THE FLORIDA STATE UNIVERSITY

FAMU – FSU COLLEGE OF ENGINEERING

VHDL DESIGN AND FPLD IMPLEMENTATION FOR SILICON TRACK CLUSTER CARD

By

SHWETA LOLAGE

A Thesis submitted to the
Department of Electrical and Computer Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 2000.

Dedicated to my parents

ACKNOWLEDGEMENTS

First, I would like to thank my major professor, Dr. Reginald J. Perry for his guidance and support. I thank Dr. Simon Foo and Dr. Bruce Harvey for their guidance as members of my supervisory committee. I thank Fermi National Accelerator Laboratory, Department of Physics, Boston University and the Department of Physics, Florida State University for giving me the opportunity to work on the DØ project. I thank the National Science Foundation and the U.S. Department of Energy for funding the DØ project. I especially thank Dr. Horst Wahl at Department of Physics, FSU for helping me understand the intricacies of this project from the point of view of a physicist. I thank the Departments of Electrical and Computer Engineering and Physics at FSU for their financial support. Finally, I wish to thank my family and friends for their support during my tenure as a graduate student.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis describes the electronics for the STC, a part of the "Silicon Track Trigger", new trigger processor which is being designed for the D0 experiment at the Fermi National Accelerator Laboratory in Batavia, Illinois, Fermilab The silicon track trigger project is done in collaboration between the Electrical and Computer Engineering Department at FAMU-FSU and the Physics Departments of Florida State University, Boston University, Columbia University, and the University at Stony Brook.

The D0 detector is a general-purpose detector for the study of antiproton-proton collisions at high energy. The construction and operation of the detector is done by the D0 collaboration, which presently consists of about 450 physicists from about 50 universities and research laboratories. The particle created in the proton antiproton collisions generates signals in a silicon micro-strip detector, which can be used to reconstruct the tracks of the particles. The new trigger processor will use these signals from the new Silicon Micro-strip Tracker (SMT) to tag collisions in which long-lived b-quarks are produced. The study of events containing b-quarks can help in addressing many fundamental questions in particle physics. The new trigger processor will add significantly to the physics capabilities of the D0 detector in these areas. The silicon track cluster card (STC) accepts the digitized data from the strips in the SMT, finds clusters of strips with charge on them, determines the centroid for these clusters, and checks which of those centroids are within roads corresponding to candidate tracks.

Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) is used to describe the behaviour model of the design. The MAXPLUS –II synthesis tool by ALTERA Corporation was used to implement the design in FPLDs. The final design is implemented in three FPLDs of the FLEX10K family by ALTERA Corporation.

CHAPTER 1

INTRODUCTION

Integrated Circuit (IC) technology has dominated the electronic world since their introduction in 1960s. Dr. Jack S. Kilby was awarded a Nobel Prize this year (2000) for his part in the invention of IC. There were gradual advancements to the IC technology through Small Scale Integration (SSI), Medium Scale Integration (MSI), Large Scale Integration (LSI), Very Large Scale Integration (VLSI) technology that evolved in the 1970s and the most recent is Ultra Large Scale Integration (ULSI) technology. ULSI has made it possible to implement powerful and compact digital circuits at low cost, as now it is possible to build chips with millions of transistors [1]. New Computer Aided Design (CAD) tools are being used. Example, the Simulation Program for Integrated Circuit Emphasis (SPICE) is used at the circuit level, and there are Hardware Description Languages (HDLs) that are used to describe and specify electronic systems at different levels of abstraction ranging from behavioral to structural level.

Application Specific Integrated Circuits (ASICs) [2] are specialized type of ICs that have evolved from the VLSI technology. ASIC has evolved from a simple array of a few hundred logic gates into a complete family of full custom and semi custom ICs using more than 1 million logic gates. The main reasons for the popularity of ASICs are reduced board space requirements, reduced development cost, increased reliability, maximized performance, and security for new designs.

Full-custom ASICs are designed without using any precompiled or preprocessed silicon. The designer works at transistor level to optimize each cell for area and performance. They

generally require a complete set of standard steps for fabrication process. Whereas, semi-custom ASICs are preprocessed chips to which the designer only needs to add the final metal interconnection. The different types of semi-custom ASICs are Standard cell and Gate arrays.

Standard cells are pre-designed circuit functions at the LSI /VLSI level of complexity that can be joined by interconnecting cells. These are cheaper, when manufacturing more than 10,000 chips, as the Non-Recurring Engineering (NRE) costs are high. The NRE cost includes the cost of work done by the ASIC vendor and the cost of the masks. Gate arrays are preprocessed wafers of logic elements. They require only one to three masking steps of metal interconnects to complete the fabrication process. They have columns of transistor arrays surrounded by inputs and outputs. The drawback of gate arrays is the lack of flexibility to add complex functions; this is due to the difficulties in creating the signal routing channels.

Programmable devices are a type of semi-custom ASICs, which can have anyone of the architecture discussed above. These are general-purpose chips that can be configured for a wide variety of applications. The first of these kinds were the Programmable Read Only Memories (PROMs)[3], which were one-time programmable devices. The more recent versions are Programmable Logic Devices (PLDs), which have high speed and high performance logic gates. A step ahead in complexity to PLDs is the Field Programmable Gate Array (FPGA) [1]. There is very little difference between an FPGA and a PLD; an FPGA is usually larger and more complex than a PLD. A FPGA typically consists of a two-dimensional array of logic blocks that can be connected by general interconnection resources. There are a lot of FPGA companies in the market. The major competitors are ALTERA and Xilinx. Table 1.1 shows the comparison between the architecture, the technology and the main products of these companies.

Table 1.1 Comparison of the ALTERA and XILINX architecture and products. [4]

	ALTERA	Xilinx
Architecture	Deterministic Complex PLDs	Non-deterministic coarse grain FPGAs
Programming elements	EEPROM	Static RAM
High density family	APEX 20KE series	Virtex series
Low cost family	ACEX series	SPARTAN – II series
Memory elements	Embedded Array Blocks (EABs)	Block SelectRAM
Logic blocks	Logic array blocks (product – term – based programming logic devices)	Configurable logic blocks (Look- up Table approach)
Maximum number of gates available	1,520,640	1,000,000
Maximum RAM bits	442,368	131,072
System gates	2,392,000	1,124,022
Logic cells	51,840	27,648
Maximum I/O bits	808	512
Voltage Levels	1.8V, 2.5V, 3.3V	2.5V, 3.3V
Dual–port memory	Two ports are used, one for reading and one for writing, so need two-memory blocks (minimum).	Same port is used to read and write.
Special features	1.Content Addressable Memory (CAM). 2. Mega-functions to model memory.	1.On chip Digital Delay-Locked Loops (DLLs). 2.Block RAM can be supplemented for external memory.

As we can see from the Table 1.1 the number of logic devices handled is very large. This growing demand of ASICs and FPGAs in the electronic industry has lead to the popularity of Hardware Description Languages (HDLs). Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL)[5] has been the result of this high demand.

VHDL evolved in the US Department of Defense (DoD) in 1983. It was intended for

documenting and modeling digital systems ranging from small chip to large systems. DoD made it public in 1985, and IEEE immediately adopted it. It was a standard in 1987, under 1076-1987. It was further upgraded in 1993, with the IEEE 1076-1993 standard [6]. There are a lot many synthesis tools to help the designer check his design. The designer creates a behavioral or structural model of his design, which can be synthesized by a synthesis tool. Thus the design verification and testing process is made a lot easier and faster. The important aspect of VHDL is that the behavior of the circuit described is independent of the logic gates available. This makes the VHDL code independent of the technology [7]. Thus code written for one technology can be easily implemented into some other technology. For example the synthesis tool *SYNOPSIS* supports both Altera and Xilinx technology.

Some of the important applications of Field Programmable Logic Devices (FPLDs) are image enhancement filters, signal processing for digital modulation and demodulation, direct digital signal synthesis, fuzzy logic embedded controllers and reconfigurable computing [8]. Reconfigurable computing technology is one of the upcoming applications. It is the ability to modify a computer system's hardware architecture in real time. Instead of having ASIC, reconfigurable computing is an effort to build ICs that can be used for a set of applications after some minor reconfigurations [9]. Thus, parts of the algorithms are hardwired into the device and they are implemented on a function-by-function basis. Since these are implementations aimed at few applications, they offer tremendous acceleration over traditional programming solutions.

With such a wide variety of applications, FPLDs are easily available in market and this approach is found to be very economical too. The work presented in this thesis is one such application of FPLDs. The electronics design for D-zero (D0) detector at the Fermi National Acceleration Laboratory is to be used to trace the path of the particles emitted from the collision of a proton and anti-proton. This experiment has a large amount of data to be

processed and the available processing time is just few microseconds. It was been proven that hardware based algorithms outperform software implementations, even though the processors executing the software are much faster than the hardware [10]. Thus hardware implementation is chosen for this project. The hardware design is developed using VHDL as the description language and implemented in ALTERA's FLEX 10KE FPLDs. The synthesis tool used is ALTERA's MAXPLUS II. This approach gives us the flexibility of software and the speed of hardware.

In this thesis, Chapter 2 has a brief description about Fermi National Acceleration Laboratory, their activities and details about the DØ project. There is also a summary of the implementation of the main data path. Chapter 3 describes the design and implementation of the main data path in detail. The Chapter 4 includes the simulation results of the VHDL model and the comparison of the results with a MATLAB model of the design. Chapter 5 describes the different design approaches studied for some of the modules of the main data path. The concluding remarks about the work are in Chapter 6.

CHAPTER 2

THE DØ DETECTOR AT FERMI NATIONAL ACCELERATOR LABORATORY

Elementary Particle Physics, also called high-energy physics, is a branch of physics that tries to elucidate the structure and properties of matter at the smallest scale. The final aim is to describe matter in terms of a small number of different fundamental constituents, and to understand their interactions in terms of a small number of different forces. In order to probe the properties of matter, it is necessary to use projectile particles of high energy, and therefore experimental studies are done using high-energy accelerators [11]. Ordinary matter is made of atoms, which in turn contains electrons that orbit the nucleus, which is constituted of protons and neutrons. Particle physics aims to study the properties of these particles. The rapid progress in the understanding of particle physics during the last thirty years has brought about the emergence of a model according to which matter is made up of two kinds of basic constituents called "quarks" and "leptons". In this model, protons and neutrons are not "fundamental", since they contain quarks. Electrons, which belong to the family of leptons, however, are considered fundamental. The four fundamental forces viz. strong, electromagnetic, weak and gravitational interactions, by which these constituents interact with each other, have all been recognized to share several important characteristics. Two of these forces, electromagnetic and weak, are now known to be manifestations of a single force called *electroweak,* and also the strong interaction that holds nuclei together appears to be very similar to the electroweak interaction. This progress in understanding was achieved by an intensive mutual inspiration of theory and experiment, and was only possible

due to a huge, unprecedented experimental effort in terms of new accelerators and very large, "universal, all-purpose detectors", designed, built and operated by collaborations comprising several hundred physicists from institutions world-wide.

The theoretical model, which has emerged from these studies, is generally referred to as the "Standard Model" (SM) of particle physics. This theoretical description has been remarkably successful: Even though many sophisticated, high precision experiments have been performed to test it with the hope to find deviations from it, it has withstood all attempts to invalidate it. On the other hand, because of theoretical shortcomings in the model, we know that the SM is incomplete. It can only be a very good approximation of a more general theory, an "extension" of the Standard Model. Thus, theorists in particle physics look for extensions of the SM, which unify all forces in nature and improve the SM. The main experimental focus in present-day research in particle physics is to study particle collisions at the highest possible energies, to discover deviations or new phenomena not predicted by the SM.

Fermi National Accelerator Laboratory (Fermilab) was founded in 1967 [11] and has been in the forefront in the exploration of fundamental nature of matter. It was here that the first of the heavy quarks, the "bottom" or "beauty" quark, was discovered in 1977. Since 1989, its accelerator has allowed operation as a "proton - antiproton collider", in which protons and antiprotons are accelerated to high energies and made to undergo head-on collisions.

These collisions result in the production of many newly created particles, many of which decay very quickly. To detect these emitted particles, Fermilab has two detectors at the TeVatron collider – the DØ detector and the CDF detector. These experiments have helped to improve our understanding of the structure of the proton, and the way its constituents, the quarks, interact with each other. These experiments also discovered the top

quark in 1995 [12,13]. Presently, both the accelerator and the experiments are undergoing an upgrade that will extend the reach of studies. The SM predicts the existence of experimentally undiscovered particle, the "Higgs Boson", which, according to the present understanding is responsible for giving matter particles mass. The main aims of the future experiments at Fermilab are (i) to study the properties of the top quark in great detail and with high precision, (ii) to find evidence for the Higgs Boson and measure its properties example its mass, and (iii) to uncover evidence for "new physics" i.e. deviations from the SM. The phenomena of interest in these experiments are very rare; i.e. they occur in only a very small fraction of all the collisions.

The DØ detector

The DØ detector is one of the two "general purpose detectors" installed at the Fermilab TeVatron collider. As most of these detectors, it consists of a central tracking detector surrounding the collision point, a calorimeter, and a muon detector.

The *tracking detector* has two components: a silicon micro-strip detector and a scintillating fiber tracker.

The Silicon Micro-strip Tracker (SMT) consists of silicon wafers i.e. p-n junctions, which are reverse biased so as to form a diode with a depletion region over the full thickness of the wafer. The passage of charged particles through the depletion region causes the generation of electron-hole pairs. Thin aluminum electrodes called "strips" on these wafers collect the liberated charge and bring them to the SVX-II chip, which contains a 32-deep capacitor array to store the signal from 32 successive beam crossings, followed by an Analog-to-Digital Converter (ADC). Digitized data are transported via an optical fiber link to the counting house where trigger and read-out or data acquisition system is located.

The aluminum strips for read out are oriented in three different positions axial, stereo and z-axis. The axial position means parallel to the axis of cylinder of the detector, the

stereo position means inclined by about two degrees with respect to the axial direction and the z-axis position means the chips are perpendicular to the axial direction.

The Central Fiber Tracker (CFT) consists of eight concentric cylinders surrounding the beam pipe; each of the 8 cylinders has several layers of scintillating fibers. Passage of charged particles through those fibers causes scintillation light to be generated, which is detected by Visible Light Photon Counters (VLPCs). The signal is split so that part of it is available at trigger Level_1 (described later).

The other parts of the DØ detector are: the *calorimeter,* which measures the energy of both neutral and charged particles, and the *muon detector,* which detects all those charged particle which are not absorbed in the calorimeter.

The DØ trigger and data acquisition system

Since the phenomena that evoke the most interest are expected to be rare, high beam intensities are needed to improve the chance of detecting them. At the high beam intensities in the new upgraded Fermilab collider, the rate of interaction between the protons and antiprotons that collide is very high; at the anticipated intensities, we expect about 7 million collisions per second. Although, most of these interactions are of no interest to the physics program of the experiment, some rare events are most likely to provide new and interesting information. For example, only one in 10 billion interactions will produce a top quark. Since the experiment can only record about 20 events per second, one would miss the interesting events if the experiment did not have a way to decide very fast which ones to record. This decision is done by a system of fast electronics called the "**trigger**". The trigger is arranged in three successive stages called Level_1, Level_2, and Level_3 [15]. The amount and quality of information, as well as the time available to make a decision increases from level to level.

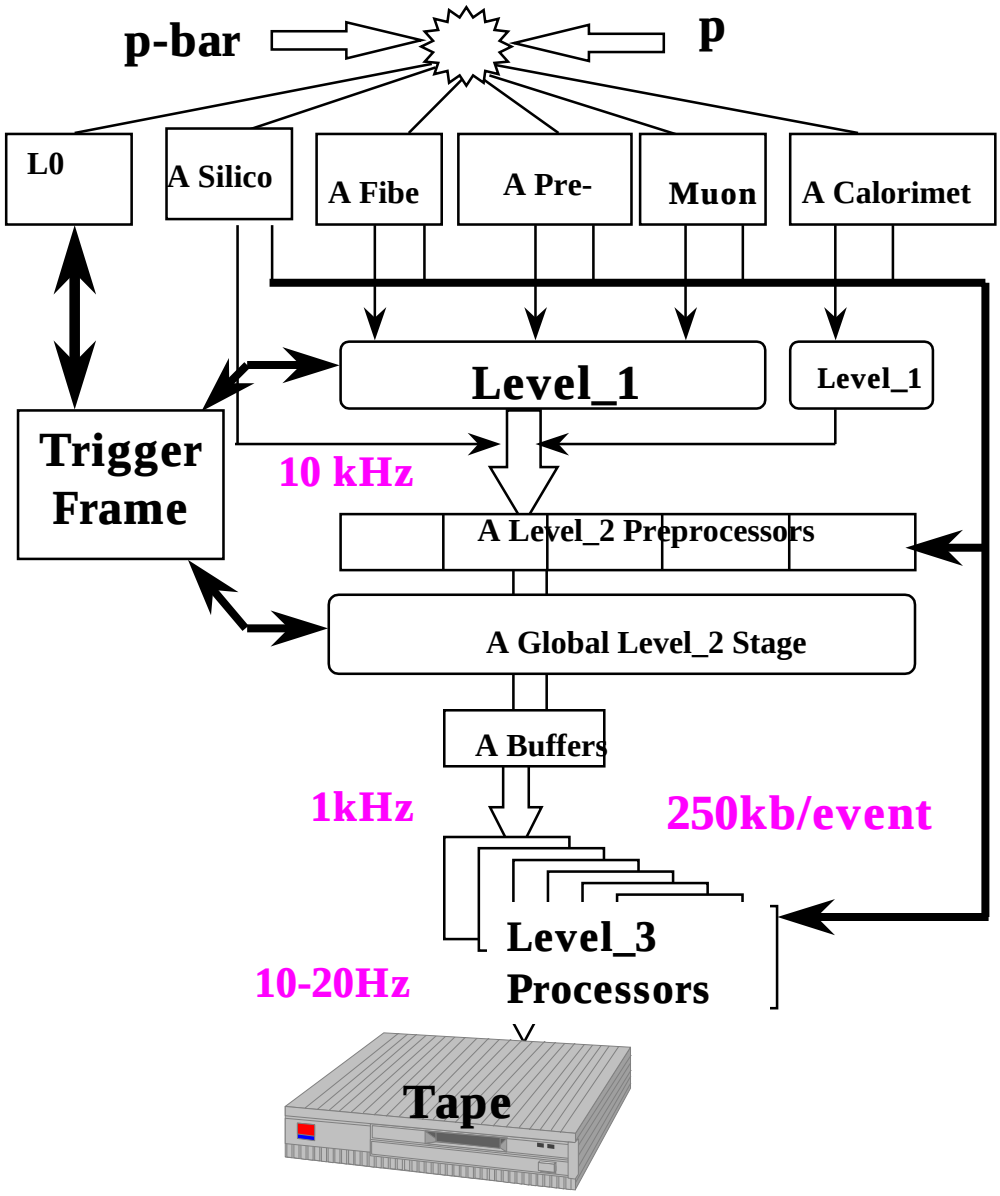


Figure 2.1 The flow of data in the DØ trigger and data acquisition system

Level_1

This level obtains preliminary information from fast "trigger pick-offs", separately from the calorimeter, fiber tracker, muon detector and the pre-shower detectors [15]. It then makes a fast decision on whether there are candidates for potentially interesting *objects*, for

example, an electron-like energy cluster in the calorimeter with energy above a given threshold, a track candidate in the central fiber tracker with momentum above a given threshold or a muon track candidate in the muon detector. Up to 128 such conditions or and-or combinations thereof can be examined. If any of these 128 conditions (Level_1 bits) is satisfied, Level_1 issues a Level_1-accept and sends the event to Level_2 for the next stage of decision-making. The input rate to Level_1 is about 7 million events per second, and the output rate is 10000 events per second.

One part of the Level_1 trigger which is of interest for this work is the *Central Track Trigger* (CTT). This compares the pattern of hits in the fibers of the central fiber tracker with preloaded patterns in a Look Up Table (LUT), which correspond to charged particle tracks with momentum in a given range. This decision is done using Xilinx Virtex FPGAs.

Level_2

The Level_2 trigger receives an input rate of 10000 events per second, i.e. it has, on the average, 100 μs to make a decision. It has access to more and more refined information than Level_1. In particular, the information from the SMT is available at Level_2. On issue of a Level_1 accept, the data from the silicon detector chips are digitized and passed on to Level_2. The digitization takes place in a full custom, mixed signal integrated circuit (SVX-II). These are hardwired directly to the detector. The charge, which is stored in terms of voltage across a capacitor, is digitized by the SVX-II to an 8 – bit word. Ten SVX-II chips are connected to High Density Interconnect (HDI) that is copper flexible printed circuit. Four HDIs are further connected to each of the two Port Cards of the "Sequencer", up to 16 of which are mounted on a VERSAmodule Eurocard (VME) crate.

The Level_2 trigger [16] is organized in two sub-stages: Level_2 preprocessors, one

for each detector, and a global Level_2 processor. The task of the preprocessors is to determine quantities which can then be used as the basis for a decision on whether to accept an event or not. The global Level_2 processor combines information from all the preprocessors and makes a decision based on this combined information. The Level_2 processor consists of Alpha processors, while the preprocessors contain a variety of boards with FPGAs, Digital Signal Processing (DSP) microprocessors and Alpha processors. When Level_2 issues an accept signal, the data is transferred to VME Receiver Card (VRC) on fiber using low-level fiber channel hardware [15], for transfer to Level_3.

One of the Level_2 preprocessors is the L2STT, the silicon track trigger processor, a part of which is the subject of this thesis.

Level_3

This is the final decision and data acquisition level in the D0 electronics. On every Level_2 accept, the Level_3 receives data from each module of the both Level_1 and Level_2, as well as directly from the various detector parts. The data corresponding to an event that has been accepted by Level_2 are sent to one of the fifty Pentium processors that make up the Level_3 system. A decision algorithm implemented in software on these processors examines the full set of information that is available and makes a decision on whether to accept or reject the event. The input rate to Level_3 is 1000 events per second, and the output or accept rate is about 20 events per second. The accepted events are then written to disk and transferred over a fast link to the Feynman Computing Center where they are recorded on permanent mass storage devices.

The Silicon Track Trigger (STT)

The STT design in the DØ detector electronics is motivated by the fact that many interesting phenomena are characterized by the presence of b-quarks. For example, top quarks decay into b-quarks. In addition the Higgs Boson, the most sought-after particle, is expected to decay into b-pairs with very high probability. The b-quarks decay after a lifetime which is long enough for them to travel a distance of about a millimeter before decaying. Thus, tracks of particles from the decay of these b-quarks appear to originate from a different point called a secondary or displaced vertex rather than from the primary interaction point called the primary vertex, which is the point where the proton and antiproton collided. Therefore, the presence of tracks, which come from a displaced vertex is a signal of a rare event, and is recorded.

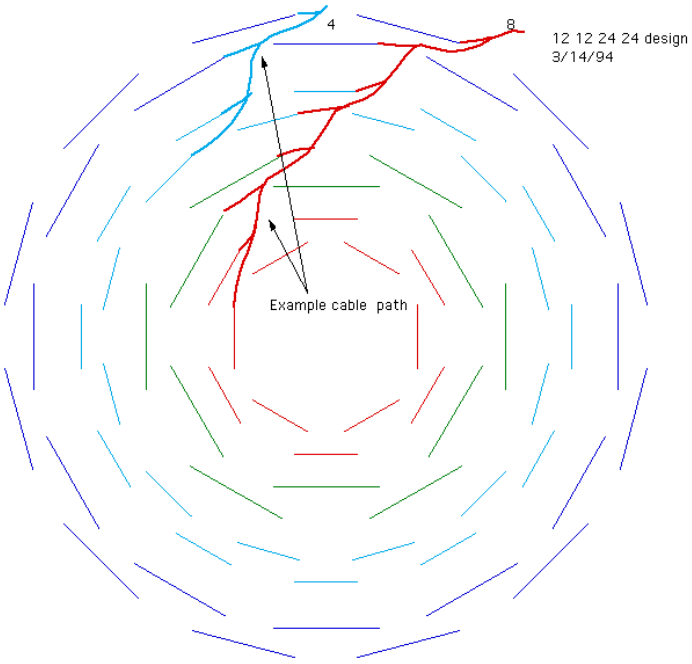


Figure 2.2 The layout of the DØ Silicon strip detector

The task of the STT is to find and reconstruct charged particle tracks in the SMT and calculate their properties such as charge, momentum, and impact parameter (i.e. distance of closest approach to the primary interaction point).

The STT is organized into three stages: the Fiber Road Card (FRC), the Silicon Track Clustering Card (STC), and the Track Fit Card (TFC). The FRC receives the information about track candidates from the Level_1 CTT, and sends this information on to the STC and the TFC. The STC receives the SMT data, preprocesses these data, finds clusters and calculates their centroids, associates clusters with track candidates that it received from the FRC, and sends all of this information on to the TFC, as well as to Level_3. Finally, sophisticated DSP microprocessors in the TFC are used to run curve-fitting algorithms to determine the track parameters such as momentum charge, and impact parameter, which are then sent to the global Level_2 processor for decision-making.

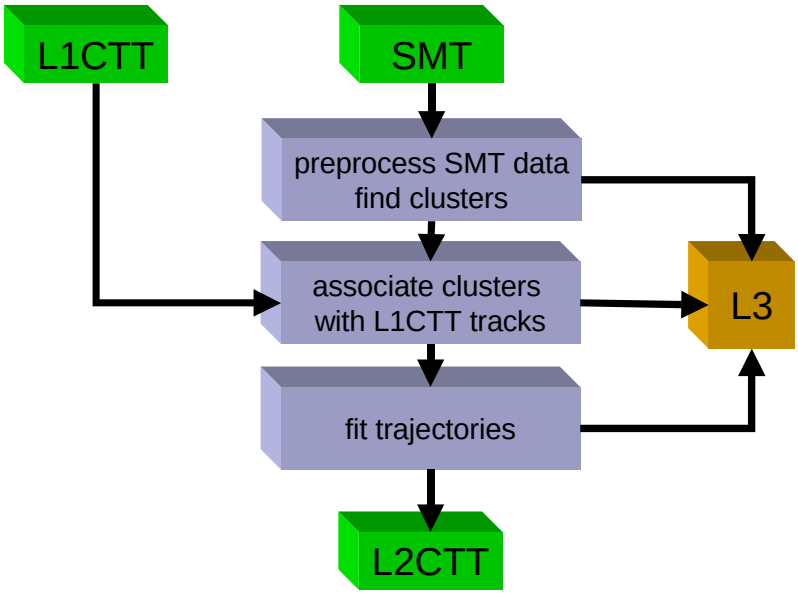


Figure 2.3 Block diagram of the STT

The silicon track cluster card (STC) [16]

Every STC card has eight identical channels of STC electronics, corresponding to different parts of the SMT detector. Thus, STC refers to one of these channels (of which

there is a total of 54 in the whole of STT). The development of the VHDL code for the STC is the subject of this thesis. The STC performs the functions of preprocessing the SMT data, finding clusters, to associate the centroids with the FRC tracks and fitting the trajectories to find the particle tracks. It acquires the SMT data and gives the final output to the track-fitting algorithm, which is the final step in the STC electronics. The main data path has three modules: Strip Reader, Centroid Finder, and the Hit Filter. The STC level also has storage module buffers for L3.

Strip Reader

This is the front end of the STC data path. It has two sub-modules, SMT Data Filter and the Strip Reader Control. It accepts the 8-bit SMT data from the VME bus at the rate of 53MHz. This data is filtered of excess "C0", end of event markers. The filtered data is corrected after checking for bad strips and a second check for the gain and offset values for the individual strips. At the output the Strip Reader formats the data obtained in a 23-bit word to be read by the Centroid Finder. The data is written into a 23-bit wide First In First Out (FIFO) bank of registers at the rate of 32MHz. The format of the 23-bit register is given in Table 2.1 below.

Table 2.1 The 23-bit data at the output of the Strip Reader.

22..21	20	19	18..11	10..7	6..0
Data-type	New data Bit	End of event bit	Data	SVX-II Chip ID	Strip number

Centroid Finder

This module also runs at 32 MHz. It has two tasks (i) to determine a cluster and (ii) to find the centroid for the cluster. The two sub-modules in this module are Cluster Finder and the Centroid Calculator. The module has the ability to find three– or five–strip clusters, selected by a downloadable parameter. The Centroid Calculator algorithm finds the centroid, i.e. the pulse-height weighted average of the strip-addresses. These centroids (11 bits, containing SVX-II chip id (4) and strip number (7)), give the position at which a particle is supposed to have passed. The data type of the centroid specifies which strip direction (axial, stereo or z-axis) is represented by the data. Hence this is tagged along with centroid, when it is passed on to the Hit Filter. The centroids tagged with the end of event bit, data type, and the pulse area are stored in the FIFO at the end of the module in the format given in Table 2.2.

Table 2.2 The 17-bit data word from Centroid Finder to the Hit Filter.

17	16..15	14..13	12..0
End of event bit	Data type	Pulse area	Centroid

Hit Filter

The Hit Filter has 46 parallel Comparators that can hold a maximum of 46 pairs of the road groups from Level_1. The two road values represent the upper limit and the lower limit of the road groups. The centroid values are pulled out of the FIFO. Only the centroids with axial data-type are compared with the roads. The z-axis centroids are stored in this module to be read out by the hit interface module. The hits are the centroids that match the road groups. These are also stored in a FIFO. The hit interface module reads them out.

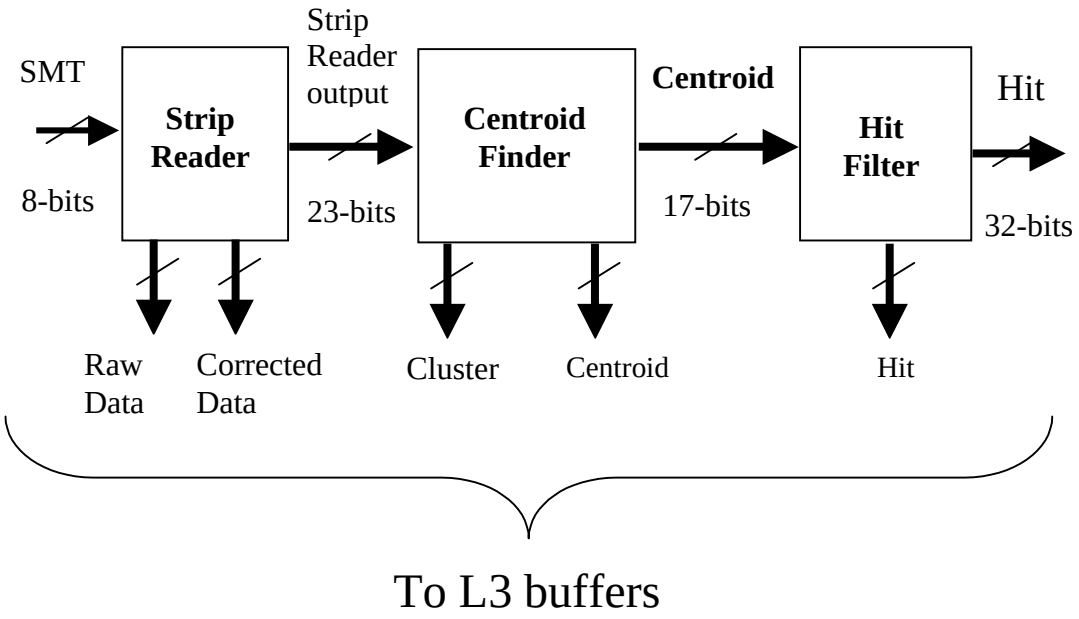


Figure 2.4 The data flow in the STC main data path with reference to the modules in the electronics.

L3 buffer

The L3 buffer is a group of five buffers that store the data processed at each step in the STC. This data can be accessed for further analysis after the results from the Level_3 are obtained.

The five buffers are –

1. Raw data: This buffer holds the filtered data coming out of the SMT Data Filter.
2. Corrected data: The corrected data is the raw data checked for bad strips and also the data from the strips is processed for gain correction and offset correction.
3. Strip clusters: This buffer holds the data values and addresses of the strips that form the cluster, the threshold values used and the cluster type.
4. Centroids: The centroids calculated in the Centroid Calculator, with their data type are stored in this buffer.
5. Hits: This buffer holds the hits out of the Hit Filter.

Main control module

The main control module is the control unit that monitors the flow of data to and from the eight STC channels. This module is the gateway between the eight channels and the other electronics of the Level_2. The reading of hits and z- axis centroids from the Hit Filter, downloading the parameters in the channel memory and reading out the L3 buffers are some of the functions of this module. The hit interface module of each channel talks to this main control module.

With this background of the DØ detector and the role of the main data path within the STT, the further chapters discuss the design methodologies and the implementation of the main data path.

CHAPTER 3

A DETAILED DESCRIPTION OF THE STC MAIN DATA PATH

The main data path consists of three modules: the Strip Reader, Centroid Finder and the Hit Filter. The data flow through each of the module and the data processing in the module is described in detail in this chapter.

Design features

The design is implemented in a FPLD and VHDL is used to describe the design behaviour. The design defines the logic for one channel of the STC. Each STC has eight such identical channels operating in parallel. Each channel can be enabled or disabled according to incoming data from the experiment. The whole design (except for the SMT data filter) is designed to operate with system clock of frequency 32 MHz. The logic has a synchronous reset signal at startup that is used to initialize the design. All the modules are in their initial states and do not start the data processing till an EVENT_START signal is issued. The SMT Data Filter in the Strip Reader runs synchronously when the channel is enabled, and the Centroid Calculator in the Centroid Finder and the Comparator module in the Hit Filter run asynchronously when the channel is enabled.

All the control modules are designed with the Moore type Finite State Machine (FSM) approach. A FSM is a state machine with finite number of states. The machine always resets into an initial state and updates states on each clock cycle. FIFO buffers are provided at the end of each module in the data path to maintain a synchronous data flow. The logic can process data for only one event, but the design can hold more than one event at the input in the SMT data, and output in the Hit Filter.

Design parameters

The parameters required for the data processing are unique for each channel. These parameters are downloaded in the channel memory on start up. The memory spaces are allocated to facilitate the main control logic to write into the memory and read out of the memory through a bi-directional bus. The control logic uses 15 address lines to access this memory space. The memory allocated area are as given in the Table 3.1.

Table 3.1 Memory mapping for the single channel.

Memory area	Memory space	Memory address
Monitor space	1K X 32	0000 – 03FF
Miscellaneous	1K X 32	0400 – 07FF
Gain Offset LUT	4K X 8	0800 – 17FF
Test data LUT	256 X 18 (1K min.)	1800 – 1BFF
Empty Space (for future use)	–	1C00 – 3FFF
Road data LUT	16 K X 22	4000 – 7FFF

Monitor space

This space holds the monitoring counters from the Strip Reader and the Centroid Finder. These counters are defined as[17] –

1. SMT counters
    i. Mismatch Counter: This counter counts the number of times there was a mismatch in SEQ ID and HDI ID.
    ii. SMT error (SERR) Counter: This counter keeps track of the error in reading the data VTM data.
    iii. Zero Error Counter: This counter increments every time a byte of zeros is not present in the data stream after the SVX-II chip-id.

2. Chip activity counters: There are nine SVX-II chip activity counters, one for each SVX-II chip. The counter for the SVX-II chip is incremented when a strip from that SVX-II chip has data on it. Thus it is an indication of the activity on that SVX-II chip for an event.
3. Cluster counters: There are three cluster counters, one for each data type. These counters count the number of clusters of each data type in an event.

Miscellaneous memory

The miscellaneous memory constitutes of bad channel memory, chip ranges, pulse area thresholds, clustering thresholds and the miscellaneous data register.

1. Bad channel memory space having 64, 32 bit wide words is actually a LUT. Addresses assigned to this memory space are from 0400 – 047F (HEX). This memory has the status of each of the 128 channels of every SVX-II chip in the detector. A channel is set to be bad on the basis of the technical survey of the detector, conducted in between runs. This status is used to eliminate any false readings on the channels.
2. Chip range memory space is at address 0480 (HEX). This memory consists of 24-bit word. Each data type has a 4-bit upper range value and a 4-bit lower value for the SVX-II chips, thus forming the 24-bit wide word.
3. Pulse area threshold memory space is at the address locations 0500-0503 (HEX). This memory is 24-bit wide and holds the three threshold values Pulse_Threshold_1, Pulse_Threshold_2, and Pulse_Threshold_3 required to calculate the pulse area for the clusters found in the event. These values are unique for each data type.

4. Clustering thresholds are downloaded in memory locations 0600-0603 (HEX). These are the Threshold_1 and the Threshold_ 2 values used in the clustering algorithm. The data value Threshold_1 is minimum data value that should be on a strip to be considered as valid data. The data value threshold_2 is the minimum peak data value of a cluster to get a valid cluster from the strips.
5. Miscellaneous data register is a 32-bit register downloaded at address location 0580(HEX). This register has the following parameters -
    i. Unique 8–bit SEQ ID for the channel.
    ii. Unique 3–bit HDI ID for the channel.
    iii. Delay: This is an 8-bit signal to indicate the delay between the FRC_START signal and the main EVENT_START signal.
    iv. Disable bit: This bit indicates whether the channel is disabled.
    v. SMT ID: This is 3-bit number, unique for each SMT data stream.

Table 3.2 Miscellaneous register downloaded at 0580 (HEX).

31..28	27	26..24	23..16	15..8	7..0
Empty	Disable Bit	SMT ID	Delay Count	SEQ ID	HDI ID

Gain offset memory

This memory holds the corrected data for each SVX-II chip in the detector, according to the gain and offset values for the SVX-II chips. This data is accessed with the SVX-II chip number and the data value for the strip.

Corrected data is calculated using following equation –

Corrected data = gain * actual data value + offset.

Test data LUT

The test data memory space holds the test data in the same format as the 18-bit data stream (refer Table 3.3). This data stream is used to check the functionality of the design.

Road data LUT

The road data memory is an external memory space, which gives a unique pair of the roads (upper and lower) 11-bits wide defining a group of roads. The upper and lower road values are unique for each 17-bit road data value obtained from the Level_1 i.e. FRC, which represents a fiber road track.

Strip Reader

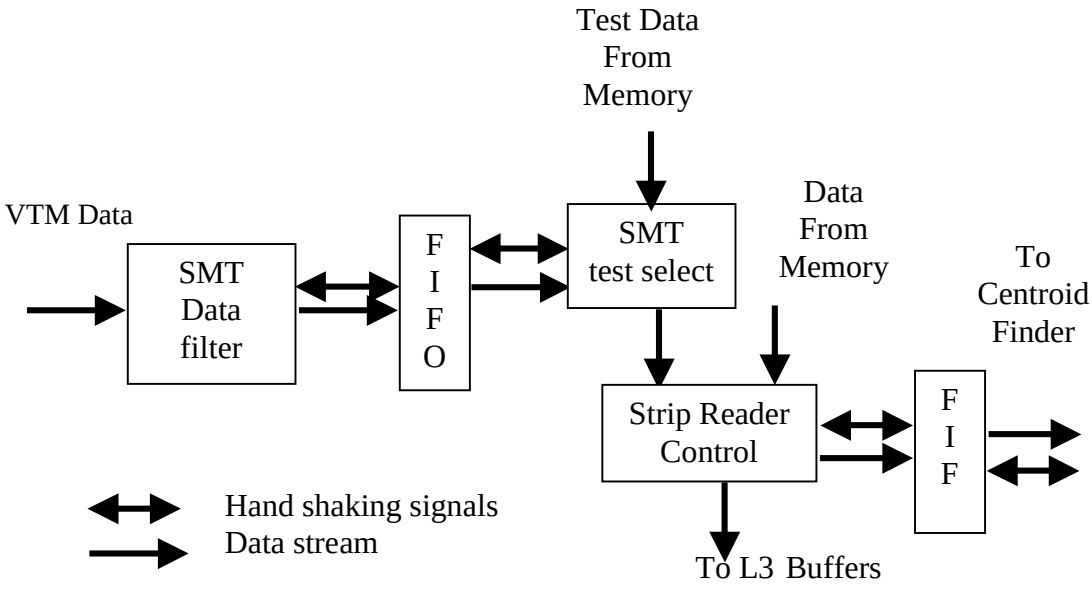


Figure 3.1 Block diagram of the Strip Reader module.

The three sub modules in the Strip Reader are the SMT Data Filter, SMT Test Select and the Strip Reader Control. The Strip Reader module has two clocks; a SMT clock running at 53 MHz, to match the speed of the SMT data coming on the VME bus and the PCI System clock running at 32 MHz, this clock determines the flow of data through the design.

SMT Data Filter module

The SMT module gets input from the VME bus. The module has a state machine that filters out the excess data at the end of event. It also converts the 8-bit VTM data stream into 16-bit data word so that the system clock even though slower than the SMT data clock can match the speed of the input data stream. The SMT Data Filter runs continuously when the channel is enabled. Any error in reading the input data stream is indicated by setting one of two error bits, one each for the higher and the lower byte of the output stream. These errors bits are also passed on to the next sub module along with the 16 bit data, thus forming an 18-bit word that is stored in the FIFO, as given in Table 3.3.

Table 3.3 Data stream from SMT Data Filter to the Strip Reader Control.

17..16	15..8	7..0
Error bits	Higher byte	Lower byte

Another task of this module is to determine the event number of the current data stream. This number is assigned after the FRC_START signal is received. This signal is used to synchronize the SMT data event and the road data event (from Level_1). The event number is tagged along with the end of event marker and written as the last word in the FIFO.

SMT Test Select module

This module uses the TEST input from the main control module to select between the test data stream downloaded in Test FIFO and the SMT data stream in the SMT FIFO. The hand shaking signals from the Strip Reader Control module to both the FIFOs are routed through this module.

Strip Reader Control module

The Strip Reader using a Moore type FSM (Reference flow chart in Appendix A.3); reads the 18-bit data stream from the intermediate FIFO (refer Table 3.3). Each channel is configured to read the data stream from a unique SEQ and HDI. Thus the first word read out of the FIFO is checked for the correct SEQ ID and the HDI ID; if there is a mismatch, the mismatch counter is incremented and mismatch bit (MM) is set. The SVX-II chip-id is identified next with a byte of zeros following it. These are important to isolate the strips with data values from the SVX-II chips. If the byte of zero is absent the zero error counter is incremented. The data type is determined by comparing the SVX-II chip-id with available SVX-II chip ranges for the three data types. Chip activity counters are present in this module to keep track of the number of strips per SVX-II chip. Once this is done, the design starts processing the strip numbers and the data values. The gain-offset memory is then accessed using address as the SVX-II chip id concatenated with the data value to get the corrected data. Before formatting the 23-bit data word (refer Table 2.1) the strip number is compared with the bad channel memory and the strips that are set to be bad are assigned a zero data value. The corrected data and the zeroed bad channel strip with the data type, SVX-II chip-id and channel number are formatted into the 23-bit wide data word and written in the output

FIFO (refer Table 2.1). An event covers many SVX-II chips but the same SEQ ID and the HDI ID. When an end of event marker is encountered, the state machine extracts the event number, which is the lower byte of the 18-bit word. The two error bits (refer Table 3.3) serve as an input to the SERR counter and are passed on as a single SERR bit.

Centroid Finder

The Centroid Finder module works at the frequency of system clock. This module has two sub-modules the Cluster Finder and the Centroid Calculator. This is the heart of the data processing in the main data path. This module finds the clusters from the strips and calculates the centroids. The detailed block diagram of this module is shown in Figure 3.2.

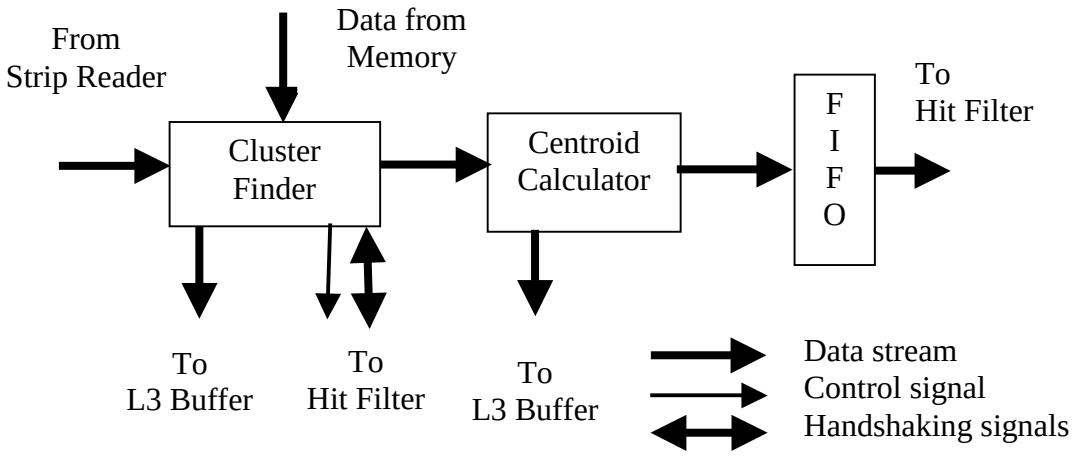


Figure 3.2-Detailed block of the Centroid Finder module.

Cluster Finder module

The Cluster Finder module has three tasks (i) interface with the output FIFO of the Strip Reader module, (ii) find the data clusters and (iii) pass on the strips of cluster to the

Centroid Calculator module. It reads out the 23-bit word and splits it into its constituents (refer Table 2.1). The strips in a cluster should have a data value greater than or equal to Threshold_1, same data type and sequential addresses. The SVX-II chip id is concatenated with the strip number to form an 11-bit strip address. The clusters are of five strips. There are five data and address buffers to store the data values and addresses of strips constituting a cluster. There are also two secondary data buffers and address buffers to store the shadow values in anticipation of a peak value greater than the one already found for a cluster. The first data value read is always stored in data buffer D3, which contains the peak value. The following data values after conferring to the above conditions are compared with data value in D3. If the new data value is greater than or equal to D3 then the peak is replaced or the following data buffers are filled, (reference flow chart Appendix A.4). The final peak cluster value should be greater than or equal to Threshold_2.

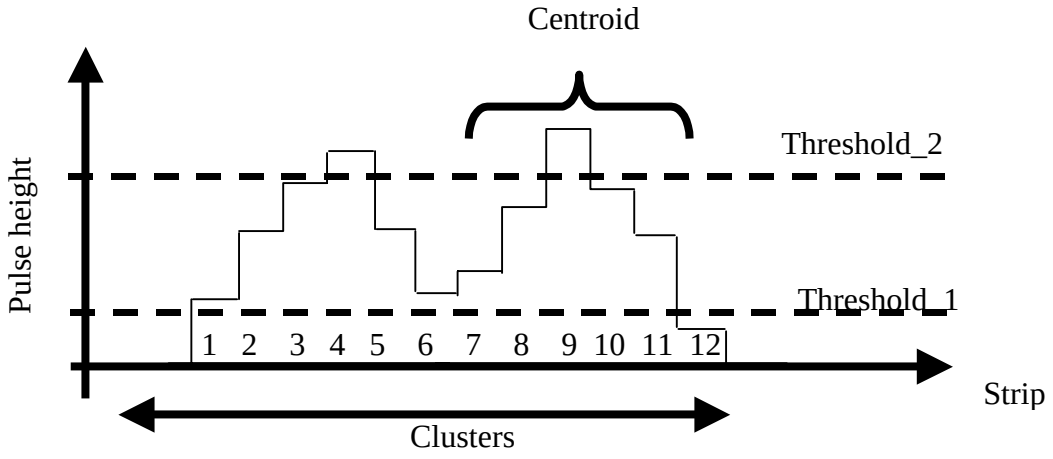


Figure 3.3 An illustration example of a five-strip cluster.

The example in Figure 3.3 is an illustration of a five-strip cluster. All the registers are initialized to zero before reading data for a new cluster.

(i) Strip 2 loaded into D1, address in add 1 = (peak address – 2)

(ii) Strip 3 loaded into D2, address in add 2 = (peak address – 1)

(iii) Strip 4 loaded into D3, peak data value and address in add3 = (peak address)

(iv) Strip 5 loaded into D4, address in add 4 = (peak address + 1)

(v) Strip 6 loaded into D5, address in add 5 = (peak address + 2)

The strip number 7 and 8 are stored in the shadow buffers B1 and B2. When the data value of strip number 9 is compared with D3 and found to be greater than data value of D3 the data buffers D1 and D2 are over written with the shadow buffers B1 and B2. The corresponding address buffers are also replaced. The data value of strip number 9 is written in to data buffer D3 and the strip address is written into add3 (i.e. the peak address).

So the new data buffers are –

(i) Strip 7 loaded into D1, address in add 1 = (peak address – 2)

(ii) Strip 8 loaded into D2, address in add 2 = (peak address – 1)

(iii) Strip 9 loaded into D3, peak data value and address in add3 = (peak address)

(iv) Strip 10 loaded into D4, address in add 4 = (peak address + 1)

(v) Strip 11 loaded into D5, address in add5 = (peak address + 2)

The end of cluster is found at Strip 11, as data value of the Strip 12 is below Threshold_1. The number of strips used for the cluster is 5 and the number of strips checked is 11. The data values (8-bits each) and the address add2 (11-bits) are passed on to the Centroid Calculator.

Centroid Calculator module

This module takes the data values of the strips in a cluster from the Cluster Finder and performs the arithmetic calculation of finding the centroid using the centroid mass principle. The calculation for the three or five strip cluster is decided according to the cluster type bit.

In Figure 3.4 below, if D2 is viewed as the origin and D1, D3, D4 and D5 as point masses, the centroid of the system for a five-strip cluster is [18]–

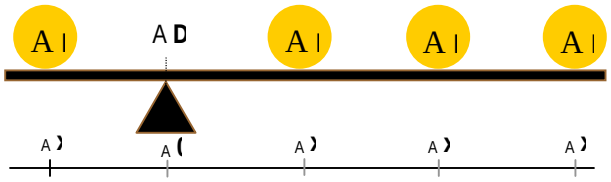


Figure 3.4 Realization of centroid calculation for a five-strip cluster [18].

$$centroid\ \ (\ five\ -\ strip\ ) = \frac{\sum_{i=-1}^{3} i.D_{i+3}}{\sum_{i=1}^{5} D_i} = \frac{\text{- D1 + D3 + 2D4 + 3D5}}{\text{D1 + D2 +D3 + D4 + D5}}$$

$$centroid\ \ (three\ -\ strip\ ) = \frac{\sum_{i=-1}^{1} i.D_{i+2}}{\sum_{i=1}^{3} D_i} = \frac{\text{- D2 + D4}}{\text{D2 +D3 + D4}}$$

The centroid value obtained from these calculations is added to the Address add2 passed on from the Cluster Finder to get the exact address of the centroid for the cluster. The

final centroid value is 13-bit wide, because of the addition of two precision bits from the calculation.

This module also finds the quantized pulse area of the cluster. The pulse area is calculated by summing the data values for all the strips constituting the cluster and comparing the sum to three threshold values stored in the channel memory [19]. Two bits are set to indicate the cluster pulse area according to the thresholds given in Table 3.4.

Table 3.4 The scheme for determining the pulse area of the cluster [19].

Pulse Area	Sum
00	< Pulse_Threshold_1
01	≥Pulse_Threshold_1, Pulse_Threshold_2 ≤
10	≥ Pulse_Threshold_2, Pulse_Threshold_3 ≤
11	≥ Pulse_Threshold_3

The centroid value its data type and pulse area are written into the output FIFO of the Centroid Finder (refer Table 2.2).

Hit Filter

The Hit Filter module handles the axial and the z-axis centroids. It stores the z-axis centroid in the z-centroid FIFO and compares the axial centroids with road data values to find the hits (refer Chapter 2 – Section “Hit Filter”).

The Hit Filter module has six sub modules. They are Hit Filter Control module, Comparator module, Hit Register module, Hit Format module, Hit Readout module, and Z-centroid module. The module handles only the axial and z-axis type of centroids. The z-axis centroids are stored in a buffer, which can be accessed by the hit interface module and the axial type of centroids are compared with the roads from Level_1 to find hits. The hits are

written in the output FIFO from where they can be pulled out by the hit interface module to be passed on to the STC Main control module.

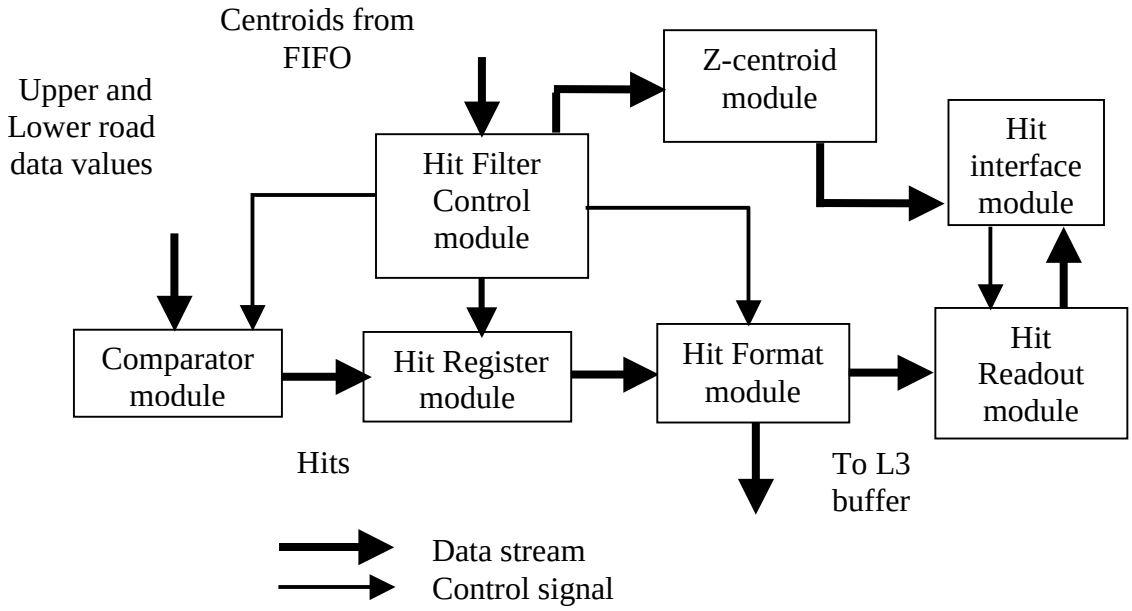


Figure 3.5 Detailed block diagram of the Hit Filter module.

Hit Filter Control module

This module controls the processing in the Hit Filter. It is activated with an EVENT_START signal. The roads from the road data memory are loaded into the comparators sequentially on every road write signal. Since the roads are loaded sequentially, each comparator corresponds to the track of Level_1. A masking register is created once all the roads are loaded into the comparators. This register masks all the comparator outputs that are not loaded. The module then reads one centroid at a time. The axial centroids are loaded into the comparators, while the z-axis centroids are passed on to the Z-centroid module. The output of the comparators is masked, and the valid hit register is passed on to the Hit Format module to find the hits and format them. The control module issues all the

control signals for the processes comparing, masking and formatting. The control signals are LOAD_ROAD, READ_COMPARATOR and HITREG_VALID.

Z- centroids module

This module formats the z-centroids in a 32-bit format and stores them in a FIFO. The hit interface module reads out the centroids from the FIFO. The 32-bit word formed is given in Table 3.5.

Table 3.5 The 32-bit word format of the Z-centroids.

31	30..28	27..26	25..24	23..16	15..13	12..0
0	SMT ID	Data type	Pulse area	SEQ ID	HDI ID	Centroid

Comparator module

The Comparator module has 46 parallel comparators. It has a capability to compare 46 pairs of roads with each incoming axial centroid. The road pair of upper and lower roads of 11-bit each is compared with 11-bit centroid value (the two precision bits are not used in the comparison, as the roads are defined as whole values).

Hit Register module

This module ANDs the output of the Comparators with the mask register to get a valid Hit Register for the each centroid when it receives a READ_COMPARATOR signal from the hit control module. This helps to filter out the false outputs from Comparators that are not loaded with roads.

Hit Format module

This module takes the valid 46-bit Hit Register when the hit control module issues the HITREG_VALID signal. The module checks for the hits serially. To make the scanning process faster, the register is split into five groups. The OR-ed output of each group indicates whether there is a hit in that group. The scanning of the register starts from the first group that has a hit, and then the logic goes through the rest of the register sequentially. The upper limit for this process is the number of Comparators loaded with roads. Whenever the bit is set, it is an indication of a hit for that particular track. The final output in the form of a 32-bit word (refer Table 3.6) is stored in the output FIFO of the Hit Filter module. At the end of hits for one centroid, the module waits for a next HITREG_VALID signal.

Table 3.6 The data format of the hits in the output FIFO.

31..26	25..24	23..16	15..13	12..0
Track number	Pulse area	SEQ ID	HDI ID	Centroid

At the end of event signal, a trailer (refer Table 3.7) is written into the FIFO. The module issues an independent end of event signal for the hit interface module.

Table 3.7 The data format of the trailer for the hits.

31..27	26	25	24..23	21..19	18..11	10..8	7..0
11110	SERR	MM	-	SMT ID	SEQ ID	HDI ID	Event no.

Hit Readout module

This module includes the FIFO in which hits are written in a unique 32 – bit word format (refer Table 3.6). When the hit interface module issues a READ HITS signal, the read request signal of the FIFO is activated and the hits are given out on each clock cycle.

The design of the data path was amended in different ways to optimize the speed, memory required and the logic cells utilized. The various approaches used for different modules are discussed in the following chapter.

CHAPTER 4

SIMULATION RESULTS OF THE VHDL MODEL AND COMPARISON WITH A MATLAB MODEL

The VHDL simulation model is described with the help of a test vector derived from Monte Carlo simulations of the response of the DØ detector to events of interest. These results are then compared to a MATLAB model of the same design. Each step in the flow of data through the main data path is given with a detailed description of the inputs and outputs of each module.

The VHDL Model

The test vector is given as data entering the SMT Data Filter from the VME bus. The test vector is given in Table 4.1.

Table 4.1 Test vector for an example simulation of the data path in hexadecimal.

Direction of the data stream →

↓

AA	77	81	00	40	03	41	0D	42	06	50	06	51	10
52	07	6B	03	6C	04	6D	05	6E	04	6F	03	77	07
78	06	79	07	7C	09	7D	10	7E	09	C0	C0	C0	C0

The data values are in HEX. This is a data stream of 8-bits each.

- The first byte is the SEQ ID – “AA”

- The second is the HDI ID – “77”
- The third is the SVX-II chip id – “81”. The SVX-II chip id should be followed by a byte of zeros – “00”.

The data stream after the byte of zeros is a strip number and the corresponding data value alternatively. The flow of the data through each module is described with reference to the detailed description of each module in Chapter 3.

SMT Data Filter

This module converts the 8-bit test data stream into 16-bit word and adds two error bits to it thus an 18-bit data stream comes out the filter. The data stream coming out is as shown in Table 4.2.

The last word in this data output is the end of event marker “C0” (in HEX) and the event number “91” (in HEX) obtained from the FRC (refer Chapter 3 - Section “SMT Data Filter”).

Strip Reader Control

This module starts on the EVENT_START signal. It pulls out the 18-bit word out of the intermediate FIFO. It converts the stream into higher byte and lower byte. The module first checks for the channel specific SEQ ID in higher byte and HDI ID in the lower byte. It waits for a valid SVX-II chip id in higher byte and the byte of zeros in the lower byte in the second word of the data stream. The most significant bit (MSB) of the higher byte; in this case bit 8 of the higher byte should be high. The SVX-II chip id decides the data type of the following strips.

Table 4.2 Output stream from the SMT Data Filter.

Error bits (2) (Binary)	Higher Byte (8) (HEX)	Lower byte (8) (HEX)
00	AA	77
00	81	00
00	40	03
00	41	0D
00	42	06
00	50	06
00	51	10
00	52	07
00	6B	03
00	6C	04
00	6D	05
00	6E	04
00	6F	03
00	77	07
00	78	06
00	79	07
00	7C	09
00	7D	10
00	7E	09
00	C0	91

The pair of strip number in higher byte and data value in lower byte follows. The corrected data is obtained by addressing the gain-offset memory with SVX-II chip id (“0001” – binary) and the data value. The bad channel information from the bad channel memory is matched with the strip number for each strip of the SVX-II chip. The resultant 23-bit word formed is as shown in Table 4.3.

Table 4.3 Output stream from the Strip Reader Control module.

Data type (Binary) (2-bits)	New data bit (1-bit)	End of event bit (1-bit)	Data (HEX) (8-bits)	Chip Id (HEX) (4-bits)	Channel ID (HEX) (7-bits)
10	1	0	03	1	40
10	0	0	0D	1	41
10	0	0	06	1	42
10	0	0	06	1	50
10	0	0	10	1	51
10	0	0	07	1	52
10	0	0	03	1	6B
10	0	0	04	1	6C
10	0	0	05	1	6D
10	0	0	04	1	6E
10	0	0	03	1	6F
10	0	0	07	1	77
10	0	0	06	1	78
10	0	0	09	1	79
10	0	0	07	1	7C
10	0	0	10	1	7D
10	0	1	09	1	7E

This chip is found to be of the axial data type, hence the data type is assigned as "10."

Cluster Finder

The Cluster Finder module gets each data value with its specific parameters packed as a 23-bit word. The module finds clusters according to the clustering algorithm as described in the flow chart attached in Appendix A.4.

The chart in Figure 4.1 shows the data values in the test data stream with the corresponding strip addresses, thus we can see from the chart clearly there will be five clusters in this data stream as is shown in the simulation results in Table 4.4.

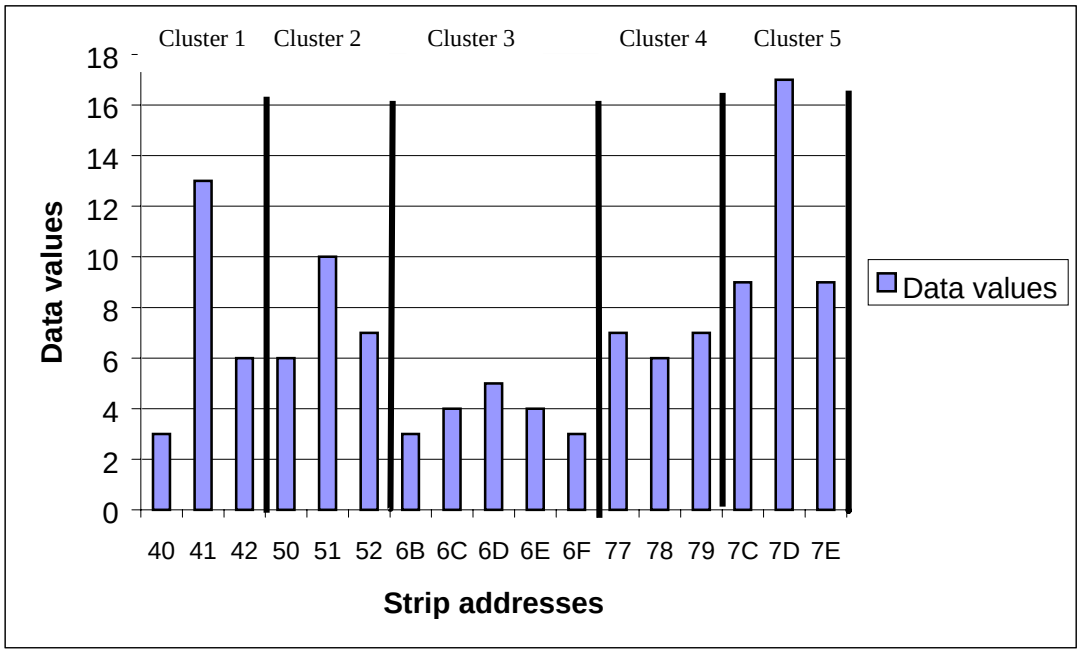


Figure 4.1 The data values in the test data stream with the corresponding strip addresses

Table 4.4: The clusters found by the Cluster Finder module

Address of D2 (11 bits)	D1 (8 bits)	D2 (8 bits)	D3 (8 bits)	D4 (8 bits)	D5 (8 bits)
0C0	00	03	0D	06	00
0D0	00	06	10	07	00
0EC	03	04	05	04	03
0F8	07	06	07	00	00
0FC	00	09	10	09	00

Centroid Calculator

The Centroid Calculator module finds the centroid according to the formula presented in the Chapter 3-Section "Centroid Calculator". The cluster type bit decides the cluster type. In this example the bit is set to "1", hence we calculate a five-strip cluster. The centroids for each of the above clusters as found by the module are given in Table 4.5.

Table 4.5: The centroids found by the Centroid Calculator module.

Data type (Binary)	Pulse area (Binary)	Centroid (11 bits HEX. 2 bits precision Binary)
10	11	0C1.00
10	11	0D1.00
10	11	0ED.01
10	11	0F8.10
10	11	0FD.01

Hit Filter

The Hit Filter, after getting the EVENT_START signal, waits for the road data. It gets the road data from FRC. The example of the road data values is given in the Table 4.6. These are the values used to compare the centroid values obtained from the Centroid Calculator.

Table 4.6: The road data values extracted from the road-data, with respect to the 17 – bit road-data value from FRC.

Lower road data (HEX)	Upper road data (HEX)
020	200
0B0	100
010	150
050	300

These road pairs are sequentially loaded into the Comparators. Each Comparator is assumed to have the same track number as the FRC track, as the roads come in sequentially. The Hit Filter now waits for the centroids to be processed. The processed centroids are read out one at a time. The z-axis type centroids are stored in a z-centroid FIFO, while the axial-type of centroid is loaded into the Comparators to find the hits. The hit format module reads out the Hit Register when they receive a HITREG_VALID signal. The hits for each centroid are written out in the output FIFO in a 32-bit format. The hits in this example are given in Table 4.7 below.

Table 4.7 The hits obtained written in the output FIFO.

Track [31..26] (Binary)	Pulse Area [25..24] (Binary)	Seq Id [23..16] (HEX)	HDI ID [15..13] (Binary)	Centroid [12..2] (HEX)	Precision [1..0] (Binary)
000000	11	AA	111	0C1	00
000001	11	AA	111	0C1	00
000010	11	AA	111	0C1	00
000011	11	AA	111	0C1	00
000000	11	AA	111	0D1	00
000001	11	AA	111	0D1	00
000010	11	AA	111	0D1	00
000011	11	AA	111	0D1	00
000000	11	AA	111	0ED	01
000001	11	AA	111	0ED	01
000010	11	AA	111	0ED	01
000011	11	AA	111	0ED	01
000000	11	AA	111	0F8	10
000001	11	AA	111	0F8	10
000010	11	AA	111	0F8	10
000011	11	AA	111	0F8	10
000000	11	AA	111	0FD	01
000001	11	AA	111	0FD	01
000010	11	AA	111	0FD	01
000011	11	AA	111	0FD	01

The whole data processing terminates at the Hit Filter. After the hits have been written in the hit output FIFO the channel waits for the hits to be read out by the hit interface module. Once the hits are read out the channel waits for next EVENT_START signal to start the data processing of new data stream on the VME bus.

The MATLAB model

The MATLAB model is functionally similar to the VHDL model, but it does not run synchronously. The model takes data from a file and stores the processed data in another file. This model was developed to generate test vectors for the different modules in the

VHDL model. The downloaded parameters are given in the form of input when the program is run. The data generated by this module is in binary i.e. 0's and 1's. The data streams generated by this model were compared to the data streams obtained from the VHDL model. The MATLAB code written to realize this model is in the attached Appendix D. The MATLAB consists of SMT filter, Strip Reader, Cluster Finder, Centroid Calculator and the Hit Filter. The flow of data through each of these modules is explained in detail below.

Main design

This is the main design file used to run all module files sequentially. The modules access the downloaded parameters file and the other data files and the data streams are written into the respective data files, reference D.1

Read downloaded parameters

This file is used to extract the downloaded parameters from the data file downloaded parameters.m and store them in the file down_data.m from where it accessed by the data processing modules, reference D.2

SMT filter module

This module reads the HEX data stream from the file "vtm_data.m". The module converts the 8-bit data stream to 16-bit data word. The error bits are also provided along with the VTM data stream. The 18-bit data word formed (refer Table 4.2) is written in binary format in file "smt_file.m", reference D.3

Strip Reader

The Strip Reader module reads the data sequentially from either the smt_data.m file or the test_data.m file, depending on the test input, which is a user input. The SEQ ID, HDI ID, data type, gain and offset are accessed from the data file down_data.m. The data is processed to get the 23-bit word (refer Table 4.3) in binary format. This data is written into the file “strip_data.m”, reference D.4

Cluster Finder

The Cluster Finder module takes the 23-bit word from file “strip_data.m” and finds the clusters according to the clustering algorithm in APPENDIX A. The threshold_1 and threshold_2 values are accessed from data file down_data.m. The data values of the five clusters with address of the data value in buffer 2 (i.e. peak address – 1) are stored in the file “cluster_data.m”, reference D.5

Centroid Calculator

The cluster type and the pulse area threshold values are accessed from data file down_data.m. The calculation of the centroid is carried out in decimal and then the result is converted into binary format. The result as in Table 4.6 is stored in the file “centroids_data.m” in binary format, reference D.6.

Hit Filter

The Hit Filter reads the road pairs from the file “roads_data.m” and the centroids tagged with their data type and the pulse area from the file “centroids_data.m”. Each centroid is compared sequentially with each road pair and the output is written out in the file

“hits.m”, reference D.7 . The output consists of the track number, pulse area and the centroid.

The VHDL module and the MATLAB module agree on the centroids and the hits. Thus the design is functionally correct. The MATLAB module thus helps to check functionality of each module individually as we can check the test streams at the end of each module. Thus the algorithmic approach helps to check the state approach taken in the VHDL model.

CHAPTER 5

DESIGN ISSUES FOR IMPLEMENTATION OF THE MAIN DATA PATH

The design for the main data path was developed using VHDL. Different design approaches were studied with the aim of developing a compact, functionally correct and fast design. The design approaches studied for the Hit Filter is presented, and the different design implementations approaches taken to fit the whole design (i.e. main data path with the L3 buffers) are also discussed in this chapter.

The Hit Filter design approaches

The Hit Filter was required to have the capability to compare 46 road data values with a centroid value at a time. Different combinations of parallel and serial implementations were studied. The results are consolidated in Table 5.1.

As can be observed from Table 5.1 the number of logic cells required increases linearly with the number of comparators put in parallel. If we have a serial and parallel combination of the comparators there is a time delay in the switching of the bus possession. Also this approach does not reduce the logic cells utilization. Thus, the final scheme of all 46 comparators in parallel was chosen for high-speed comparison. The outputs of the comparators are read out serially so that they can be put out in the required 32-word format.

Table 5.1: The result of comparison for putting filters in parallel

	No of filters in parallel	No of inputs	No of outputs	No. Of LCs required
1.	2	37	2	101
2.	4	39	4	199
3.	6	41	6	297
4.	8	43	8	395
5.	10	45	10	493
6.	12	47	12	591
7.	14	49	14	689
8.	16	51	16	787
9.	18	53	18	885
10.	20	55	20	983
11.	22	55	22	1081
12.	24	59	24	1179
13.	26	61	26	1277
14.	28	63	28	1375
15.	30	65	30	1473
16.	32	67	32	1571
17.	34	69	34	1669
18.	36	71	36	1767
19.	38	73	38	1865
20.	40	75	40	1963
21.	42	77	42	2061
22.	44	79	44	2159
23.	46	81	46	2257

The different implementation schemes of the overall design

The hardware implementation of the overall design, i.e., the main data path with the L3 buffers was tried in FLEX 10KE FPLDs. There also exists an external memory of 16 K for the road data.

During this implementation four design approaches were studied. These approaches are discussed in detail in this section.

Approach 1

The synthesis tool was allowed to fit the design in the minimum possible number of FPLDs. The tool required a minimum of five FPLDs as shown in Table 5.2.

Table 5.2: Results of the compilation: Approach 1

Chip name	Chip	Inputs	Outputs	Memory Bits	Logic cells	EABs
Strip_reader_ hitfilter_l3 _schematic	EPF10K200 EGC599-1	241	187	24772 (25%)	2003 (20%)	22 (91%)
Strip_reader_ hitfilter_l3 _schematic-1	EPF10K200 EGC599-1	122	316	58752 (59%)	4211 (42%)	22 (91%)
Strip_reader_ hitfilter_l3 _schematic-2	EPF10K30 ETC144-1	42	37	4608 (18%)	368 (21%)	2 (33%)
Strip_reader_ hitfilter_l3 _schematic-3	EPF10K30 EQC208-1	59	63	8192 (33%)	334 (19%)	2 (33%)
Strip_reader _hitfilter_l3 _schematic-5	EPF10K50 EQC208-1	86	29	4324 (10%)	2225 (77%)	10 (100%)
Total				100648	9141	58

The tool tried to fit in the memory blocks first and then the logic into the FPLDs thus chose the FPLDs according to the memory capacity first and then fitted the logic cells into these FPLDs. In this approach the sizes of the FPLDs is varying, which is not preferable for the design.

Approach 2

To study the behaviour of the synthesis tool, the Hit Filter forced to fit in one FPLD named Hitfilter_schematic, and the L3 buffers is forced to fit in one FPLD named L3_schematic. The tool was allowed to fit the Strip Reader chip (i.e. the Strip Reader and

the Cluster Finder modules) in as few FPLDs as possible. The results are as shown the Table 5.3.

Table 5.3: Results of the compilation: Approach 2

Chip name	Chip	Inputs	Outputs	Memory Bits	Logic cells	EABs
Hitfilter _schematic	EPF10K100 EBC356-1	87	170	10532 (21%)	4340 (86%)	12 (100%)
L3 _schematic	EPF10K200 EGC599-1	175	291	79424 (80%)	2941 (29%)	24 (100%)
Strip_reader _hitfilter_l3 _schematic_1	EPF10K200 SFC484-1	169	123	10692 (10%)	1860 (18%)	19 (79%)
Total				100648	9141	55

After fitting the assigned modules to their FPLDs the synthesis tool fitted the memories in the L3_schematic FPLD and the excess logic into the Hitfilter_schematic FPLD, thus fitting the whole design in three FPLDs. This approach is not acceptable as the logic for the Strip Reader is spread into three FPLDs and thus there will be additional propagation delay on critical signals.

Approach 3

In this approach, the synthesis tool was given some guidance by forcing the Hit Filter to fit in one FPLD the Hitfilter_schematic, the L3 buffers in one FPLD the L3_schematic. The Strip Reader with some memory modules is forced to fit in one FPLD the Strip_reader_chip-1 and the Cluster Finder with remaining memory modules is forced to fit in one FPLD the Strip_reader_chip-2. The results are given below in Table 5.4.

Table 5.4: Results of the compilation: Approach 3

Chip name	Chip	Inputs	Outputs	Memory Bits	Logic cells	EABs
Hitfilter _schematic	EPF10K100 EBC356-1	77	144	10532 (21%)	4012 (80%)	12 (100%)
L3 _schematic	EPF10K130 EFC484-1	183	175	40960 (62%)	1576 (23%)	13 (81%)
Strip_reader _chip-1	EPF10K200 EGC599-1	179	216	45120 (45%)	3244 (32%)	18 (75%)
Strip_reader _chip-2	EPF10K130 EFC484-1	124	183	4036 (6%)	309 (4%)	14 (87%)
Total				100648	9141	57

The Strip Reader requires two FPLDs as each memory space when assigned to an Embedded Array Block (EAB) utilizes minimum of more than one EAB. The EABs can store 8-bit wide word, for word lengths greater than 8-bits the EABs are concatenated. Thus even the small memory spaces were using more than one EAB even though the actual space utilized was 2-3 words with word lengths going up to maximum 32-bit wide. Thus the whole design fitted successfully in four FPLDs. Thus, the Approach 4 was taken to reduce the number of FPLDs.

Approach 4

From the conclusion of Approach 3 the EAB assignment of the small memory modules was removed and they were implemented using logic cells. The results for the design with the changes are given in Table 5.5.

The design is now found to fit successfully in three FPLDs which are very close to each other in size, thus for the final layout the largest FPLD of the FLEX10KE family can be chosen to permit further changes in the design. Since the logic cells replaced some of the EABs, a comparison of the change in the number of memory bits used and the logic cells utilized to substitute the EABs is given in Figure 5.1, Figure 5.2 and Figure 5.3, where 1

denotes approaches 1, 2, 3 and 2 denotes approach 4.

Table 5.5: Results of the compilation: Approach 4

Chip name	Chip	Inputs	Outputs	Memory Bits	Logic cells	EABs
Hitfilter _Schematic	EPF10K100 EBC356-1	77	144	10532 (21%)	4012 (80%)	12 (100%)
L3 _Schematic	EPF10K130 EFC484-1	183	175	40960 (62%)	1576 (23%)	13 (81%)
Strip_reader_ chip_schematic	EPF10K200 SBC356-1	76	174	45120 (45%)	4773 (47%)	17 (70%)
Total				96612	10361	42

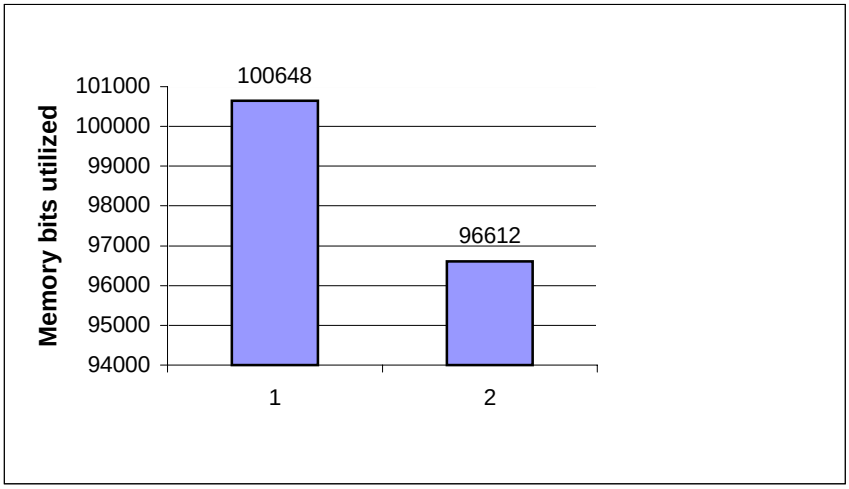


Figure 5.1 Comparison of the memory bits utilized

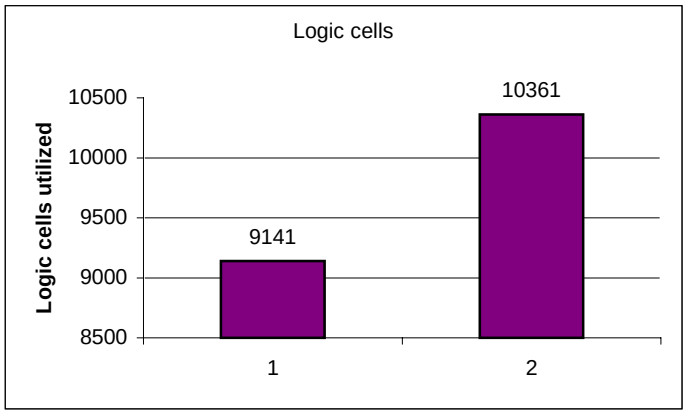


Figure 5.2 Comparison of the logic cells utilized

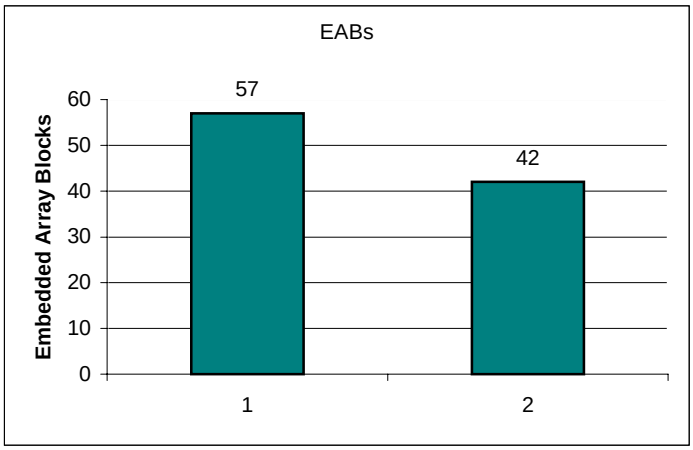


Figure 5.3 Comparison of the EABs utilized

Implementation of the design using Quartus software

The design presented in this thesis is for one STC channel of the Level_2 of the D0 detector. There will be eight such identical channels running in parallel (refer Chapter 2 – Section “Level_2”). All the channels are proposed to fit on to one Printed Circuit Board (PCB).

The design is successfully fitted in three FPLDs, but this amounts to 24 FPLDs for the eight STC channels and additional FPLD for the Main Control module (refer Chapter 2- Section “Main Control module”). Therefore 25 FPLDs should be fitted on to one PCB. This is not a difficult task, but the size of the PCB required will be really large and this is not suitable for the design. Thus a new approach has been examined to fit the whole design in a single APEX20KE FPLDs.

The APEX20KE is one of the latest FPLDs offered by ALTERA. The synthesis tool used for implementing the design in the APEX20KE is *QUARTUS.* The strip reader module with bad channel memory, gain offset memory, test data memory and the monitor space was successfully implemented in an APEX20KE. The results of this implementation are shown

in Table 5.6.

Table 5.6 Implementation of the Strip Reader module in APEX 20KE.

Device name	EP20K300EBC652-1
Logic elements	2643/11520 (22%)
Pins	395/408 (96%)
Memory bits	49024 /147456 (33%)
ESBs	28/72 (38%)

The APEX20KE family has chips of high memory capacity and large number of logic elements. The specifications of the largest FPLD available in this family are given in Table 5.7.

Table 5.7 Specifications of EP20K1500E

(Largest FPLD in APEX20KE family) [20].

Voltages	2.5 V and 1.8 V
Maximum system gates	2,392,000
Typical gates	1,500,000
Logic Elements	51,840
ESBs	216
Maximum RAM bits	442,368
Maximum macro-cells	3,456
Maximum user I/O pins	808

CHAPTER 6

SUMMARY

This thesis describes the implementation of the main data path that constitutes one channel of the Silicon Track Card (STC) card in the Level_2 of the D0 detector at Fermi National Acceleration Laboratory. There are eight such channels per card and the D0 detector has 54 such cards mounted around the accelerator within the detector.

Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) is used to describe the behavioral model of the main data path constituting the Strip Reader, the Centroid Finder and the Hit Filter. The design is implemented on three FLEX10KE FPLDs. First, the functional correctness of the design was verified, and then timing studies were conducted. When implemented in the low-memory FLEX10K FPLD, the design requires three FPLDs per channel of STC. However, the timing requirements were not met. The limiting paths in the timing studies were found to the paths writing data into the memory spaces. These are going to be accessed once every event and the time available for this is a few minutes. The data processing signals are not present in these limiting paths.

To improve the timings and to also to avoid the connections between the FPLDs for one channel, implementation of the design of a single channel in one FPLD is studied. In the course of these studies, the Strip Reader module was alone implemented in the APEX20K family by ALTERA Corporation. This family of FPLDs has a high density of logic gates and a large memory capacity. The results obtained from the implementation were used to predict the resources required for one channel of STC. The results show that three channels of STC can be implemented in one EP20K1500E, which the largest device available in the

APEX20K family. The new design implementation will be able to meet the design requirements, as the propagation delays on the signals will decrease. Thus the new design implementation will help to improve the performance of the design logic and also help to reduce the Printed Circuit board (PCB) complexity.

In addition to the APEX20K FPLD family, the Virtex family by XILINX is also a good option for use as the larger FPLD (refer Table 1.1).

BIBLIOGRAPHY

1. M. J. S. Smith, *"Application Specific integrated Circuits"*, Addison – Wesley, 1997.

2. J. P. Huber and Mark W. Rosneck, *"Successful ASIC Design the First Time Through"*, Van Nosstrand Reinhold, New York, 1991.

3. V.L. Burton, "*The Programmable Logic Device Handbook*", Blue Ridge Summitt PA, Tab book, 1990.

4. B. Dipert, *" Low-cost programmable logic: How low should you go?"* EDN, March 16 2000.

5. G. Bostok, *"Programmable Logic Handbook",* Collins Professional Books, 1987.

6. J. Bhasker, *"VHDL Primer"*, Prentice Hall Series of Innovative Technology, 1999.

7. D. R. Coelho, *"The VHDL Handbook"*, Kluwer Academic Publisher, 1989.

8. J. Villasenor and W.H. Mangione-Smith, *"Configurable Computing"*, Scientific American, June 1997, pp 66-71.

9. W. H. Mangione-Smith et. al., *"Seeking Solutions in Configurable Computing"*, Computer, Vol.30, No.12, 1997, pp 38-43.

10. E. Waingold et al., *"Baring It All to Software: Raw Machines"*, Computer, Vol.30, No.9, 1997, pp 86-93.

11. Fermi National Accelerator Laboratory, Profile

12. F. Abe et al., *Phys. Rev. Letters 74: 2626-2631*, 1995.

13. S. Abachi et al., *Phys Rev. Letters 74: 2632-2637, 1995*.

14. G. C. Blazey, *"The DØ Run II Trigger"*, Department Of Physics, Northern Illinois University, DeKalb, Illinois.

15. *"A Silicon Track Trigger for the DØ Experiment in Run II - Proposal to Fermilab – DØ note 3516",* September 1998.

16. J. T. Linnemann, *"The DØ Level 2 Trigger"*, Proceedings of the 2000 Meeting of the Division of Particles and Field of the American Physical Society, Columbus, Ohio, August 2000.

17. W. Earle, E. Hazen, U. Heintz, M. Narain, *"Specification for the STC Daughtercard"*, Technical Report by Physics Department, Boston University February 15 2000.

18. S. Lolage, K. A. Meyers, R. Brown, R. Perry, *"VHDL Implementation of the Baseline Centroid Finder Algorithm"*, Internal Presentation report, Electrical and Computer Engineering Department, FAMU-FSU COE, January 2000.

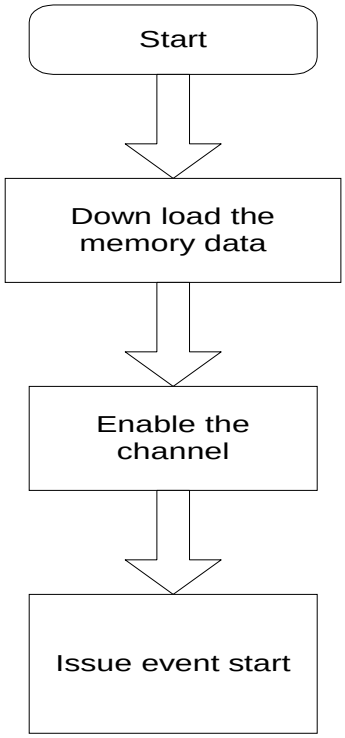
19. W.E. Earle, E. Hazen, M. Narain, U. Heintz, *"Specifications for One Channel of the STC Logic"*, Technical report by Physics Department, Boston University, June 22 2000.

20. ALTERA, *"Data Sheet for APEX20K, Programmable logic Device Family"*, version 2.06, March 2000.

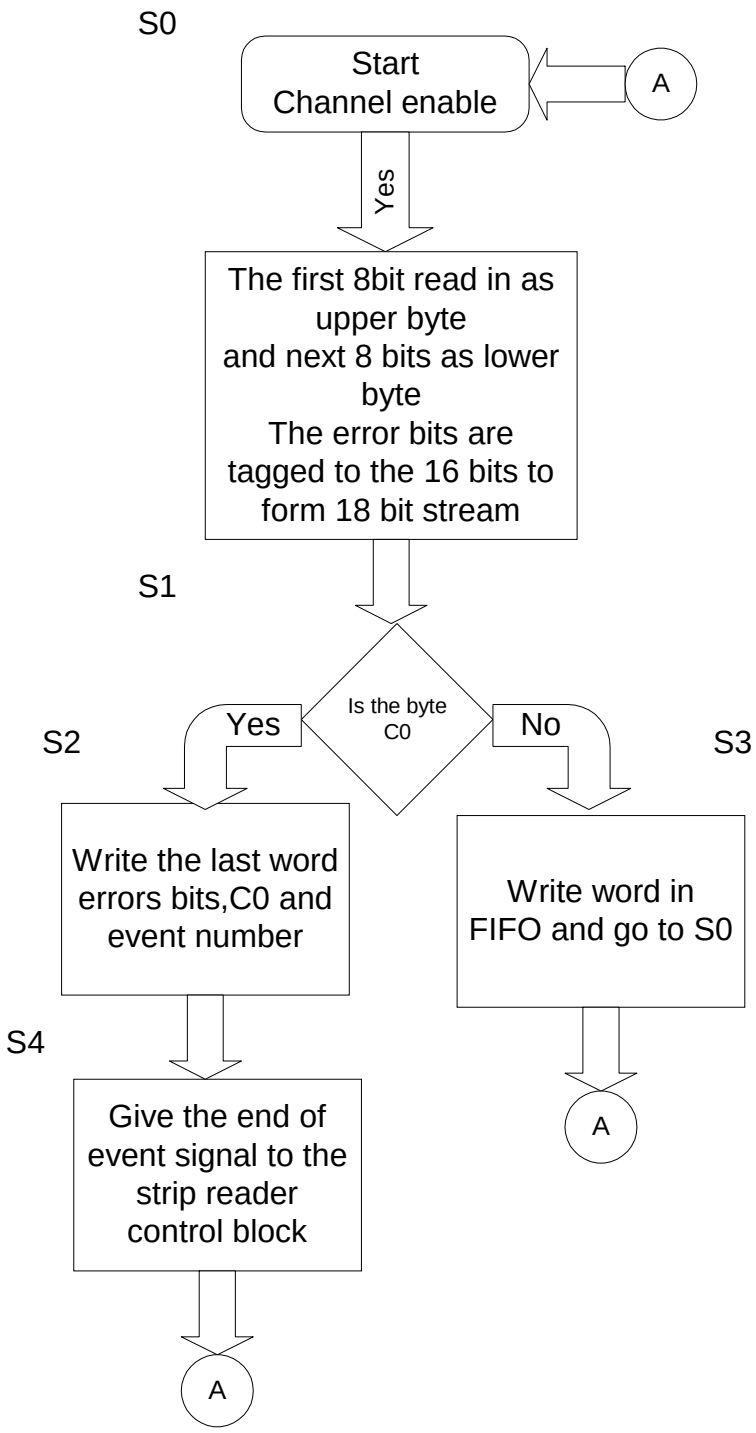
APPENDIX – A

FLOW CHARTS OF THE MODULES OF THE STC MAIN DATA PATH

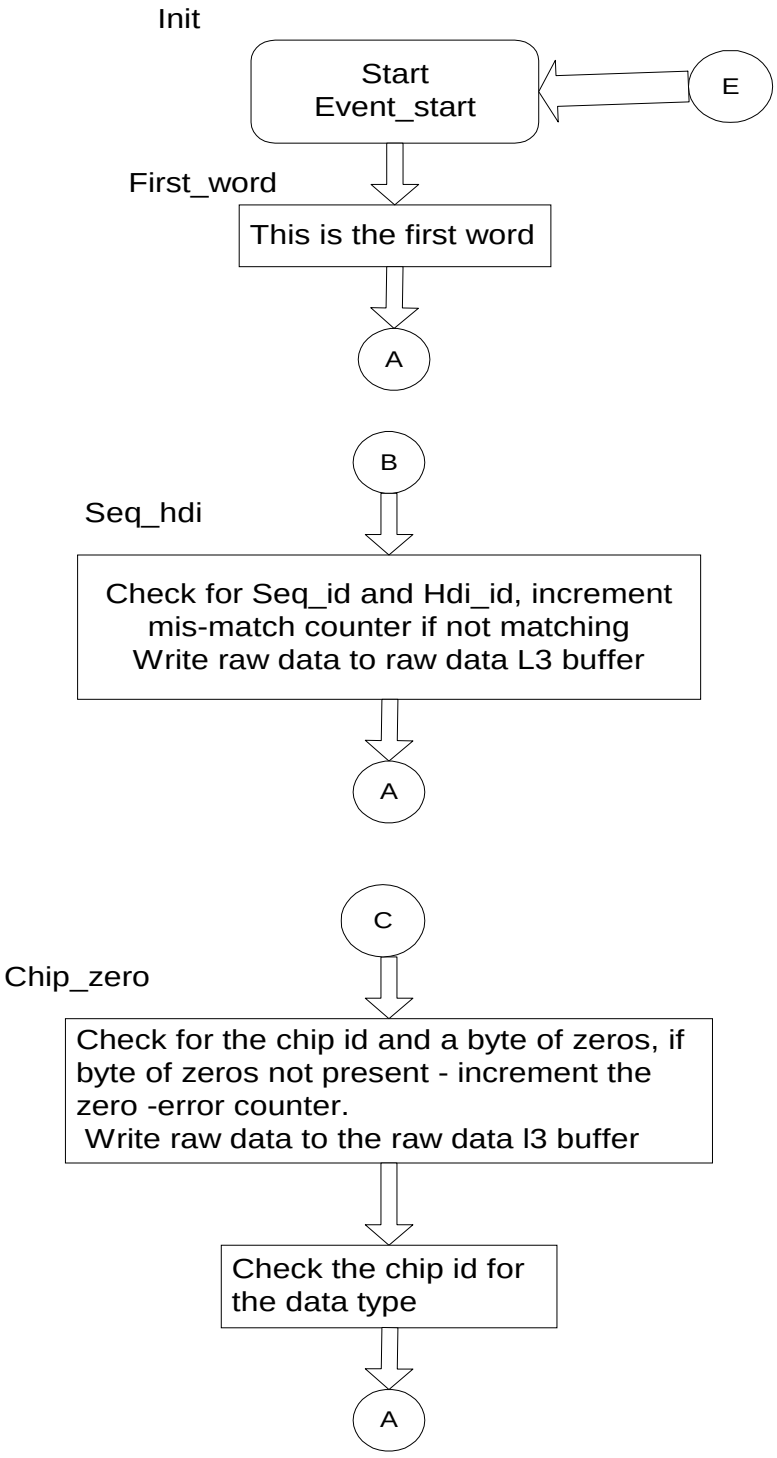
A.1 SIGNALS FROM THE MAIN CONTROL MODULE



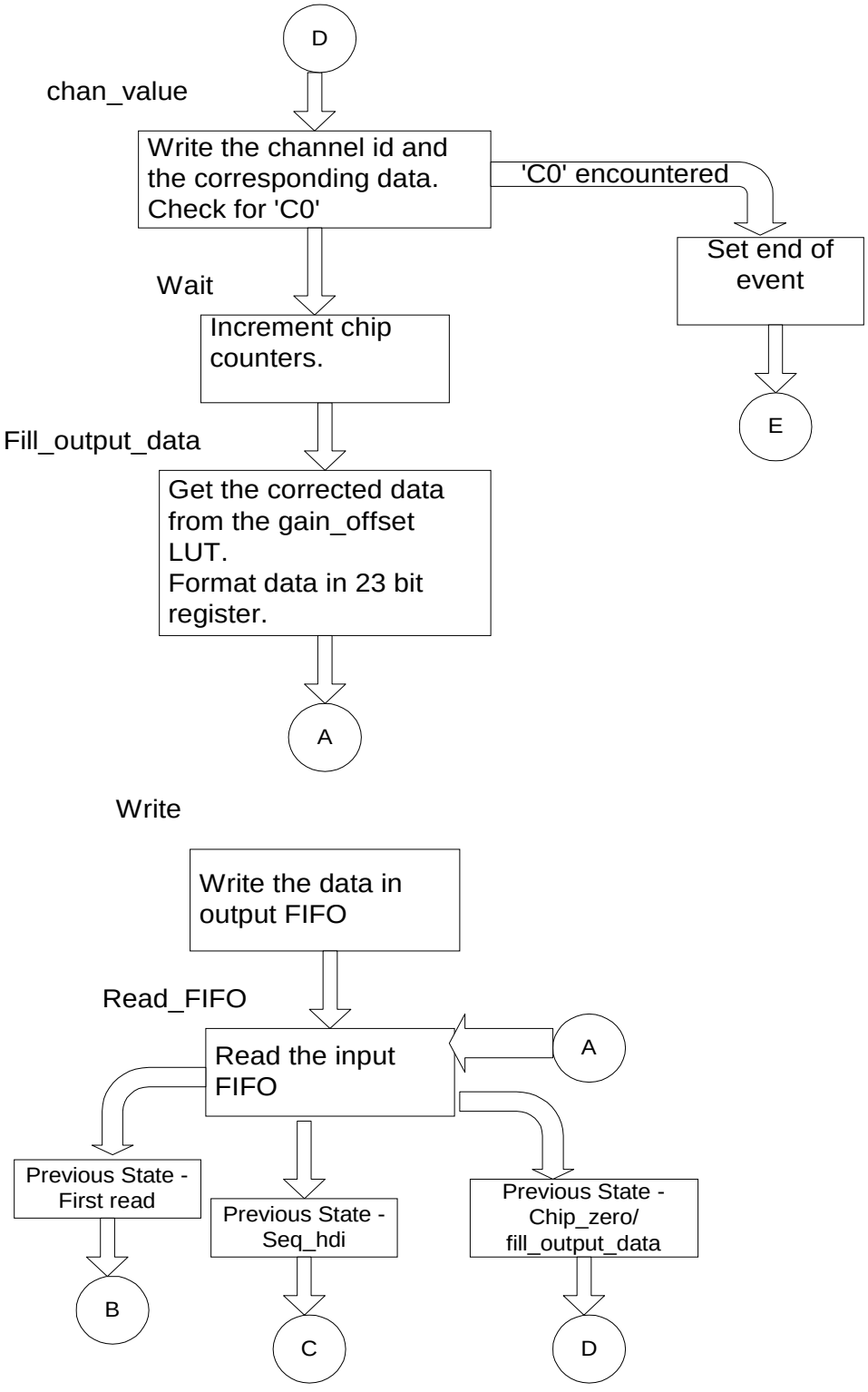
A.2 SMT DATA FILTER



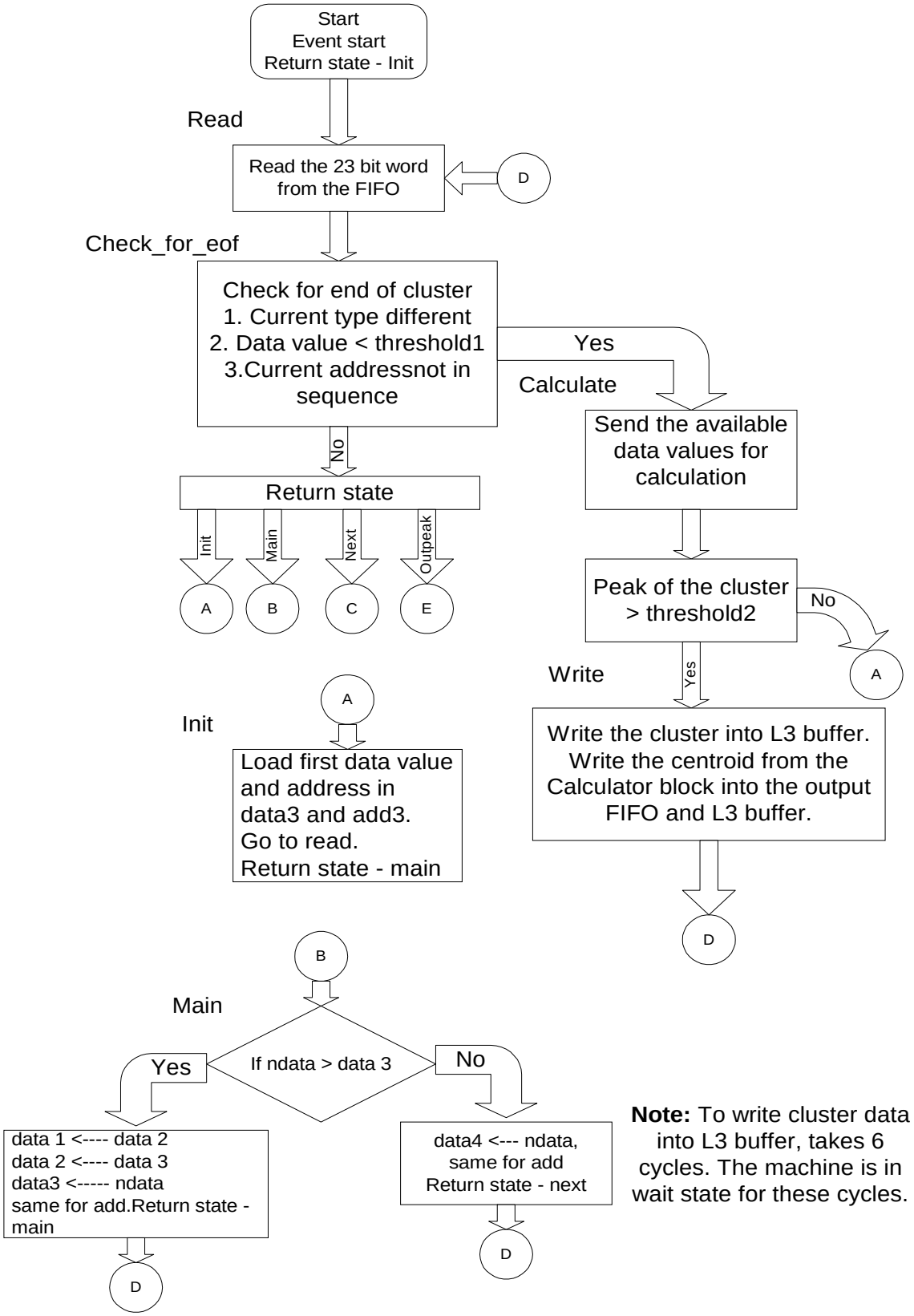
A.3 STRIP READER CONTROL



A.3 STRIP READER CONTROL (CONTINUED….)

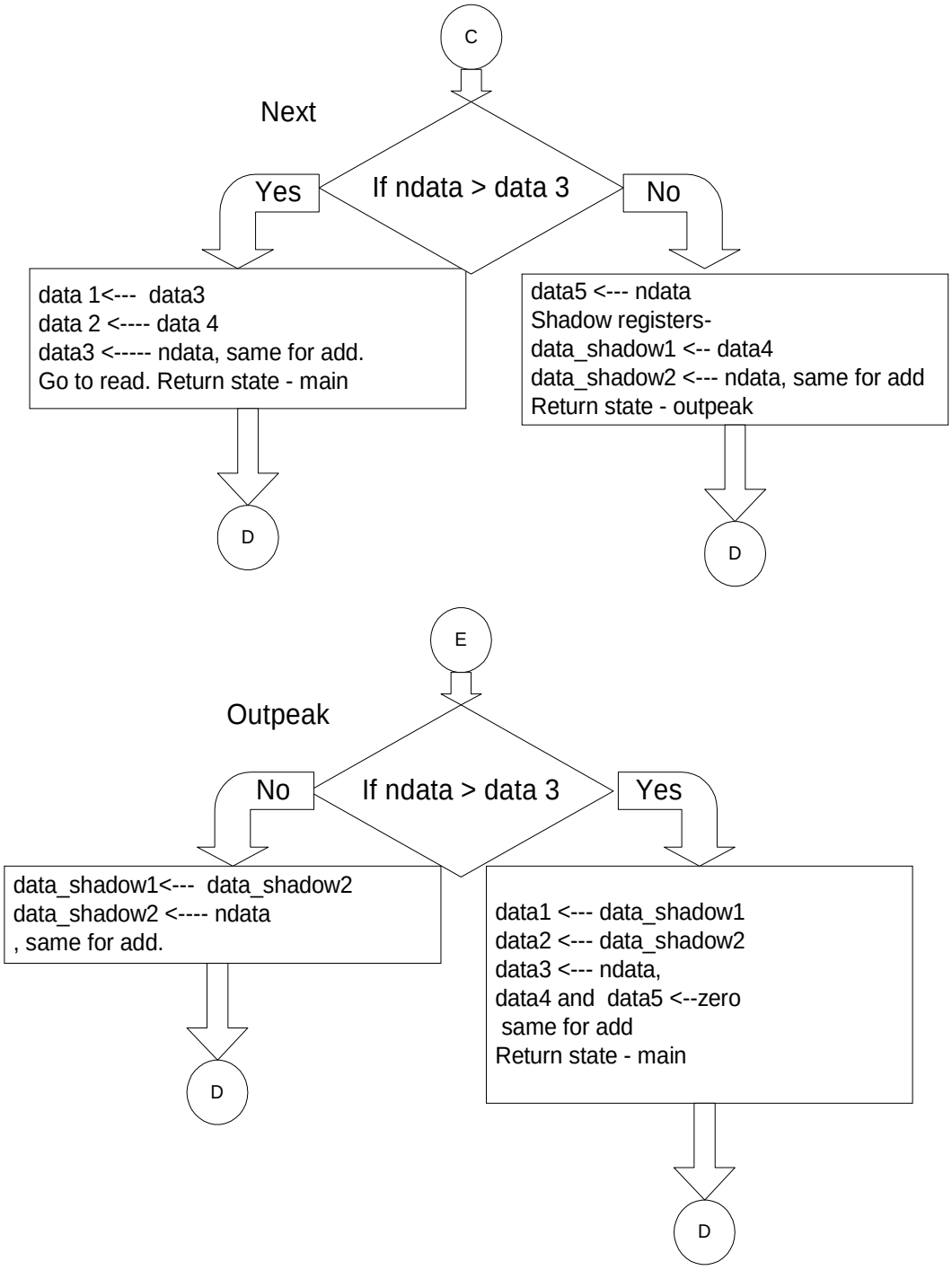


A.4 CLUSTER FINDER

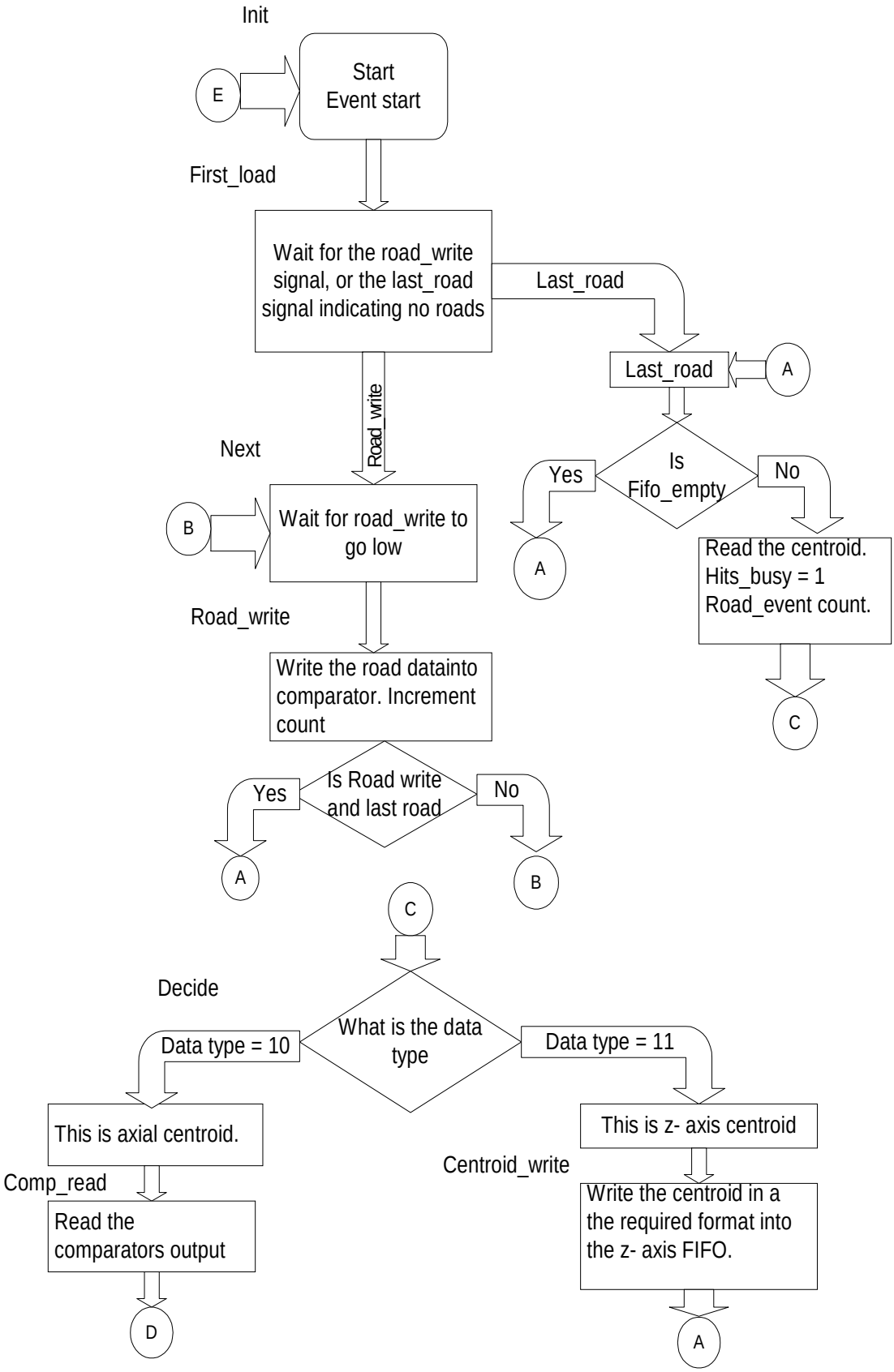


Note: To write cluster data into L3 buffer, takes 6 cycles. The machine is in wait state for these cycles.

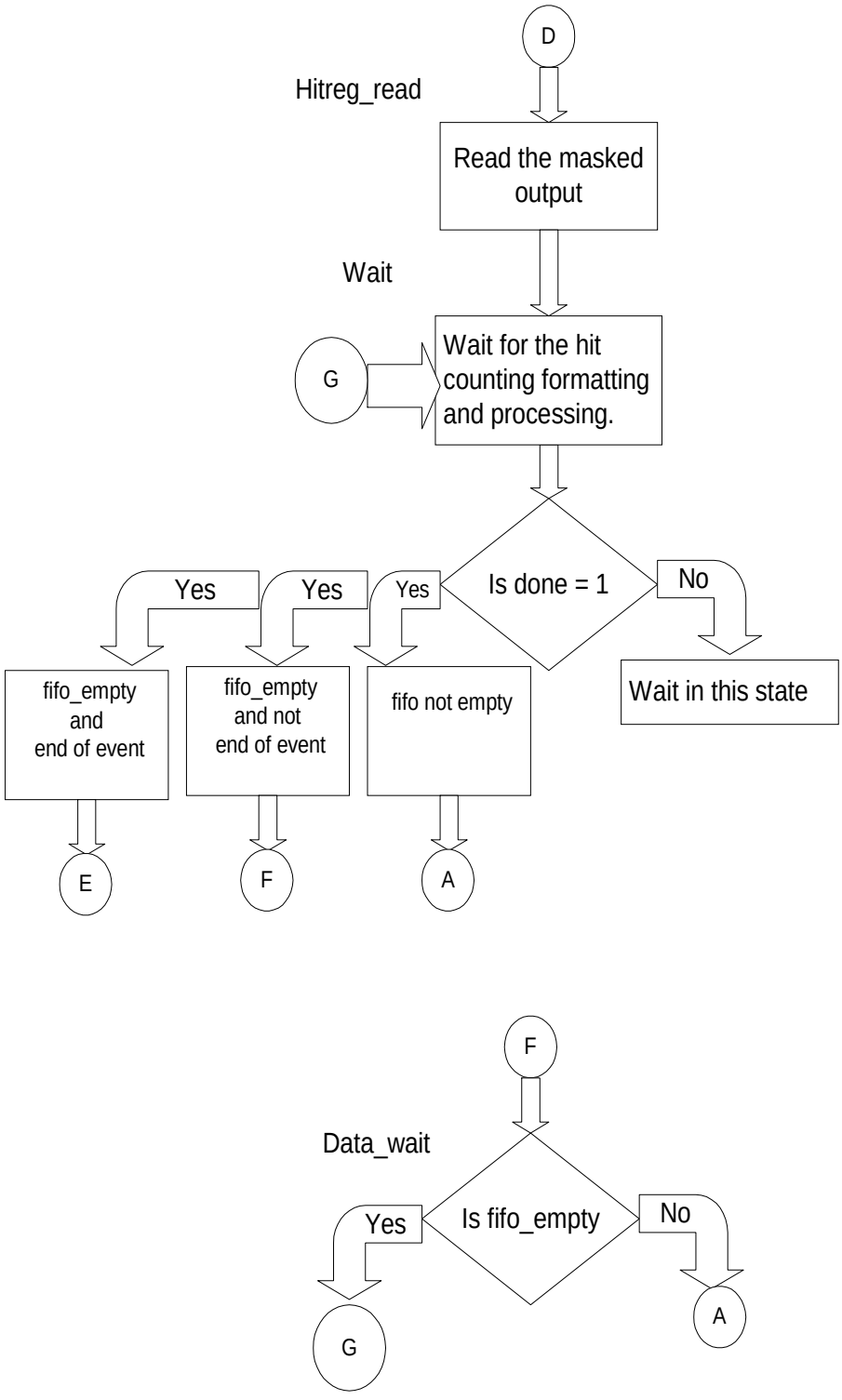
A.4 CLUSTER FINDER (continued)



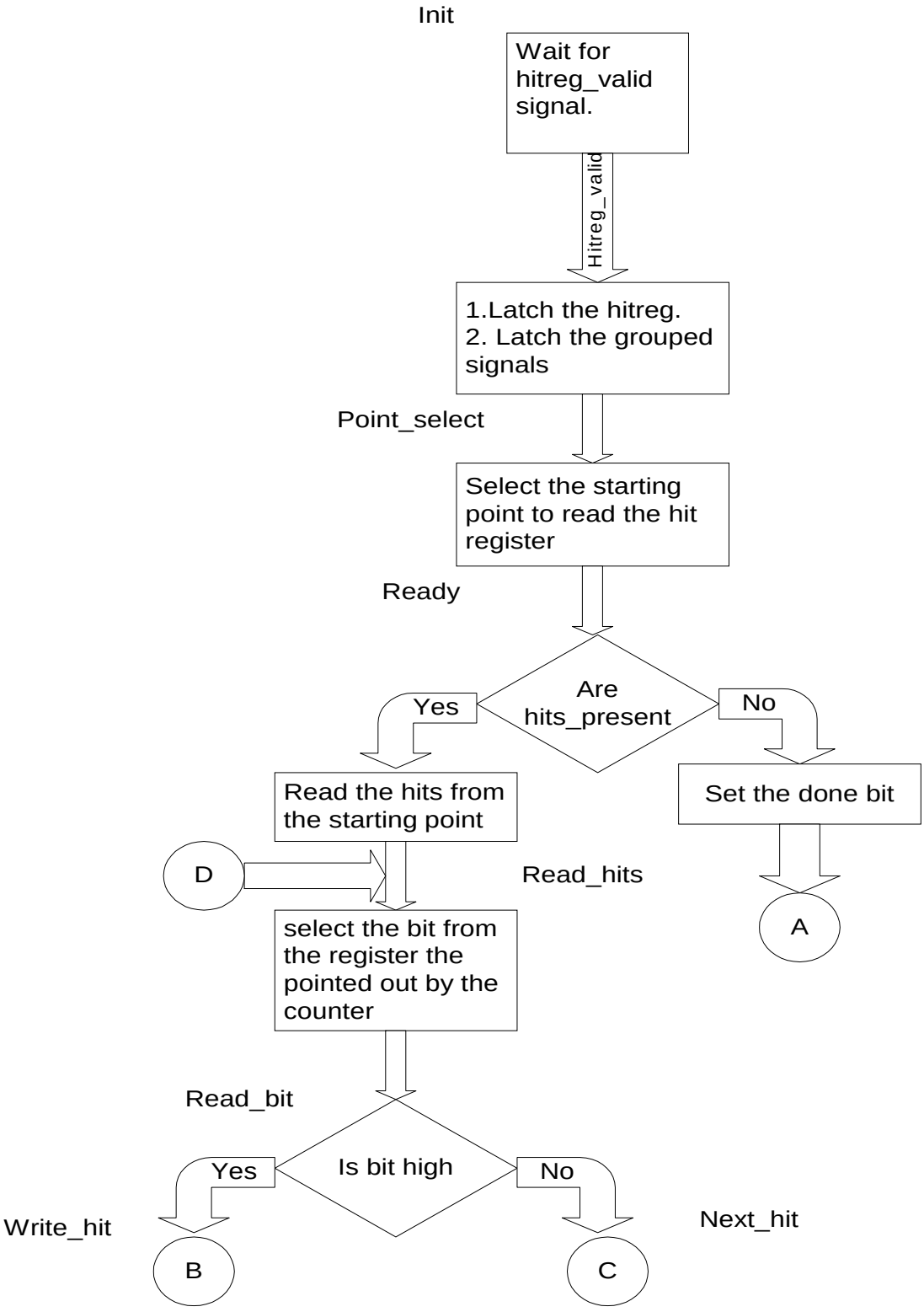
A.5 HIT FILTER – Hit filter control logic



A.5 HIT FILTER – Hit filter control logic (continued)



A.6 HIT FILTER - Hit filter hit_format logic



A.6 HIT FILTER - Hit filter hit_format logic (continued)