Design of High-speed Dynamic Packet Filtering Firewall for IPv6 based on FPGA

BING LI, PENGCHENG CAI, XIN GUO, LONGFEI ZHANG
School of Integrated Circuits
Southeast University
Sipailou No.2, Nanjing, Jiangsu Province, China
CHINA

pc229c@163.com

Abstract: Network security issues are becoming increasingly acute. A computer network exposed to the Internet without any security protection is at great risk. This paper proposes the packet filtering firewalls system based on FPGA. This system connects the high-speed firewall filtering module and the CPU (OR1200). The Internet Protocol Version 6 (IPv6) network data package is sent to test successfully by the PC client. Experimental results show that the system process the whole IPv6 data packet with 920ns and the complete design uses a small portion of the Altera StratixII/ EP2S60F672C5 FPGA, 25% of the logic blocks and 11% of the memory blocks. So the system not only has ideal functionality but greatly improves the data transmission speed.

Key words: network security; IPv6; high-speed data package filtering; FPGA

1. Introduction

This paper studies the basic technology principles of dynamic packet filtering and proposes a dynamic packet filtering module with protocol filtering, port filtering, source IP and destination IP filtering based on FPGA. The design is implemented by filtering the communication data included in the header information, letting safe data packets that match the rules through and abandoning those don't; the header information includes source IP address, destination IP address, data packet type (Transport Control Protocol (TCP) package, User Datagram Protocol (UDP) package and Internet Control Messages Protocol (ICMP), etc.), the TCP or UDP source port number and destination port number and the MAC address.

The packet filtering firewall which is implemented by software proposed in literature [1] can be flexible in the configuration of filtering strategies but is slow in speed. Literature [2] gives a design of firewall based on FPGA whose speed is lifted and that can be flexible in the configuration of filtering strategies, but the processor adopted is customized and the performance can't be upgraded in real time. By adopting SOPC technology based on FPGA it realizes packet filtering logic and filtering algorithm; with OR1200 processor based on Wishbone Bus it can realize the dynamic updating of matching policy and the coordination of processing sequence of each module[3]; The design in the present paper is faster in matching speed and more flexible in the updating of filtering policy; besides, it adopts OR1200 processor which is all open-source, which has high optimal velocity and much better performance than NIOS II [4][5].

Structures of this paper are as follows: the second part presents the whole design

scheme of the dynamic packet filtering firewall and explains the function and implementation of each component; the third part analyzes the firewall filtering modules and gives a detailed description of the structure of each module; the fourth part illustrates the experiment results and verifies the superiority of this design by comparing the processing time of each filed in this design with that in other configurations; the last part gives the conclusions and prospects and references are listed in the end of the paper.

2. The Dynamic Packet Filtering Firewall

Figure 1 shows the overall layout of the designed dynamic packet filtering firewall system. The design is composed of the following parts: 1) OR1200 processor; 2) Wishbone Bus interconnection matrix; 3) SDRAM controller; 4) UART controller; 5) GPIO modules; 6) Flash controller; 7) Ethernet controller; 8) packet filtering firewall modules.

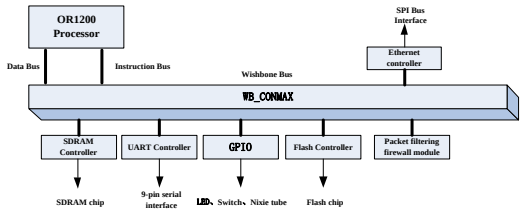

Fig.1 System block diagram

Among these parts, the OR1200 processor is the IP core based on Wishbone Bus from the opencores[3]. The packet filtering firewall modules are in charge of the filtering of IPv6 data packet; OR1200 processor controls each slave device through Wishbone Bus. The host sends Ethernet data packet via cables, and when Ethernet controller receives the data packet and generates the interrupting signal, OR1200 processor interrupts the processing and sends the IPv6 data packet to the packet filtering firewall modules. The packet filtering firewall modules will extract and then parse the fields from the IPv6 data packet and thus decide to either receive or discard the data packet.

3. Packet Filtering Firewall Modules

The filtering rule in this paper can be updated by the processor and Figure 2 shows the overall layout of the dynamic packet filtering firewall module which is composed of four major modules: the Wishbone Bus port module, the Content Addressable Memory (CAM) module, the netmask RAM module and the IPv6 data packet extracting module.

The CAM module is in charge of the matching operation of relevant data fields including TCP/UDP destination port number, TCP/UDP source port number, destination IP address, source IP address and source Media Access Control (MAC) address. The OR1200 processor operates the initial configuration and real-time updating of data on the CAM module.

The netmask RAM module is used to store subnet mask data of the source IP address and the destination IP address.

The IPv6 data packet extracting module parses the corresponding data packet and extracts the fields of corresponding implication according to the header information and then sends them to relevant CAM module for matching.

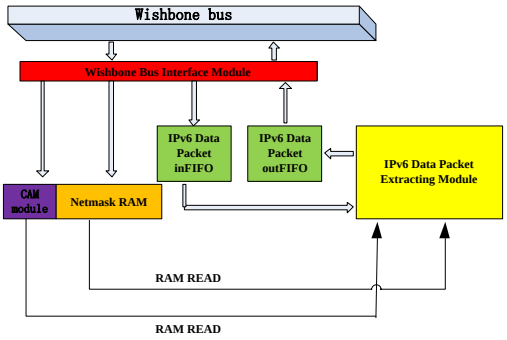

Fig.2 The dynamic packet filtering firewall block diagram

3.1 Wishbone Bus Interface Module

Figure 3 shows the signal distribution of this module and its functions include: a. receiving the initialization and dynamic updating data of

the CAM module; b. transmitting the initialization and dynamic updating data of the netmask RAM module; c. receiving the IPv6 data packet from the OR1200 processor; d. delivering the successfully matched IPv6 data packet in the outFIFO memory to the OR1200 processor. This module is in charge of the transmission of data by sending the data from the OR1200 processor to the relevant modules and thus implementing the operation of the configuration data.

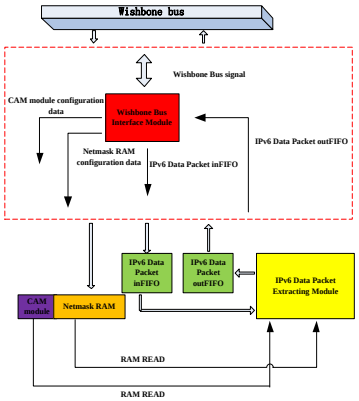

Fig.3 The signal distribution of Wishbone Bus Interface Module

3.2 Content Addressable Memory (CAM) Module

Content Addressable Memory (CAM) is fast in matching data. The CAM module in this design is implemented by dual port RAM and the specific signal definitions are as shown in Figure 4. The dual port RAM is 4 Kbits of memory block and its read data port is 16bit*256 and write data port is 1bit*4096. Since the system synthesis is implemented by calling the relevant IP core on an Altera FPGA device, each port of the dual port RAM is configured separately to supply two operations of CAM: write and read [6].

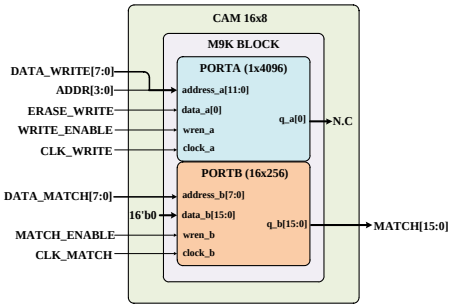

Fig.4 CAM signal

The write operation is illustrated in Figure 5. Among the 12-bit address line in the dual port RAM, the high 8-bit is used to represent the valid data to write and the low 4-bit represents the address of the data to be written. After the 12-bit data is written, put write/erase signal to 1 and hold it for one clock cycle, then the data is already written in the CAM module.

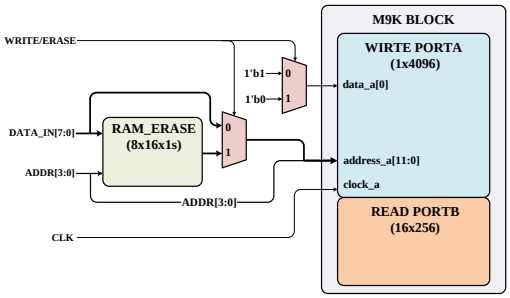

Fig.5 the operation of CAM write

The write operation is shown in Figure 6. The 8-bit matched data is sent to the port B address wire of the dual port RAM, and then the match_enable is put to high level for one clock cycle. If there is data for matching in the dual port RAM, it will output 16-bit output signal and any high signal among the 16 bits indicates successful matching.

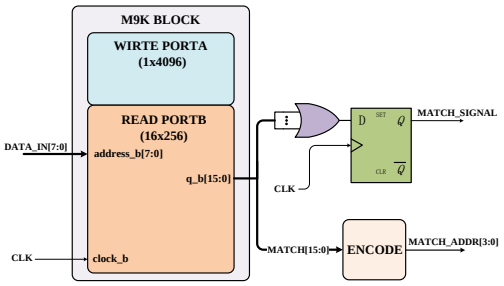

Fig.6 operation of CAM read

Since the source IP address and destination IP address of the IPv6 data packet are both 128 bits and a CAM can match data of only 8 bits, 16 CAM are needed to be cascaded. The matching operation of source/destination IP address is shown in Figure 7. The 128-bit source/destination IP address is divided into 16 segments and sent to the source/destination IP address CAM

module. The 16bit output of source/destination IP address CAM module is the 16 segments anding.

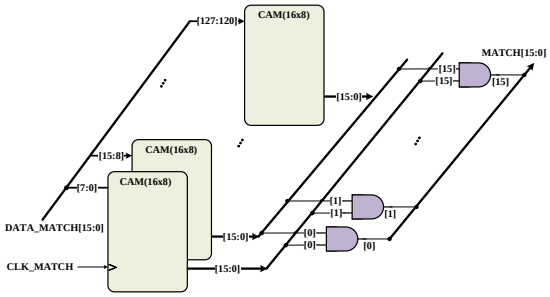

Fig.7 source/destination IP address matching

In addition, the 16-bit source/destination port number and 48-bit source physical (MAC) addresses also need matching and the mode is the same as that of the source/destination IP address CAM module. This thesis focuses on the implementation of IPv6 data packet filtering examination and CAM resources needed are as listed in Table 1.

Table 1. CAM module resources

Mode	Capacity (words)	Size (bit)	Component (CAM)	Memory block(Kb)
Dest port	16	16	2	2x4Kb
Sour port	16	16	2	2x4Kb
Dest IP addr	16	128	16	16x4Kb
Sour IP addr	16	128	16	16x4Kb
MAC addr	16	48	6	6x4Kb

3.3 Netmask RAM Module

The netmask RAM module stores the subnet mask of IP addresses. It matches the IPv6 data packet of certain IP address range and the RAM is implemented by calling the relevant IP core.

Figure 8 illustrates the operation process of matching IP address range by the collaboration of netmask RAM and IP address CAM. The IPv6 data packet extracting module parses the IPv6 address anding with the data from netmask RAM module, and the result is sent to CAM for data matching. Thus filtering the data of a certain range is implemented and comparing to accurate matching of each IP address, it's faster and less consuming in CAM resources. Furthermore, since CAM needs a high occupancy of memory resources, this mode also saves memory resources for users. CAM resources needed in this design are as listed in Table 2.

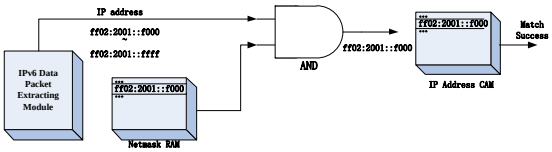

Fig.8 Netmask RAM module operational process

Table 2. Netmask RAM module resources

Mode	Capacity (words)	Size (bit)	Component (RAM)	Memory bkock(Kb)
Dest IP addr	2	128	32x128	1x4Kb
Sour IP addr	2	128	32x128	1x4Kb

3.4 IPv6 Data Packet Extracting Module

Figure 9 illustrates the block diagram of the IPv6 data packet extracting module. This module is composed of a finite-state machine (FSM) and the implementation modes refer to literature 14. There are altogether 10 states in the FSM: wait state, MAC data matching state, extracting data packet information state, extracting IPv6 data packet head, netmask RAM comparison state, source/destination IP address comparison state, extracting TCP/UDP head, successful matching state, unsuccessful matching state and written outFIFO state.

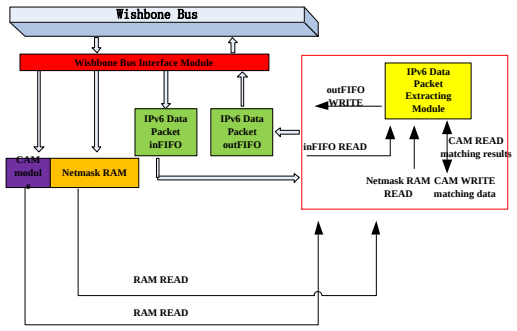

Fig.9 The signal distribution of IPv6 Data Packet Extracting Module

The setup of the FSM is illustrated in Figure 10. This finite-state machine is the most critical and most difficult part of the whole hardware design. The figure shows the state switching information and conditions needed for the switching of two adjacent conditions as well as functions of each condition. As Figure 10 shows, the FSM has 10 states totally and each state completes a special function.

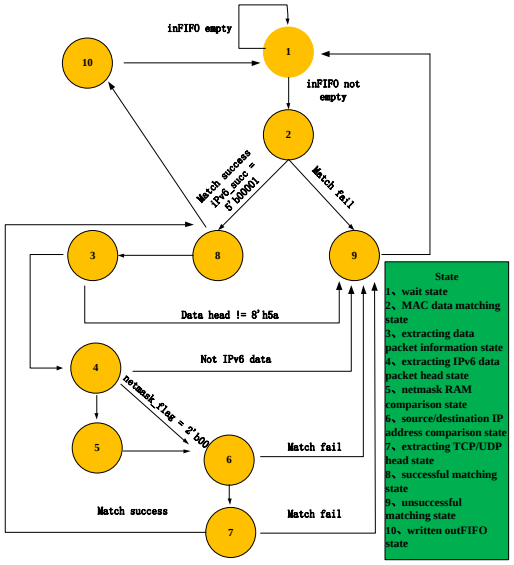

Fig.10 The FSM diagram

This part gave a detailed description of each module of the firewall hardware design and functions of the whole design and each separate module. The whole design completes the tests on an Altera FPGA device and is compared with the IPv4 hardware firewall based on FPGA. Though the design in this thesis consumes more FPGA resources, it is aimed at IPv6 network data packet and the bit wide of the source/destination IP address that needs filtering is 4 times as that of IPv4 data packet. Besides, bus and peripherals are added to the design for the convenience of observing the experimental results. As a result, the implementation proposed in this paper bears certain advantages in lowering the consumption of hardware logic blocks. FPGA Resources needed in the two implementations are listed for comparison in Table 3.

Table 3. FPGA resources

Protocol type	IPv4[4]	IPv6
Device type	Stratix II/EP2S60F672C5	Stratix II/EP2S60F672C5
Combinational ALUTs	4595/48352 (10%)	7471/48352 (15%)
Dedicated logic registers	3169/48352 (7%)	6032/48352 (12%)
Logic untilization	11%	25%
Total block memory bits	164096/2544192 (6%)	275708/2544192 (11%)

4. Experimental Results

To test the processing speed and efficiency of the design, a corresponding testing environment is set up. The IPv6 data packet are sent via cable from the contract award program on the host PCs to the FPGA device. The data packet processed by the FPGA device is then printed through the serial ports and the indicator light which can indicate the successful matching of the data packet will light for 10 seconds. The Nixie tube will show the number of the data packets that have been filtered or discarded, which enables the users to conveniently view the number of valid data packets which are sent. The files that handle the initialization and configuration of the OR1200 processor are compiled by cross-compiler on Linux and the executable files generated are downloaded to Flash on the

FPGA development board by Flash downloading tools, which avoids the loss of files in the case of blackout. The whole processing flow of the IPv6 data packet is illustrated in Figure 11.

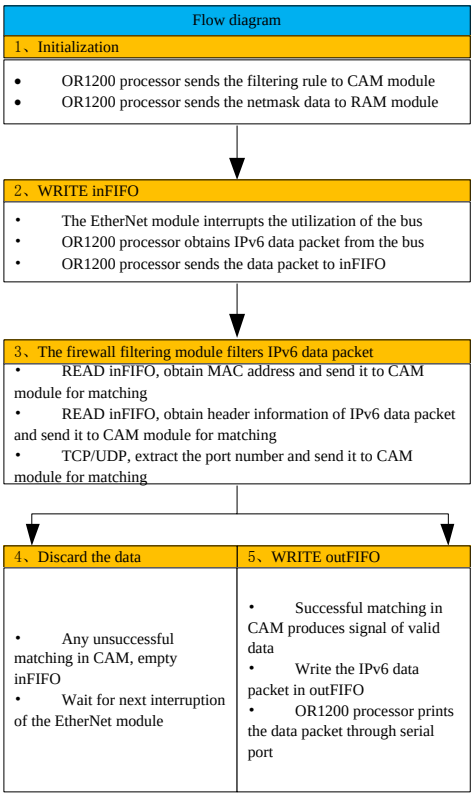

Fig.11 IPv6 data packet processing flow diagram

Figure 12 is the diagram of the testing environment which includes a host PC, a cable, a serial port line and the Altera development board.

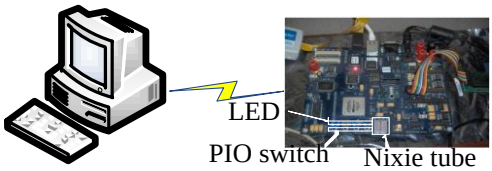

Fig.12 Testing diagram

When matching the IPv6 data packet, the next header value is included in the fore packet and value 6 or 17 is corresponding to TCP or UDP packet. In this case the source MAC address, source IP address and destination IP address will be matched, so as source port number and destination port number. In other cases, only the source MAC address, source IP address and destination IP address need matching. Table 4 lists the processing time to handle different types of data packets. It is worth mentioning that the working frequency of the FPGA device is 50 MHz.

Table 4. Processing time to handle different types of data packets

Type	Clock cycle	Process time(ns)
TCP/UDP	388	7760
ARP/ICMP/…	379	7580

During the parsing of the IPv6 data packet, the source port number, destination port number, source IP address, destination IP address, source IP address range, destination IP address range and source MAC address are accordingly extracted and matched to the CAM module. Table 5 lists the simulated processing time of each field.

Table 5. Processing time for matching of each field

Firewall mode	Process time(ns)[4]	Process time(ns)
Source port	1840	140
Destination port	1840	140
Source IP addr	1920	120
Destination IP addr	1920	220
Source IP addr range	none	440
Destination IP addr range	none	540
MAC addr	1920	300

Table 6. Overall processing time

Match mode	Total time(ns)[1]	Total time(ns)
accurate	10000	920
range	none	1560

Time listed in Table 5 shows the sum of time it takes from the moment the FSM obtains the fields and sends them to the relevant CAM module for matching to when the matching results are found out and

comparison is made with that in literature [7], which indicates that the packet filtering firewall proposed in this paper can largely improve the processing speed. Table 6 gives the sum of matching time in two modes and it is much faster than that in software operation mode[3]. What's more, the OR1200 processor can timely update the matching rule and thus realize dynamic updating.

5. Conclusion

The dynamic packet filtering firewall on SOPC based on an Altera FPGA device proposed in this paper can be implemented in two modes for users to choose from: accurate matching and rough matching. Both modes have improved a lot in the processing speed. Since the CAM module is set up by dual port RAM, the matching results can be worked out only one clock cycle after the write enable input of the matching data and thus it greatly improves the processing speed. As for the consumption of blocks, the complete design uses a small portion of the Altera StratixII/ EP2S60F672C5 FPGA, 25% of the logic blocks and 11% of the memory blocks, which realizes the balance of the processing speed and the consumption of the blocks. This design focuses on the IPv6 data packet to adapt to the reality that the IP addresses are running out. As a result, the performance of the dynamic packet filtering firewall design in processing speed, area optimization and security can meet the need of future network security.

References:

[1] Yingxu Lai, Guangzhi Jiang, Jian Li, Zhen Yang. Design and implementation of distributed firewall system for IPv6[C]. The International Conference on Communication Software and Networks,2009.

[2] Gouri Shankar Prajapati, Nilay Khare, A framework of a internet firewall for IPv6 using FPGA[J]. The International Journal of Computer Applications(0975 - 8887) Volume 50-No.21, July 2012.

[3] http://www.opencores.org

[4] Raouf Ajami, Anh Dinh, Embedded Network Firewall on FPGA [C], The Eighth Internation Conference on Information Technology: New Generations IEEE 2011.

[5] Arief Wicaksana, Arif Sasongko, Fast and reconfigurable packet classification engine in FPGA-based firewall[C], The International Conference on Electrical Engineering and Informatics, Bandung, Indonesia,17-19 July 2011.

[6] J. Brelet. Using Block RAM for High Performance Read/Write CAMs. XilinxCorporation, xapp204 v1.2 edition, May 2000.

[7] Raouf Ajami, Anh Dinh, Design a hardware network firewall on FPGA[J], IEEE CCECE 2011-000674