Specification of the VETO system at European XFEL

Revision History

Revision #	Date	Author	Comments
0.1	29.11.2011	P. Gessler	First draft based on original document
0.2	20.12.2011	P. Gessler	Added draft description of VETO system
0.3	01.03.2012	P. Gessler, T. Gerlach et. al., J. Coughlan	Added LPD and DSSC texts, corrections
0.4	28.03.2012	P. Gessler	Reformatting, added info from T. Gerlach to DSSC part
0.5	30.04.2012	P. Gessler, P. Goettlicher	Added AGIPD part, title, correction (AC coupling, typos)

Table of Contents

Introduction 4
System Overview 4
VETO sources 4
VETO Unit 5
VETO User 5
Clock and Control System 5
2D Detectors 5
Special Case: Clock and Control acts as VETO Unit 5
Description of possible decisions 6
VETO 6
GOLDEN 6
NO-VETO 6
Examples 6
Native implementation with limited storage 6
Native implementation for data reduction 7
Reduced implementation with limited storage 7
Input and Output Interfaces of the VETO Unit 8
Input from a VETO source 8
Physical parameters 8
Protocol 9
Output of the VETO Unit 11
Physical parameters 11
Protocol 11
VETO via the Clock and Control system (for 2D Detectors) 12
Connection to the VETO Unit 12
Interface to the Front-end Electronics 13
Physical parameters 13
Protocol 13
Appendix 16
Requirements and implementation for certain VETO users 16
LPD Detector 16
DSSC Detector 17
AGIPD Detector 19

Introduction

At XFEL different types of detectors and diagnostic sensors will be used at the experiments and along the beam lines. For those systems providing information for each individual bunch, a maximum of 2700 data sets per 10Hz are expected. Especially of detectors providing 1D or even 2D data sets per bunch will produce a high amount of data bandwidth, which has to be transported through the individual data acquisition (DAQ) chains till it will be stored on a file system.
For some detectors the number of data sets, which are possible to transport from the detector to the DAQ chain, is limited. That implies, that it is not possible to record all possible 2700 bunches. However, currently all those detectors provide a way to remove unwanted data sets in an early stage in the detector head and therefore frees up space for other, possible better, data sets. In this way a much more efficient use of the limited storage could be achieved.
This described mechanism is called VETO, as a low latency signal providing a decision, if one data set should be removed from the storage acts as a veto right to the detector.
For those detector or diagnostic systems, which are able to provide information for all possible bunches, the VET system could be used to reduce the amount of data to be transmitted through the DAQ chain in an early stage, if the data sets which are rejected are not worth to look at or to be saved anyway.

The following sections provide a detailed description and specifications of the whole VETO system and information about the interfaces and protocols used.

System Overview

An overview of the VETO system including interfacing elements is shown in Figure X.

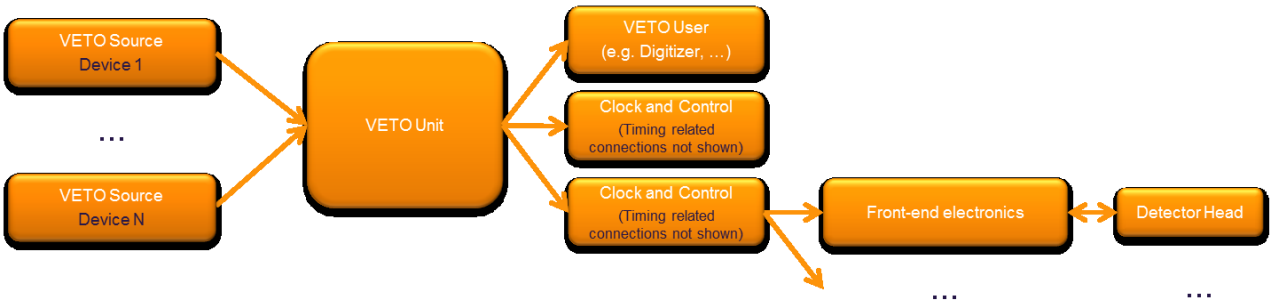

Figure 1: Overview block diagram of the VETO system interconnections with interfacing elements.

VETO sources

Different type of devices could act as a VETO source. An important requirement is, that it has to deliver information, which is relevant in order to make a decision if a data set of a connected VETO user could be rejected or removed from the storage. Furthermore, this information has to be delivered with as low latency as possible, as this it has to be further processed and transmitted to

the front-end electronic of the VETO user and applied to the storage cell as fast as possible in order to make efficient use of it.
Examples of possible VETO sources are: 0D detectors like (Avalanche) Photo Diodes, photon beam line or machine diagnostics like Toroid Protection System and the Timing and Machine Protection system. There might be others in the future.

VETO Unit

The VETO Unit collects all information from VETO sources and processes this data in order to classify each bunch for each individual output (e.g. VETO User). In order to accomplish that, different techniques will be applied like threshold detection, comparison, evaluation of logical connections and so on.
The final decision will be then transmitted to the next stage, which might be a Clock and Control board or a direct VETO User.

VETO User

A VETO User is defined as a final receiver of the VETO information. It will use the classification information created by the VET Unit based on the VETO Source input data. Based on the classification, it will reject, remove or tag the data set for the related bunch and takes further required steps depending on the implementation of the individual device and its DAQ chain.

Clock and Control System

The Clock and Control System (CC) is a MicroTCA based solution for providing a common clock reference, synchronization, VETO and status interface for the currently developed AGIPD, DSSC and LPD 2D detectors (and possibly more in the future). CC is developed at University Collage London. Further information can be found in X (which also includes the original VETO interface definition between the CC and 2D detectors).
Besides the reference clock, timing and status read out functionality (which is not shown in the diagram and will not be discussed in this document) the VETO function is limited to work as a type of gateway. The already processed and classified information will be received from the VETO Unit and then send to the Front-end electronics of the 2D detectors in a serial data stream at a data rate about 100MHz. Therefore no further evaluation or processing is foreseen.

2D Detectors

The 2D detectors consist of two elements shown in the block diagram: the front-end electronics (FEEs) and the detector head. The detector head includes the actual detector semiconductor as well as the ASIC, which is directly bonded to the detector. The ASIC is connected to the FEE. The FEEs include besides power supply, control logic, clock and synchronization FPGAs, which process the gathered data from the detector head and transmits it to the next DAQ element in the chain. The FPGA will decode the VETO information received from the CC system and provides this to the ASIC, which will then react on that. Additionally the information might be also used in the FPGA in order to assign the correct bunch numbers to the data sets read from the ASIC.

Special Case: Clock and Control acts as VETO Unit

A special case might be possible especially in the early implementation stage of the whole system, when the CC system could implement the VETO Unit along with the base functionality.

If this is the case, then the VETO sources are connected directly to the CC board and the FPGA used for implementing the CC functionality accommodates the VETO Unit logic discussed earlier as well. This would simplify the hardware set up and still allows implementation and test of the VETO system.

Description of possible decisions

Based on the information of VETO sources the VETO unit will derive a decision of the most likely quality of a certain bunch measurement. Those decisions will act as classifications, which could be used at VETO users to increase the storage efficiency. This section will describe the possible classes in more detail and gives examples of how the might be used.

VETO

If a certain bunch is classified as VETO, that means, that the measurement of this bunch was most likely not successful. Why this is the case is defined by the used VETO sources and the decision matrix defined in the VETO unit and will most likely depend on the individual experiment and setup.
If a VETO user receives an as VETO classified signal, the related information for that bunch could be freed (e.g. the related buffer reused or the related data not be transmitted to the next stage in the DAQ chain.

GOLDEN

If a certain bunch is classified as GOLDEN, that means, that there is high likelihood, that the measurement for that bunch was very good. As a result, a VETO user should ensure that the related information is kept (e.g. not deleted or overridden) and delivers this information to the next DAQ stage.

NO-VETO

If a bunch can be classified neither as VETO nor as GOLDEN, it will be NO-VETO. Therefore NO-VETO is defined as the default class.
If an explicit NO-VETO signal is received by a VETO user, different reactions are possible dependent on the policy and working principle of the VETO user (see examples below).

Examples

Native implementation with limited storage

The first examples is based on a detector, which provides 512 buffer cells for data acquired from up to 512 different bunches. If the data of one cell is regarded as not useful, it could be tagged as free. It further allows to flag cells as keep. All un-flagged cells will be overwritten, if no free cells are available, but new data sets are captured by the detector (e.g. number of bunches are higher than the number of buffer cells)

Now imagine a machine setup, where 1000 bunches reach the detector. Each data set per bunch will be saved in one buffer cell. While the data is saved, information on the VETO line is reaching the front-end electronics of the detector and defines bunch 10 and 20 as VETO and bunch 300 and 800 as GOLDEN. All others are NO-VETO. Our example detector will now handle the buffer cells in the following way:
Buffer 10 and 20 are tagged as “free”, as soon as the information on the VETO line is available. If the information about bunch number 300 (GOLDEN) are available at the detector, buffer cell number 300 is tagged as “keep”.
If bunch 512 was captured, all buffer cells were used once, but cell number 10 and 20 are tagged as free. Now the storage system wraps around and starts from the beginning to fill up all cells defined as “free”. After reusing cell 10 and 20 with data from bunch 513 and 514 all cells are used. Now there is the default policy to overwrite all cells, which are not tagged as “keep” or are free anyway. Therefore the detector will overwrite now step by step the already used cells except for cell number 300, which is tagged as keep. The front-end electronics will do the book keeping of the cell usage and relation of cell number and bunch number in parallel.
If the data of bunch 800 was saved the information from the VETO line is available, it is recognized as a GOLDEN event and therefore its cell tagged as keep.
After all 1000 bunches had been processed, the system stops and transmits the data sets to the next DAQ chain stage with the following result: The information of all GOLDEN events had been captured and the remaining cells are filled with the latest bunches (300, 489-1000).
Some remarks: The underlying policy (overwrite everything except for GOLDEN events) assumes, that there might be better events to come, than we had in the first bunches (unless they are recognized as GOLDEN). If there are more than 512 bunches, it is quite likely, that the first bunches in the train are not kept, if they are not regarded as GOLDEN by the VETO system.

Native implementation for data reduction

Similar to the previous example, a scenario might be possible, where the storage in the detector is not limited, but the amount of data to be transmitted on to the next stages in the DAQ chain should be reduced.
Based on the classification of bunches done in the VETO unit the front-end electronics of a detector can reduce the number of data sets to be transmitted. A policy can be defined, that not data is transferred, which was classified as VETO. Even further reduction could be made, if only data is transmitted defined as GOLDEN. By adjusting the decision matrix for that specific VETO user in the VETO unit the sensitivity could be adjusted.

Reduced implementation with limited storage

If a detector is not able to implement the previously described or comparable implementation it has two possible options: Ignore GOLDEN events or ignore VETO events combined with different policies.
In this example the detector has the same storage capacity of 512 cells as described in the first example. But in this case it is only possible to tag certain cells to be reused.
If the GOLDEN events are going to be ignored, the standard behavior of the system will be to fill up the available storage cells. If a VETO event had been received, the related cell will be tagged as reusable. If all are filled, it will overwrite the cells tagged as “reusable”. The result will be, that

the detector will keep all data related to the first 512 bunches not classified as VETO. As a result, GOLDEN and NO-VETO will be regarded as the same. It is possible, that events defined as GOLDEN will be not available in the later DAQ stages.
If the VETO events are going to be ignored, the default behavior of the system will be to still fill up the buffer cells till all are used. All cells will be tagged as reusable, unless a GOLDEN event for a certain bunch was received. Therefore it will automatically reuse all cells again and again unless certain cells are not tagged as reusable. The result will be, that the detector will keep all data related to GOLDEN events plus the latest bunches fitting in the remaining cells. It is possible, that event, classified as VETO will still be available in later DAQ stages.
Which of these two implementations is possible or the better one depends on the detector, the experiment, the preferences of the experimenter, the VETO sources and the decision matrix. Ideally both policies are implemented and could be selected by the user.

Input and Output Interfaces of the VETO Unit

In this chapter the input and output interfaces of the VETO Unit will be discussed and the physical parameters and used protocols provided.

Input from a VETO source

VETO sources have to provide bunch related information with low latency in order make use of it in the detector and diagnostic front-ends. Additionally, for some sources a distance of more than one kilometer might be possible. Therefore this interface is designed to allow optical long distance connections as well as short connections in a serial way. In order to reduce latency, it has to work with a high data rate and should have no protocol overhead for the bunch related data.

Physical parameters

The physical connection between the VETO source and the VETO unit is implemented high speed serial connection between the FPGA of the VETO source and the FPGA of the VETO unit. The VETO unit provides small-form pluggable (SFP) sockets on the face plate of the board, which allows different types of transceivers. The most commonly used are optical transceivers with wavelengths around 800nm with multimode fiber cables as well as transceivers around 1310nm or 1550nm with single mode fiber cables for longer distances. Besides the SFP solution there is also an in-crate connection possible. The VETO unit (as well as the CC) will be based on MicroTCA.4. In this standard point-to-point connections between boards in a crate are possible. Those connections also allow a VETO source to VETO unit connection.
The bit rate for the transmission will be 2.5GB/s. As this data rate is the same as in PCI express v1.0 (PCIe gen 1.0), which is the standard communication in the crate to a CPU, the reference clock should be available to almost all boards, which might be a VETO source.
The communication is defined to be uni-directional: the VETO source is the transmitter and the VETO unit is the receiver.

Parameter	Value
Connection type (Backplane)	Point-to-point lines channel 12-15
Connection type (Front panel)	SFP with 1310nm single mode fibers
Bitrate	2.5Gbps

Table 1: Physical parameters of the VETO unit input (also applies to the output).

Protocol

The protocol used for the input and output of the VETO unit is implemented in three layers (see X and the following sections).

8B10B Encoding

The lowest layer defines an 8B10B encoding as commonly used on serial high speed transmissions. It encodes eight bits into 10 bits based on an encoding table in order to ensure long term DC balancing of the signal. Encoded data words are denoted as Dxx.x, where xx.x is replaced by the actual data word. Additionally a small number of special bit combinations are defined as komma characters (denoted as Kxx.x, where xx.x will be replaced by the actual number of the used komma character), which cannot be generated by combining any other defined ten bits word. They are used to synchronize the word boundaries on the receiver side (called komma alignment).

Frame formats

The next protocol layer defines the frame format and special metadata to be transmitted. It combines special komma characters with data and CRC to ensure synchronization, data integrity and low protocol overhead to reduce latency. The transmission of bits is always grouped into 20 bits including two 8B10B encoded words. Defined 20 bit words are shown in Table 2.

Mnemonic	8B10B words	Description
<IDLE>	K28.5, D0.0	Sent all the time no data has to be transmitted
<DATA_1>	Dxx.x, Dxx.x	First data word (or only one) to be sent 2 highest bits: index number of transmitter 14 lower bits: payload data to be transmitted (defined in higher level protocol)
<DATA_N>	Dxx.x, Dxx.x	Further payload data (defined in higher level protocol)
<BUNCH_ID>	Dxx.x, Dxx.x	Identifies the bunch number related to the <DATA_X>
<EOF>	K28.6, D0.0	Defines the end of a frame
<INFO>	K28.4, D0.0	It defines the beginning of the metadata frame
<INDEX>	D0.0, Dxx.x	2 lowest bits defines the index number of transmitter
<PRODUCER_NAME>	8x Dxx.x, Dxx.x	16 ASCII characters naming the transmitter
<PRODUCER_TYPE>	Dxx.x, Dxx.x	Defines the type of producer as a number
<FORMAT_TYPE>	Dxx.x, Dxx.x	Defining the data type sent in DATA_1/DATA_N
<TRAIN_ID>	2x Dxx.x, Dxx.x	64 bits defining the train ID for the coming data

Table 2: Defined words for the VETO input and output frame format and metadata protocol layer.

If there is no data to be transmitted, <IDLE> words are send all the time and even between any other types of word.
Before the first bunch related data is going to be produced and transmitted, metadata about a transmitter is prepared and sent to the receiver. The metadata frame has the following format:
…<IDLE><INFO><INDEX><PRODUCER_NAME><PRODUCER_TYPE><FORMAT_TYPE><TRAIN_ID><CRC><EOF><IDLE>…

The <INFO> word denoted the beginning of the metadata frame. The next non-idle word identifies the <INDEX> number of the transmitter. This is important, if more than one transmitter (source of data) is transmitted through one serial connection. This is possible, if sources are daisy-chained or if sources are aggregated in a special concentrator device. If that is done, the different information is time multiplexed and identified by the mentioned <INDEX> number in the metadata as well as in the bunch related data. The numbering is implemented in the following way: The first source in the chain (identified by having no input from another source) is number 0. The next in the chain will receive the packets of source index 0 and increment it by 1. This will be continued until the fourth source identified as index 3. If there are more than four sources chained, the last ones won't sent any further data and report an error to their user application.
The following 16 8B10B encoded bytes are ASCII characters identifying the source of the data in a human readable representation (<PRODUCER_NAME>).
Besides the human readable ASCII representation of the producer, the field <PRODUCER_TYPE> will provide a unique number per producer class (e.g. Toroid Protection System, Bunch Arrival Time Monitor, VETO Unit,…) in a number representation to allow a receiver to easily check if the producer is compatible with the receiver. The number assignment will be based on a table continuously updated on the XFEL FPGA wiki page (see X).
The next field is the <FORMAT_TYPE> of the bunch related data. Based on a look-up table this allows the determination of the number of expected bunch related data words as well as its contend (defined in the higher level protocol). In this way it is possible to generically use the data at the receiver side or at least ensure that the interpretation is compatible with the current implementation.

Besides the metadata the important bunch related data will be transmitted to the receiver. The format of the frame is the following:
…<IDLE><DATA_1><DATA_N>…<BUNCH_ID><EOF>…
As low latency of the transmission is important, the packet header had been reduced to only two bits to encode the four possible producers in a chain. Those two bits are the highest in the <DATA_1> word. The remaining 14 bits are used by the higher level protocol discussed in the next section.
If there are more than 14 bits to be transmitted, consecutive <DATA_N> packets can be transmitted, each carrying 16 bits. The number of words (bits) to be sent is defined by the <FORMAT_TYPE> transmitted in the metadata frame.
The payload data is followed by the <BUNCH_ID> identifying the bunch number in a train.
To ensure consistency of the data transmitted and detect simple bit errors, a cyclic redundancy check value is calculated at the transmitter side and attached to the transmitted data in the following field.
Finally the frame is closed by an end-of-frame (<EOF>) word to delimit the transmitted packet.

Bunch related packet format

The previous section described the frame format of the bunch related data. This included undefined data (<DATA_1, DATA_N>). This data will be used by the applications in the FPGAs to transmit their processed bunch related information to a receiver. What has to be transmitted differs depending of what information is detected and processed at a certain receiver. It might contain one or more values of different bit length with different interpretation like floating point, fix point (with different decimal point positions), integer or boolean values.

What is transmitted in the DATA part of the frame will be defined as <FORMAT_TYPE> as described in the metadata frame.
An example would be a beam position monitor delivering for each bunch position information. For this application two values (one for X and one for Y position) might be transmitted. The representation might be an 18 bit fixed point number representing the deviation from the center in millimeters [1].
The goal is that the number of format types will be as low as possible. It should be tried to reuse already defined formats if possible. In this way it is easier to generically make use of data sources as VETO sources. However, which format types will be defined in the future and which will be implemented to be used in the VETO unit will be defined at a later stage here.

Output of the VETO Unit

The output of the VETO unit provides the result of the decision matrix implemented by the VETO unit based on all input information. It describes for each bunch if the corresponding data set should be kept or be removed in the VETO user. Similar arguments given in the input related section are also true for the output: possible long distance to VETO users, low latency, high data rate and so on.
Therefore, and also to simplify the design, the same physical parameters and the lower protocol is identical to the input specification. Only the higher level protocol is different and will be described in more detail in the related paragraph in this section.

Physical parameters

The physical parameters of the output of the VETO Unit are identical to the input parameters and are summarized in Table 1 and the according section.

Protocol

The lower level protocols are the same as described in the input section and in Table 2. That means that the data transmission is serial using 8b10b encoding. During the time period, where no data has to be sent, <IDLE> packets are transmitted. In the time frame between two bunch trains, metadata about the transmitter side is sent in order to allow determination of the type and status of the transmitter from the receiver side without external configuration. When bunch related data is available, it is sent through the link and will be decoded at the receiver side.
The important aspect here is the definition of the <FORMAT_TYPE> and its representation in the DATA part in the bunch related frame as well as the value for the <PRODUCER_TYPE> and the name in the <PRODUCER_NAME> field. These definitions are shown in Table 3 and Table 4.

Field	Value
<PRODUCER_NAME>	"VETO UNIT XXXXXX" (XXXXX defines a location name)
<PRODUCER_TYPE>	1
<FORMAT_TYPE>	1
<DATA_1>	Lowest 14 bits used as shown in Table 3

Table 3: Fields value definition for the VETO unit output.

[1] This just a possible implementation used as an example. The actual implementation of the XFEL BPMs and XBPM are not taken into consideration.

Command	Reserved (bit 13 – 3)	Command bits (bits 2 – 0)	Description
VETO	0000000000	010	Identifies this bunch to be vetoed
NO-VETO	0000000000	001	No VETO or GOLDEN defined for this bunch
GOLDEN	0000000000	011	Identifies this bunch as GOLDEN
START	0000000000	100	Information from the first bunch had been received from the earliest VETO source
STOP	0000000000	101	Information of the last bunch in train from latest VETO source had been processed
RESERVED	0000000000	110, 111	Reserved

Table 4: Bunch related DATA bits definition for VETO unit output.

Table 4 defines the usage of the 14 data bits sent along with the <BUNCH_ID> in the bunch related data frames it will be described in more detail in the following paragraphs.
If we assume, that more than one VETO source is connected to the VETO unit, the time when bunch related data packets arrive might differ. The reason for that might be the cable length or read-out and processing time of the VETO source. When the packet about the first bunch arrives from the earliest VETO source, a START packet is sent to the VETO user (or CC) in order to inform it, that the bunch processing and evaluation started.
From then on VETO, NO-VETO and GOLDEN packets are sent, as soon, as the decision about the classification is possible. Some examples: if the earliest VETO source delivers an information bor bunch 10, which directly leads to the decision that the bunch can be VETOed, a VETO packet is sent for that bunch. If a decision cannot be made until the latest VETO source delivered its information, the packet will be sent after that time.
When the information about the last bunch in the train from the VETO source with the longest delay arrived and had been processed, the STOP packet is sent, to inform the VETO users (or CC), that no further packets for this train will be sent.

Note: also the fixed and variable delay protocol might be implemented here

VETO via the Clock and Control system (for 2D Detectors)

If the VETO information is distributed through the Clock and Control System, it will receive the output data stream of the VETO unit as usual VETO users and then distributes it to the connected FEEs of the detectors. The important issues here are, that the physical connection as well as data rate, latency, order and lower protocol are different than the output of the VETO unit. The translation and pipelining is done in the CC system.

Connection to the VETO Unit

The Clock and Control board acts as a VETO user from the VETO unit's point of view. The connection is therefore implemented as described in the output section of the VETO unit. Depending on the position of the board it may use the in-crate point-to-point connection or the fiber cable via SFP connectors.

Interface to the Front-end Electronics

The connection between the Clock and Control board and the FEEs fulfills different tasks: it provides a reference clock (frequency and phase) derived from the machine by multiplying 4.5MHz (1.3GHz / 288) by 22 yielding 99.3MHz, it synchronizes the DAQ via START, STOP and RESET commands and further information, reads back status information from the FEEs and finally distributes the VETO information. In this document only the VETO aspect will be described and others are only mentioned, if it is relevant to the VETO implementation. More details on the Clock and Control functions and interfaces can be found in X.

Physical parameters

The connection between the Clock and Control board and the FEEs is done via RJ45 connectors and ordinary network cables. However, the usage of the differential pairs is different than normal network interfaces. The pinout of the connector is shown in Figure 2.

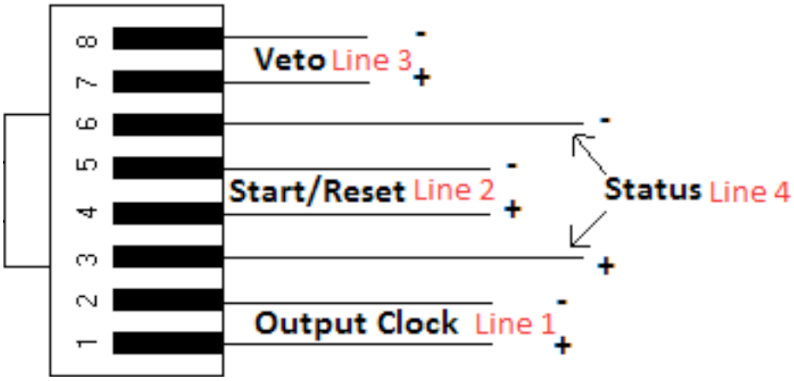

Figure 2: Pinout of the RJ45 connector to interface with the FEEs (taken from X).

Important for the VETO information transmission are the VETO pair (Line 3) and the clock pair (Line 1). The VETO line carries the serial data, which encodes the actual VETO information as calculated by the VETO unit, transmitted to the Clock and Control board and there translated for the FEEs. The bit rate exactly matches the clock frequency of the clock line. Therefore each VETO bit will be latched on the receiver side with the rising edge of the clock line, which implements a source synchronous transmission (a possible skew between the VETO and clock line will be compensated in the CC).

Table 5 summarizes the physical parameters of the connection.

Parameter	Value
Transmission type	Source synchronous (clock + data)
Logic levels	LVDS
Coupling	AC (Status on CC side, others on FEE side)
Rate	99.31 Mbps

Table 5: Overview of the physical parameters of the interface between the Clock and Control board and the front-end electronics of the detectors.

Protocol

The protocol on the VETO line is different than the one on the VETO unit output. On one hand side there is no 8b10b encoding and related comma characters used. Furthermore, there are dif-

ferences in synchronization and ordering of the events and they depend on two possible modes of the transmission: variable delay and fixed delay protocol.
The general packet format of the protocol is illustrated in Table 6.

Command	Start bit	Command bits	Bunch ID (12 bits)	Reserved (4 bits)	Description
VETO	1	10	XXXXXXXXXXXX	0000	Identifies bunch ID to be vetoed
NO-VETO	1	01	XXXXXXXXXXXX	0000	No VETO defined for that bunch ID
GOLDEN	1	11	XXXXXXXXXXXX	0000	Identifies bunch ID as GOLDEN
RESERVED	1	00	XXXXXXXXXXXX	0000	Reserved

Table 6: Packet format of VETO information transmitted by the CC to the FEEs.

Each data packet consists of 19 bits. The first bit acts as start bit and defines the beginning of a packet. The following two bits encode the type of the related bunch as calculates by the VETO unit. Three options are foreseen: 10 = VETO, 01 = NOVETO and 11 = GOLDEN. The combination 00 is reserved. The next 12 bits provide the bunch number starting from one (all zeros is reserved). The remaining four bits are reserved and should be ignored on the receiver side.
An important aspect of the protocol is, that the delivery of each packet is synchronized to the 4.5 MHz of the bunch rate. This aspect will be ensured by the CC system by pipelining and resynchronizing the incoming data stream from the VETO unit before transmitting it further to the FEEs. In this process there are two slightly different schemes possible and closeable via configuration defined as variable and fixed latency protocol.

Fixed delay protocol

The fixed delay protocol was specifically designed to fulfill the requirements of the LPD detector, but might also be used for other systems if required. The special properties of this protocol are:

- The order of the bunch related information is from one to the last bunch in a train
- Each packet classifies the related bunch once and no change is possible
- The delay (latency) of the whole system consist of
    - a system dependent delay (VETO sources, VETO unit, CC, …) ΔT
    - plus a configurable delay allowance Δt, which depends on the detector requirement
- The number of packets transmitted matches the number of bunches in the train

Figure 3 shows an example of the timing relations, where one packet is represent as a single pulse and it's contend appended to it as text.

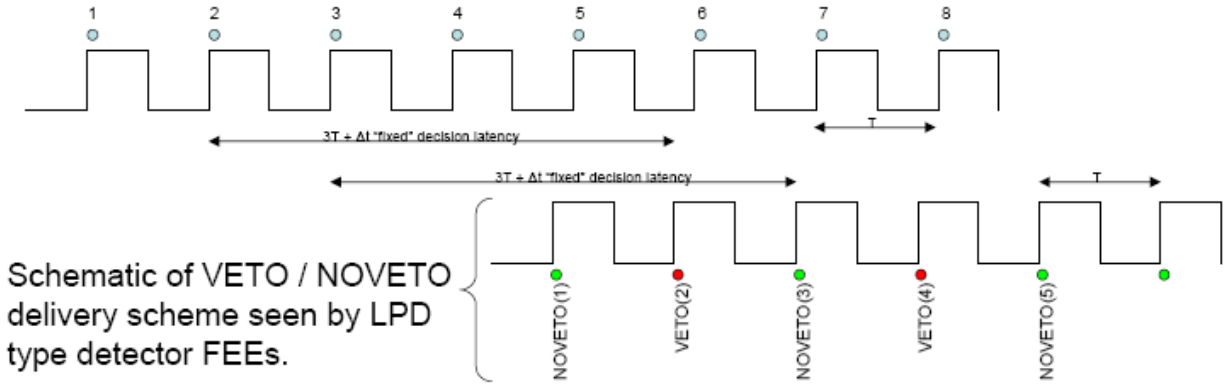

Figure 3: Example of the timing of packet delivery for the fixed delay protocol as seen by the LPD detector (taken from X).

Variable delay protocol

The variable delay protocol shares the aspect of synchronization to the 4.5MHz bunch rate with the previously described fixed latency protocol. However, the main difference to it is, that it allows independent delays of delivering individual VETO or GOLDEN event packets and therefore enables a FEE to make more efficient use of its buffer cells. The main aspects of this protocol variant are

- The order of bunches described in a packet could be non-monotonic (e.g. VETO 104 earlier than VETO 101)
- If there is no GOLDEN or VETO event to be sent, NO-VETO packets are sent
- NO-VETO bunch IDs are always increasing and marks the actual bunch counter
- A given bunch ID could be received twice (NO-VETO then VETO or GOLDEN)
- After a bunch was identified as GOLDEN or VETO no re-classification is possible
- Packet delivery is pipelined in CC (e.g. the VETO 104 and 101 may have arrived in the same time T slot and delivered in consecutive slots)
- The initial delay until the first packet will be transmitted will be as short as possible and defined by the intrinsic delays of the VETO system
- When the bunch ID of NO-VETO packets reaches the number bunches in the train the it will continue increasing the number
- The number of packets sent will be configured in the CC

Figure 4 illustrates an example of a transmission with the variable delay protocol as it will be seen by the AGIPD and DSSC detectors.

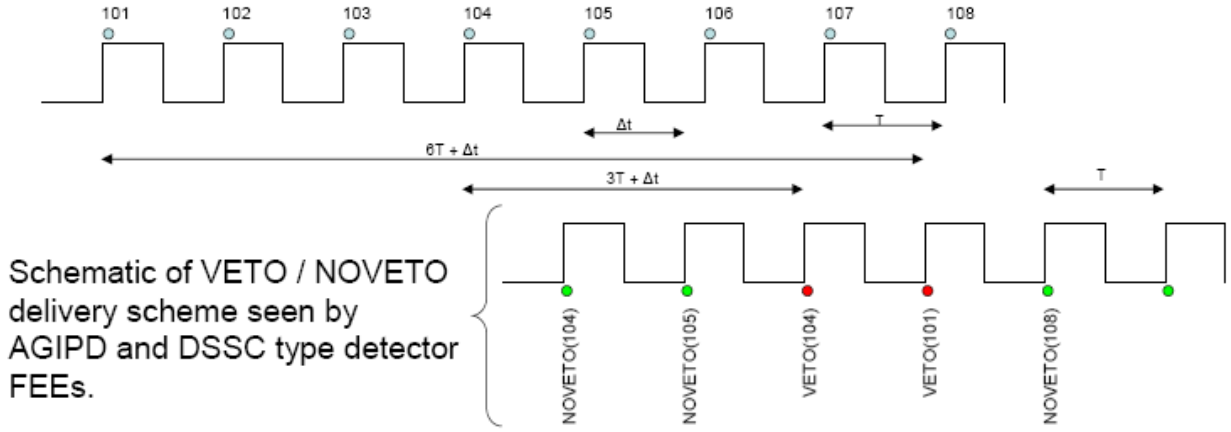

Figure 4: Example of the timing of packet delivery for the variable delay protocol as seen by the AGIPD and DSSC detectors (taken from X).

Appendix

Requirements and implementation for certain VETO users

This section provides an overview of the requirements related to the VETO usage and gives a short description of the internal implementation of the vetoing procedure.

LPD Detector

The memory write architecture of the LPD ASIC is shown in Figure 5. The amplified signals from the gain stages are buffered to an analogue memory to be stored. This analogue memory is 512 x 3 x 512 locations in size. This corresponds to 512 pixels with 3 gain values per pixel and 512 possible X-Ray bunches. The memory is addressed by a write pointer which enables each of the 512 memory frames in the pipeline to be written to by the gain stages via memory buffers. Every frame is written to at the same rate at which bunches arrive (4.5MHz) until the 512 frames are full and the pointer wraps around to the start of the pipeline where it begins overwriting the memory. For XFEL operation we expect ~ 3000 X-ray bunches (called a bunch train) at a rate of 4.5MHz. Clearly our memory depth of only 512 frames means we cannot expect to store data from all 3000 bunches, so we expect to receive a trigger from off chip to say which frames to keep. This is implemented by having a trigger pointer that follows the write pointer along the pipeline. The delay between the start of the write pointer and the trigger pointer will be determined by the latency of the veto system so control of both pointer start times are controlled from off chip through a high speed (99MHz) serial command interface. This same interface is used to send a trigger to the ASIC which is derived from the veto logic decision for that bunch (in combination with static bunch fill pattern information). When a trigger command is received by the ASIC the trigger pointer sets a flag in another register that then causes the write pointer to skip that location once it wraps around. In this way the data from the bunch that caused the trigger is protected from overwriting.
Once a location has been marked as to be skipped it can never be over written during this bunch train. Thus LPD only operates in the fixed delay protocol mode. The latency can in principle be set

to any value up to the maximum pipeline length of 512. However, due to the operation of the pipeline pointers the maximum number of usable locations is limited to 512 minus the latency value. If the number of triggers were to exceed this maximum the pipeline would become corrupted. Logic in the FEM will prevent this from happening. Therefore in practice the lower the latency can be kept the longer the effective pipeline that is available. If multiple veto sources are enabled they should be synchronised in the veto unit with the slowest source.

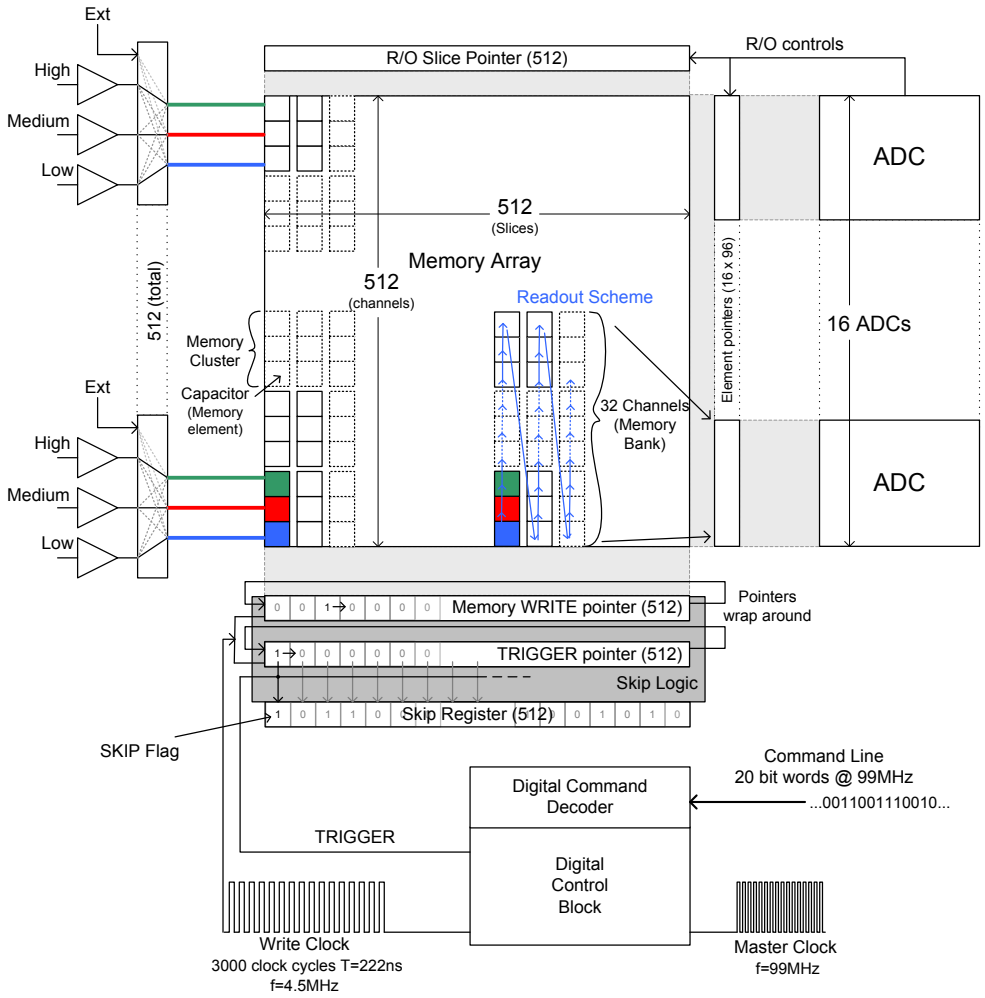

Figure 5 LPD ASIC Memory Write Architecture

DSSC Detector

This section will explain the specification of the DSSC veto strategy, and how it is implemented in the detector electronics.
The first part describes the general idea of the DSSC veto mechanism.
The second part gives a more detailed insight into the realization and implementation.

Overview of the DSSC VETO Mechanism

The DSSC VETO mechanism is implemented using a three-staged “veto chain”, which maps on the three readout electronics stages of the detector (Patch Panel Transceivers (PPT), I/O Boards (IOB), and ASICs).
A VETO telegram sent by the Clock & Control system (C&C) is received by the FPGA of the PPTs. The PPTs convert the VETO telegram into an ASIC specific command (*cmd_veto*), which is then transmitted through the IOBs to the ASICs.
On the ASICs, the VETO mechanism is implemented with a fixed latency (in count of events / bunches). That is, in order to realize a variable latency (0 to *max_latency* = 128), the *cmd_veto* must be delayed by *max_latency* - *curr_latency* cycles to hit and veto the requested bunch. The FPGA on the IOB must keep track of which bunch was vetoed during a train to provide the proper bunch ID information when transmitting the data to the Trainbuilder.

The current DSSC VETO strategy does not provide for GOLDEN events, and is similar to the one described in the section “Reduced implementation with limited storage” (with GOLDEN events ignored) of the VETO Specification Document. However, there are two major differences:

1) The vetoed storage cell is reused immediately.

2) There is no wrap-around in the address counter of the storage cells. Once the 700 cells have been used, no more frames are captured.

A VETO example

Suppose a machine setup, where 1000 bunches reach the detector. For simplicity, the bunch ID counter (as well as the ASIC storage cell address counter) increases by one, starting at “1”. During the first 100 bunches, no VETO telegram is send. With Bunch 101, a VETO is sent with bunch id #70. Since *max_latency* = 128, the transmission of the *cmd_veto* to the ASIC has to be delayed by another 128 - 30 = 98 cycles. That is, at bunch #198, the veto command is issued to the chip. At that point, the storage cell #70 of bunch #70 is reused for bunch #198. Accordingly, bunch #199 is stored in cell #198.
Now, another 100 bunches come in. At bunch #300 (cell #299), a new VETO is send for bunch #250. The *cmd_veto* is delayed by 128 - 50 = 78 events, which means that bunch #378 is stored in cell #249, and bunch #379 is stored in cell #377. The figure below displays the scenario described above.

					VETO #70							VETO #250
Addr	1	…	70	…	101	…	198	199	…	249	…	299
ID	1	…	70 101	…	101	…	199	200	…	250 378	…	300

Addr	300	…	376	377	…
ID	301	…	377	379	…

When the last cell has been written (cell #640), no further VETO will be processed. Since *max_latency* = 128, none of the last 128 cells will be reused, even if there could have been a VE-

TO received. This makes sense, as no further NO-VETOs could be processed either, as no more cells are available and the address counter is not implemented as a ring counter.

The VETO Mechanism inside the ASIC (SRAM Address Generator)

When the ASIC receives a veto command, it immediately overwrites an event with a fixed latency. The latency is variable only for entire bunches and can be programmed by slow control in between the bunches. The mechanism is implemented as follows: Before any vetoes are received, the SRAM is filled linearly by just incrementing the address. For each stored event, the address to which the event was written is put into a shift register with a variable length and the address is incremented for the next event. The length of the shift register is programmed by slow control. As the shift register is shifted for each stored event, the output of the shift register always points to the memory location to which the event which should be vetoed next was written. In case the ASIC receives a veto command, it takes the address from the output of the shift register, overwriting the event to be discarded. In this case it does not increment the address for the next event. The next event is therefore written to the address proceeding the address to which the event before the veto was stored. In case of several vetoes in series, the procedure is the same. The ASIC returns to the normal incrementing addresses only when it does not receive a veto.

AGIPD Detector

AGIPD receives the decision, that a bunch is rejected from the central vetoing system.

The requirements for hardware are:

- Signal transfer in a 4 pair cable
- AC- coupled LVDS signals on receiver side.
  - In: System clock ~99MHz
  - In: Data in:
  - In: Start of train etc.
  - Out: Status
- Each quadrant needs an input cable. (For 1Mega-Pixel, these are four)

The requirements for the protocol:

- The bunch number can be sent in a random order with random latency.
- The data-in should contain the bunch number to be vetoed and the same line should also be used to transfer other information like train number etc.
- The data in is a synchronous bit-stream to the system-clock and the bunches.
- For each bunch ZERO or ONE bunch can be rejected.
- Same cable for other command like train-start and train-end.

Comment: Early rejects on the time scale of a few bunch-clocks helps to make best usage of the storage cells especially at the end of the train, but it isn't a requirement.

Handling within the control part of the detector head:

An FPGA in the quadrant-part of the detector head, doing all the synchronous control, receives the signals within one bunch. During the next bunch the information can already be sent for the frontend ASICS in a translated format. The next bunch the freed storage cell is used. That is the fasted usage of a freed storage cell. If other cells are still not in use, the quadrant electronics it-self decides about the time for reusing a freed storage cell.

Parallel to the translation and transfer to the ASIC, the quadrant-FPGA fills bookkeeping tables, so that all times fast accesses for queries,

which bunch is in which storage cell,
which storage cell contains which bunch and
which storage cells are free,

are available.
The tables for the two first questions will be transferred together with the data to the train builder.
They will also be made available via the control link of the quadrant to the controlling system.

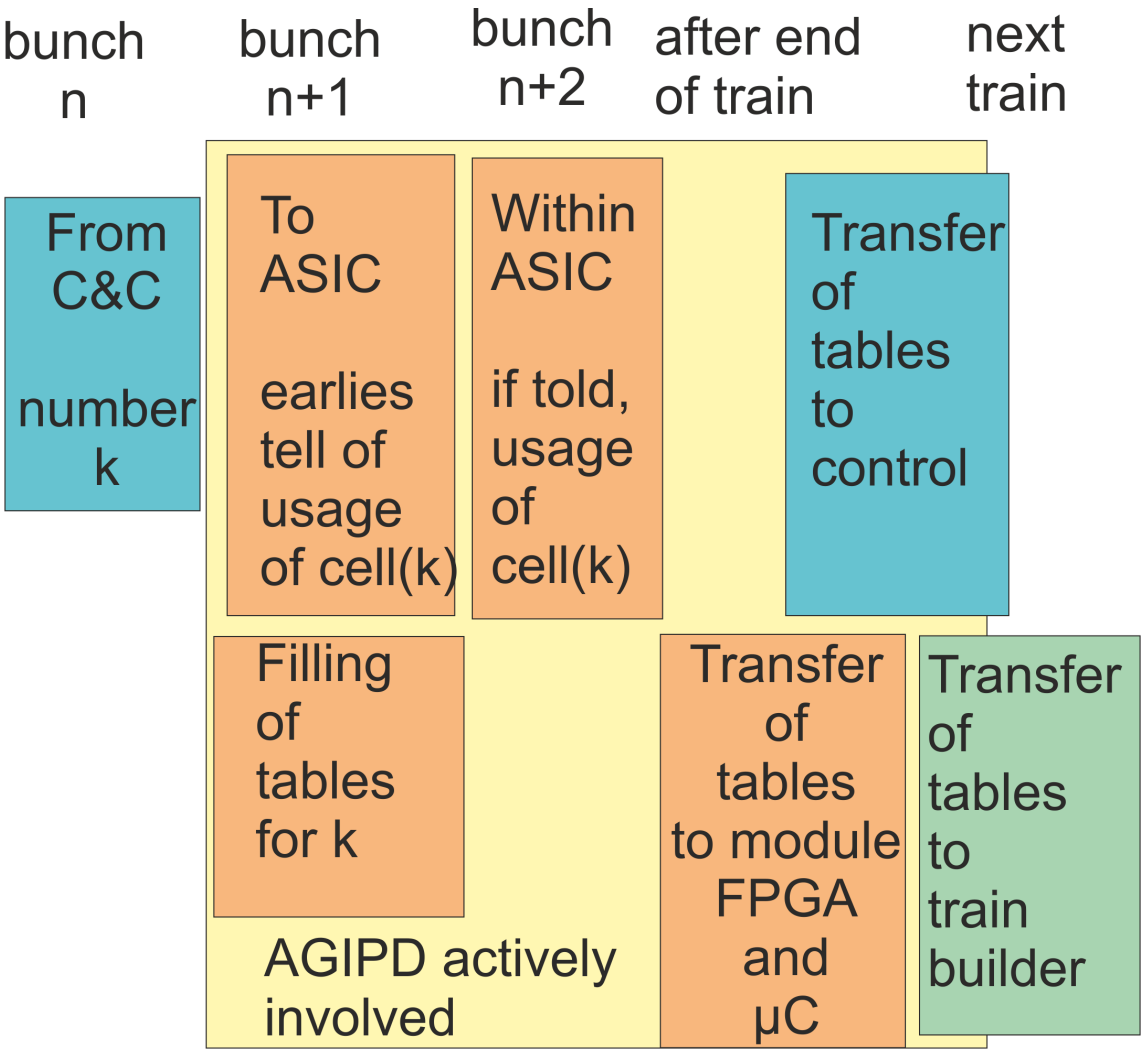

Figure 6: Timing of the reject within the detector head.