# A CAM-BASED, HIGH-PERFORMANCE CLASSIFIER-SCHEDULER FOR A VIDEO NETWORK PROCESSOR

Srivamsi Tarigopula

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2008

APPROVED:

Saraju P. Mohanty, Major Professor
Elias Kougianos, Committee Member
Murali Varanasi, Committee Member
Krishna Kavi, Chair of the Department of Computer Science and Engineering
Oscar Garcia, Dean of the College of Engineering
Sandra L. Terrell, Dean of the Robert B.

Tarigopula, Srivamsi. A CAM-based, high-performance classifier-scheduler for a video network processor. Master of Science (Computer Engineering), May 2007, 82 pp., 3 tables, 24 figures, references, 67 titles.

Classification and scheduling are key functionalities of a network processor. Network processors are equipped with application specific integrated circuits (ASIC), so that as IP (Internet Protocol) packets arrive, they can be processed directly without using the central processing unit. A new network processor is proposed called the video network processor (VNP) for real time broadcasting of video streams for IP television (IPTV). This thesis explores the challenge in designing a combined classification and scheduling module for a VNP. I propose and design the classifier-scheduler module which will classify and schedule data for VNP. The proposed module discriminates between IP packets and video packets. The video packets are further processed for digital rights management (DRM). IP packets which carry regular traffic will traverse without any modification. Basic architecture of VNP and architecture of classifier-scheduler module based on content addressable memory (CAM) and random access memory (RAM) has been proposed. The module has been designed and simulated in Xilinx 9.1i; is built in ISE simulator with a throughput of 1.79 Mbps and a maximum working frequency of 111.89 MHz at a power dissipation of 33.6mW. The code has been translated and mapped for Spartan and Virtex family of devices.

ACKNOWLEDGEMENTS

I would like to extend my unconditional thanks to Dr. Mohanty, who has been a motivation towards my research and guided me through my thesis work. I would like to thank Dr. Kougianos for taking out time to review my work and his support. My heartfelt thanks go to Dr. Varanasi, for his support and valuable advice.

My friends and colleagues have been invaluable; the friendly staff at UNT also has my appreciation. Finally and certainly not the least, I would like to thank my family for being infinitely supportive. Time spent at UNT will always be remembered as good time.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

# CHAPTER 1

# INTRODUCTION

Internet protocol television (IPTV) is one the fastest growing industries in the world. It is estimated that all most 20% percent of television friendly families will switch to IPTV by the end of this year, and this figure is predicted to be doubled by the end of next year as quoted by IPTVnews.net. This shows the dependency of a regular individual on the internet network. Internet connects the whole world and it might not be an easy but a very luring step to merge internet and television, this comes with advantages such as, single framework to support two different functionalities, better quality more embedded functionality in television channel transmission and so on, however, Internet has an imminent danger of being susceptible to hacking and piracy. To protect the digital rights of the television content being streamed, video network processor (VNP) is proposed. This thesis aims at researching the architecture of the processor and, building the basic blocks of the processor, which start with the classifier-scheduler, this module tailored in such a way that new modules can be embedded easily. Motivating factors of the thesis are highlighted.

## 1.1 Motivation

The transmission of digital television (DTV) over the Internet protocol is called IPTV [1], [2], [3]. IPTV has been embraced by the new world for its various advantages, such as accessibility of internet and television over a common framework, better quality of service (QoS), low cost and low energy bill. This technology also poses a potential threat which is inherent to Internet – data theft.

To secure the transmission of television and video data transmitted over the internet, from the content provider to the end user; the design of VNP has been proposed and this thesis aims towards developing the data path, elements and architecture and placement of this box, and the combined classifier and scheduler module , classifier – scheduler has been proposed, this is conceived to be a system on chip design – SoC [4].

VNP is an application specific router which picks up IP packets with raw “moving picture experts group” (MPEG) video data streams of real time protocol (RTP) from the content provider [5]. VNP secures the, data through watermarking, scrambling and encryption and then either broadcast, multicast or singlecast the packet. The thesis focuses on classifying IP packets carrying MPEG streams from the content provider, IP packets carrying regular data and forwarding them to the processing units and scheduling them accordingly.

## 1.2 History of Digital Television (DTV)

Transmitting television content through digital signals, instead of transmitting it through analog signals is called as digital television. To view digital video streams, either a digital television, or its corresponding decoder or a demodulator (places where analog television is still transmitted and used) are used, these decoders and demodulators are called as set top boxes are to be used, these are connected to the end user’s television set. There are several standards to transmit digital television, such as national television system committee (NTSC), phase alternating line (PAL), and high definition television

(HDTV) [6], [7]. These services can be multiplexed to receive a variety of services and resolutions.

## 1.3 Internet Protocol Television (IPTV)

IPTV takes one step ahead of DTV and embeds DTV data into IP packets and transmits it onto the internet network, so that, both television and TV can be accessed through one framework. The traditional set top boxes which took care of decoding DTV data have been modified to take care of extracting TV data from IP packets. Usually a locally placed server (local to the content provider) picks up streams of video and audio data preferably MPEG streams and embeds video data into IP packets. This infrastructure takes care of video on demand (VoD), authentication, fragmentation, channel change etc [8]. It always helps if the content provider has a proper tie up with Internet service providers (ISP's) rather than acting as a normal internet client as, having a proper tie up, helps in handling of QoS and quality of experience (QoE). IPTV supports as mentioned MPEG standards, which include MPEG – 2, MPEG – 4 Part 2, MPEG – 4 part 10(AVC) [9], Microsoft VC – 1. The bottle neck of this technology would be, some digital subscriber lines (DSL) which support low data rates, cable is optimum as it supports around 4.5Mbps, however newest DSL – ADSL2 can support around 25Mbps and can be used to transmit IPTV.

## 1.4 Network Processors

Network processors are family of devices which target the processing of network based applications. These processors allow the developers to implement latest protocols and services without changing the underlying hardware and give

an advantage over usual central processing units, by giving better performance, and saves power as the processor does not have to constantly switch between number of processes [10].

This family of processors includes,

1. Active routers
2. Switches
3. Firewalls
4. Network monitors
5. Intrusion detection and prevention engines

Network processors mimic limited functionality of the central processing units to achieve specific network based applications such as – packet discrimination and forwarding, encrypt incoming data streams, recognize certain data patterns in the traffic based on look up table information, transmission control protocol (TCP) and user datagram protocol (UDP) off load processing, detection of intrusion and intrusion prevention.

Network processors are usually a set of hardware equipment which can be implemented in either – application specific integrated circuit (ASIC), application specific instruction processor (ASIP), co-processor technology, field programmable gate array (FPGA), or on general purpose processor (GPP) technologies.

## 1.5 Network Processor as an Active Router

A router is a networking device which helps one peer transmits data to another. A router, routes and forwards IP packets based on the host and destination IP addresses. Routers are also called as level 3 switches and they can also act as firewalls. Active routers, have processing units embedded onto them which enable them to process a packet before sending it to next hop [11]. A router generally looks up the host and destination IP addresses against a lookup table and either drop the packet (if there is no entry of the specific addresses) or forward it onto the next computer or the router. A router forwards the packet to the specific hardware port while taking care that it is also matches the logical port to which the packet is being forwarded.

A router cross checks the incoming packet's host and destination IP address against a look up table and routes it accordingly, it not only directs the packet to a valid destination, but also makes sure that , the IP packet does not go where it is not intended, this helps in clearing the data lines of undesired clogging and congesting . These are used at the enterprise servers, core, and access and edge networks and at the end user's location.

### 1.5.1 Packet Classification and Scheduling

In simple terms, the process through which a router, or a network processor forwards a packet onto the fly towards its logical destination with the help of host IP address, destination IP address and concerned routing table entrees is called classification [12]. Various search algorithms have been developed; initial was hash tree search which evolved into binary tree, radix tree

and so on. Various forms of hardware starting with random access memory (RAM), have been used, eventually content addressable memory (CAM) is being used to match the immediate logical address that matches the incoming packet with specific host and destination IP addresses.

Classification of packets in a router consists of two planes namely [13],

1. Control plane
2. Forwarding plane

As a packet comes in, the router has to compare the host and destination IP addresses, and other fields in the packet and then forward it. Control – plane takes care of updating the router's look up table at regular interval [14]. The reason to update the router table is that, if the router is moved from one node to another, it is automatically given a new IP address, if there is another node being added around the router in question or a node is down the router table needs to be updated, routing metrics need to be updated regularly and accordingly and control – plane takes care of that.

Forwarding – plane is where classification and also scheduling takes place, often referred to as the transport – plane, the forwarding – plane takes care of forwarding the inbound IP packets, based on pre set rules, parallel processing is carried on in high end routers, which help route several packets in parallel.

A packet scheduler prioritizes packets, checks for the attributes, such as traffic, congestion, jitter and decides which packets go onto the network from a particular computer [15]. A subset of packet scheduling can be considered as grade of service (GoS). GoS is the probability of a call being blocked beyond allowable time period this is expressed as decimal fraction.

## 1.6 Contribution of This Thesis

This thesis proposes basic architecture of the VNP, key decisions as to the placement and data path of the VNP, and the design of VNP specific Classifier – Scheduler module, which supports both classification and scheduling of IP packets. Recognizing VNP patterns within the IP packet and forwarding the data of specific packets for processing. Every network processor has to be supported with appropriate classification and scheduling functionality and this module has been designed specifically to support VNP.

The classifier – scheduler classifies IP packets based on the host and destination IP addresses and various sorting rules, and recognizes specific VNP patterns and schedules then with the help of control plane. CAM has been explored to develop this module. The proposed architecture takes advantage of CAM for a high performance realization. The classifier – scheduler module schedules in a first in first out (FIFO) fashion and also keeps the QoS parameters under consideration.

The proposed architecture has been prototyped using Xilinx ISE 9.1i and simulated through the inbuilt ISE simulator, which comes with the Xilinx webpack.

The design is achieved in parts of both structural and behavioral coding. The prototype classifier-scheduler module differentiates between normal IP packets and VNP processed and to be processed by VNP packets and splits the data and send it to the processing elements to be secured and then sews back data to the header and the hardware based scheduler, schedules the packet in a FIFO manner, this module works in the forwarding – plane of the network processor.

## 1.7 Organization of This Thesis

This thesis is organized as follows; the first chapter gives an introduction to the motivating factors of this research. The related research has been discussed in Chapter 2. An explanation of VNP and its advantages is given in Chapter 3. Chapter 4 discusses the design considerations of the classifier-scheduler module, the proposed algorithm. In Chapter 5 the prototype of the proposed architecture for the classifier-scheduler module is given. Chapter 6 discusses the prototype development and the challenges faced during the design. Chapter 7 concludes the work and gives an idea of the future work.

# CHAPTER 2

# RELATED RESEARCH

VNP is a network traffic processor its processes raw MPEG IP packets (of IPTV) and secures the digital rights through various encoding and watermarking techniques. However as with all network processors, VNP has to be supported by appropriate classification and scheduling, these are the back bone of VNP and their working is based on several constraints. There has been considerable work done in the field of IPTV, digital rights management (DRM), encoding techniques etc. This chapter gives a glimpse of the related research in this field.

## 2.1 DTV and IPTV

International telecommunication union – telecommunication standardization sector (ITU-T), focus group (FG IPTV) defined IPTV as – a multimedia service of delivering television, audio, text, graphics or data over IP based networks managed to provide the required level of QoS, quality of experience(QoE), security, interactivity and reliability [16]. This thesis presents a module in VNP which not only routes the IPTV packets, but also recognize packets from authorized users and secure the payload of the IP packets, so that only authorized users with specifically programmed set top boxes will be able to extract the MPEG streams. IPTV will be transmitted through UDP packets carrying RTP streams. request for comments (RFC's) such as RFC 3550 defines the standards to be used to transmit real time video and audio, this takes care of VoD, but IPTV aims at transmitting real time channels with minimum delay and jitter [17]. RTSP, RFC – 2326, defines the establishment and control of single or

several time synchronized streams of continuous media. As the RTP packet flows down the kernel of the operating system, it is encapsulated and sent to the physical layer, where it is classified and scheduled, and VNP picks this payload and secures it using watermarking, scrambling and encoding techniques.

## 2.2 Digital Rights Management (DRM)

Broadcasting digital data onto an unsecured network is a potential threat to the data and makes it vulnerable to piracy and theft, and in order to protect the right, the data is secured through use of various algorithms, and this process of securing data in order to keep it authentic is called DRM. DRM dates back to 1991, when Fiat and Naor considered the question “is it possible to have a key management scheme with revocation, but without the participating parties having a two-way handshake?” [18]. DRM has evolved to be an access control technique used to protect copyright of digital data. Various algorithms have been used to facilitate DRM such as matrix based encryptions such as, using content protection for prerecorded media (CPRM) key matrix , tree based encryption schemes such as logical key hierarchy were also used. The most popular way of securing DRM is through watermarking.

## 2.3 Network Processors

Network processors add specific processing engines to achieve desired processing of the IP packet. Due to the growth in the complex functionalities provided by the network and the data rates that are being supported, researches have constantly tried to improve the performance of network processors. Key design issues such as placement of processing engines, the data rates that are

to be supported have been discussed by Wolf [19]. A network processor as defined by Shah, in his thesis is “an application specific integrated processor for the networking application domain”. He has discussed the flexibility, performance, power, and cost per unit cost to develop and cost to integrate various network processors. He discusses various network processors such as - nP7xxx series by applied micro circuits, MSP 5000 by BRECIS Communications, PXF Toaster2 by Cisco and so on [20]. On set of real time audio and video transmission has pushed the researchers further to achieve high speed designs for network processors, one such work is given in [21], which is a combined work of university of California and STMicroelectronic Inc. This paper proposes a multi distributed memory system to efficiently breakdown the high speed, memory intensive tasks and achieve them in parallel, similar work of distributing the load has been discussed in [22][23]. The system on chip based design for network processors is discussed in [24], [25].

## 2.4 Packet Classification for Network Processors

When a packet enters the network processor, the processor will first classify it, check for the validity of the IP packet, by checking the host and destination IP addresses, and classifies them based on several rules that are stored, then forward them to the application plane and later check for the availability of the port and forward the IP packets to be scheduled to the respective hardware ports.

There are several issues that a classifier takes care of:

1. Extracting header information, such as destination IP or Time-to-Live – TTL.
2. Check the Data link data and extract the packet
3. Analyzing other fields of the packet
4. processing the packet through the ingress and egress interfaces

For example, 10.0.0.1 and 10.20.0.10 are the host and destination IP addresses respectively of an incoming IP packet, the classifier compares the bits and cross checks if these are updated in its look up table, if yes it proceeds to check the other fields in the packet such as network layer addresses, the number of hops, and so on. Now how this comparison is done has lead a lot of research. As a packet usually will have to go through a number of routers and gateways before it reaches the intended destination, a poor classification algorithm, might slow the traffic drastically.

The existing packet classification algorithms can be categorized in four groups –

1. Basic data structures
2. Geometry based
3. Heuristic based and
4. Hardware based.

Basic data structures search the incoming bit stream either using linear search, or caching or set-pruning tiers, geometry based search, searched either using area based quad tree, or grid of tiers, or fast inverted segment tree

mechanisms. Heuristic uses recursive flow algorithm or tuple-space search. However this research focuses on hardware based searches – these include mechanisms such as CAM, or bitmap intersection.

The pros and cons of some of the basic classification algorithms are tabulated under table 2.1. *N* represents the number of rules, *W* is the width of dimensions, *d* is the number of dimensions being used, *l* represents the levels of tree, *T* is the number of tuples.

TABLE 2.1 Comparison of existing classification algorithms- hardware and software [26],[27],[28], [29],[30].

| Algorithm | Time | Storage | Updates | Observations |
|---|---|---|---|---|
| CAM - Hardware | *1* | *N* | *1* | Simple, Fast, but takes up a lot of space and is Costly, incremental updates are supported and updates in parallel |
| Bit vector - Hardware | *dW + N/memwidth* | *dN2* | --- | No incremental updates; works well for multiple dimension and a small number of rules |
| Recursive flow Classification | *d* | *Nd* | --- | Not suitable for large sets of rules (> 6000); pre-processing and large storage space. 10Gbps line rates in hardware and 2.5Gbps rates in software. |
| Hierarchical Intelligent Cutting | *d* | *Nd* | --- | Parameters can be tuned to trade-off query time against storage requirements. |
| Tuple-space search | *T* | *N* | *1* | Supports multiple dimensions if the number of tuples are small. Only supports prefixes; generic rules increase storage complexity. |
| Grid-of-tires | *W d-1* | *NdW* | *NdW* | Rebuild for each update; could be used for last 2 dimensions of a multi-dimensional hierarchical trie. |
| Area based Quad Tree | *aW* | *NW* | $aSqrt^{a}(N)$ | a is a tunable integer parameter |
| Fast Inverted Segment | *(l + 1) W* | *l x N1 + 1/l* | -- | Tree must be recomputed on update |

| | | | | |
|---|---|---|---|---|
| Linear Search | *N* | *N* | *1* | Simple, poor scaling |
| Hierarchical trie | *Wd* | *NdW* | *d2W* | Maximizes the number of rules in each leaf, introduces a parameter which guides in partitioning process |
| Set-pruning trie Cross-producting | *dW* | *Nd* | *Nd* | Fast and Costly, updating not supported. |

VNP needs to have a higher level of control – plane functionality as the control – plane needs to take care of the IP stack and also the data scheduling to all the processing modules.

## 2.5 Packet Scheduling for Network Processors

After the packet is processed by a set of internet stack processing units, and once the packets are classified, the scheduler forwards them to the specific ports, based on the QoS factors [31], [32]. Handling arbitrary header lengths requires constant updates from the controller. As the packets come in, they have to scheduled to the hardware ports that match their logical ports, there are several mechanisms, developed in packet scheduling [33], [34] such as

1. First in first out (FIFO)
2. Random early detection (RED)
3. Deficit round robin (DRR)
4. Earliest deadline first ( EDF)
5. Best effort packet scheduling (BPA)

6. Round Robin (RR)
7. Priority queuing - variations
8. Virtual clock (VC)
9. Self clocked fair queuing (SCFQ)

Scheduling decisions are based on fairness, immunity from other flows, guarantee of the packet reaching the correct path. Comparison of some of the algorithms is given in table 2.2.

TABLE 2.2 Comparison of some basic scheduling algorithms [35], [36], [37], [38],[39].

| Algorithm | Functionality | Advantages | Disadvantages |
|---|---|---|---|
| First in first out (FIFO) | Packets are queued based on first come first serve basis | No congestion | Streams with low priority can steal complete bandwidth at a time, no immunity from other flows, unfair to packets with priority |
| Round Robin | Scheduler picks packet in a round robin fashion, 0 to n-1 | No congestion | No guarantee of delay or bandwidth. Time wasted servicing empty queues. Packet size is not taken into consideration |
| Priority Queuing | Packets set in priority queue are serviced first | Good throughput, low delay and good utilization of bandwidth | Low priority queues may not be serviced for long. |
| Windowed priority queuing | Packets set in priority queues are serviced but service is limited | Low priority queues can also be processed | Delay increases on high priority queuing. |
| Virtual clock | Certain bandwidth is guaranteed per flow | Every stream of data gets a part of the bandwidth | When a queue does not have data to stream, bandwidth is wasted |
| Worst case Weighted fair queuing | Packets are scheduled based on earliest finish time | Fairness is maintained irrespective of packet size or profile | Highly computational |
| Deficit round robin | Variation of deficit weighted round robin | Can handle packets without computing the mean size | Highly computational and complex in implementation |

## 2.6 Content Addressable Memory (CAM)

There are several packet classification and look up algorithms that are embedded into routers. CAM is one of the fastest classification algorithms; it basic advantage is that it scales well in the time domain, it also has a disadvantage, it does not scale well in space domain.

Amongst the latest algorithms that is being used for classification and look up [40], hypercuts and distributed cross producting of field labels (DCFL) are popular, wherein these techniques utilize redundancies in filter sets in addition to employing specific search engines for specific fields in the strings, these do give higher throughput and scalability, but they offer higher latency and resource utilization. Variations of CAM seem to offer a better solution in terms of performance, efficiency and scalability. CAM is chosen as a basic classification and also the scheduling technique for the proposed Classifier – Scheduler module.

### 2.6.1 Variations in CAM

CAM is an associative memory technique; it sorts out fixed length binary words. This is helpful where the packet header length is fixed and strictly hierarchical, however is the packet length varies CAM cannot classify effectively. To achieve parallelism (which in turn allows us to process higher amount of data, and also to classify data of variable length) ternary content addressable memory (TCAM), is used. TCAM achieves this functionality by incorporating a ‘don’t care’ bit in its functionality [41]. Label encoded content addressable memory (LECAM) is also a high performance classification technique which uses parallel search

engines. Each field of the string of data matched in parallel using various sub engines. A controller which manages a count value for every match and which triggers additions or deletions to the search engine data structures in the control – plane is required.

## 2.7 System on Chip (SoC) Technology

SoC is an embedded system which integrates functionalities and components of a system onto a single chip. The system on chip can be an all digital or analog or a mixed signal system. A SoC typically consists of hardware components and software and a controller which takes care of the effective inter communication of all the components [42],[43],[44].

SoC typically consists of memory blocks, microcontrollers, voltage regulators, timing sources, external interfaces digital signal processing units. This research aims on emulating the SoC for VNP and emulating SoC usually implies the mapping of functionalities that have been developed onto an FPGA chip. The design flow, emulation, placement and routing and verification are the basic steps in achieving the SoC design.

## 2.8 Securing Video in IPTV

There have been innumerable ways to encrypt data, be it transmitting over air or through a wired media, or to secure digital data , some of these techniques are combined together to secure IPTV so that only the authorized user’s would be able to view television data, conventional ways of securing data include:

1. Compression and Encryption
2. Watermarking
3. Scrambling

### 2.8.1 Compression and Encryption

Compression is an encoding technique, where less number of bits are used to transmit the original data. Compression can be lossy or loss less. A specific stream of bits is represented by another stream of bits which are less in count that the original stream. In lossy compression the quality of the extracted stream of data might decrease and in loss less the quality of the extracted stream remains intact [45]. Compression and encryption go hand in hand, where in, in encryption a set of bits are represented by another set of bits and transmitted and the receiver has a key for the encrypted sequence and simply replaces the encoded stream with original stream.

### 2.8.2 Watermarking

Watermarking is one of the copyright protection techniques, where in the simplest way or watermarking is placing a visible image over the image or video which is to be watermarked [46]. Watermarking can be extended to invisible watermarking, where if a copyright issue arises for a particular image or a video only the concerned party can extract the hidden data and prove its authentication.

### 2.8.3 Scrambling

Scrambling is one of the cryptographic techniques [47]. A given stream of data is scrambled using either a 32 bit or 64 bit key, and the message is passed on, once the message reaches the destination, the user who has the key will be able to decrypt the information and get the original data.

## 2.9 Field Programmable Gate Array (FPGA) Technology

Custom logic cells are pre programmed by the manufacturer and give us limited functionality, FPGA, however is a logic cell which contain 64 to 10000 identical logic gates FPGA's are slower than ASIC's, and cannot handle complex designs, but are idle to check run programs designed by individual designers and check their functionality [48], [49].

A typical FPGA is an array of configurable logic blocks – CLB's and routing channels, with input and output interfaces, when a program is dumped onto an FPGA, the CLBs are fused according to the program functionality. Once a hardware descriptive language - HDL program is written it can be run on the FPGA with the help of software translators which place route and translate the design so that the logic cells on FPGA can be fused accordingly. The design automation tools are equipped with complex functions and libraries, which simplify and speed up the design process [50]. A single FPGA can be programmed again and again to various HDL programs or schematics.

The common hardware descriptive languages are very high scale integrate circuit hardware descriptive language – VHDL and Verilog [51], [52]. When a program is written it is simulated at various stages, with the help of its

test bench and its register transfer level – RTL description, and with the help automation and synthesis tools which map the net list which in turn help translate the program to gate level description and once the simulation is checked it is laid onto an FPGA board.

# CHAPTER 3

# VIDEO NETWORK PROCESSOR

Video network processor belongs to the family of network processors, the thesis aims to develop this processor into a full scale digital rights management box, which can be placed any where in the network and it will tune it self to pick up data packets and check for patterns in the data streams, recognize IPTV packets from the content providers, control and request packets from the user and process them for authenticity and digital rights management. This chapter gives an over view of what video network processor is, its basic architecture, working, its placement considerations, concerns and proposed remedies.

## 3.1 Why Do We Need the Video Network Processor (VNP)?

For real-time copyrighted video broadcasting through IPTV the proposed VNP processes raw MPEG IP packets from the content provider, checks for authenticity, secures the rights of the content provider digitally – through watermarking, and further secures it by encryption and scrambling, VNP will be deployed not only at the content provider but also throughout the Internet network, that is the core, edge and access networks, which implies that VNP will have different levels of service parameters – based on where it is deployed [53].

Deployment of VNP is discussed in Fig 1. The user register's himself with a content provider, to be able to view channels and videos of his choice. The content provider receives digital video from various channels on a constant basis. To authenticate the users who are receiving the data and to secure the digital

video that is being sent across the Internet the content provider uses an enterprise server, which encapsulates this raw MPEG data in IP packets and sends them to VNP, classifies the packets, checks for digital rights and then throws them onto the internet.

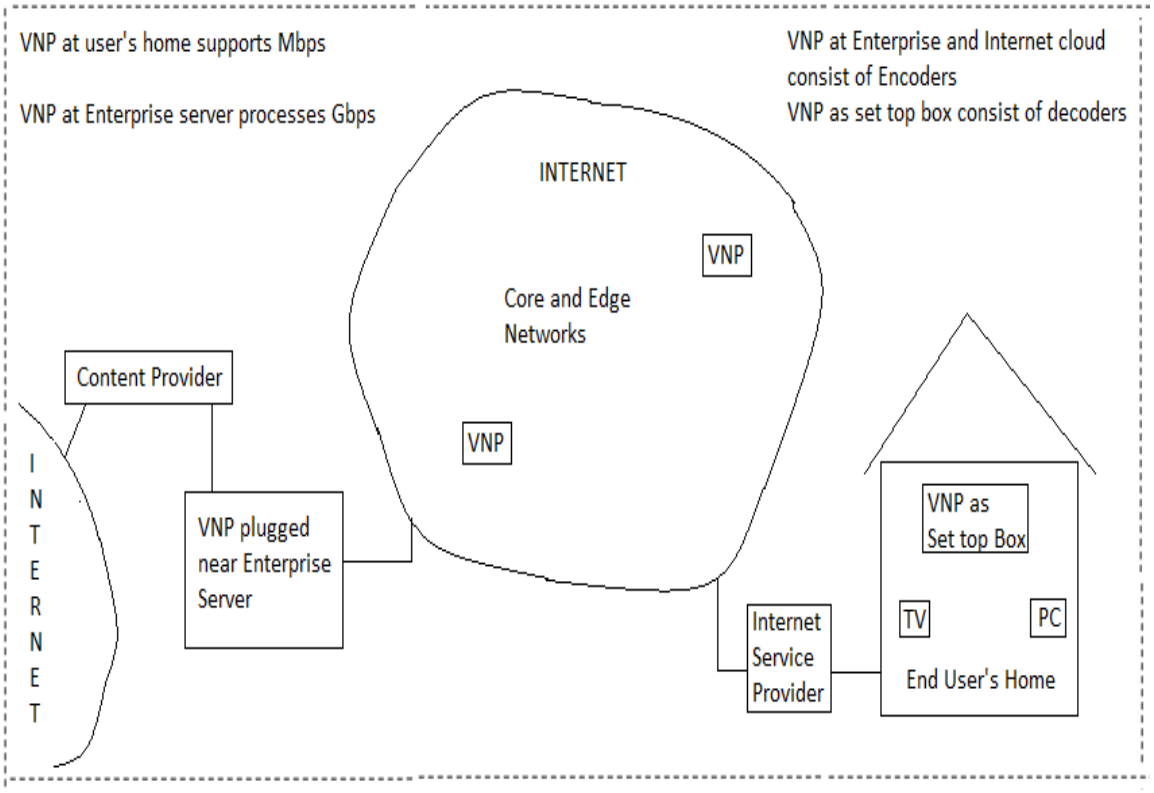


FIGURE 3.1 Deployment of video network processor – in the Internet.

VNP will be placed in core, edge, and access networks, so that it can keep checking for authenticity of VNP packets throughout the network. In figure 3.1, SBOX represents the set-top box which is a user end VNP. For example when a user requests video (video on demand,) it is broadcasted through the network by the VNP, that is there is no single cast in this scenario. Once the packets reach

the user, the set top box splits internet data and digital TV into two different streams so that the user can view them separately.

## 3.2 Placement and Requirements of VNP in the Context of IPTV

The specifications of VNP that would be placed at the content provider will differ from the VNP that is placed in the middle of the core network and the VNP that is acting as the set top box for the end user. The decisions are based on the incoming traffic, outgoing traffic, data being processed per second and the number of ports being supported and if the processing elements in the VNP would be securing the IP packets or decoding the secured packets, the maximum transferable unit – MTU, makes a lot of difference, another important point would be if the VNP being used can completely replace the enterprise processor or be hooked into the existing network processors [54].

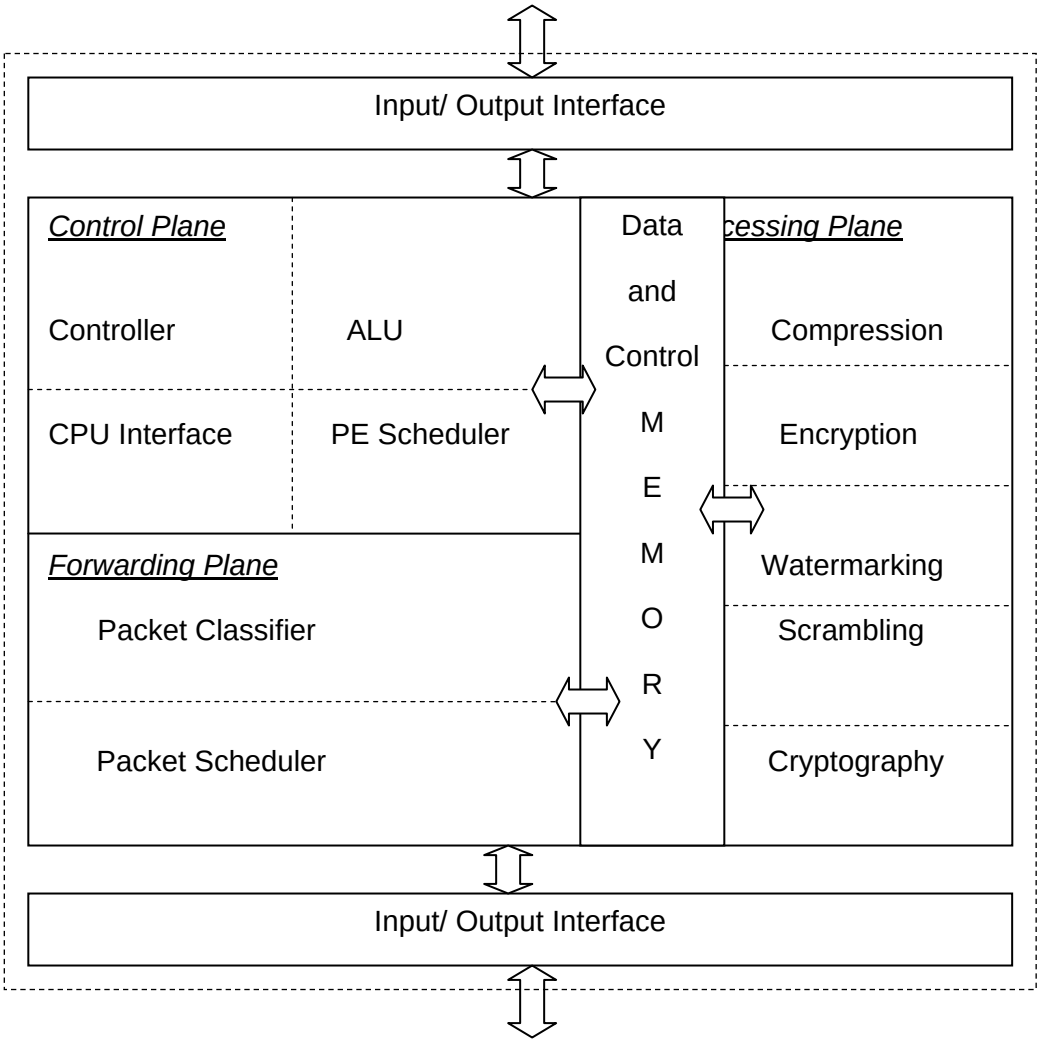


FIGURE 3.2 Video network processor – internal architecture.

Description of a proposed architecture of VNP is given in figure 3.2. In VNP as soon as the packet arrives from the content provider, it is check for authenticity by the classifier, and then it cross checks the destination against its look up table, the packet is stripped of its header and the data is forwarded to processing modules , starting with the compression module (if required), then it is forwarded to the watermarking module and then encrypted and scrambled , after this the scrambled data is encapsulated in an IP packet again and the original

header information is attached to it , and sent across the internet. The engines in the VNP are always monitored by the control memory and the central processing unit interfaces which are in turn switched on and off by the voltage scheduler and power scheduler, thereby reducing power dissipation. The voltage scheduler can be implemented as a separate engine controlling all other engines but can also be incorporated within specific engines. Since the packets being processed here are IP packets, variable length strings are to be considered.

## 3.3 Network Aspect of the VNP Classifier - Scheduler

As an IP packet is created it can either be an ICMP, UDP, TCP, HTTP [55] or any other data or control packet, it trickles down the Internet stack, and the data in the packet gets encapsulated into one header after another until it is encapsulated into the physical layer packet and routed out, this implies that the VNP will need to mimic the Kernel to strip data out of the various headers and this would need the control plane of the router to be able to classify headers of various sizes. The classifier will keep getting and giving back updates from and to the controller and this demands that the classifier be able to classify IP packets with variable header size. The functionality of Classifier – Scheduler module depends immensely on the packet structure.

### 3.3.1 An IP Packet

Internet packet is a datagram through which computers talk to each other, and the network through which they pass is packet switched, and communication is established either through TCP session or through UDP packets. TCP session is initiated through a handshake and if there is packet loss, the intermediate or

destination computers send out a request to resend packets, if data packets are lost, after retransmission of the lost packets, the kernel can put the formerly received packets and the new packets in the original sequence and send data to the application. However this is time taking, and if real time data is to be transmitted, or video or voice is to sent, because if a packet is lost, even if it retransmitted, there would be no point keeping all the packets waiting for one lost packet as video and audio, are real time and certain sessions also require speedy real time data rather than application waiting for every single packet. In such case UDP is used, which is also known as unreliable protocol, it streams packets in a much faster way and is ideal for real time data, where in packets are transmitted and even if a packet is lost it does not affect the end user, the end user is only affected if and only if a chunk of packets are lost, this causes – pause, jitter, loss of video or audio data for a certain time. UDP packet format is mimicked for the classification. UDP supports broadcast and multicast. In the Internet protocol, once the transport layer attaches the UDP header, it is again trickled down to the network and physical layer to be thrown onto the fly. Session Description Protocol – SDP allows us to predict the upcoming packets statistics, so that we don't have to split every packet and analyze it from the first structure, we can compare certain structures and predict the upcoming packets, and the controller in VNP should be capable of recognizing all the protocols and packet types.

As soon as the packet hits VNP, it will recognize the physical layer's header, a typical IP packet is as shown in figure 3.3. The payload of this packet can either contain, TCP, UDP, ICMP or any other higher layer protocol packet.

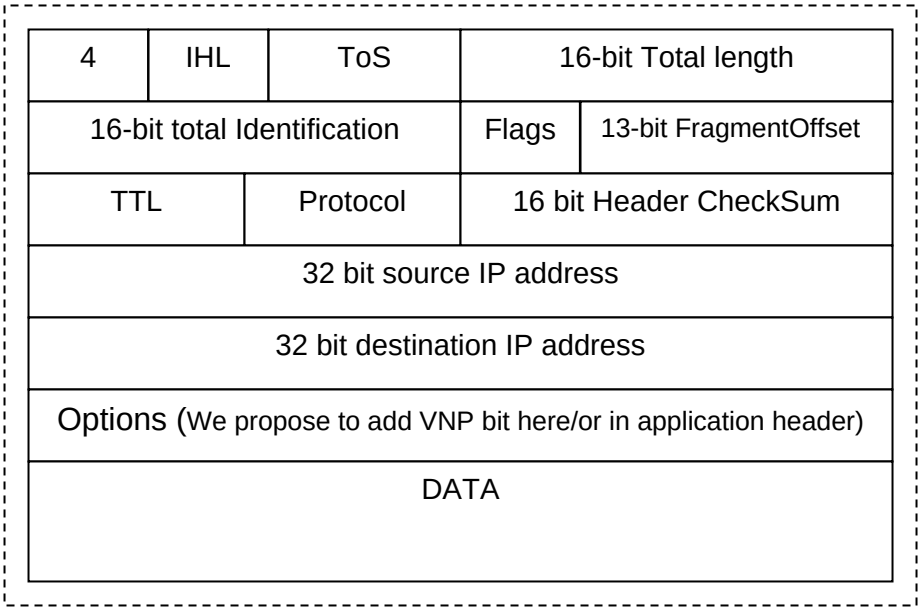


FIGURE 3.3 Internet protocol physical layer packet.

IP packets do not have a fixed payload length, as the MTU of ethernet is different from that of Core network and packet fragmentation occurs when ever needed according to the traffic being allowed on the fly, and VNP will need to identify that and classify and schedule accordingly. The functionality achieved so far in the classifier works on the forwarding – plane and control will need to mimic as mentioned above the functionality of the operating system.

### 3.3.2 IP Stack

The Internet protocol stack, on which the Internet works on is divided into five layers [56], [57].

1. Application layer
2. Transport layer
3. Network/Internet Layer
4. Data Link Layer
5. Physical Layer

A brief explanation of the working of the Internet protocol stack is given below, which in turns explains the forwarding and control layer of the VNP. The IP stack is implemented by the operating system's Kernel, it can be better understood with the help of figure 3.4

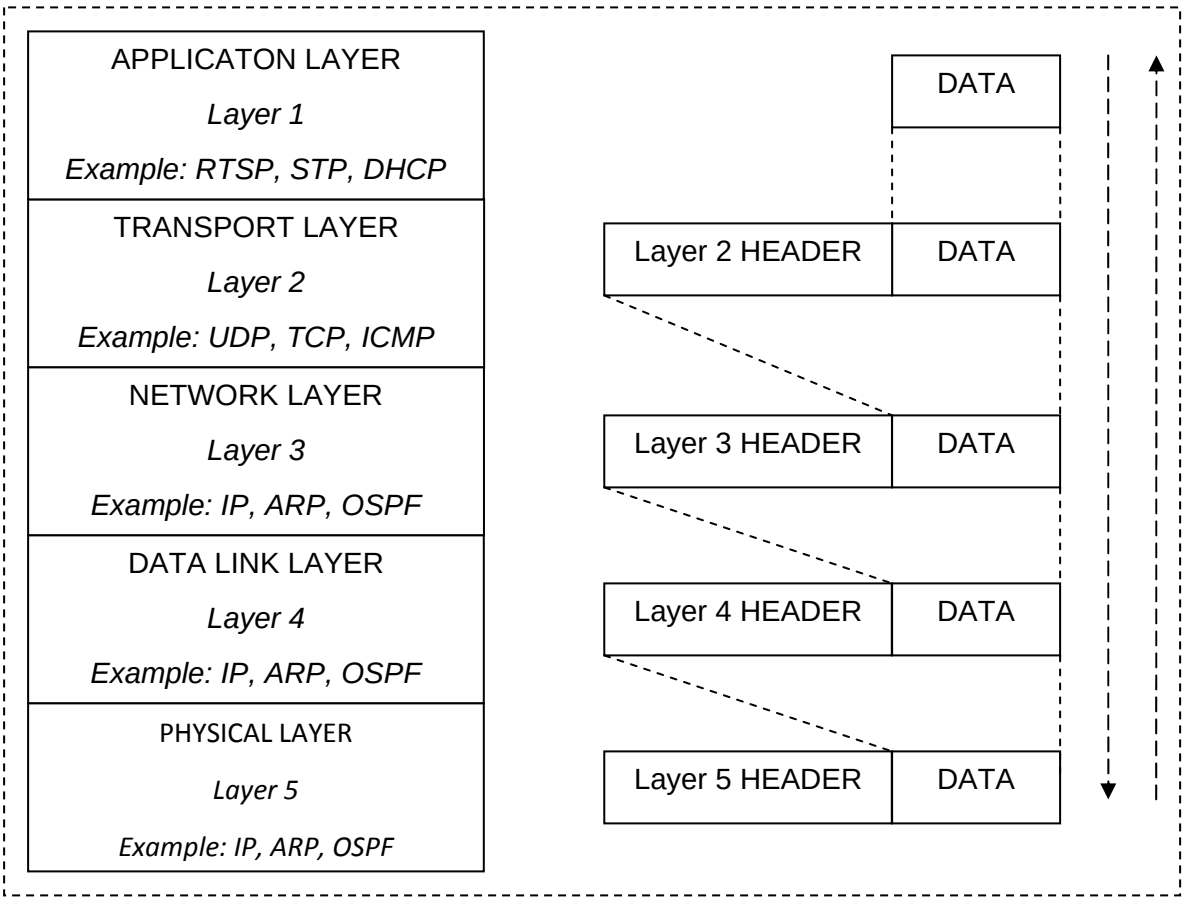


FIGURE 3.4 Internet protocol stack.

Application layer is at the top of the stack, where in takes original data from the application that is running on the system, if a user is browsing a webpage the application layer hands over the actual browser information, takes the login information and such, to watch a video online, the application layer must identify it as video and not as simple string of bits. Application layer protocols include – RTP, RTSP etc.

This data then trickles down the Internet protocol stack and reaches the transport layer, once the transport layer gets this data, it encapsulates this data, in its header which specifies a specific protocol appropriate to the data

type. For example transport layer gives UDP to RTSP data, and the data is not loss or time sensitive, it assigns the TCP header type to it. If the data is a control signal it assigns ICMP to it and so on. This transport layer packet now flows down to network/internet protocol layer where it is encapsulated, (the header of earlier layers is considered as payload) with this layer's header such as IP or address resolution protocol (ARP) header.

Then the packet then moves onto data link layer, where the type of transportation is decided and that particular header is added, like, Wi – Fi, WiMAX, PPP so on. Then the packet finally reaches, the physical layer and it is attached with a header and footer, this gives the physical layer data and it is forwarded onto the fly. Once the packet reaches the destination, first the physical layer header is stripped, and then the data link layer strips its header, and network and transport remove their headers and application layer is presented with its data.

VNP works on three layers – physical, data link and network. VNP will not be supported by any server or a PC, but it is a standalone processor, which needs to mimic, a server or at the least a PC's kernel to be able to strip the packets of its header, based on the type of header, as headers of various layers have different size, physical layer header is of fixed length, but the data can be encapsulated in variety of combinations of protocols, which makes it extremely difficult for a normal processing element to predict the exact header load in a particular packet, unless it is able to mimic the functionality of the IP stack. This

functionality will be taken care by the control – plane, and forwarding – plane, where the classifier – scheduler module is located, utilizes this functionality, the module is constantly updated by the control – plane and utilize the information in splitting the packet and forwarding it for processing.

## 3.4 Why Choose CAM for VNP?

An interesting variation of content addressable memory is the TCAM introduces a ‘don’t care’ bit, which allows us to check for the highest priority match [58]. That is, once an incoming bit stream hits the TCAM for classification module, the incoming bit stream is shifted one at a time, to find a match, it may have multiple match strings, the more number of hits in a group, the highest priority is given to that particular string to be belonging to a particular subgroup.

The VNP that is under consideration is placed near the content provider and the classification in VNP at this point of its placement purely concentrates on the incoming and outgoing IP address, as far as the forwarding plane is concerned. It is however desirable that CAM is able to classify a header of various lengths, rather than one specified header field and hence CAM is the preferred classification algorithm. The advantages of CAM over TCAM in this scenario is that it is more simpler and faster and does not introduce latency as TCAM (due to its ‘don’t care’ bit, multiple match criteria), the introduction of TCAM is not required in this particular case, it is however inevitable in the control plane, the control plane will have to classify data strings based on the protocol and data that it is carrying.

To minimize the number of blocks used in VNP, a single module must be able to work in different modes, that is both CAM and TCAM as suggested in [59] .

# CHAPTER 4

# THE PROPOSED COMBINED CLASSIFICATION AND SCHEDULING ALGORITHM

Classification and scheduling work as the back bone of the VNP, and because classification and scheduling are part of forwarding – plane functionality and they have to forward data to the processing plane based on the control plane information, certain constraints and specifications have been set on the design of the algorithm. This chapter deals with the problem definition, and the proposed algorithm for the combined classification and scheduling algorithm.

## 4.1 Packet Classification in VNP – Problem Definition

Below described is the forwarding – plane of the processor in this classifier – scheduler design. As describe in the earlier section the VNP will have an inbuilt processor which will have to collect data from the nodes on the network so that the routing tables can be updated, however the forwarding of the packets is the aim.

1. The packet classifier of VNP should cross check the host and destination IP address of the incoming packet and forward it if valid or drop it if invalid.
2. VNP should not only do this, but it has to be able to differentiate between a VNP packet, packet that has to be made a VNP packet and the normal internet traffic packet.

3. The classifier needs to mark the packet as a VNP packet after the packet has arrived from the content provider and been secured through the processing elements in the VNP.
4. The classifier needs to split the header and the payload so that the payload can be processed and then attached back to the header.
5. Forward the secure datagram to scheduler to be scheduled.

However, there are concerns regarding the payload size of the IP packet.

1. The packet payload size, MTU varies from one network to another, that is the payload size in Ethernet varies from, the payload size in the core network, the MTU size and this differentiation is usually taken care by the operating system kernel.
2. Different protocols have different headers and MTU's, and the Classifier, must be able to recognize and handle various MTU's.
3. As the data trickles down the IP network stack, it is gradually encapsulated in one header after another and every protocol will have a different header size, RTSP, which is used to transmit MPEG streams has a different header as opposed to SDP.
4. The classifier must be able to process packets with of variable header sizes.

## 4.2 Packet Scheduling in VNP – Problem Definition

IP scheduler are implemented in software, where in as and when an IP packet is generated, the processor picks it up, checks for the traffic parameters such as – traffic, congestion and simply schedules as to which packet can move onto the outgoing stream based on its destination IP address. However design of a scheduler to be embedded in VNP will have to take input from the controller, for the traffic and congestion and schedule the packets. Packets are scheduled in FIFO fashion, if the controller gives an update of the traffic; the scheduler needs to push out the packet to the end of the queue.

1. The scheduler for VNP will have to broadcast, multicast and single cast based on the placement of the VNP.
2. The scheduler will have to take into consideration the nature of IP packet, if it is a normal IP packet or a VNP packet.
3. It needs to schedule in a FIFO fashion and make sure that when there is congestion the packet at that point of time moves to the end of the queue.

Flag bit is proposed, which can help the scheduler to be able to differentiate between VNP and non VNP packets, and this flag will also function as an internal mode flag which will help identify the type of casting and QoS check at regular intervals so that the scheduler does not send the packet on a congested fly.

## 4.3 Packet Classification and Scheduling Algorithm

Packet classifier is the first element, the packet hits, after entering the input/output – I/O interface. The data bits are picked up and classified based on Content addressable memory classification, packet is then split as header and data, data is forwarded to processing elements and after the data is processed it is appended to the header and then match of the output port is taken based on RAM search and the packet is thrown out – after the port QoS is checked, it is either single cast, multicast or broadcast based on the control plane information.

The algorithm proposed in this thesis processes one packet at a time. If the control – plane of the VNP is developed it can dwell and check for session description protocol and the classifier will not have to go through the data bit after bit, but processes the chunks of data as they come in, as data is usually sent in bursts rather than a continuous stream. Pseudo code of the proposed algorithm is given in figure 4.1.

FIGURE 4.1 Algorithm 1_pseudo code for the proposed combined classification and scheduling algorithm.

```
begin process  -- String = Data + Header
if clk = 'high'
  {     load string into data shift register;
        for  i in 0 to header-1 loop
                { run data through CAM
                        output the corresponding addr's of the matched string
                              if 'Match' is low
                                 { drop the packet }
                              else if 'Match' is high
                               {
                                if VNP = 'high'
                               { split packet;
                                  Write data to PE cache;
                                   update 'hops' in header and store in cache;
                                          after data is processed – concatenate;
                                   forward packet to scheduling;
                                 }
                                else
                                 {  update 'hops' in packet
                                     forward packet to scheduling;
                                 }
                                  }
        end loop
for i in 0 to addrs loop
                { compare the CAM given address against RAM look up table;
                  output the corresponding port address;
                }
        end loop
for i in 0 to String loop
```

```
                { if QoS is high
                        {   check mode;
                            forward in a first in first our fashion;
                        }
                  else
                        {   wait for specified 'network up time'
                            run through forwarding to port again;
                        }
end loop
 }

else if clk = 'low'

wait for clk to be high

end process;
```

As soon as the data comes in it is loaded into the Input buffers, 'read' mode of the CAM is enabled. CAM then starts reading the data from the buffers in little endian fashion. After the data is run through CAM if there is a valid match in the look up table, CAM writes the location of the data, that is the address of the matched data is given out and 'match' signal is high. If there is no match of the data run through CAM, a garbage address is given out and 'match is set to low. The classifier further checks for the match signal and if 'match is low, the incoming packet is dropped and if 'match' is high, the data is split from the header. Header is stored in cache and data is forwarded to the processing elements represented here as PE. At the same time in parallel, the address that is given out by CAM is matched against RAM look up table for scheduling. RAM

gives the port number to which the packet needs to be forwarded. Once the data is processed it is sewed back to the header and it is forwarded to the appropriate port based on the output of CAM.

A flow of events in the proposed algorithm is given in given in figure 4.2 and figure 4.3.

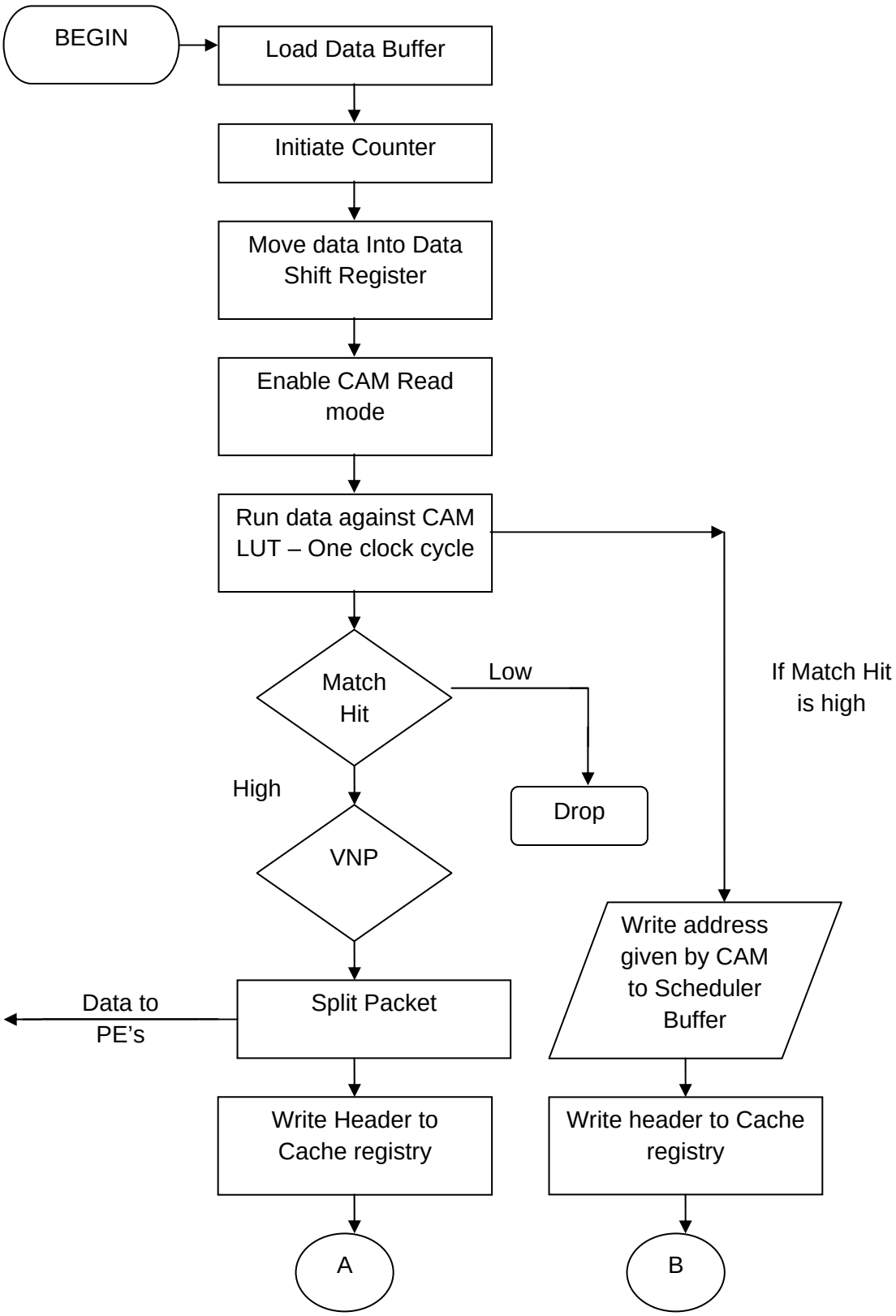


FIGURE 4.2 Flow chart of combined classification and scheduling algorithm (cont. in 4.3).

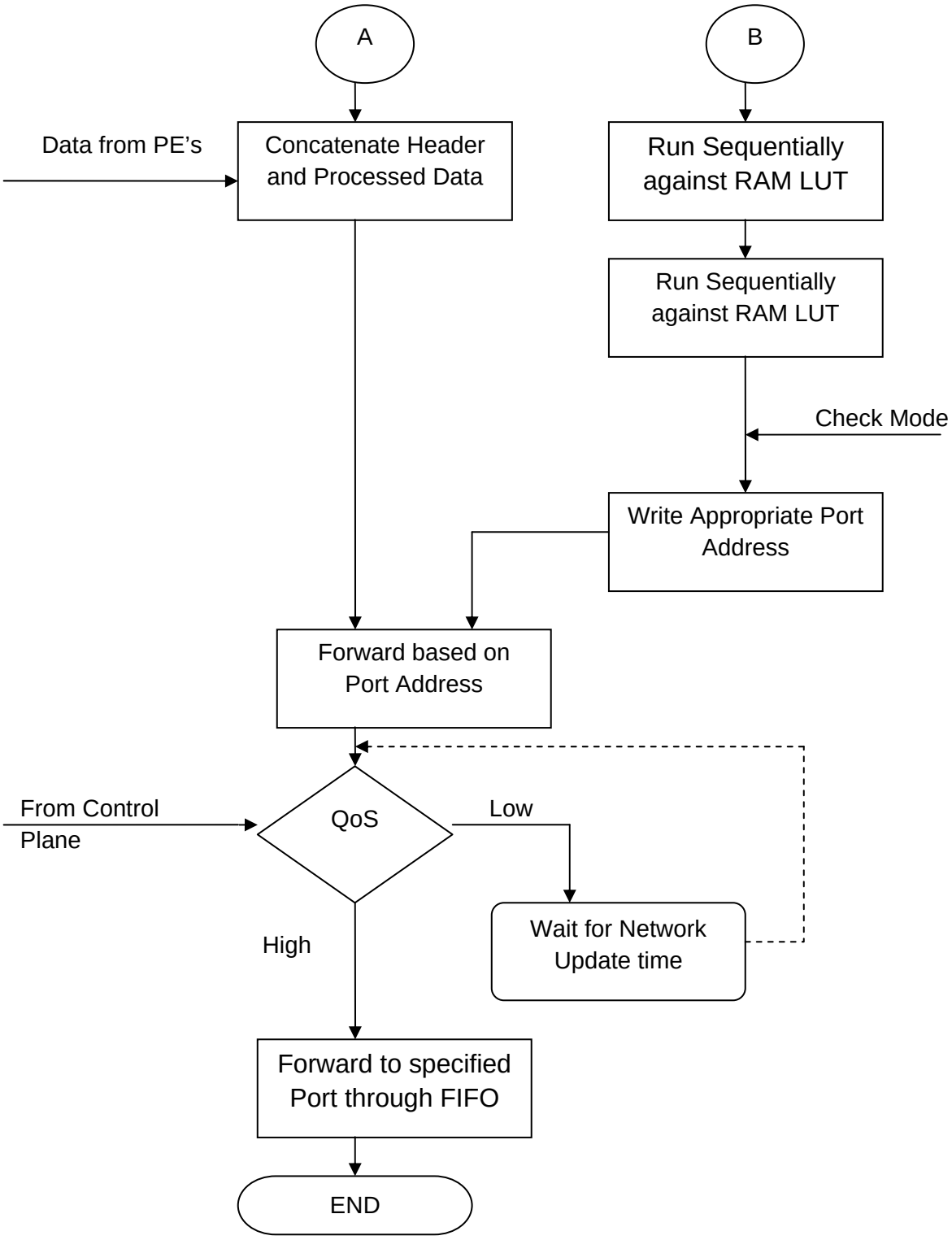


FIGURE 4.3 Flow chart of combined classification and scheduling algorithm (cont. from 4.2).

### 4.3.1 CAM and RAM Inter Working Algorithm for Combined Classifier and Scheduler

CAM uses data search words entirely of 1's and 0's [60].In CAM classification is done by dividing the bits in words. Typically 8 bits are considered as one word. And the number of words that can be searched is called as the depth of CAM. CAM can classify several words in parallel; this is what makes it fast compared to other classification techniques. As the data comes in it is split into words and each word is sent through individual comparison blocks and compared to pre stored data. For example, consider a CAM of 32 words, that is it can classify 32*8 bits; all the 8 bit comparison blocks are connected in parallel.

A better understanding of CAM can be achieved by comparing it with random access memory (RAM). The classifier will have pre stored strings, which can be rule sets, or in our case host and destination IP addresses, and flag bits, which are stored in a table which is referred to as the look up table, as the bits flow in, the classifier will compare the incoming bits to the bits stored in the look up table [61]. If there is a match the corresponding match signal is given out, if there is no match garbage value is given as out put. Where as in a RAM, data is a location is read by an address, but CAM searches for match and there is a match, address of the array is given as the output [62]. CAM does not use address lines in its read mode and hence its memory can be easily increased. CAM can compare a string in one clock cycle due to this functionality.

Basic CAM mechanism is simply to compare 2 bits at a time its out put and sent through an and gate whose other input is the comparison of two more

bits in the same block, the result of this ‘and’ gate is sent through multiplexer and when signal is set to high, it gives out the result, several blocks work in this fashion in parallel and CAM can classify 8 bits or 8 bytes or 32 bytes based on this principle.

Pseudo code for the CAM and RAM lookup table working is given under algorithm 2 and flowchart in figure 4.4.

FIGURE 4.4 Algorithm 2_pseudo code for CAM and RAM inter working for combined classification and scheduling.

```
begin process
if clk = 'high'
  { initiate counter;
        for i in 0 to numberofwords-1 loop
          { read 2 bits per comparator;
            2 comparators per block;
            one word per block;
            perform 'and' on the output of comparators in block – output is S;
            if S = '0'
                  Multiplexer Output = '0';
            else if S = '1'
                  multiplexer output = '1';
            register output of all blocks;
            compute 'Match Hit'
    }
    if 'Match Hit' = '1'
    for i in 0 to addrs-1 loop
    { write output address to Scheduler ;
      compare each bit against the RAM lookup table
---RAM takes less space but require a lot of clock cycles, but as the
 --number of ports are very less, RAM is easier to implement;
        update the counter and write to forwarding block connected to FIFO;
            }
if clk = 'low'
wait for clk = 'high'
end process;
```

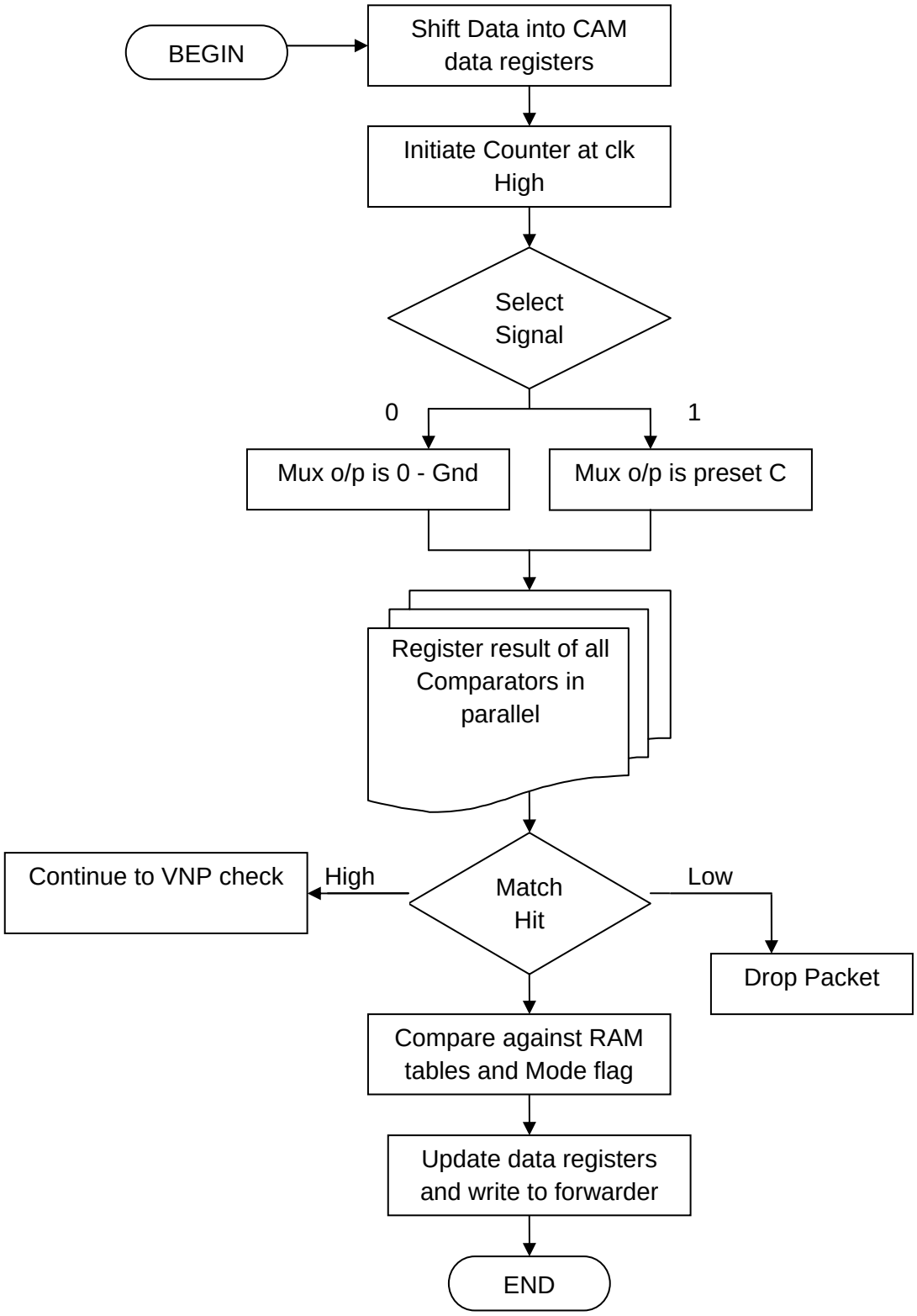


FIGURE 4.5 Inter working of CAM and RAM.

Data is read and sent as an input to buffers, which enable CAM, initiate the counter of CAM and start reading data into the shift registers, read 4 bits per comparator block, read all comparators in parallel. If the multiplexers within the comparators get the select signal as 0 the output is given as 0 or it is set to the carry in signal. Once CAM gives the address on its output line, it is automatically loaded into the RAM buffer, and RAM reads and compares the bits sequentially and gives the appropriate port number at the output. This algorithm, works in a hierarchical fashion, taking care of the forwarding – plane functionality of the VNP. The forwarding – plane would require constant updates from control – plane to function properly.

# CHAPTER 5

# ARCHITECTURE OF THE CLASSIFIER-SCHEDULER MODULE

This chapter discusses proposed architecture of the classifier – scheduler module. While designing the architecture considerations such as the buffer size, depend on the traffic that the VNP will handle based on its placement. Traffic can go up to *Gbps*, data shift registers of CAM module will need to shift IP packet string length at a time, reading 8 bit word per shift, and give out the address of the data stored in the Look up table.

## 5.1 Block Level Description of Classifier – Scheduler Module

As proposed in the VNP architecture, VNP can be divided into control, forwarding and processing planes. The classifier – scheduler falls under the control and forwarding – planes of the VNP and helps start the functionality of the processing plane. A high level schematic of classifier – scheduler module is given in figure 5.1

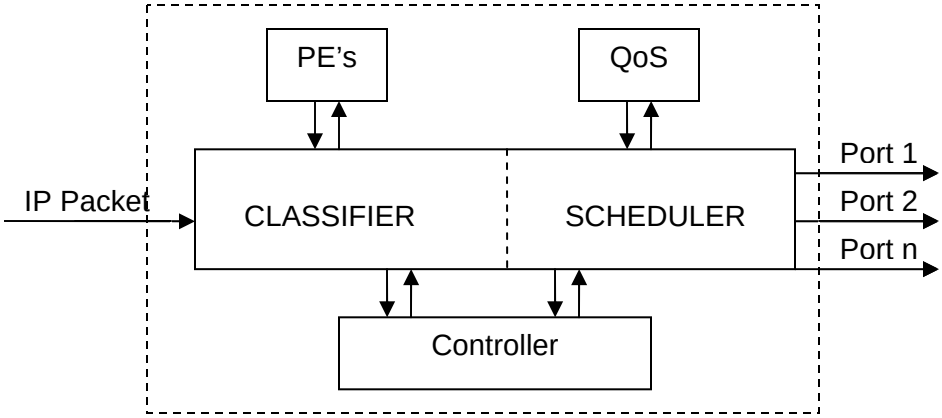


FIGURE 5.1 Block level description of proposed combined classifier-scheduler.

The classifier – scheduler module classifies and sends data to processing elements and sews back the data with the header after the processing is done and the scheduler picks them up and with the help of update from controller schedules the packets based on QoS parameters [63],[64]. The Classifier – Scheduler is tied to the controller through data buses and to the memory with address buses. As soon as the string of data – in our case the IP packet flows in, the classifier checks the header against CAM look up tables, strips the header and stores it in cache, rest of it is forwarded to PE's, after the PE's send the data back, registry is updated and it is joined with the header. The scheduler's registry is constantly updated by the QoS updater and once a new packet comes in it takes the update and schedules it or pushes the packet back in the queue. The proposed architecture is better explained with the help of the inter working of lookup tables and the mechanisms used for it. Figure 5.2 gives a high level idea of the working of the forwarding plane of VNP.

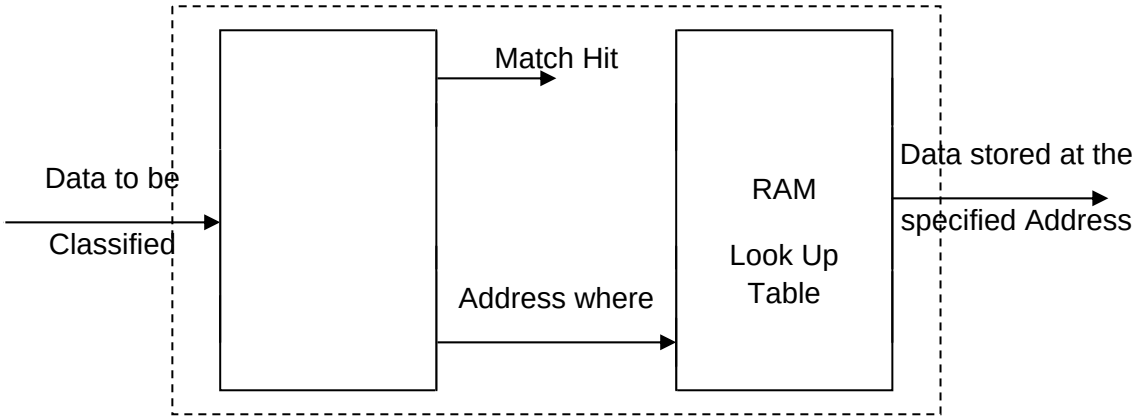


FIGURE 5.2 Block level description of inter working of CAM and RAM.

Working of CAM and RAM in read and write mode follows the below given pattern.

CAM and RAM both store data in arrays.

1. CAM takes in the data to be sorted, and sorts it in parallel. CAM's address bus accesses every word in write mode.
2. CAM and RAM both have similar write modes.
3. CAM read mode results in giving out the address and RAM's read mode results in data for the given address.
4. When tailored for VNP, CAM stores all the headers which can be classified, and rule set for VNP check.
5. Once this is classified, CAM throws out the address location where the data has been stored, and if all the rule sets match, 'Match Hit' is given as high.

6. RAM then picks up the address and loads in its cache and searched for data stored in its look up table at that location.
7. Appropriate data is thrown out, and the rest of the operations are taken care by the forwarding plane.

## 5.2 Architecture of the Proposed Classifier-Scheduler

The classifier – scheduler module is the core of VNP and is built in hierarchy, and it compares one word at a time, its architecture had both classifier and scheduler built into it, and it is just the mode in which it works that helps differentiate the packet processing.

### 5.2.1 Operation of the Classifier Module

Fig 7 gives us an idea of the architecture of the classifier, as and when the packet comes in, the CAM picks it up. CAM is built using hierarchy, initially 4 bits are compared to the LUT and that out come is matched to an outcome of another 4 bits, this makes up a cam word, on similar lines [65], [66], 1 word is compared to another and the module can classify a header file of ‘n’ words, single word comparator (SWC) compares 4 bits at a time, the detailed architecture is given in figure 5.3.

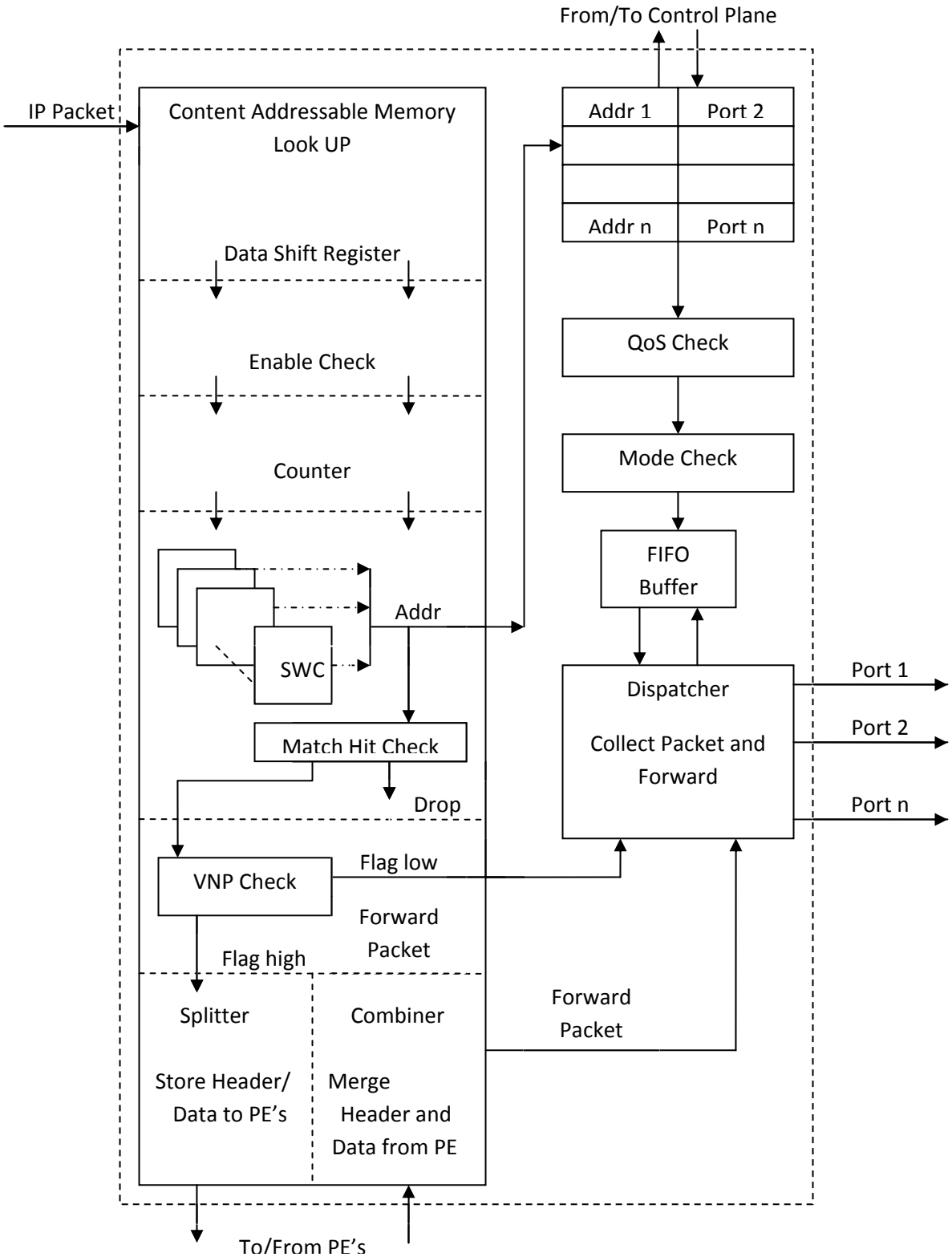


FIGURE 5.3 Internal architecture of packet classifier-scheduler.

The CAM lookup table designed is based on the ability of comparing bits in a hierarchical manner.

### 5.2.2 Operation of the Scheduler Module

A packet scheduler, is usually implemented in software, and one of the leading scheduling software's is the QoS scheduler, this gets constant updates of traffic and congestion and schedules packet accordingly, a similar functionality is achieved in the scheduling module, as soon as 32 bytes hit the scheduler, it shifts the data and checks if it is a VNP packet, it gets a constant update from the QoS update module and the controller, once the mode and QoS gives a go the packet is scheduled to appropriate port. The scheduler utilizes FIFO as its buffering technique, FIFO simply forwards packets as they come in, however if the QoS of a particular outgoing queue is not good, the scheduler registry is updated and that packet or packet stream is pushed to the end of the queue. FIFO functionality is explained with the help of figure 5.4.

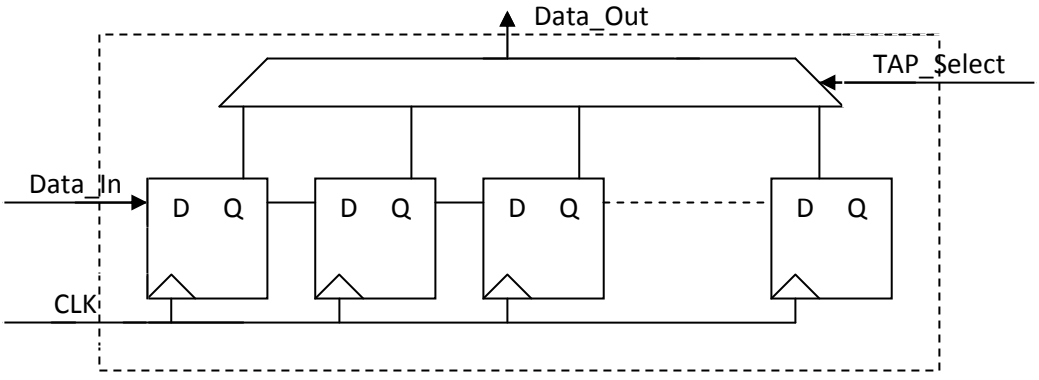


FIGURE 5.4 Block level description of FIFO buffer.

FIFO is also known as a named Pipe, the FIFO has two important nodes, that is head and tail, the packets enter the tail and are scheduled after they make it to the head after waiting for the packets in front of it are scheduled. Also the two states of FIFO are FIFO Empty and FIFO Full, this involves the address register, as read reached write, FIFO ‘empty’ is triggered, and as ‘write’ reaches ‘read’, FIFO ‘full’ signal is triggered. And in the proposed architecture when the QoS gives an update that there is congestion or traffic, the packet is moved to the tail of the queue.

# CHAPTER 6

# PROTOTYPE DEVELOPMENT

This prototype has been written through Xilinx 9.1i and simulated using its built in simulator. CAM architecture has been considered as the basis for the classifier – scheduler and has been simulated and the synthesis report shows that 2211 cells were used. Based on the address that is given out, a flag bit has been added in the options field in the physical packet address, so that the classifier can easily identify a VNP packet from a non VNP packet, if the flag is high, the packet is considered a VNP packet, if not it is dropped, In all the algorithms, data is read in little endian fashion.

## 6.1 Basic Design Considerations

The design hierarchy in VLSI systems is as follows [67]

1. System
2. Module
3. Gate
4. Circuit and
5. Device

A digital design engineer can start his design from any of these level, however a top to bottom approach is always preferred as system level design can be done through HDL design, which can with the help of translators be broken down to module and gate and circuit and design level, if there is a bug in

the system level description, it is easier to decode, and fix before the chip is meant for manufacturing. However if there is a bug in the manufactured chip itself, there is no debugging and rerunning , the chip will be simply discarded.

Designers prefer to play with layout design, which is at gate level , this helps the designers to, figure out, unwanted capacitances and inductances, however becomes tedious as and as the transistor count on the chip increases. One can imagine the Man power that goes in checking or designing a gate level program or netlist of a chip containing hundred's of gates, the chips that are manufactured at this time have millions of transistors in them.

This thesis has performed the system level design of the classifier – scheduler in VHDL and simulated it using ModelSim. Here is a brief description of the tool being used. The code is written in VHDL in Xilinx 9.1i on windows XP platform and simulated using the inbuilt ISE simulator.

VHDL is a hardware descriptive language which can be used to model a digital system. The digital system can be as simple as a logic gate or as complex as a complete electronic system, once the VHDL code is written it can be dragged onto an FPGA board and implemented, based on its complexity.

## 6.2 Constructs in VHDL

There are five different types of primary constructs, called design units, which are:

1. Entity declaration

2. Architecture body
3. Configuration declaration
4. Package declaration
5. Package body

The entity declaration describes the external view of the entity, like the input and output names. Architecture body contains internal description of the entity, such as set of concurrent or sequential statements. A configuration declaration is used to bind one architecture body to many architecture bodies. An entity may have many different configurations. Encapsulation of set of related declarations is called package declaration.

Architecture body is what describes the modeling styles of the entity which can be any one of the three:

1. Structural
2. Data flow
3. Behavioral

In structural the entity is described through a set of interconnected components. If an entity is described primarily using concurrent signal assignments it falls under the data flow style of modeling. Behavioral modeling describes the behavior of the entity in question, as a set of statements that are executed sequentially in the specified order.

Once a VHDL code is written to run on an FPGA or to design an ASIC, however it is not run on the host machine's Kernel like C or JAVA or other software programming languages, so for us to check the correctness and verify the working of the hardware program, a test bench is needed. A test bench has three main purposes

1. To generate stimulus for simulation
2. To apply this stimulus to the entity under test and collect the output responses
3. To compare the output responses with the expected values

Stimulus is automatically applied to the entity under test by instantiating the entity in the design of modules in VNP has been a combination of structural and behavioral models.

ISE simulator is an integrated simulator that comes with Xilinx Webpack, this is a simulation tool for HDL programs such as VHDL, Verilog, to start with the simulation. VHDL design of the thesis has been simulated using the ISE simulator. To simulate an HDL file the starting steps are to,

1. Create a working library
2. Compile design files
3. Link to resource libraries
4. Run simulation
5. Debug results

ISE simulator has features, simulation for all of Xilinx leading devices, such as Cool Runner™ II, Spartan™-3, and Virtex™-4.

## 6.3 Configuring CAM for Look Up

The CAM can be adjusted to match variable header length; the main trick would be to make the controller shift to the assumption that the stored header is shorter or longer as compared to earlier header length. This would involve the active participation of control plane of VNP, where in LUT metrics are changed according to control signal, in this thesis , the control plane rules are changed rather than the forwarding – plane rules. The CAM lookup table designed is based on the ability of comparing bits in a hierarchical manner; try to Xilinx primitives are used to achieve this. Figure 6.1, describes this functionality of the basic comparison block.

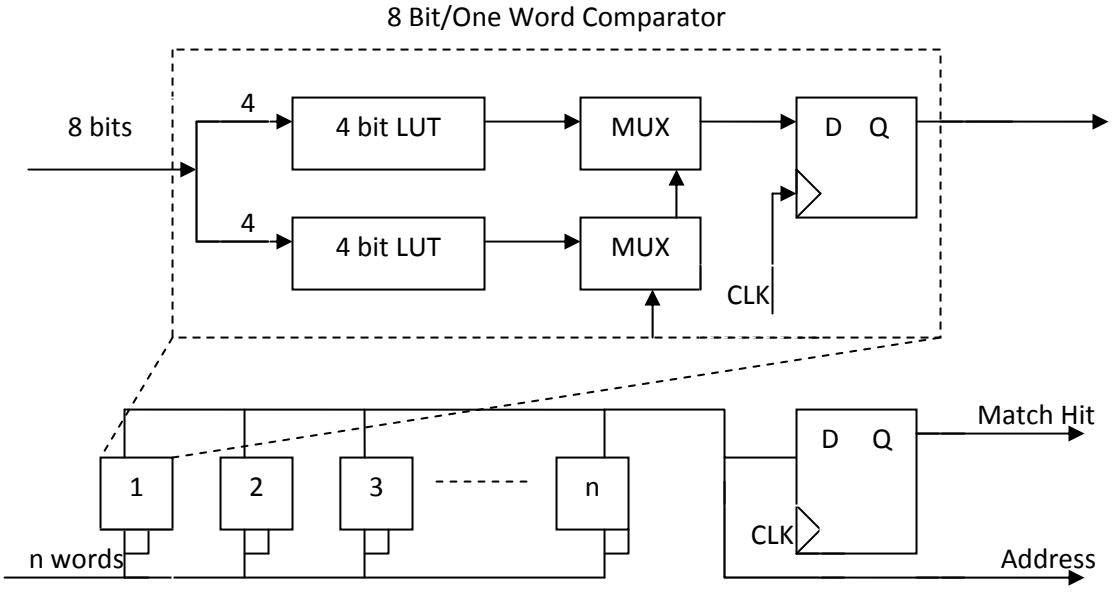


FIGURE 6.1 Configuring lookup for CAM.

The working of the basic module of CAM, is represented in figure 6.1 , the basic block is a 4 bit LUT, as 8 bits flow in (one CAM word), it is channeled as 2 streams of 4 bits, once that is done, the 4 bits are cross checked against the LUT, once that is done, the match signal is sent out, this is done in parallel with all the words that come in, Above figure represents an ‘n’ word comparator and as all the bits are matched, the address and a match signal is given out, if a match is not found, the match signal is low and garbage value is given as the address.

## 6.4 Challenges in Prototyping

This code, when laid down on an FPGA, that FPGA will talk to only another ASIC or FPGA which is programmed to support this packet structure only, this packet and FPGA cannot talk to a normal router, as – UDP’s RFC 4293, is violated, the packet sent would be an experimental packet, it simply mimics the real IP packet, and defines all the fields, such as TTL, data length etc, but this will still need to be supported by the control unit, as the operating system still needs to be mimicked, and its kernel to come up with the functionalities of TCP, UDP, RTP, RTSP, SMTP and so on. The functionality which has been described above describes the forwarding – plane operation of the router, the controller will need to talk to rest of the network and keep updating the LUT’s and support the functionality of control – plane of the router.

In real time once, the physical layer header (20 bytes) is stripped, further studies are needed to prod into the header of above layers – that is network,

application and session, and which are not fixed. The classifier supported by CAM tables to check for variable length header, can be achieved with some minor modifications. However the module will still need to send and receive constant updates from the controller, which will be an image of that part of the kernel of the operating system which takes care of breaking down and scheduling of a packet.

## 6.5 Simulation and Experimental Results

The system level designs of the combined classifier and scheduler, has been written, this will utilize an experimental packet. The code is written in an hierarchical fashion and is a combination of structural and behavioral type of coding.

Experimental packet of length 32 Bytes is considered, since in internet – real time data is always sent in bursts, this module considers a burst of 32 packets at a time. As the stream of data comes in the classifier – scheduler considers first 20bytes of this string of data as header. The data is provided by the test bench, which is 32 bytes, 32 times. As the first string of data enters, data shift registers are updated with the first 20bytes of data, rest of the bytes are considered as don't care bits through the CAM. If VNP is set as high, data and header are split, because classifier – scheduler is not yet connected to processing elements, the data is shifted by 2 shifts and update header (so as to mimic number of hops update). After the data is also shifted, it is combined with the string and wait in the forwarding queue. Meanwhile, the address thrown by CAM is taken by RAM buffer and appropriate port number is thrown out, Four

ports have been considered. These two functions are performed in parallel; once the port number is collected experimental data string is forwarded to specific particular port.

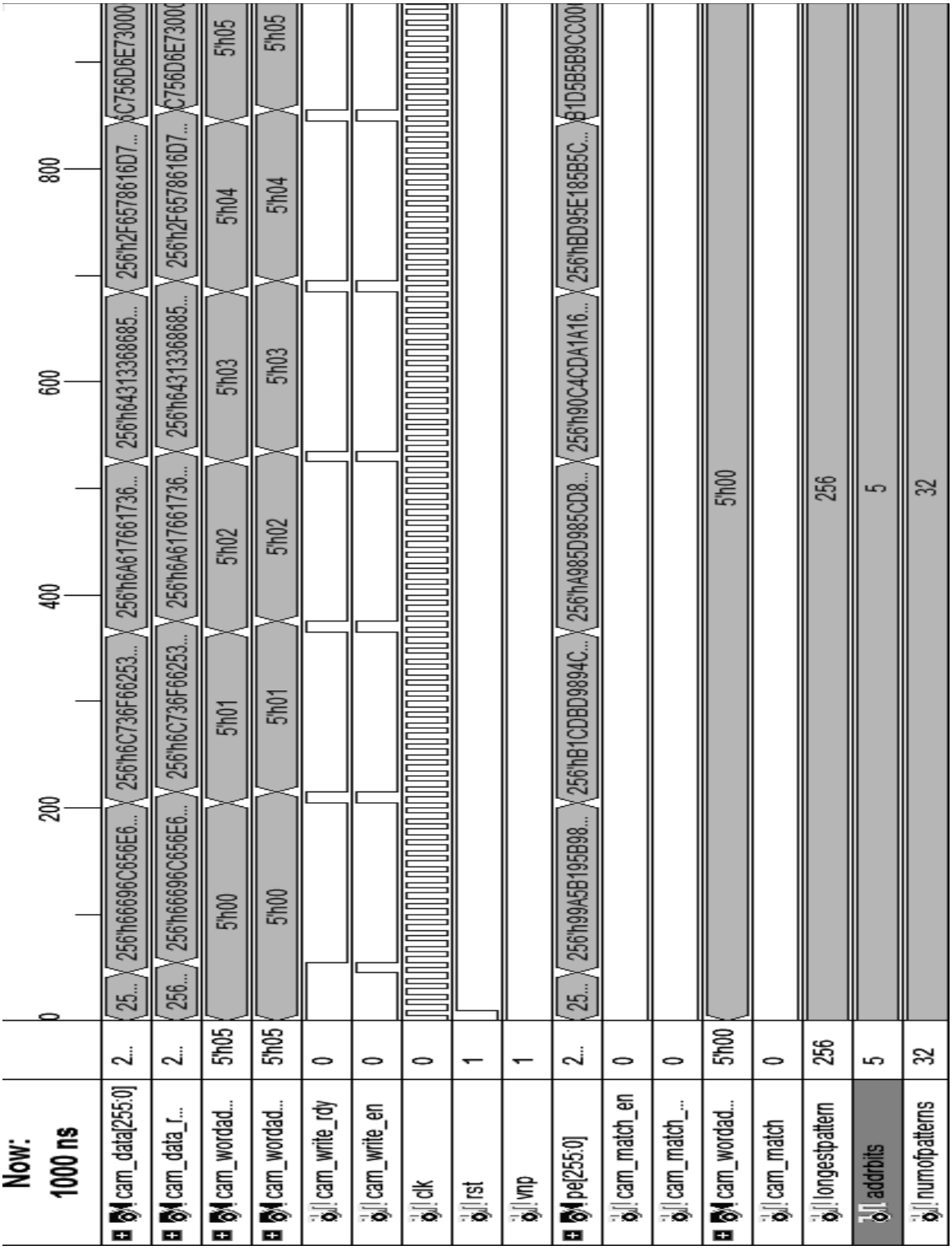


FIGURE 6.2 Simulation of experimental version of the classifier – scheduler in classification mode.

Figure 6.2, shows that after the incoming data stream is read, it is split and the address is separated from the data and the data is them merged, and sent out, and this is evident from the delay observed in the out coming stream.

cam_data represents the string that is being processed, clk – represents clock, rst – represents reset, pe – represents the processing elements port through which data is sent and also taken in. cam_match – represents the Match Hit signal. We find that when ‘vnp’ signal is high data is being processed as proposed. Because this is an experimental setup, this prototype has the flexibility of adding vnp directly in options, if not vnp flag will be a part of application layer header information. The address that is given as output is automatically read by the scheduler buffer to find out the appropriate port.

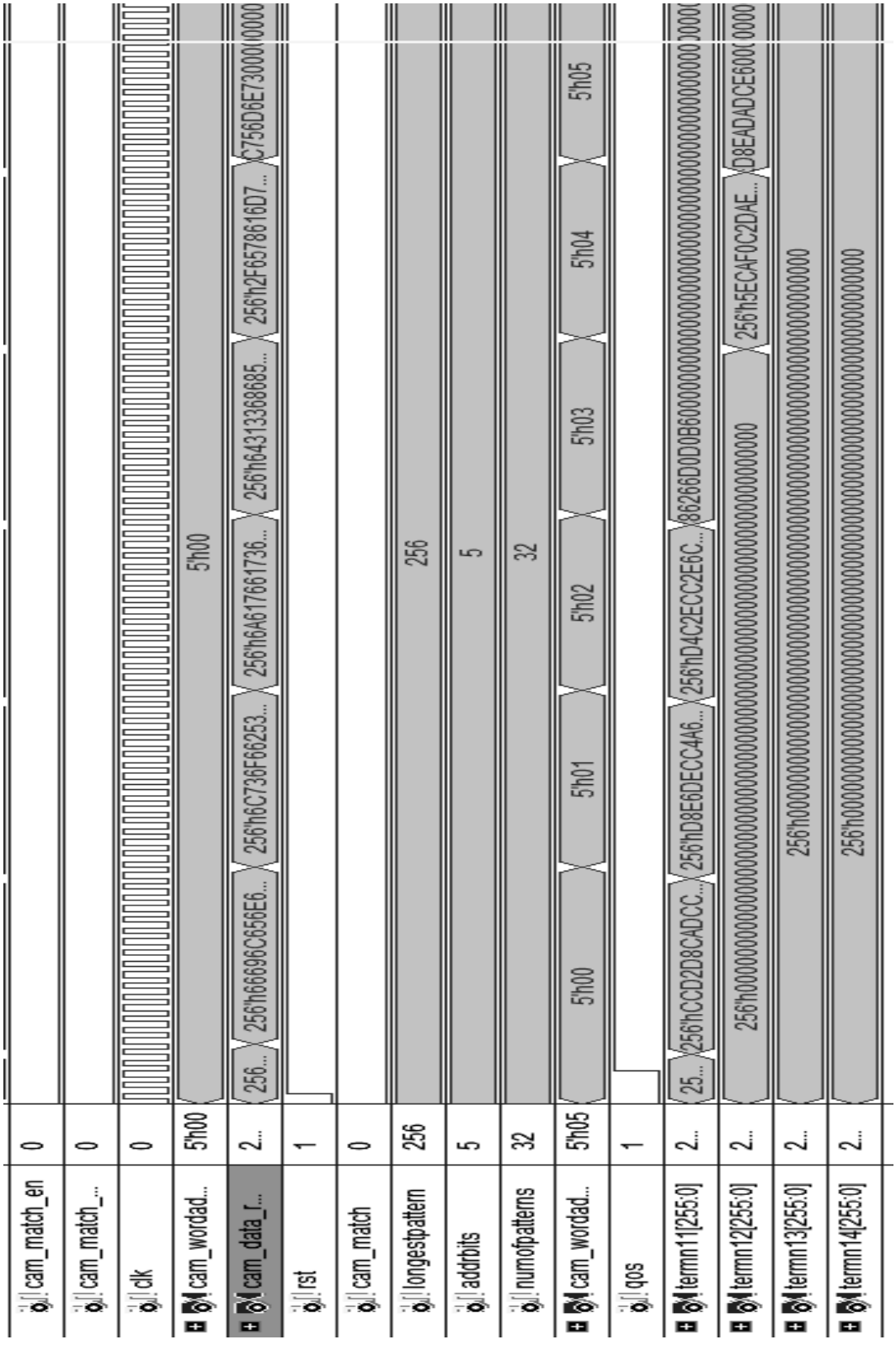


FIGURE 6.3 Simulation of experimental version of the classifier – scheduler in scheduler mode.

It is seen in figure 6.3 that, after that data flows in, if the VNP is high, and the QoS attribute is high, data is sent through ports, and when the broadcast is high, all ports transmit data, which is typical in case of UDP, classifier – scheduler module proposed here will be placed in VNP at the content provider rather than the VNP placed in the core network, hence the broadcast mode of the processor is activated.

Termn11, Termn12, Termn13, Termn14 represent the 4 ports that have been assigned to VNP. Based on the address loaded into the buffer, the scheduler forwards the packets to those particular slots.

CAM classification is the choice of classification and figure 6.3, gives the simulation results of the structural level design of the CAM, which tried to handle variable header length. CAM engages in a read through the look up tables after it gets data from its input ports, all the bits are compared in parallel and the address of the matched data stream is given in the output, if there is no match, garbage value is given out. Synthesis report is displayed in figure 6.4 and figure 6.5, and output results are summarized in Table 6.1.

.

```
=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================
---- Source Parameters
Input File Name                    : "class_sched.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "class_sched"
Output Format                      : NGC
Target Device                      : xc2s15-6-cs144

---- Source Options
Top Module Name                    : class_sched
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Resource Sharing                   : YES
Asynchronous To Synchronous        : NO
Multiplier Style                   : lut
Automatic Register Balancing       : No

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 100
Add Generic Clock Buffer(BUFG)     : 4
Register Duplication               : YES
Slice Packing                      : YES
Optimize Instantiated Primitives   : NO
Convert Tristates To Logic         : Yes
Use Clock Enable                   : Yes
Use Synchronous Set                : Yes
Use Synchronous Reset              : Yes
Pack IO Registers into IOBs        : auto
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
```

FIGURE 6.4 Synthesis report: Summary of the experimental classifier – scheduler.

```
=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : class_sched.ngr
Top Level Output File Name         : class_sched
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 1554

Cell Usage :
# BELS                             : 1240
#      GND                         : 1
#      INV                         : 1
#      LUT2                        : 67
#      LUT2_L                      : 1
#      LUT3                        : 88
#      LUT4                        : 138
#      LUT4_D                      : 8
#      LUT4_L                      : 64
#      MUXCY                       : 866
#      MUXF5                       : 5
#      VCC                         : 1
# FlipFlops/Latches                : 45
#      FDC                         : 33
#      FDCE                        : 12
# Shift Registers                  : 858
#      SRL16E                      : 858
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 271
#      IBUF                        : 264
#      OBUF                        : 7
=========================================================================
Device utilization summary:
---------------------------

Selected Device : 2s15cs144-6

 Number of Slices:                     1090  out of    192   567% (*)
 Number of Slice Flip Flops:             45  out of    384    11%
 Number of 4 input LUTs:               1225  out of    384   319% (*)
    Number used as logic:               367
    Number used as Shift registers:     858
 Number of IOs:                        1554
 Number of bonded IOBs:                 272  out of     86   316% (*)
 Number of GCLKs:                         1  out of      4    25%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------
```

FIGURE 6.5 Final synthesis report of the experimental classifier – scheduler.

Xilinx gives us the flexibility to view the register transfer level (RTL) description of the code and its is displayed in figure 6.6, figure 6.7, figure 6.8. From the viewer's point of view right hand side represent the inputs and the feedback lines of the classifier – scheduler module and the left hand side represent the output lines, which are 4 output ports, address if the IP packet and data going to the processing elements.

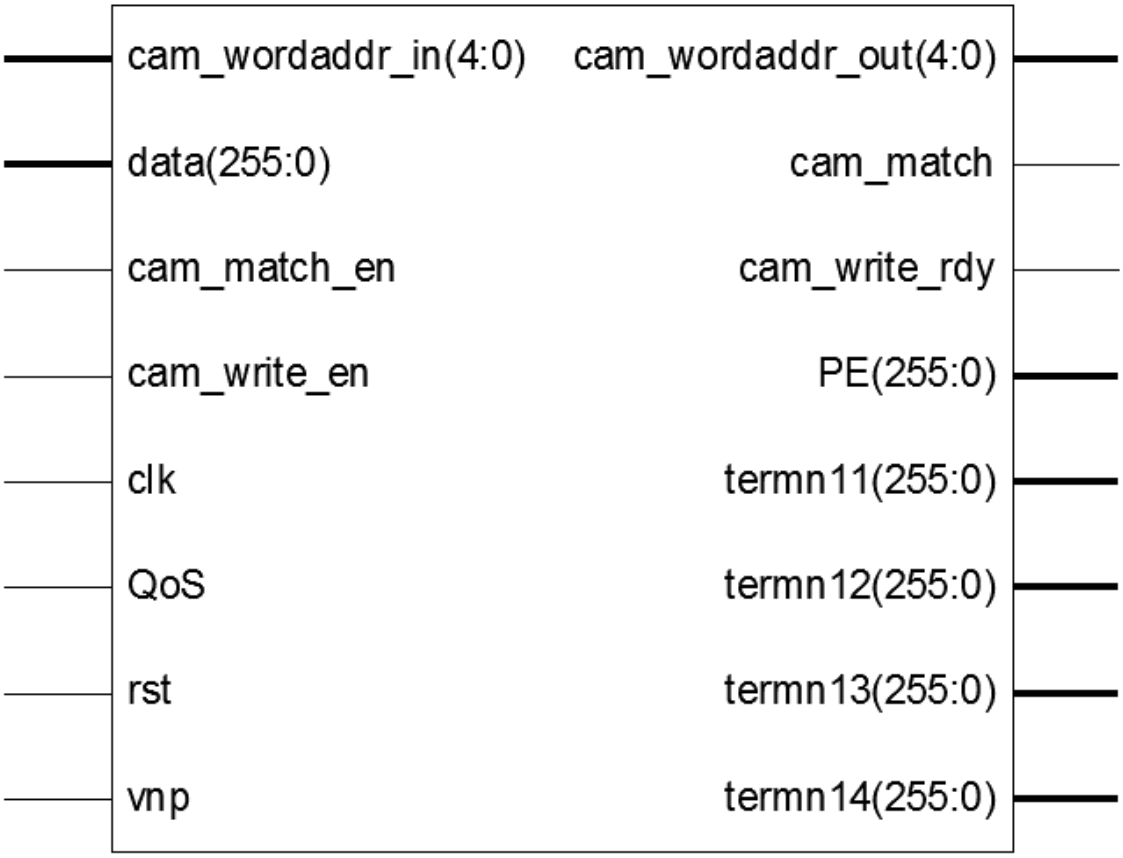


FIGURE 6.6 High level RTL achematic of classifier - scheduler.

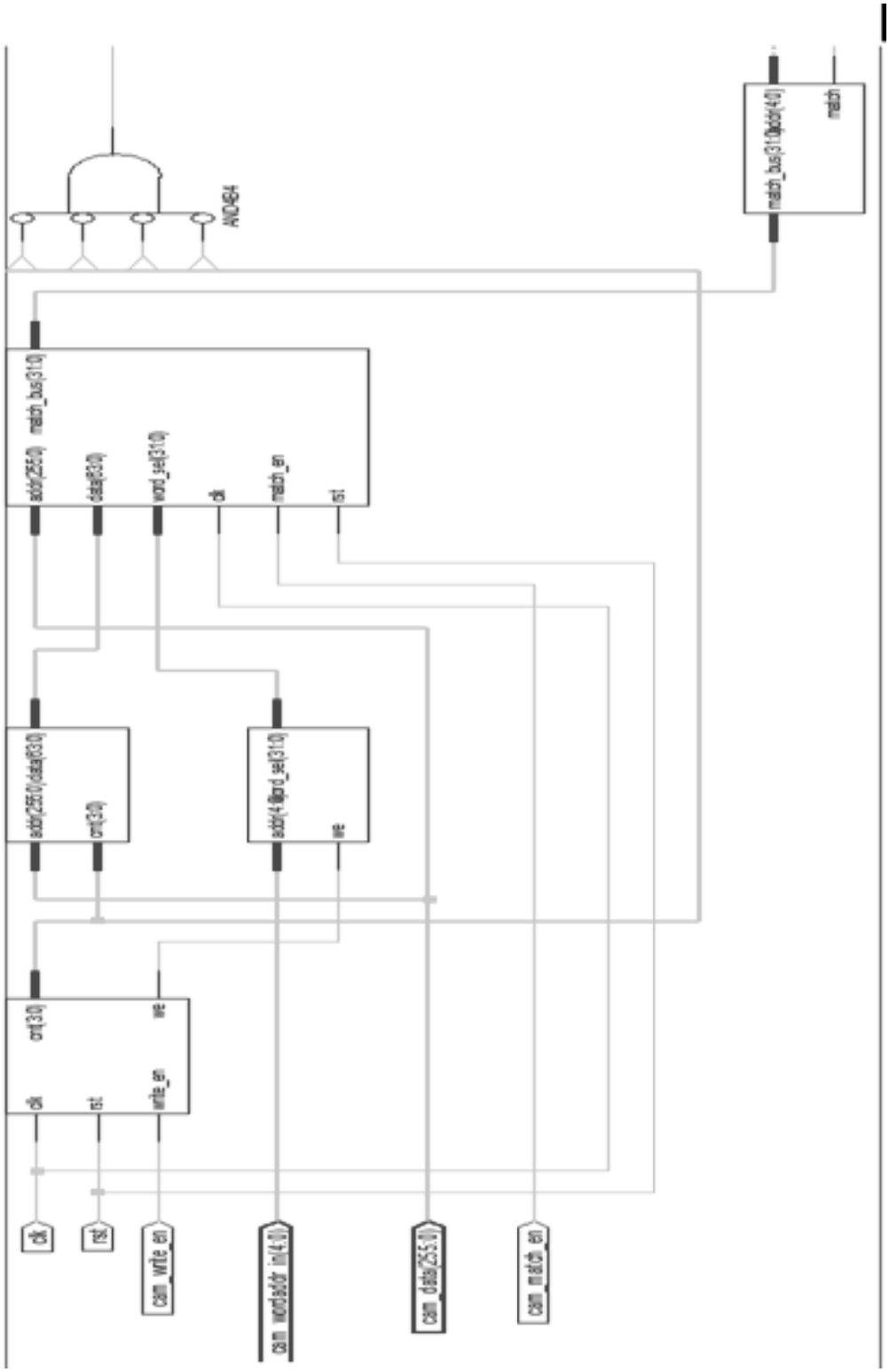


FIGURE 6.7 Synthesized RTL of classifier –scheduler represented in figure 6.6.

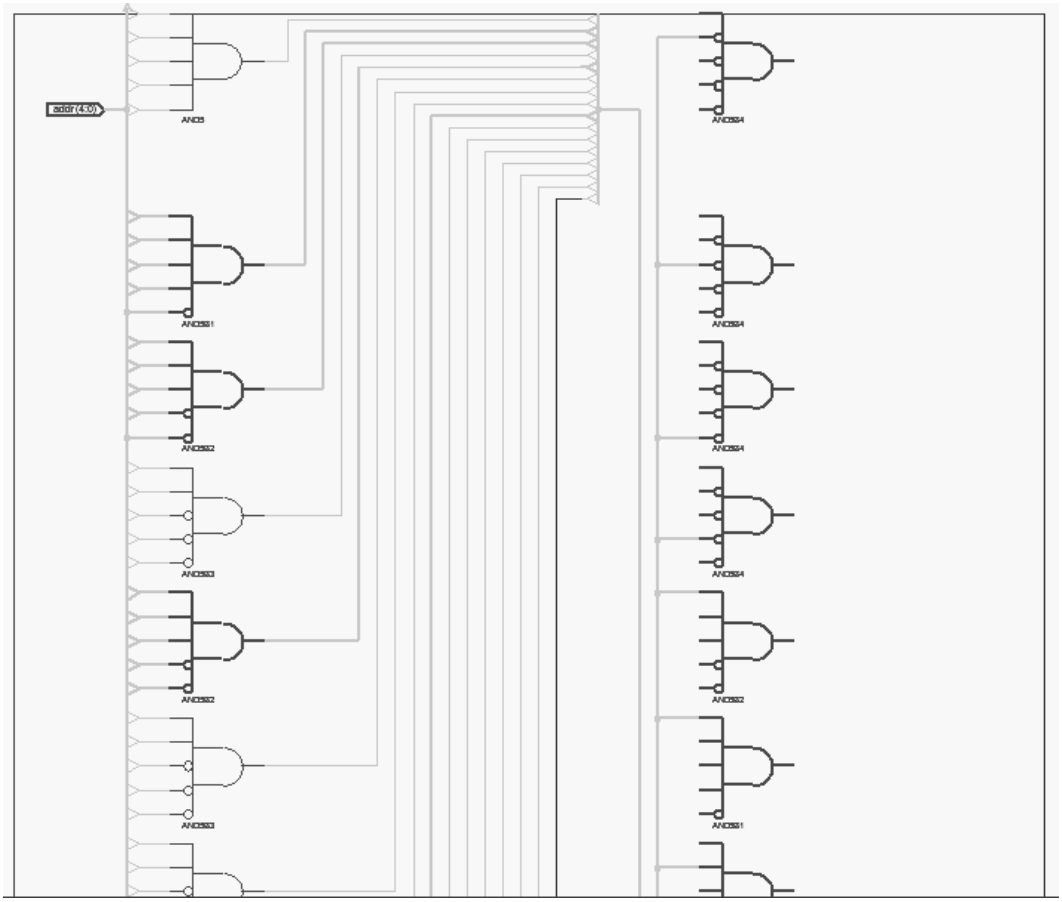

FIGURE 6.8 Logic level description of the RTL given in figure 6.7.

Xilinx gives us the flexibility of simulating the implementation of the code onto an FPGA, its mapping, placement and routing. The code has been simulated under four different board scenarios and have listed the results obtained in table 6.1. The power consumption has been calculated under the ambient temperature of 25 $^{o}$C and $V_{in}$ was kept at 2.5 volts.

TABLE 6.1 Comparison of design metrics of classifier – scheduler for various FPGA technologies.

| Design metrics / Technology used | Virtex XVC800 6BG560, speed -6 | QPro Virtex Hi-Rel XQV1000 | Spartan 2E XC2S600E FG456 speed -7 | Automotive Spartan 3 E XA3S1600E FGG400 |
|---|---|---|---|---|
| Power (*mW*) | 32.16 | 32.16 | 33.6 | 101.74 |
| Frequency (*MHz*) | 104.482 | 84.083 | 111.896 | 112.524 |
| Logic cells | 2408 | 2408 | 2408 | 2371 |
| Memory (*Kb*) | 215672 | 224056 | 211788 | 235956 |
| Throughput (*Mbps*) | 1.671 | 1.345 | 1.790 | 1.8 |

It is the manufacturer's choice, whether he would prefer the best throughput and risk high power dissipation as is the case with Automotive Spartan3, the module is unable to run at $V_{in}$ of 2.5, it only runs in the range of 1.19 to 1.3 volts, or go for an overall performance and choose the rest of the three. Graphs comparing the performance of technologies on which the code has been run.

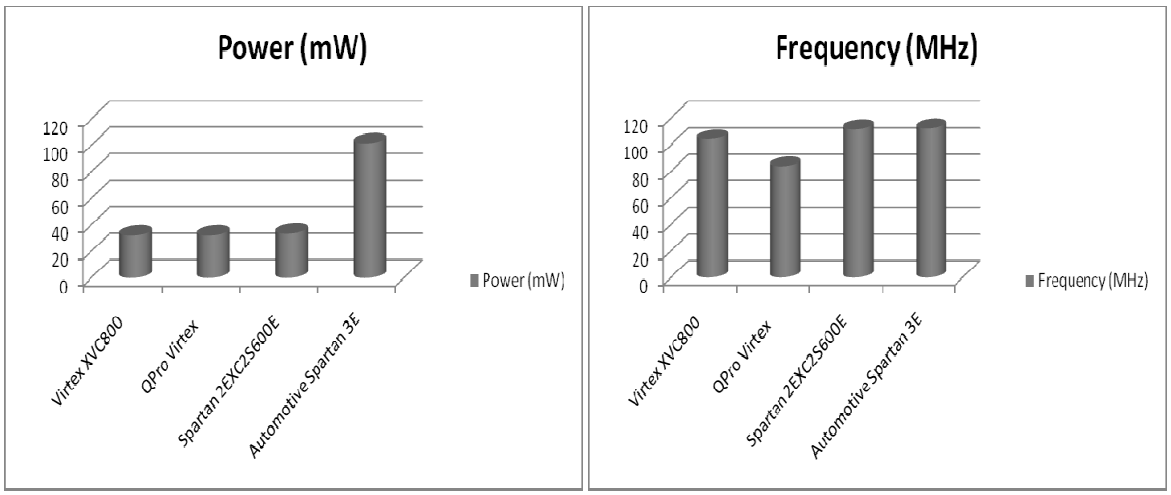


FIGURE 6.9 Power and Frequency comparison of the classifier – scheduler in various technologies.

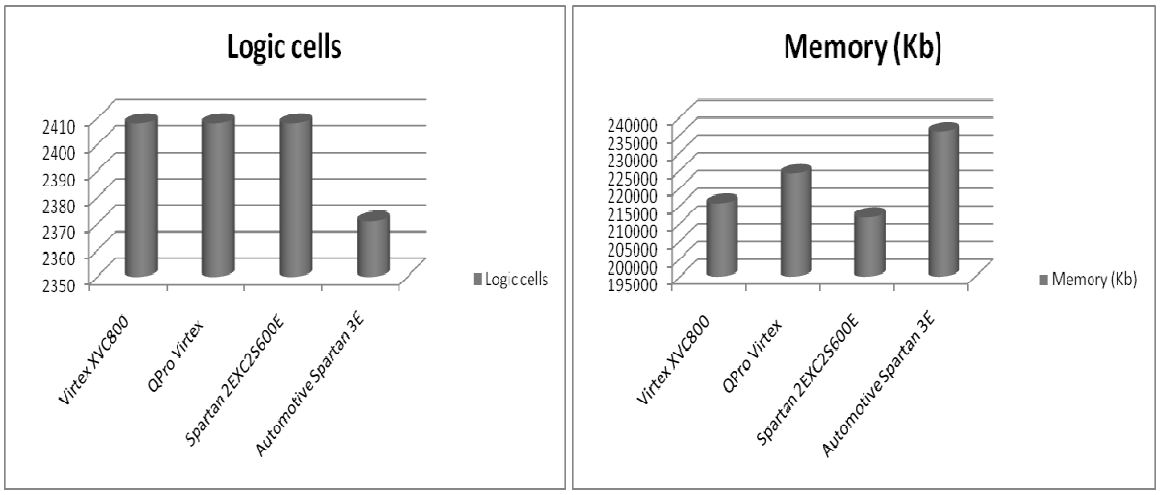


FIGURE 6.10 Logic cell count and memory consumption; comparison of the classifier –scheduler in various technologies.

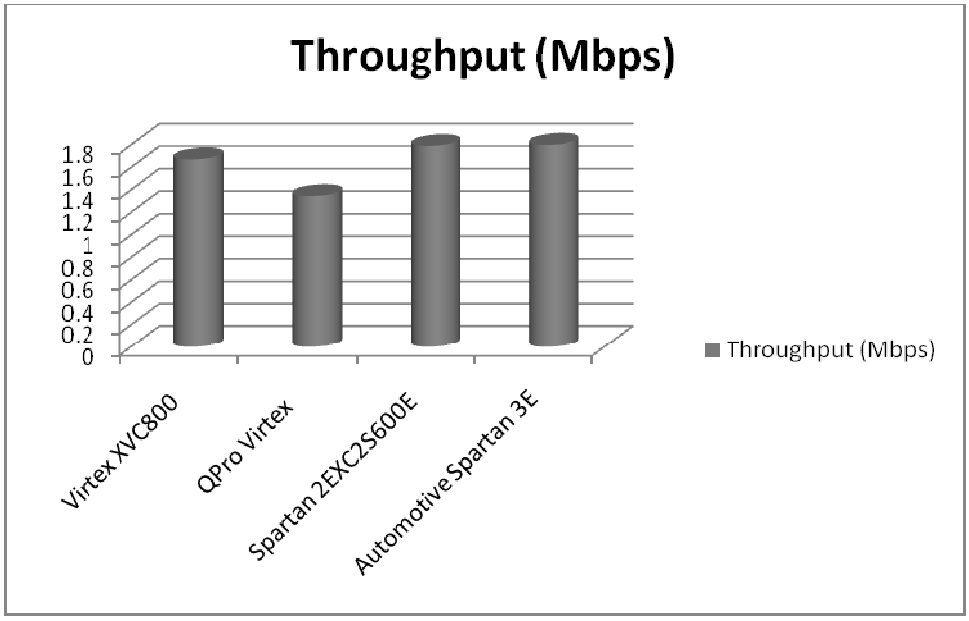


FIGURE 6.11 Throughput comparison of the classifier- scheduler in various technologies.

It can be inferred from the data given above that Automotive Spartan 3 gives the highest throughput of the classiffer-schedu ler module. The floorplan given for the classifier – schedul er is shown in figure 6.12 and figure 6.13.

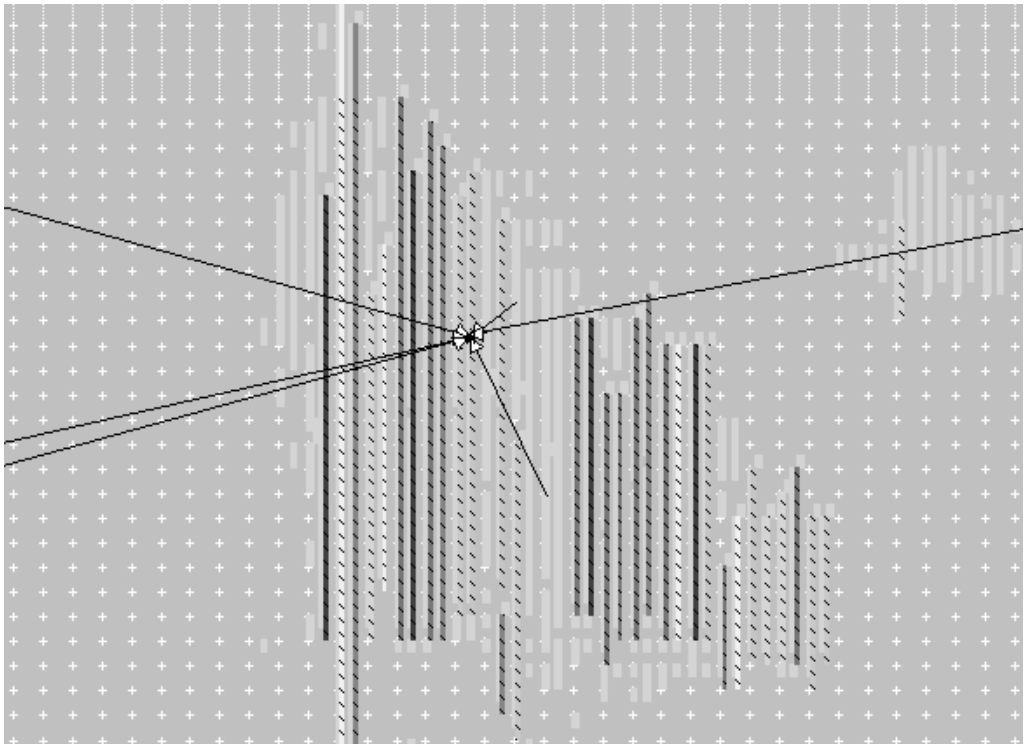

FIGURE 6.12 Floorplan design given by xilinx for classifier – scheduler.

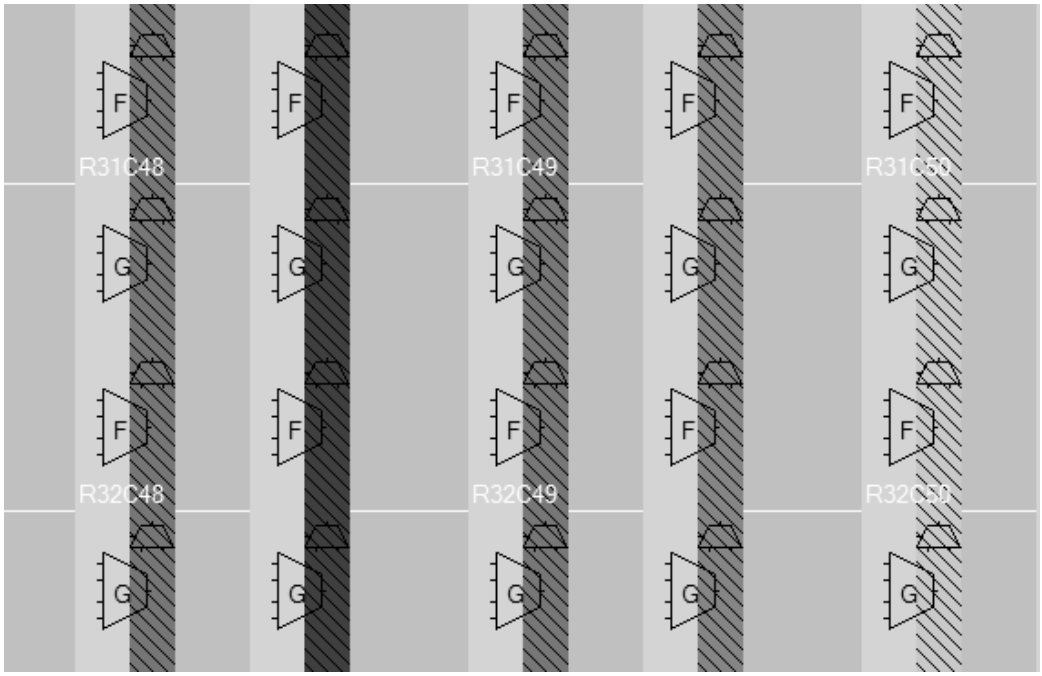


FIGURE 6.13 Gate level description of floop plan of classifier – scheduler given in 6.11.

# CHAPTER 7

## CONCLUSION AND FUTURE WORK

This thesis is the initiation of the development of the basic modules of VNP. The prototype has been able to simulate the forwarding – plane of VNP consisting of the classifier and scheduler, this proves that a complete system can be developed, and with the help of a good control plane design, a system that is not just experimental but one which complies with RFC's can be built. This thesis has achieved the classifier – scheduler in structural and behavioral design. The classifier – scheduler can be used for other network processors with appropriate modifications. I have come up with the basic data path and architecture of VNP, and discussed the complexities of developing VNP as a system on chip design.

Future studies may include the building of control – plane of the VNP to be able to comply with RFCs, and the watermarking, feedback and control plane for the classifier - scheduler to complete the VNP. The design can be further improved to use low power techniques like power gating and step up and step down converters to handle power issues. Voltage scheduling would be introduced, which will in turn help with power scaling.

# REFERENCES


[1] M. Robert; “The dawn of digital TV,” *IEEE Spectrum Online*, 42 (10): 26-31, Oct. 2005.

[2] U. Sandberg; “Building direct-to-home television and entertainment networks for Europe using the new digital video broadcast standards,” in *Proceedings of the IEEE International Broadcasting Convention*, pp. 175-177, 1995.

[3] B. Alfonsi; “I want my IPTV: Internet protocol television predicted a winner,” *IEEE Distributed Systems Online*, 6 (2), 2005.

[4] K. Wing, W. Hwang, and Y. Katayama; “System-on-chip layout compilation.” US Patent 5883814.

[5] D. Gall; “MPEG: A video compression standard for multimedia applications,” *Communications of the ACM*, 34 (4), April 1991.

[6] T. Vaughan; “Digital television (HDTV) and analog television (NTSC) broadcast systems,” US Patent 5,774,193; 1998.

[7] T. Perry; “HDTV and the new digital television,” *Spectrum IEEE*, 32 (4), April 1995.

[8] D. Deloddere, W. Verbiest; “Interactive video on demand,” *IEEE Communications* 32 (5), May 1994.

[9] B. Tseng, C. Lin, J. Smith; “Using MPEG-7 and MPEG-21 for personalizing video,” *IEEE Multimedia*, 11 (Jan-Mar): 42-52, 2004.

[10] G. Pierre, K. Faraydon, P. Bromley; "Network processors: A perspective on market requirements, processor architectures and embedded S/W Tools," in *Proceedings Design, Automation, and Test in Europe* (DATE '01), pp. 4-20, 2001

[11] S. Keshav, R. Sharma; “Issues and trends in router design,” *IEEE Communications* 36 (5): 144-151, May 1998

[12] P. Gupta, N. McKeown; “Algorithms for packet classification,” *IEEE Network* 15 (2): 24-32, March-April 2001

[13] J. Dent; “Achieving higher levels of electronic integration through system on chip,” *IEEE Aerospace and Electronic Systems* 16 (10): 36-41, Oct. 2001

[14] F. Zhao; “Synchronizing routing information of control plane and routing plane gracefully,” in *Proceedings of the 2006 Workshop on High Performance Switching and Routing*, p. 6, June 2006.

[15] G. Chuanxiong; “SRR: An O(1) time complexity packet scheduler for flows in multi service packet networks,” in *Proceedings SIGCOMM’01*, August 27-31, 2001.

[16] Agilent Technologies; “IPTV quality of experience, test solution” (white paper).

[17] J. Lee, J. Kim, S. Kim, C. Lim, J. Jung; "Enhanced distributed streaming system based on RTP/RTSP in resurgent ability," in *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pp. 568-572, 2005.

[18] J. Lotspeich; “Anonymous trust: Digital rights management using broadcast encryption,” in *Proceedings of the IEEE* 92 (6), June 2004.

[19] T. Wolf, J.S. Turner; “Design issues for high-performance active routers,” *IEEE Journal Selected Areas in Communications* 19 (3): 404-409, March 2001.

[20] N. Shah; “Understanding network processors.” Thesis, Department of Computer Science, UC Berkeley, 2001.

[21] R. Rao, S. Dey, A. Nguyen, K. Faraydon; "On-chip communication architecture for oc-768 network processors," in *Proceedings of the 38th Conference on Design Automation (DAC'01)*, pp. 678-683, 2001

[22 ] L. Kencl, J. Le Boudec; “Adaptive load sharing for network processors,” in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2002)* 2: 545-554, 2002

[23] G. Goldszmidt and G. Hunt; “Scaling Internet services by dynamic allocation of connections,” in *Proceedings of the 6th International Symposium on Integrated Network Management (IFIP/IEEE)*, pp. 171-184, 24-28 May 1999.

[24] Faraydon K; "Network processors: The new frontier in SoC design and validation," in *Proceedings of the DATE Conference*, 2000

[25] K. Lahiri, A. Raghunathan, S. Dey; "Efficient exploration of the SoC communication architecture design space," in *Proceedings of the International Conference on Computer Aided Design*, pp. 424- 430, Nov. 2000.

[26] T. Srinivasan, N. Dhanasekar, M. Nivedita, R. Dhivyakrishnan, A. Azeezunnisa; "Scalable and parallel aggregated bit vector packet classification using prefix computation model," in *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, pp.139-144, 2006

[27] S. Singh, F. Baboescu, G. Varghese and J. Wang; "Packet classification using multidimensional cutting," in UCSD Technical Report CS2003-0736, 2003.

[28] B. Xu, D. Jiang, J. Li; “HSM: A fast packet classification algorithm,” in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA 2005)* 1 (28-30): 987-992, March 2005.

[29] P. Gupta P, McKeown N; “Algorithms for packet classification,” *IEEE Network* 15 (2): 24-32, Mar/Apr 2001

[30] V. Srinivasan, S. Suri; “Packet classification using Tuple space search,” in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 135-146, 1999

[31] K. Zuver; “Debugging of an Internet packet scheduler using the identify software,” *Syndicated–Technical Newsletter for ASIC and FPGA Designers*.

[32] D.E. Wrege, J. Liebeherr J; “A near optimal packet scheduler for QoS networks,” in *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '97)* 2 (7-11): 576-583, April 1997.

[33] G. Hasegawa, T. Matsuo, M. Murata, H. Miyahara; “Comparisons of packet scheduling algorithms for fair service among connections on the Internet,” in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2000)* 3(26-30): 1253-1262, March 2000.

[34] Jinggang W, Ravindran B; “BPA: A fast packet scheduling algorithm for real-time switched ethernet networks,” in *Proceedings of the International Conference on Parallel Processing* (18-21), pp. 159-166, Aug. 2002.

[35] M. Shreedhar, G. Varghese; "Efficient fair queueing using deficit round robin," in *Proceedings ACM SIGCOMM Computer Communication Review* 25, Oct 1995.

[36] P. Goyal, M. Harrick; “Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks,” in *Proceedings ACM SIGCOMM Computer Communication Review* 26 (4), Oct. 1996.

[37] D. David, S. Shenker; “Supporting real-time applications in an integrated services packet network: Architecture and mechanism,” in *Proceedings ACM SIGCOMM Computer Communication Review* 22 (4), Oct. 1992.

[38] J. Richard, X. Jun; “On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms,” in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* 13 (1): 15-28, Feb 2002

[39] Z. Ming, A. Krishnamurthy; “Probablistic packet scheduling: Achieving proportional share bandwidth allocation for TCP flows,” Department of Computer Science, Princeton University; Department of Computer Science Yale University.

[40] F. Baaboescu, G. Varghese; “Packet classification for core routers: Is there an alternative to CAMs?” in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2003)* 1 (30): 53-63, April 2003

[41] M. Nourani, J Mohammad; “A TCAM-based parallel aachitecture for high-speed packet forwarding,” *IEEE Transactions Computers* 56(1): 58-72, 2007.

[42] D. Taylor, W. Spitznagel; “On using content addressable memory for packet classification.” Applied Research Laboratory, Washington University in Saint Louis.

[43] J. Cohn, D. Stout, P. Zuchowski, S. Gould, T. Bednar, D. Lackey; "Managing power and performance for system-on-chip designs using voltage islands," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD '02)*, pp. 95-202, 2002

[44] B. Cordan; “An efficient bus architecture for system-on-chip design,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 623-626,1999.

[45] Y. Chu; “Dynamic data encryption system based on synchronized chaotic systems,” *IEEE Electronic Letters* 35 (4), 1999.

[46] P. Dong, J. Brankov; “Digital watermarking robust to geometric distortions,” *IEEE Transactions on Image Processing* 14 (12), December 2005.

[47] V.A.J Goor; “Address and data scrambling: Causes and impacts on memory tests,” in *Proceedings of the 1st IEEE International Workshop on Electronic Design, Test and Applications*, pp. 128-136, 2002.

[48] S. Brown, J. Rose; “Architecture of FPGAs and CPLDs: A tutorial,” Department of Electrical and Computer Engineering, Toronto University.

[49] M. Borgatti, F. Lertora, B. Foret, L. Cali; “A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O,” IEEE Solid-State Circuits 38 (3): 521-529, Mar 2003.

[50] S.M. Scalera, “The design and implementation of content switching FPGA,” in *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 78-85, April 1998.

[51] Xilinx tool box: “VHDL/Verilog libraries and models,” www.xilinx.com

[52] VHDL tutorial: “What is VHDL?” McGill Center for Intelligent Machines.

[53] S. Mohanty; “A video broadcasting network processor with digital rights management,” Computer Science and Engineering, University of North Texas.

[54] S.N Bhatti; “QoS-sensitive flows: Issues in IP packet handling,” *IEEE Internet Computing* 4 (4), Aug 2000.

[55] P.P.K Lam; “UDP-liter: An improved UDP protocol, for real-time multimedia applications over wireless links,” in *Proceedings of the 1st International Symposium on Wireless Communication Systems*, pp. 314-318, Sep 2004.

[56] B. Leiner; “The Darpa Intenret protocol suite,” *IEEE Communications* 23 (3): 29-34, 1985.

[57] F. Baker; “Requirements for IP version 4 routers,” Network Working Group, Cisco Systems.

[58] H. Miyatake, M. Tanaka, Y. Mori; “A design for high-speed, low-power CMOS fully parallel content-addressable memory macros,” *IEEE Journal of Solid-State Circuits* 36 (6), June 2001.

[59] M. Nourani and S. Vijayasarathi; “A reconfigurable CAM architecture for network search engines”, Center for Integrated Circuits and Systems, University of Dallas, Richardson.

[60] XAPP application note: “Content addressable memory (CAM) in ATM applications”; application note: Virtex Series and Virtex-II Series, XAPP202 (vo1.2) January 6, 2001.

[61] J. Lunteren, “Searching very large routing tables in wide embedded memory,” in *Proceedings of IEEE Globecom* 3: 1615-1619, November 2001.

[62] United States Patent 5706224: “Content addressable memory and random access memory partition circuit.”

[63] G. Nilsen, J. Torresen; “A variable word-width content addressable memory for fast string matching,” in *Proceedings of the IEEE Norchip Conference* 8-9: 214-217, Nov 2004

[64] G. Hetherington; “Logic BIST for large industrial designs: Real issues and case studies,” in *Proceedings of the International Test Conference*, pp. 358-367, 1999.

[65] A. Guccione; “Content addressable memory implemented using programmable logic,” United States Patent 6278289.

[66] T. Kumaki; “A flexible multi port content addressable memory.” Department of Computer Science, National Defense Academy, Yokosuka, Japan.

[67] W. Fichtner, “Design of VLSI systems,” Embedded Systems, Springer Berlin, Column 284/1987.