FPGA-BASED SYSTEM FOR NETWORK FLOW IDENTIFICATION

Z. Baruch, A. Peculea, M. Suciu, Z. Majo

Computer Science Department, Technical University of Cluj-Napoca,
26-28, Bariţiu St., 400027 Cluj-Napoca, Romania
E-mail: Zoltan.Baruch@cs.utcluj.ro, Adrian.Peculea@cs.utcluj.ro

Abstract: *This paper presents the design and implementation of an FPGA-based system for real-time network flow identification. The system identifies data flows based on packet inspection. The main advantage of this system is that it reduces significantly the processing time required for the flow identification. For the hardware implementation, a Xilinx Virtex-II Pro FPGA device and the Xilinx Embedded Development Kit (EDK) software are used. This embedded system represents the first step for designing a reconfigurable router with QoS (Quality of Service) support.*

Keywords: *FPGA, computer networks, flow identification, QoS, embedded systems*

1. INTRODUCTION

A network flow represents a data stream carrying information between a server and a client. Network flow identification is essential to support Quality of Service (QoS) implementation. For instance, it helps identifying the applications that present certain QoS constraints. Also, it allows determining the traffic characteristics in order to evaluate the required QoS policies and it helps network traffic monitoring to examine its changing tendencies. Since the number of flows forwarded by a router is very large, the flow identification and processing presents real-time constraints. These constraints can be completely met by means of hardware implementation.

In this paper we describe the design and implementation of an FPGA-based system for network flow identification. This system identifies flows based on five fields in the packet headers. The implemented system represents the first step for designing a reconfigurable router with QoS support. Prior to the hardware implementation, we developed a software application to test several network flow identification methods.

This paper is organized as follows. Section 2 provides background information regarding flow identification. Related works are described in Section 3. Section 4 describes the software application for network flow identification. Section 5 presents the design and implementation of the FPGA-based system for network flow identification, and Section 6 concludes the paper.

2. BACKGROUND

Network flow identification can be used for several goals, such as determining the packet processing method and traffic distribution. QoS technologies treat differently packets that belong to different flows. Therefore, it is important to identify various types of packets by inspecting their contents. In case of integrated services, traffic identification consists of traffic flow identification based on the headers' contents. Typically, the following information is used: source IP address, destination IP address, protocol identifier, source port number, and destination port number. In case of differentiated services, traffic identification represents the process of packet identification based on a set of specified rules.

There are several categories of flow identification algorithms. Linear search algorithms use a list of rules stored in descending priority order. A packet is compared sequentially to each rule until a rule that complies with all the fields of the packet is found. The hierarchical tries algorithm is based on several multidimensional hierarchical search trees. The set-pruning tries, grid-of-tries, AQT and FIS algorithms are also based on search trees. The hierarchical intelligent cuttings algorithm [4] uses a search tree that contains a small number of rules in the leaves. These rules can be compared sequentially. The tuple-space search algorithm [10] divides the classification into an exact number of steps. The set of rules mapped to the same field is stored in a hash table. The bitmap intersection method [6] is based on the fact that the set of rules used for packet identification represents the intersection of a number of rules associated to each dimension.

The CAM method uses a ternary associative memory that stores the rules in descending priority order and performs the comparisons in parallel. As opposed to random-access memories (RAMs), in which the stored data are identified by means of a unique address assigned to each data word, CAM words are identified by their content. CAMs are very useful in applications where intensive search operations are to be performed [2]. Based on the values they can store, there are two types of CAMs: binary and ternary [8]. Binary CAMs can only store binary digits ('0', '1'), while ternary CAMs can store binary digits as well as "don't care" values ('X'). Ternary CAMs may have a global mask as well, which allows the search pattern to also contain "don't care" values. This is useful when the width of the search pattern is small, so that two or more entries can be stored in the same CAM location. Several networking applications have been identified for using CAMs, including Ethernet address lookup, address filtering, routing, security, or information encryption on for high-performance data switches, firewalls, bridges, and routers [1].

The main technologies used for QoS are Integrated Services, Differentiated Services, Multiprotocol Label Switching, and traffic engineering. For integrated services, the following flow identification algorithms are used: CAM-based search, hashing-based schemes and binary search. Hashing-based schemes involve calculation of a hash function and further comparisons if there is collision. For differentiated services, three types of packet identification algorithms are used: caching, geometrical, and tries. The caching method uses a cache memory to store the information that defines the last flows. The geometrical method involves localization of a point in a multidimensional space. The tries method is based on a binary tree with each branch labeled with 0 or 1 [11]. For traffic distribution, two types of algorithms are used: direct hashing and table-based hashing. In case of direct hashing, a hash function is applied that may include as information used for flow identification at least two of the five fields that identify a flow. The main drawback of direct hashing is that the traffic is distributed evenly. Table-based hashing eliminates this drawback, allowing traffic distribution in ratios defined by the user. This method first divides the traffic and then it maps it to the output links based on an allocation table. The allocation table determines the percentage of traffic that will be sent to an interface.

3. RELATED WORK

To our knowledge, there is no hardware system designed specifically for network flow identification described in the literature. However, several research topics use FPGA devices and content-addressable memories (CAMs) for network processing applications. One type of such application is packet classification, where the packets are compared against a set of filters. For instance, packets may be classified based on the header fields and some strings in the packet con-

tent. Song and Lockwood [9] proposed a packet classification architecture called BV-TCAM (*Bit Vector Ternary CAM*), which combines the BV algorithm and a TCAM. The hardware implementation of the BV algorithm performs source and destination port lookup, while the other fields are checked for matching by a TCAM. Ditmar [3] uses fixed-length and variable-length CAMs for IP characterization, which is closely related to packet classification. Another type of networking application is represented by deep packet inspection, which examines the entire packet content and not just the header. Yu [15] proposed a TCAM-based architecture for deep packet inspection that uses multi-match classification. In this type of classification, packets are compared against a set of filters and each result is reported. CAMs are also suitable for network flow analysis and monitoring. Luk *et al.* [7] proposed a combined hardware-software architecture for network flow analysis, which allows for processing multiple flows in parallel.

4. NETWORK FLOW IDENTIFICATION APPLICATION

In order to test the network traffic identification algorithms, we implemented a software application for traffic flow identification. The flow identification is performed based on five fields contained in the packet header. These fields are the following: source IP address, destination IP address, protocol identifier, source port number, and destination port number. The values of these five fields from a packet are compared against the values of the same fields that define the previously identified flows to determine whether the packet belongs to a flow that is already stored or it belongs to a new flow.

The main design problem of a real-time traffic flow identification application is the comparison of the captured packet information against the information stored about the already identified flows. At the router level, the number of flows is very high and the processing time must be extremely low.

For real-time flow retrieval, a method based on hash function has been used. For this method, an optimized algorithm has been implemented, which allows to reduce the time required for flow retrieval due to the small number of iterations.

The application identifies the flows, stores them, and elaborates statistics about the characteristics of these flows. In order to capture the packets from the entire subnet, the application should run on the router. However, since the router is usually a dedicated equipment, a hub or switch may be used, as presented in Fig. 1.

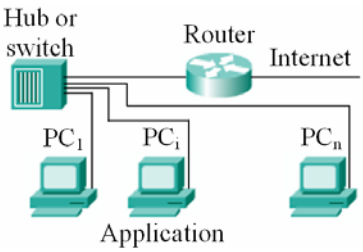

Fig. 1. Network configuration for testing the software application.

A hub transmits each received packet to all the other ports. A switch must be configured to transmit the packets from all the other ports to the port connected to the computer that runs the application.

In order to capture all the packets transferred in the subnet, the network interface card is set to promiscuous mode. Each packet is inspected and the fields that identify the flow to which the packet belongs are extracted from its headers. The next step is to compare the identified flow against the existing flows that are stored in a table. If the flow is new, it is inserted into the table. If the packet belongs to an existing flow, the statistic data of the flow are updated.

The graphical interface of the application displays the flow number as it was identified and the values of the fields used for flow identification. Other information displayed is the maximum length, average length, minimum length of the packets, and the total number of packets belonging to each flow. The information required to evaluate the performance of the flow identification algorithm, represented by the number of iterations required to retrieve the flow, is also displayed, as shown in Fig. 2.

The application has been developed for the .NET platform in the C# language. It allows visualizing the network flows, their characteristics, and the performance of retrieval methods. The application also allows testing various algorithms and optimization techniques that allow for real-time processing of data flows in computer networks in order to support QoS implementation.

Flow no.	Max. len.	Avg. len.	Min. len.	Packet no.	Iter. no.
41	100	100	100	1	2
42	291	283	251	5	1
43	72	72	72	2	1
44	251	251	251	1	1
45	72	72	72	2	2
46	244	129.3333	72	3	1
47	230	111.5	72	4	1
48	232	104.4	72	5	1
49	238	105.2	72	5	1
50	232	104	72	5	1
51	238	113.5	72	4	1
52	118	118	118	1	1
53	118	118	118	1	2
54	234	104.4	72	5	1
55	228	103.2	72	5	1

Fig. 2. Statistics data that characterize the flows.

5. FPGA-BASED SYSTEM FOR NETWORK FLOW IDENTIFICATION

For the implementation of the network flow identification system, we used a Xilinx XC2VP30 FPGA device from the Virtex-II Pro family. This device contains an array of 80x46 configurable logic blocks and integrates two PowerPC 405 processors.

As hardware development platform, we used a Xilinx XUP Virtex-II Pro board [14], which allows implementing complex hardware systems, containing one or more central processing units (CPUs), configurable logic and a large number of peripheral devices that communicate with the CPUs. The main components available are the following: a high-speed Ethernet interface, three serial ATA ports, two RS-232 serial ports, two PS/2 ports, a DDR SDRAM of 256 MB, a serial EEPROM to store the FPGA device configuration, an XSGA video interface, a USB 2.0 interface, and a serial port for debugging.

For the implementation we chose a combined hardware/software solution, which allows exploiting the advantages of both solutions: the flexibility of the software solution and the high performance of the hardware solution. The time-critical operations are implemented in hardware, while the operations that are less demanding are implemented in software.

As software development system, we used the Xilinx Embedded Development Kit (EDK), version 7.1i [13], which allows to implement complex systems containing both hardware and software modules, and provides tools for testing and debugging the designed system. The EDK also requires the corresponding version of the Xilinx ISE design package to generate the configuration bitstream for the FPGA device.

The block diagram of the network flow identification system is illustrated in Fig. 3. The main components of the system are the MicroBlaze processor, the dual-port BRAM (Block RAM), the BRAM controllers, the EMAC (Ethernet Media Access Controller), the INTC interrupt controller, the UART controller for the serial interface, the CAM, and the MDM (Microprocessor Debug Module).

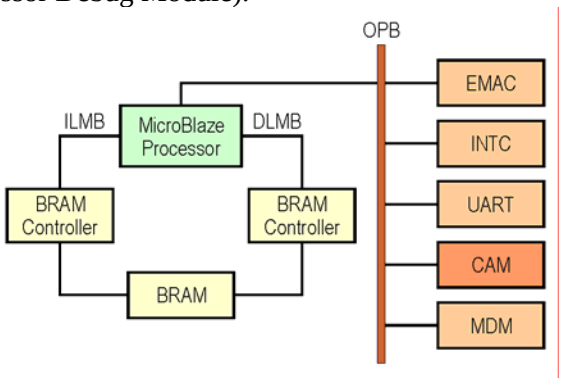

Fig. 3. Block diagram of the network flow identification system.

The MicroBlaze is a 32-bit soft processor core provided by Xilinx EDK. The on-chip dual-port BRAM is connected to the processor using an Instruction-side Local Memory Bus (ILMB) and a Data-side Local Memory Bus (DLMB). These buses use a simple synchronous protocol to provide single-cycle access to the on-chip BRAM. The peripheral modules are connected to the processor using an On-chip Peripheral Bus (OPB). These buses are part of the CoreConnect architecture standard specified by IBM [5].

The CAM performs the comparison between the fields of data packets that uniquely identify a certain flow between two computers. The comparison is performed in parallel for all the words in the CAM, which allows obtaining the result of the comparison in a single clock cycle. An extra clock cycle is required to write the contents of the packet fields into the argument registers. The CAM word size is 104 bits: 2×32 bits for the source and destination IP addresses, 2×16 bits for the source and destination ports, and 8 bits for the protocol identifier. For testing, we used a 16-word CAM in order to reduce the synthesis time. After testing and debugging, the size of the CAM has been extended to 512 words. Above this size, the synthesis time and the memory requirements increase significantly.

Fig. 4 illustrates the block diagram of the CAM. The processor accesses the CAM through five 32-bit software-addressable registers, Reg0 to

Reg4. The first four registers can only be used for writing. Before the search operation, the source IP address is written into Reg0, the destination IP address is written into Reg1, the source and destination port numbers are written into Reg2, and the protocol ID is written into Reg3. For a write operation, the write address is written into Reg3 and the WE bit of this register is set. Reg4 contains the result of the search and it can only be used for reading. Bit 31 of this register represents the match bit. If this bit is set, bits 8..0 of Reg4 contain the match address.

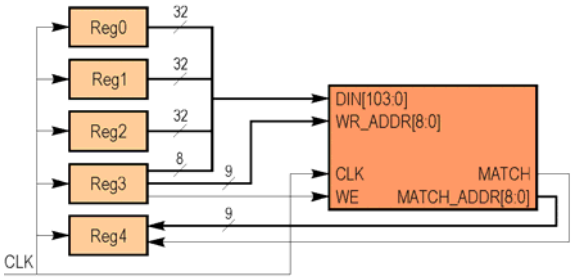

Fig. 4. Block diagram of the CAM.

The CAM is connected to the on-chip peripheral bus through an interface called OPB IPIF (OPB Intellectual Property Interface), as illustrated in Fig. 5. Part of this interface, the OPB IPIC (OPB Intellectual Property Interconnect) is generated automatically by the Core Generator module from the Xilinx ISE software. The other part of the interface has been written in VHDL.

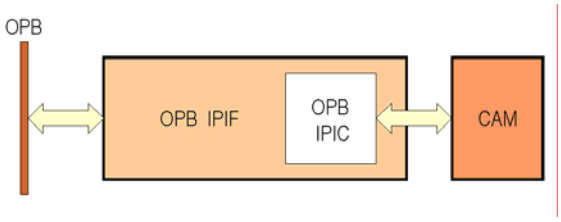

Fig. 5. The interface between the CAM and the OPB.

Table 1: Synthesis Results for the Network Flow Identification System

Resource Type	Used	Total	Percentage
Slice registers	3,533	27,392	12%
Occupied slices	4,197	13,696	30%
Block RAMs	48	136	35%
4-Input LUTs	6,048	27,392	22%

Table 1 contains some results from the reports generated by the synthesis tools used to implement the embedded system in the Xilinx XC2VP30 FPGA device.

For the CAM, a software driver has been developed, which has been written in C. The main functions that can be performed with this driver are the following: searching an argument word in the CAM, reading from the CAM and writing into the CAM.

The MicroBlaze processor executes a program that initializes and controls the hardware modules. In order to be able to detect all the packets that travel in the network, the program sets the EMAC into promiscuous mode. To initialize the interrupt system, the program uses the EMAC module's driver to register the callback functions required to service the interrupts. When a frame is sent or received, or when an error occurs, the corresponding interrupt function is called. In the main loop, the program extracts the required fields from the data packets, stores them into the argument registers, and sends the command to the CAM. The program also performs the required operations for managing the data flows and displaying the statistics about the identified flows.

For testing the hardware system, we connected the development board to a hub and monitored the traffic between a server and a PC connected to the same hub. This connection allows monitoring all the packets that travel between the server and the PC.

6. CONCLUSIONS

In this paper, we presented the design and implementation of an FPGA-based system for network flow identification. The flows are identified based on five fields contained in each packet header. For the implementation, we used a Xilinx Virtex-II Pro FPGA device. The hardware system contains a CAM that performs the comparison between the five fields of incoming packets and the previously stored information about the identified flows. Only two clock cycles are required to write the contents of the packet fields into the argument registers and to retrieve the result of the comparison. The main advantage of the hardware implementation is that it reduces significantly the time required for flow identification. This implementation, how-

ever, does not represent a pure hardware system, but rather a combined hardware/software design. Only the time-critical operations are implemented in hardware, while other operations are implemented in software. The advantage of this combined solution for this real-time flow identification system is that flexibility is increased and the hardware resources required are reduced.

The hardware implementation of the network flow identification system represents the first step for designing a reconfigurable router with QoS support. The next step would be the design of an embedded system that determines the network traffic characteristics, and the last step would be the design of the reconfigurable router that performs the constraint-based routing.

REFERENCES

[1] Azgomi, S., "Using content-addressable memory for networking applications", http://www.commsdesign.com/main/1999/11/9911feat3.htm, accessed February 2006.

[2] Baruch, Z., Structure of Computer Systems, U.T. PRES, Cluj-Napoca, Romania, pp. 174-185, 2002.

[3] Ditmar, J. M., A Dynamically Reconfigurable FPGA-based Content Addressable Memory for IP Characterization, Master Thesis ELE/ESK/2000-3, Kungliga Tekniska Högskolan, Stockholm, 2000.

[4] Gupta, P., and McKeown, N., "Packet classification using hierarchical intelligent cuttings", in *Proc. Hot Interconnects* VII, Stanford, 1999.

[5] IBM Corp., On-Chip Peripheral Bus, Architecture Specifications, Version 2.1, 2001.

[6] Lakshman, T. V., and Stiliadis, D., "High-speed policy-based packet forwarding using efficient multidimensional range matching", in *Proceedings of ACM SIGCOMM*, pp. 191-202, 1998.

[7] Luk, W., Yusuf, S., Sloman, M., Brown, A. W., Lupu, E. C., and Dulay, N., "Combined Hardware-Software Architecture for Network Flow Analysis", in Proceedings of Int. Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA), pp. 149-155, Las Vegas, 2005.

[8] Pagiamtzis, K., "Content-Addressable memory introduction", http://www.eecg.toronto.edu/~pagiamt/cam/camintro.html, February 2006.

[9] Song, H., Lockwood, J. W., "Efficient Packet Classification for Network Intrusion Detection Using FPGA", in *Proceedings of International Symposium on Field-Programmable Gate Arrays* (FPGA '05), pp. 238-245, Monterey, 2005.

[10] Srinivasan, V., Suri, S., and Varghese, G., "Packet classification using tuple space search", in *Proceedings of ACM SIGCOMM*, pp. 135-146, 1999.

[11] Srinivasan, V., and Varghese, G., "Fast address lookups using controlled prefix expansion", in *ACM Transactions on Computer Systems*, Vol. 17, No. 1, pp. 1-40, 1999.

[12] Wang, Z., Internet QoS: Architectures and Mechanisms for Quality of Service, Morgan Kaufmann, San Francisco, 2001.

[13] Xilinx Inc., Embedded System Tools Reference Manual, Embedded Development Kit 7.1i, UG111 (v4.2), 2005.

[14] Xilinx Inc., Xilinx University Program Virtex-II Pro Development System, Hardware Reference Manual, UG069 (v1.0), 2005.

[15] Yu, F., High Speed Deep Packet Inspection with Hardware Support, Technical Report No. UCB/EECS-2006-156, University of California at Berkeley, 2006.