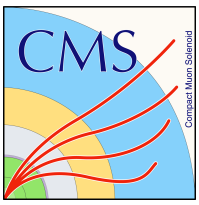
In cooperation with:



Trigger-System Development for the HGCAL Cassette Testing

Supervisor:
Zoltan Gecse

Intern:
Mattia Tinfena

Final report
Summer internship 2023

Contents

Introduction

Abstract

The Large Hadron Collider (LHC) will be upgraded to increase the instantaneous luminosity, which will result in an increase in the number of collisions from the 15 pile-up events of the first run of phase 1 to the 150-200 pile-up events of phase 2. Increasing the number of collisions means increasing the amount of radiation that the experiments will absorb. To meet these challenging conditions, the CMS calorimeters need to be upgraded. The new version of the hadronic HGCAL (High Granularity CALorimeters) in the endcaps will be assembled and tested in Fermilab's laboratory. This report describes the new trigger system for testing the HGCAL's cassettes.

1.1 Introduction to LHC

The Large Hadron Collider (LHC)1.1 is the world's largest and highest energy particle accelerator, located at CERN between Switzerland and France. It is a proton-proton collider of 17 mi in circumference with a depth of 574 ft. In the ring there are four crossing points where the accelerated particles collide. In correspondence of these crossing points there are 4 detectors each designed to detect different phenomena: CMS, ATLAS, ALICE and LHCb. LHC should have been turned off at the end of the run 3 that is currently ongoing. However since the discovery of the Higgs Boson in the 2012, it was decided to upgrade it for phase 2 called HL-LHC (High Luminosity LHC).

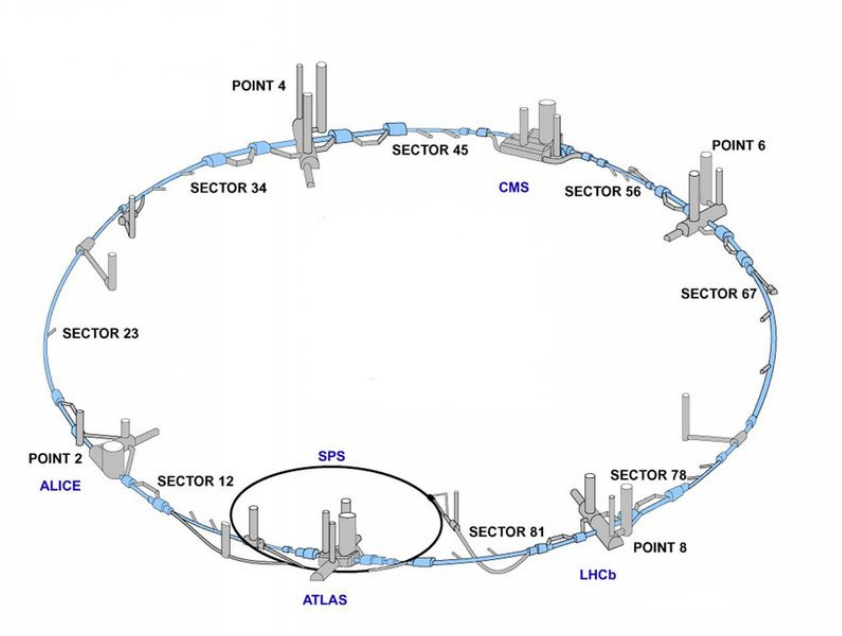


Figure 1.1: Scheme of LHC.

1.2 Introduction to CMS

The Compact Muon Solenoid (CMS) experiment is one of the two (with ATLAS) large general-purpose particle physics detectors built in 2008 with the following goals: the search for the Higgs Boson and the experimental validation at high energies of the standard model.
Collisions take place in the center of the detector, and than the interaction point is surrounded by various layers of sensors. The principal sensors are in the barrel an in the endcaps and are:

- The tracker;
- The electromagnetical calorimeters;
- The hadronic calorimeters;
- The magnets;
- The muon chambers;

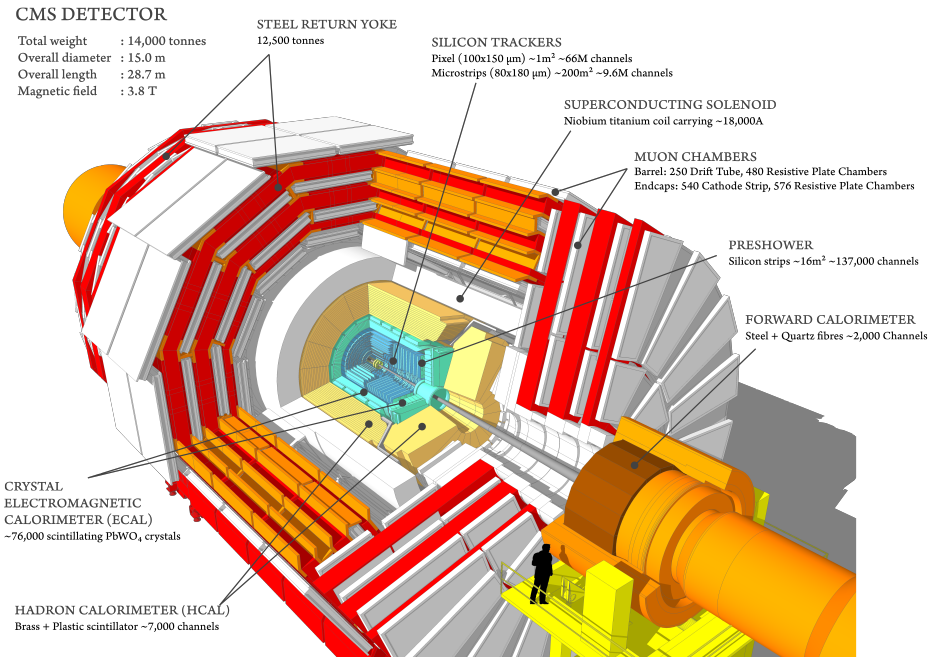


Figure 1.2: CMS experiment.

CMS and ATLAS discovered the Higgs Boson in 2012, but for these experiments, a lot of work has still to be done. With the new phase of LHC the next objectives are:

- Explore physics at the TeV scale;
- Further study the properties of the Higgs boson;
- Look for evidence of physics beyond the standard model, such as supersymmetry, or extra dimensions;
- Study aspects of heavy ion collisions.

1.3 HGCAL

By increasing the number of collisions in the LHC by a factor of 10, the amount of radiation that the sensors have to absorb increases by 10 times. The new version of the hadronic calorimeters in the endcaps are called HGCAL. The part exposed to high radiation is covered by silicon sensors, while the low radiation area is covered by plastic scintillators. The silicon sensors are arranged in cassettes, with each cassette containing about 50 silicon sensors[3].

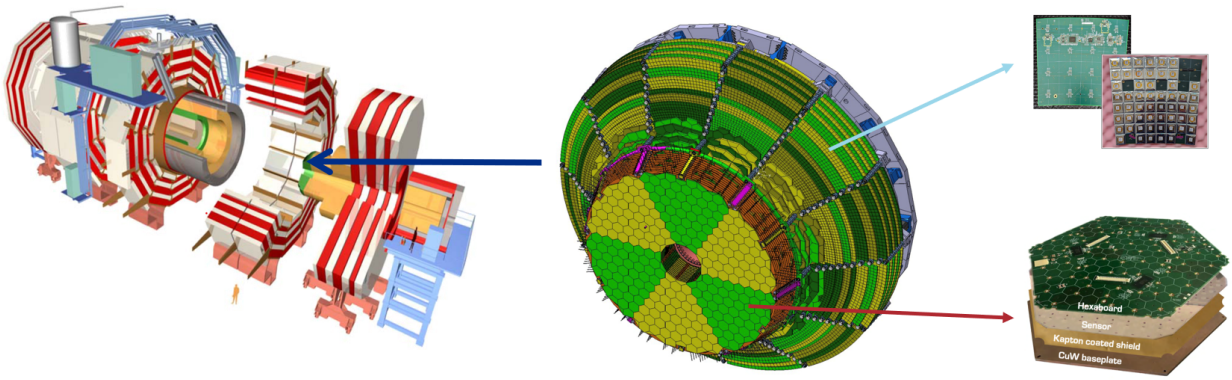


Figure 1.3: HGCAL in the endcap of the CMS experiment, in the higher part there are the plastic scintillators, in the lower part there are the silicon sensor.

The main requirements [4] for the HGCAL upgrade can be summarized as follows:

- **Better radiation tolerance:** to withstand radiation levels 10 times higher than current levels;
- **Higher density:** to preserve lateral compactness of showers;
- **Higher fine lateral and longitudinal granularity:** to enable particle flow algorithms and associate energy clusters to individual tracks;
- **More precision measurement of the time of high energy showers:** to obtain precise timing from each cell with a significant amount of deposited energy;
- **Ability to contribute to the level-1 trigger decision.**

1.4 HGCAL's cassettes at Fermilab

All the 525 cassettes for the hadronic portion of the HGCAL will be produced and tested at the Fermilab and assembled in the SiDet facility.The cassettes will be assembled in the Lab C where there are 4 assembling station and 2 cold rooms for long term testing[2].

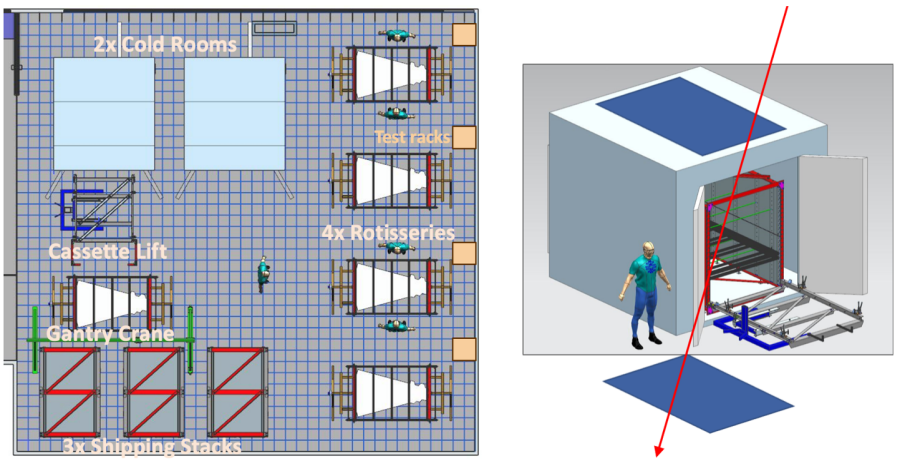


Figure 1.4: On the left the assembling station, one the right the cold room for testing the cassettes.

The cassette burn-in test stand will have about 20 cassettes stacked and taking cosmic data for 2 weeks. Will be installed 2 planes of scintillators, above and below the cassettes to detect cosmic events and trigger the recording of the corresponding Trigger and DAQ events.

Trigger-System Requirements and Technology

2.1 Trigger constraints

2.1.1 Time resolution

The charge collected in the silicon sensors of the HGCAL detector is measured by sampling the amplitude of the signal when it peaks. This measurement is done once every 25ns. The width of the peak is about 1ns as shown in Figure 2.5. So it is important that only those cosmic muons are used for calibration that are within 1ns from the rising edge of the 40MHz clock of the HGCAL detector. Therefore the cosmic trigger system needs a capability of measuring the time of arrival of the cosmic muons with a 1ns time resolution.

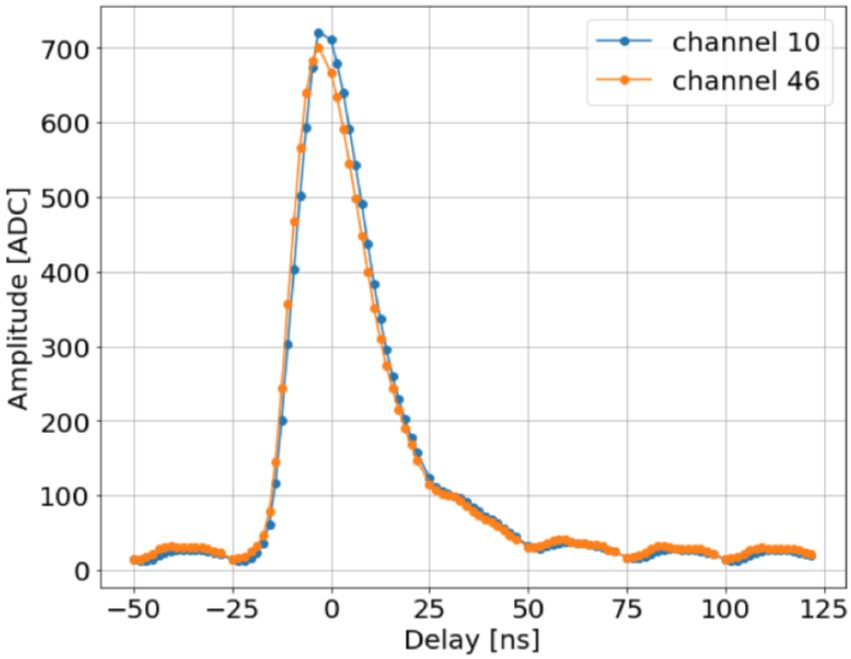


Figure 2.5: Shape of the trigger signal.

2.1.2 Space resolution

Knowledge of the exact muon path is not needed for triggering. The system upgrade to allows the offline analysis using the x-y location of the scintillator trigger hit should be too expensive. So, we do not need spatial resolution in the scintillator trigger, 1 channel per layer is enough.

2.2 Technology adopted

The single channel requirement allows using a single slab of scintillator or an array of extruded scintillators with WLS fibers merged on one PMT. Using 2 PMTs allows to calculate the mean time of arrival that is independent of the location of the hit along the WLS fiber. It appears the hit is always in the middle of the fiber, regardless of the true origin. This eliminates the 25ns uncertainty caused by the size of the plane and the length of the fibers (5mt).

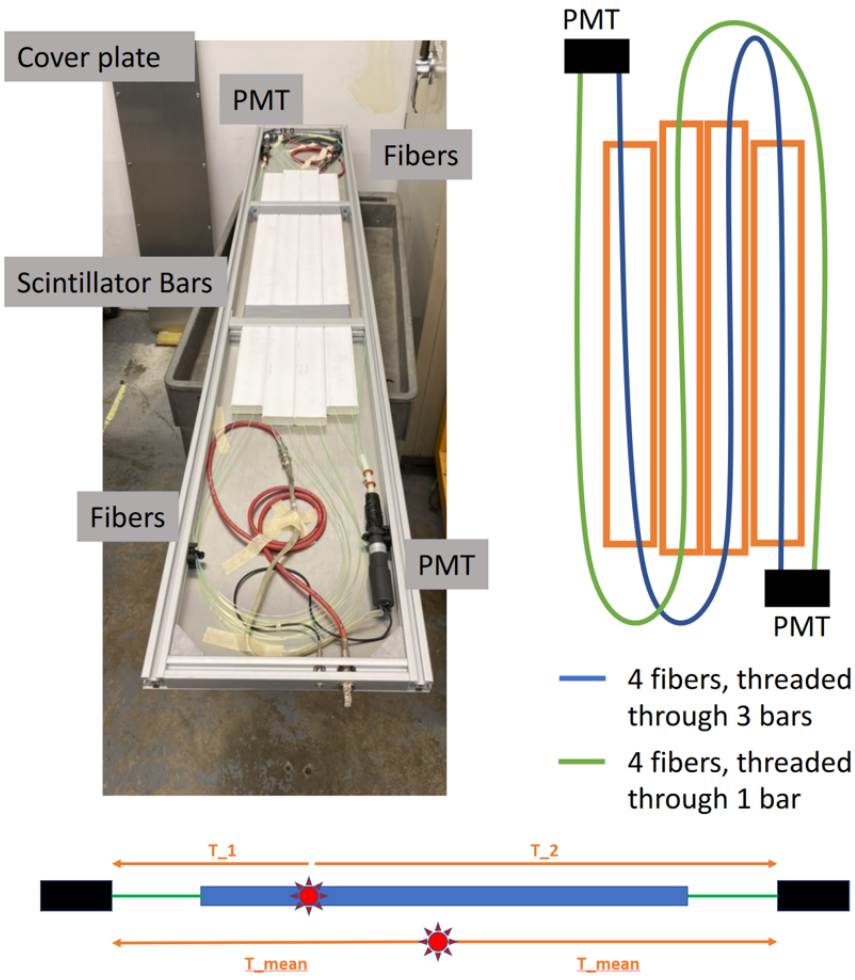


Figure 2.6: Testing system setup[1].

All the system is driven by the 40 MHz clock, so we need to compute the arrival time of the cosmic muon respect the edge of the clock and save it to the disk during the tests. To compute this large amount of information in this small period of time we need an high performance hardware, for this reason we choose to use an FPGA (Field Programmable Gate Array). To connect the fiber system with the FPGA we need a tailored input board.

Development of the Trigger Interface Card

3.1 Actual testing system

When I arrived at Fermilab, the system was made up of two scintillator planes with four PMTs3.7. The signal from the PMTs passed through an amplifier and a CFD (Constant Fraction Discriminator). To compute the meantime, there were three meantimers. The final signal from the meantimer was read by a ZCU102 through a DIO5 input board and sent to an oscilloscope to be saved on a PC.

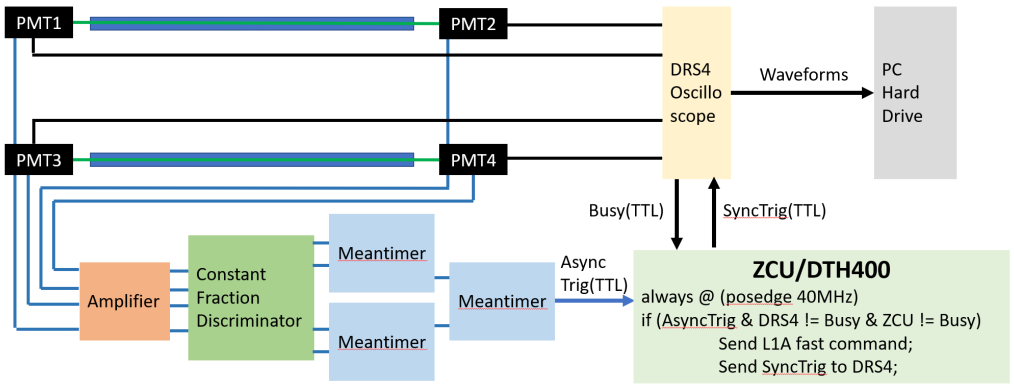


Figure 3.7: Testing system setup.

3.2 New testing system

The DIO5 is a bord made from cern to use the HPIO (High Performance Input Output) pins of the FPGA. The ZCU102 has two HPIO slots, but will be used in the final project to send data out at high speed trough optical fibers. We need a new setup to read the data from the PMTs, the idea is to read the data directly from the CFD. In this way, we can read 4 slower input and compute the meantime directly in the FPGA.

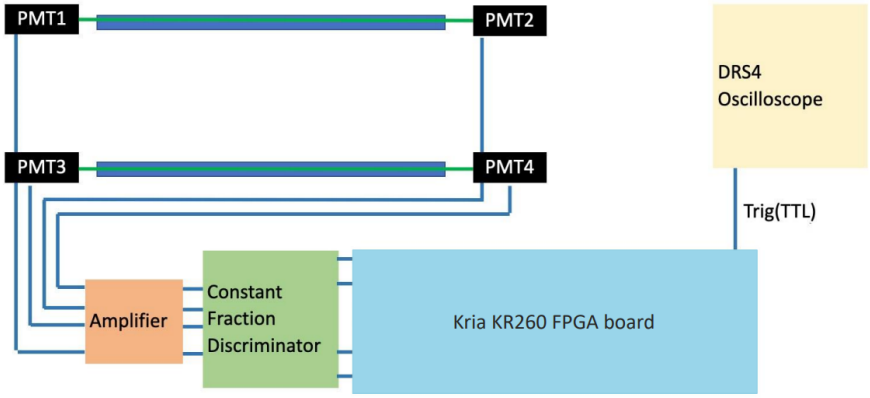


Figure 3.8: Testing system final setup.

3.3 Trigger interface card

The better solution is to use the PMOD connectors that are high-density pins. To use the PMOD pins, we need a new tailored interface card. To test this new system, we bought a breadbord with the pmod connectors and we designed an input stage for each channel to protect the FPGA from overcurrent and overvoltage.

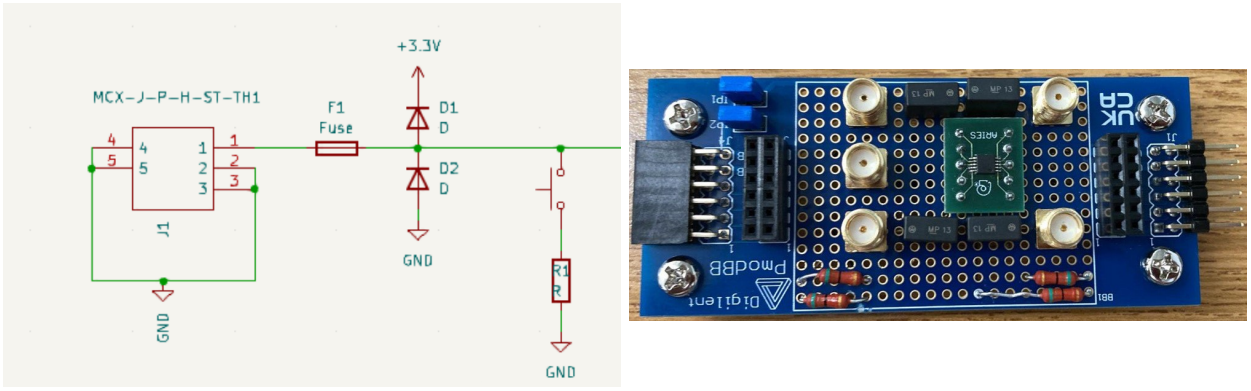


Figure 3.9: On the left the schematic of each input stage, on the right the photo of the first version of the board.

On the same board we provide also an output stage to test the quality of the trigger signal compared with the trigger signal from the meantimers. During the first test the results where encouraging but we noticed a lot of disturbance on the output caused by the crosstalk with the input.

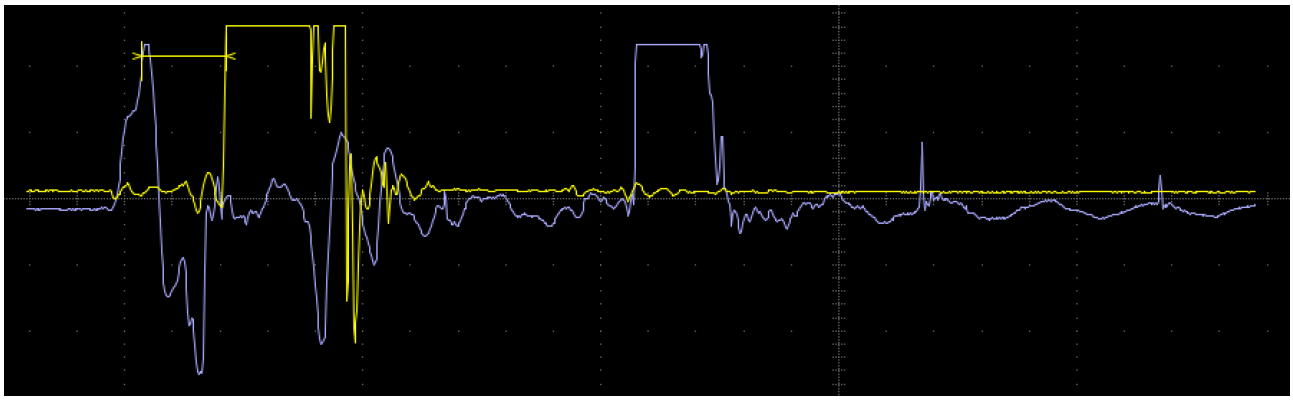


Figure 3.10: In yellow the signal from the meantimers, in violet the signal from the FPGA.

To solve the crosstalk problem, we developed a new board only with the input stage. We kept the old board for the output in a different PMOD connector. For the test, we used the Kria KR260 board that has 4 PMOD connectors and the same SoM as the ZCU. With this new setup, we solved the problem of crosstalk.

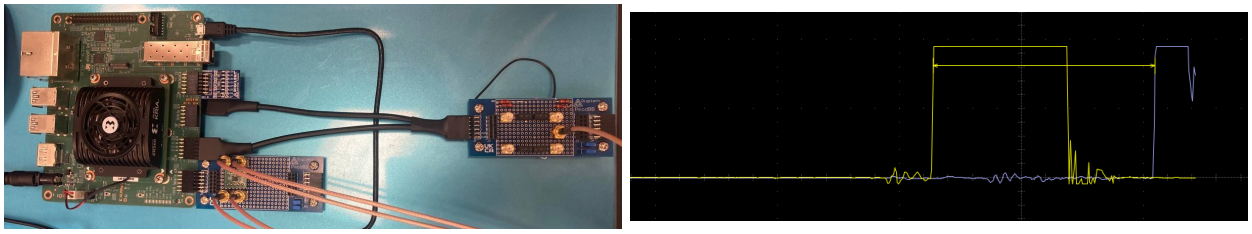


Figure 3.11: The setup used to solve the crosstalk problem.

3.4 Final board

After obtaining positive results, we designed a Printed Circuit Board (PCB) using KiCad for the final project to use with the ZCU. The board has 4 connectors that can be used as output or input (soldering the termination resistor). Each channel is separated from the others by a ground plane to avoid crosstalk. The track size is reduced to avoid delays. The connectors are on the top layer, and all the other components are on the bottom layer to avoid unintentional contacts. Using multiple boards connected to different PMOD connectors will make it possible to keep the inputs and outputs separated. To solve the limitations of the PMOD connectors, each input is sent through 2 different pins.

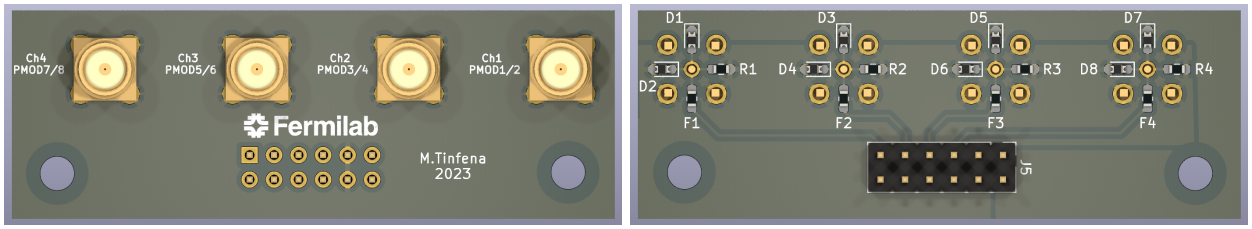


Figure 3.12: Top layer and bottom layer of the final PCB.

Firmware development for the trigger system

4.1 PMOD limitations

The PMOD connectors have a maximum speed of 250 MHz. To read 32 bits every 25 ns, we need a speed of 1.28 GHz. To achieve this speed, we connect two pins to have the same signal and read the data at 320 MHz using the DDR (Double Data Rate) technology. The first input is read with the 320 MHz main clock, while the second input is read with the same clock delayed by 90 degrees of phase.

4.2 Firmware structure

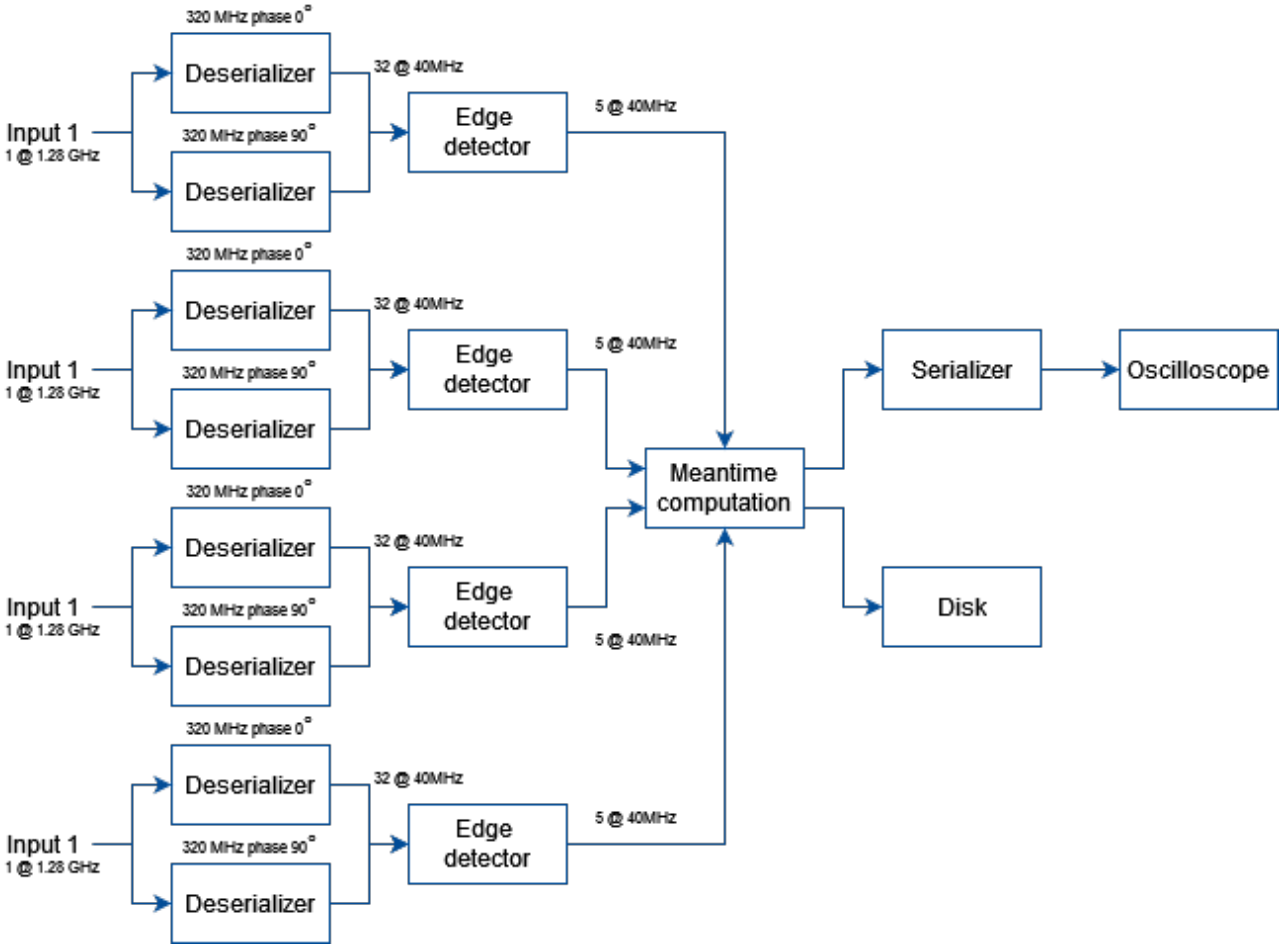


Figure 4.13: Firmware diagram.

The system has four serial input, each one is sent to two deserializers, and then a 32-bit word is created. The 32-bit word is then scanned to search for the peak (the muon impact). Once the peak

is detected, the arrival time is computed. If a peak is detected in all four inputs at the same time, the overall meantime is computed and saved in memory, and the trigger signal is sent out. During the test to compare our results with the meantimers, we sent out a serialized signal to the oscilloscope.

The entire project was implemented in Vivado 2021.2. All clock sources are generated with the clock wizard and are derived from the clock provided by the Zynq Ultrascale+ SoM. A processor system reset is provided as reset source in case of unwanted blocks. For debugging purposes, a block to control the correct operation of the clock was created. This block toggles an LED with a period of one second if all clocks work properly. The main block also has a feature to toggle an LED for each input every time that a peak is detected and each time a trigger signal is sent out. In this way, it is possible to visualize through LEDs that everything in the testing system is well connected and working.

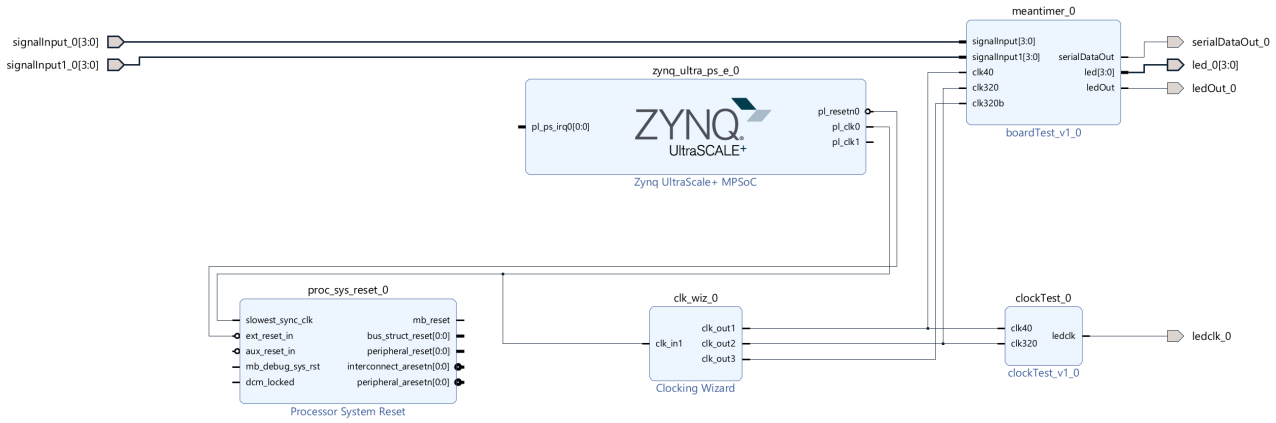


Figure 4.14: Block design.

Testing of the trigger system

5.1 Test setup

To test the input board and the firmware, we constructed a test setup with two prototype trigger planes. We assembled the first plane with a BCF91 fiber, which has a 7 ns decay time. The second plane was assembled with a BCF9929A fiber that has a 2.5 ns decay time. To compare the performances of the two fibers, we measured the time resolution by comparing the timing of two planes using the setup shown in the image below.

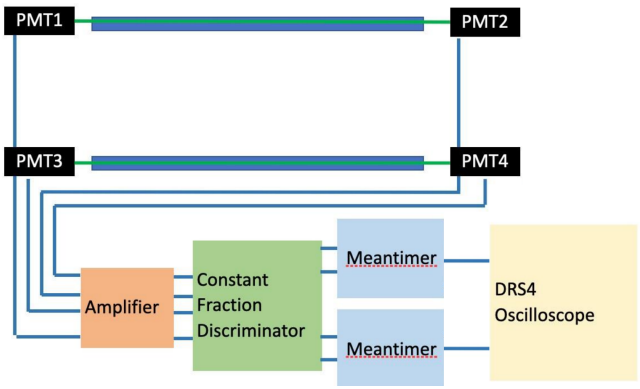


Figure 5.15: Setup to compare the fiber's performance.

The test showed a resolution of about 2 ns for the difference with fast fibers. Considering that we have to compute the meantime, the resolution falls to about 1 ns. This result allows us to respect the time resolution constraint to detect correctly the peak of the trigger signal corresponding to the hit of the cosmic muon.

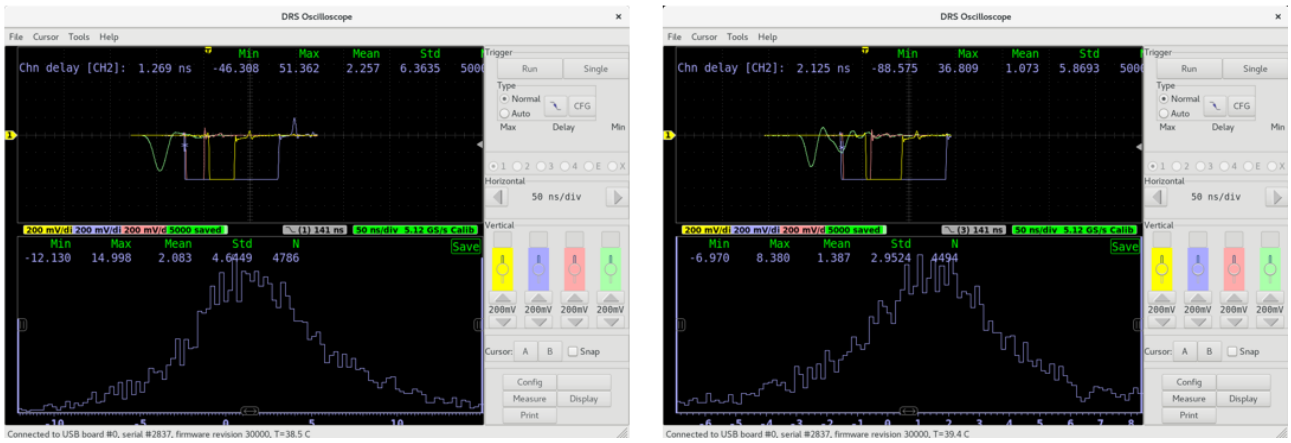


Figure 5.16: On the left the old fibers, on the right the new ones.

5.2 Final results

After preparing the testing system, we tested the input board and firmware. The board was produced by OSHPark and tested with the final setup (two separate boards to avoid crosstalk). The boards acquired data properly. Finally, we compared the firmware performance with the meantime computation from the commercial NIM system. The results were positive, and we obtained a standard deviation of 1 ns. This result validated the PCB layout, FPGA input clocking, and firmware logic.

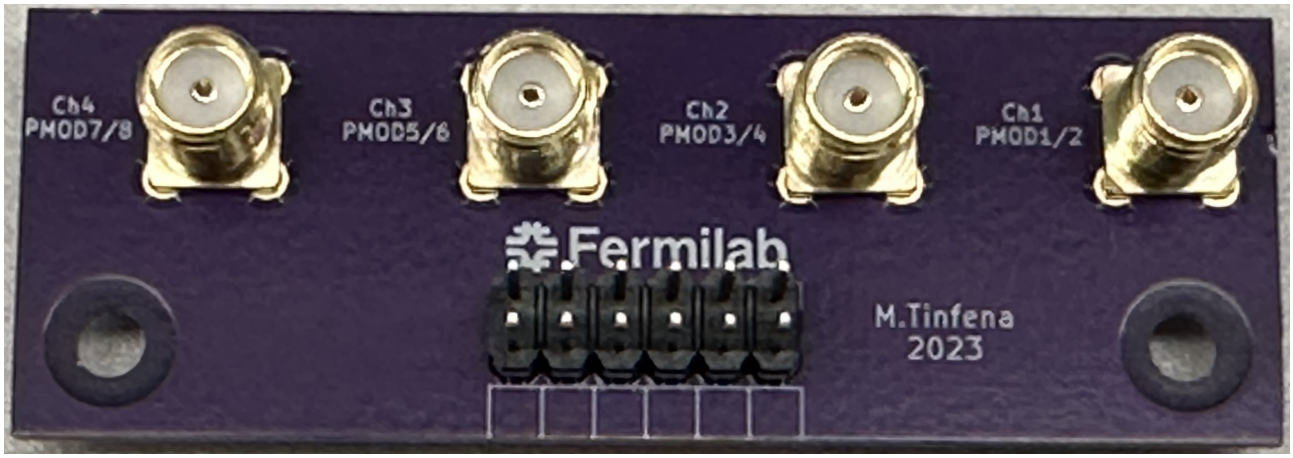


Figure 5.17: The photo of the produced board.

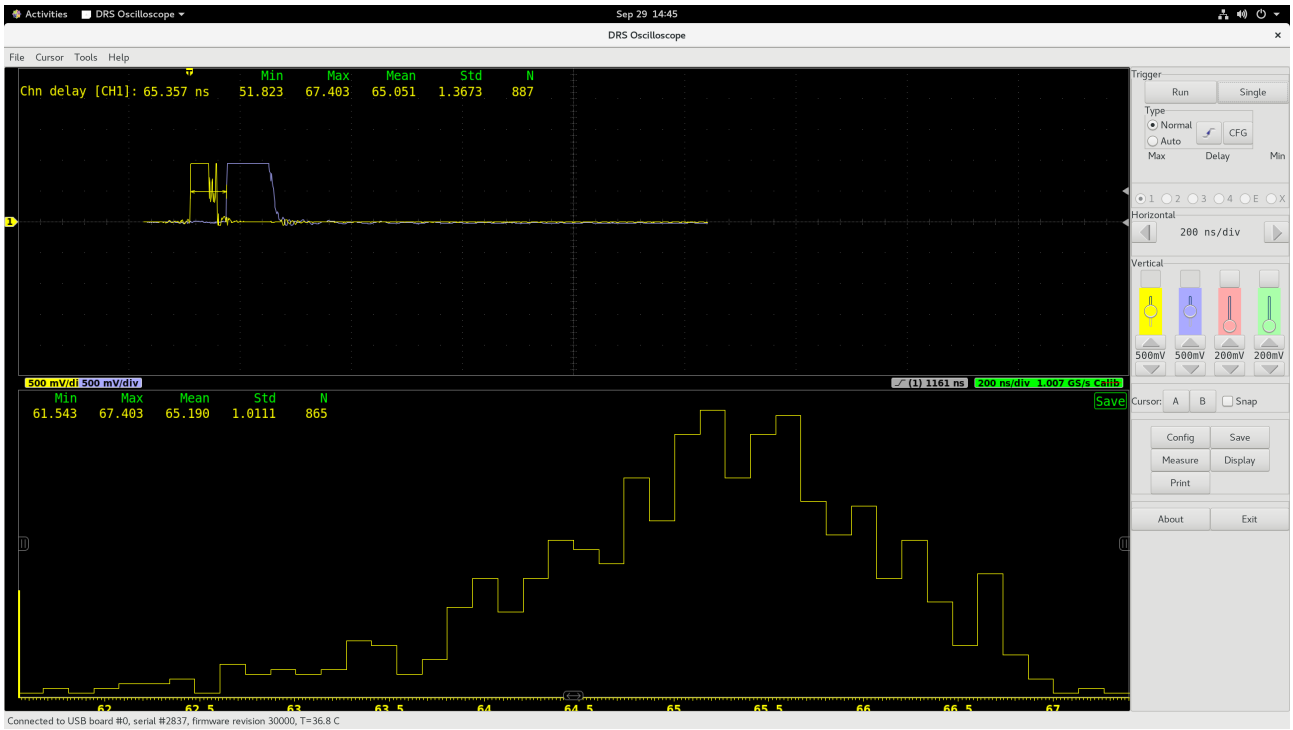


Figure 5.18: Comparison between the NIM output (yellow signal) and FPGA's meantime computation(violet signal).

Summary

During my internship, I had the opportunity to work on a variety of tasks related to developing a trigger system. Specifically, I was responsible for assembling the scintillator trigger planes with WLS fibers and connecting them to PMTs. This involved working with delicate equipment and ensuring that the components were properly aligned and connected. I also connected the trigger planes to the NIM logic system, which required me to have a deep understanding of the system's architecture and how it functioned.

In addition to these tasks, I developed a PMOD interface card to bring the signals to an FPGA board. This required me to have a strong understanding of digital electronics and how to design circuits that could interface with other systems. I learned how to program FPGA with the Vivado suite and the SystemVerilog language, which allowed me to develop the firmware to compute the meantime of the input signals. This was a challenging task that required me to have a deep understanding of both hardware and software.

Finally, I tested the system and obtained good results. This involved running various tests on the system and analyzing the data to ensure that it was functioning correctly. Overall, my internship provided me with valuable hands-on experience in developing a trigger system, as well as a deeper understanding of digital electronics and programming.

For this amazing experience I would like to thank
my supervisor Zoltan and my co-supervisor Aidan
for the patience and the support throughout the internship.

Appendix

A.1 Deserializer

```
module deserializer (    input logic clk40,
                         input logic clk320a,
                         input logic clk320b,
                         input logic serialDataIn,
                         input logic serialDataIn1,

                         output logic [31:0] myOutput1
                    );

logic risingedge;
logic fallingedge;
logic risingedge1;
logic fallingedge1;
logic [31:0] myOutput;


initial begin
    myOutput <= 32'b0;
    myOutput1 <= 32'b0;
end

  IDDRE1 #(
      .DDR_CLK_EDGE("OPPOSITE_EDGE"), // IDDRE1 mode
      .IS_CB_INVERTED(1'b1),          // Optional inversion for CB
      .IS_C_INVERTED(1'b0)            // Optional inversion for C
   )
   IDDRE1_inst (
      .Q1(risingedge), // 1-bit output: Registered parallel output 1
      .Q2(fallingedge), // 1-bit output: Registered parallel output 2
      .C(clk320a),    // 1-bit input: High-speed clock
      .CB(clk320a), // 1-bit input: Inversion of High-speed clock C
      .D(serialDataIn),    // 1-bit input: Serial Data Input
      .R(1'b0)     // 1-bit input: Active-High Async Reset
   );

     IDDRE1 #(
      .DDR_CLK_EDGE("OPPOSITE_EDGE"), // IDDRE1 mode
      .IS_CB_INVERTED(1'b1),          // Optional inversion for CB
      .IS_C_INVERTED(1'b0)            // Optional inversion for C
   )
```

```
   IDDRE1_inst1 (
      .Q1(risingedge1), // 1-bit output: Registered parallel output 1
      .Q2(fallingedge1), // 1-bit output: Registered parallel output 2
      .C(clk320b),   // 1-bit input: High-speed clock
      .CB(clk320b), // 1-bit input: Inversion of High-speed clock C
      .D(serialDataIn1),   // 1-bit input: Serial Data Input
      .R(1'b0)     // 1-bit input: Active-High Async Reset
   );

logic clk40reg, clk320reg;
integer clk40phase;

initial begin
    clk40phase <= 1'b0;
    clk40reg <= 1'b0;
    clk320reg <= 1'b0;
end

always @(posedge clk320a) begin
    clk320reg <= clk40reg;
    if(clk40reg != clk320reg) begin
        clk40phase <= 1;
    end
    else
        clk40phase <= clk40phase + 1;
        if(clk40phase >= 7) begin
            clk40phase <= 0;
        end
end

always @(posedge clk40) begin
        clk40reg <= !clk40reg;
        myOutput1 <= myOutput;
end

always @(posedge clk320a) begin
    myOutput[4*clk40phase] <= risingedge;
end
always @(posedge clk320b) begin
    myOutput[4*clk40phase + 2] <= risingedge1;
end
always@(negedge clk320a) begin
        myOutput[4*clk40phase + 1] <= fallingedge;
end
always@(negedge clk320b) begin
        myOutput[4*clk40phase + 3] <= fallingedge1;
end
endmodule
```

A.2 Serializer

```
module serializer    (    input logic clk40,
                          input logic clk320,
                          input logic [31:0] myInput,

                          output logic serialDataObuf
                     );

logic fallingedge, risingedge;
logic clk40reg, clk320reg;
integer clk40phase;

initial begin
    clk40phase <=0;
    clk40reg <= 1'b0;
    clk320reg <= 1'b0;
    serialDataOut1 <= 1'b0;
    serialDataObuf <= 1'b0;
end

always @(posedge clk320) begin
    clk320reg <= clk40reg;
    if(clk40reg != clk320reg)
        clk40phase <= 1;
    else
        clk40phase <= clk40phase + 1;
        if(clk40phase >= 15) begin
            clk40phase <= 0;
        end
end

always @(posedge clk40) begin
        clk40reg <= !clk40reg;
end

always@(posedge clk320) begin
    risingedge <= myInput[2*clk40phase];
end

always@(negedge clk320) begin
     fallingedge <= myInput[2*clk40phase + 1];
end

ODDRE1 #(
       .IS_C_INVERTED(1'b0),              // Optional inversion for C
       .IS_D1_INVERTED(1'b0),             // Unsupported, do not use
       .IS_D2_INVERTED(1'b0),             // Unsupported, do not use
       .SIM_DEVICE("ULTRASCALE_PLUS"),    // Device version for simulation
       .SRVAL(1'b0)                       // Initializes the ODDRE1 Flip-Flops
   )
```

```
    ODDRE1_inst (
        .Q(serialDataOut1),    // 1-bit output: Data output to IOB
        .C(clk320),    // 1-bit input: High-speed clock input
        .D1(risingedge), // 1-bit input: Parallel data input 1
        .D2(fallingedge), // 1-bit input: Parallel data input 2
        .SR(1'b0)  // 1-bit input: Active-High Async Reset
        );

     OBUF OBUF_inst (
          .O(serialDataOut1), // 1-bit output: Buffer output
          .I(serialDataObuf)  // 1-bit input: Buffer input
        );
endmodule
```

A.3 Edge detector

```
module channel (input  logic dataIn,
                input  logic dataIn1,
                input  logic clk40,
                input  logic clk320a,
                input  logic clk320b,

                output logic hasEdge1,
                output logic [7:0] timeFromEdge1,
                output logic [31:0] waveForm1,
                output logic edgeToggle1
                                );


        //Deserialize the data in the differential buffer
        logic serialData;
        logic serialData1;

        IBUF IBUF_inst (
      .O(serialData), // 1-bit output: Buffer output
      .I(dataIn)  // 1-bit input: Buffer input
   );

        IBUF IBUF_inst1 (
      .O(serialData1), // 1-bit output: Buffer output
      .I(dataIn1)  // 1-bit input: Buffer input
   );
```

```
    logic [31:0] myInput;

    deserializer des(    .clk40(clk40),
                         .clk320a(clk320a),
                         .clk320b(clk320b),
                         .serialDataIn(serialData),
                         .serialDataIn1(serialData1),


                         .myOutput1(myInput)
                     );

        //Compute the phase
    integer haveSignal;

    initial begin
        haveSignal  <= 0;
                timeFromEdge1 <= 8'b0;
                edgeToggle1 <= 1'b0;
    end

        always @(posedge clk40) begin
                hasEdge1 <= 0;
                haveSignal <= 0;

        if(!haveSignal)begin
            for(int i=31; i>=0; i=i-1) begin
                if(myInput[i] == 1) begin
                    haveSignal <= 1;
                    timeFromEdge1 <= 31 - i;
                    hasEdge1 <= 1;
                    edgeToggle1 <= ~edgeToggle1;
                end
            end
                end
        else if(haveSignal < 4) begin
            haveSignal <= haveSignal + 1;
            timeFromEdge1 <= timeFromEdge1 + 32;
                        hasEdge1 <= 1;
        end
           waveForm1 <= myInput;
    end
endmodule
```

A.4 Meantime computation

```
module meantime (
                    input logic signalInput [3:0],
                    input logic signalInput1 [3:0],
                    input logic clk40,
                    input logic clk320a,
                    input logic clk320b,

                    output logic serialDataOut,
                    output logic led[3:0],
                    output logic ledOut
                    );

    initial begin
        serialDataOut <= 0;
        ledOut <= 1'b0;
    end

    logic hasEdge1 [3:0];
    logic [7:0] timeFromEdge1 [3:0];
    logic [31:0] waveForm1 [3:0];


    channel channels [3:0] (
                            .dataIn(signalInput),
                            .dataIn1(signalInput1),
                            .clk40(clk40),
                            .clk320a(clk320a),
                            .clk320b(clk320b),
                            .hasEdge1(hasEdge1),
                            .timeFromEdge1(timeFromEdge1),
                            .waveForm1(waveForm1),
                            .edgeToggle1(led)
                            );

    wire [31:0] sum;
    logic [7:0] meantime;
    logic [31:0] total;
    logic [31:0] signalOutput;
    logic [7:0] out;
    integer trigger;

    initial begin
        meantime <= 32'b0;
        total <= 32'b0;
        out <= 8'b0;
        signalOutput <= 32'b0;
        trigger <= 0;
    end

    assign sum = 32'b0 + timeFromEdge1[0] + timeFromEdge1[1] +
                 timeFromEdge1[2] + timeFromEdge1[3];
```

```
    always@(posedge clk40) begin

        meantime <= 32'b0;
        total <= 32'b0;
        trigger <= 0;
        signalOutput <= 32'b0;

        if(hasEdge1[0] && hasEdge1[1] && hasEdge1[2] && hasEdge1[3]) begin
            total <= sum;
            meantime <= {sum[9:2]};
            ledOut <= ~ledOut;
        end

        if(meantime > 95) begin
            out <= meantime - 96;
            trigger <= 3;
        end
        else begin
            meantime <= meantime + 32;
            out <= 8'b0;
        end

        if(trigger == 3) begin
            for(int i = 0; i <= out; i++) begin
                signalOutput[31 - i] <= 1;
                trigger <= trigger - 1;
            end
        end
        else if (trigger > 0) begin
            signalOutput = 32'b1;
            trigger <= trigger - 1;
        end

    end

    serializer ser (
                        .clk40(clk40),
                        .clk320(clk320a),
                        .myInput(signalOutput),

                        .serialDataOut1(serialDataOut)
                    );
endmodule
```

Bibliography

[1] Zoltan Gecse Aidan Grummer. System tests at fnal.

[2] Zoltan Gecse. Cosmic trigger system for cassette testing.

[3] Zoltan Gecse. Status and overview of the cms hgcal.

[4] Damien Thienpont. Hgcroc3-si datasheet.