Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Master's Thesis Nr. 510

Systems Group, Department of Computer Science, ETH Zurich

in collaboration with

CERN

A Very Cool Master Thesis

by

Paolo Rondot

Supervised by

Prof. Alonso Gustavo

April 2024–September 2024

Acknowledgements

Abstract

Contents

Chapter 1

Introduction

Chapter 2

Background

2.1 CERN

2.1.1 ATLAS Experiment

2.1.2 Liquid Argon Calorimeter

2.2 High Level Synthesis

Chapter 3

Latome Hardware Design

3.1 Specifications

3.1.1 The Liquid Argon Calorimeter Data Path

The LAr Calorimeter is made of 180,000 cells which are then summed up to form 34,048 supercells. The 12-bit readings of 8 supercells are serialized into one packet and sent through one optic fiber. The 116 LATOME boards are each connected to 48 optic fibers, thus receiving data from $48 \times 8 = 384$ supercells. These boards are responsible for transforming the ADC levels into energy levels, reorganizing the data and summing some energies for LATOMEs covering specific regions of the detector and distributing the data to the Feature Extraction (FEX) system. The FEX system is responsible for processing the data and sending it to the Level 1 Trigger system.

The FEX system is made of 3 groups: the Global Feature Extractor (gFEX), the Jet Feature Extractor (jFEX) and the Electron Feature Extractor (eFEX). Within these groups, gFEX only contains one board which gets summed data from all LATOMEs, thus showing the lowest granularity but with one board covering the whole detector. Then the jFEX is made of 6 boards, each receiving data from 19 LATOMEs, and finally the eFEX, with the highest granularity, is made of 24 boards, each receiving data from 4 LATOMEs.

3.1.2 The LATOME Specifications

The part of the LATOME firmware responsible for the conversion of ADC levels into energy levels is referred to as *User Code* in the LAr group. This

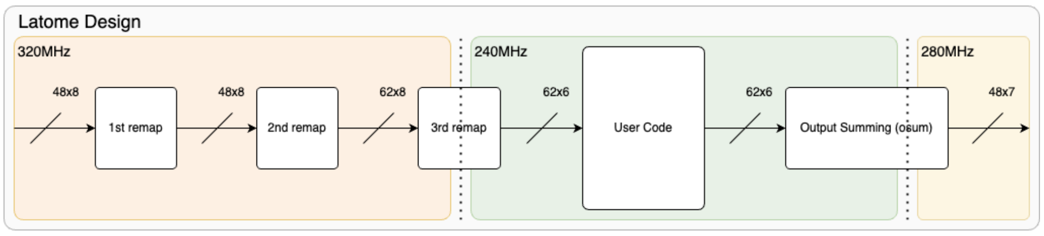

Figure 3.1: Original VHDL Design

code is not owned by the LATOME HLS team. However, the team is responsible for the different data organization steps, also called mapping or remapping, and the summing of energies for LATOMEs in specific regions of the detector. Additionally, since the output energies are only encoded on 10 bits, it is necessary to compress the data from 12 to 10 bits via a Multi Linear Encoder (MLE).

Initially, the whole LATOME firmware was written in VHDL. The implementation of all the functionalities took over 8 years and still had timing violations. Because in the next runs of the LHC, the amount of energy and collision is expected to increase, the LATOME design should be optimized to leave space for other functionalities, and sustain the increase in data.

3.2 Existing Designs

3.2.1 Original VHDL Design

The original design is characterized by more clock domain crossings than the current one.

Because bunch crossings (BC) happen at a frequency of 40MHz, it is necessary to run the different blocks of the design at multiples of this frequency. The figure3.1 shows three different frequencies being used. At the first remap stage, since the frames are made of 8 words, it is necessary to run the blocks at 320MHz. Then the frames become 6 words long, corresponding to a frequency of 240MHz. Finally, the FEX systems expect 7 32 bits words, thus the frequency is 280MHz.

These clock domain crossings made the original implementation very complex...

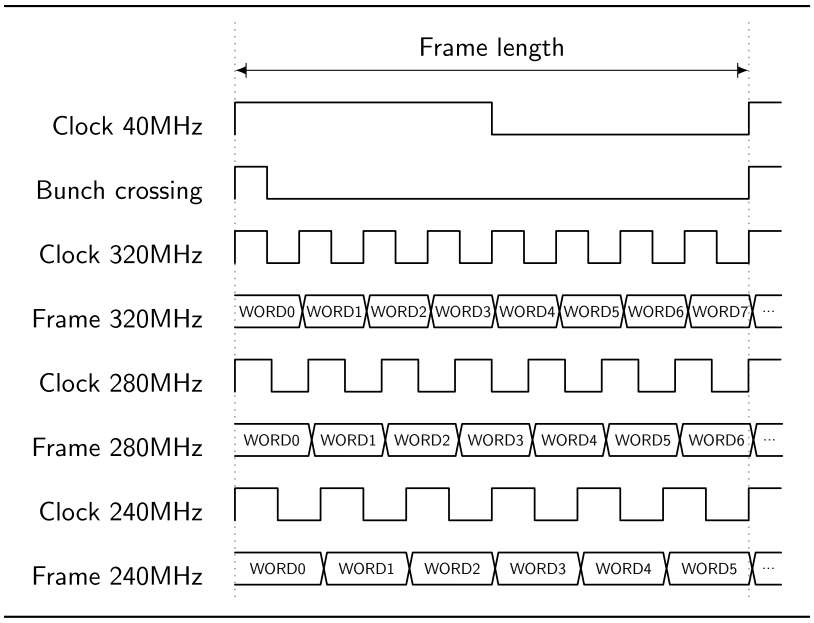

Figure 3.2: Different clocks used in the design

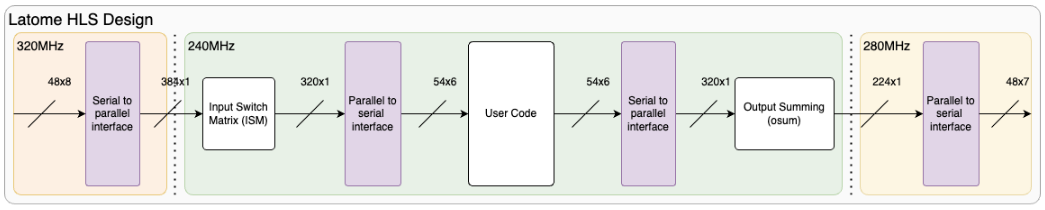
Latome HLS Design
320MHz
48x8
Serial to parallel interface
384x1
240MHz
Input Switch Matrix (ISM)
320x1
Parallel to serial interface
54x6
User Code
54x6
Serial to parallel interface
320x1
Output Summing (osum)
280MHz
224x1
Parallel to serial interface
48x7

Figure 3.3: Current HLS Design

3.2.2 Current HLS Design

When developing the new firmware from the specifications in HLS, very interesting insights from Dr. Marcos Oliveira emphasizing benefits of space division multiplexing were taken into account. By using parallel full frames, the frequency of the design blocks is not locked anymore depending on the frame size. This allows to reduce the logic dedicated only to synchronization and time-division multiplexing which can become very complex when there are dependencies between the words of the frame.

The figure 3.3 shows that all the remapping and processing blocks except for the User Code are being processed using parallel data. This is possible thanks to the serial-to-parallel and parallel-to-serial data transformation blocks. This approach drastically simplifies the design and showed very promising results in terms of area reduction, reduced latency.

3.3 Space vs Time Division Multiplexing

Space division multiplexing consists in making all the data available at the same time by using parallel inputs. This approach is more natural and makes the design simpler to describe as time dependencies are removed. For example when using serial data, i.e. time division multiplexing, if the word from time $t = 0$ of the frame needs to be available at the same time as the one of $t = 4$, the logic becomes very complex. This is not the case when using parallel data, since the whole frame is the input.

Additionally, in most cases, space division multiplexing yields a lower latency. Taking the example of the LATOME firmware, in the VHDL implementation which uses serial data, the remapping stages take more than 30 clock cycles to process the data. By using parallel data, a simple Input Switch Matrix could be implemented and processes the data in only 1 clock cycle.

In terms of area, space and time division multiplexing can yield very different results; depending on the situation, one might be more advantageous than the other. When the design is devoid of dependencies between the words, the resource sharing unlocked by time division multiplexing can be very advantageous at the price of a higher latency. On the other hand, since buses are wider with parallel data, routing can become convoluted with, for example, very large multiplexers.

Chapter 4

Leveraging Time Division Multiplexing

Chapter 5

Evaluation

Chapter 6

Conclusion