A trigger readout system for the Forward Calorimeter (FoCal)

by

Joseph Samuel Neyland

Master Thesis

Department of Physics and Technology
University of Bergen

November 2012

Abstract

The University of Bergen, Norway is currently involved in the development and building of the Forward Calorimeter (FoCal) as an upgrade to the ALICE (A Large Ion Collider Experiment) detector at CERN (European Organization for Nuclear Research).

In the case of particle detectors, as at CERN, trigger systems are utilized for data acquisition (DAQ). Data to be acquired pertains to charged particles passing through a charged particle detector. However, it is not desirable that data pertaining to absolutely all such events is data stored for further processing, as this would waste an incredible amount of memory and time. Therefore, a trigger system in this context must include a mask which can discard events which are considered uninteresting so that only relevant data is stored. At the same time, events of interest must be immediately allowed to be measured and stored. The focus of this work has been to develop the trigger readout system in the hardware description language called VHDL. The system allows a user to configure the aforementioned mask such that only desired events are recorded.

The result of this work is a trigger readout system which functions in simulation, yet remains to be tested and verified on physical devices.

Acknowledgements

The work described in this thesis was carried out within the Microelectronics Research Group at the University of Bergen, Norway between August 2011 and November 2012.

I would first like to thank my supervisor Kjetil Ullaland for his indispensable assistance and for sharing his knowledge with me throughout my work on this thesis, as well as being tough but fair.

I would also like to thank those who have assisted me in my work who also are involved in the development in the Forward Calorimeter: Shiming Yang, Dominik Fehlker and Johan Alme. They were available and helpful, while motivating me to learn and adjust to the challenges involved with this work. Thank you also to Christian Torgersen and Truls Andersen for providing the technical details of the Trigger Crate Interface.

A special thank you goes to my fellow students in the Microelectronics Research Group. It was a joy sharing the workspace in room 426 as well as the lecture hall with Thomas Bjørnsen, Augusta Pithlace, and Vegard Holsen. I wish them luck on the completion of their own work.

I must also thank Aleksander Simonsen, Thomas Bolstad, and Odne Hellebø, who were my fellow students from my bachelor studies in physics, who have been supportive and helpful; offering constructive criticism of my work in addition to inspiration through intellectually stimulating conversation.

Lastly, I want to thank my wife Sunniva Neyland, who always has supported me in my work.

Bergen, November 2012 Joseph Samuel Neyland

Contents

1 About This Work

The Forward Calorimeter (FoCal) is a proposed upgrade to the ALICE (A Large Ion Collider Experiment) detector. It will be located such that it will detect charged particles in the region just outside of the time projection chamber and near the beam axis. Its purpose is to contribute to quark gluon plasma physics and small x physics. The Forward Calorimeter's collaboration team includes physicists from many universities throughout the world. Here, at the University of Bergen Institute of Physics and Technology, contribution to the development of the Forward Calorimeter has been carried out by the microelectronics group and the nuclear physics group. I, as part of the microelectronics group, have been working on the development of electronics for the calorimeter. I entered the project in autumn 2011 while the physical detector tower was being developed.

The purpose of my work was to design a trigger readout system for the FoCal. It must be able to read in event data from a trigger detector and be able to be configured for pin sensitivity by a user. These two purposes result in the design of two main components which transmit data to each other across a physical serial link. The configuration mechanism and the readout mechanism must be able to occur simultaneously, and data must be able to transmitted between both components simultaneously. The trigger readout system was designed in VHDL. The design and its submodules were either designed "from scratch" by myself, or adapted for the system from Xilinx source code. The modules which I designed myself with interconnections and simulated are:

- The trigger data unit with all submodules
- The trigger register map
- The register control side A
- The communication protocol modules A and B with all submodules
- The serializing and deserializing components

The modules which were adapted for the trigger system whose source code was written by Xilinx are:

- The Hamming encoding/decoding modules
- The 8b/10b encoding/decoding modules

2 Introduction to the Proposed Forward Calorimeter (FoCal) Upgrade to the ALICE Detector

2.1 Purpose of FoCal

The Large Hadron Collider (LHC) at CERN buried deep under the ground level in Geneva, accelerates particles along two beams moving in opposite directions. These beams collide at four points where detectors are located. These detectors include the ATLAS, LCHb, the CMS, and ALICE. The proposed Forward Calorimeter would operate in conjunction with the ALICE detector, which is, as all of the other detector names, an acronym. This stands for A Large Ion Collider Experiment. The ALICE experiment aims to recreate the conditions just after the Big Bang in a laboratory environment through the collision atoms near the speed of light, which is meant to give physical insight in quark gluon plasma physics [5].

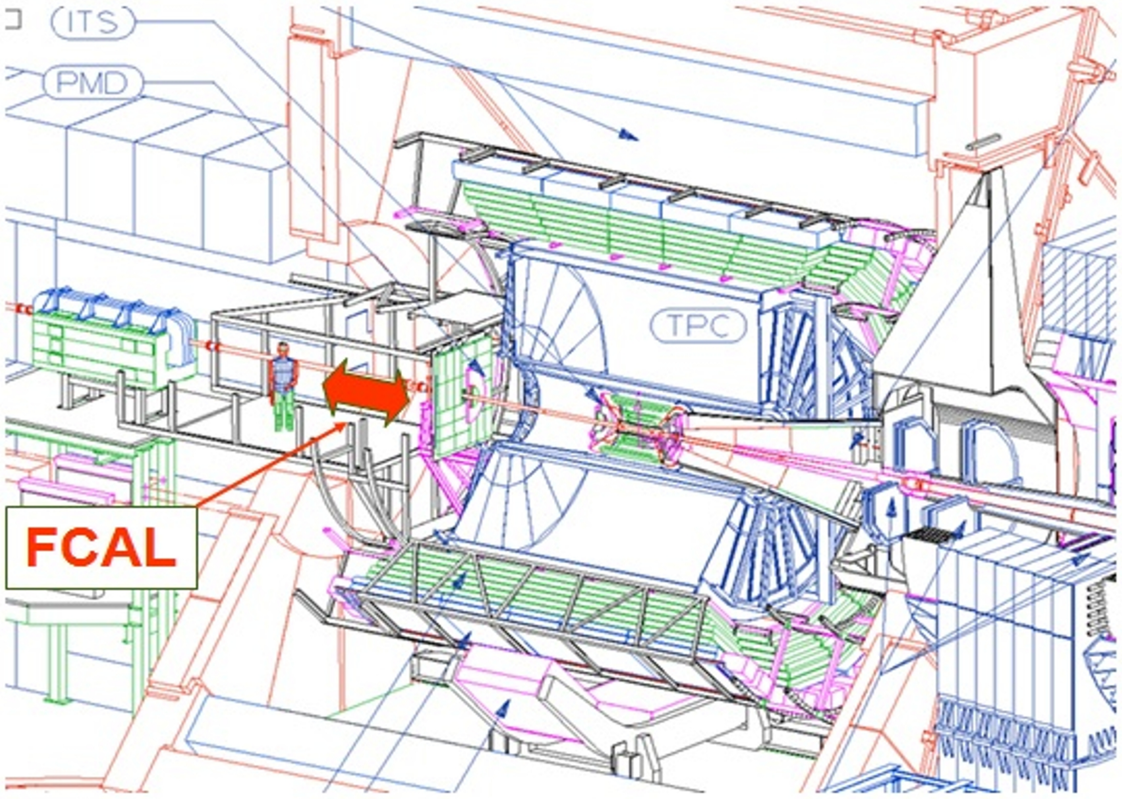


Figure 1: Possible placement of FoCal

The two purposes of the FoCal as an upgrade to the ALICE detector are (1) to gain further insight into quark gluon plasma physics (QGP) through the detection of collisions between lead atoms and (2) to contribute to small-x physics through the collisions of two protons or a proton and a lead atom [22]. Specifically, the FoCal will contribute to the further understanding of the inner structure of the proton and the lead nuclei, with special attention given to the Color Glass Condensate (CGC), which is a a component of the hadron wave function [9]. This will be possible to observe in extremely high energy collisions at LHC. The detector would be placed near the particle beam axis in the fluid rapidity region outside of the detection chamber in order to detect particles which, upon scattering, are reflected along or near the beam axis and back out of the detection chamber (see Figure 1).

2.2 Trigger Systems and DAQ

A trigger system's purpose is to indicate when relevant data is available when instructed to do so by a "trigger", which is often a physical event of interest. In electronics, trigger systems span a variety of uses. One of the most recognizable of these is that of the oscilloscope. An oscilloscope, which measures voltage as a function of time, can be set to wait for a certain event to occur within the system it measures and display that waveform on its output screen. If the oscilloscope is not configured to trigger on any particular event, it will display all changes in signals, often resulting in a jumbled waveform from which it can be difficult to gather information.

The most common and often-used form of triggering is called edge-triggering, which allows one to observe the basic timing characteristics and amplitude of a waveform [16]. This, as the name implies, uses a rise or fall in the measured voltage as a trigger for measurement. Rise and fall correspond to positive edge and negative edge-triggering. The voltage level at which triggering occurs, called the threshold, is able to be calibrated by the user. When the threshold is reached on an edge, the oscilloscope will display the waveform and the measurements immediately proceeding on the display screen depending on sensitivity to the positive or negative edge.

In the case of particle detectors, as at CERN, trigger systems are utilized for DAQ. Their main purpose is to maximize data for processes of interest with minimal wasted memory or time [10]. In this case, processes of interest are the detection of charged particles in a detector. However, it is not desired that absolutely all such events are measured and their relevant data stored for further processing. A trigger system in this context must include a mask which can discard events which are considered uninteresting so that less irrelevant data is stored. At the same time, events of interest must be immediately allowed to be measured and stored. The trigger detector in conjunction with the trigger readout system achieves this. An analog signal is detected by a trigger detector and can be calibrated for a threshold triggering level such that the event becomes a trigger for measurement.

2.3 Conceptual Design of FoCal and the Readout System

The physical Forward Calorimeter detector tower is composed of a sandwich structure of 24 alternating pairs of silicon and tungsten layers [3]. There are four

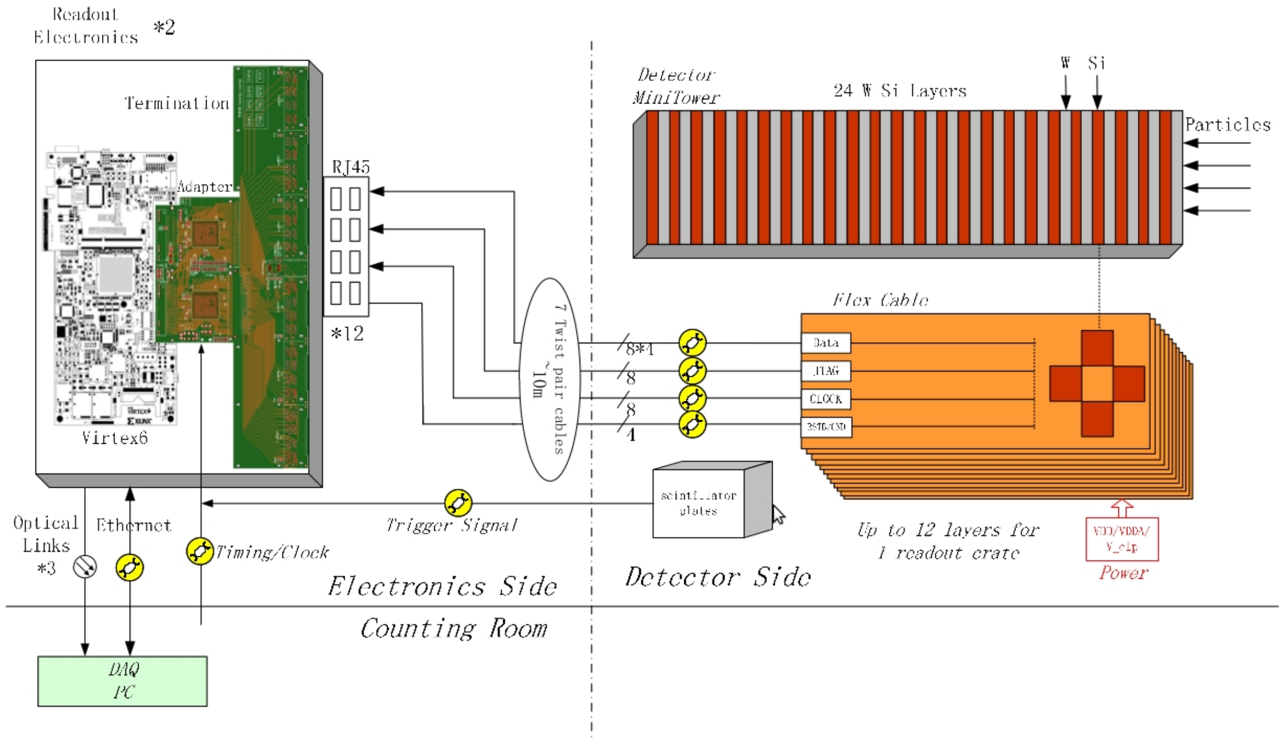


Figure 2: Physical overview of the FoCal readout system in its entirety [7]

MIMOSA pixel sensors on each silicon layer, giving a total of 96 sensors. Each pixel sensor contains a square array of 640x640 pixels. The readout electronics for each set of four Mimosa sensors require the connection of seven twist pair cables to the readout module. Four of these are for the data output for the four sensors, while the following three are shared between them: one for JTAG, one for clock and control, and one for chip reset signal and signal ground. These cables are up to ten meters in length and are connected to the readout electronics. For a physical depiction of this system, see Figure 2.

As is the case for trigger systems in general, a trigger-based readout system is necessary in the scope of the FoCal to be able to conclude exactly when a charged particle has been detected. This is due to the nature of the operation of the MIMOSA pixel detectors and their slow response to the presence of a charged particle. The MIMOSA chip constantly reads its own detector pixels for events using a rolling shutter mechanism, as explained in [6]. The pixel array is divided into 80 groups of 8-pixel columns, and each column is checked for events individually in succession (see Figure 3).

The rolling shutter mechanism begins by reading the column which resides furthest to the right in the pixel array, called"Frame N", and moves of to the next columns in succession. When the final column "Frame N + 79" is read, the shutter mechanism jumps back to the first column. Each column is "wiped clean" after each scan. This is repeated constantly, where the user is informed of the time when the first column is scanned in each full iteration. When an ionizing particle crosses

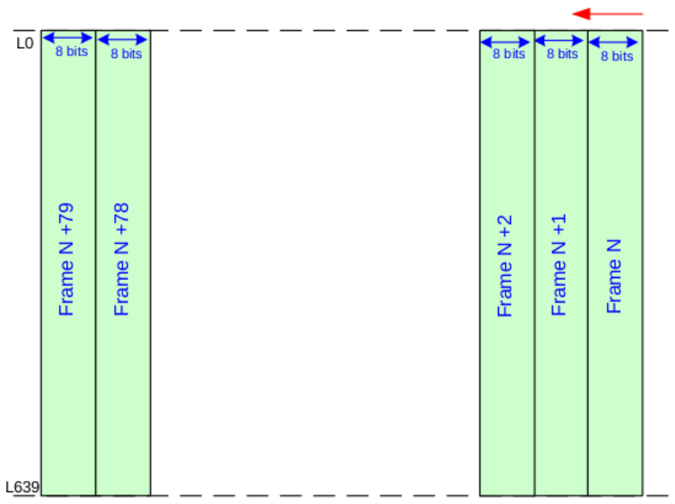


Figure 3: Diagram of the rolling shutter function in MIMOSA pixel array [6]

the silicon bulk, the generated electrons and holes move within the electric field, perpendicular to the wafer, toward the nearest electrode, where they are collected as a current pulse which is used for particle localization, see Figure 4. The readout frequency of the MIMOSA chip is 160 MHz on four differential pairs. This gives a line readout time of 1μs and a total readout time per chip of 640μs.

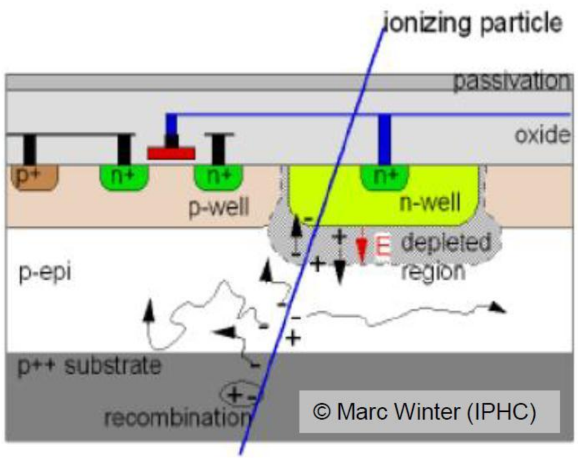


Figure 4: A charged particle passing through the silicon detector [21]

If a charged particle passes through a certain region of the pixel detector, it will not be registered by the MIMOSA chip until the readout pointer reaches that frame. The trigger detector, however, is made aware of such an event much more quickly. Therefore, the trigger detector informs of when an event has occurred and the MIMOSA pixel detector measures where an event has occurred.

The constant operation of the pixel detectors requires the use of trigger detectors to reduce the amount of data which is stored. As the time spent measuring no detection of particles is often vastly greater than the time in which events are measured, there is an equally large ratio of useless data to useful data being read from the MIMOSA pixel detectors. To tackle this, trigger detectors in the form of scintillator plates are placed at either end of the detector tower. The trigger detector detects events; allowing the opportunity to process the incurring data by sending the trigger signals to FPGAs. This will be further described in later sections.

2.3.1 Scintillators and the Readout Mechanism

The scintillators used in this system are the main initiating component in the trigger readout system. It is therefore necessary to understand the nature of the signals emitted by the scintillator upon detection of an event in order to configure the trigger card to respond to these and convey information.

The detecting material of a scintillator can either be organic or inorganic. Upon an event caused by, for example, a charged particle passing through the scintillator, it will excite the electrons or the molecules in the material depending on whether the scintillator is inorganic or organic. These excited states are then slightly reduced in energy within the scintillator before they fall back to their ground states, emitting another photon of lower energy. This photon is guided along the material and is eventually converted into an electrical signal through the photoelectric effect. This current is amplified before being discharged through a load resistor. An analog voltage pulse of a few mV is produced for a single photon, which is proportional to the energy of the incoming particle. This pulse is then widened in the time domain before it is sent to the trigger card. The trigger card will then receive this signal and convert it to a digital pulse. The change in analog pulse height can trigger the rise of the digital signal to logic '1' as specified by the user. This can also be configured to only detect positive change in pulse height, negative change in pulse height, or both changes in the pulse height. This corresponds to edge-triggering, which serves as the initiating element the readout mechanism within the scope of this specific trigger system.

2.3.2 The Trigger Crate Interface

The trigger signals convey the information that there has been an event. The FoCal detector presently uses a Trigger Crate Interface (TCI). The main task of the trigger system is not only to inform but to initiate the transfer of data from readout. An advantage of utilizing a trigger detector is that it responds more quickly than the pixel detectors do upon the detection of an event. If the data is not found to be of interest, then it is discarded. event data is transmitted in multiple formats corresponding to 36 inputs, which are summarized as follows [17]:

- 12 Low-Voltage Differential Signaling (LVDS) inputs
- 8 Emitter-Coupled Logic (ECL) inputs

- 8 Nuclear Instrumentation Standard (NIM) inputs
- 8 Positive Emitter-Coupled Logic (PECL) inputs

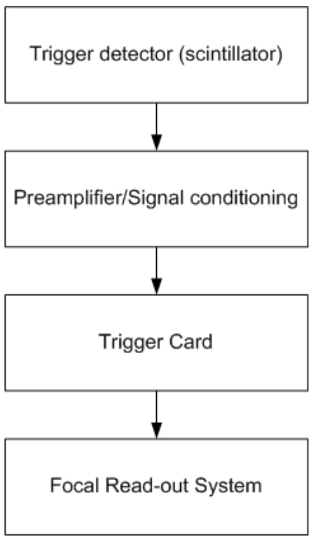

Trigger detector (scintillator)
Preamplifier/Signal conditioning
Trigger Card
Focal Read-out System

Figure 5: A charged particle passing through the silicon detector

The output pins of the trigger card convey a digital signal when an event is detected in the scintillator. The signal output from the scintillator is sent through the preamplifier/signal conditioning module within the TCI, which provides the digital signal in the trigger card to be read out through the Focal readout system (see Figure 5). Within the scope of the trigger system, these pins are configurable to detect '0' to '1' transitions, '1' to '0' transitions, or both transitions. When such an event is detected, the trigger readout system transmits corresponding data to a user.

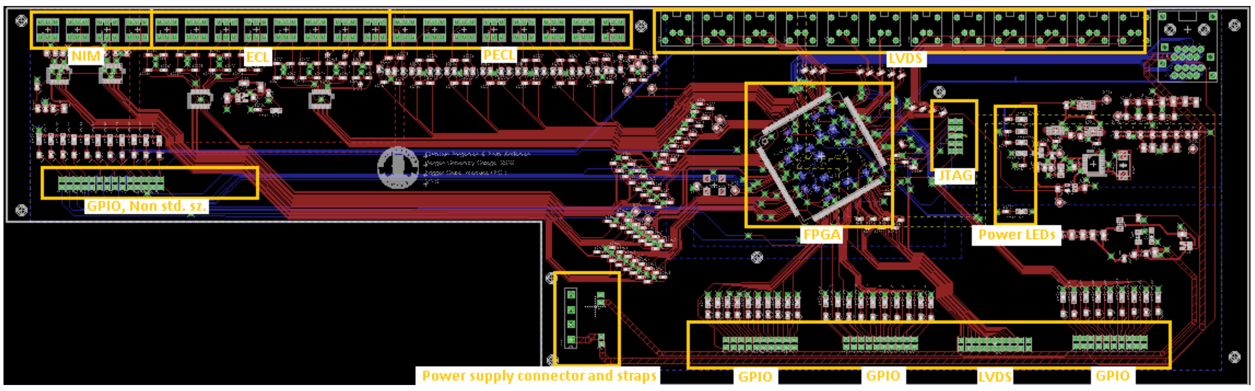


Figure 6: picture of trigger card with inputs and FPGA [17]

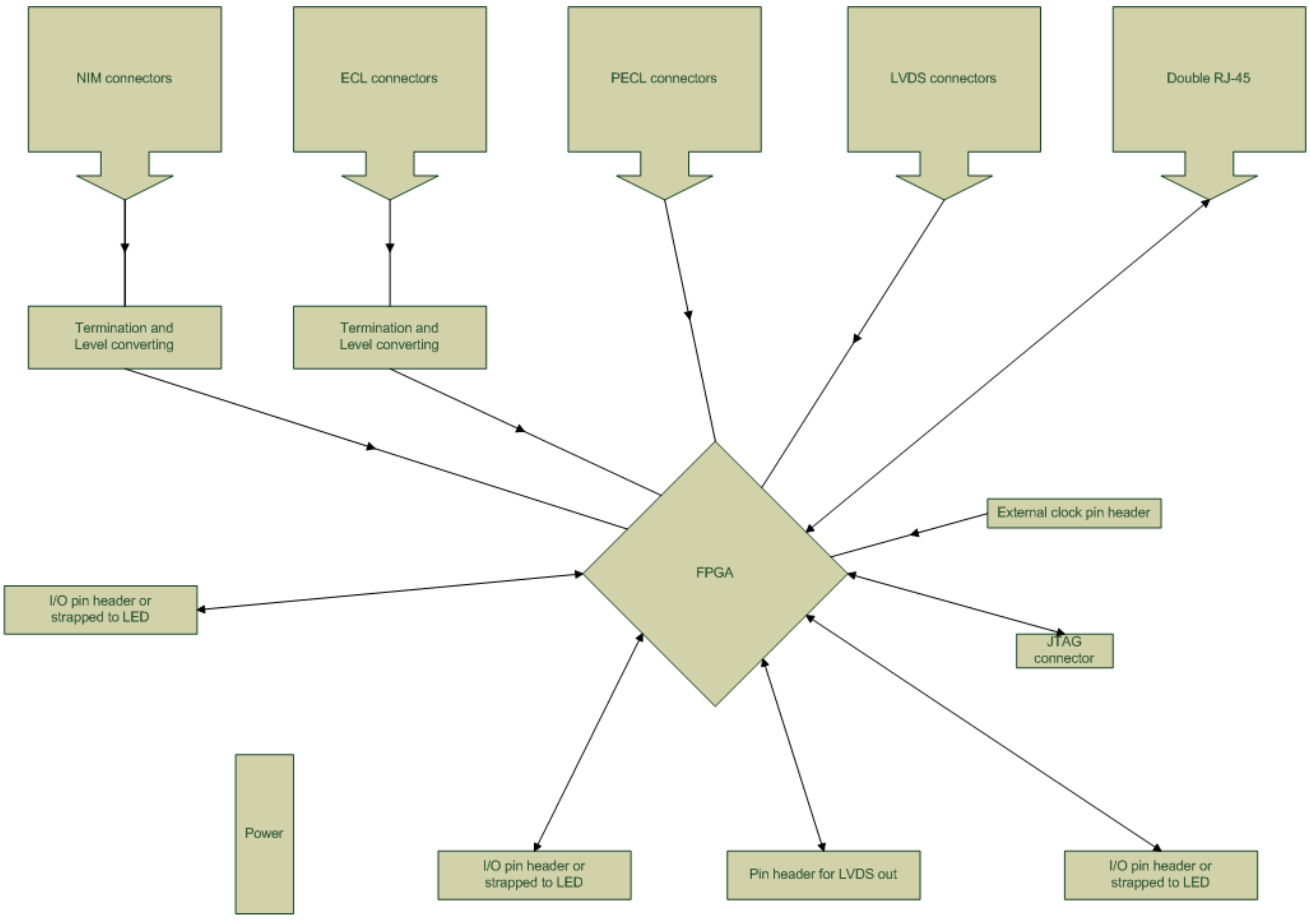

NIM connectors
ECL connectors
PECL connectors
LVDS connectors
Double RJ-45
Termination and Level converting
Termination and Level converting
External clock pin header
FPGA
I/O pin header or strapped to LED
JTAG connector
Power
I/O pin header or strapped to LED
Pin header for LVDS out
I/O pin header or strapped to LED

Figure 7: Overview of trigger card I/O and FPGA [17]

3 Design Approach and System Overview

The trigger readout system includes two basic units, which are the main unit where a user can define trigger parameters, and an extended unit where the programmable trigger interface card resides. These two units are referred to as A and B, respectively. They reside on each end of the the trigger readout system. In order to tackle the challenges associated with the design of such a system, a systematic design scheme is necessary. If these two units were adjacent to each other physically, they would be able to transmit and receive parallel data with minimal delay. However, the two units A and B will not reside adjacent to each other, but will be connected by physical links, as mentioned earlier. This corresponds to the physical layer in the OSI model, which requires the design of a communications system between the two units. Parallel data will require serialization before being transmitted across the physical link between units.

The International Organization for Standardization (ISO) Open Systems Interconnection (OSI) model serves to describe and define the functions of a communications system categorized in layers of abstraction. This is called the seven layer model. In the design discussed in this text, the first three layers of this model were defined and implemented, and are defined as follows (see for example [13]).

- **Level 1: Physical** - defines connector and interface specifications and medium requirements.
- **Level 2: Data link** - allows a device to access the network to send and receive messages, offers a physical address so that data can be sent on the network, interfaces with a device's networking software, and provides error detection capability.
- **Level 3: Network** - provides an end-to-end logical addressing system so that a packet of data can be routed across several layer 2 networks.

3.1 The Relationship between the Two Units

The relationship between the two systems must be defined in order to determine how the second and third layers of the seven layer model should be implemented. In the case of a master-slave relationship, one system commands another, transmitting commands along the physical link and then receiving a confirmation of transmission from the slave unit. In the case of this system, unit B sends event data to unit A upon the detection of an event. Unit A should then send confirmation of successful transmission back to unit B. This fits within the framework of master-slave communication from unit B to unit A.

However, unit A must be able to send configuration parameters to unit B. These parameters are used to configure the trigger card's I/O pins to detect specific trigger events. Unit B should then send confirmation of successful transmission of these parameters to unit A. This defines the master-slave relationship from unit A to unit B. Therefore, it is clear that both units function as masters and slaves simultaneously. The nature of the relationship between units A and B is then defined to be master-master. With this relationship defined, the implementation of levels two and three of the seven layer model can be more clearly defined.

3.2 Challenges Associated with the Master-Master Relationship

This sort of relationship presents a number of challenges. One challenge pertains to the need for the configuration of the trigger card and the transmission of event data to be able to occur simultaneously. This results in the necessity of an asymmetrical communications system. To tackle this challenge, a map of registers was designed in order to store configuration parameters and event data, called the Trigger Register Map (TRM). It is an array of registers containing 8 bits each. Specific addresses in the TRM are reserved for pin configuration of the trigger card while others are reserved for data pertaining to events which are sent by the trigger card to side A. This allows for the two types of data to be written simultaneously in the same unit while being organized and categorized by which type it is. The TRM has been chosen to reside on unit B because that is the side where the trigger card resides, which is to be configured by way of a neighboring module called the Trigger Data Unit (TDU). Specific details pertaining to the use of the registers in the TRM will be discussed later in the text.

Another challenge is to design the system such that they are able to transmit and receive simultaneously. This is not possible in a half-duplex communications system. To solve this problem, each unit contains a main communications protocol module A or B, each of which is divided into transmitter and receiver. Each protocol module can transmit data across its own dedicated serial link to the opposite protocol module's receiver. The transmitters and receivers within each protocol module are able to operate simultaneously and communicate with each other as well as their neighboring modules within units A or B.

3.3 Unambiguous and Robust Communication

The two units must be able to interpret which type of data is being sent or received in order to know what to do with said data. In other words, unambiguous communication is necessary. The protocol modules transmit the relevant data to each other in packets of identical length and structure for the sake of simplicity. These packets allow for unambiguous communication as they contain the following information:

- The type of data being transmitted or received (event data or configuration parameters).

- Whether the data contained in the packet is to be written to a register or if data is to be read from a register at a specific address.
- Whether the packet contains a first-time command or an acknowledgement of successful transmission.

As mentioned earlier, data must be sent from one unit to the other across a physical link. This gives rise to the possibility of bit errors associated with serial transmission. Thus, the inclusion of modules which can increase the robustness of data transmission is necessary for the prevention of transmission errors. As a result, units A and B include identical encoding/decoding (ENC/DEC) modules which are responsible for data encoding, decoding, serialization, etc. for transmission across the physical serial link. For a complete block diagram of the trigger system, see Figure 8.

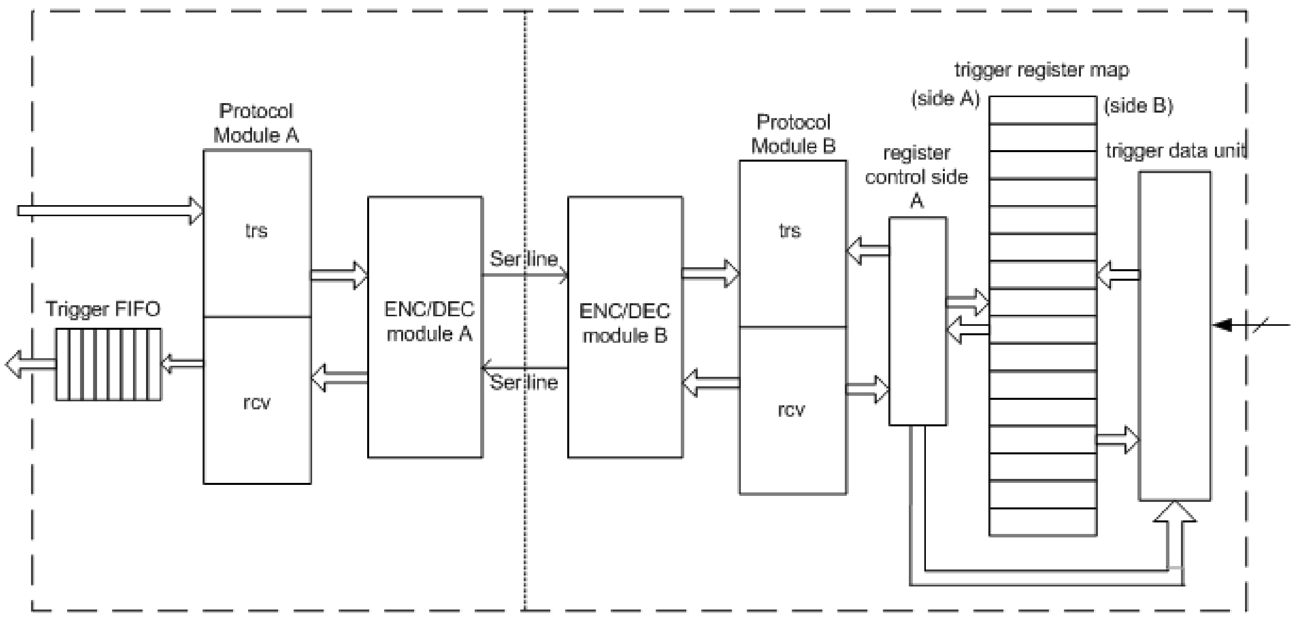


Figure 8: Block diagram of the specific designed communications system

If a packet is, for example, constructed in the transmitter of protocol module A, it is then sent to the encoding end of ENC/DEC module A which encodes and serializes the packet to be sent along the transmission line to ENC/DEC module B where it is parallelized, decoded, and checked for transmission errors before it is sent to the receiver end of protocol module B. The protocol module's receiver, depending on the detection of transmission errors, handles the packet as commanded before passing the packet to the transmitter. The transmitter, upon receiving this packet (now called the acknowledge packet), transmits this back through the ENC/DEC module B back to ENC/DEC module A along the other transmission line where it is once again parallelized and decoded before being sent to the receiving end of protocol module A, where acknowledgement is registered and recognized. This is when the entire data transmission process is completed and terminated.

3.4 Error Handling and Performance

As is the case with all digital systems, there is a certain probability that transmission errors will occur even after taking steps to prevent them. Therefore, this system must be able to recognize if a packet contains an error as well as account for unsuccessful transmission.

The system recognizes successful or unsuccessful transmission by waiting for acknowledgement of transmission. If unit A sends a packet to unit B, then unit A waits for an acknowledge packet, which confirms transmission. The minimum amount of time it takes to transmit a packet and register the reception of an acknowledge packet is 71 clock cycles. If the acknowledge packet does not arrive after 300 clock cycles, then the waiting unit concludes that there has been an error in transmission, and it ceases to wait for said acknowledge packet. This timeout error is logged in a status register, which counts timeout errors. Thus, the unit must send the original packet again. Both systems, when they transmit packets to each other, wait for acknowledgement packets for confirmation of successful transmission. However, this does not limit the word rate of transmission. Unit B, which sends event data, is able to transmit packets for the first time while simultaneously waiting for acknowledgement packets. Thus, the acknowledgement mechanism is not detrimental to the system in terms of word rate, but rather serves to verify successful transmission.

The system also is able to recognize bit errors in transmission. When a unit receives a packet, it simultaneously receives information from the ENC/DEC module in the form of two bits which indicate parity error. Parity will be discussed in a later section. If the packet contains parity errors, then the data is assumed to be corrupted and unable to be used. It must be sent again.

3.4.1 Event Word Rate and Event Transmission Performance

The rate at which event data can be sent is limited by the frequency of the transmission clock and the amount of bits to be transmitted across the physical link. The frequency of the transmission clock is 100 MHz. Each packet containing event data, after being encoded in multiple phases, contains 34 bits to be sent over the physical link. This results in a minimal transmit time of 340 ns per event data packet. The maximum event word rate is then calculated to be approximately 2.94 MHz.

The arrival of a trigger pulse into the system is a stochastic process. A trigger pulse may be measured at any time. Thus, a mechanism to tackle the possible rapid successive arrival of event data must be included in this system. Trigger event data is able to be read at any time, but there are, as discussed, limitations pertaining to the rate of transmission. In order to tackle this issue, a FIFO, which functions as a derandomizing buffer, collects all event data and is available at every clock cycle. This converts the stochastic process into a constant process. The FIFO can be written to or read from whenever the system requires to. The depth of the FIFO is chosen such that it can be emptied at a rate which corresponds to 10% of the time it takes for a MIMOSA chip to scan its entire pixel array, which is 640μs. 10% of the scan time, which is 64μs, is a reasonable worst case scenario for trigger readout. A FIFO which would require a maximum time of 64μs to empty with 340 ns word-reading intervals should have a maximum depth of 189. Therefore, a FIFO with a depth of 180 was chosen. To have very few slots in the FIFO would defeat its purpose, as it would be able to hold too few event data words.

4 Specific Designed Trigger System

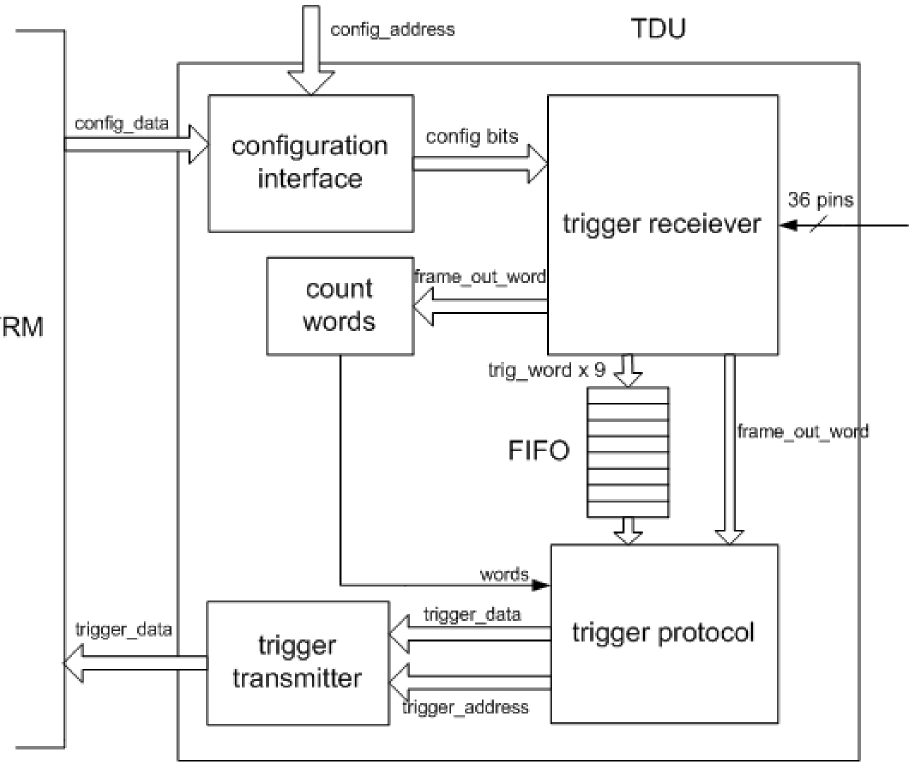


Figure 9: Block diagram of the trigger data unit in its entirety with external and internal I/O

4.1 Introduction to the Trigger System

As illustrated earlier in Figure 8, the TRM and the trigger data unit (TDU) reside on unit B. The purpose of the trigger system is to collect and transmit event data to the user on the opposite end of unit A. Event data is collected through the 36 pins of the trigger card, which are connected to the TDU shown in Figure 9. As mentioned, the user must also be able to enable, disable, and configure the edge sensitivity of the of the pins. This mechanism is achieved through the TRM, which contains these configuration parameters.

The flow of data through the trigger system is divided into two streams depending on the type of information to be conveyed. These streams can be followed by observing Figure 9. If data originates from the pins of the trigger card, then this pertains to event data. This data is read through the trigger receiver before being passed through the FIFO to the trigger protocol module which organizes each packet to be transmitted. These words are then passed to the trigger transmitter which places the data into the TRM. The event data is then accessible to the protocol module on unit B, which transmits this data across the physical link to the user on unit A. The other stream pertains to configuration parameters coming from the user on unit A. The configuration interface receives these parameters from the TRM, before they are organized and passed to the trigger receiver, where these parameters are implemented. Transmission is complete in this direction when the configuration parameters reach the trigger receiver. These two streams must be able to occur simultaneously, which is why there are two separate modules which interface with the TRM. These operate in parallel such that they do not depend on each other.

4.1.1 Challenges and Obstacles

The pins of the trigger card may convey event information at any moment without any specific relation to the system clock. However, the internal units of the trigger system operate synchronously. Therefore, the system must be able to read event data asynchronously and transmit these signals synchronously with the clock.

Since there are 36 readout pins, it is theoretically possible that every pin may convey event data at exactly the same time. Although this is highly unlikely, the trigger system must be able to divide this data into as many words as required before transmitting them to the user on unit A. At the same time, it is possible, and much more likely, that only one pin may convey any event data at a given time. In such a case, only one word is required to be sent. The words cannot be sent simultaneously over a physical link. They must be sent one after the other. In addition, there must be 340 ns of dead time between the transmission of each word in order to avoid congestion of data flow traffic in the trigger system. The system must be flexible enough to send up to as many as event data words required depending on the amount of data coming in, while wasting as little time as possible.

Each packet must be able to convey exactly what sort of event has occurred on a specific pin. The user at unit A must be able to receive a packet and, from it, conclude which pin or pins have detected a positive edge or negative edge. If one were to assign each pin two bits which would convey positive or negative edge events, then each packet would contain 72 bits of data, most of which would contain zeros; completely irrelevant information. This would be an ineffective solution. The trigger system, therefore, conveys as much information as possible in as few bits as is effective with the whole system taken into account.

4.2 18 bit I/O Packet Structure

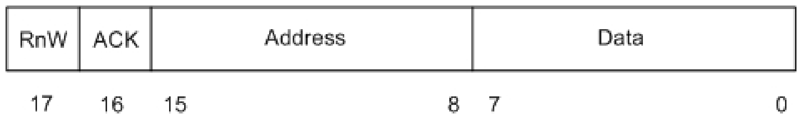


Figure 10: 18 bit packet structure

Each packet contains 18 bits which include an 8 bit data word, an 8 bit address location word, a read-not-write bit (RnW), and an acknowledge bit (ACK) (see

Figure 10). The inclusion of the two flag bits is a standard practice and is necessary for the interpretation of the data and address information being transmitted or received, see for example [14]. The RnW bit serves the simple purpose of informing unit A or B of that which is to be done with the input data and input address. If RnW is asserted, then data is to be read or has been read. Otherwise, data is to be written or has been written. The ACK bit allows the unit to know if transmission is completed before terminating transmission. Without the ACK bit, the receiving end of the unit would not be able to interpret whether the packet it receives is a first-time instruction from the opposite unit or if it is a confirmation of successful transmission that the opposite unit has sent.

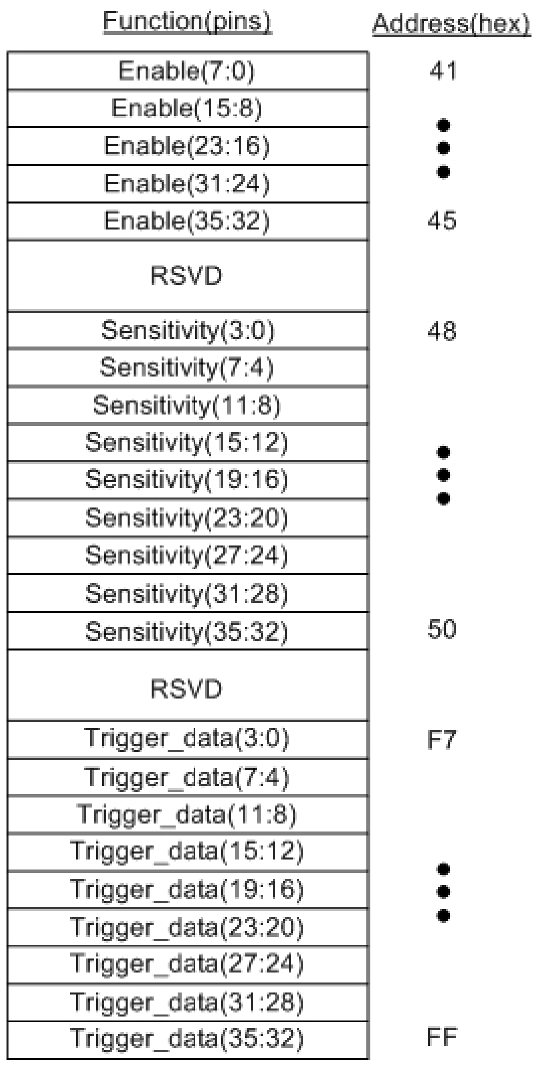
The data and address bits serve a dual purpose depending on whether the packet is transmitted from unit A or unit B. If this information has been transmitted from unit A, then it is meant to be used to convey configuration parameters. The data word can also contain enable information for each pin. In order for the side B to be able to know which pins are meant to be configured upon the reception of an 18 bit data packet, the 8 bit address word must carry this information. The address word serves this purpose, which is discussed in the scope of the TRM below. If an 18 bit packet is transmitted from side B, then it conveys event data, while the address corresponds to the pins which have detected such events.

4.3 The Trigger Register Map

The TRM is an array of 8 bit registers, with each address corresponding to specific sets or readout pins. The data and address information work together to be able to configure each readout pin individually. For the sake of discussion, the three forms of configuration will be henceforth referred to as enable, positive edge sensitivity (0 to 1 trigger pulse transitions) and negative edge sensitivity(1 to 0 trigger pulse transitions), or both, as discussed in Section 2.3.1. The user at unit A is able to write configuration parameters to or read configuration parameters from each register at any time.

Only one bit is required to enable a pin. Therefore, enable information for eight pins is stored in each 8 bit register. Each pin can therefore be given a specific index in an 8 bit data word read from a specific address in the TRM. Since there are 36 readout pins, five 8 bit registers are required to store all enable information for each pin. Thus, the enable information is stored in a 5 x 8 bit array where each row and column combination corresponds to one pin, with the exception of the four most significant bits in the fifth row. Bit value '1' corresponds to an enabled state.

For each pin, two bits are required to configure positive and negative edge sensitivity. This is because each pin can be configured to be sensitive to both changes as well as each one individually. Therefore, it is only possible for each register to contain sensitivity parameters for four pins, where each neighboring pair of bits corresponds to its own individual pin. As a result, nine registers are required for the



Function(pins)	Address(hex)
Enable(7:0)	41
Enable(15:8)	•
Enable(23:16)	•
Enable(31:24)	•
Enable(35:32)	45
RSVD	
Sensitivity(3:0)	48
Sensitivity(7:4)	
Sensitivity(11:8)	
Sensitivity(15:12)	•
Sensitivity(19:16)	•
Sensitivity(23:20)	•
Sensitivity(27:24)	
Sensitivity(31:28)	
Sensitivity(35:32)	50
RSVD	
Trigger_data(3:0)	F7
Trigger_data(7:4)	
Trigger_data(11:8)	
Trigger_data(15:12)	•
Trigger_data(19:16)	•
Trigger_data(23:20)	•
Trigger_data(27:24)	
Trigger_data(31:28)	
Trigger_data(35:32)	FF

Figure 11: A map of the TRM with address locations for configuration parameters and event data.

configuration of 36 readout pins. The two bits for each pin reside side-by-side, and if their form is expressed as [S1:S0], then the assertion of S0 will cause a pin to be sensitive to positive edge transitions and the assertion of S1 will cause a pin to be sensitive to negative edge transitions. S1 and S0 may be asserted simultaneously.

The final 16 register addresses are reserved for event data being sent from side B to side A. Event data is expressed in the form of two bits. These two bits can either be given the value "01", which conveys a positive edge transition, "10", which conveys a negative edge transition, and "00", which indicates no transition. These are stored in nine registers which correspond to each pin in a similar fashion as the sensitivity parameters correspond to each pin. The only difference is that each address which is designated to contain trigger readout data begins with the four binary bits "1111". The unit A communication protocol module is then able to use this flag to verify that the 18 bit packet it is receiving contains trigger readout information.

4.4 The Trigger Data Unit

The trigger data unit (TDU) serves a dual purpose. It communicates with the TRM to receive trigger parameter information from side A and it organizes the readout information received by each pin on the trigger card. This information is then prepared into an 8 bit data word and an 8 bit address word before being sent to a designated location on the TRM, as discussed earlier. These operations are able to occur simultaneously. The user at unit A will be able to configure anywhere between one and four adjacent pins simultaneously with one such packet. The discussion of the TDU will begin with the configuration of readout pins before continuing to the handling of the event data.

4.4.1 Configuration Interface

The configuration interface (CI) serves the purpose of reading configuration parameters from the TRM and conveying this to the trigger receiver. The interface is informed when new configuration parameters have been written to the TRM along with the address of the register. After reading the configuration parameters from the TRM at that address, it uses this this address to decide which pins shall be configured with these received parameters (see Figure 11).

There is a challenge pertaining to the different number of enable bits (four bits) per four pins, and the number of sensitivity bits (eight bits) per four pins. If enable information is to be, for example, configured on pins 3 down to 0, this is conveyed in an 8 bit data word sent to the designated address "01000001". These eight bits correspond to pins 7 down to 0, which include two different groups of pins. The CI has no way of knowing which half of the data within this register is "relevant" in terms of the user's intention at unit A. In order to avoid the necessity of the addition of a flag bit to determine which half of the word to send, the CI merely sends both halves in parallel. If there is no change in, for example, bits 7 down to 4, then there will be no negative consequence. However, if there is a change, then it must have been the user's intention to change them, causing no problem. The configuration of sensitivity words through the CI is simpler because all eight bits are required to configure one group of four adjacent pins. The CI simply sends these bits to the corresponding group of pins as specified in the address location of the register within the TRM.

4.4.2 Trigger Receiver Function

The trigger receiver is organized into nine readout modules to which four adjacent readout pins are connected. Each module, referred to as a "trigger module", contains four sets of two registers which store the configuration parameters for each pins. The configuration parameters are received from the configuration interface. As mentioned earlier, these pins deliver a digital pulse, which had been converted from an analog event pulse from the scintillator. Since these pulses arrive asynchronously, it is necessary for each pin to be able to be read in events at all times, as long as they are enabled. Thus, each pin is connected to a trigger sensor, which is depicted in Figure 12. The trigger sensor is sensitive to the configuration parameters described earlier, and, in summation, the trigger sensors act as a mask for incoming trigger pulses. The disabled pins will convey no information, while enabled pins will convey only the transition(s) to which they are configured to be sensitive. This is implemented in the AND gates depicted in the schematic. The configuration parameters are read through the two registers which the trigger receiver had received from the configuration interface.

The receiver is not only be able to read in event data, but also convey it through the FIFO and onward toward the trigger protocol module for organization. This mechanism requires the that the receiver must be ready to pass event information whenever an event is detected, along with some indicator of the pin which conveyed the event so that the trigger protocol module may be able to construct an 18 bit packet which contains this information.

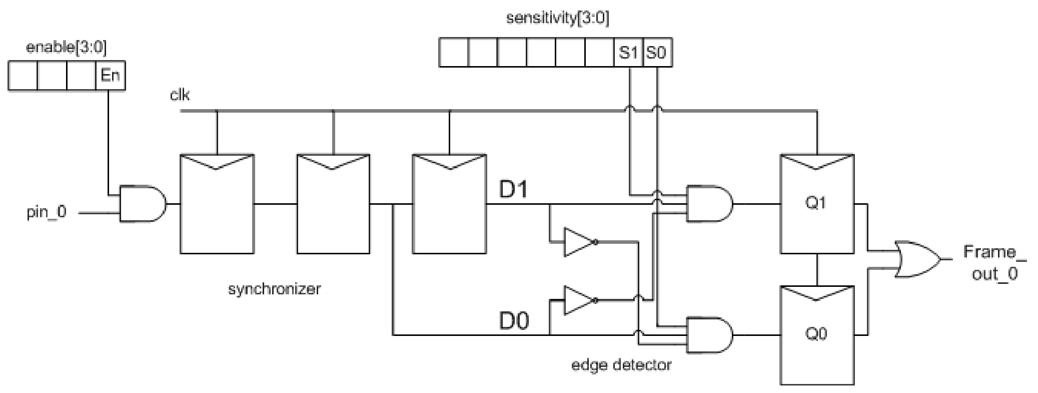


Figure 12: RTL schematic of one trigger sensor, where pin_0 is used as a specific example.

In order to avoid transmission errors associated with metastability, it is necessary to include a simple synchronizer; see for example [19]. This includes the first two clocked registers. Each pin is connected to an AND gate along with an enable bit before being clocked through two registers, as shown in Figure 12. The edge detector consists of a block of combinational logic with inputs D0 and D1, collected from the readout pin through the synchronizer. The detection of an edge corresponds to D0 and D1 having different logical values. A pulse may maintain a logic '1' for multiple clock cycles, and this must not trigger any readout of a transition. Therefore, the edge detector was designed to block any transmission of D1 and D0 while they have the same value, even if those values are '1' and '1'. However, if these values differ, and if that type of transition has been configured to be detected, then these two values will be transmitted through the combinational logic to Q1 and Q0 with the assertion of *frame_out* for one clock cycle. The *frame_out* bit is a handshaking signal that triggers the transmission of the values of Q1 and Q0 to the trigger protocol module. If an edge is detected on any enabled pin, an 8 bit word labeled *trig_out* composed of a concatenation of each of the four pins' Q0 and Q1 is transmitted at that time through the FIFO to the trigger protocol module for processing before transmission to unit A. There are nine such words corresponding to nine trigger readout modules.

A minimum of one *trig_out* word and a maximum of nine are able to be transferred to the FIFO at a time. However, it would be erroneous to transfer, for example, seven words containing only zeros if two *trig_out* words actually convey relevant event data. Therefore, only these words containing relevant event data are sent to the FIFO while the others are not. The trigger protocol module must constantly be able to send the words it receives from the FIFO, and it needs to be able to know how many words containing event data it needs to send. This is solved by sending a *frame_out_word* which is a concatenation of every *frame_out* bit in one instance of event data reception. Since each asserted *frame_out* bit corresponds to a word containing event data, the amount of ones in the *frame_out_word* equals the amount of words to be sent by the trigger protocol module. The amount of words is calculated in a separate module called "count words" (see Figure 9). This module returns an integer value with the number of words to the trigger protocol module. The trigger protocol module will send as many packets containing event data as the value it receives from "count words". This value can be updated as more triggers arrive.

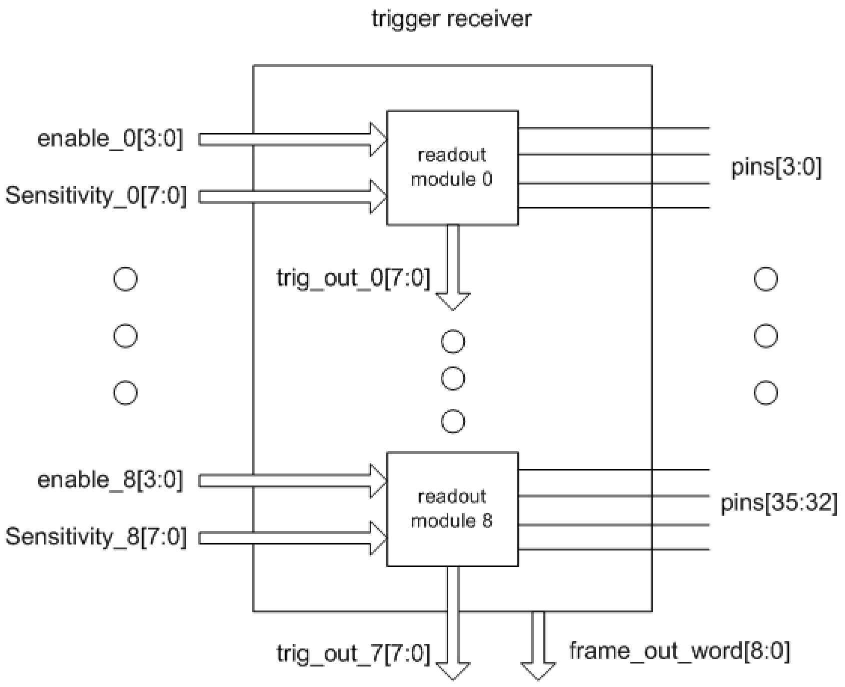


Figure 13: Block diagram of the trigger receiver in its entirety.

4.4.3 Trigger protocol module and trigger transmitter functions

The trigger protocol module collects the *trig_out* words from the FIFO and prepares these for transmission before sending each one-by-one until all words are sent toward unit A. The amount of words it sends depends on the word count received by the module "count words". The reception of such a word with at least one '1' in it triggers the handling and transmission mechanisms. It is not sufficient to merely pass a *trig_out* word received along to the trigger transmitter. It must be sent along with its corresponding TRM address so that the user at unit A may be able to conclude which pin(s) detected an event. This is achieved through a multiplexer-like function which chooses the address to be sent as a function of the the bit location of a '1' in the *frame_out_word*. The pin locations corresponding to event data words to be sent in the TRM can be seen in Figure 11. After calculation of the address, the trigger protocol module sends the event data and address to the trigger transmitter. However, this cannot occur for each clock cycle since this would congest data-flow traffic. In order to make sure that there is no such congestion, the protocol module waits for 34 clock cycles (corresponding to 340 ns) between the sending of each word. When finished sending all of these words, it is ready to receive more event data.

Event data and address are then transmitted to the trigger transmitter by the protocol module, whose sole purpose is to interface with the TRM for the sake of simplicity. It activates the TRM for the reception of the event data at the specified address which it received from the trigger protocol module. The TRM then informs the communications system that event data is ready to be transmitted across the system to unit A.

5 The Specific Designed Communications System

5.1 Introduction to the Communications System

In oder to transmit configuration parameters or event data from one side of the system to the other, a robust communications system is necessary. The communications system serves three basic purposes, listed below:

- Provide a protocol interface so that the two units can communicate with each other and interpret data packets.
- Increase the robustness of transmission through multiple encoding/decoding mechanisms.
- Allow for serial transmission between the two sides A and B.

These three purposes are fulfilled by the three main components within each unit A and B in the communications system. These three units are the protocol modules, the ENC/DEC modules, and the serializer/deserializer modules. The protocol modules serve the first purpose on the list above. Each protocol module serves as a mediator between the ENC/DEC modules and their respective outside units. The ENC/DEC modules provide the robustness of communication, and the serializer/deserializer modules allow for serial transmission along the physical link.

Since both units have their own master clock, there must be a sampling mechanism on either receiving end of the data transmission lines such that data may be received with minimal errors. This mechanism will be discussed later in the text. The register control side A (RCSA), which serves as a bridge between the trigger data unit and the protocol module on unit B, is also included in the discussion of the communications system and not the trigger system because it resides on the communication side of the TRM. It mediates the writing and reading of configuration parameters to/from the TRM as specified by the user on side A.

5.2 Register Control Side A

The RCSA serves to allow for the writing and reading of configuration parameters to the TRM as specified by the user on side A. The user must be able to both read from and write to the TRM, as it is possible that the user at side A could somehow lose or "forget" the configuration parameters they wrote to it. The writing mechanism of the RCSA must serve two purposes. It is necessary for it to inform the TDU that configuration parameters have been written at a specified address in addition to simply writing to or reading from the TRM. Figure 14 depicts the state machine which defines the control flow of the RCSA. When the FSM is in the "ready" state,

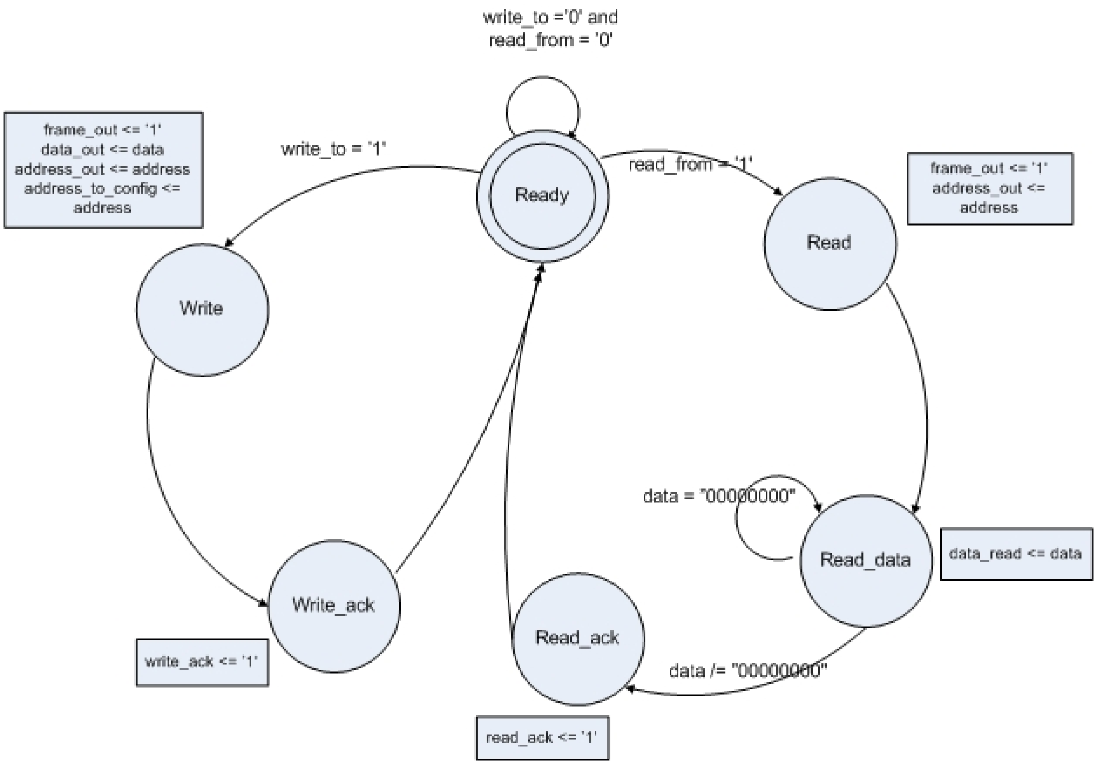

write_to ='0' and read_from = '0'
Ready
write_to = '1'
read_from = '1'
frame_out <= '1' data_out <= data address_out <= address address_to_config <= address
Write
Read
frame_out <= '1' address_out <= address
data = "00000000"
Read_data
data_read <= data
Write_ack
write_ack <= '1'
Read_ack
data /= "00000000"
read_ack <= '1'

Figure 14: The FSM for the register control on side A

it waits for read or write instructions from the protocol module B receiver. The FSM then executes one of these instructions by asserting control signals which the TRM obeys. If it receives instructions to write, it writes to the TRM in addition to sending the address to the TDU. Upon completion of reading or writing, the RCSA raises a *done* flag which informs the unit B protocol module that it is finished. If the RCSA received instruction to read from the TRM, it will read and send the 8 bit data contained within the TRM at the specified address along with the *done* flag back to the protocol module for transmission to unit A.

5.3 The Protocol Modules

Protocol modules are required in a system such as this to allow for the construction and deconstruction of data such that the two systems are able to interpret each other's commands at the same layer of abstraction, see for example [4]. Since the protocol modules must be able to send and receive simultaneously, the full-duplex solution was chosen. Thus, the protocol modules are divided into transmitter and receiver components with the opportunity for data flow between the two. In order to fully grasp the functionality of the protocol modules, they must be discussed in terms of their transmitter and receiver components individually.

The protocol modules require the reception of instructions in order to execute their purpose. The receiver and transmitter modules within each protocol module

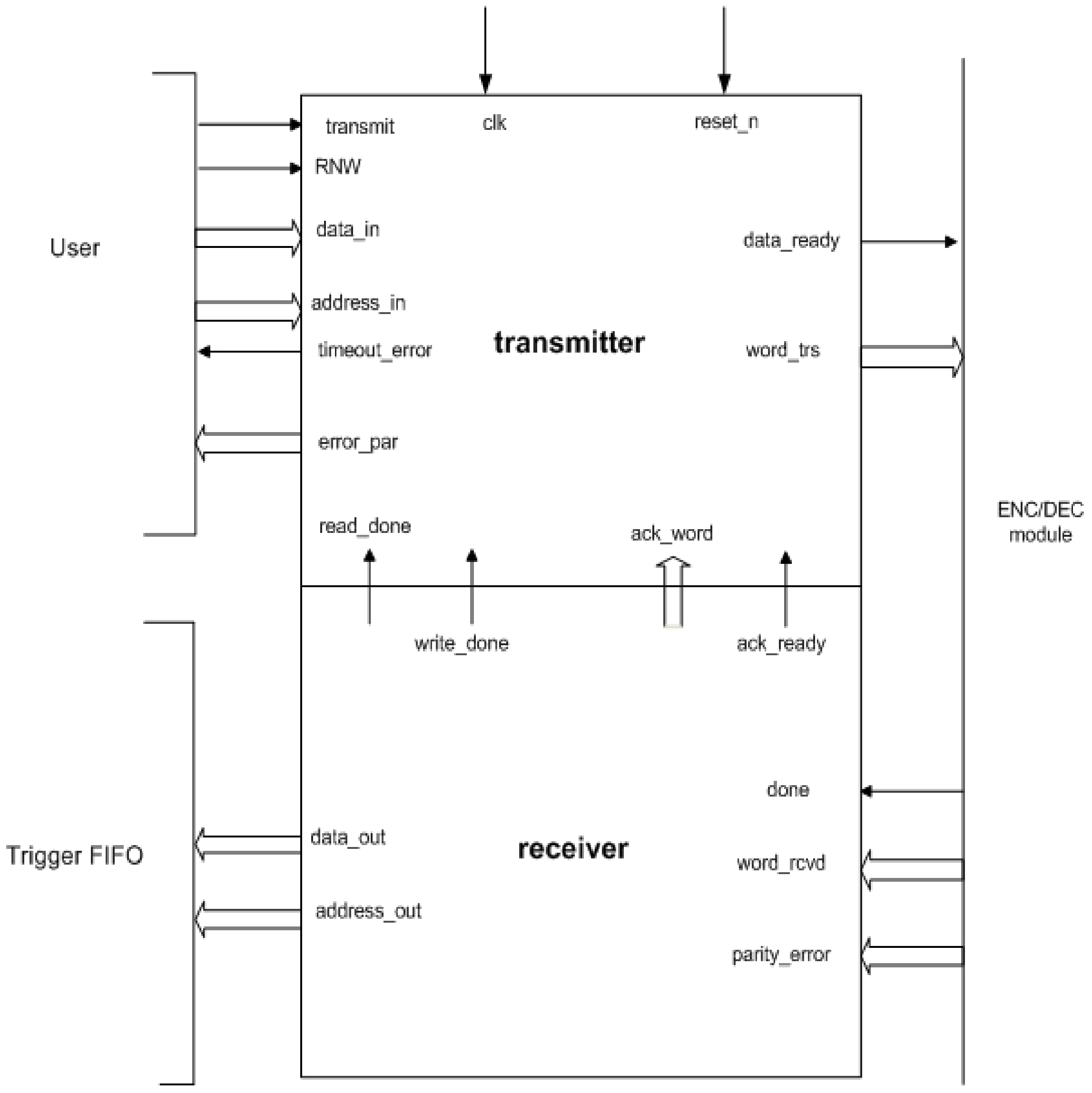


Figure 15: A general depiction of the protocol module with I/O signals

receive their instructions individually. The two protocol module transmitters can receive their instructions from either their immediate neighboring outside unit or the receiver within the same protocol module. The two protocol module receivers receive their instructions from the 18 bit packet which they accept from the opposite unit across the physical link. In order to account for errors in transmission, they are able to inform their masters of such transmission errors or timeout errors. The protocol modules A and B are not identical and their specific functionality must therefore be discussed individually. The following discussion will follow the flow of data through the two protocol modules by way of their transmitters and receivers. It begins with the protocol module A transmitter before continuing to protocol module B receiver, then to the protocol module B transmitter and back to the protocol module A receiver. Two directions of data flow are simultaneously discussed.

5.3.1 Protocol Module A Transmitter Function

The purpose of the transmitter in protocol module A is to organize and transmit data, address, and instructions to the opposite module as specified by the user on unit A. As mentioned earlier, the transmitter has four input connections with the user: an 8 bit address input, an 8 bit data input, a RnW input, and a control input called *transmit*. There are also output connections to the user, one of which, called *busy*, informs the user that new data and address are not ready to be received by the protocol module, since it is busy executing other tasks. Apart from these are the global clock and reset inputs. Lastly, there are four inputs into the transmitter from its own receiver. See Figure 15 for a diagram of the protocol module with I/O connections. The two signals which determine what is to be done with the data and address specified by the user are the RnW and *transmit* signals. Table 1 describes the transmitter's interpretation of the possible commands able to be given by the user on side A.

RnW	control	description
0	1	Write the configuration parameters to the specified address on the TRM
1	1	Read the configuration parameters from the specified address on the TRM
X	0	Do nothing

Table 1: A summary of the possible transmitter functions when commanded by the user at unit A

Upon the assertion of the *transmit* signal by the user, the protocol module accepts the RnW value, the data, and the address. The protocol module then assembles the 18 bit packet to be transmitted through the ENC/DEC modules. When data and address are received by the protocol module, it asserts the busy signal which will remain asserted until the entire course of transmission is complete and acknowledgement of successful transmission is received. After this initial transmission, the transmitter shifts to the "wait_w" or "wait_r" state depending on the value of the RnW bit (see Figure 16). The transmitter will remain in this state until it receives confirmation that the word was transmitted to the TRM. However, if for any reason confirmation of successful transmission is not received after a configurable amount of clock cycles, the protocol module interprets this as a timeout error and the user is informed of unsuccessful transmission. The *timeout_error* flag is raised, and the protocol module reverts to the "ready" state, with *busy* set to zero.

Confirmation of successful transmission is conveyed by the protocol module's own receiver unit by returning the original transmitted packet along with the assertion of the aforementioned ACK bit, which is found in bit position 16 in an 18 bit packet. The packet received, which is now called the "acknowledge packet" should be exactly the same as the packet transmitted with this one difference. If the value of the RnW bit of the acknowledge packet is '1', it is interpreted as having been of a read command. Therefore, the 8 bit data within the acknowledge packet contains the configuration parameters which were read from the TRM at the earlier-specified address. These configuration parameters passed to the user by the transmitter along with the assertion of the control signal *done_reading* for one clock cycle before shifting back to the "ready" state. If the acknowledge packet has an RnW bit of value '0', then this is considered to be a write receipt. In this case, the transmitter asserts the *done_writing* for one clock cycle to inform the user of successful writing to the TRM before returning to the ready state. When back in the "ready" state, the entire transmission process has been completed and the *busy* signal falls back to zero; informing the user that it is ready to receive new data and address information.

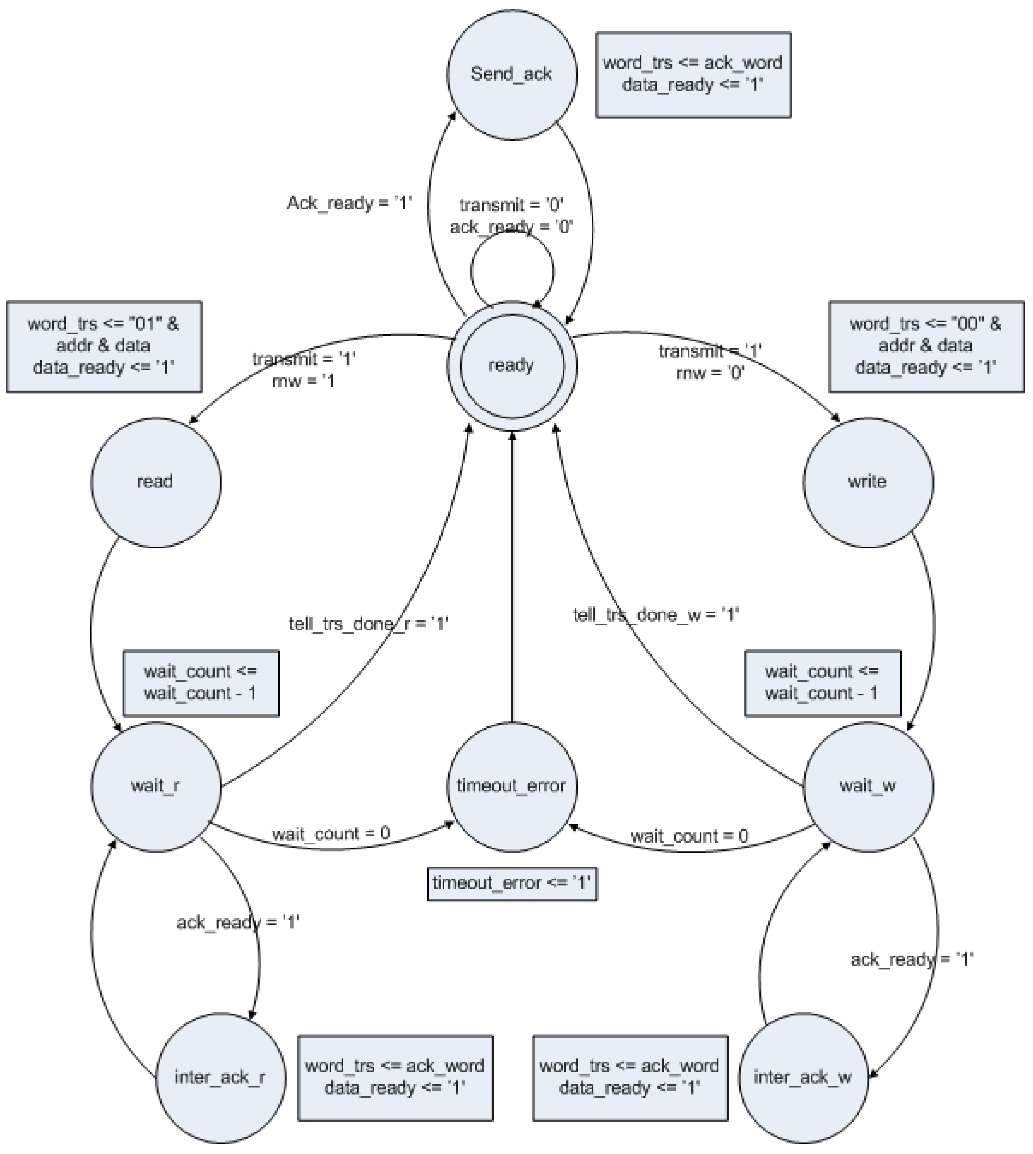


Figure 16: FSM diagram of the protocol module A transmitter

The transmitter may also receive its commands from its own receiver. This corresponds to the command to send an acknowledge packet to protocol module B. This is the case when the received packet originated in unit B. The transmitter must be available to transmit an acknowledge packet at all times in order to avoid unnecessary timeout errors at unit B. Therefore, when the transmitter has already sent a packet from the user and is waiting for an acknowledge packet (corresponding to the wait_w or wait_r state as depicted in Figure 16), it is available to transmit an acknowledge packet to unit B if the receiver indicates that one is stable and ready to be sent. Acknowledge packets are conveyed to the transmitter and the transmitter is commanded to transmit such a packet by way of the the assertion of the control signal *ack_ready* for one clock cycle by the receiver. The acknowledge packet, having been accepted by the transmitter, is then transmitted back to protocol module B. After this transmission, the transmitter continues to wait for an acknowledge packet corresponding to the original packet sent by the user.

bit[17]:RnW	bit[16]:ACK	description
0	1	Send writing done acknowledge packet to protocol module B
1	1	Send read done acknowledge packet to protocol module B

Table 2: A summary of the possible protocol module A transmitter functions depending on the values of the received RnW when ACK is equal to '1'

5.3.2 Protocol Module B Receiver Function

Now the discussion shifts across the physical link to the protocol module on unit B. The purpose of the protocol module B is not only to receive configuration parameters being transmitted from the user on unit A, but also to convey these parameters to the neighboring TRM through the RCSA. It receives its instructions exclusively from the RnW and ACK bits from the 18 bit packet it receives. In a similar fashion as the transmitter, the receiver has four possible instructions determined by these bits.

The receiver is sensitive to a handshake signal called *done* in the same way that the transmitter is sensitive to the control signal *transmit*. It is generated by the neighboring ENC/DEC module when an 18 bit packet is ready to be passed to the receiver. Upon the assertion of *done*, the receiver collects the transmitted packet as well as a two bit word which indicates parity error. Parity will be discussed later in the text. After the transmitted packet is received, the protocol module interprets 17 and 16 as the RnW and ACK bits.

It is simplest for the sake of discussion to first describe the receiver function for first-time reception, namely when the ACK bit is equal to '0'. In this case, the receiver will then shift its state to "write_to" or "fetch" depending on whether the RnW is equal to '0' or '1', respectively (see Figure 17). In each of these states, corresponding instructions are sent to the TRM through the RCSA. The "write_to" state conveys the configuration parameters to the TRM to be written at the specified address along with the assertion of the control signal *write_to*. The "fetch" state sends the address and the control signal *read_from* to the RCSA and waits for the it to provide the configuration parameters from the specified register address. Upon

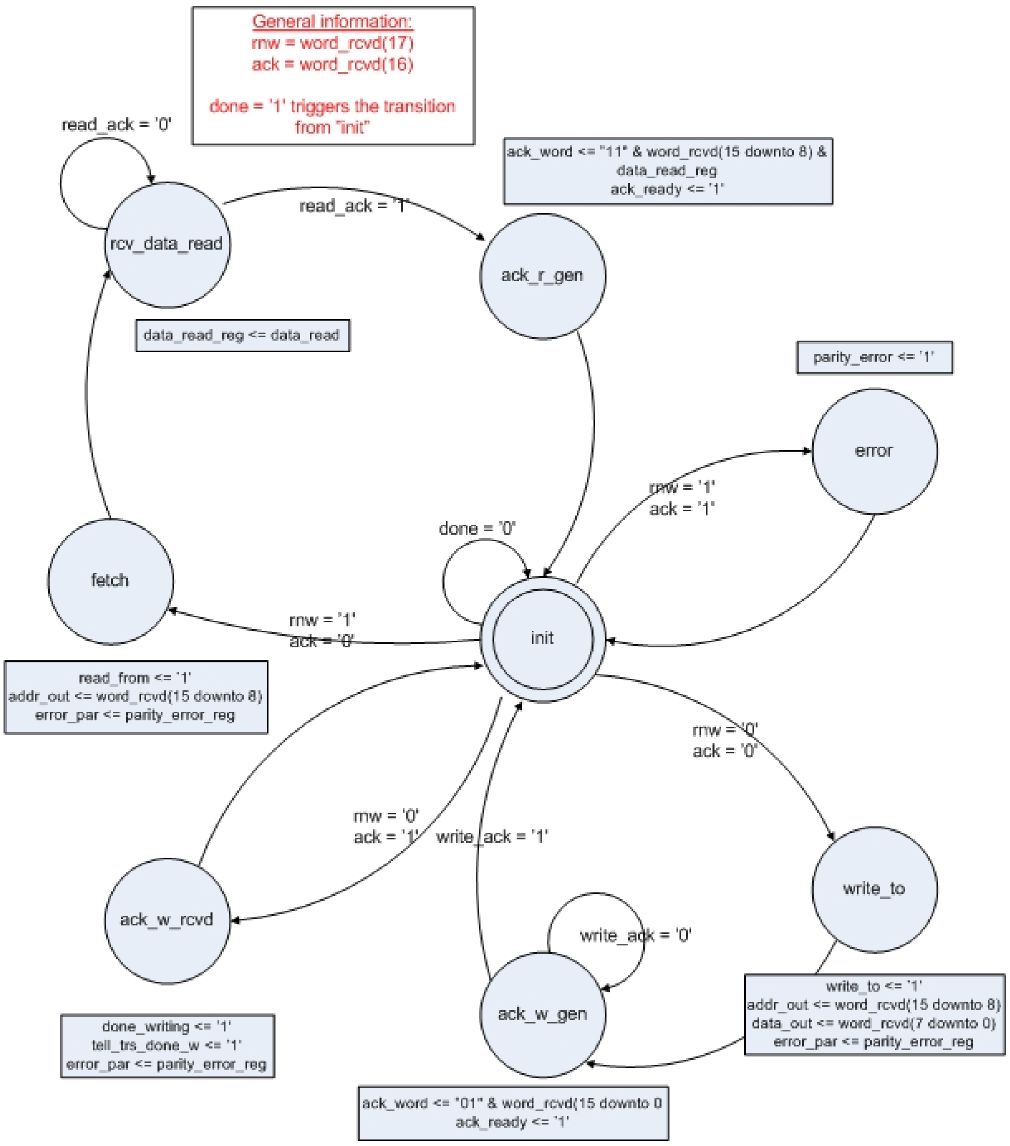


Figure 17: FSM diagram of the protocol module B receiver

completion of either one of these states, the state machine shifts to another state in which the newly concatenated acknowledge packet is passed to the transmitter with the assertion of the *ack_ready* signal to be processed as described earlier. After this is completed, the state reverts to "ready" and more 18 bit packets can be received.

If the ACK bit within the packet received is equal to '1', then the packet being received is the acknowledge packet corresponding to a packet which already had been sent by the trigger system earlier. Depending on the value of the RnW signal, the packet is processed by way of the "ack_w_rcvd" or the "error" state before returning to the "ready" state. It is not possible for the protocol module B receiver to receive an acknowledge packet pertaining to successful reading from unit A. This is because there simply is nothing to be read from unit A. Therefore, if the acknowledge packet received contains RnW and ACK bits which are both asserted, the protocol module state machine will interpret this as a result of a parity error. The instructions within a received packet are interpreted by the receiving end of the protocol module as listed in Table 3.

bit[17]:RnW	bit[16]:ACK	description
0	0	Write the received configuration parameters to specified address in the TRM. Assert ACK bit and pass the acknowledge packet to the transmitter.
0	1	Finished writing to the trigger FIFO neighboring protocol module A. Check parity and inform transmitter that writing is complete.
1	0	Fetch data from received address in the TRM and wait until finished. Assert ACK bit afterwards and pass the acknowledge packet containing configuration parameters to the transmitter.
1	1	ERROR: it is not possible within the logic of the system to receive the flag bits "11" as it is not possible for unit B to read from unit A.

Table 3: A summary of the four possible protocol module B receiver functions depending on the values of RnW and ACK

5.3.3 Protocol Module B Transmitter Function

The protocol module B transmitter differs in multiple ways from the transmitter in protocol module A. For the simplicity of this discussion, its function will be explained for event data transmission and acknowledge packets separately. Upon the transmission of event data and address information, which it receives from the TRM, the RnW bit will never be asserted, as side B never has the need or possibility to read any data from side A. Thus, protocol module B will never be given the command to read data from protocol module A, but will only write event data and address information to the user on side A through the aforementioned trigger FIFO connected to protocol module A.

The transmitter may also send send an acknowledge packet back to side A. There are two types of such packets that it may send, depending on the instructions it originally received from unit A. If protocol module B's receiver was commanded to write configuration parameters to the TRM, then it will send an acknowledge packet whose purpose is to communicate that the parameters have been successfully or unsuccessfully written. If successful, it will contain exactly the same bits as the originally sent packet with along with the assertion of the ACK bit. If the receiver was instructed to read from the TRM, then the acknowledge packet will contain the configuration parameters which were read from the TRM along with the address specified by the user at unit A. The reading and writing mechanism is, however, overseen by protocol module B's receiver. Therefore, the transmitter merely sends the acknowledgement packets it receives from its own receiver. As a result, the instruction set in this transmitter does not depend on any received RnW value from

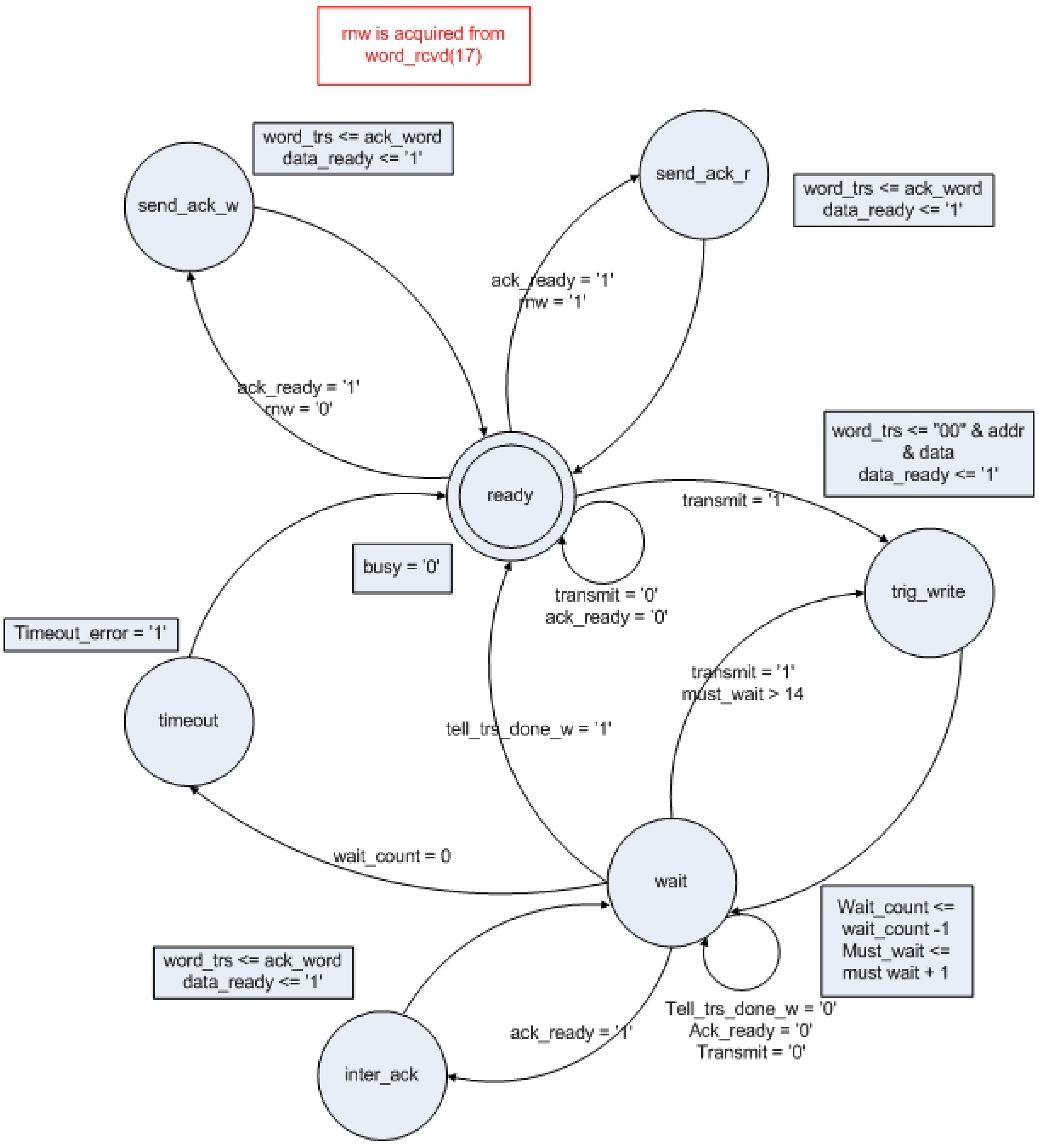

rnw is acquired from word_rcvd(17)
word_trs <= ack_word data_ready <= '1'
send_ack_w
send_ack_r
word_trs <= ack_word data_ready <= '1'
ack_ready = '1' rnw = '1'
ack_ready = '1' rnw = '0'
word_trs <= "00" & addr & data data_ready <= '1'
ready
transmit = '1'
busy = '0'
transmit = '0' ack_ready = '0'
trig_write
Timeout_error = '1'
transmit = '1' must_wait > 14
timeout
tell_trs_done_w = '1'
wait_count = 0
wait
Wait_count <= wait_count -1 Must_wait <= must wait + 1
word_trs <= ack_word data_ready <= '1'
Tell_trs_done_w = '0' Ack_ready = '0' Transmit = '0'
ack_ready = '1'
inter_ack

Figure 18: FSM diagram of the protocol module B transmitter

the outside, as is the case with protocol module A, but rather upon the control signals *transmit* and *ack_ready*. The response of the transmitter to these signals is summarized in Table 4.

As is the case with the protocol module A transmitter, the B transmitter asserts the *busy* signal after having sent event data and address to the trigger FIFO. *busy* will fall back to zero when the transmitter receives confirmation that the transmission is complete. This is communicated through the acknowledge packet in the same fashion as in protocol module A. If an acknowledge packet does not return after a certain number of clock cycles, then it is assumed that there were systematic errors which interrupted transmission of data. Thus, the *timeout_error* flag is raised for one clock cycle before returning to the "init" state. The *busy* signal has the same function as in A, but not the same purpose. Event data may be sent through the protocol module B transmitter even while the *busy* signal is asserted. This is

transmit	*ack_ready*	description
1	0	Write event data and address to the trigger FIFO.
0	1	Send the read configuration parameters along with address or the writing done acknowledge packet to protocol module A.

Table 4: A summary of the two possible protocol module B transmitter functions depending on the values of transmit and ACK

to accommodate the fact that event data may be measured and ready to be sent with short intervals. Thus, the *busy* signal in protocol module B serves a different purpose; namely to indicate to the interested user when transmission is currently occurring.

In order to allow for the transmission of acknowledge packets at any time, the "wait" state, as pictured in Figure 18, serves a similar purpose as the wait states in the protocol module A transmitter. It waits for an acknowledge packet before reverting to the "init" state. However, it is also able to shift to the "inter_ack" state such that acknowledge packets may be sent at any time; even while waiting for an acknowledge packet from the transmission of event data. However, there is a difference between protocol module A and B in terms of the function of the "wait" state. Since event data can arrive at any time, it is necessary that the transmitter be able to send such event data as often as possible. It cannot send event data on every clock cycle, as this would congest the traffic of data flow. It can, however, send event data after a minimum wait time of 14 clock cycles. This value was found in simulation. Thus, within the "wait" state, a clock cycle counter begins and, when it reaches 14, is able to shift to the "trig_write" state, where event data is transmitted to unit A.

5.3.4 Protocol Module A Receiver Function

The protocol module A receiver has already been discussed in part while describing the protocol module A transmitter. It receives its instructions from the RnW and ACK bits of the received packet in the same way that protocol module B receiver does. The instructions to the protocol module A receiver are summarized in Table 5.

The ACK bit determines whether the packet being received contains event data or if it is an acknowledge packet. If it is an acknowledge packet, then the outputs of the protocol module differ according to whether the RnW bit is equal to '1' or '0'. If RnW has value '1', then the packet contains configuration parameters read from the TRM. This data is sent out to the user along with the assertion of the *done_reading* flag for one clock cycle and transmission is terminated.. If RnW is not asserted then the packet is considered to contain confirmation of successful writing to the TRM without containing any more valuable information to the user. Thus, the *done_writing* flag is raised for one clock cycle and transmission is terminated. In both

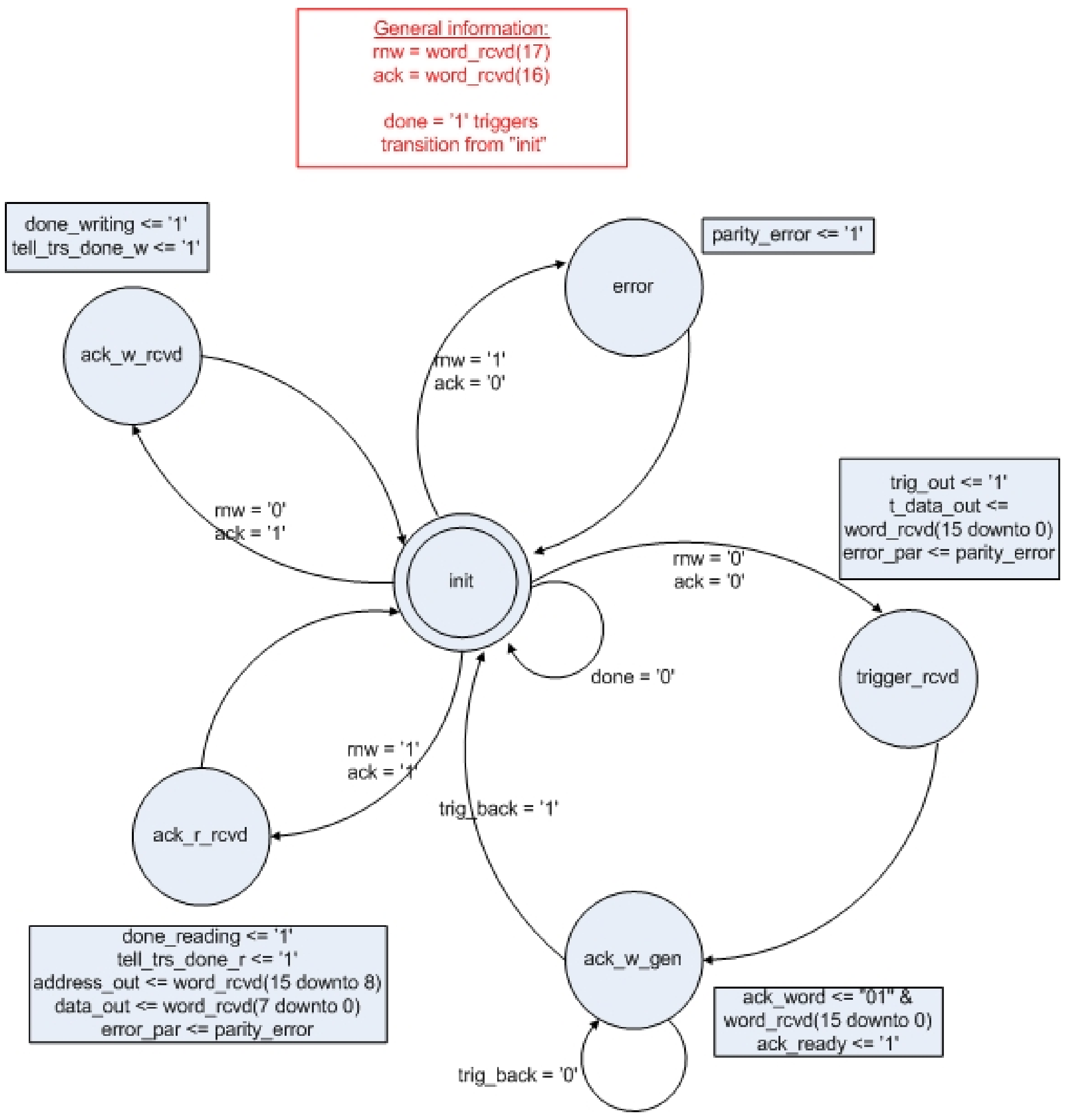


Figure 19: FSM diagram of the protocol module A receiver

cases, the protocol module's transmitter is informed of successful transmission with the assertion of a *tell_trs_done* flag if there are no parity errors detected. Otherwise, the parity error information is passed to the user and the . This triggers the protocol module transmitter's transition back to the ”ready” state. Lastly, if the protocol module receives instructions to write to the trigger FIFO, then it does so before asserting the ACK bit in the received packet and passing it to the transmitter.

bit[17]:RnW	bit[16]:ACK	description
0	0	Write event data and received address to the trigger FIFO. Assert ACK bit and pass the acknowledge packet to the transmitter.
0	1	Finished writing to the TRM. Check parity and inform transmitter that writing is complete.
1	0	ERROR: it is not possible within the logic of the system to receive the flag bits "01" as it is not possible for unit B to read from unit A.
1	1	Finished reading from the TRM. The received packet contains configuration parameters and address information. Check parity and inform the transmitter that reading is done.

Table 5: A summary of the two possible protocol module A receiver functions depending on the values of RnW and ACK

5.4 The ENC/DEC Modules

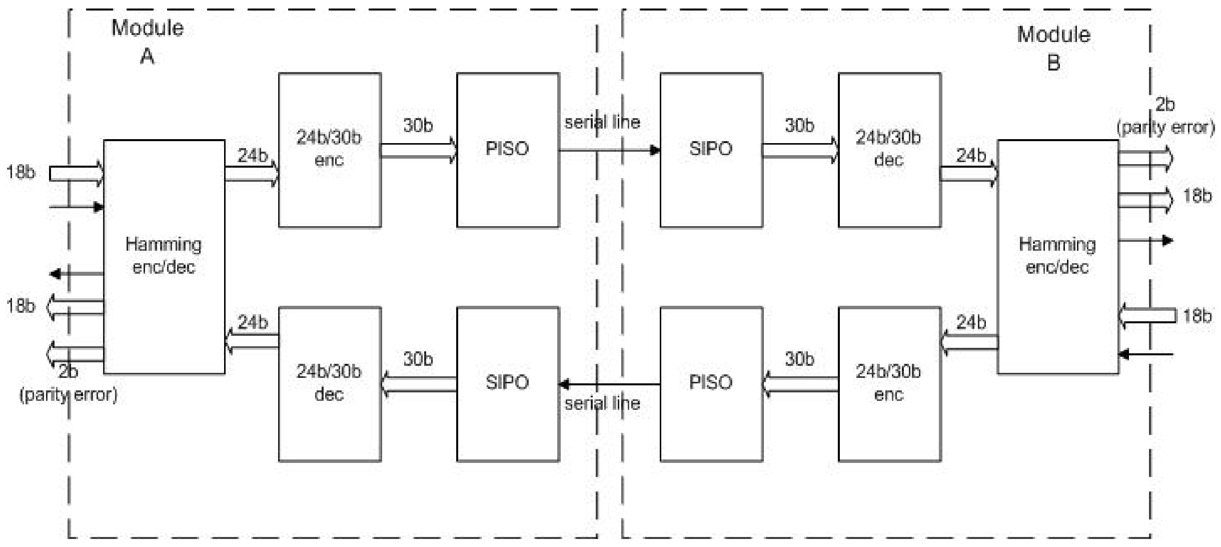


Figure 20: The two identical communication modules and their components

The ENC/DEC modules increase the robustness of transmission by allowing for the correction of bit errors and the prevention errors which can associated with serial transmission. These two purposes are achieved through the use of the a Hamming parity encoder/decoder module and 8b/10b encoding/decoding modules (see Figure 20).

Bit errors are known to occur digital systems. Some of the most common types of bit errors in digital circuits include [1]:

- **Gaussian bit errors**: Errors resulting from random noise in a circuit.

- **Pattern sensitive bit errors**: Certain bit positions of the word being transmitted or stored have more errors than other positions within the word.
- **Systematic bit errors**: Similar to pattern sensitive bit errors, but can be correlated to a physical event, such as intervalled power supply noise.
- **Systematic burst errors**: Similar to systematic bit errors, but these can occur over multiple bits in a row.

These types of errors cannot be completely prevented, but the bit-error rate (BER) can be greatly reduced by including error-correcting code and 8b/10b encoding/decoding.

5.4.1 Parity Encoding/decoding

When data is transferred along a transmission line, there is a certain probability that errors may occur in one or more bits, changing their value. In order to combat this problem and decrease the BER in transmission, an error-correcting code called Hamming Code was utilized and implemented in the design using firmware designed by Xilinx, see [15], adapted to encode/decode 18 data bits. This code allows for the correction of single bit errors and the detection of double bit errors through the addition of parity bits to the original data packet, as illustrated in [8]. This sort of code is called SECDED, or "single-error correcting double-error detecting". The concept of parity, in this context, refers to the even or odd number of '1's in a word composed of bits. A parity value '0' of a set of bits implies that the set contains an even number of '1's, while there are an odd number of '1's in a set of bits with parity '1'. The correct amount of parity bits for single error correction is determined by the use of the equations below [18],

$$\begin{aligned} \text{totalbits} &= 2^c - 1 \qquad (1)\\ databits &= 2^c - c - 1 \qquad (2)\\ paritybits &= c \qquad (3) \end{aligned}$$

where c is the amount of parity bits required for single error detection and correction. If an extra bit is added, then the code can also detect double errors.

A whole number of parity bits c gives the possibilities of 7, 15, 31, etc. total bits for $c = 3$, 4, or 5. In the case at hand, there are 18 data bits, which does not conform to a whole number c. The choice of the number of parity bits for single error detection is then chosen to be the lowest amount that gives the possibility of more than 18 data bits. This is fulfilled by choosing $c = 5$. 5 parity bits for single error detection allows for 31 data bits, while a choice of 4 parity bits would only allow for 15 data bits, which is less than 18 and thus cannot be used. Lastly, in order to allow for the detection of double errors, one extra parity bit is added to the 5 already selected, giving 6 total parity bits and a new word size of 24 bits after parity encoding.

A central concept for the understanding of the mechanics of Hamming encoding is that of the "Hamming distance" between two words. Hamming distance is defined to be the number of bits that differ between two words. For example, "1101" and "1011" have a Hamming distance of two because two bits differ. Notice that position is relevant. It is the bits in their specific positions which are compared pertaining to Hamming distance. A single bit error corresponds to a hamming distance of one between the transmitted word and the received word. The addition of parity bits to the word serves to increase the minimum Hamming distance between valid words. Such a code as is implemented in this communications system results in a minimum Hamming distance of four between valid words, which allows for the correction of a single bit error and the detection, but not correction, of double bit errors.

The values of these parity bits are evaluated by first arranging the data bits D0 to D17 in ascending order. Then, the parity bits P0 to P4 are added in position numbers which are powers of two, see below. Notice that the parity bits are placed in positions 1, 2, 4, 8, and 16 from the left, if one begins counting positions with one instead of zero.

$$P_0, P_1, D_0, P_2, D_1, D_2, D_3, P_3, D_4, D_5, D_6, D_7, D_8, D_9, D_{10}, P_4, D_{11}, D_{12}, D_{13}, D_{14}, D_{15}, D_{16}, D_{17} \quad (4)$$

Notice that the parity bits are placed in positions 1, 2, 4, 8, and 16 from the left, if one begins counting positions with one instead of zero. This arrangement allows one to organize the data bits for parity calculation. Each parity bit is evaluated by way of specific data bits. Parity bit P0 will use every other bit starting with D0 and excluding other parity bits. P1 will use every two bits starting with D0, excluding parity bits. P2 will use every four bits starting with D1, P3 will use every eight bits starting with D4, and P4 will use "every 16" bits starting with D11. These data bits can be arranged using a parity-check matrix, as shown in equation 5.

$$\begin{bmatrix}
D_{17} & & D_{15} & & D_{13} & & D_{11} & D_{10} & & D_8 & & D_6 & & D_4 & D_3 & & D_1 & D_0 \\
D_{17} & & D_{15} & & D_{13} & D_{12} & & D_{10} & D_9 & & & D_6 & D_5 & & D_3 & D_2 & & D_0 \\
D_{17} & D_{16} & D_{15} & D_{14} & & & & D_{10} & D_9 & D_8 & D_7 & & & & D_3 & D_2 & D_1 & \\
 & & & & & & & & D_{10} & D_9 & D_8 & D_7 & D_6 & D_5 & D_4 & & & \\
D_{17} & D_{16} & D_{15} & D_{14} & D_{13} & D_{12} & D_{11} & & & & & & & & & & & \\
D_{17} & D_{16} & D_{15} & D_{14} & D_{13} & D_{12} & D_{11} & D_{10} & D_9 & D_8 & D_7 & D_6 & D_5 & D_4 & D_3 & D_2 & D_1 & D_0
\end{bmatrix} \quad (5)$$

Each parity bit is evaluated by executing the XOR boolean logic operation between bits. The first row is used to evaluate P0, the second row is used to evaluate P1, etc. These six parity bits are added to the 18 bit transmission packet in the first six bit positions 0 to 5, while all of the other 18 bits occupy locations 6 to 23 such that the new transmission word contains 24 bits.

The decoding mechanism involves the calculation of a ”syndrome”, which is used to generate a mask which is XOR’ed with the 18 bit received packet upon the detection of a single-bit error. The syndrome is calculated first by evaluating the parity of the data bits in the same fashion as described earlier. If there is a single-bit error, then these bits will contain some different values than the original parity bits. The syndrome is then used to select the correct mask, as shown in Figure 21. See Appendix B for the specific choice of correction mask.

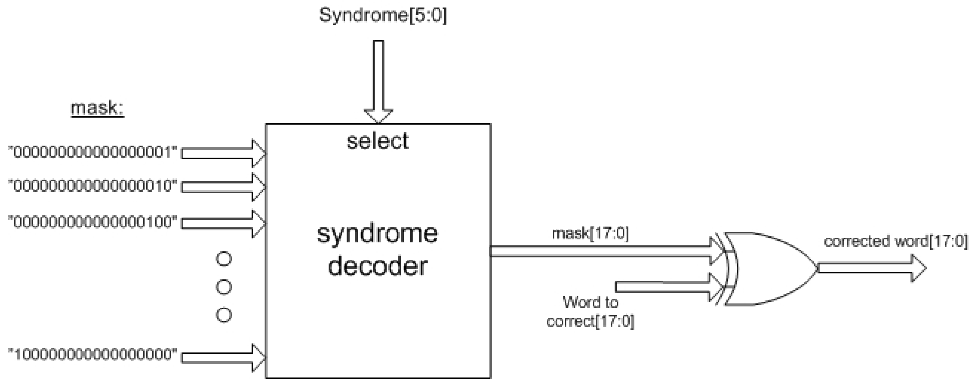


Figure 21: mask selection and single-bit error correction

If a bit error is detected, the Hamming ENC/DEC must inform the user of this. This is done through the protocol module, as mentioned earlier. The protocol module receives a two-bit word which indicates parity error, as shown in Table 6. The protocol module receives this word and communicates it to its master unit.

parity error word[1]	parity error word[0]	description
0	0	no error and no data corrected
0	1	single-bit error detected and corrected
1	0	double-bit error detected
1	1	single parity checkbit error

Table 6: A summary of the interpretation of the parity error word

5.5 8b/10b Encoding/decoding

Today, the 8b/10b data transmission method has become the standard for high-speed serial links [11]. It translates byte-wide data into a 10-bit data packet with the purpose of eliminating the unwanted DC component in data transmission after serialization [20]. The presence of a DC component gives rise to unwanted energy loss and can lead to transmission errors. In order for this component to be eliminated, the binary logic levels ’1’ and ’0’ must be positive and negative voltages of equal magnitude, which is referred to as a non-return-to-zero (NRZ) line code. There must be an equal amount of transitions between logic high('1') and logic low('0') within the scope of transmission. The equal amount of '1's and '0's prevents the aforementioned unwanted DC effects, preventing these potential transmission problems and improving the robustness of transmission.

Within the system, this component consists of three 8b/10 encoding/decoding blocks which were designed by Xilinx, see [2]. These were modified for interconnection within the system and assigned to the three bytes in the 24 bit word which it receives from the parity encoder. These three bytes are encoded separately into 10 bits each and are concatenated to form one 30 bit encoded output word which is passed along to the serializer (see Figure 24). There are multiple mechanisms involved with 8b/10b encoding, which are discussed below.

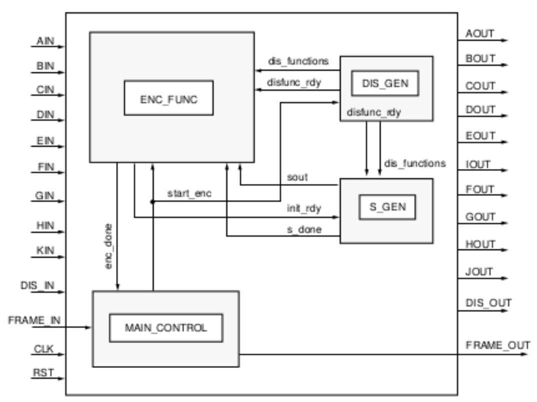


Figure 22: A single block 8b/10b encoder with internal modules

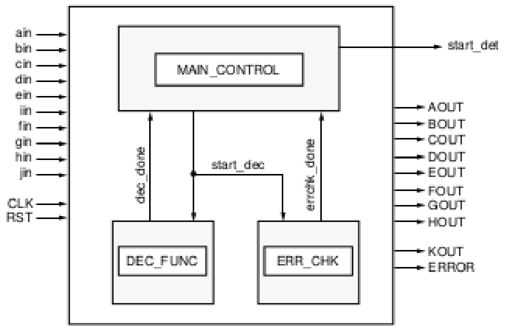


Figure 23: A single block 8b/10b decoder with internal modules

5.5.1 Disparity

In order to grasp the 8b/10b encoding scheme, the concept of "disparity" must be understood, as this is the fundamental concept associated with 8b/10b encoding. Disparity, whose root word is "parity", is a measurement of the difference between the amount of ones and zeros which have been transmitted, the purpose of which is to attain balance between these values. For example, a 10 the bit word "0100110001" has disparity "-2" because it contains two more zeros than ones. A word containing an equal number of ones and zeros is described to have "0" disparity. As there is an event amount of bits in an encoded 10 bit word, it is only possible for each word to have disparity "±2" or "0". It is then conceivable that it is possible for a complete 30 bit output word to have disparity "±6", which would give rise to a DC component and thus defeat the purpose of 8b/10b encoding. This is however solved by the passing of the first disparity value to the next encoding block and the next disparity value to the final encoding block (see Figure 24). Each block takes the disparity output from the previous block into account with the purpose of maintaining DC balance. For example, if the first 8b/10b ENC module gives an output word with disparity "+2", then the next word, if originally encoded to have

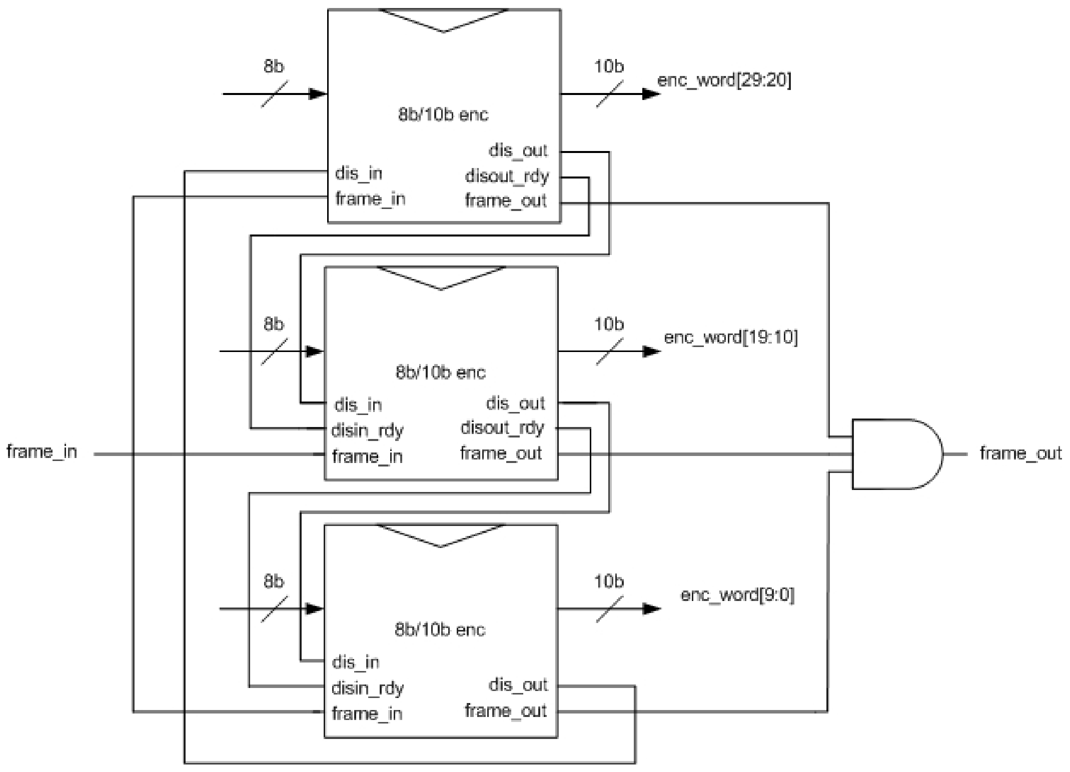


Figure 24: Block diagram of the VHDL implementation and interconnections between the three 8b/10b encoding modules. Clock inputs are at the triangles on the top of each module

a value of "0110111010", will invert the values of the six least significant bits. The final encoded word in that 8b/10b ENC module would then be "0110000101", which has disparity "-2". This results in the net disparity "0" for the first two words. The disparity of each word is calculated in the submodule "dis_gen" (see Figure 22).

5.5.2 Encoding

The 8b/10b encoding scheme follows an algorithm which changes any 8 bit word into a 10 bit word with a maximum disparity of ±2 and a possible disparity zero. An 8 bit "ABCDEFGH" word is first partitioned into two words "ABCDE" and "FGH" before they are separately encoded into 6 bit and 4 bit words "abcdei" and "fghj", respectively. This is referred to as 5b/6b encoding and 3b/4b encoding. Each word can have a of maximum two possible values zero or opposite disparities, as mentioned earlier. For example, if a 3 bit word is encoded into "0010" then it has the possibility of being changed to its opposite value "1101" for the purpose of disparity balance. This is made possible by the S signal which is described below. A 3 bit word which would be encoded to a 4 bit word with an equal amount of '1's and '0's need not have an opposite value because there would be no change in disparity upon transition. These words are finally concatenated together to build the encoded 10 bit word. The encoding logic allows for any byte value to be encoded into a 10 bit word of disparity "0" or "±2". See Appendix A for a description of the 8b/10b encoding logic.

5.5.3 S Generation

S is a control signal utilized in the 8b/10b encoding logic (see Appendix A for detailed description of its generation through encoding logic). Its purpose is to assist in the determination of the 3b/4b encoding by using the running disparity information from each 8b/10b encoding block which would have been determined earlier in the encoding process. The 5b/6b encoding does not utilize the S signal. Its generation occurs within a submodule of the 8b/10b encoding block called "s_gen" (see Figure 22). The flexible 3b/4b encoded word is chosen to contain the number of '1's and '0's in order to achieve DC balance with the use of the S signal.

5.5.4 Decoding

The encoded 30-bit word, after being transmitted serially to the opposite unit, is then decoded through three 8b/10b decoder blocks, the structure of each block is shown in figure 23. The 10-bit word to be decoded is, in a similar fashion as the encoder, broken up into two words of 6-bit and 4-bit length before being decoded back into the original 5-bit and 3-bit words, respectively. These are then concatenated into bytes and the three decoded bytes are concatenated to form the orignal 24 bit word. The decoding logic is the complement to the encoding logic described in Appendix A. The decoding logic does not depend on disparity, but only an error detector, which is embedded in each 8b/10b decoding module. Since the 8b/10b encoding mechanism is so specific, there are multiple errors that can be detected by the decoder by simply checking if there are illegal combinations of ones and zeros, depending on position of these bits within the word. The method for finding illegal combinations is also described in Appendix A.

5.6 Serialization, Transmission, and Deserialization

The innermost units in each ENC/DEC module are the serializer and deserializer. These reside on the outputs and inputs of both modules, such that data may be serialized, transmitted, deserialized and handled subsequently. First, parallel data is received by the serializer, which fills a shift register when informed by the serializer's FSM that parallel data is available.

Data may be sent sporadically and the units must be able to receive data at any time. In addition, it is assumed that the serial line will be most often inactive, transmitting nothing. Therefore, no clock will be transmitted from one unit to another across any serial link. This gives rise to the necessity of a mechanism which allows for each unit to receive serialized packets at any time with minimal error without receiving any clock. Each unit's own on-board clock must be used. There are multiple schemes which fulfill such a purpose. Some of these include:

- Embedding the clock into the serial data by the utilization of
  - Phase Encoding (PE)
  - Pulse-width modulation (PWM)
- Sampling the data at a higher frequency and letting the majority of sampled values determine the bit values of the serial data.

The application of PE and PWM would require an encoding scheme other than the 8b/10b encoding which was used within the scope of this design. PE, which uses NRZ encoding where the bit values change on every clock cycle, is realizable through the use of, for example, Manchester Encoding. The change on every clock cycle embeds the clock into the code itself. PWM embeds the clock into the code by giving each value, including zero, a pulse width which increases with the value to be sampled. This is most often utilized when many values, not just one and zero, are to be transmitted and interpreted. The method of sampling the data at a higher frequency to capture incoming data was utilized in this system.

The clock frequency of the transmission each way is configured to be at 100 MHz. Each unit A and B has its own transmission clock at that frequency, but there these clocks are separated from one another and are not configurable to be in phase within this system. This presents the possibility that the receiving clock may not sample the incoming data in the correct place in relation to the data's waveform, causing transmission errors. As each unit has its own clock, there is no need for transmission of any master clock over a physical link. Therefore, there must be a sampling mechanism such that the data being transmitted across the physical link can be received with minimal chance of failure.

To achieve this, start bits "10" are placed at two new bit locations which become the least significant bits of the word to be transmitted. The stop bits "01" are placed at the back end of each word in similar fashion. After serialization and addition of start and stop bits, the data is transmitted serially at 100 MHz. On the receiving end, a sampling clock at frequency 300 MHz waits for the start bits "10" incoming data. When these arrive, they are sampled at this sampling frequency, giving three values for each '0' and '1'. A maximum of three of these values can be correct for each of these bits and minimum two can be correct. Such is also the case with the rest of the bits as the reception register is filled (see Figure 25). After each bit is sampled three times, the three values are passed to a majority gate, where the correct value of each bit is the one which receives at least two "votes". These values are passed to their own registers and concatenated into one parallel word. Deserialization is then complete and the data moves forward through the unit synchronized with the unit's own generated clock.

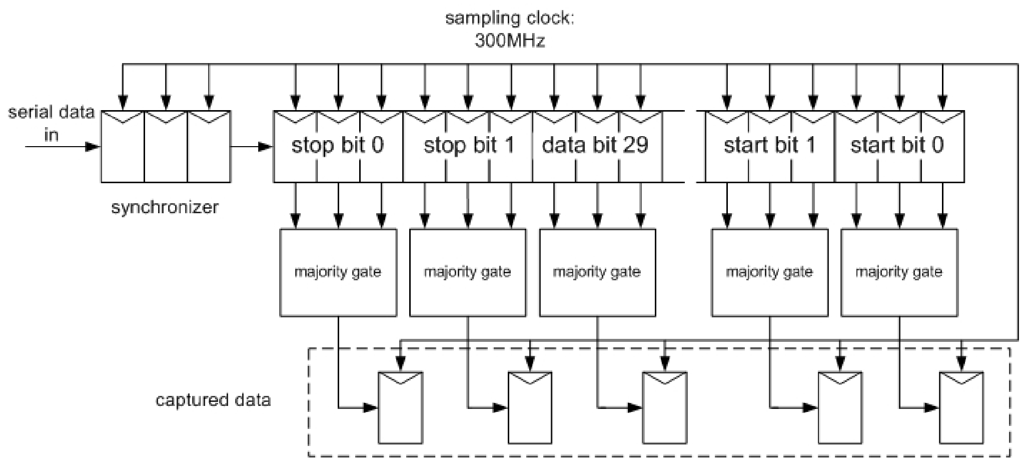


Figure 25: RTL diagram of the sampling mechanism [12]

6 Test and Verification Strategy

6.1 Simulation

The testing and verification of a system such as this must occur in two steps. First, simulation in software, then synthesis to hardware implementation. The purpose of simulation is to test the behavioral functionality of the design. In this case, the ModelSim by Mentor Graphics was utilized for simulation. The system, which was designed in VHDL, describes the behavior of an idealized device. Both behavioral and structural descriptions are included in the code. This design consists of many units with sub-modules which in turn include other submodules and so on.

Basic simulation requires the design of a VHDL "testbench". The VHDL testbench functions as "the outside world", and treats the system as a device with input and output pins. The user is able to structurally construct the system to be simulated using VHDL code and generate a clock and input vectors. When simulated, it is possible to view output as well as input waveforms. The basic simulation strategy employed in the design and development of the discussed trigger readout system was to first design the system in VHDL, design a testbench for the system and, lastly, simulate the behavior through the testbench by applying input test vectors.

6.2 Synthesis and Verification Strategy

In order to verify the behavior of the VHDL design on hardware, synthesis must be performed. Synthesis is performed on the Xilinx ISE environment. In the first stage, the entire system (which includes both units A and B), was synthesized on one FPGA, where the physical links were merely modeled by a connection between the two units. This verified the basic functionality of the entire system and assist in the debugging of simple errors in the translation from simulation to synthesis. The test scheme included the configuration of four pins through the TRM from "unit A". Then, after configuration, input vectors on these four pins were asserted. The TDU transmitted trigger event packets back through unit B to unit A.

The next stage has not yet been implemented. The plan is outlined as follows. The entire system will be programmed, once again, on the same FPGA, the only difference being that the physical links between unit A and unit B will be a wire connecting an output pin from unit A to an input pin on unit B and vice-versa. This will test the aforementioned sampling mechanism on each deserializer, but only to a certain extent, as the FPGA will generate one clock to be shared by both units. Thus, the two units will have the exact same frequency and phase. The errors associated with serial transmission will be able to be measured and subsequent adjustments in the system can be made. The test scheme for the configuration of pins will remain the same as in the first test phase.

Lastly, unit A and unit B will be synthesized on their own individual FPGAs with serial links connecting them. Unit A will be synthesized on the Xilinx virtex 6 FPGA in the FoCal readout electronics box and unit B will be synthesized on the Actel ProAsic FPGA attached to the trigger card. After synthesis on these FPGAs, one readout pin on the trigger card will be chosen to verify functionality. This pin will be configured from unit A done as earlier, and a real stimulus will be applied to the chosen pin on side B. The trigger event information will be read from unit A, verifying functionality. In addition, this testing scheme will completely verify the sampling mechanism for the reception of serial data across the physical links, as each FPGA will need to generate their own clock.

Eventually, the trigger system ought to be verified for functionality in connection with the actual FoCal detector tower with readout electronics and DAQ with stimuli at regular intervals and then stimuli in random intervals to test the robustness of the system. This would be the final step in the verification process and would determine complete functionality.

7 Conclusion

The purpose of this thesis was to develop a trigger readout system for the Forward Calorimeter. This system is able to collect event data from the trigger readout pins and organize it for transmission across a physical link to the user residing on the other side. The user is also able to configure the output pins to be enabled and edge-sensitive by transmitting configuration parameters to the pins across a physical link. The master-master relationship between the two individual units separated by the transmission lines is resolved, and they are able to transmit and receive relevant information between each other simultaneously. The design remains to be completely verified through synthesis and testing, as explained in Section 6. The maximum event data word rate was calculated to be 2.9 MHz with a 100MHz transmission clock.

This system, although it meets the general purpose of a trigger readout system, can be built upon such that it can be more sophisticated and user-friendly. One possibility it to develop a graphical user interface (GUI) for the user on side A. The user would be able to utilize this GUI to configure trigger parameters and enable pins visually through keyboard commands or mouse-clicking. A diagram would be able to display visually which pins are activated or deactivated, making it more simple for the user to keep track of the status of the system on the other end. Instead of having to manually construct the data and address information for each pin, the user would simply choose a pin/pins and the three possible configuration options for it/them. The 18 bit word to be transmitted for configuration would then automatically be sent to the protocol module on unit A for transmission.

Appendix A: 8b/10b encoding and decoding

8b/10b encoding is separated into two processes: the 5b/6b encoding process and the 3b/4b encoding process. These are discussed separately.

5b/6b Encoding Logic

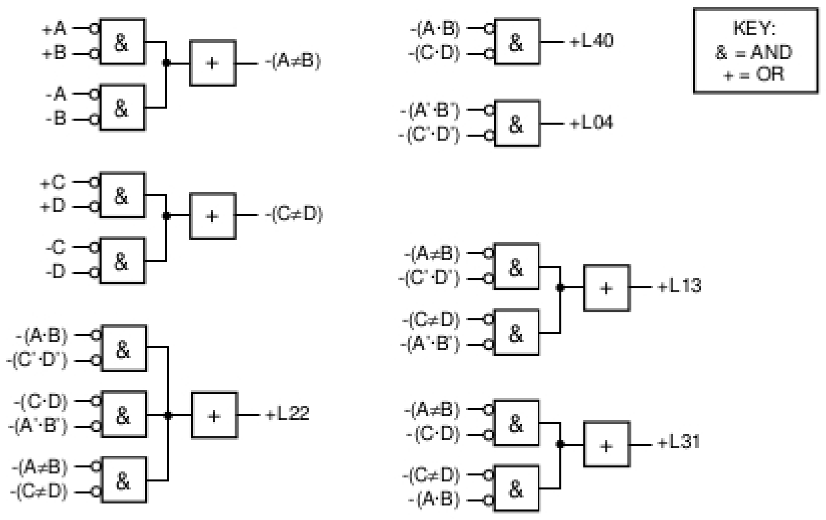


Figure 26: 5b/6b encoding function signals

Before the 8b/10b encoding can take place, multiple function signals must be generated in order for the logic to take the number of '1's and '0's in the input word into account. These are generated in the logic pictured in figure 26. Notice that only the first four bits in the 5b word to be encoded are utilized in this logic. The names of the encoding functions include the letter L and two numbers. The number to the left indicates the amount of '1's and the number to the right indicates the amount of '0's in the word. +L31 has, for example, three '1's and one '0'. Each of the five "L" encoding functions receives a value before they are used in final 5b/6b encoding logic shown in figure 27.

3b/4b Encoding Logic

Similar to the 5b/6b encoding logic, the 3b/4b encoding logic requires the generation of multiple encoding functions before final encoding can take place. The difference is that these functions do not only depend on the input word's structure but also on the disparity information. This information is found as illustrated in figure 28 while other functions dependent on the word structure are found in figure 29. After this logic is executed, the *S* signal can then be generated as shown in figure 30.

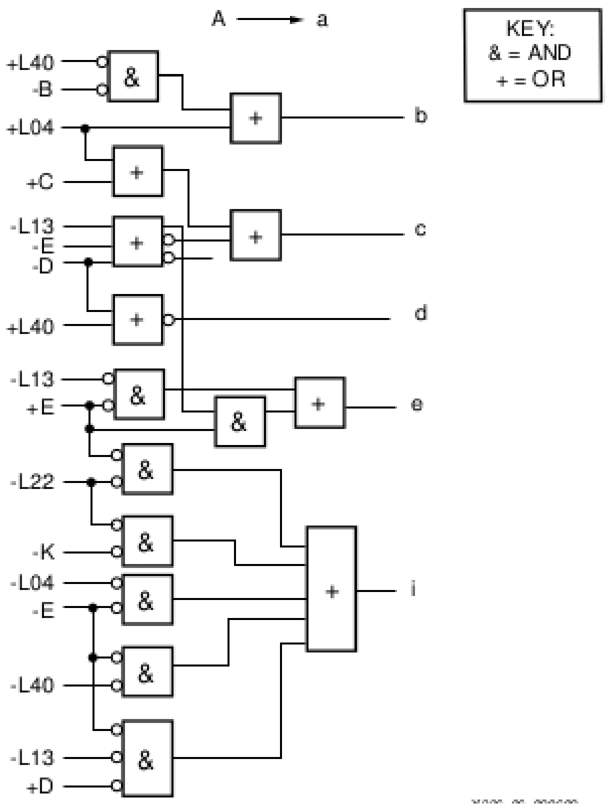


Figure 27: final 5b/6b encoding

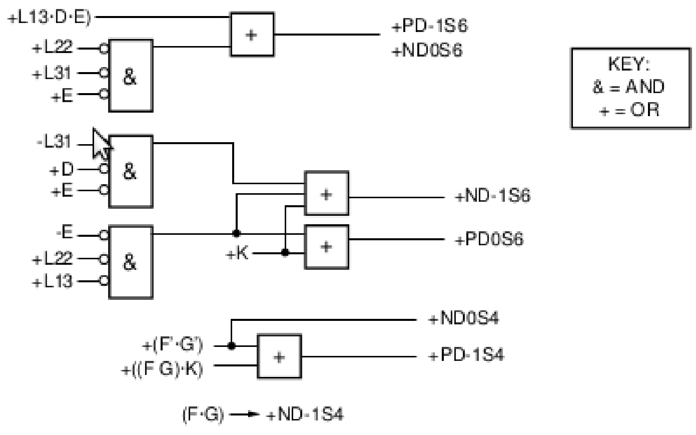


Figure 28: disparity classification signals

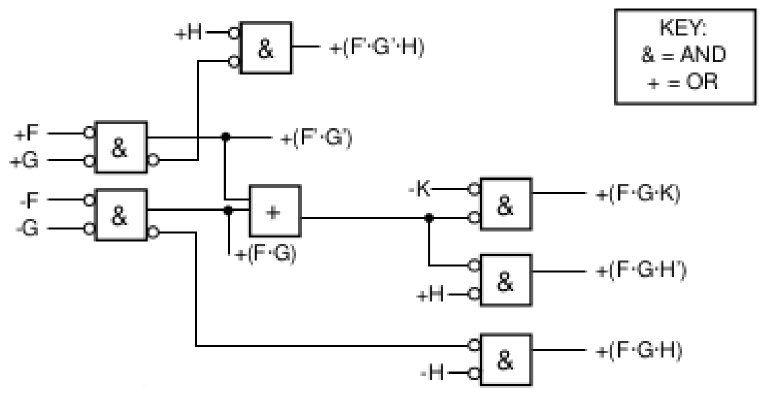


Figure 29: 3b/4b function signal generation

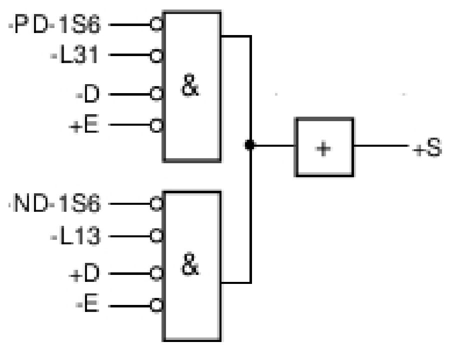


Figure 30: S signal generation

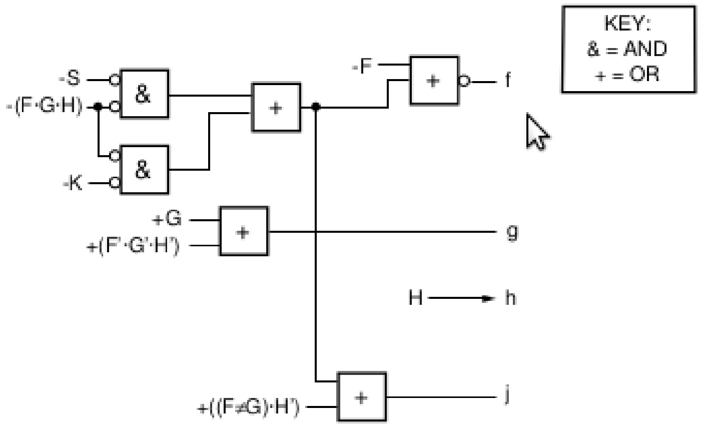


Figure 31: 3b/4b encoding scheme

Error Detection

The 8b/10b decoder detects transmission errors in the error check module by comparing the word received to the 8b/10b encoding rules. A tool which is utilized for the decoding is the called the preliminary bit classification decoding functions [2]. These are similar to the encoding functions, which were denoted with the letter L. In the case of decoding, the letter P denotes a decoding function. Following the letter P are two numbers, the first of which represents the number of '1's in the four least significant bits of the word to be decoded. The second number represents the number of '0's in the same four bits. For example, the assertion of P13 implies that the four least significant bits of the word to be decoded contains one '1' and three '0's. A list of rules which, when broken, are interpreted to be errors is shown below [2].

- All 6b and 4b subblocks of a packet must have either positive, negative, or zero disparity (difference between "1s" and "0s").
- The disparity out of nonzero disparity blocks must alternate in polarity (positive and negative).
- All data bytes must follow the disparity rules.
- The following conditions also apply to coding rules and are attributed to errors:

$$a = b = c = d \tag{6}$$
$$P13 \cdot \overline{e} \cdot \overline{i} \tag{7}$$
$$P31 \cdot \overline{e} \cdot i \tag{8}$$
$$f = g = h = j \tag{9}$$
$$e = i = f = g = h \tag{10}$$
$$i \neq e = g = h = j \tag{11}$$
$$(e = i \neq g = h = j) \cdot \overline{(c = d = e)} \tag{12}$$
$$\overline{P31} \cdot e \cdot \overline{i} \cdot \overline{g} \cdot \overline{h} \cdot \overline{j} \tag{13}$$
$$\overline{P13} \cdot \overline{e} \cdot i \cdot g \cdot h \cdot j \tag{14}$$

Appendix B: Choice of correction mask for single-bit error correction in Hamming decoding for 18 data bits

The Hamming decoding scheme involves correction of single-bit errors through the XOR operation between the error word and the correction mask. The mask is chosen according to the calculated syndrome. The value of the syndrome in base-10, discluding the ’1’ in the most significant position, denotes the location of the bit error to be corrected. Since the syndrome ”000000” implies no error, the bit location is equal to the aforementioned base-10 value of the syndrome minus one. Some values, however, are not included in this scheme. This is because the parity checkbits reside at these positions. As a result, the base-10 values 1,2,4,8 and 16 are not included in the syndrome/mask decoding scheme. See Table 7 for details.

syndrome	databit position in 17b word	correction mask
100011	0	000000000000000001
100101	1	000000000000000010
100110	2	000000000000000100
100111	3	000000000000001000
101001	4	000000000000010000
101010	5	000000000000100000
101011	6	000000000001000000
101100	7	000000000010000000
101101	8	000000000100000000
101110	9	000000001000000000
101111	10	000000010000000000
110001	11	000000100000000000
110010	12	000001000000000000
110011	13	000010000000000000
110100	14	000100000000000000
110101	15	001000000000000000
110110	16	010000000000000000
110111	17	100000000000000000
others		000000000000000000

Table 7: mask choice based on syndrome value

Acronyms and abbreviations

ACK Acknowledge

ALICE A Large Ion Collider Experiment

ASIC Application Specific Integrated Circuit

ATLAS A Toroidal LHC Apparatus

BER Bit-Error Rate

CERN European Organization for Nuclear Reasearch

CI Configuration Interface

CMS Compact Muon Solenoid

CPLD Complex Programmable Logic Device

CTP Central Trigger Processor

ECL Emitter-coupled Logic

ENC/DEC Encoder/Decoder

FEE Front End Electronics

FIFO First In First Out

FoCal Forward Calorimeter

FPGA Field Programmable Gate Array

FSM Finite State Machine

GUI Graphical User Interface

HDL Hardware Description Language

ISO International Organization for Standardization

LHC Large Hadron Collider

LHCb LHC-beauty

LVDS Low Voltage Differential Signaling

NIM Nuclear Instrumentation Standard

NRZ Non Return to Zero

MIMOSA Minimum Ionizing Particle Metal Oxide Semiconductor Active pixel sensor

OSI	Open Systems Interconnection
RnW	Read-not-Write
RTL	Register-Transfer Level
SECDED	Single-error correcting, Double-error detecting
TCI	Trigger Crate Interface
TDU	Trigger Data Unit
TP	Trigger Protocol
PE	Phase Encoding
PECL	Positive Emitter-Coupled Logic
PWM	Pulse-Width Modulation
RCSA	Register Control Side A
TRM	Trigger Register Map
VHDL	Very high speed integrated circuit Hardware Description Language
XOR	Exclusive OR

List of Figures

List of Tables

References

[1] *An introduction to error location analysis*, Agilent Technologies: Test and Measurement, literature.agilent.com/litweb/pdf/5980-0648E.pdf, Application note 1550-2.

[2] *Design of a 16b/20b encoder/decoder using a coolrunner-ii CPLD*, Tech. Report XAPP391, Xilinx, 2006.

[3] *Technical design report for a nosecone calorimeter for the PHENIX experiment*, Tech. report, Brookhaven National Laboratory, 2007.

[4] P. Buis, *The ISO layering model*, Computer Science Department, Ball State University, September 1996.

[5] ALICE Collaboration, *The ALICE experiment at the CERN LHC*, Journal of Instrumentation **2008 JINST 3 S08002** (2008).

[6] A. Himmi et al, *PHASE-1 user manual*, Tech. report, Institut de Recherches Subatomiques IN2P3-CNRS / ULP, Strasbourg, France, 2008.

[7] D. Fehlker et al, *Highly segmented electromagnetic calorimeter prototype*, University of Bergen, October 2011.

[8] R. Hamming, *Error detecting and error correcting codes*, Bell Systems Technical Journal **29** (1950), 147–160.

[9] E. Iancu, *Gluon saturation at small x*, (2001).

[10] L. Babukhadia, S. Dasu, and G. Punzi, *Triggering in particle physics experiments*, IEEE Nuclear Science Symposium, November 2002.

[11] C. Loberg, *Troubleshoot and verify 8b/10b encoded signals with a real-time oscilloscope*, Tektronix, July 2012.

[12] M. Munkejord, *Development of the ALICE busy box*, Master's thesis, Univerity of Bergen, 2007.

[13] P. Simoneau, *The OSI model: Understanding the seven layers of computer networks*, Expert Reference Series of White Papers, Global Knowledge, 2006.

[14] W. Stallings, *Computer organization and architecture*, eighth ed., Pearson, 2010.

[15] S. Tam, *Single error detection and double error correction*, Tech. Report XAPP645, Xilinx, 2006.

[16] Tektronix, *Fundabemtals of triggering*, 2011.

[17] C. Torgersen and T. Andersen, *Trigger crate interface*, Tech. report, Bergen University College, 2012.

[18] J. Wakerly, *Digital design principles and practices*, vol. 3, Prentice Hall, 2001.

[19] N. West and D. Harris, *Integrated circuit design*, ch. 9, pp. 373–374, Addison-Wesley, 2011.

[20] A.X. Widmer and P.A. Franasazek, *A DC-Balanced , partioned-block, 8b/10b transmission code*, IBM J. Res. Develop. **27** (1983), no. 5.

[21] M. Winter, *Introduction to CMOS pixel sensors*, CERN, 2011.

[22] Y. Hori, H. Hamagaki, and T. Gunji, *Simulation study for forward calorimater in LHC-ALICE experiment*, Journal of Physics: Conference Series **293** (2011).