Simulating TPC Readout Electronics

Consectetur adipisicing elit, sed do tempor incididunt ut labore et dolore magna aliqua

Håvard Rustad Olsen

Master's thesis in Software Engineering at

Department of Computing, Mathematics and Physics,
Bergen University College

Department of Informatics,
University of Bergen

June 2015

Acknowledgements

Hvard Helstrup, Johan Alme, Dieter, Arild, Christian, (Damian).

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

ALICE A Large Ion Collider Experiment. 11–17, 20

ALTRO ALTRO ASIC. 16, 17, 26, 27, 29

ASIC Application Specific Integrated Circuits. 16, 17, 26, 27

BT Binary Tree. 31

C++ A object-oriented programming language.. 20, 21

CERN European Organization for Nuclear Research. 11–13

CRU Common Readout Unit. 18, 26, 29, 32

FEC Front-End Card. 17, 26, 27, 31

FIFO First-In-First-Out. 22, 26, 27

FPGA Field-Programmable Gate Array. 29

GBTx Giga Bit Transceiver. 18, 26, 27, 32

GEM Gas Electron Multiplier. 17

LHC Large Hadron Collider. 11, 13, 14, 17

MWPC Multi Wire Proportional Chamber. 17

Priority Queue Datastructure which sorts elements based on a priority(numerical value). 31

QCD Quantum Chromodynamics. 15

RCU Readout Control Unit. 16, 18

SAMPA SAMPA ASIC. 17, 26–32

SystemC A simulation library building on C++. 20–22, 25

TeV Terra Electron Volt. 14

TPC Time Projection Chamber. 11, 14, 16, 17, 26

Verilog A Hardware description language. 25

VHDL A Hardware description language. 25

Zero suppression Suppression schema/algorithm. 29, 30, 32

Chapter 1

Introduction

This chapter will cover the motivation, as well as the scope and goal of this report.

1.1 Motivation

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the world's largest particle accelerator, hosting multiple ongoing experiments. After a run period of more than 3 years, the LHC at the CERN will be shut down from 2018 until 2021.[1] The purpose of this shutdown is to do maintenance on various equipment in the LHC, as well as significant upgrades to the different detectors, one of which is the detector for the A Large Ion Collider Experiment (ALICE). ALICE consists of multiple sub-detectors, which combined collect an enormous amount of data. This amount is expected to increase after the shutdown period as the interaction rate of the LHC will increase. Due to the increase in data output, the ALICE collaboration is seeking to upgrade and enhance the detector capabilities.[2] This includes a partial redesign of the readout electronics, upgrades to multiple sub-detectors and additional hardware upgrades.

The Time Projection Chamber (TPC) is the ALICE detector's main sub-detector for tracking and identifying particles. A starting/temporary? design for the new TPC readout electronics is made, and the different components are currently being developed. As this is still being worked on, many questions about the different components are yet to be answered. Are the current specifications sufficient to handle the expected increase in output from the detector? Do they have the necessary bandwidth to be able to send the data with minimal sample loss. Are the buffer memory enough to handle the traffic. Is it possible to optimize the current solution in any way?

The previous paragraph motivates us to find a reliable way of determining a sufficient design for the readout electronics, while being both time and cost efficient. One strategy for solving this problem, which will be further explored in this thesis is creating a simulation of the system. Doing a simulation requires

designing a accurate representation of the readout electronics, and creating a testbench where it is possible to configure and run multiple tests.

1.2 Research Question and thesis goal

Given the motivation and introduction given in section 1.1 the research question for this thesis becomes:

Is it possible to design and implement a simulation which directly represent the readout electronics, and in doing so will it have an optimizing effect?

Further explained, the main tasks of this thesis will be to create a computer model of the readout electronics main components, and run multiple simulations on it. Experimenting with different configurations in order to find bottlenecks, faulty design or areas of improvement. The experiments should be logged, and the results will be presented in an organized fashion.

1.3 Report structure

Chapter 2 will give the reader the background information to be able to understand the different academic and scientific terms used, as well as some information about the context of the report. This includes information about CERN, the ALICE experiment and the physics most relevant to the thesis. It will discuss the current readout electronics as well as the proposed upgrade. Chapter 3 is going further into the problem discussed in this report, initial plans on solving the problem, and information about the tools used. Chapter 4 will talk about the implementation of the simulation, what problems occurred along the way, and the chosen solution. The chapter will go into the design, as well as code snippets from the implementation. With the information given in chapter 4, chapter 5 will discuss the results of the different simulations run, and evaluate the solution. Chapter 6 will conclude the thesis with some closing words, and work that can be done in the future.

Chapter 2

Background

This chapter will give the reader the background needed to set the rest of the thesis in context.

2.1 CERN

CERN is a European research and scientific organization based out of Geneva near the Franco-Swiss border.[3] CERN is a collaboration between 21 countries with a member staff of over 2500, and more than 12000 associates and apprentices. The organization was founded in 1954 and has since then been the birthplace of many major scientific discoveries. These are not limited to discoveries in the field of physics, but includes the creation of the World Wide Web.[4] Currently the biggest project at CERN is the LHC particle accelerator, which serves as the foundation for multiple experiments in the field of particle physics.

2.2 The Large Hadron Collider

Starting up on 10 September 2008, LHC is the latest construct added to CERN's particle accelerator complex.[5] It consist of a 27 kilometre underground ring of superconducting magnets and accelerators which boost the energy of the particles travelling inside the collider. The collider contains two adjacent parallel high-energy particle beams. These beams consist of protons extracted from standard hydrogen atoms by stripping them of electrons. Along the collider ring there are four intersect points where collision occurs. Each point corresponds to the location of a particle detector - ATLAS, ALICE, CMS and LHCb. The particle detectors are each built and operated by a large collaborations, with thousands of scientists from different institutes around the world. The beams travel at close to the speed of light and are guided by a magnetic field, which is created and maintained by superconducting electromagnets. Superconducting meaning that its in a state where it can most efficiently conduct electricity, without resistance or energy loss. Achieving this state requires cooling the magnets to -271.3° C , which is done by the distribution of liquid helium. The layout of the LHC ring as well as its four collision points can be seen in Figure 2.1.

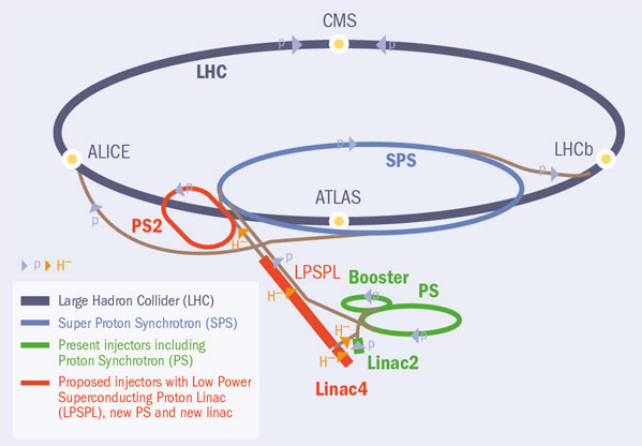


Figure 2.1: The Large Hadron Collider

The beams travelling inside the LHC reach an energy-peak of 7 Terra Electron Volt (TeV), which means that on impact with each other the collision reach an energy of 14 TeV.[6] During a normal run of the collider there will be about 600 million particle collisions per second during a period of 10 hours. This leads to a huge amount of data for each of the detectors to read out. ALICE is the detector which produce the most data per collision, with a design value of about 1.25 GB/s. The high amount of data per collision is produced primarily by the TPC sub-detector, which records a high number of points per track, and has a low momentum threshold. Detectors like ATLAS and CMS are designed with a higher momentum threshold, but can cope with significantly higher collision rates than ALICE. ALICE is designed for the study of heavy ion reactions, where particle correlations at low momentum is an important measure.

2.3 ALICE

2.3.1 Introduction

ALICE is a so called heavy-ion detector, which means it studies collisions between heavy nuclei of high energy.[7] When the experiment started heavy-ion collision lead ions was used, but they have now switched to colliding lead with protons, both produce a extreme high amount of temperature and density. They will eventually run collisions with lead to lead ions again as both types create interesting results. Lead to lead collisions create Hot Nuclear Matter, while lead to protons create Cold Nuclear Matter. The explanations for these types of matter is beyond the scope of this thesis and will not be discussed further. The high temperature and density is necessary to produce a phase of matter called quark-gluon plasma.

2.3.2 Quark-gluon plasma

Shortly after the big bang, the universe was filled with a extremely hot cluster of all kinds of different particles moving around at near light speed.[8] Most of

these particles were quarks, fundamental building blocks for matter, and gluons which ties quarks together in order to form heavier particles. Normally quarks and gluons are very strictly tied together, but in the conditions of extreme temperature and density as in the time shortly after the big bang, that they are allowed to move freely in an extended volume called quark-gluon plasma. The existence of quark-gluon plasma and its properties is one of the key issues in Quantum Chromodynamics (QCD). The ALICE collaboration studies this, observing how it behaves.

2.3.3 The detector setup

The detector weight is about 10,000 ton, it is 26 m long, 16 m wide, and 16 m high.[9] It consists of 18 sub-detectors, each with its own set of tasks regarding tracking particles and collecting data. This large number of sub-detectors are needed in order to get the full picture of the complex system which is being studied(i.e different types of particles and the correlations between them). Most of the detector is embedded in a magnetic field, created by a large solenoid magnet, which makes particles formed in collision bend according to their charge, and behave differently relative to their momentum. High momentum equals near straight lines while low momentum makes the particles move in spiral-like tracks. During lead to lead collisions the collision rate peaks at 8 kHz(Where Hz is defined as number of events per second). This number is still a lot lower in practice because the ALICE detector uses a triggered readout, which only triggers on head-on(central) collisions. The maximum readout rate of the current ALICE detector is 500 Hz, which is more than enough to track central collisions. Figure 2.2 shows a cross section of the detector as it is today with the red solenoid magnet, and all sub-detectors labeled.

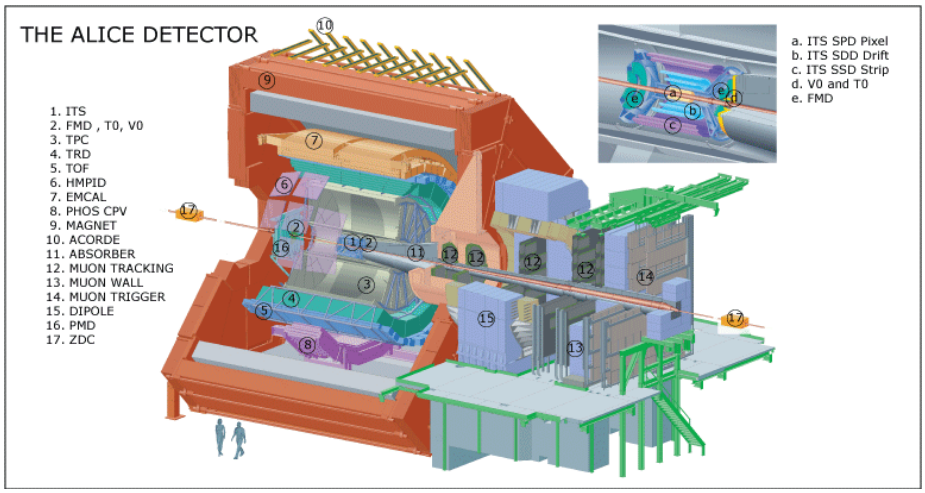


Figure 2.2: The ALICE detector

2.4 The TPC detector

2.4.1 Intro

One of the most important sub-detectors, and the one that is relevant for this thesis is the TPC detector. Located at the center of the ALICE detector it is among the first entry point when gathering data from a particle collision. It is a $88m^3$ cylinder filled with gas. The gas works as a detection medium, which means that charges particles from a collision crossing will ionize the gas atoms, freeing electrons that move towards the end plates of the detector. The readout is done by specially designed readout chambers, which are capable of handling the high amount of data produced in heavy-ion collisions.

2.4.2 Readout electronics

Signals from the readout chambers are passed along to the front-end readout electronics, which today consist of 4356 ALTRO Application Specific Integrated Circuits (ASIC) chips.[10] ASIC is the term used for specially customized chips, rather than chips with a more general-purpose use.[11] The ALTRO chip is made up of 16 asynchronous channels that digitize, process and compress the analogue signals from the readout chambers. It operates on a so called triggered readout mode. In short when ALTRO receives the first trigger, it stores the following data stream into memory, holding on to it until it is ready to pass on the data. The front-end electronics are able to readout data at a speed of up to 300 MB/s.

The ALTRO chip sends the digitized signals further down the readout chain to the Readout Control Unit (RCU), where it is further processed and shipped to and stored in the online systems. The schematics is shown in Figure 2.3.

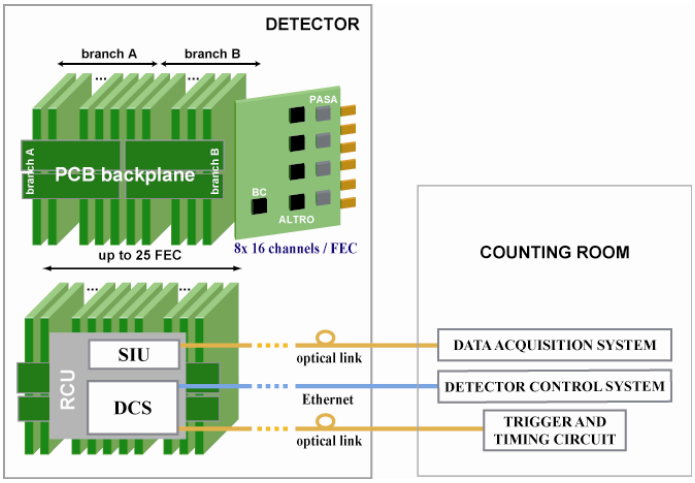


Figure 2.3: Readout schematics for the current TPC detector

2.5 Long Shutdown 2

As mentioned in 1.1 the LHC ring will be shut down for about 3 years, starting 2018. During that time the ALICE detector will undergo an extensive upgrade. The upgrade strategy for ALICE is based on the expected increase in collision rate to 50 kHz, and will now track every collision. Essentially this comes down to a increase by a factor of 100, compared to what is achievable today.

To be able to handle the increase in collision rate the TPC will receive upgrades to both its readout chambers, and front-end readout electronics. The current Multi Wire Proportional Chamber (MWPC) based read-out chambers will be switched out for Gas Electron Multiplier (GEM) detectors, which has a much higher readout rate capability. Signals will be passed from the new read-out chambers to the Front-End Card (FEC) via a readout pad structure similar to the one presently used. There are multiple pad structure depending on its location on the detector, but the difference in structure is not relevant for this thesis. What is relevant however is that more data is expected from low pad numbers, an example of a pad structure is shown in figure Figure 2.5.

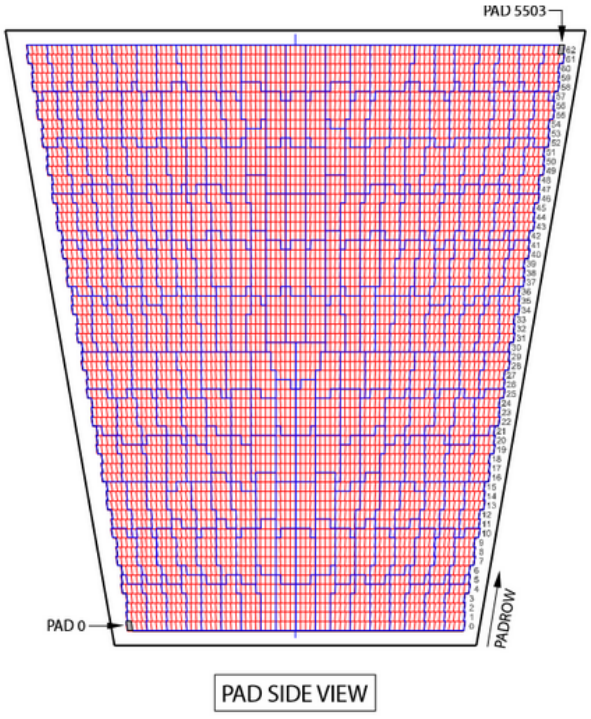


Figure 2.4: Pad structure of an Inner Readout Chamber(IROC) (Credit to Christian Lippmann)

The entry point in the FEC is the new custom-made ASIC, the SAMPA, which is replacing the ALTRO chip.[12] The SAMPA chip is capable of processing signals asynchronously in 32 individual channels, each channel is directly

connected to a single pad. They are further on digitized and concurrently transferred to the Giga Bit Transceiver (GBTx), which enhances the signal strength and transmits them via multiple optical fiber links to the Common Readout Unit (CRU). The CRU can be thought of as the new RCU and serves as an interface to the online systems. The data flow from the detector, and a working schematics can be seen in figure **??**. Chapter 3 will go into more detail about the readout electronics in the context of our simulation.

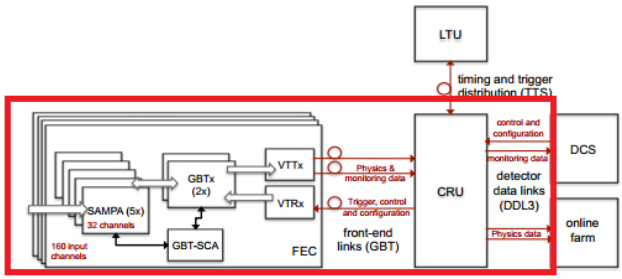


Figure 2.5: Schematics of the readout electronics (From [13])

Chapter 3

Simulations

SystemC, Starting design of the simulation, plans for implementation and test runs

3.1 Simulation

General simulation theory (Should this be in a previous chapter?)

3.1.1 Theory

A simulation can be seen as the imitation of a real-world system and its operations over time. This requires a model representation of the system which is accurate enough to conduct experiments on, which produce real-like results. The model should include key characteristics, specifications and functions of the selected system, but in a simplified fashion. A simulation model can take many forms as it can be used in different contexts ranging from physical object such as electrical circuits, bridges, and even entire cities too abstract systems like a mathematical equation or a scientific experiment.

As the model represent the system itself, the simulation represents its operations over a set period of time. The simulation is normally conducted in a controlled environment that makes it possible to observe, monitor and log results. To achieve efficient experimenting using a simulation, it should be easy to change its parameters with respect to what is being tested.

There are many benefits of simulating a system instead of creating and test the real thing. A simulation will in most cases be very time efficient, you can conduct the same kinds of experiments on the system in a much shorter time compared to the real thing. This means that more information about the systems behavior and limitations in less time, which in turn can result in a better final product. Creating the real-world system can often be very expensive, which may limit the amount of prototypes or test-products that are possible to create. Therefore using results of a simulation to fine tune the specifications before starting to produce prototypes will cut unnecessary development costs by a significant margin.

Taking the upgrade of the readout electronics for the ALICE detector as an example to further address this point one can see the usefulness of not having to create multiple custom hardware components, all with different purposed specification. In regards to the readout electronics, another important point is that the proposed designs might work well from the start, but there is always room for improvement. Finding out that the design doesn't need as much memory, or less optic fiber cables can impact the overall production costs. One way to efficiently and accurately simulate hardware components is by creating a virtual computer simulation.

3.1.2 Computer Simulation

Using computers to do simulations is in this day and age become more and more useful because of their incredible computational power, and ability to produce fast results. This is important as simulations often becomes quite complex, both in regards to computational complexity and hard to work with. Therefore it can be wise to use existing tools to help make the process easier. There is an array of different tools that can be used to various kinds of simulations. They vary from complete frameworks, with graphical user interfaces too tools which helps programmers write there own simulation program. The later requires of course the most work, but will most often end with the better results as you can custom tailor your simulation on a lower level then with a complete framework. A programming tool that is made for creating simulations is the SystemC library, which will be discussed in the following section.

3.2 SystemC

Explain how SystemC works, what benefits and downsides

3.2.1 Background

SystemC is a system design library based on C++. It provides an interface for easily create a software models that representing a hardware architecture, and together with standard C++ development tools it is possible to quickly build a full scale simulation. Following the standards of C++, SystemC is built to be easy to understand for both software and hardware developer, resulting in clearer cooperation between them while developing the hardware design. The SystemC library provides a object-oriented approach to model design, where a single C++ class represents a model. This makes it easy to separate concern between the different models in your simulation.

When simulating a hardware system there is a couple of key points to be aware of, firstly you need to be able to handle hardware timing, clock cycles, and synchronisation. One of the benefits of SystemC is that it takes care of all of this, again taking advantage of the object-oriented nature of C++ to extend its capabilities through `classes`. Here is some of the other features SystemC provides, with emphases on the ones needed to understand code snippets shown in this thesis.

- **Modules**
  - Container `class` representing a hardware model.
- **Processes**
  - In short, processes are methods inside a module which describe the module functionality.
- **Ports**
  - Port represent the input and output points of a module, they can be connected to other modules through Channels. Ports can be both one directional or bidirectional.
- **Channels**
  - Channels are the wires connecting two Ports. SystemC comes with three predefined channels: fifo(First-In-First-Out), mutex, and semaphore. It is possible to configure custom channels, but in most cases it is not necessary.
- **Signals**
  - Signals represent data sent between modules via ports. They can be arbitrary data types like `bool` or `int`, but also user defined types.
- Rich set of data types
  - SystemC supports all data types defined in C++ as well as multiple custom types.
- Clocks
  - SystemC comes with the clocks, which can be seen as timekeepers of the system during a simulation.

3.2.2 Small example

To get a basic understanding of how a SystemC simulation looks like, it is useful to see it in action. The following Figure Figure 3.1 and Listings 3.1-3.3 make up a very trivial example with only 2 modules; a `Producer` and a `Consumer`. The `Producer` will increase a counter every clock cycle, and send a `bool` value based if the count is an even number, and send this value to the `Consumer`, which registers how many times the `Producer` counted an even number. The example uses a First-In-First-Out (FIFO) channel, connected between an output port on the `Producer`, and an input port on the `Consumer`.

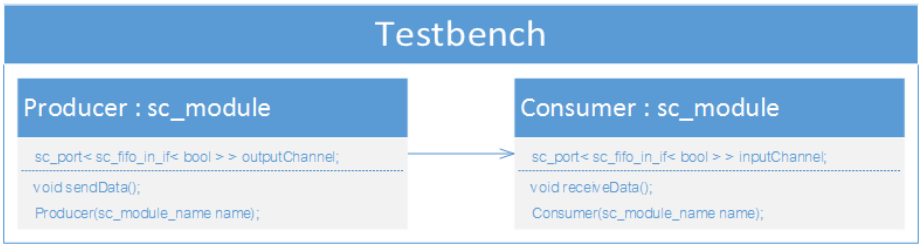


Figure 3.1: Basic SystemC example

```
1  SC_HAS_PROCESS(Producer); //macro to indicate that the module
       has process
2
3  //Constructor with name of module as parameter
4  Producer::Producer(sc_module_name name) : sc_module(name){
5    SC_THREAD(sendData); //Registrer the sendData thread
6  }
7
8  //Thread which runs until the simulation is over.
9  //Clock frequency: 100 Mhz; 1 / 10 ^ 7 = 10 nanoseconds
10 void Producer::sendData(){
11
12   bool signal = false; //signal value
13   int count = 0; // count variable
14
15   while(true){ //infinite loop
16
17     if(!(count % 2)){ // if count is even, signal = true
18       signal = true;
19     }
20
21     outputChannel->nb_write(signal); //write signal to output
           channel
22
23     signal = false; //reset signal
24     count++; //increase count
25     wait(10, SC_NS); //End of a clock cycle, wait 10
           nanoseconds
26   }
27 }
```

Listing 3.1: Hello

```
SC_HAS_PROCESS(Consumer); //macro to indicate that the Module
    has 1 or more processes

//Constructor with name of module as parameter
Consumer::Consumer(sc_module_name name) : sc_module(name){
  SC_THREAD(receiveData); //Registrer the receiveData thread

}

//Thread which runs until the simulation is over.
//Clock frequency: 100 Mhz; 1 / 10 ^ 7 = 10 nanoseconds.
void Consumer::receiveData(){

  int numberOfEvens = 0; // counts number of evens
  bool receivedSignal = false; //received signal variable

  while(numberOfEvens < 10){ //stop lopp when received 10 evens

    if(inputChannel->nb_read(receivedSignal)){ //receiving
        signal; nb_read returns true if signal is read.
      if(receivedSignal){
        numberOfEvens++; // if signal is true, count was even.
      }
    }
    wait(10, SC_NS); //End of a clock cycle, wait 10
        nanoseconds
  }
  sc_stop(); //Force stop simulation.
}
```

Listing 3.2: Hello

```
int sc_main(int argc, char* argv[]) {

  Producer producer("Producer");
  Consumer consumer("Consumer");

  sc_fifo<bool> channel(20); //(First-In-First-Out) channel
      with depth of 20.

  //Connecting Producer-Consumer channel.
  producer.outputChannel = channel;
  consumer.inputChannel = channel;

  sc_start(); //Alternative: sc_start(30, SC_NS) - Specified
      simulation lenght.

  return 0;
}
```

Listing 3.3: Hello

SystemC can be used to create very low level hardware descriptions and models, and can interface directly with hardware description languages like VHDL and Verilog. This is one way to create a simulation, and the models will be very accurately represented by doing so. The other way is to have a high level of abstraction, leaving out the unimportant details and focus solely on the expected problem areas. There are benefits and drawbacks for both ways, but sticking to a high abstraction level can in complex cases make it a lot easier to work with the model design and allows you to focus on the important parts.

Chapter 4

Problem Description

Explain the model, introduce the problem

Fler nesten at det mangler noe her..... (Trenger jeg en introduksjon her?)

4.1 Model Design

Different design patterns, and plans for the electronic The hardware design which is being simulated is already briefly shown in figure **??**. The proposed schematic shown there consists of 12 FEC cards for every CRU. Each FEC consists of 5 SAMPA and 2 GBTx ASICs, with the CRU being connected to them via 24 optical links. Out of the 3 main chips, the SAMPA and the CRU are the most interesting as they are still being developed and testing them can give a lot of valuable feedback. The GBTx is a completed component, so even though it is part of readout electronic being simulated, it will only be a very shallow abstraction of it. This means that it will remain as an empty module who's objective will be to just pass along received data to the correct output links. One important note about the GBTx input and output links. Each GBTx has 10 input e-links, each with a transfer rate of 320 Mbit/s, giving an effective input speed of 3.2 Gbit/s per GBTx. The output is 1 optical fiber link with a speed of 3.2 Gbit/s, giving the GBTx the same input and output speed. This is the reason letting data flow directly through the GBTx in the simulation is possible. The next sections will go into details about the more important components.

4.1.1 SAMPA

The SAMPA ASIC is based on the work from its predecessor, the ALTRO chip. Just like the ALTRO chip it will be the first stop for signals being tracking in the TPC detector. The signals will be processed, compressed, digitized, and temporarily stored in the SAMPAs memory before it is passed along. The SAMPA has 32 integrated channels, which separately and asynchronously process the analog signals coming from the detector.[13] Each channel has a readout speed of 10 bit on a 10 MHz clock, which combined results in 3.2 Gbit/s. The channels also have there own FIFO buffer memory where signals coming in are

stored as they wait to be sent along. The most efficient size for these buffers are one of the things the simulations will hopefully provide. The output links for the SAMPA chip consists of 4 e-links connecting them to the GBTx. Each e-link has as said in the previous section a speed of 320 Mbit/s, which sums up to 1.28 Gb/s.[12] Since each SAMPA and GBTx has a specific number of output and input links, there is only certain setups which are desirable. This is why the proposed schematic uses 5 SAMPA and 2 GBTx chips for each FEC. That setup gives exactly 20 output links from the SAMPA chips, and 20 input links on the GBTx chips.

As the ALTRO, the SAMPA can be run in triggered readout mode, but in addition can be run continuously. Being able to read out continuously is a necessary upgrade to handle the increased data load coming from the detector. During continuous mode the data acquisition is uninterruptable, meaning that there is no pause between reading two consecutive events from the detector. The difference it makes compared to triggered mode can be seen in figure **??**. Every event, from now on referred to as time frames, is 1024 clock cycles long, and all 32 channels of the SAMPA use the same time frame. This means that every 1024 clock cycle a 1024 long time window is initiated for all 32 channels, meaning they can readout 10 bit data samples 1024 times during this window. A synchronization input allows multiple SAMPA ASICs to align their time frames with respect to each others.[12]

The SAMPA creates data packets from the data assembled from each time frame. Consisting of a header of fixed size 50 bit, followed by a list of 10 bit samples, created from a single time frame. Even though a time frame consists of 1024 clock cycles, in practice a maximum of 1022 samples are received each time. This is due to 2 * 10 bit words is required to represent cluster size (size of consecutive samples) and a timestamp. Figure Figure 4.2 shows the structure and format of the packets.

The header consists of information regarding the data, such as address for the channel and chip, number of data words in the time frame and packet type. The packet type is used as a marker to see if anything out of the ordinary has happened to the data. This can be if there is no samples in the time frame, causing the packet type to just become a channel fill packet. It can indicate of the stream of data was cut short because the FIFO buffer was full, causing buffer overflow. In the case of buffer overflow all data for the particular time frame are discarded and the empty packet is sent with type overflow. Overflow can cause a lot of data to get discarded if the SAMPA can't empty the buffers fast enough, this can happen if the buffers don't have enough space. Seeing as the input rate is 3.2 Gbit/s and the readout speed is 1.28 Gbit/s, the SAMPA can receive up to 2.5 more data per second then it can pass along. This is why the FIFO buffers are necessary, and finding a size which is sufficient, without going overboard is crucial.

There have been done some calculations on how much data will actually be received from the detector at any given time. It is estimated that on average over all channels for every SAMPA there is around 30% occupancy. This means

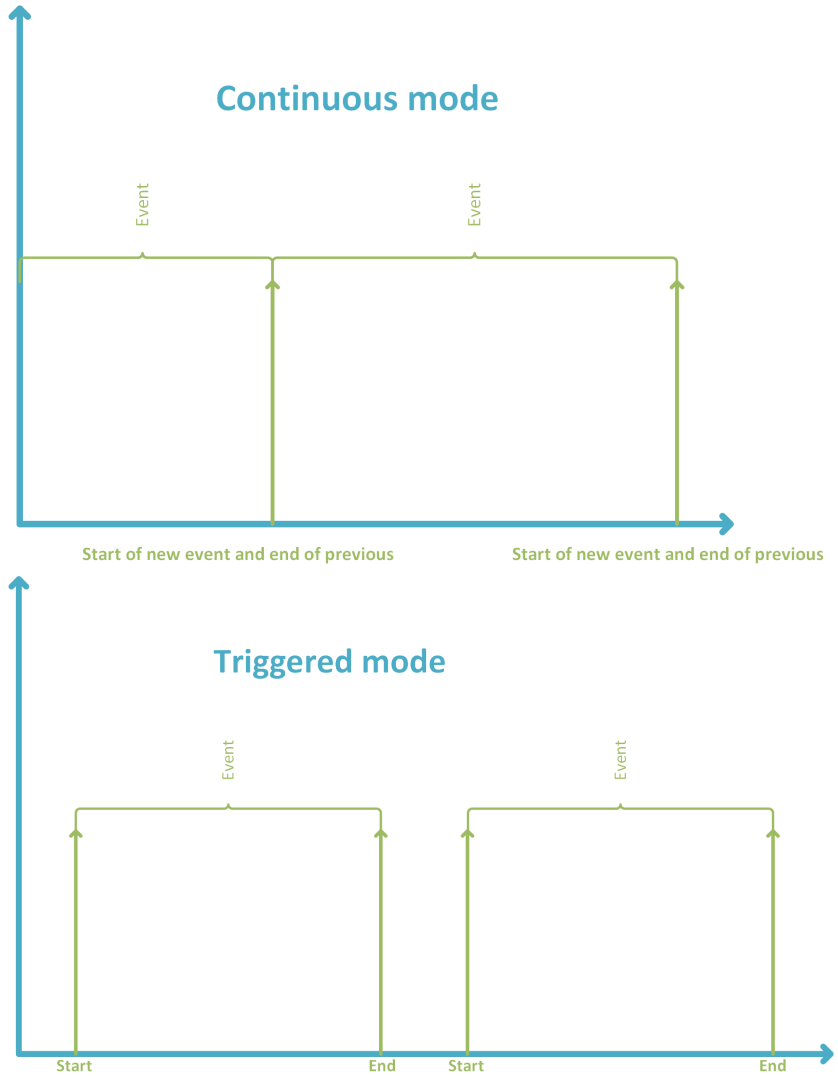


Figure 4.1: Continuous vs Triggered mode

that on a global average there is 30% data in every given time frame. Some channels may be full while others are empty, and some may have 40%, but on average there is 30%, which means 306 samples out of 1022 for every time frame. Taking this into account when calculating the input speed of the SAMPA gives 960 Mbit/s which the design should be able to handle without any buffer overflow. Even though there is an estimated average occupancy there can still be some channels which time frame after time frame gets a lot more then that, so how much can the design handle? This is some of the question the simulation will give answers to.

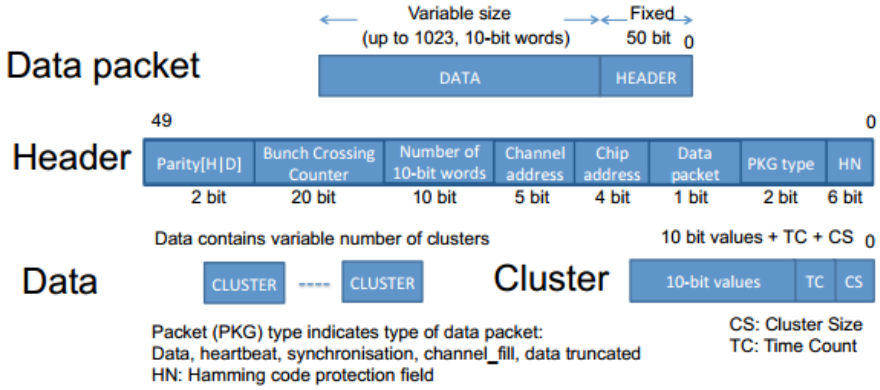


Figure 4.2: Data packet format (From [12])

4.1.2 CRU

The CRU serves as an interface between electronics directly on the detector and the online computing systems. It is based on Field-Programmable Gate Array (FPGA) processors, with optical fiber used as input and output. *Hva brr jeg ta med her?*

4.2 Signal processing in the SAMPA

The SAMPA chips will receive and process a huge amount of data, both relevant signals and background noise. In section 4.1.1 we talked about occupancy and amount of samples in each time frame. The estimated amount of 30% refers to relevant samples, removing or compressing the background noise. Seeing as it will always be some interference in the background, there will always come samples with data, and gathering all will be a waist of time and space that could be used on the actual collision data in the detector. Figure Figure 4.3 shows 2 actual events collected from the 2 different ALTRO channels, the events will look similar after the upgrade and we can use this as a starting point. The x-axis expresses the current time bin within a time frame from 0 to 1021. Here one can see that every sample in the time frame has some value most with 48-52, as well as certain peaks here and there. Those peaks are what is interesting, everything else is considered noise and should be removed. *Hva forusaker sty?* There are a number of ways to reduce the amount of noise, and/or compress the data to a manageable size. What has been used with the current setup and is also discussed to use in in the upgraded setup is Zero suppression.

4.2.1 Zero suppression

The Oxford Dictionary of Computing (6 ed) defines Zero suppression in the following way: "The elimination of nonsignificant zeros. While numerical data is being processed it may be expanded to a uniform number of digits by the addition of nonsignificant zeros to the left of the most significant digit. For printout or display these nonsignificant zeros may be suppressed."[14]. This is

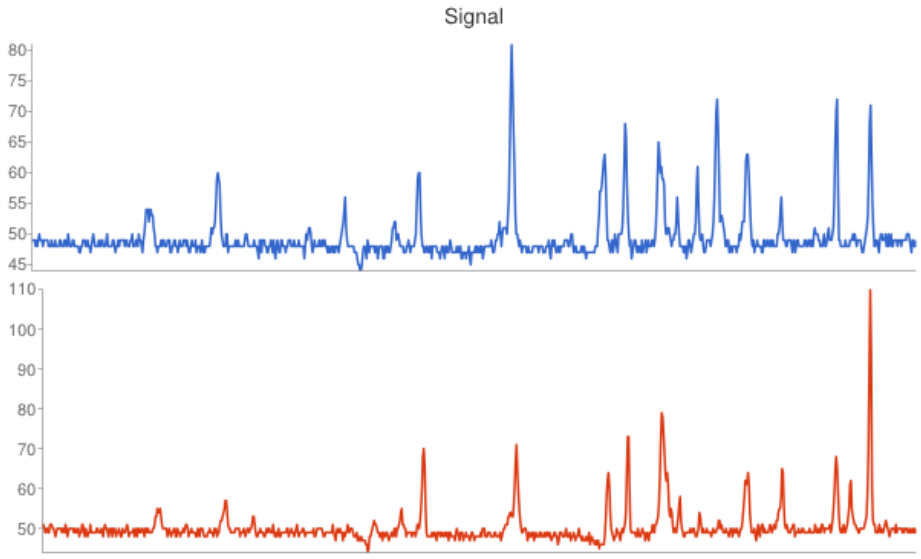


Figure 4.3: Two signals from a previous experiment

a pretty good summary of what you wish to accomplish with Zero suppression. To remove the background noise without discarding any important samples, a baseline for the Zero suppression must be established. The problem with this is that the baseline may shift, in the case of our 2 example time frames the first one has a visibly lower baseline by 1 or 2. In the upgrade plans described in [12] it specifies how this signal processing will take place. A short summary is that it is performed in 4 steps:

- Initial correction by subtracting the signal using a fixed, previously set signal baseline.
- Tail cancellation filter used to cancel slowly varying samples (e.i samples with values close to each other without large spikes).
- A second baseline correction which using both a global threshold for each SAMPA and one for specific to each channel.
- The actual Zero suppression which removes everything under a threshold.

Stemmer dette? In some later discussion regarding the upgrade there has been questions if the described method is insufficient. The theory behind the discussion is that the baseline shift will be to great? to be able to do efficient Zero suppression without loosing important samples in the process. This encourage finding another way of processing the signals. One proposed method is to use Huffman coding on the signal values.

4.2.2 Huffman Coding

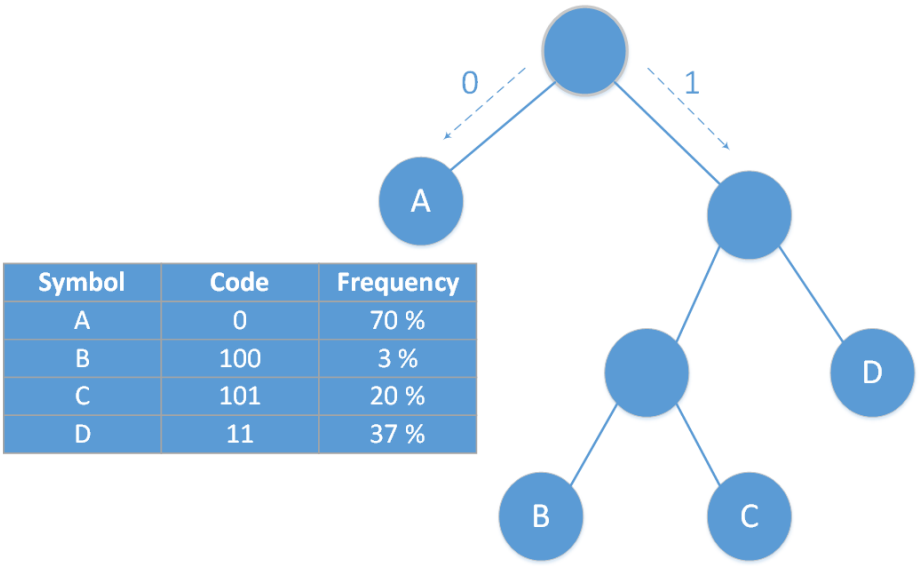
Huffman is a method used to achieve data compression.[15] It works by assigning binary codes to symbols in order to reduce the number of bits used to encode

the symbol. By looking at the frequency of appearance for every symbol used one a frequency table sorted by most frequent. One thing to note is that since the binary codes is of variable length, they may not all be uniquely decipherable. For instance, if the codewords looks like the following: {0,01,11,001}, the code 0 is a prefix to 001. This is solved by using the right data structure to store the codes, the one most used is a *full* Binary Tree (BT). A *full* BT is a tree where every node either has zero or two child nodes. The symbols are then generated by the path from the root to a leaf node, where left and right indicates 0 or 1. Figure Figure 4.4 shows an example of a Huffman tree using made up frequencies for the letters A to D. Here you can see the advantage of sorting by frequency, since the most frequent symbol A only needs one bit to store. Creating the Huffman tree can be implemented using the following pseudo-code algorithm:

```
1  //Input: An array f[1..n] of frequencies
2  //Output: An encoding tree with n leaves
3  //let H be a Priority Queue of integers, ordered by f
4  function Huffman(f) {
5    for(int i = 1; i <= n; i++){
6      H.insert(i);
7    }
8    for(int k = n+1; k <= 2n - 1; k++){
9      i = H.deletemin();
10     j = H.deletemin();
11     //Create a node numbered k with children i,j
12     f[k] = f[i] + f[j];
13     H.insert(k);
14   }
15
16 }
```

Listing 4.1: Huffman algorithm [16]

Need to have Dieter look over this paragraph In the context of compressing data coming from the detector there are one particular foreseen complication. First of, generating the Huffman tree needs values from the detector, so how do one create a tree with high compression factor without knowing this? One answer to this is to generate a tree using existing data from previous experiments, but update the tree when receiving new data. This gives us a uncertain compression factor in the beginning, but it will become better over time. Because of a shifting baseline encoding the signal values directly may lead to a large Huffman tree, and the best tree for one channel may not be the same for another. It is inefficient to create a separate tree for each channel, as there will be 160 channels for every FEC. A possible solution to this is to encode the derivative of each signal in a time frame compared to the previous value. In other words, for every signal n you store the value: *signal(n) - signal(n - 1)*. Doing so takes away the problem caused by shift in the baseline as it only stores the difference between two signals. This method requires that the first value of every time frame is stored somewhere(maybe the header of a SAMPA packet) in order to decode it later on.



Symbol	Code	Frequency
A	0	70 %
B	100	3 %
C	101	20 %
D	11	37 %

Figure 4.4: Huffman tree with four symbols.

4.3 Designing the simulation model

With all of the information regarding the different components already specified, creating a simulation model should be more then feasible. There will be in total 3 main modules part of the simulation: the SAMPA, GBTx and CRU. The tasks, objectives and goals that this all boils down to is summarized in the table below.

- **Tasks**
  - Designing a model which is accurate, simple and customizable.
  - Create a simulation test bench that allows for quick changes in order to run multiple simulations.
  - Run different stress tests on the system, find out where it breaks and why.
  - Run focused simulations on the SAMPA channel buffers.
  - Run simulations which compares Zero suppression and Huffman encoding.
  - Gather, and compile the simulation data into a readable and understandable format.
- **Goals**
  - Verifying that the design can handle the expected data.
  - Focus on the
  - blabla
  - blabla

Closing words

Chapter 5

Solution implementation

Code snippets, Incremental implementation stages and the final implementation, (before and after huffman), using real data vs random. Implementing fluxiation into the simulation

5.1

Chapter 6

Evaluation and results

Running the tests, results from different tests, Evaluating the final product

Chapter 7

Conclusion and Future work

Conclude the thesis, talk about the impact it has and its usefulness in future planing of the front end electronics.

Bibliography

[1] Long Shutdown 2 @ LHC. https://indico.cern.ch/event/315665/session/7/contribution/37/material/paper/1.pdf. Accessed: 2015-01-09.

[2] Werner Riegler. The ALICE Upgrade plans - Article. http://ph-news.web.cern.ch/content/alice-upgrade-plans. Accessed: 2015-01-12.

[3] CERN - Article. http://home.web.cern.ch/about. Accessed: 2015-01-12.

[4] The birth of the web - Article. http://home.web.cern.ch/about. Accessed: 2015-01-12.

[5] The Large Hadron Collider - Article. http://home.web.cern.ch/topics/large-hadron-collider. Accessed: 2014-11-14.

[6] The Large Hadron Collider - Brochure. http://cds.cern.ch/record/1165534/files/CERN-Brochure-2009-003-Eng.pdf. Accessed: 2015-01-16.

[7] The ALICE experiment - Homepage. http://aliceinfo.cern.ch/Public/en/Chapter2/Chap2Experiment-en.html. Accessed: 2015-01-17.

[8] Quark-Gluon plasma - Article. http://home.web.cern.ch/about/physics/heavy-ions-and-quark-gluon-plasma. Accessed: 2015-01-18.

[9] The ALICE experiment - Article. http://home.web.cern.ch/about/experiments/alice. Accessed: 2015-01-17.

[10] ALTRO - Article. http://aliceinfo.cern.ch/Public/en/Chapter2/Chap2_TPC.html. Accessed: 2015-02-13.

[11] ASIC - Definition. http://www.radio-electronics.com/info/data/semicond/asic/asic.php. Accessed: 2015-02-13.

[12] Upgrade of the Readout Trigger System - Techincal Design Report. http://cds.cern.ch/record/1603472/files/ALICE-TDR-015.pdf. Accessed: 2015-02-13.

[13] Upgrade of the ALICE Time Projection Chamber - Techincal Design Report. http://aliceinfo.cern.ch/Public/en/Chapter2/Chap2_TPC.html. Accessed: 2015-02-13.

[14] Daintith and Wright. zero suppression. "http://www.oxfordreference.com/10.1093/acref/9780199234004.001.0001/acref-9780199234004-e-5900". Accessed: 2015-02-24.

[15] Ince. Huffman coding. "http://www.oxfordreference.com/10.1093/acref/9780191744150.001.0001/acref-9780191744150-e-1565". Accessed: 2015-02-25.

[16] Dasgupta Papadimitriou and Vazirani. *Algorithms*. Alan R. Apt, 2008.