOCs in the market, some with 4x and 12x connections. Because of higher density, 12x AOCs are preferred within first tests. Until all variants for implementing an Opto-Converter have been tested and the radiation tolerance is assured, there will be no final decision. However, all interim solutions should be

good enough for beam times. Most likely, the assembly of final solutions will be a combination of a special solution and COTs parts, so that there is a good availability and a high reliability guaranteed.

5.3.2 Prototype

The first decision concerning the Opto-Converter prototype was picking the right connector for the AOC. Therefore, the most important requirement for the CBM is density, due to restricted space within the detector region. A project at the Computer Architecture Group University of Heidelberg is HD-AOC [91]. This AOC is based on a high-density, 12x connector from Samtec and is 2.5 times denser than the best currently available market participant called CXP. HD-AOC uses special fiber attachment structures and optimized COTs components to achieve an ideal build-up height. The connector has been developed together with Samtec and became a standard connector defined by the HyperTransport™ Consortium (HTC) [92]. Samtec offers a male HDLSP Series Copper Cable and a female HDI6 connector with HDC Case [93]. The HDI6 connector delivers a double stacked build-up, with two levels, each 12x bidirectional. It is qualified for a bandwidth of above 10Gbit/s for each lane. This is enough for the target use of 5 to 10 Gb/s and results in an overall bidirectional maximum bandwidth of 240Gbit/s per HDI6 unit.

As basic components besides communication connectors, the board needs a power connector for usage of HD-AOCs and must deliver access to their I2C bus for status readout, test, and the configuration optimization of the AOCs. As the power supply, a sata power connector seems useful, because it is a widely used standard and directly delivers 3.3V supply voltage which is required for powering the AOC. For measurement and analysis of differential signals, test pads should be added for being able to probe them directly. Because of availability, usage of the Ventoux FPGA board is proposed for testing with three HDI6 connectors, each using 4x lanes for communication. Thus, a board structure using three 4x HDI6 merged into one 12x HDI6 turns out as a useful configuration. Figure 5-13 shows the suggested Opto-Converter prototype as block diagram. PCB design, production and assembly can be supported with the available H-Spice models for high-speed simulation. Addition-

ally, existing library parts within the Cadence environment were reused and experiences with the HDI6 connector within formerly developed high-speed designs were useful during design phase and in the assembly process.

This board enables various usage scenarios. One important scenario for CBM is radiation testing of HD-AOC. The HD-AOC is under full assembly control, so testing includes the possibility of exchanging parts among several suppliers for VCSELs, photodiodes, and their control chips. Additionally it serves as the Opto-Converter prototype for Hub testing and might be useful to aggregate Hubs within test beam times.

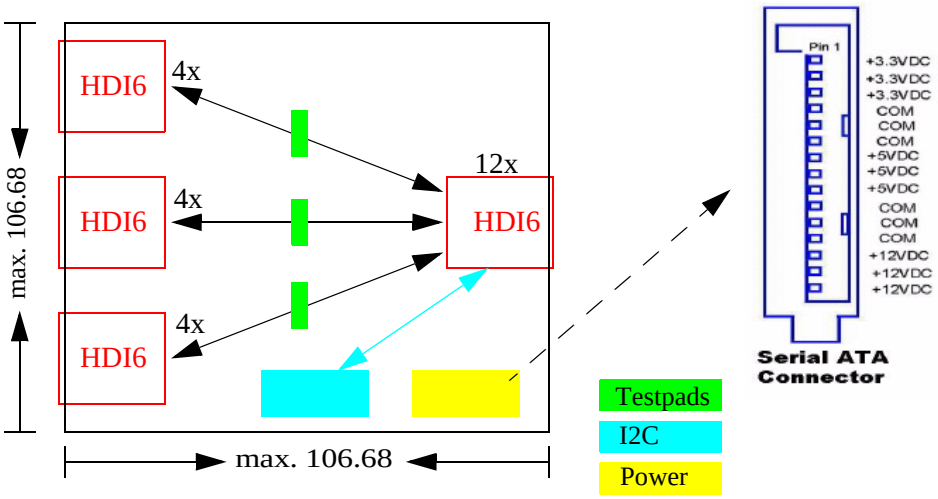


Figure 5-13: Opto-Board Structure

In figure 5-14, the planned test setup for radiation beam time tests is presented. There are two Opto prototype boards within the proposed test setup connected with one unidirectional 12x cable. This is ideal for radiation tests, because an identical setup can be used to analyze the send and receive part of the cable assembly without interference and only by repositioning. As platform for pattern generation and analysis, the Ventoux FPGA Board in combination with the HDLSP Samtec copper cables can be used. The Ventoux board is a FPGA

based board developed by CAG for lab testing. It uses three HDI6 connectors. Each 12x connection is only assembled with a 4x lane connection due to SERDES availability of the FPGA. For testing either two separate Ventouxs for send and receive, or one Ventoux using both HDI6 connector levels to separate send and receive are usable. FPGAs can be used here, because due to copper connections with Opto-Converters, they can be placed securely out of a beam. The Ventoux board can be used as a stand alone or within a standard PC, including drivers and API. However, the complete environment is available and like for used COTs components, the technology is well known and reliable. As new hardware built for the test setup, besides the Opto-Converter prototype, some specially assembled unidirectional AOCs with different VCSEL and photodiodes are required. Implementation of firmware for analysis can focus on analysis and pattern generation, because all basic modules already exist and they can be reused. In addition, software can be built directly on top of a well functioning API.

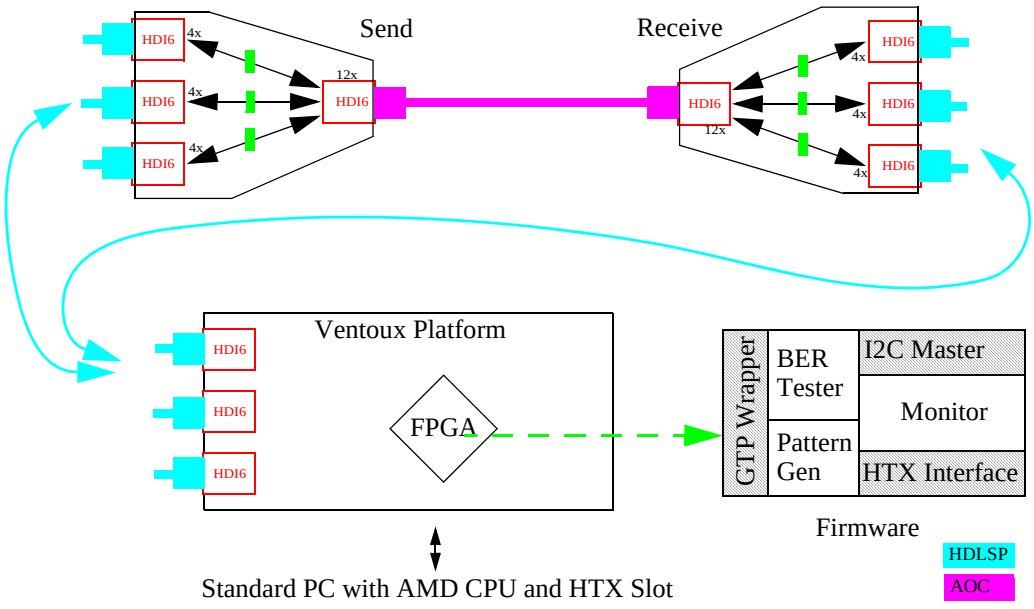


Figure 5-14: Radiation Test Setup

5.3.3 AOC Prototyping

Development of AOC prototypes using the high density HDI6 connector, based on the HDLSP Series Copper Cable design, have been done. The first with our concept designed AOC [94], using the special fiber attachment meachnism and a high density connector, already showed satisfying results.

PCB conception for AOC has multiple design possibilities. The first decision was concerned with the PCB connector. Here, either a thin PCB having an extra soldered connector can be used or a plated footprint on the PCB with ideal direct plugging thickness. Both solutions were designed and tested. Because AOC cables are rarely plugged, the direct PCB method was preferred. This solution causes fewer costs and requires no extra assembly steps. For the assembling of driver, transimpedance amplifier (TIA), photodiode arrays, and VCSEL arrays, there are two techniques. Bonding them onto the PCB or using a flip-chip technique for directly attaching them onto PCB pads. Due to signal integrity reasons, the flip-chip variant was chosen, but availability and costs for the arrays were too high. Thus, a wire bond variant has been developed. However, for future designs having higher speeds flip-chip is still of interest. The HDLSP connector has 12 send channels on the top and 12 receive channels on the bottom side. Signal integrity led to a design without vias. Thus, PCB was assembled on both sides, top for driver and VCSELs and bottom for TIA and photodiodes. Difficult assembly of bonding structure on both sides and the production yield for send and receive structures led to an innovative split design that has two thin PCBs, one for sending and one for receiving. In addition to easier assembly, yield increases by joining only working structures.

Figure 5-15 presents the latest so far developed version 3 HD-AOC design. On the left, the PCB uses a cavity for the chip is shown. This is the split version having one flat PCB for send and one for receive. These PCBs are then either screwed or soldered together. The I2C buses of both PCBs are connected via these screwed connections. This current PCB design works reliably. On the right of the figure, a PCB with a bonded chip and VCSEL of this pure wire bonded solution is depicted. Because of the cavity, all bonds between chip and

VCSEL array are almost straight and thereby optimal for signal quality. Below the arrays, a copper plane was added to assure assembling quality. The chip bonding structure has optimized power and ground rings and bondfinger structures to enable efficient automatic bonding. Finally, experiences from the various AOC designs led to a well-optimized PCB structure for the AOC fulfilling all required features.

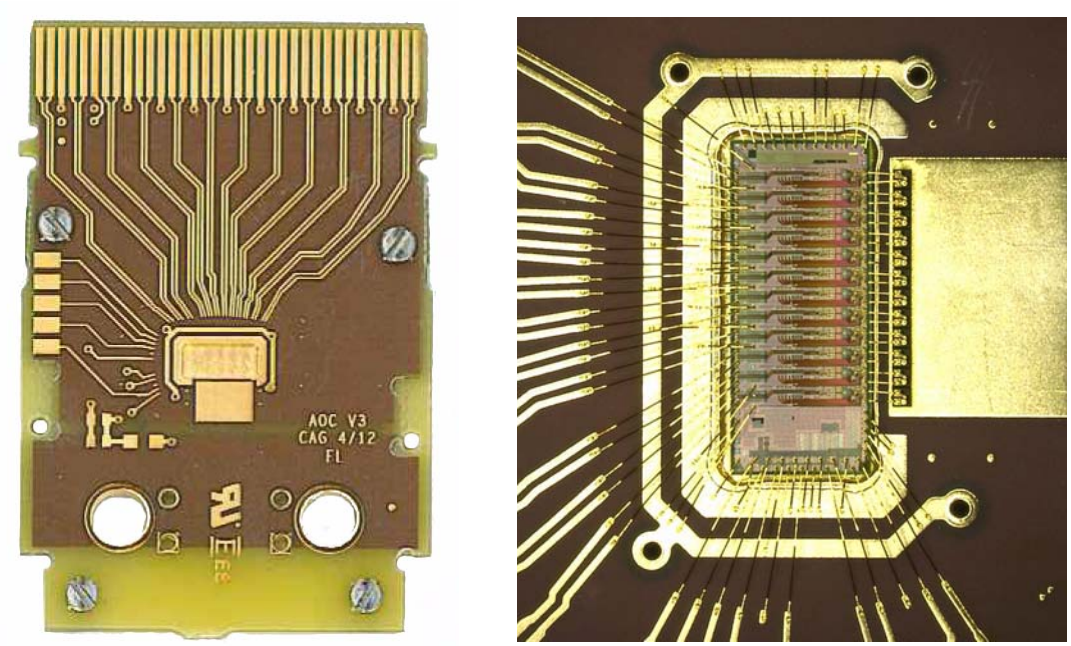


Figure 5-15: AOC PCB (left) and Bonded Structure (right)

The fiber attachment mechanism is based on work done in the context of a PhD thesis [95]. This mechanism is based on a direct replication process using a UV adhesive. This adhesive is stamped onto the assembled PCB and is UV cured. Therefore, an alignment is required to place the structure directly in µm precision on top of a VCSEL or photodiode. The resulting exactly placed structure contains fiber funnels, fiber guiding structures, and a mirror. The fibers are directly attached in a 90° angle and light is deflected by the mirror using total internal reflection. For keeping the fibers in the funnels, a glas substrate is placed on top of them. Once fibers are inserted, they are glued by a UV adhesive into the structure. Since the complete fiber front surface is engulfed with UV adhesive no polishing is required. This is an efficient repeatable concept of coupling fibers on top of active components. Due to min-

imum distance between fibers and components, a high efficiency for light coupling is achieved. In figure 5-16, an assembled AOC PCB is presented with stamped fiber attachment structure and inserted fibers. In the picture, a glas plate is visible. It was used to assure that all fibers stay in the insertion structure until they are glued. The concept and manual prototypes were successful and now development is focused on automation of stamping and fiber insertion. For the stamp alignment and stamping process, a half-automated environment is already running. Thus, optimizing the fiber insertion mechanism and increasing the yield are the main topics for future investigations.

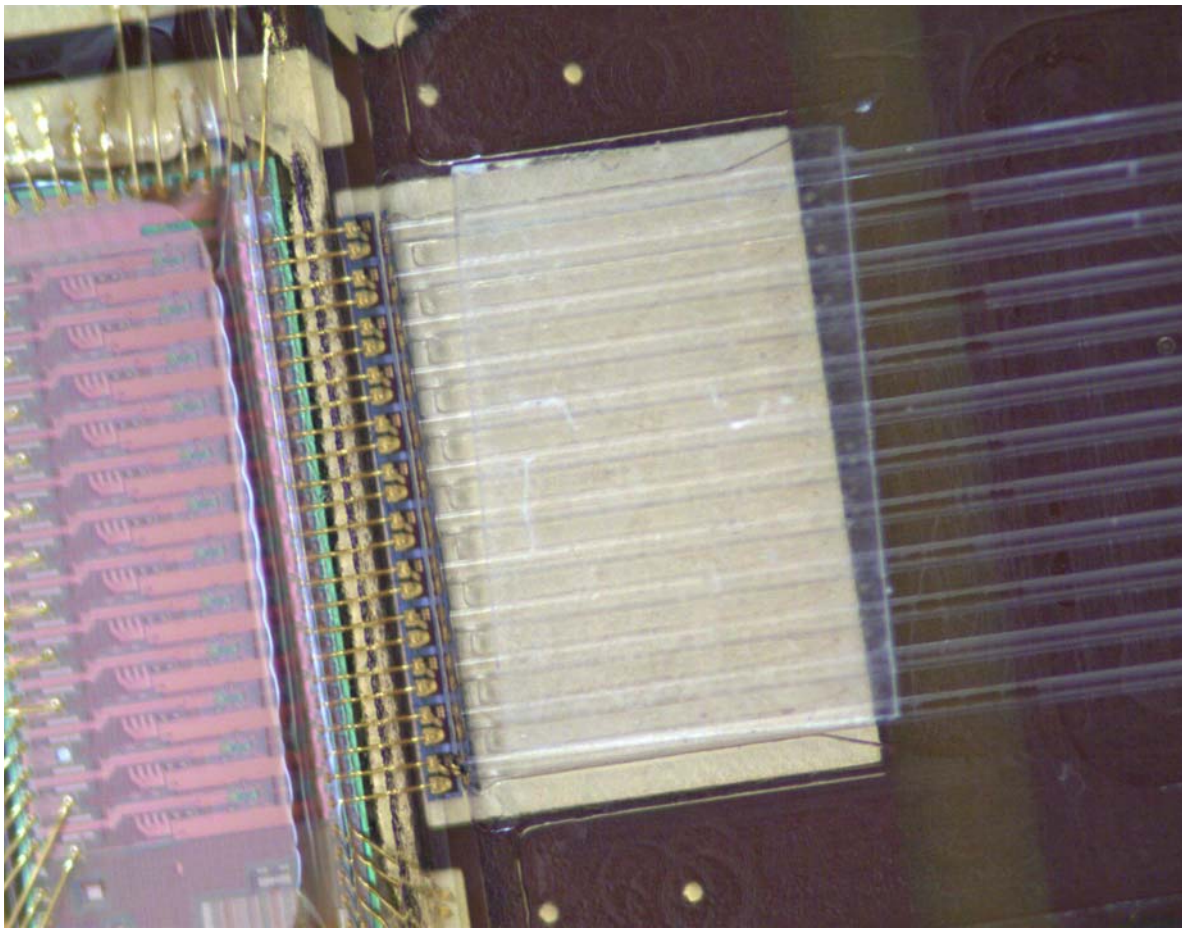


Figure 5-16: Fully Assembled AOC

5.3.4 AOC Prototype Measurement

Before the opto-converter prototype began, the HD-AOC V3 had to be tested and the AOC transmitter and receiver cable parts for opto-converter testing required qualification. Therefore, a Ventoux board starting with 4x lanes was connected to an AOC as pattern source. The target opt-converter speed of 5 Gb/s was selected for testing together with a pseudo-random bit stream (PRBS) pattern. First, an optical power meter was used for measurement, which resulted in 2.8 mW optical power for four lanes. The goal was to measure optical eye diagrams and electrical receive eye diagrams at the opto-converter prototype board. Then within this measurement setup, each of the 12x lanes can get a qualification.

The AOC prototypes have 12x ribbon fiber cables with MPO connectors and the optical probe has a FC single fiber connector. Thus, coupling structures had to be used for the transformation from MPO to FC. Figure 5-17 depicts the measurement setup with these coupling structures. The optical power meter was used to measure the coupling loss in between classifying the resulting optical eye diagram. As shown in the figure, three measure points were taken. The first one at the female MPO was directly attached to the AOC transmitter. Here, a value around 0.8 mW for a single lane was measured. Then, the optical signal was coupled into a MPO-to-LC transformer and the second measurement result of 0.7 mW was taken. Up to this point, 50 µm multimode fiber was used. A LC-to-SC 62.5 µm multimode fiber cable was attached. The SC connector showed a value around 0.5 mW. This was then attached to the oscilloscope and optical waveforms were measured.

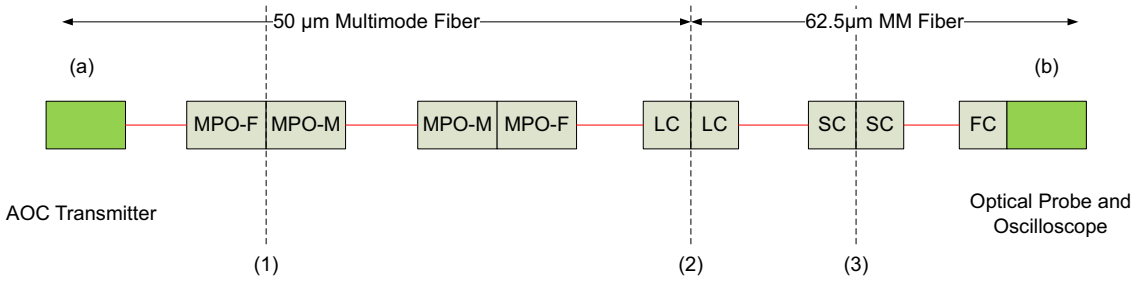


Figure 5-17: AOC Optical Measurement Setup

Figure 5-18 presents the obtained eye diagrams from this setup. On the left of the figure, the electrical eye diagram is shown at the Ventoux sender side with a scale of 500mV/div and 100ps/div. It is measured at point (a) in figure 5-17. This eye diagram is detectable but not perfect. There are two PCBs and one connector in between the FPGA and the measurement point. Additionally, the FPGA outputs run almost at full speed, which might be one of the reasons for the noise visible in the diagram. It can be assumed that there is the visible inter symbol interference, because of non-optimized configurations, e.g. pre-equalization can be refined. This leaves room for further optimization. The optical eye diagram, to the right of the figure, shows a detectable open eye with an optical power almost at 0.5 mW, which fits with the optical power tests. Its scale is 100µW/div and 50 ps/div and its measure point is shown as (b) in figure 5-17. It still shows inter symbol interference.

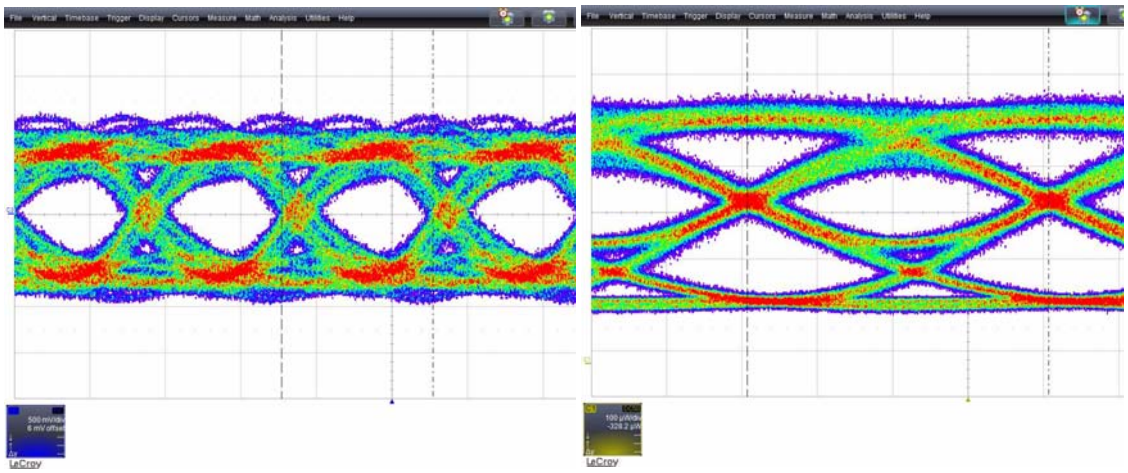


Figure 5-18: AOC Eye Diagram, Electrical (left) - Optical (right)

The measurement setup for electrical receive signals is depicted in figure 5-19. It shows the Ventoux as a pattern source and an AOC connected to the opto-converter board. In some tests, the AOC was replaced by a HDLSP series copper cable to have a reference. The electrical measurement point is a termination on a PCB plugged into the outgoing HDI6 connector of the opto-converter. Thus, between the AOC receiver and the measure point are three PCBs and two HDI6 connectors.

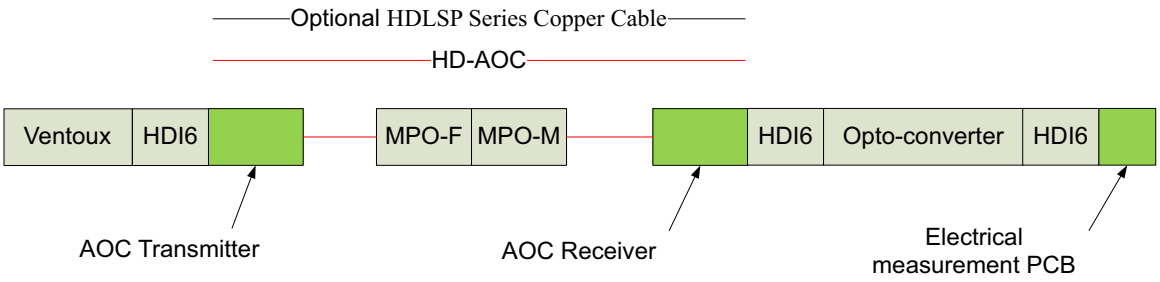


Figure 5-19: AOC Electrical Measurement Setup

At the receiver side, the opto-converter board was directly used to measure the electrical signal. In figure 5-20, these measurement results are presented with a scale of 200mV/div and 100ps/div. The left picture shows the received eye diagram of the AOC having a detectable but noisy eye. The picture to the right is done using a 0.7 meter copper cable showing even more inter symbol interference. The eyes are detectable and the best channels achieve an error rate of 10^{-13}. This depicts that the opto-converter prototype quality is good enough for testing, but for final solutions during design, more attention should be paid to signal integrity design and the used FR4 should be as short as possible. Using this measurement setup, the AOC test cables were qualified for opto-converter beam time tests.

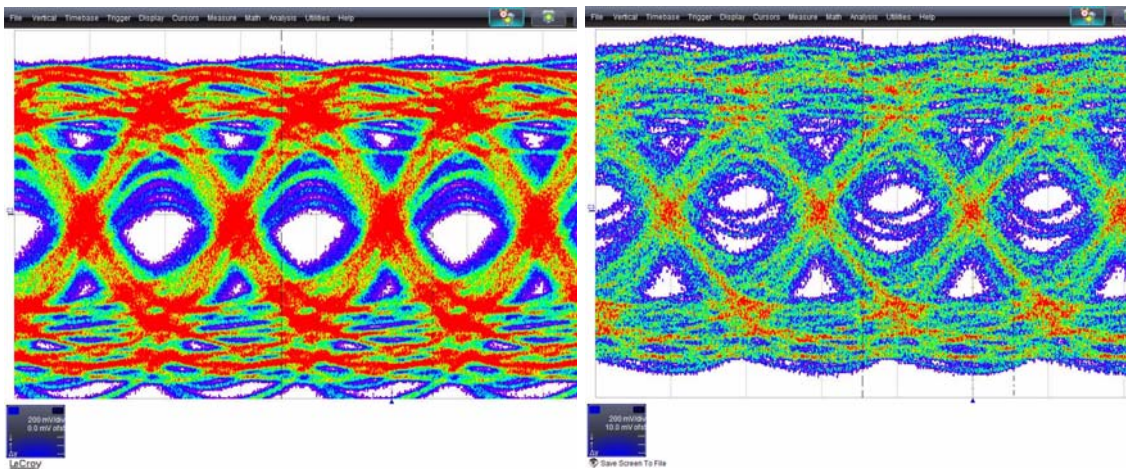


Figure 5-20: Opto-Converter Eye Measurement, AOC (left) - Copper Cable (right)

5.3.5 Opto-Converter Prototype Measurement

The Opto-Converter prototype has been designed, developed, and assembled in cooperation between GSI and CAG. Additionally, a firmware for the Ventoux FPGA was prepared for testing AOCs using PRBS patterns. Figure 5-21 shows the setup used during beam time AOC radiation tests and for laboratory testing. For beam time testing, unidirectional 12x cables were assembled using IPtronics [96] driver and TIA chips having ULM [97] photodiodes and three kinds of VCSELs attached. As 12x array VCSEL, parts from VI-Systems [98], GigOptix [99], and ULM Photonics were deployed. This enabled four different tests for the beam time, one test for a photodiode and three for the VCSELs located directly within the beam.

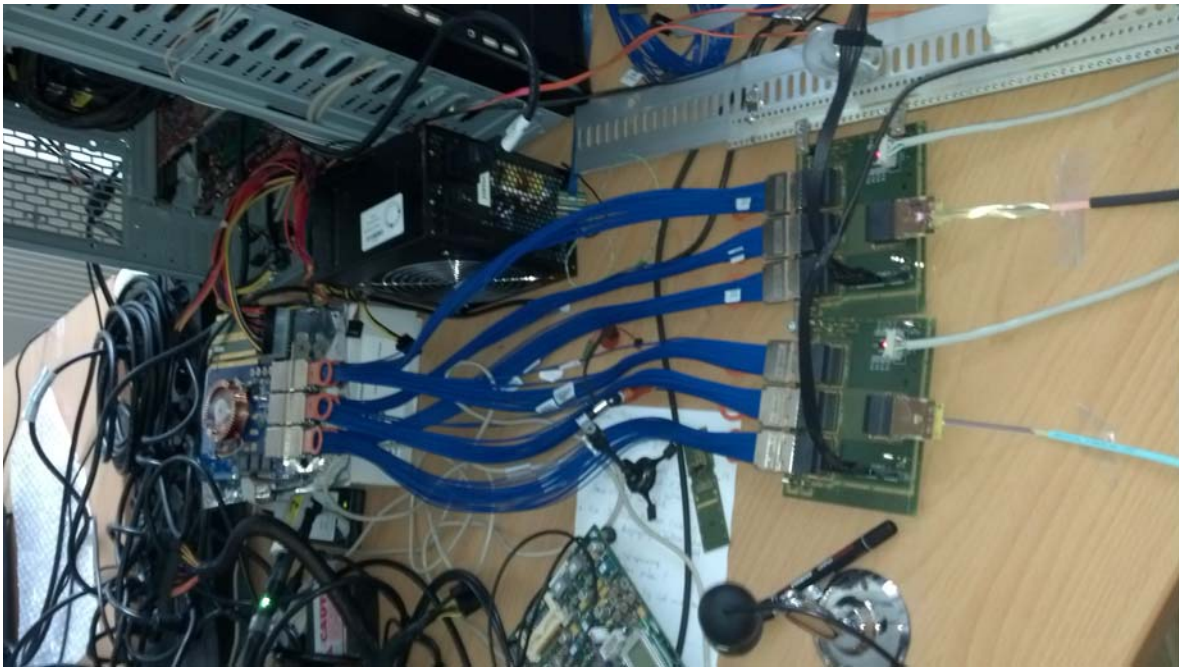


Figure 5-21: Lab Setup for Opto-converter Board Prototype and AOC Prototype Tests

First the results showed no significant increase of error rates compared to the laboratory tests. There were differences among the channels measured, but the best error rates were at least 10^{-13}. These differences are most likely due to tolerances of the coupling structure and fiber insertion process. An optimized hardware enhancing fiber insertion handling is in pro-

duction. It is assumed that after optimizing the insertion technique, all 12 channels will show reasonable error rates. However, after a successful test time of about 6 minutes being in the beam, either the transmitter or receiver stopped working. After a power cycle, the set-ups worked again. Because of all channels stopping simultaneously, it is assumed that a total dose effect did affect the IPtronics control chips. The beam time tests were successful, but further more detailed analysis is required.

5.4 HUB and Opto-converter System Integration

5.4.1 Technical Data

The HUB ASIC is planned in a 65nm technology version. Its currently planned structure supports 32 FEE CBMnet lanes to the front-end and a 4x lane to the next hierarchical stage. Adding the required LVDS pins, some service pins, the estimated number of SERDES pins, and the power-ground pins, the resulting ASIC requires a 19 x 19 bump array. This 361 bumps in total lead to an approximate ASIC size of 3420 x 3420 µm using 180 µm pitch. Its package will be around 19 x 19, which has about 361pins and a size of 2 cm. A 1.0 V or 1.2 V core voltage depending on LP or GP process and a 1.8 V I/O voltage are required. Between 2 - 4 routing layers are required to fanout the connections. The power dissipation is approximately around 5 - 7 W. LVDS connections between detector ASICs and HUBs, considering the 500 Mb/s data lanes and the 250 DDR MHz clock lane, should not be longer than 2 meters. The fast 4x lanes attached to the Opto-converter supporting a minimum data rate of 5 Gb/s. The CML output drivers are planned to be capable of driving approximately 10 cm FR4, a connector and the attached Opto-converter connection. Due to equalization, there is a 100 cm twinax cable as a connection to the Opto-converter, but the maximum length is under evaluation.

The Opto-converter, as a component for electrical-optical conversion for the diff CML output signal of the HUB, needs to be compact and has to combine at least three of the above described hub devices. Because ribbon fiber cables will be used to reach the required density, which leads to the direct use of 12 x connectors to reduce the number of components. The HDI6, as the most compact connector for 12x as COT available, has a width of 23.9 mm a height of 13.4 mm and a depth of 19.8 mm without a shell. For the connector fanout, a multilayer stackup should be used supporting four breakout layers for an optimal design regarding the HyperTransport Node Connector 1.0 Specification. The counterpart has a width of 29.9 mm, a heigth of 6.1 mm, and a length of 27 mm before it becomes a 12x rib-

bon fiber. There are two power supply pins for 3.3 V in each 12x connector defined. For currently used COTs parts, the AOC using this connector has a power dissipation of around 1100 mW. Typical is 500 mW for the VCSEL driver, and 600 mW for the TIA.

Other COTs components could also be used after radiation testing to deliver the capability for the electrical to optical conversion, but there has been no final decision yet. Due to the fact that all of the above used components are the most compact ones for a reasonable price in the market, these components are used as a working assumption in the following analyzes.

5.4.2 DAQ Read-out Structure Scenarios

In the case of the STS, front-end detector ASICs are using 4x channels for data transmission to get the required 2 Gb/s data bandwidth per chip. It is planned to integrate eight of these ASICs onto a FEB. This would exactly deliver 32 lanes for one HUB ASIC, which fits a suggested form factor estimated in a first design analysis. Thus, it probably makes sense to

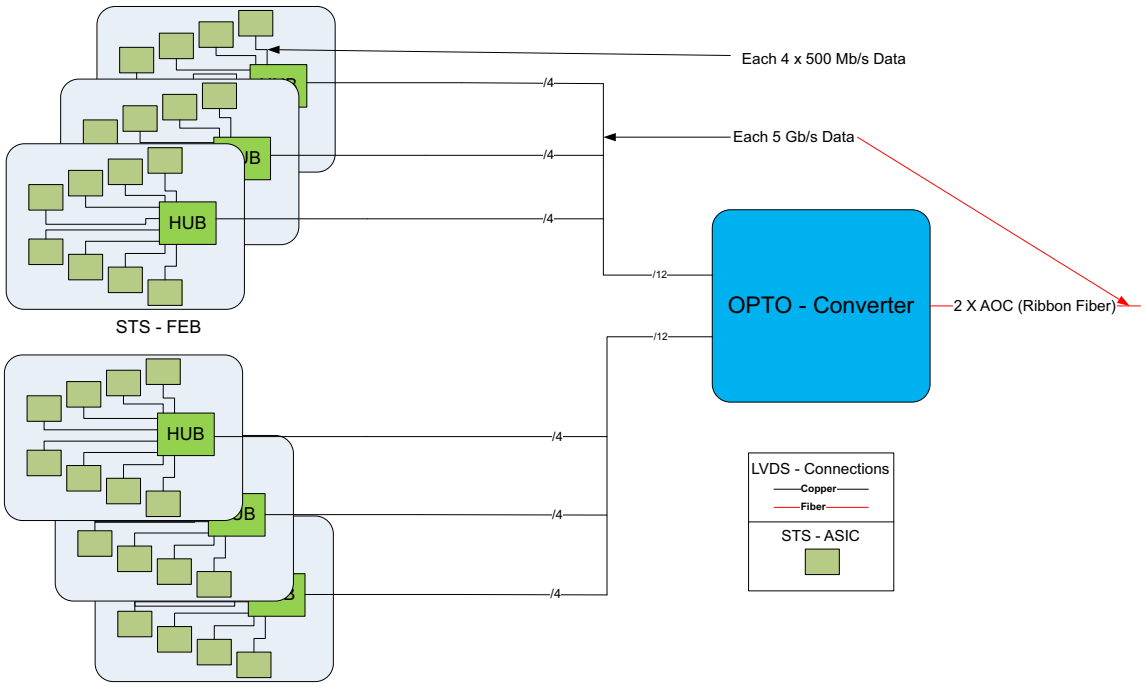


Figure 5-22: STS Readout Chain using HUBs (Standard Structure)

assemble a HUB ASIC directly onto the FEB. An advantage of this approach is a controlled relatively short connection from the detector ASICs to the HUB ASIC. The Opto-converter may then be placed into the same assembly box as the FEBs in an area with the lowest possible radiation to keep the distance as short as possible, having the 5 Gb/s data stream in mind. Figure 5-22 shows this default setup having an optimal data concentration of three FEBs per HUB ASIC and three HUB ASICs for each AOC. The proposed double stacked connector leads to an Opto-converter board that provides at least two AOCs. In case of a reduced data generation, because of placement and detector specific reasons, it makes sense to use less then four lanes connecting a HUB ASIC to a DPB in the next hierarchy level. Due to its crossbar, designed with fault tolerance and reconfiguration kept in mind, it is possible to decrease bandwidth by using, for example, only two lanes towards a DPB. A possible configuration is presented in figure 5-23 attaching up to six HUBs onto one 12x connector from an Opto-Converter. This example shows the flexibility of the fully free configurable approach followed within the HUB HW concepts.

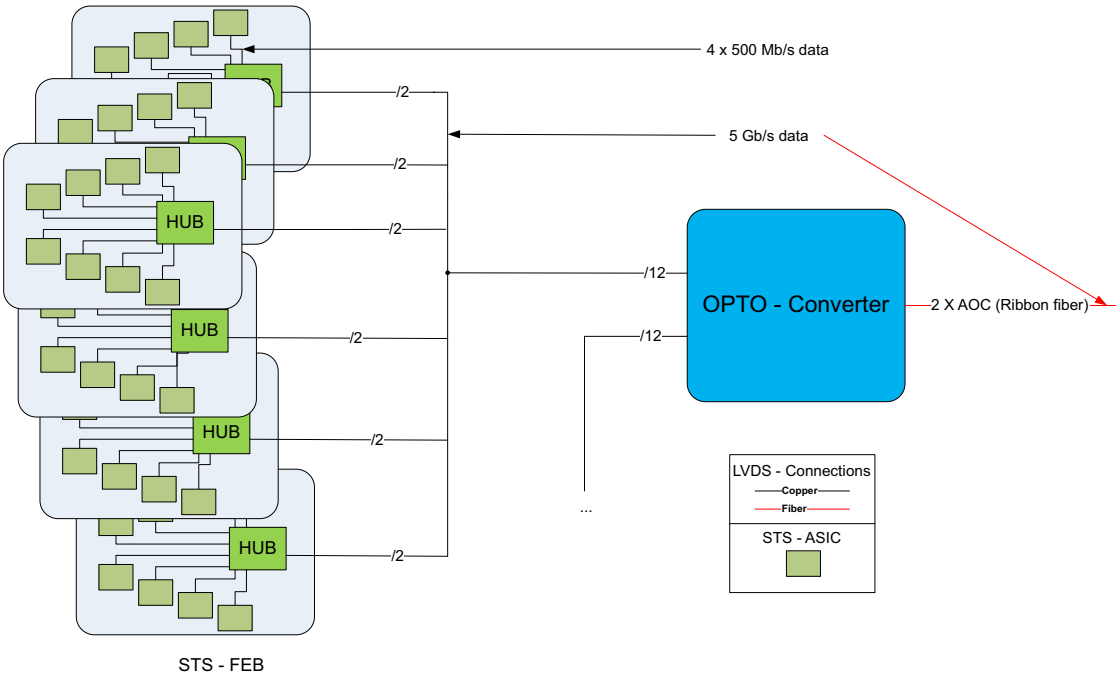


Figure 5-23: STS Readout Chain using HUBs that have reduced Data Generation

The TRD setup using SPADIC ASICs as FEE detector chips requires two data links from each device to provide a 1 Gb/s bandwidth. The detector build-up scenario includes modules using two different FEB types. One is a FEB containing five SPADICs and the other contains four. A TRD module is readout by one identical FEB type and Opto converters supply only one FEB type at a time. Figure 5-24 shows a feasible standard setup with three FEBs connected to one HUB, using only three high-speed lanes with 5 Gb/s at the top part and a standard setup that uses four high-speed lanes at the bottom. With up to 10 lanes per FEB delivering 500 Mb/s, each of the fast lanes towards the DPB can be filled up to 100%. Thus, theoretically three links can be enough to provide a sustainable bandwidth to support three FEBs. In case of errors, causing a retransmission or some control messages interacting between HUB and DPB, this will lead to backpressure and stall the data acquisition. Thus for this scenario there must be a time separation between collecting data and controlling devices.

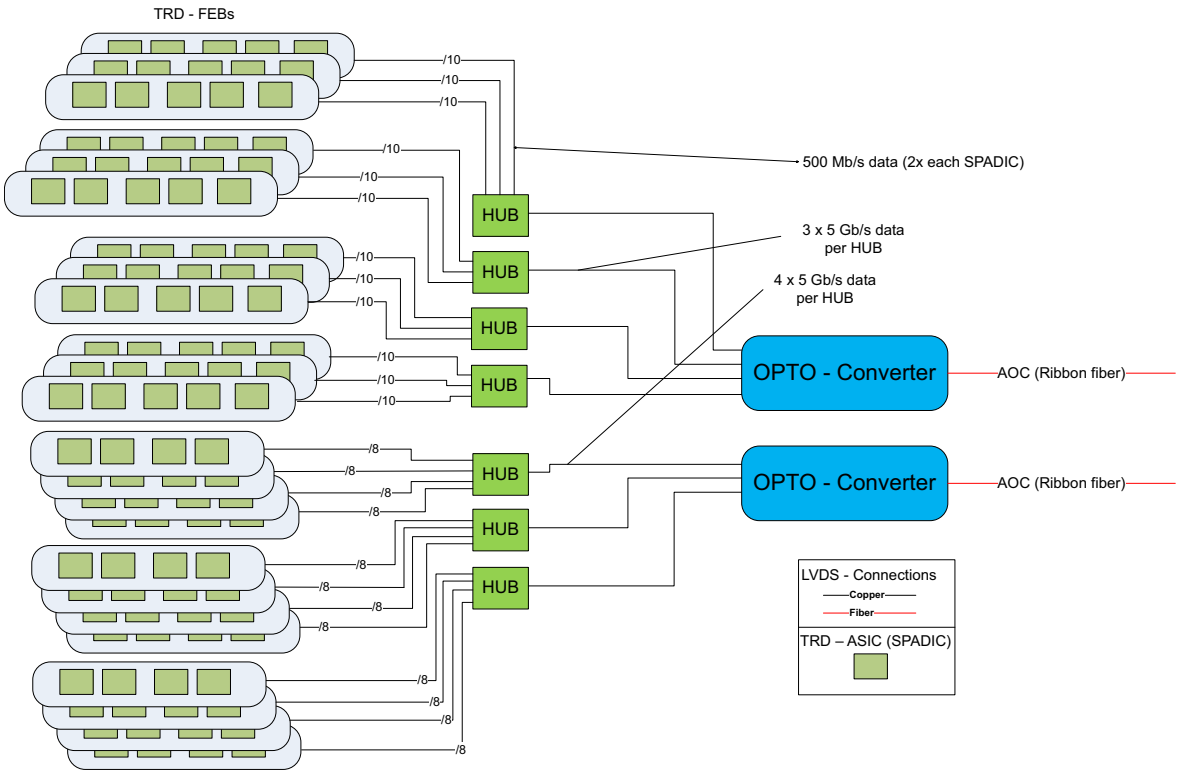


Figure 5-24: TRD Readout Chain for Special Usage and Standard Setups

This will only work with a non-continuous data stream. However, the standard read-out chain should use the complete 4x link to stay with the higher flexibility and fault tolerance features.

A different solution would be to adapt the planned HUB ASIC design in a way to support both main read-out scenarios, but with a slightly different concept, which would remove a lot of the flexibility provided by the suggested design. When the hub provides 20 x 2x connections (80 LVDS pairs) instead of 32 x 1x (96 LVDS pairs), then the full 4x link could theoretically be used to 100%. Of course, the stall problematic will not change, so the system has to be used in a special way. For the native 1x connection, there would always be a waste of one LVDS pair and only 20 x 1x connects would be possible. The 4x scenario of the STS stays identical. Figure 5-25 depicts a setup using the Type2 HUB. Here, either four 8x or four 10x connections could be used. Fault tolerance and control message support remains the main problem in this scenario. After considering the different scenarios, it is clear that a suggested structure of 32 channels and a 4x connection to the DPB is a useful solution. It provides flexibility, fault tolerance, and controllability.

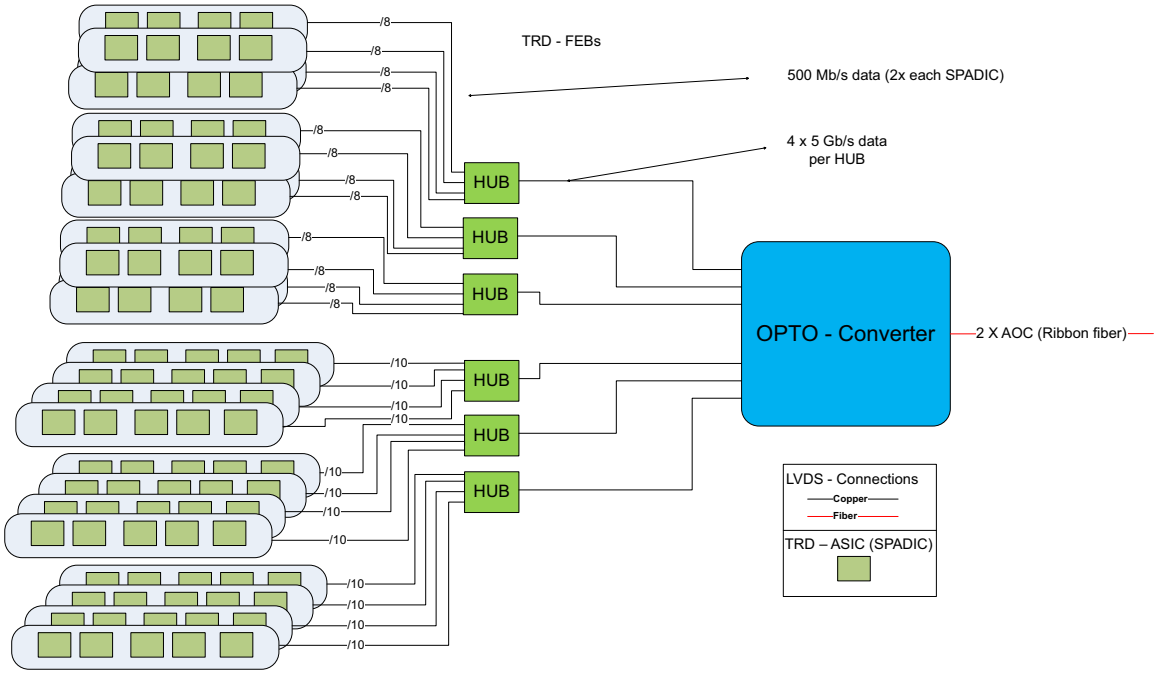


Figure 5-25: TRD Read-out using the Type2 HUB Design

5.5 Conclusion

This chapter described the analysis and the prototyping for the HUB ASIC. It started with the requirements and analyzed the data flow strategies. A concept was presented to bundle multiple 500Mbits/s FEBs together to sustain higher bandwidth connections like 5Gb/s. The HUB ASIC concept includes the complete handling of FEB frontend data flow, the slow control and the synchronization mechanisms. An analysis of possible packages and dimensions for HUB ASIC production was done. In addition, a solution for the electrical to optical conversion was presented using the HD-AOC and an opto-converter prototype. Finally, build up structures for detector readout were analyzed to assure usability. Successful prototyping has been done for the AOC, the opto-converter, and the HUB ASIC itself, using a SP605. The feasibility of the HUB ASIC solution was shown.

Chapter 6

Networks based on DLM Synchronization

In previous chapters, the innovative DLM based network synchronization was presented. The focus of this chapter is on transferring this methodology onto high performance computing interconnection networks. These networks profit from efficient synchronization mechanisms and deterministic latency link behavior, for example, by running synchronous in order to be able to reduce communication latency. In addition to the CBM like DLM operation, other sub variants using synchronization messages are described. They are based on similar ideas, but they work without priority request insertion hardware support.

6.1 Overview

This chapter intends to show that the invented synchronization mechanism using DLMs for synchronization is useful for high performance computing (HPC). The deterministic latency scheme requires the network to run in a fully synchronous mode. Upon first impression, this seems to be challenging for a potentially multi-thousand node HPC cluster systems, but does lead to several advantages required in future systems. Besides enabling DLMs as an efficient mechanism for synchronization, deterministic latency based systems allow bypassing of all synchronization buffers at least within network links. Thus, communication latency is reduced by a few clock cycles for each node. This seems useful in the latency sensitive, more and more growing HPC clusters, and might become a special feature. Nevertheless, it remains challenging especially due to guarantees provided by clock distribution.

After adjustment and initialization of a deterministic latency based synchronization system, DLMs itself can be used for counter synchronization like in CBM. This seems primarily useful for sensor and detector systems, but the synchronous arrival of DLMs can trigger a start and stop functionality. Even more, it can trigger an action at the exact same time all over the system. This might cause command lists or a special feature to be initiated. Supplemented with a multicast like usage of DLMs suggested for CBM, which only addresses nodes of special groups, DLMs become a powerful utility for HPC computing clusters.

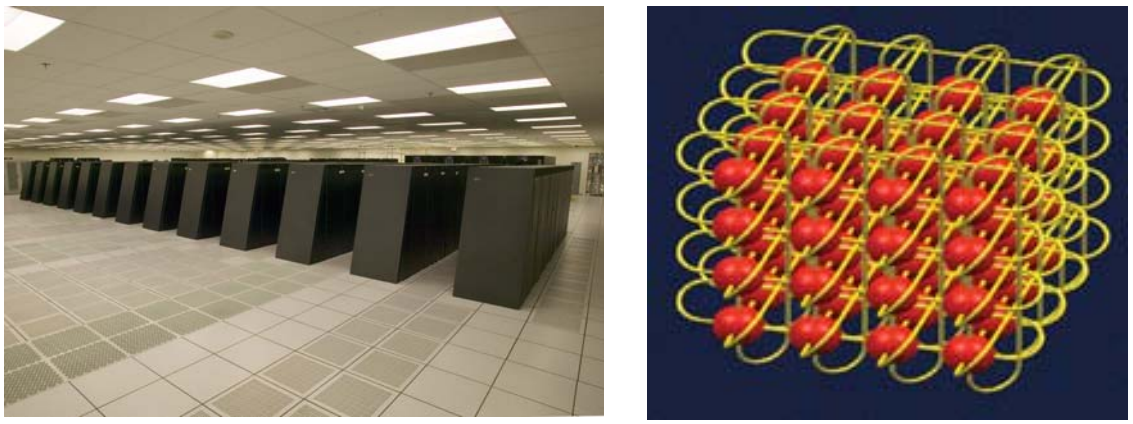


Figure 6-1: BlueGene/L (left) and 3D-Torus (right) [101]

The synchronization has to be transferred and analyzed onto HPC systems. Therefore, examining parallel and scalable architectures [100] in literature a 3D torus as a typical use case scenario appears reasonable for evaluating DLM usage for HPC. A 3-D torus is a direct interconnection network. Each node in this network has six links, two for each dimension in plus and minus direction. Its structure is a 3D grid having wraparounds at the edges. It has typically a $n \times n$ structure. The wraparounds as special ability half their diameter compared to a 3-D grid to $2\lfloor n/2 \rfloor$. It is a popular structure in HPC, because the wraparounds reduce the average number of hops within a network for reaching a goal significantly with only a few additional links. Figure 6-1 shows the BlueGene/L [101] as an example system and the 3D-Torus topology used in its interconnection network. In the following subchapters, the usefulness of the new synchronization mechanism concepts will be shown for 3-D torus networks.

6.2 Synchronization concepts

6.2.1 Priority request insertion

Before using a DLM based synchronization on a 3D-torus HPC network system, there are two basic abilities to be guaranteed. The first required ability is a reliable system wide clock distribution. This could be done by a separate clock distribution tree or network, but using recovered link clocks as shown in CBM is preferable. Therefore, a clock master has to be defined. A master can be anywhere in the network, but has to provide a high quality clock. The master clock quality directly affects synchronization performance, because it should be guaranteed that no bit slip occurs. At least they must be firmly avoided. In case of a bit slip, it must be detected and a readjustment has to be performed. The second ability is deterministic link behavior at least concerning synchronization messages. The stable system reference clock is used to clock the synchronization message processing logic. This logic must be designed with a deterministic behavior that results in the identical clock cycle processing time for each DLM received. The mechanism to guarantee this deterministic behavior of sending DLMs is priority request insertion (PRI). PRI besides the identical processing time

assures that DLMs are directly in a fixed time inserted into the send channel. This can even be in the middle of messages. On a receiver, they are filtered out of the stream and split messages are rejoined. This mechanism requires additional hardware, but provides deterministic latency.

On this basis of a synchronous network with integrated clock distribution, the suggested unified communication scheme can be used. It provides all communication classes including DLMs enabling synchronous actions and thereby collective operations. Of course, for the different communication types and tasks separate network structures can be used. However, emerging development shows that a unified connection scheme is advantageous and a target for industry, especially valuable since it concerns cost and complexity. In the Blue Gene/L [102] and Blue Gene/P [103], [104] architectures, there are multiple networks providing these functionalities. They include, besides some other networks, a separate network collective for barrier & interrupts and a clock distribution network. With a unified concept, the Blue Gene/Q [105], connected as 5D-torus, is operated. It uses unified connections for most of the earlier networks used, but it still has a separate clock distribution.

In case of an error, e.g. a clock master failure, another node can take over and become the clock master. The latency in the system defines the time a take over requires and depending on local clock quality this may work without a bit slip and synchronization may be preserved. Otherwise, synchronization has to be recalibrated. Alternatively, with high effort, a backup system could be used, which can take over and guarantee persistent synchronization. Another clock master, that permanently checks and calibrates it to be synchronous and ready for take over, could provide this system.

These preconditions enable the mapping of a synchronization mechanism onto a network, e.g. a 3-D torus. Similar to a clock master, as synchronization master (sync root) somewhere in the network a node can be selected. Starting at the sync root, a synchronization system can be established by initialization routines measuring and adjusting network links. All the required information must be stored. Therefore, either extra RAM structures or RFs can be used. After calibration, the synchronization hierarchy is mapped as a tree in a kind of 3D

wave structure onto the network. Figure 6-2 depicts this wave based synchronization structure in a simplified 2D view. The sync root is defined as synchronization level zero. Every further link hop defines the next synchronization level. Thus, synchronization messages spread like a 3D wave through the network. The amount of synchronization levels is equivalent to the diameter of the 3D-torus. There are two kinds of usage for synchronization in this system. The first one is to fully synchronize all nodes. Therefore, the precalibration assured that all nodes have the clock cycle values stored for triggering a synchronous action exactly at the time a DLM arrives at the highest synchronization level. Thus, the time a synchronous event takes is equivalent to communication latency from the master to the most distant node. The second possible synchronization scheme is to have synchronous groups. A group could be one or more synchronization levels. This gives the advantage of faster triggering, but depending on the running applications, post processing for time corrections has to be done.

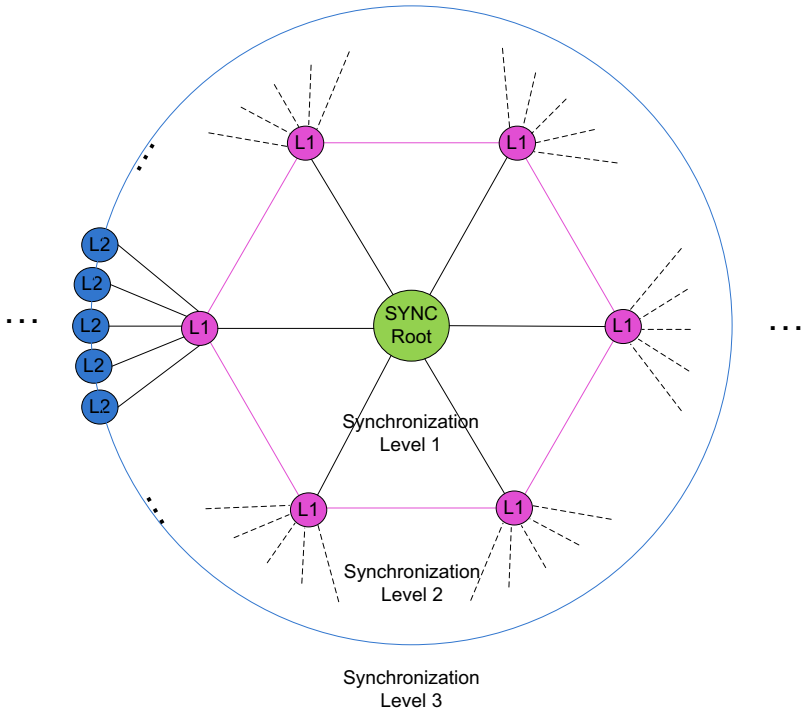


Figure 6-2: Synchronization Wave Scheme

The full synchronization allows the usage of DLMs for running a global counter scheme, thus synchronizing local counters with DLMs similar to CBM. Another important application area for DLMs is gang scheduling [106]. Gang scheduling as scheduling policy requires a fine-grain synchronization to be efficient. For gang scheduling, interacting threads running in parallel on a compute cluster use a busy waiting scheme. This avoids context switches and assures that participating nodes are not in sleeping mode and thereby depending on the application, the performance increases. Thus, gang scheduling can benefit from efficient fine-grain DLM synchronization and the synchronous actions started. All processes can be started at the same point time, which supports busy wait implementations. Furthermore, it is possible to schedule or reschedule all processes simultaneously on a cluster or a partition.

This system approach requires additional hardware for the PRI and storage for calculated values and adjustment parameters, but it is of interest for HPC because of a mechanism delivered to reach synchronization with the minimum possible amount of synchronization latency. It only depends on interconnection network quality constraints. Some sub variants requiring less hardware are described in the following sub sections. These variants are only discussed for a full system synchronization scheme.

6.2.2 Fixed Slot insertion

Based on identical system requirements, fixed slot insertion is an alternative to the PRI DLM insertion mechanism. This solution needs less hardware resources and is easier to implement. It simply works by having fixed slots in time where DLMs can be inserted into the data stream. Thus, they can not be integrated within other messages. These fixed slots waste bandwidth and decrease link utilization, because long messages always have to wait until they do not block a DLM slot. In addition, the deterministic latency for each link increases, which affects the synchronization latency and thereby, a time lag of full synchronous system actions.

6.2.3 Fixed max delay insertion

A sub variant of fixed slot insertion is fixed delay insertion. This mechanism creates its insertion slots as soon as the DLMs arrive instead of having reserved slots. The availability of a slot is thereby guaranteed by reserving it in the future after a maximum packet length. Thus, link utilization and bandwidth are less affected than for fixed slot insertion, but the link latency maximizes to the standard value plus the maximum packet length. However, it is a disadvantage for the network, because maximum delays accumulate in regards to system synchronization latency.

6.2.4 Balance counter concept

An interesting alternative to PRI is the balance counter concept. DLMs have to be extended by a counter in order to enable this mechanism. The counter may work with minus and plus values. The final triggered DLM action is done slightly in the future after theoretical DLM minimum system synchronization latency for creating a margin to balance insertion inaccuracies. Whenever a DLM is inserted into a link and has to wait, this wait time then accumulates onto the counter. At the end, the counter value is subtracted from the margin which resulted in a wait time to synchronize. When allowing minus values, each link has a small amount of increased latency. This may decrease the counter on successful early insertion. The plus and minus counter concepts are equivalent, but the use of minus values reduces the required counter bits.

This Balance counter concept has the disadvantage that in a worst-case scenario a DLM arrives and its counter value is too high to stay in the allowed margin. Therefore, if a counter has already reached its allowed maximum, one solution is to terminate currently running messages, insert the DLM, and then resend the message. Thus, synchronization functionality can be preserved, but it affects utilization.

However, this mechanism has, on average, a latency better than with fixed slot insertion and fixed delay insertion. It does not affect the bandwidth too much, but its disadvantage is higher complexity and implementation effort.

6.2.5 Conclusion

The new message based synchronization mechanisms using DLMs can be mapped onto a 3D-torus structures representing a HPC use case scenario. PRI allows reaching theoretical minimum values for system synchronization latency. This first analysis concludes that it is worth further effort to implement a prototype and analyze advantages of triggering system wide synchronous actions. The sub variants without PRI, which are easier to implement, may be used for first prototyping, but PRI is a superior algorithm and its usage should be the goal. Nodes with the largest distance to a synchronization root define minimum synchronization pulse time granularity. Thus, the primary optimization target for a network system evaluating DLM synchronization is the reduction of communication latency for each link.

Chapter 7

Conclusion and Outlook

This chapter concludes the dissertation with a discussion of the goals achieved. It summarizes the research work done in the context of the CBM experiment focusing on the message based synchronization mechanism, the CBMnet protocol, ASIC development, and the CBM DAQ network solution. Finally, an outlook on plans and future is given including possible activities within the CBM experiment.

7.1 Conclusion

This work started with analyzing requirements and possible features for the DAQ readout network for the CBM experiment. Therefore, an initial analysis of the state of the art and possible solutions available for direct usage within CBM was discussed. However, there was no appropriate candidate delivering required features and performance. This led to the decision of creating a specialized CBM solution, because otherwise the required performance seemed to be out of reach. However, even during ongoing research and development, steady results and development of other groups or companies were reviewed.

In the context of research work done to find reasonable solutions for the CBM DAQ network synchronization, a new synchronization mechanism based on synchronization messages was created. This synchronization mechanism is usable within different topology structures and is directly integrated into communication traffic streams, avoiding separate synchronization network structures. The innovative message based synchronization approach has not only shown its usefulness for CBM, it seems also beneficial for applications in the area of HPC.

Furthermore, for the CBM DAQ network, a unified interconnection network with precise time synchronization was created. Therefore, the CBMnet protocol implementing all required features for this unified interconnection network was designed and developed. It can be used either on copper connections or for fiber communication. At the front-end for possible copper connections, some of the features may run on separate nets. But for standard communication, unified communication on bidirectional links is used. This enables running a DAQ readout network without protocol conversions. Therefore, compatible implementations for different FPGAs and ASICs were implemented. In addition to standard CBMnet protocol modules, for ASIC developers, a set of generic modules is supplied to enhance future ASIC developments. This generic module conception decreases ASIC development time and reduces the risk through the use of hardware proven IP. A CBM DAQ network interconnection supports four types of communication as clock distribution,

slow control messages, data messages, and precise synchronization. The clock distribution is done using clock recovery mechanisms and is supported by jitter cleaning strategies. Thus, it is precise enough, around 5ps RMS jitter, to avoid bit slips and in conjunction with the deterministic latency hardware implementation, it guarantees a basis for synchronization. The CBMnet protocol handles all other three communication types with its traffic classes: DCM, DTM, and DLM. This solution provides all special requirement of CBM. It supplies hardware with reusable modules and enables flexible build-up variants to support different DAQ structures. The protocol includes single bit error correction mechanisms to complement standard methods used as triple redundancy for delivering radiation tolerance. A continuous data stream is produced by the self-triggered front-end electronics. CBMnet is designed to ease data aggregation and rate conversion. This enables the handling of high data bandwidth of up to several TB/s generated in the CBM DAQ network. Due to optimized implementation and advantageous concepts, compact hardware was created and the limited space restrictions for CBM can be met. The DLM based CBMnet synchronization mechanism delivers synchronization on a link bit clock level and guarantees precise timing for CBM.

In addition, the design of a first HUB ASIC version and its FPGA prototyping has been completed. It delivers features such as data aggregation, rate conversion, synchronization, and control of front-end electronics (FEE) in the near detector region. This ASIC can handle all traffic classes such as DLM, DCM, and DTM. Furthermore, it controls numerous with electrical links attached FEE boards. It combines FEE detector ASIC data streams at low speed. Afterwards, it sends the data with increased speed to the next hierarchy level. An electrical to optical conversion is required to work close in conjunction with the HUB ASIC. Ideal COTs components have been selected and together with an innovative fiber attachment principle, an AOC based solution was created. This solution was successfully prototyped. It delivers high density, helps in regards to potential separation, and provides the communication distance capability. Thus, a solution with sufficient bandwidth and compact hardware setups can be provided for CBM.

The main goal of creating a new synchronization mechanism has been reached. The invented DLM based synchronization mechanism was successfully tested. Additionally, a unified precise synchronization protocol has been created for the CBM experiment. Numerous occurring problems have been solved. Various firmware and hardware implementation have been accomplished. Specific dense solutions enabling efficient data aggregation and readout structures have been successfully designed and prototyped. In the end, not only concepts and developments have been done within this research, but furthermore, an innovatively complete readout network solution fulfilling all CBM requirements was designed and created. This solution is capable for use in the final CBM experiment detector setup allowing the planned experiment performance to unfold.

7.2 Outlook

The construction work for FAIR began this past summer and the groundwork is currently ongoing. Status of CBM shows progress for all parts and much prototyping has been done. Current plans for CBM and FAIR define it as partially running operational with SIS100 in 2018. In this time frame, a lot of further hardware revisions for all parts are required and numerous test beam times will be done. The well-proven CBMnet optical readout chain will be used in upcoming beam times for testing and improving various hardware and software components. Thus, test beam times have to be supported. Integraton of CBMnet generic modules into various additional FPGA boards, as e.g. the ROC3, and support for numerous FEE detector ASIC developments is required.

Provided that all administrative circumstances permit, the HUB ASIC presented in this dissertation will be designed and developed. The HUB ASIC is planned as a 65nm ASIC and will show a significant impact on quality and potential for the CBM DAQ readout network. The basic design concepts are suggested and successful prototyping has been done, but there is still a long way to go for a complex and challenging ASIC like this. However, the chip development can now be started, but besides the digital design for the high-speed SERDES development, an external full custom design IP is required. For achieving an integra-

tion of this SERDES, a very promising collaboration has been started with the Advanced VLSI Design Laboratory at the Indian Institute of Technology Kharagpur. All preconditions are set up to enable a successful development of the HUB ASIC in the future.

An additional task is DPB integration for CBMnet modules. However, this is not the only task in this context, because DCS components are integrated into the DPB and must be supported. Finally, a control and synchronization system, concerning the detector attached via CBMnet, for the ECS in the service building region of CBM needs to be designed and developed. This control system is completely separated from the environment control system where COTs components can be used. All these tasks are in strong correlation to the presented work.

Last but not least, the new innovative synchronization mechanism using DLMs needs to be implemented onto a HPC 3D-torus for further analysis. It has the potential to show significant impact for future HPC applications.

A Abbreviations

ABB	Active Buffer Board
ACK	Acknowledgement
ADC	Analog Digital Converter
AOC	Active Optical Cable
ASIC	Application-Specific Integrated Circuit
CAG	Computer Architecture Group
CBM	Compressed Baryonic Matter
CBMnet	CBM Network Protocol
CDR	Clock Data Recovery
CERN	Conseil Européen pour la Recherche Nucléaire
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
DAQ	Data Aquisition
DABC	Data Acquisition Backbone Core
DCB	Data Combiner Board
DCM	Detector Control Message
DCS	Detector Control System
DDR	Double Data Rate
DLM	Deterministic Latency Message

DPB	Data Processing Board
DTM	Data Transport Message
EDA	Electronic Design Automation
ECC	Error-Correcting Code
ECS	Experiment Control System
EOP	End of Packet
FAIR	Facility for Antiproton and Ion Research
FEC	Forward Error Correction
FIFO	First In First Out
FLES	First Level Event Selector
FLIB	FLes Interface Board
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GLP	Generic Link Protocol
GSI	GSI Helmholtzzentrum für Schwerionenforschung GmbH
HADES	High Acceptance Di-Electron Spectrometer
HDL	Hardware Description Language
HPC	High Performance Computing
IP	Intellectual Property
LHC	Large Hadron Collider
MGT	Multi-Gigabit Transceivers
NACK	Non-Acknowledgement
NTP	Network Time Protocol
OSI	Open System Interconnect
PCB	Printed Circuit Board

RMS	Root Mean Square
PTP	Precision Time Protocol
PRI	Priority Request Insertion
RF	Register File
SDH	Synchronous Digital Hierarchy
SEE	Single Event Effect
SEL	Single Event Latchup
SERDES	Serializer-Deserializer
SET	Single Event Transient
SEU	Single Event Upset
SFP	Small Form-factor Pluggable
SONET	Synchronous Optical Network
SOP	Start of Packet
SOSC	Start of Slow Control
SPADIC	Self-triggered Pulse Amplification and Digitization asIC
PRBS	Pseudo-Random Bit Stream
STS	Silicon Tracking System
STSXYTER	Silicon Tracking System XYTER
TAI	International Atomic Time
TIA	Transimpedance Amplifier
TMR	Triple Modular Redundancy
TRD	Transition Radiation Detector
TTC	Timing, Trigger, and Control
RF	Register File
UTC	Coordinated Universal Time

UVM	Universal Verification Methodology
WR	White Rabbit

B Appendix

Generic Modul interface

In the following, a generic module interface example is given showing the SPADIC interface code. It starts with an interface for clocks and resets. It delivers all SPADIC system clocks derived from the received clock and all resets derived from an external reset. The next part is the i2c interface. These wires have to be connected directly to appropriate IO cells. Then the direct RF access is presented. It includes directly used registers for controlling and configuring the SPADIC. In addition, the shift chain interface delivers control connections for the analog shift chain. It is internally attached to a sub RF interface. In the next two blocks, the special CBMnet interface providing data send channels and a DLM interface are shown. Followed by synchronization signals especially created for the SPADIC and some test pins for external LEDs. A link connection interface is the last part of the generic module interface. It must be connected to the LVDS IO cells.

```
module spadic_lsra_top
 #( parameter SIMULATION = 1'b0
  ) (
                // Interface for clocks and resets
                output wire clk2,
                output wire clk10,
                output wire clk2_res_n,
                output wire clk10_res_n,
                output wire bclk_res_n,
                input wire pin_res_n,

                // Interface for I2C external device
                input wire scl,
                input wire sda_in,
                output wire sda_out,
```

```
// Direct RF access
output wire [15:0] REG_CbmNetAddr,
output wire [15:0] REG_EpochCounter,
output wire [8:0] REG_threshold1,
output wire [8:0] REG_threshold2,
output wire  REG_compDiffMode,
output wire [5:0] REG_hitWindowLength,
output wire [31:0] REG_selectMask,
output wire [4:0] REG_bypassFilterStage,
output wire [17:0] REG_aCoeffFilter,
output wire [23:0] REG_bCoeffFilter,
output wire [8:0] REG_scalingFilter,
output wire [8:0] REG_offsetFilter,
output wire [7:0] REG_groupIdA,
output wire [7:0] REG_groupIdB,
output wire [483:0] REG_neighborSelectMatrixA,
output wire [483:0] REG_neighborSelectMatrixB,
output wire [15:0] REG_disableChannelA,
output wire [15:0] REG_disableChannelB,
output wire  REG_disableEpochChannelA,
output wire  REG_disableEpochChannelB,
output wire  REG_enableTestOutput,
output wire  REG_testOutputSelGroup,
output wire  REG_enableTestInput,
output wire [20:0] REG_enableAdcDec,
output wire [15:0] REG_triggerMaskA,
output wire [15:0] REG_triggerMaskB,
output wire REG_enableAnalogTrigger,
output wire REG_enableTriggerOutput,

//Analog shift chain interface
output wire bitinShiftReg,
output wire writeShiftReg,
output wire lastShiftReg,
output wire initreadShiftReg,
output wire readShiftReg,
input wire  bitoutShiftReg,

// Interface to CBMnet to send data
output wire data2send_stopA,
input wire data2send_startA,
input wire data2send_endA,
input wire [15:0] data2sendA,
output wire data2send_stopB,
input wire data2send_startB,
input wire data2send_endB,
input wire [15:0] data2sendB,
output wire link_active,

// Interface to CBMnet for DLM control
input wire dlm2send_va,
input wire [3:0] dlm2send,
```

```
 output wire [3:0] dlm_rec_type,
 output wire dlm_rec_va,

 //Sync signals
 output wire dataSync,
 output wire [1:0] adcSync1,
 output wire [1:0] adcSync2,
 output wire [1:0] adcSync3,
 output wire [1:0] adcSync4,

 //Test outport Pins (to be checked)
 output wire SERDES_ready,
 output wire user_pin1,
 output wire user_pin2,

 //Connection to Link
 input wire serial_clk_in,
 input wire data_in,
 output wire dataA_P,
 output wire dataA_N,
 output wire dataB_P,
 output wire dataB_N
);
```

Altera Stratix IV Board

The Altera prototyping board described in Section 3.7 is presented in figure B-1. Technical specifications:

- HTX Connector with 8bit transceiver based bidirectional interface
- Altera Stratix IV GX 230 (or larger) speed grade -2, -3 or -4
  (For more details, please refer to the Stratix IV user manual)
- 256 MB onboard DDR3-1066 memory
- 2 CX4 connectors routed to FPGA transceivers
- Marvel 88E1111 Ethernet solution
- USB2 connectivity via Mini-USB connector and Cypress CY7C68013A High-Speed USB Peripheral Controller
- Additional external connectivity with Stratix LVDS interface
  using Samtec connectors

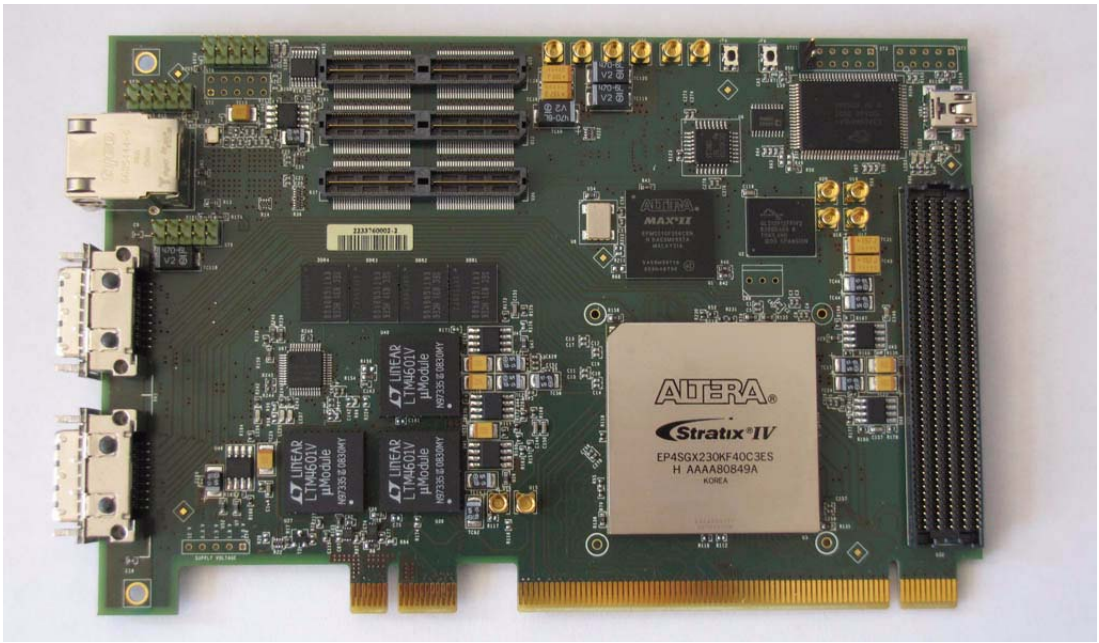


Figure B-1: Altera Stratix IV Board

Tests have been done using the Altera board with an integrated jitter cleaner. The cleaned clock was required as the HTX reference clock during system boot. At the beginning, these tests were not successful. In following measurements, a power problem was found. It is shown in figure B-2. The problem was that the 3.3V power (purple) needed to supply the LMK03033 was not stable early enough, compared to the main FPGA power (light blue). Then system reset was too early and the FPGA tried to program the LMK. After a wait counter was inserted, this sequencing problem was fixed. This was a useful hint for ROC3 development, which has an integrated LMK03033.

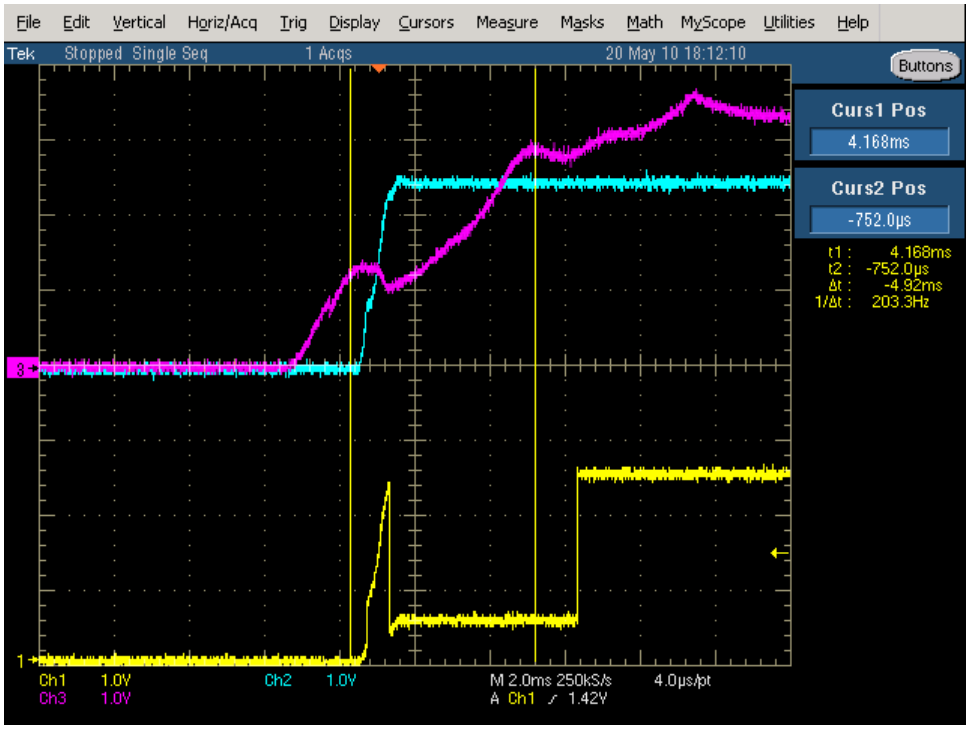


Figure B-2: Altera Board LMK03033 Power Sequence

AOC I2C Problem

During bring up of the AOC PCB V3 a problem occurred. It was not possible to program the ICs over I2C. The measurements showed, as depicted in figure B-3, that the VCSEL driver and TIA ICs were not capable of pulling down the I2C bus to acknowledge the transfer. The solution for first tests of the AOCs was to use an I2C master ignoring acknowledgements. This test worked. Further analysis showed that the ICs were not completely I2C specification conform and were only able to pull down the signal level below a required threshold for a pull-up of 4.7 KOhm attached to the bus. The test systems used up until now have to be adapted.

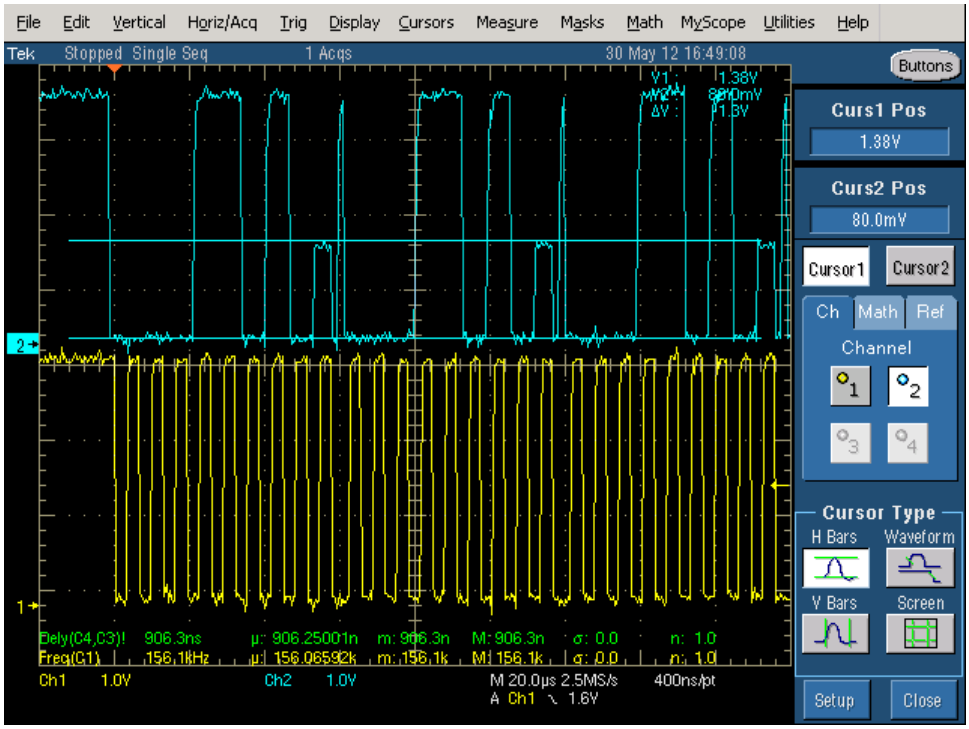


Figure B-3: AOC I2C Sequence with Pull-Down Error

Additional Jitter Measurements

In figure B-4 and figure B-5 screenshots for 250 MHz and 125 MHz jitter measurements are presented. In both jitter measurement diagrams, the upper waveform shows the measured clock and the lower waveform the track of jitter between pairs of clock cycles. The clock frequency measurement values are calculated in P2, below the waveforms. The track is visualized using the Δ-period level changes resulting from P5. The RMS jitter values are calculated in P6. The 250MHz cleaned clock shows a mean RMS jitter of 4.7788ps. Due to test setup noise, there is a periodic peak every roundabout three µs. This can be ignored, because the other test setup for 125 MHz did not show this effect. The 125 MHz cleaned clock shows a mean RMS jitter of 5.5093ps.

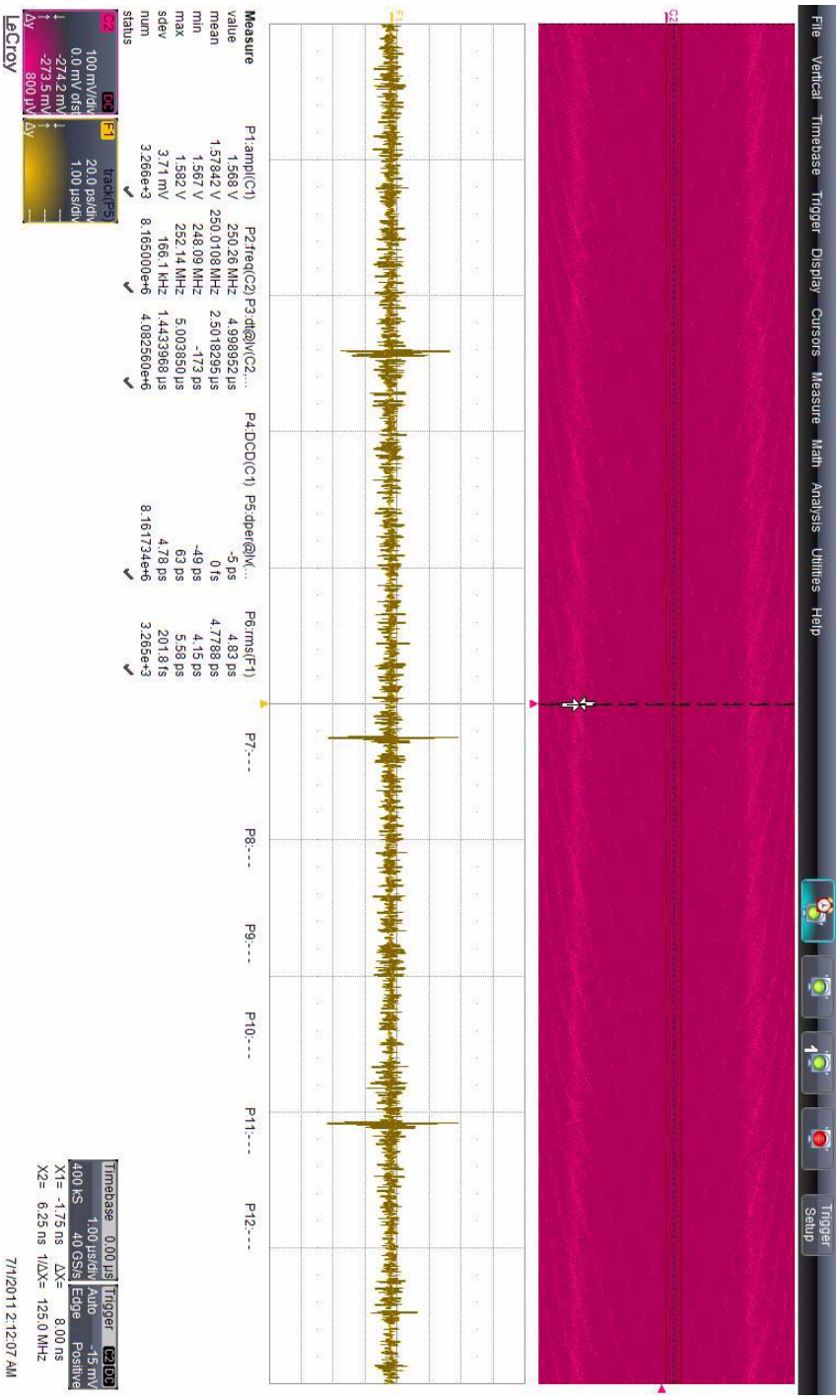


Figure B-4: Jitter cleaned clock at 250 MHz

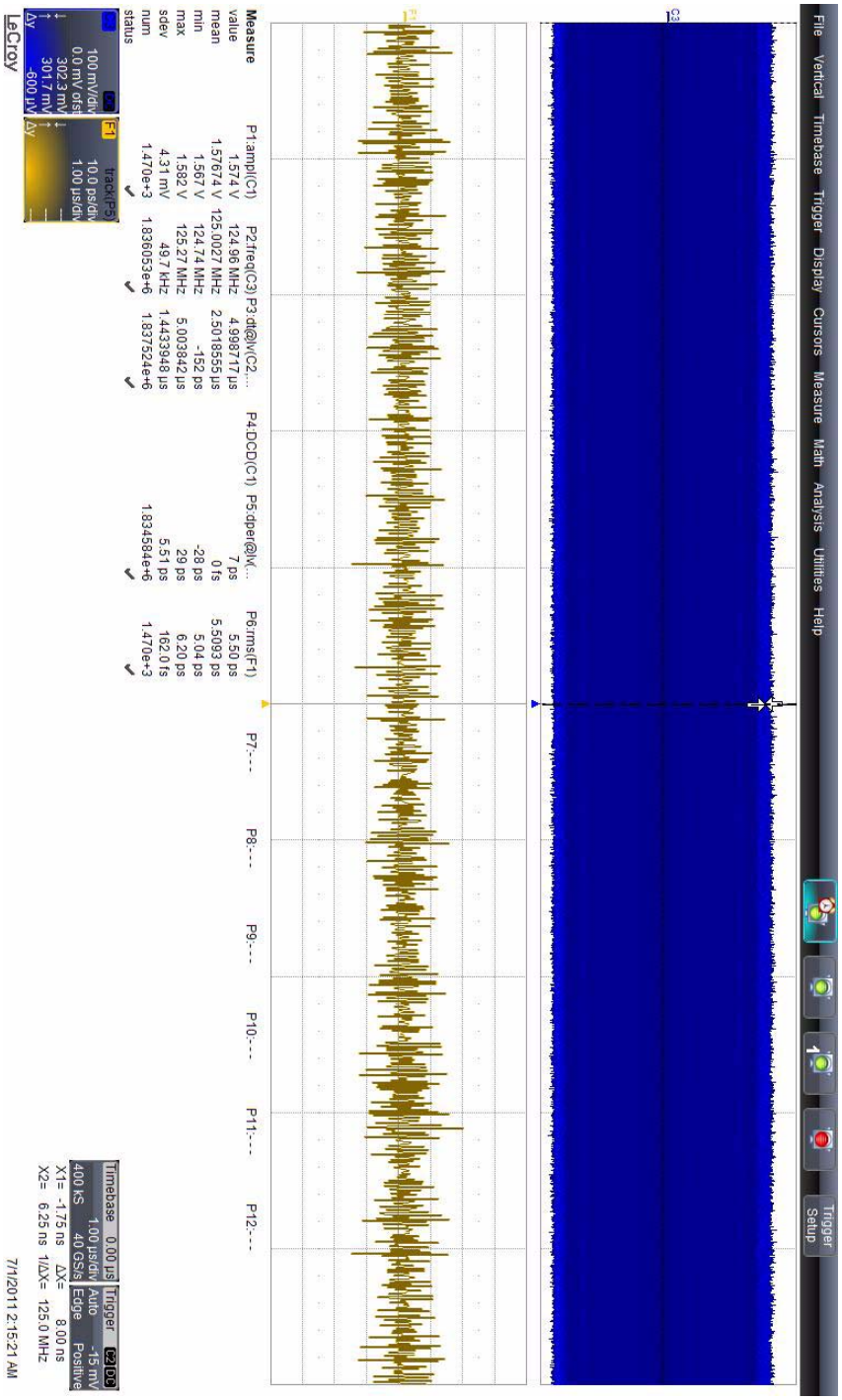


Figure B-5: Jitter cleaned clock at 125 MHz

HUB SERDES concepts

The HUB ASIC development requires an external High-speed SERDES IP. Therefore, a direct collaboration with Prof. Bhattacharyya from the Advanced VLSI Design Laboratory at the Indian Institute of Technology Kharagpur was started. In figure B-6, the SERDES block diagram designed together with this Indian group is presented. It shows the incoming and outgoing LVDS serial data streams to the left, transformed from or into standard CMOS signals with the 20 bit processing width, before the 8b/10b coding stage. Additionally, this path includes eye detection logic. Further from the incoming data stream, a recovered reference clock is generated by the CDR analog block. From the reference clock, the word clock is derived. This word clock will be adjusted by the digital logic implementation to guarantee the right relation between the clocks. All used registers are placed within a RF in the digital part of the design for easy control and test access using standard interfaces like I2C. The first conceptions are done and this is only a promising starting point, but exact details, simulations and models, and prototype mini ASICs will soon appear.

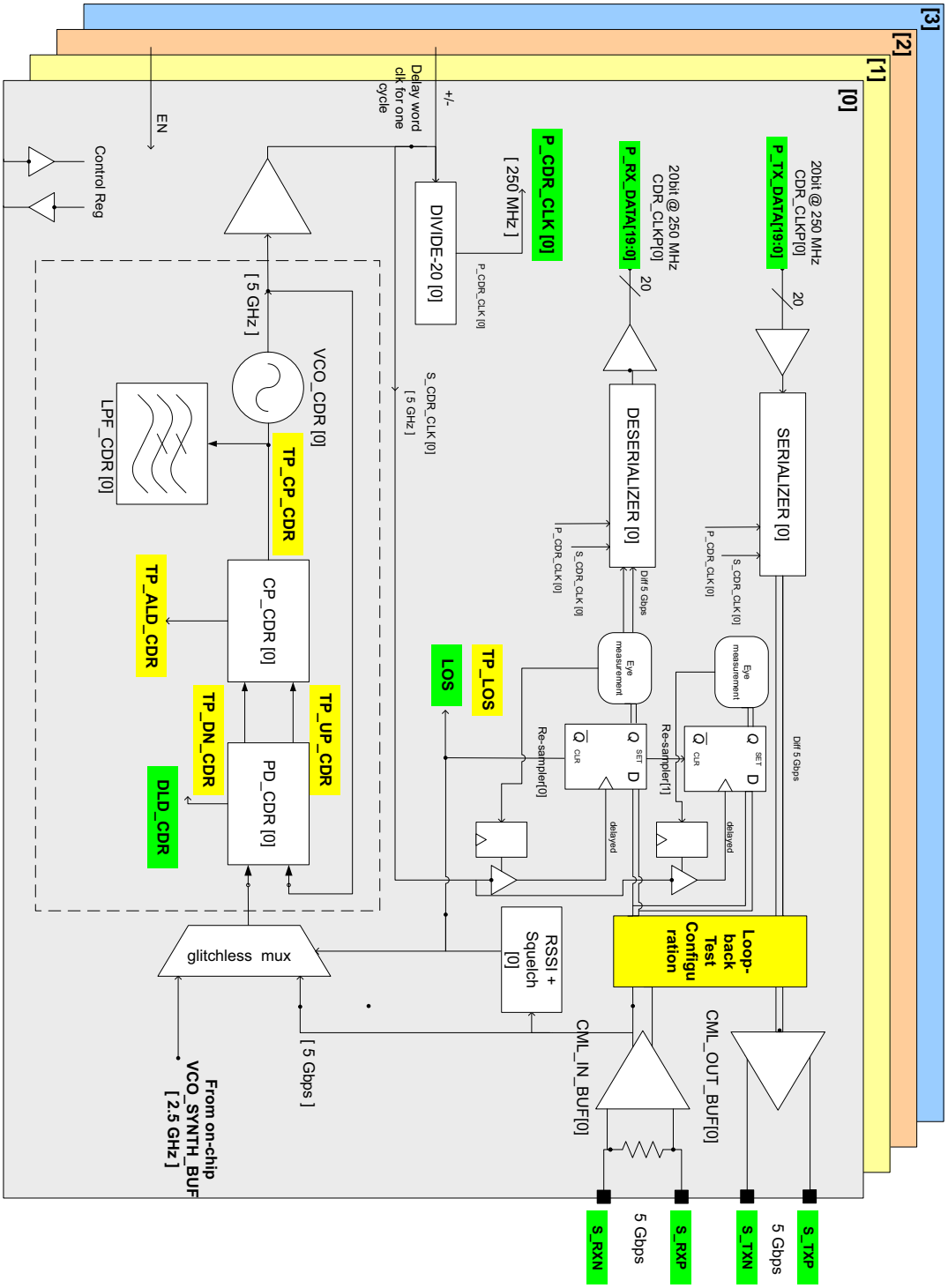


Figure B-6: SERDES Blockdiagram

List of Figures

List of Tables

R References

[1] H.R. Taylor, “Data Acquisition for Sensor Systems”, Springer, ISBN 978-0-412-78560-3, 1997.

[2] Uyless Black, *“ATM Foundation for Broadband Networks”*, Prentice Hall PT, ISBN 0-13-297178-X, 1995.

[3] H. H. Gutbrod, I. Augustin, H. Eickhoff, K-D. Gross, W. F. Henning, D. Kraemer, and G. Walter, FAIR Baseline Technical Report Executive Summary, Gesellschaft fuer Schwerionenforschung mbH (GSI), Member of the Hermann- von- Helmholtz- Association of National Research Centres (HGF), Sept. 2006.

[4] The FAIR website, [Online Oct. 2012]. Available: http://www.gsi.de/fair/index.html.

[5] The GSI website, [Online Oct. 2012]. Available: http://www.gsi.de/.

[6] D. Cleary, "A Lab to Get the Measure of Matter", Science, Vol. 318, pp. 738-739, Nov. 2, 2007.

[7] B. Friman, C. Hoehne, J. Knoll, S. Leupold, J. Randrup, R. Rapp, and P. Senger,*"The CBM Physics Book - Compressed Baryonic Matter in Laboratory Experiments,";* Springer, ISBN 978-3-642-13292-6, January 27, 2011.

[8] The CBM website, [Online Oct. 2012]. Available: http://www.gsi.de/fair/experiments/CBM.

[9] R. Bär, U. Krause, V. Schaa, M. Thieme, W. Schiebel, *“Development of a new Control System for the FAIR Accelerator Complex at GSI”,* TUP107, ICALEPCS 2009, Kobe, Japan, Oct. 2009.

[10] D. Beck et al., *“White Rabbit Technology as Basis for the FAIR Timing System“,* GSI Scientific Report 2011, p. 333, PHN-ACC-RD-45, Jan. 2012.

[11] J. Duato, S. Yalamanchili, and L. Ni, *”Interconnection Networks: An Engineering Approach”*, ISBN 10: 1-55860-852-4, ISBN 13: 978-1-55860-852-8, CA, USA, July 2002.

[12] M.N. Ellanti, S.S. Gorshe, L.G. Raman, W.D. Grover, *“Next Generation Transport Networks - Data, Management and Control Planes”*, Springer, ISBN 0-387-24067-5, 2005.

[13] Altera SDH/SONET Protocol Solutions, [Online Oct. 2012]. Available: http://www.altera.com/technology/high_speed/protocols/sonet/pro-sonet.html

[14] Exar SDH/SONET Products website, [Online Oct. 2012]. Available: http://www.exar.com/communications/sdh-sonet

[15] M Yan,*"“SONET/SDH Essentials",* white paper, SONET Aggregation and T/E Carrier Applications Group, Exar Corporation, 2008.

[16] ITU-T Recommendation, *"Network node interface for the synchronous digital hierarchy (SDH)"*, ITU-T Recommendation G.707, 2007.

[17] “Synchronous Optical Network (SONET) – Network Element Timing and Synchronization”, ATIS-0900105.09.1996(R2008), 2008.

[18] D. Mills, et al.,*”Network Time Protocol Version 4: Protocol and Algorithms Specification"*, Internet Engineering Task Force (IETF), RFC 5905 NTPv4 Specification, June 2010.

[19] D. L. Mills, *"Network Time Protocol Version 4 Reference and Implementation Guide"*, NTP Working Group, Technical Report 06-6-1, University of Delaware, June 2006.

[20] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE 1588 Precision Time Protocol (PTP) Version 2 Specification, IEEE, March 2008.

[21] A. Soppelsa, A. Luchetta, and G. Manduchi, *"Precise Time Protocol Assessment on a Prototype System",* 16th IEEE NPSS Real Time Conference 2009 (RT 09), Beijing, China, May 2009.

[22] J. Serrano et al., *"The White Rabbit Project”*, TUC004, ICALEPCS 2009, Kobe, Japan, Oct. 2009.

[23] The White Rabbit Project, [Online Oct. 2012]. Available: http://www.ohwr.org/projects/white-rabbit

[24] E. G. Cota et al., *"White Rabbit Specification: Draft for Comments" ,* Specification version 2.0, July 2011.

[25] O.S. Bruening et al., *"LHC Design Report"*, ISBN 92-9083-224-0, Geneva, Switzerland, 2004.

[26] The Large Hadron Collider description at CERN, [Online Oct. 2012]. Available: http://public.web.cern.ch/public/en/LHC/LHC-en.html

[27] B. Taylor, *"Timing Distribution at the LHC",* 8th Workshop on Electronics for LHC Experiments, Colmar, France, Sept. 2002.

[28] P. Moreira, A. Marchioro, and K. Kloukinas, *"The GBT, a Proposed Architecture for Multi-Gb/s Data Transmission in High Energy Physics",* Topical workshop on electronics for particle physics 2007, Prague, Czech Republic, Sept. 2007.

[29] P. Moreira et al., *"The GBT Project",* Topical Workshop on Electronics for Particle Physics 2009, pp. 342–346, Paris, France, Sept.2009.

[30] P. Moreira, et al., *"The GBT-SerDes ASIC prototype",* Topical Workshop on Electronics for Particle Physics 2010, Aachen, Germany, Sept. 2010.

[31] The HADES Collaboration website, [Online Oct. 2012]. Available: http://www-hades.gsi.de/

[32] Michel, M. Böhmer, I. Fröhlich, G. Korcyl, L. Maier, M. Palka, J. Stroth, M. Traxler, and S. Yurevich, *"The HADES DAQ System: Trigger and Readout Board Network",* IEEE Transactions on Nuclear Science, Journal Paper, Volume: 58, Issue:4, pp. 1745, Aug. 2011.

[33] Norbert Abel, Frank Lemke, Wenxue Gao, *"Design and implementation of a hierarchical DAQ network"*, Deutsche Physikalische Gesellschaft EV (DPG08), Fruehjahrstagung, Darmstadt, Germany, March 10-14, 2008.

[34] Frank Lemke, Ulrich Bruening, *"A generic link protocol for the CBM DAQ system"*, ISBN 978-3-9811298-6-1, p. 57, CBM Progress Report 2008, Darmstadt, Germany, 2009.

[35] C. J. Schmidt et al., *"Test results on the n-XYTER ASIC, a self triggered, sparcifying readout ASIC",* presented at the Topical Workshop on Electronics for Particle Physics 2007, Prague, Czech Republic, TWEPP-07, Sept. 3-7, 2007.

[36] N. Abel, N. Schroer, B. Stopfkuchen, M. Block, and U. Kebschull, *"Development of the Read Out Controller for the nXYTER Front End Board"*, p. 53, CBM Progress Report 2007, Darmstadt, Germany, 2008.

[37] H. Fröning, M. Nüssle, D. Slogsnat, H. Litz, and U. Brüning, *"HTX-Board: A Rapid Prototyping Station",* 3rd annual FPGAworld Conference, Stockholm, Sweden, Nov. 16, 2006.

[38] W. Gao, A. Kugel, A. Wurz, G. Marcus, and R. Maenner, *"Active Buffer Design for Event Building in CBM Experiment",* presented at the 16th IEEE NPSS Real Time Conference RT09, Beijing, China, May 2009.

[39] J. Adamczewski-Musch, H. G. Essel, N. Kurz, and S. Linev, *"Data Acquisition Backbone Core DABC release v1.0",* 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09), Prague, Czech Republic, March 21-27, 2009.

[40] J. Adamczewski-Musch, H. G. Essel, N. Kurz, and S. Linev, *"First Release of Data Acquisition Backbone Core (DABC)",* presented at the 16th IEEE NPSS Real Time Conference RT09, Beijing, China, May 2009.

[41] J. Adamczewski-Musch, H. G. Essel, N. Kurz, and S. Linev, *"Dataflow Engine in DAQ Backbone DABC",* IEEE Transactions on Nuclear Science, Vol. 57, No. 2, April 2010.

[42] F. Lemke, D. Slogsnat, N. Burkhardt, and U. Bruening, *"A Unified Interconnection Network with Precise Time Synchronization for the CBM DAQ-System",* 16th IEEE NPSS Real Time Conference 2009 (RT 09), Beijing, China, May 10-15, 2009.

[43] F. Lemke, D. Slogsnat, N. Burkhardt, and U. Bruening, *"A Unified DAQ Interconnection Network with Precise Time Synchronization",* IEEE Transactions on Nuclear Science (TNS), Journal Paper, Vol. 57, No. 2, April 2010.

[44] F. Lemke, S. Schenk, U. Bruening,"*The adapted CBM network structure design and CBMnet V2.0 implementation"*, ISBN 978-3-9811298-9-2, p. 61, CBM Progress Report 2011, Darmstadt, Germany, 2012.

[45] J. Gebelein, H. Engel and U. Kebschull, “FPGA Fault Tolerance in Radiation Susceptible Environments”, 11th European Conference on Radiation and its Effects on Components and Systems (RADECS), Sept. 20–24, 2010.

[46] S. Manz, N. Abel, J. Gebelein and U. Kebschull, *“An universal read-out controller”*, Topical Workshop on Electronics for Particle Physics 2010, Aachen, Germany, Sept. 20–24, 2010.

[47] J. de Cuveland and V. Lindenstruth, *“A First-Level Event Selector for the CBM Experiment at FAIR”* , 18th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2010), Taipei, Taiwan, Oct. 18-22, 2010.

[48] Sergei Linev, *“High performance event building with InfiniBand network in CBM experiment”*, IEEE 18th Real-Time Conference 2012 (RT12), Berkeley, CA, USA, June 11-15, 2012.

[49] Hubert Zimmermann, *“OSI Reference Model - The ISO Model Architecture for Open Systems Interconnection”*, IEEE Transactions on communication, Vol. COM-28, No. 4, April 1980.

[50] The HyperTransport 3.1 specification, HyperTransport Consortium, August 18, 2008.

[51] A. X. Widmer and P. A. Franaszek, *"A DC-Balanced, Partitioned-Block, 8B/ 10B Transmission Code"*, IBM Journal of research and development, Volume 27, Number 5, Sept. 1983.

[52] D. Schoenfeld, H. Klimant, R. Piotraschke, *“Informations- und Kodierungstheorie”*, 4. Auflage, Springer Vieweg, ISBN 978-3-8348-0647-5, Wiesbaden, 2012.

[53] David Slogsnat, *“Thightly-Coupled and Fault-Tolerant Communication in Parallel Systems”*, Dissertation, MADOC Document Server, University of Mannheim, Germany, 2008.

[54] Niels Burkhardt, *“Fast Hardware Barrier Synchronization for a Reliable Interconnection Network”*, diploma thesis, University of Mannheim, Germany, Aug. 2007.

[55] Ulrich Bruening, *"Lecture Notes to Hardware Design and Simulation"*, http://www.ra.ziti.uni-heidelberg.de/index.php?page=lectures&id==ss11&lecture=hwe, University of Heidelberg, Germany, 2011.

[56] "LMK03000 Family - Precision Clock Conditioner with Integrated VCO", Texas Instruments Incorporated, Copyright © 1999-2012.

[57] F. Lemke, S. Kapferer, A. Giese, H. Froening, U. Bruening, *"A HT3 Platform for Papid Prototyping and High Performance Reconfigurable Computing"*, Second International Workshop on HyperTransport Research amd Applications, Mannheim, Germany, Feb. 9th, 2011.

[58] E. Greulich, *"System Design of an HT3 Verification Platform based on a high-performance FPGA"*, Diploma Thesis, Mannheim, Germany, 2009.

[59] Altera Corporation products website, [Online Oct. 2012]. Available: http://www.altera.com/devices/dvcs-index.html

[60] Xilinx Inc. products website, [Online Oct. 2012]. Available: http://www.xilinx.com/products/silicon-devices/index.htm

[61] Frank Lemke, Sven Schenk, Ulrich Bruening, *"Prototype Results of an Optical Communication Network for the CBM DAQ-System"*, ISBN 978-3-9811298-7-8, p.54,CBM Progress Report 2009, Darmstadt, Germany, 2010.

[62] Frank Lemke, Sebastian Manz, and Wenxue Gao, *"Time synchronization and measurements of a hierarchical DAQ network"*, Deutsche Physikalische Gesellschaft EV (DPG10), Fruehjahrstagung, Bonn, Germany, March 15-19, 2010.

[63] Frank Lemke, Sven Schenk, Ulrich Bruening, *"Demonstrator beam time results for the clock distribution and synchronization of the CBM-DAQ system"*, ISBN 978-3-9811298-8-5, p. 60, CBM Progress Report 2010, Darmstadt, Germany, 2011.

[64] F. Lemke, S. Schenk, U. Bruening, *"Experiences and results using the CBMnet protocol including precise time synchronization and clock distribution"*, Deutsche Physikalische Gesellschaft EV (DPG11), Fruehjahrstagung, Muenster, Germany, March 21-25, 2011.

[65] Frank Lemke, *"FSMDesigner4 - Development of a Tool for Interactive Design and Hardware Description Language Generation of Finite State Machines"*, Diploma thesis, University of Mannheim, Germany, Aug. 2006.

[66] The FSMDesigner Project website hosted on SourceForge, [Online Oct. 2012]. Available: http://sourceforge.net/projects/fsmdesigner/

[67] Frank Lemke, Mondrian Nüssle, Ulrich Brüning, *Verification of finite state machines using FSMDesigner4*, CDNLive! EMEA 2007, Canvas Conversation Session, Munich, Germany, April 28-30, 2007.

[68] Frank Lemke, Mondrian Nüssle, Ulrich Brüning, *Experience with the FSMDesigner4 high level design entry tool for design and verification in research and teaching*, CDNLive! EMEA 2009, Academic Track, Munich, Germany, May 18-20, 2009.

[69] Sven Schenk, *"Architecture Analysis of Multi-Gibabit-Transceivers for Low Latency Communication"*, Diploma Thesis, Mannheim, Germany, 2008.

[70] Virtex-4 RocketIO Multi-Gigabit Transceiver, UG076 (v4.1), Xilinx Corporation, San Jose, USA.

[71] B. Mohr, N. Zimmermann, Y. Wang, B. Thorsten Thiel, R. Negra, S. Heinen, F. Lemke, S. Schenk, R. Leys, U. Bruening, *"Implementation of an RF-DAC based Multistandard Transmitter System"*, CDNLIVE! 2011, Academic Track, Munich, Germany, May 3-5th, 2011.

[72] B. Mohr, N. Zimmermann, B.T. Thiel, J.H.Mueller, Y. Wang, Y. Zang, F. Lemke, R. Leys, S. Schenk, U. Brüning, R. Negra, S. Heinen, *"An RFDAC Based Reconfigurable Multistandard Transmitter in 65nm CMOS"*, IEEE 2012 RFIC Symposium, Montreal, Canada, June 17-19, 2012.

[73] Christian Leber, *"Efficient hardware for low latency applications"*, Dissertation, University of Mannheim, Germany, 2012.

[74] Tim Armbruster, Peter Fischer and Ivan Peric, *"A Self-Triggered Amplifier/ Digitizer Chip for CBM"*, TWEPP 09 Conference Proceedings, p.457, Paris, France, September 2009.

[75] The SPADIC Project, [Online Oct. 2012]. Available: http://spadic.uni-hd.de/

[76] T. Armbruster, P. Fischer and I. Peric, *"SPADIC - A Self-Triggered Pulse Amplification and Digitization ASIC"*, IEEE Nuclear Science Symposium, Knoxville, USA, Oct. 2010.

[77] Tim Armbruster, Peter Fischer, Michael Krieger, Ivan Peric,*SPADIC – Self-triggered readout ASIC for CBM*, Deutsche Physikalische Gesellschaft EV (DPG12), Fruehjahrstagung, Mainz, Germany,March 19-23, 2012.

[78] K. Kasinski,1 R. Szczygiel and P. Grybos, *"TOT01, a time-over-threshold based readout chip in 180nm CMOS technology for silicon strip detectors"*, The 12th International Workshop on Radiation Imaging Detectors (IWORID), Robinson College, Cambridge U.K., July 11–15, 2010.

[79] K. Kasinski, R. Szczygiel, and P. Grybos, *"Evolution of a prototype Silicon strip detector readout ASIC for the STS"* , CBM Progress Report 2010, ISBN 978-3-9811298-8-5, Darmstadt, 2011.

[80] F. Lemke, S. Schenk, U. Bruening, *"The Hierarchical CBM Network Structure and the CBMnet V2.0 Protocol"*, Deutsche Physikalische Gesellschaft EV (DPG12), Fruehjahrstagung, Mainz, Germany,March 19-23, 2012.

[81] Frank Lemke, Ulrich Brüning, *"Design Concepts for a Hierarchical Synchronized Data Acquisition Network for CBM"*, IEEE 18th Real-Time Conference 2012 (RT12), Berkeley, CA, USA, June 11-15, 2012.

[82] Clos, Charles, *"A study of non-blocking switching networks"*, Bell System Technical Journal 32 (2): 406–424, ISSN 00058580, March 1953, retrieved 22 March, 2011.

[83] R. Boppana and C. Raghavendra, *"Designing efficient Benes and Banyan based inputbuffered ATM switches"*, IEEE International Conference on Communications (ICC'99), Vancouver, B.C., Canada, 1999.

[84] B. Geib, "Hardware Support for Efficient Packet Processing", Dissertation, University of Mannheim, Germany, March 2012.

[85] H. Litz, H. Froening, U. Bruening, *"HTAX : A Novel Framework for Flexible and High Performance Networks-on-Chip"*, Fourth Workshop on Interconnection Network Architectures: On-Chip, Multi-Chip (INA-OCMC) in conjunction with Hipeac, Pisa, Italy, January 25 - 27, 2010.

[86] UVM community webside, [Online Oct. 2012]. Available: http://www.uvmworld.org

[87] M. Nicolaidis et al., "Soft Errors in Modern Electronic Systems", ISBN 978-1-4419-6992-7, Frontiers in Electronic Testing, Vol. 41, Springer, 2011.

[88] N. Kumar and D. Zacher, "Automated FSM Error Correction for Single Event Upsets", 2004 MAPLD International Conference, Washington, D.C., USA, September 8-10, 2004.

[89] Philipp Schaefer, *"Synchronization of Front-End Electronics in a Data Acquisition Interconnection Network"*, Bachelor Thesis, Mannehim, Germany, June 26, 2012.

[90] Spartan-6 FPGA SP605 Evaluation Kit, [Online Oct. 2012]. Available: http://www.xilinx.com/products/boards-and-kits/EK-S6-SP605-G.htm

[91] HD-AOC Project, [Online Oct. 2012]. Available: http://ra.ziti.uni-heidelberg.de/index.php?page=projects&id=aoc

[92] **"**HyperTransport™ Node Connector Specification", Rev. 1.0b, 12/09/2009.

[93] Samtec, Inc., [Online Oct. 2012]. Available: http://www.samtec.com

[94] D. Wohlfeld, F. Lemke, H. Froening, S. Schenk, and U. Bruening, *"High Density Active Optical Cable: from a Concept to a Prototype"*, SPIE Photonics West, Optoelectronic Interconnects and Component Integration XI, San Francisco, California, January 22-27, 2011.

[95] Denis Wohlfeld, *"Simulation, Analysis, and Fabrication of Miniaturized Components with Applications in Optical Interconnects and Parallel Microscopy"*, Dissertation, University of Heidelberg, Germany, Dec. 2009.

[96] IPtronics products, [Online Oct. 2012]. Available: http://iptronics.com/Products.html

[97] ULM Photonics, [Online Oct. 2012]. Available: http://www.ulm-photonics.com/

[98] Vertically Integrated Systems, [Online Oct. 2012]. Available: http://www.v-i-systems.com/_rubric_e/index.php?rubric=VI+Systems+EN

[99] GigOptix, [Online Oct. 2012]. Available: http://www.gigoptix.com/

[100] K. Hwang, *"Advanced Computer Architecture"*, ISBN 007053070X, 9780070530706, McGraw-Hill Education (India) Pvt Limited, 2003.

[101] Advanced Simulation and Computing - BlueGene/L, [Online Oct. 2012]. Available: https://asc.llnl.gov/computing_resources/bluegenel/configuration.html

[102] A. Gara et. al, *"Overview of the Blue Gene/L system architecture"*, IBM Jorunal of Research and Development, vol 49, no. 2/3, pp. 195 – 212, March/ May 2005.

[103] N. Tallet and P. Vezolle, *"Blue Gene/P Application User Worshop"*, PSSC Montpellier Blue Gene Team, IBM, 2008.

[104] The Blue Gene/P Team, *"An overview of the BlueGene/P project"*, IBM Journal of Research and Development, vol. 52, no. 1/2, pp. 199 – 220, Jan./ March 2008.

[105] D. Chen et al. ,"The IBM Blue Gene/Q Interconnection Network and Message Unit", SC '11 Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis Article No. 26 , ISBN: 978-1-4503-0771-0, ACM, New York, NY, USA, 2011.

[106] D. G. Feitelson and L. Rudolph, "Gang Scheduling Performance Benefits for Fine-Grain Synchronization", Journal of Parallel and Distributed Computing, 1992.