A Flexible Design of Network Devices Using Reconfigurable Content Addressable Memory

Qutaiba Ali
Computer Engineering Department, University of Mosul, Iraq

Abstract: *The content addressable memory is a memory unit that uses content matching instead of addresses. content addressable memory are used in different networking, telecommunications and storage applications because of their parallel, fast search capabilities. This paper presents a new method (called array method) for designing Reconfigurable content addressable memory (RCAM). The behavior of the new method was described using VHDL and implemented using FPGA technique. Then, the performance of the method was compared to other traditional content addressable memory design methods. The proposed RCAM is configured and used as the main part of different network security devices and units (Ethernet Address Filtering Unit, WLAN MAC Address Filtering Unit ,firewall on chip, QoS packet classification unit, Routing Table Search Unit and Network Intrusion Detection System search unit). The successful implementations of the suggested RCAM prove its high flexibility to be used in different network applications.*

Keywords: *Content addressable memory, network security devices, FPGA, and throughput.*

Received August 18, 2008; accepted November 5, 2009

1. Introduction

Content Addressable Memories or (CAMs) are a class of parallel pattern matching circuits. In one mode, these circuits operate like standard memory circuits and may be used to store binary data. Unlike standard memory circuits, however, a powerful match mode is also available. This match mode permits all of the data in the CAM device to be searched in parallel. While CAM hardware has been available for decades, its use has typically been in niche applications, embedded in custom designs. Perhaps the most popular application has been in cache controllers for central processing units. Here CAMs are often used to search cache tags in parallel to determine if a cache “hit" or “miss" has occurred. Clearly in this application performance is crucial and parallel search hardware such as a CAM can be used for a good effect [12].

A second and more recent use of CAM hardware is in the networking area [11]. As data packets arrive into a network router, processing of these packets typically depends on the network destination address of the packet. Because of the large number of potential addresses, and the increasing performance demands, CAMs are beginning to become popular in processing network address information. Examples of such applications are: Ethernet and WLAN address filtering, Internet protocol filtering, Data compression, Pattern recognition, Cache tags, Fast look-up for routing tables and Searches for data switches, firewalls, bridges, and routers usually, CAM size (in bits) is expressed as the multiplication result of word_width (bit) by the number of words (CAM depth).CAM chips were implemented using different techniques. However, in order to design a reconfigurable CAM, Field Programmable Gate Array (FPGA) technique (because of its flexibility) is used [1, 3]. Basically, designers and researchers in this field use one of three different ways to implement a CAM on FPGA devices [8]:

1. SRL16 design: It introduces a methodology for designing flexible, small to medium size CAMs. By using shift register primitives built into an FPGA slice, a reconfigurable Look Up Table (LUT) (two LUTs per slice) is used to implement a single clock cycle read CAM. A 4-bit CAM word fits into each LUT. The most important features of this method are [8]:
   - One read clock cycle (or match access time).
   - 6 write clock cycles.
   - Generic “word width” from four bits up to any multiple by four bit value.
   - The CAM depth is a multiple by 16 word value.
   - Implemented using 10 complex VHDL code modules.
2. Distributed SelectRAM-based implementations offering large word width and depth adapted specifically for Asynchronous Transfer Mode (ATM) applications. The most important features of this method are [6]:
   - Implementation = 10 bits per LUT.
   - Read operation requires 16 clock cycles.
   - Write operation requires one clock cycle.
   - Implemented using 4 complex VHDL code modules.

3. CAM using BlockRAM memory: this solution is optimal for applications requiring one or two clock cycles for both read and write and 8-bit width. This methodology is based upon the True Dual-Port feature of the FPGA's block memories. The most important features of this method are [3]:
    - Implementation = Block RAM memory (CAM16x8 in each memory)
    - Read operation requires one clock cycle
    - Write operation requires one clock cycle after one erase cycle.
    - Implemented using 8 complex VHDL code modules.

The CAM design methods mentioned earlier can be evaluated using the following metrics [8]:

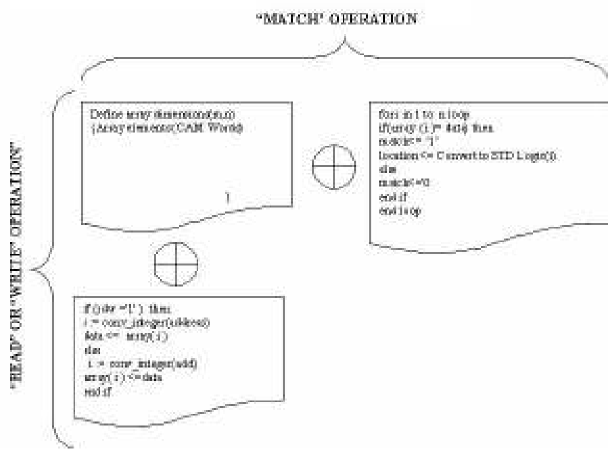
1. Maximum frequency: it can be defined as the inverse of the maximum electronic propagation delay through the designed CAM circuits.
2. Number of 'Match' clock cycles: it reflects the speed at which the CAM gives the match result between the input and stored data.
3. Number of 'Write' cycles: it reflects the speed at which the 'CAM' stores (or write) a new data word into the CAM.
4. Area Utilization: it can be defined as the amount of FPGA chip resources allocated for the CAM parts.
5. Complexity of the VHDL Code: it is preferable to use a simple VHDL code which needs only the setting of the CAM dimension, rather than using multiple complex VHDL modules.

In this paper, a new CAM design method (called array method) is presented. The design was implemented on different Xilinx platforms such as Spartan and Virtex devices. The suggested design is used efficiently to build various network devices.

2. Description of the Suggested Array Method

The CAM design methods mentioned earlier were built according to VHDL Dataflow Style Architecture [4], which specifies the circuit as a concurrent representation of the flow of data through the circuit. In the dataflow approach, circuits are described by showing the input and output relationships between the various built-in components of the VHDL language. The dataflow style works fine for small and primitive circuits. But as circuits become more complicated, it is usually advantageous to switch to behavioral style models (this could explain the use of many complex VHDL modules in the previous design methods). The current design method depends on the behavioral style architecture [4] which provides no details as to how the design is implemented in actual hardware. VHDL code written in a behavioral style does not necessarily reflect how the circuit is implemented when it is synthesized. Instead, the behavioral style models how the circuit outputs will react to (or behave) the circuit inputs. It is the VHDL synthesizer tool that decides the actual circuit description. In one sense, behavioral style modeling is the ultimate "black box" approach to designing circuits.

The main idea of the array method is based mainly on storing the CAM data inside data array distributed over the area of the chip. The VHDL statement"array" is used for this purpose. In order to implement the comparison operation (with all the array elements) in one clock cycle, the "Loop" statement together with a "Conditional IF" statement is used in response to a Clock event. On the other hand, normal "read' and "write" CAM operations, is achieved by reading from or writing to the CAM's array. Figure 1 Illustrates the algorithm of the VHDL design.



```
Define array dimensions(n,m)
{Array elements(CAM Words)
}
```

```
for i in 1 to n loop
if(array (i)= data) then
match<= '1'
location <= Convert to STD Logic(i)
else
match<='0
end if
end loop
```

```
if (rdwr ='1') then
i := conv_integer(address)
data <= array(i)
else
i := conv_integer(add)
array(i) <= data
end if
```

Figure 1. The basic structure of the array design method.

3. VHDL Description of the Suggested CAM

The operation of the suggested CAM was described in VHDL and the timing diagram of the different cases is shown in Figure 2. The pin diagram of the suggested CAM is shown in Figure 3, while Table 1 lists the action(s) expected by each pin(s).

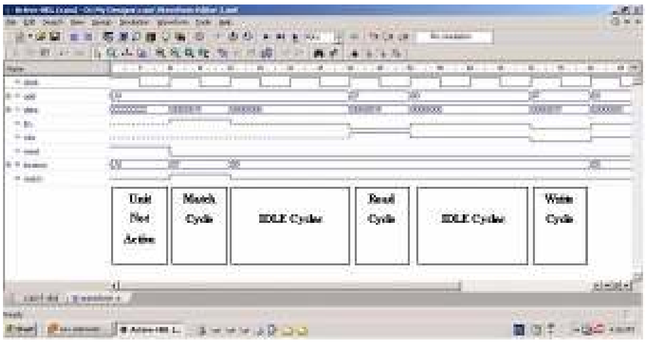


Figure 2. The timing diagram of the suggested CAM.

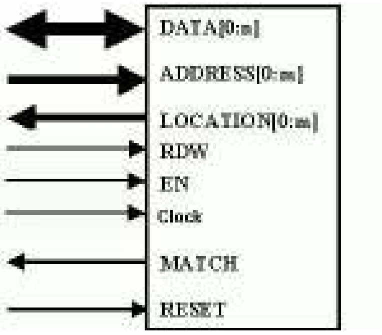


Figure 3. The pin diagram of the suggested CAM.

Table 1. The suggested CAM pin's actions.

Pin	Description	Function
D[0-n]	Data lines of the CAM	Transfer data to/from the unit
A[0-m]	Address lines of the unit	Give the address of an array location
EN	Operational mode of the unit	'1' Match mode '0' Read or Write mode
RDW	Read/Write signal	'1' Read form unit '0' Write to unit
Match	Comparison result	'1' Match found '0' Match not found
Location[0-m]	Match address	Gives the location of the matched word
Reset	Unit activity	'0' CAM is active '1' CAM is not active

The timing diagram shows that the suggested CAM gives the matching result in one clock cycle. Also, any Read or Write operation could be finished in one clock cycle (depending on the width of the system data bus which determines the number of clocks needed to transfer a single word).

4. FPGA Implementation of the Suggested CAM

Experimentally, the suggested CAM design was implemented on Spartan3E (XC3s400e) platform. The performance of the *Array method* was investigated by measuring the area utilization of the device and the maximum frequency against the CAM size. The size of the CAM could be changed by varying either the word length or the number of words (CAM depth). Figures (4 to 7) show the effect of such variations.

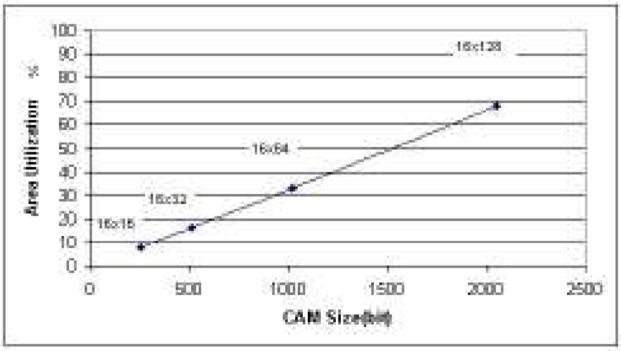


Figure 4. Effect of varying CAM depth on utilization.

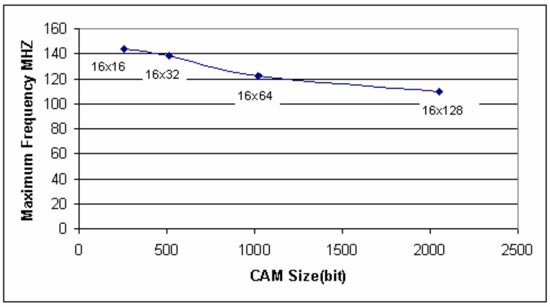


Figure 5. Effect of varying CAM depth on frequency.

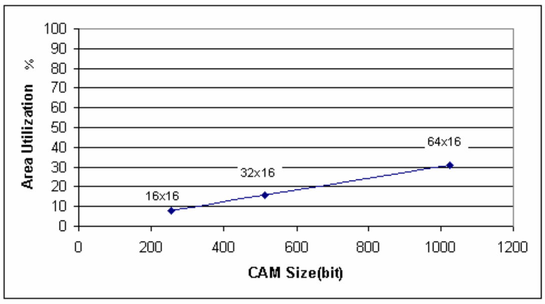


Figure 6. Effect of varying CAM width on utilization.

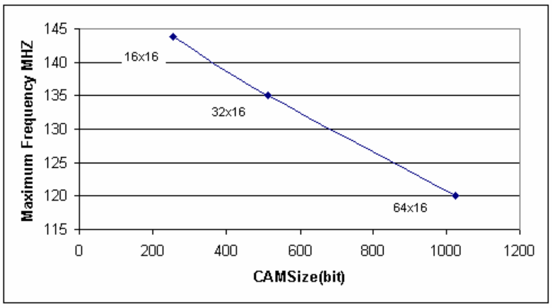


Figure 7. Effect of varying CAM width on frequency.

The following remarks could be extracted from the figures:

1. Area utilization of the device increases with the increment in the CAM size. The calculations show that each stored bit occupies a single slice of the chip.
2. The maximum working frequency decreases with the increment in the CAM size. This could be caused by the increment in the electronic propagation delay results from the additional logic circuits.

In order to optimize the value of the working frequency while changing the word length, Parallel processing method is used. Figure 8 Explains that this method depends on dividing the main array into several simultaneously processed sub-arrays. For example a (64x16) CAM has a maximum frequency value of (120 MHz), when dividing the CAM array into four (16x16) sub-arrays working in parallel, the maximum frequency becomes (143 MHz). Again, the reduction in the maximum signal path and hence, electronic propagation delay, enhances the maximum working frequency of the design.

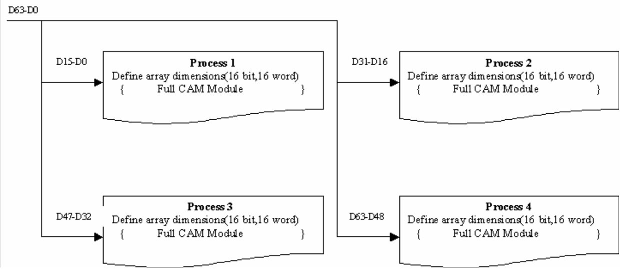

D63-D0
D15-D0
Process 1
Define array dimensions(16 bit,16 word)
{ Full CAM Module }
D31-D16
Process 2
Define array dimensions(16 bit,16 word)
{ Full CAM Module }
D47-D32
Process 3
Define array dimensions(16 bit,16 word)
{ Full CAM Module }
D63-D48
Process 4
Define array dimensions(16 bit,16 word)
{ Full CAM Module }

Figure 8. Parallel processed array method.

5. Comparing the Array Method with Traditional CAM Design Methods

In this section, the performance of the suggested method is compared with the previously mentioned CAM design methods [3, 8]. Table 2 summarizes the comparison results among various methods to build a (32x8 bit) CAM, which was then implemented on a Vertex (XCV50) platform.

Table 2. Comparisons with other design methods.

Array Method (CAM32x8)
Percentage of Slices: 38% Maximum Frequency (for Write & Match) : 108 MHz
SRL16 Method (CAM 32 x 8)
Percentage of Slices: 13% Maximum Frequency : 125 MHz (for Read and Match) and 6.25 MHz for (Write)
Block RAM Method (CAM32x8)
Two block RAMs and 16 slice Maximum Frequency :90 MHz for (read and write) and 222 MHz (Match)
Distributed SelectRAM(CAM256x16)
Percentage of Slices: 5% Maximum Frequency : 83 MHz for (read and write) and 7 MHz for (Match)

It is obvious that the main disadvantage of the investigated method is the high area utilization as compared to the other methods. In order to give a better understanding of the benefits of the current method, the characteristics of the different CAM design methods was summarized in Table 3.

Table 3. The characteristics of the different CAM design methods.

Method	Advantages	Disadvantages
SRL16E	One clock cycle for Match or Read & Flexible CAM size	Require 16 clock cycle for Write & Complex Design Method
Distributed SelectRAM	One Clock Cycle for Read or Write & Used for large CAM size	Require 16 clock cycle for Match, Built for specific application (ATM) & Complex Design Method.
Block RAM	One clock cycle for Match, Read and Two clock cycles for Write, High working frequency & Optimized area utilization	Word length is limited to 8 bit only & Complex Design Method.
Array Method	The easiest design method, Flexible CAM size, one clock cycle for Match & one Clock Cycle for Read or Write.	Relatively high area utilization.

6. Reconfigurable CAM in the Field

In this section, RCAM is used to build several network devices and units. The purpose is to investigate its ability to be a flexible solution in different situations. In all the following designs, parallel processing method is applied to enhance its working frequency with a sub-array word length of (8 bit). The first case (Ethernet Address Filtering) is described in details with the necessary timing diagrams. The other units have similar timing diagrams. For better performance, all the suggested designs were implemented in FPGA using Vertex2 (XC2V2000) platform. Throughput values for all devices are calculated as the product of ((bus width × maximum frequency)/ number of clocks to finish a MATCH cycle).

A. Ethernet Address Filtering Unit

In order to enhance the security of the networks, modern Ethernet switches have the ability to achieve layer 2 packet filtering separately by each of its ports [1]. This is done by supplying these ports with a layer 2 packet filtering unit. The action of this unit is to compare the source MAC address portion of each incoming Ethernet packet with pre-stored addresses. If a match is found, then the packet is passed to the switch, otherwise, it is discarded. In this unit, RCAM is considered to be the heart of the design, see Figure 9.

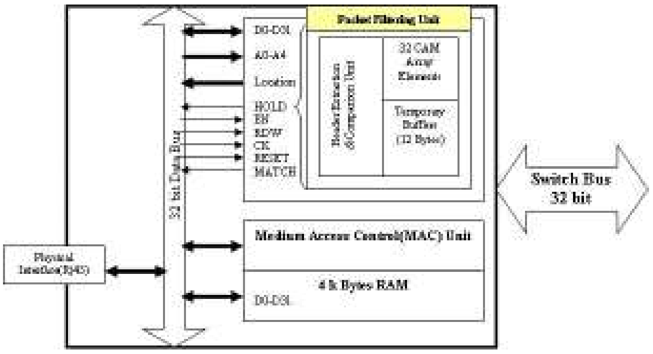


Figure 9. Ethernet port supplied with the suggested address filtering unit.

A VHDL module for RCAM size (48x32) is built to describe the operation of this unit. Layer 2 packet filtering unit may work in one of two modes; programming mode or operation mode:

a. Programming Mode: When the signal on pin EN is '0', packet filtering unit works in the programming mode, in which, MAC addresses stored in the array can be changed (read or written) by the network administrator. The sequence of events in this mode is as follows:

   1. EN (Enable) signal is '0'.
   2. RWD (READ/WRITE) signal is either '1' for reading from the unit or '0' for writing to the unit.
   3. HOLD signal is set to '0' in order to allow the packet transfer operation.

4. The address of the certain location in the array is established.
5. The 48 bits MAC address is moved to/from the unit. It is first stored temporarily in a 48 bit buffer (inside the unit) before moving it to/from the bus.

The above steps require two clock cycles to be finished, because the 48 bit MAC address transferred first as 32 bit then as 16 bit (since the bus width is 32 bit). Figures 10 and 11 show the timing diagram of *Read and Write* operations.

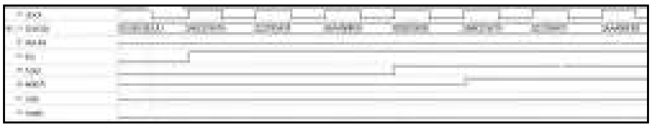


Figure 10. Timing diagram of the *Read* operation.

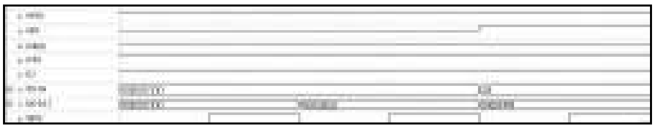


Figure 11. Timing diagram of the *Write* operation.

b. Operation mode: It is the mode in which packet filtering unit performs its packets investigation functions depending on their addresses. This is summarized as follows:

1. Whenever a packet is moved from the buffers inside the port's MAC unit, its source and destination addresses are stored into two 48 bits buffers inside the unit (for the comparison purposes). This operation requires 3 clock periods because the bus is 32 bit width.
2. The packet filtering unit generates a 'HOLD' signal to pause the packet transfer procedure until completing the comparison process.
3. The comparison process between the pre-stored MAC addresses and the source MAC address of the packet is achieved using the Content Addressable Memory (CAM). When the entered MAC addresses match on of the addresses stored in the CAM, the 'Match' signal becomes"1" .which permits to transfer the packet to the switch's buffers. Otherwise, it becomes "0" and discards the packet.
4. The 'HOLD' signal is returned to "0" and 'Reset' signal becomes '1' to deactivate the unit and to allow the completion of the packet transfer operation to its proper stack. The timing diagram of the operation mode is shown in Figure 12.

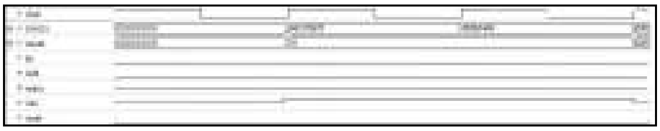


Figure 12. Timing diagram of the *MATCH* operation.

The statistics obtained from the FPGA implementation report using Vertex2 (XC2V2000) platform indicated that Ethernet packet filtering unit occupied (14%) of the chip and it may works at a maximum frequency value of (227 MHz). Throughput of this device is (1 Gbps).

B. WLAN MAC Address Filtering

Whereas one Access Point (AP) or a group of APs can be identified by a Service Set IDentifier (SSID), a client computer can be identified by the unique Medium Access Control (MAC) address of its 802.11 NIC. To increase the security of an 802.11 network, each AP can be programmed with a list of MAC addresses associated with the client computers that are allowed to access the AP. If a client's MAC address is not included in this list, the client is not allowed to associate with the AP. MAC address filtering (along with SSIDs) provides improved security, but it is best suited to small networks where the MAC address list can be efficiently managed. Each AP must be manually programmed with a list of MAC addresses, and the list must be kept up-to-date. In practice, the manageable number of MAC addresses filtered is likely to be less than 255[6].

In this application, RCAM is assumed to be a part of an access point and responsible for the MAC address filtering operation. RCAM is used to store the 48 bit MAC addresses of the authorized clients and it performs a comparison operation with the source address field of any arrived packet. In the VHDL module, the RCAM size is set to be (48(word width) × 128 (CAM depth)) and its data bus is assumed to has (32bit) width. The operation of the MAC address filtering unit begin by receiving the required *source address* (as extracted by the WLAN port) field from the incoming *Request to Send* (RTS) packet as shown in Figure 13.

Frame Control (2 Bytes)	Duration (2 Bytes)	Destination Address (6 Bytes)	Source Address (6 Bytes)	CRC (4 Bytes)

Figure 13. Frame format of RTS packet.

This step requires 2 clock cycles. Then, the extracted field is compared with all the pre-stored MAC values to discover its acceptance statues. Meanwhile, the packet transfer operation is paused by the 'HOLD' signal from the MAC address filtering unit. After finishing the comparison procedure (in *one* clock cycle), the filtering unit either generates an accept (MATCH signal is "1") or not-accept (MATCH signal is "0") signal to decide accepting or rejecting the join request packet. The unit may work in another mode, in which, permitted MAC address values is written to (or read from) the CAM's array. This mode is initiated when the 'EN' input signal is set to '0'. In this mode, two clock cycles is needed to finish a single read/write process. The block diagram of the proposed unit is shown in Figure 14.

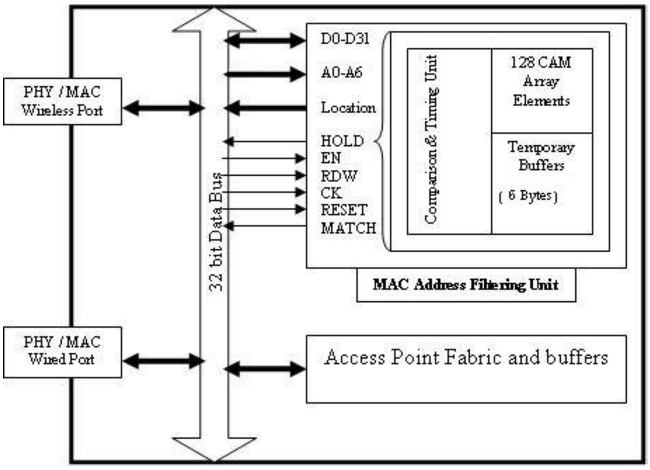


Figure 14. WLAN access point supplied with the suggested address filtering unit.

The comparison and timing unit is responsible for the management of the source address reception procedure (into the temporary buffers), handling the Read/Write procedures to/from the CAM's array and performing the comparison operation.

Running the VHDL simulation shows that it needs an extra three clock cycles to handle packet checking procedure. The proposed MAC address filtering unit is implemented in FPGA using Vertex2 (XC2V2000) platform. The statistics obtained from the implementation report indicated that MAC address filtering unit occupied (57 %) of the chip and it may works at a maximum frequency value of (200 MHz) with a throughput value of (2.13 Gbps).

C. Firewall on Chip (FoC) Unit

A firewall is a set of related devices, deployed strategically between a company's private network and one or more unsecured networks (particularly including the Internet). A firewall functions to protect the resources of the private network from users on the outside (external) networks. The first line of defense in firewall protection, and the most basic, is the packet filter firewall. Packet filters examine incoming and outgoing packets and apply a fixed set of rules to the packets to determine whether they will be allowed to pass. The packet filter firewall is typically very fast because it does not examine the data in the packet. It simply examines the type of the packet along with the source and destination IP addresses (32 bit each) and port numbers (16 bit each), and then it applies the filtering rules [9]. Firewall on Chip (FoC) is a packet filtering unit consists of a single chip only. This unit could be a part of a larger network device such as switches or routers [1]. FoC could be designed to have a similar procedure to that mentioned earlier in this section.

A VHDL module RCAM size (96x64) is built to perform the actions of a FoC device. It was assumed to have a system data bus of 32bit width. The RCAM is configured to have 64 filtering rules (CAM depth), each rule has a 96 bit length (64 bit for both source and destination IPs and 32 bit for both source and destination ports). For design simplicity, it was assumed that 'ANDing' is the logic relation between the various parts of the rule. This allows treating any rule as one block rather than separated fields. Also, it was assumed that the Store and Forward switch (or router) is a part of an Ethernet network which has the header shown in Figure 15.

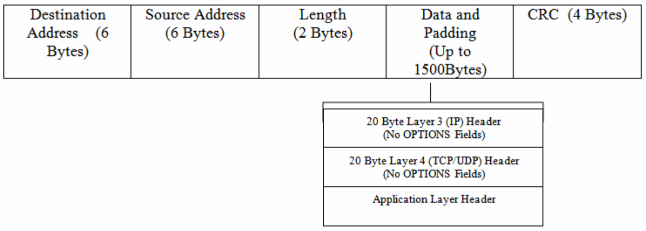

Destination Address (6 Bytes)
Source Address (6 Bytes)
Length (2 Bytes)
Data and Padding (Up to 1500Bytes)
CRC (4 Bytes)
20 Byte Layer 3 (IP) Header (No OPTIONS Fields)
20 Byte Layer 4 (TCP/UDP) Header (No OPTIONS Fields)
Application Layer Header

Figure 15. Ethernet frame format.

The operation of the FoC unit begins on the reception of any packet, and prior to sending it to the switch's buffers. The required fields (Destination and Source IPs, Destination & Source ports) is extracted from the different headers and stored in a temporary buffer inside FoC unit. This step requires 10 clock cycles, after which a HOLD signal is generated to pause the packet transfer procedure. After completing the comparison process (in one clock cycle), FoC either generates a '1' MATCH signal (for access) or '0' MATCH signal (for deny). Also, FoC may work in the programming mode, in which rules are read/written to/from the FoC and needs 3 clock cycles to be finished. Figure 16 Shows the block diagram of the proposed FoC unit.

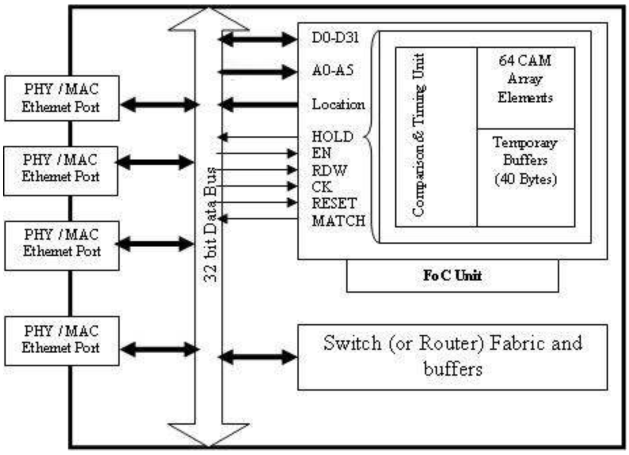


Figure 16. Block diagram of the suggested FoC.

The proposed FoC unit is implemented in FPGA using Vertex2 (XC2V2000) platform. The statistics obtained from the implementation report indicated that FoC unit occupied (57 %) of the chip and it may works at a maximum frequency value of (206 MHz) with a throughput value of (314 Mbps).

D. QoS Packet Classification Unit

Packet classification is crucial for Quality of Service (QoS) because it enables the switch or router to differentiate the traffic streams and treat them differently depending on their individual requirements. All QoS is based on a classification scheme – the more thorough and flexible the classification capabilities, the more advanced the capability to support QoS. Any special treatment of the traffic (prioritization, scheduling, bandwidth distribution, filtering) may be based on the result from the classifier[12].

The higher the throughput of the switch, the more flows there are that require identification for this special treatment. The benefit of the classification scheme rapidly diminishes if the implementation creates its own bottleneck. Classification can be divided into two parts; data extraction, where the relevant fields are extracted from the packet header, and data comparison, where the extracted fields are compared to predefined data. Once a packet can be assigned to a flow, it can be forwarded and queued with an associated class of service that may be defined on a per flow basis. A flow can be uniquely identified by a 4-tuple consisting of source IP address (32 bits), source TCP or UDP port (16 bits), destination IP address (32 bits), destination TCP or UDP port (16 bits) and Type of Service (TOS) (8 bits). This represents a minimum of 104 bits to uniquely identify a flow in IPv4[12].

From the above discussion, it is obvious that RCAM is an essential part in the QoS Packet Classification Unit. It was assumed that the switch's (or router's) ports (in which the Qos units is localized) has 16 priority queues. Our VHDL module is modified to have a (104 × 16) CAM dimensions with a system bus has a 32bit width. Figure 17 shows the Block diagram of the proposed QoS Packet Classification Unit.

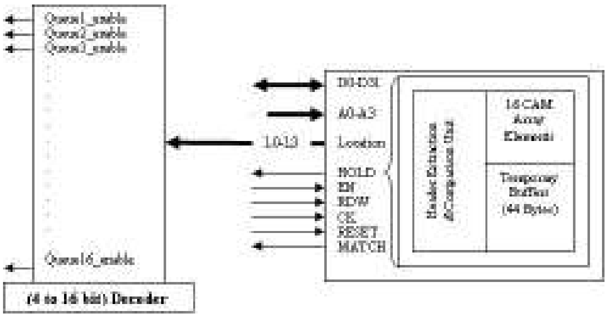


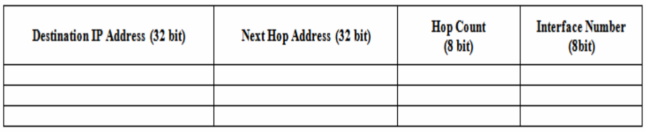
Figure 17. Block diagram of the suggested QoS packet classification unit.

The QoS unit task begin when the switch transfer a packet to one of its ports. The unit receives the packet and stores it temporarily to find its destination queue. If a 'Match' is found , then the (4 to 16 bit) decoder enables one of the queues (depending on the 'Location' value) and start the packet transfer operation to that queue. Otherwise, the packet is transferred to the lowest priority queue. Running the simulation indicates that 12 clocks is needed by the QoS unit (one for comparison and 11 for packet transfer operation to the temporary buffers) to finish packet directing operation. Also , the unit may works in the programming mode, in which rules is read/written to/from the unit's array and needs 4 clock cycles to be finished.

After implementing the proposed QoS unit in FPGA using Vertex2 (XC2V2000) platform, it was found that the unit occupies (16%) of the chip and it may works at a maximum frequency value of (232 MHz) with a throughput value of (322 Mbps).

E. Routing Table Search Unit

The routing table is used by the routing module to determine the next-hop address of the packet. Every router keeps a routing table that has one entry for each destination network. The entry consists of the destination network IP address, the shortest distance to reach the destination in hop count, and the next router (next hop) to which the packet should be delivered to reach its final destination, as shown in Figure 18. The hop count is the number of networks a packet enters to reach its final destination. In an Routing Information Protocol(RIP) algorithm, this value does not exceed 255. A router should have a routing table to consult when a packet is ready to be forwarded[9].



Destination IP Address (32 bit)	Next Hop Address (32 bit)	Hop Count (8 bit)	Interface Number (8bit)

Figure 18. Routing table fields.

In order to get a fast routing decision , content addressable memory must be used. Only the destination IP address is stored in the CAM, and it is known here as the search key. The search key is simultaneously compared with all CAM entries and the highest priority match is returned. The corresponding next hop and interface number can then looked up in an associated data RAM, see Figure 19.

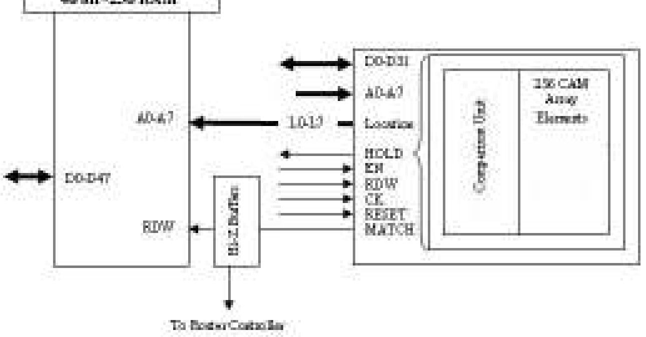


Figure 19. Block diagram of the suggested routing table search unit.

CAM VHDL module is modified to handle the above situation. The array dimensions was set to be (32 bit × 256) without temporary buffers. When a packet arrived, the header extraction process is done by the router's fabric and the corresponding destination IP address is compared with all CAM elements in one clock cycle. If a 'Match' is found, then the routing table contents is read from the RAM. Otherwise, the Hi-Z buffers become inactive and a signal 'Destination Not Found' is sent to the router's controller. Our simulation results shows that it needs three clock cycles to give the routing decision (one for CAM operation and two for RAM response). In the programming mode operation, both CAM and RAM are activated to store the routing table contents and it needs 3 clock cycles to finish this operation (one for storing IP value in the CAM and two for writing the rest of the table in the RAM).

After implementing the proposed searching unit in FPGA using Vertex2 (XC2V2000) platform, it was found that the unit occupies (76%) of the chip and it may works at a maximum frequency value of (190 MHz) with a throughput value of (2 Gbps).

F. Network Intrusion Detection System (NIDS) Search Unit

Network Intrusion Detection Systems (NIDS) are an important tool to protect network systems from external attack. NIDS are used to identify and analyze packets that may signify an impending threat to organization's network [15]. This could be done by storing special signatures (which represent a threat indicator) then comparing the data of the incoming packets against these signatures. Inspecting incoming packets for tell-tale signatures can be time consuming especially if the number of possible signatures is large[15]. In this manner, RCAM could be used to accelerate the keyword match operation. In our design, we reconfigure the RCAM to work as the search engine of an NIDS unit. It was configured to have the size of (80bit x 64 words), which represents a 47 Snort Web Attacks rules with an average of 10 character per word[15]. We assumed that RCAM operation is assisted by other units which receive packets, extract the data words and characters, then applying each word to the RCAM to be compared with its contents, as shown in Figure 20.

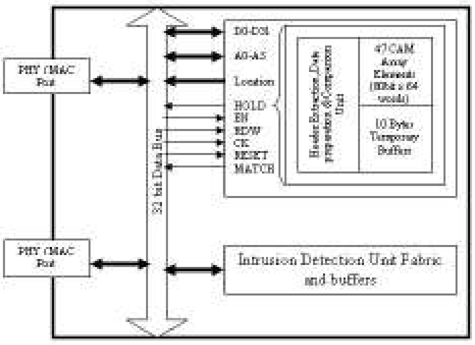

PHY (MAC Port
PHY (MAC Port
32 bit Data Bus
D0-D31
A0-A5
Location
HOLD
EN
RDW
CK
RESET
MATCH
Header Extraction_Data preparation &Comparison Unit
47 CAM Array Elements (80bit x 64 words)
10 Byte Temporary Buffer
Intrusion Detection Unit Fabric and buffers

Figure 20. Block diagram of the suggested NIDS.

The data comparison begins after storing the intended word inside the (10 Bytes) temporary buffers, and needs 4 clock cycles to be finished (3 clocks for transferring the 80 bit word through 32 bit data bus and one clock for *Match* operation). In the same manner, it takes 3 clock cycles to finish Read/Write operation of the rules. The statistics obtained from the FPGA implementation report indicate that RCAM unit occupied (48 %) of the chip area and it may works at a maximum frequency of (236 MHz) with a throughput value of (1.89 Gbps).

7. Comments on the Implementations

The successful implementations of various CAM sizes to meet the needs of different network devices, candidate the suggested method to take its place as a flexible and easy to adopt CAM design method. Also, its high throughput values allows to build high response devices suitable for working in high speed networks. Table 4 summarize the performance of the array method when worked as a part of different network devices.

Table 4. Different CAM implementations on network devices.

Network Device	CAM Size (Bit)	Area Utilization	Maximum Frequency (MHz)	Throughput (Gbps)
Ethernet Address Filtering Unit	48x32	14%	227	1
WLAN MAC Address Filtering	48x128	57%	200	2.13
Firewall on Chip (FoC) Unit	96x64	57%	206	0.314
QoS Packet Classification Unit	104x16	16%	232	0.322
Routing Table Search Unit	32x256	76%	190	2
Network Intrusion Detection System(NIDS)Search Unit	80x64	48%	236	1.89

8. Conclusions

In this paper, a new CAM design method is presented. The operation of the design was described in VHDL and implemented using FPGA technique. The main features of this method are its simplicity, flexibility and high response. Different configurations of RCAM are used to design various network devices used in many applications which proves its remarkable design flexibility and high system throughput. This method could be used to build small to medium size CAMs. However, the rapid development in the FPGA manufacturing and their increased capacities would allow to use this method in designing large sized CAMs.

References

[1] Bandi N., Schneider S., Agrawal D., and ElAbbadi A., "Hardware Acceleration of Database Operations Using Content Addressable

Memories," *in Proceedings of the 1st International Workshop on Data Management on New Hardware*, Baltimore, Maryland, USA, 2005.

[2] Brelet J., *Designing Flexible, Fast CAMs with Virtex Family FPGAs*, Xilinx Inc., 1999.

[3] Brelet J., *Using Block RAM for High Performance Read/Write CAMs*, Xilinx Inc., 2000.

[4] Brown S. and Vranesik Z., *Fundamentals of Digital Logic with VHDL Design*, McGraw Hill Including, 2000.

[5] Cole E., Krutz R., and Conley J., *Network Security Bible*, Wiley Publishing Inc., 2005.

[6] Defossez M., *Content Addressable Memory (CAM) in ATM Applications*, Xilinx Inc., 2001.

[7] Khan J. and Khwaja A., *Building Secure Wireless Networks with 802.11*, Wiley Publishing, 2003.

[8] Lakshminarayanan K., Rangarajan A., and Venkatachary S., "Algorithms for Advanced Packet Classification with Ternary CAMs," *in Proceedings of SIGCOMM'05 Conference,* Philadelphia, Pennsylvania, USA, pp. 193-204, 2005,

[9] Lockwood J., Naufel T., Turner J., and Taylor D., "Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX)," *in Proceedings of ACM International Symposium on Field Programmable Gate Arrays (FPGA)*, USA, pp. 87-93, 2001.

[10] Moss B., "An Egress Queue Manager for 8x100 Mbps and 1x1Gbps Ethernet Switch Port Controllers," *Msc Thesis*, Simon Fraser University, 2002.

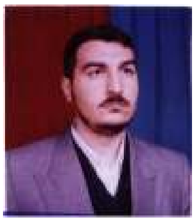
[11] Pagiamtzis K. and Sheikholeslami A., "Content-Addressable Memory Circuits and Architectures: A Tutorial and Survey," *Computer Journal of IEEE Journal of Solid State Circuits*, vol. 41, no. 3, pp. 712-727, 2006.

[12] Paul S. and Bhunia S., "Reconfigurable Computing Using Content Addressable Memory for Improved Performance and Resource Usage," *in Proceedings of Design Automation Conference Conference*, California, USA, pp. 786-791, 2008.

[13] Sanguinetti J. and Pursley D., "High-Level Modeling and Hardware Implementation with General Purpose Languages and High level Synthesis," *White Paper*, Forte Design Systems, 2002.

[14] Taylor D., Turner J., Lockwood J., Sproull T., and Parlour D., "Scalable IP Lookup for Internet Routers," *Computer Journal of IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 522-534, 2003.

[15] Yi S., Kim B., Oh J., Jang J., Kesidis G., and Chita R., "Memory Efficient Content Filtering Hardware for High Speed Intrusion Detection Systems," *in Proceedings of SAC'07 Conference,* Seoul, Korea, pp. 264-269, 2007.



Qutaiba Ali received the BS and MS degrees from the Department of Electrical Engineering, University of Mosul, Iraq, in 1996 and 1999, respectively. He received his PhD degree (with honor) from the Computer Engineering Department, University of Mosul, Iraq, in 2006. Since 2000, he has been with the Department of Computer Engineering, Mosul University, Mosul, Iraq, where he is currently an assistant professor. His research interests include computer networks analysis and design, embedded network devices, and network security. He instructed many topics (for post and undergraduate stage) in computer engineering field during the last ten years and has many publications in numerous journals and conferences. He acquire many awards (form National Instrument INC) and appreciations form different parties for excellent teaching and extra scientific research efforts. Also, he was invited to join many respectable scientific organizations such as IEEE, IENG ASTF, WASET and many others. He was participate as technical committee member in three IEEE conferences in USA, Malaysia, and Egypt and joined the editorial board of five scientific international journals.