DESIGN A HARDWARE NETWORK FIREWALL ON FPGA

Raouf Ajami, Anh Dinh

Department of Electrical and Computer Engineering,
University of Saskatchewan, Canada

ABSTRACT

Catastrophe events can happen when a computer or a computer network is exposed to the Internet without any security protection. The security issues can be mitigated by setting up a firewall between the inside network and the outside world. This paper describes a design of a highly customizable hardware packet filtering firewall to be embedded on a network gateway. A packet filtering firewall controls the header field in each network data packet based on its configuration and permits or denies the data passing through the network. An Altera FPGA platform is used for implementing and evaluating the hardware network firewall. The hardware design provides much faster speed compared to traditional software applications.

Index Terms—Network firewall, FPGA

1. INTRODUCTION

Network security is one of the main concern for the network administrating personnel. The network must ensure safety and reliability from malicious and destructive attacks. The network should know how the attacks are being carried out and how to stop them. One of the tools to be used for this purpose is the network firewall [1]. It is more efficient to have a network firewall at the gateway of the site rather than protecting each single computer in the network. In other word, a network firewall is more powerful and manageable than its smaller counterpart. It is beneficial to know source or destination IP address so that the organization can permit or deny access for a known or a range of IP addresses. A packet filtering firewall controls the header field in each network data packet based on its configuration and permits or denies the data passing thorough the network. This designed firewall is a high speed packet processing firewall can be configured to different modes of operation and has the ability to accept and apply configuration changes in real time. The modes, which this firewall operates on, are based on header information in Link, Internet and Transport layers of the Transport Control Protocol/ Internet Protocol (TCP/IP) protocol stack. The following six options are the different modes implemented in this firewall:

- TCP/UDP destination port number
- TCP/UDP source port number
- Source IP address
- Destination IP address
- Source Media Access Control (MAC) address
- Combination of source IP address and destination port number

The main block was built using Verilog Hardware Description Language (HDL) to speed up the packet processing. A processor based embedded system with real-time operating system was designed to achieve highly customized and on-the-fly configuration change in the firewall. The whole design was implemented and evaluated on an Altera FPGA device. The design is more advanced than other hardware firewalls in literature [2-4].

2. A HARDWARE NETWORK FIREWALL

Figure 1 shows the overall layout of the designed hardware network firewall. The main modules in the embedded network firewall are the Nios II 32-bit microprocessor module, the Ethernet module, the Content Addressable Memory (CAM) module, the Netmask RAM module, the Arbiter module, and the Network Firewall module (NFM). All of these modules tightly works together to achieve a powerful, flexible and easy to configure packet filtering firewall.

The *NIOS II* is a soft core 32-bit RISC microprocessor from Nios CPU family, introduced by Altera as a general processor to be used in Altera FPGA devices to make a complete system on chip design [5]. Like any other micro-controller, the standard Nios II reference design comes with several peripherals. In the Nios II architecture, both memory and peripheral Input/Output are mapped into one integrated address space. Some of the peripherals play highlighted roles in this design.

The *Ethernet module* is the LAN91C111 from Standard Microsystems Corporation. The LAN91C111 control and data pins are hardwired to the FPGA. Tools are available for the Nios II embedded system to communicate with the network through this IC. The chip includes MAC and PHY engines and is able to process Carrier Sense Multiple Access With Collision Detection (CSMA/CD) protocol in 10/100 Mbps full duplex. Data transfer can only happen among the modules in the system when there is a shared data and address bus.

CCECE 2011 Niagara Falls, Canada
978-1-4244-9789-8/11/$26 ©2011 IEEE

The *Arbiter* is a control module to allocate shared resources (i.e., the Ethernet module) between the Nios II and the NFM modules and lets only one-to-one communication at a time. Figure 2 shows a block diagram of the arbiter and its related modules. The Ethernet module and the Nios II data bus are bi-directional. The arbiter uses tri-state buffer, which is controlled by the signal from the Nios II to allocate the data bus.

The *Network Mask* uses two RAM modules, each with sixteen 32-bit words to store network mask for source and destination IP addresses. In the source and destination IP address condition, the firewall can not only permit or deny a single IP address, but it can also decide what range of IP addresses can access the network beyond the firewall. This can be possible by either entering every single IP address in the source or destination IP address in the CAM module, or by specifying an IP range (network ID and network mask). Every single location of the netmask RAM module is pulled and tested against the data stored in the CAM IP module and as soon as a match is found, the packet is validated and passed through the firewall. If no match is found, the IP address has not been configured in the permitted IP address range and hence the packet must be removed. If there are more data fields in the Netmask RAM, this procedure takes more clock cycles to finish.

Content Associative Memory (CAM) is a type of memory system for fast searching application in hardware [6]. CAM acts differently from the other types of memory such as RAM and ROM; CAM is fed with data instead of address. CAM searches the entire memory for data and if the data is available in memory (data already stored previously), the location of data is returned. Since the whole operation takes only one clock cycle, CAM requires memory cell and associative comparison hardware circuitry for conducting parallel search of each memory cell. This makes CAM very fast but also costly due to the consumption of a large number of hardware logic blocks. Because of the high speed data search, CAM offers fast address lookup and low packet processing latency. The CAM is used to store source and destination TCP/UDP port number, source and destination IP address and MAC address field. Dual port memory along and extra logic are used to implement the CAM modules for this network firewall design.

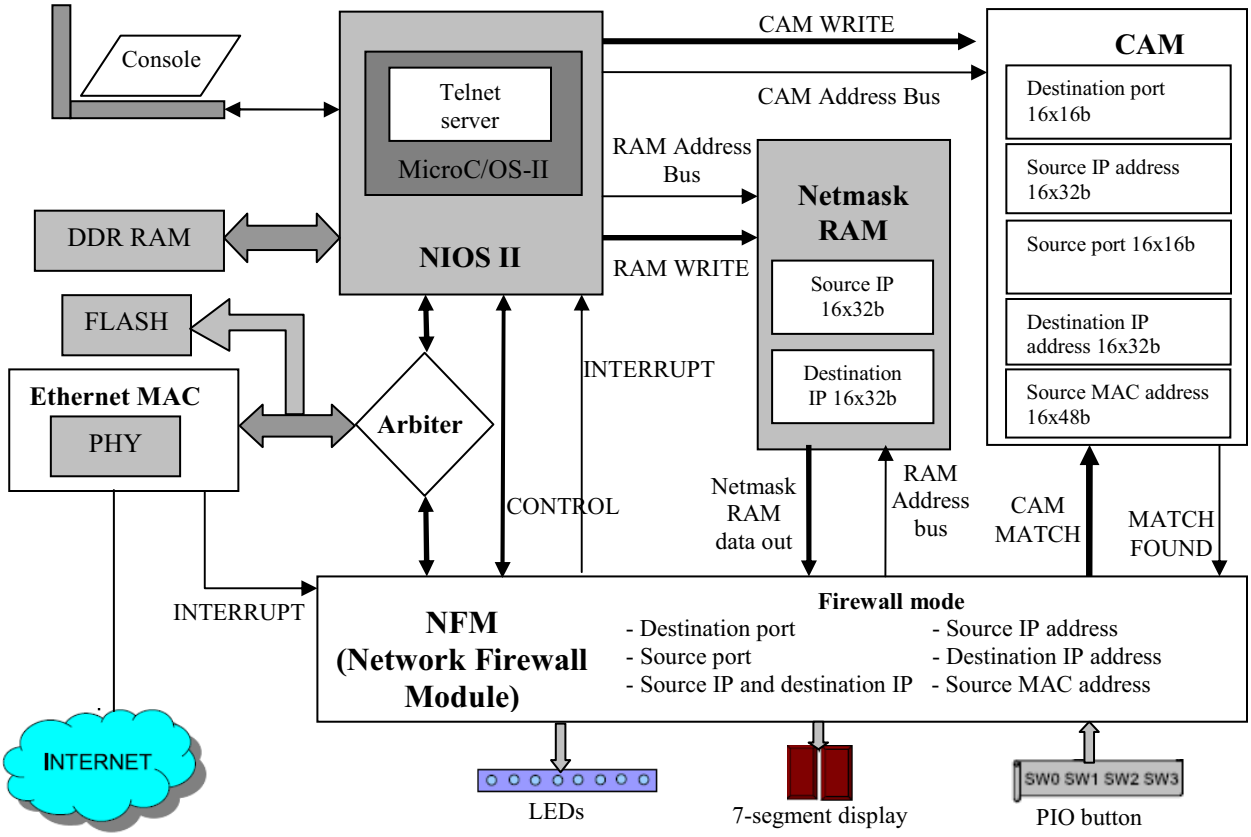

Figure 1. The hardware network firewall block diagram

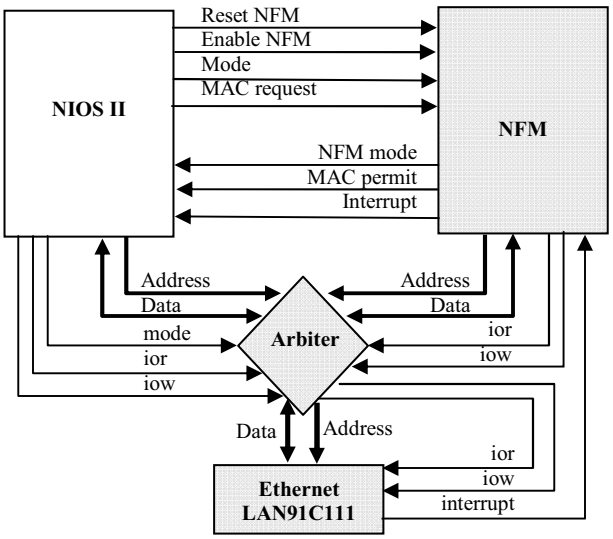

Figure 2. The arbiter and connected modules

The CAM template module is a 16 word by 8 bit (CAM 16x8) in total of 128 bits and is made by a 4 Kbits of dual port memory block on FPGA. Figure 3 shows the CAM 16x8 block diagram and its internal dual port RAM block wiring. There are two operations of CAM: write and read. The write operation happens by providing the data and address to the DATA WRITE (as MSB) and ADDR (as LSB) input of the CAM. The read operation is done by providing the value to DATA MATCH input of the CAM and enabling the CAM read enable input. If the data is already written in the CAM, the location appears in the MATCH output. Table 1 lists the characteristics of the custom built CAM for different modes of the firewall operation.

Table 1: CAM modules in the network firewall

Mode	Capacity	Size	Component	Memory block
Destination port	16 words	16-bit	2 CAM 16x8	2x4Kb
Source port	16 words	16-bit	2 CAM 16x8	2x4Kb
Destination IP address	16 words	32-bit	4 CAM 16x8	4x4Kb
Source IP addr.	16 words	32-bit	24 CAM 16x8	4x4Kb
Source MAC address	16 words	48-bit	6 CAM 16x8	6x4Kb

The main functionality of the hardware network firewall is the *Network Firewall Module (NFM)*. This Verilog module contains a 5-state state machine. The machine changes the state based on the input signals and current conditions. After system initialization and enabling the firewall by the Nios II, the NFM waits for an interrupt from the Ethernet module. Figure 4 illustrates how the NFM process works in a self-explained flow chart.

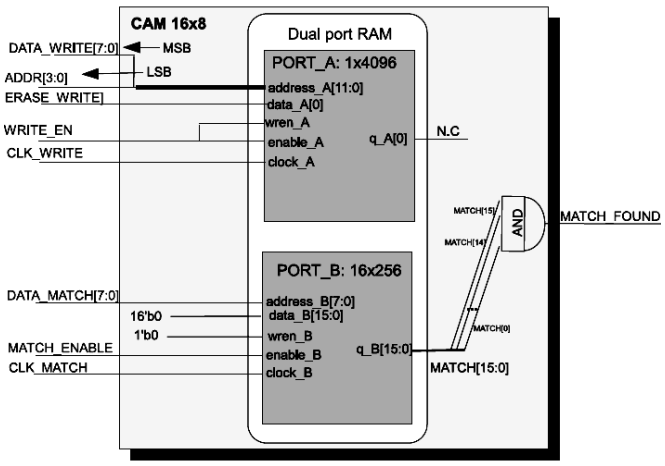

Figure 3. CAM 16x8 block diagram (built by dual RAM port).

Initially, when the system is powered up, the firewall module is in inactive mode as the Nios II has not been programmed to run any code or operating system (OS). Nios II Integrated Development Environment (IDE) development tool is used to upload essential applications including RTOS to run on the Nios II. The software initializes all the modules according to the selected mode of operation. When the Ethernet module finds any TCP/IP traffic targeted toward the Nios II, it interrupts the NFM. Based on the initial operation mode, the NFM extracts the necessary field off the Ethernet frame and inquires the permission from the CAM module. If the CAM finds the selected field in its memory block (i.e., match found), it gives the pass permission to the NFM and the NFM interrupts the Nios II for the receiving packet. In the case of an invalid packet, the NFM drops the packet and waits for the next interrupt from the Ethernet. After each Ethernet interrupt, the packet status is reported to the Nios II by the NFM for monitoring purposes. The firewall operation mode and configuration can be changed any time using the Telnet server running on the Nios II. Table 2 shows the resource required to implement the design into an Altera FPGA device. Smaller device with higher speed grade can be targeted to reduce cost of the system.

Table 2: FPGA resource

Family/Device	Stratix II/EP2S60F672C5
Combinational ALUTs	4595/48352 (10%)
Dedicated logic registers	3169/48352 (7%)
Logic untilization	11%
Total block memory bits	164096/2544192 (6%)

4. EXPERIMENT RESULTS

Altera Embedded Design Suites (EDS) was used to write the software modules to handle initialization and configuration of the Nios II and the firewall to coordinate and run Telnet server software. The modules also handle the configuration

change and monitoring operation of the firewall. Figure 5 shows a small size network setup including two host PCs, the Altera development board and a network hub. It should be mentioned that in order to capture all of the traffic in a network, the network devices should not be connected to each other through a network switch since a switch sends traffic only to a switch port to which the traffic is intended for and all other traffic for other devices are filtered. But a network hub broadcasts all of the traffic to all of the ports in the device and therefore any host in the network can see the traffic. A switch intelligently investigates the Ethernet destination address in all of the incoming traffic and builds up a MAC address table based on connected devices to each switch port.

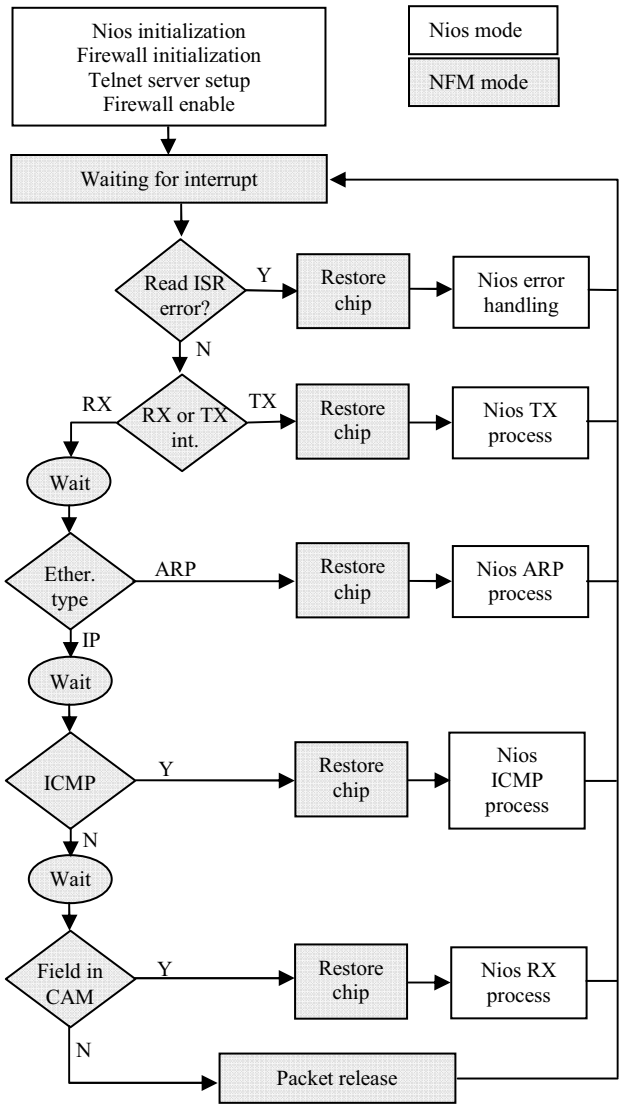

Figure 4. NFM process flowchart.

There are 4 software tools used for this hardware network firewall testing: Wireshark, Tcpdup, Ping, and Network MAPper (NMAP) [7,8]. The first test after the setup is to *ping* the firewall to make sure that it is alive in the network and can respond to an Internet Control Message Protocol (ICMP) request. Real time status monitoring on the Nios IDE console shows the ICMP request and reply from the firewall. An LED indicator on the circuit board also illuminates when the ICMP packet arrives. It takes 88 clock cycles to process each ICMP packet (i.e., 1.76µs at the 50 MHz FPGA clock).

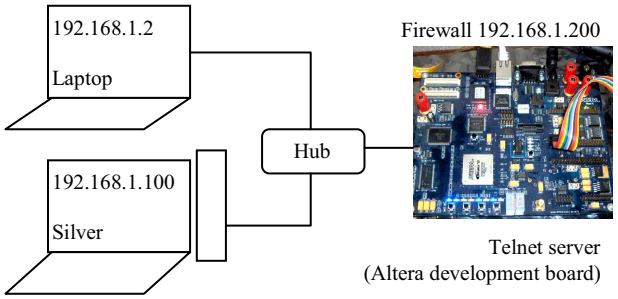

Figure 5. Test network set-up.

Table 3 lists the testing results of the timing for packet transmission interrupt, packet reception interrupt, ARP packet and ICMP packet processing time. Table 4 shows the timing data result for different modes in the hardware network firewall. The processing time takes between 1.84µs and 4.2µs. This processing time is not causing any substantial delay for packet processing. For simplicity, only the source IP address and destination port number mode testing is described in detail. In this mode, at first each packet is investigated for source IP address matching. If the source IP address of the packet is found in the CAM, the process goes to the next step which is the destination port number matching. If the destination port number of the packet is also found in the destination port number CAM then the packet is permitted to pass through the firewall. To test this, the laptop IP address has been configured in the firewall along with destination port 30. The result shows it takes 153 clock cycles (3,060ns at the FPGA 50MHz clock) for the valid source IP address along with the valid destination port number. However, for an invalid destination port along with a valid IP address packet, it takes 169 clock cycles (3,380ns). There are 127 and 210 clock cycles for a valid packet and an invalid packet processing respectively when the firewall performs without monitoring functionality. And again, the same as in the source IP address worst case situation, 236 clock cycles are exhausted for dropping an unauthorized packet by the firewall since it has to search all of the Netmask RAM cell memories against the source IP address CAM for a MATCH.

Table 3: Interrupt processing time

Interrupt time	Clock cycle	Processing time (ns)
Transmission	9	180
Receive	Varies	Varies
ARP	62	1240
ICMP	88	1760

It takes 123µs to process an Ethernet frame by the LAN91C111 or any other 100Mbps Ethernet module. Each Ethernet module is 1,538 Bytes (without considering VLAN tagged frame) or 12,304 bits at 100Mbps is equivalent to 123.04µs processing time. The hardware firewall package processing of 1.84-4.2µs is just a small fraction of the processing time of the Ethernet module. As a result, the network firewall adds only 2-4% to the packet processing time. This time indicates that the design can process up to 2.9Giga bits of Ethernet data per second. By adding the hardware firewall on a network gateway, data transfer experiences no delay. A comparison with the Linux software firewall *iptables* also shows this design performs much faster and not sensitive to the packet size or the number of rules in the firewall configuration.

Table 4: Number of clock cycles for invalid packet dropping

Firewall mode	With monitoring #clock/ns	Without monitoring #clock/ns
Destination port	118/2360	92/1840
Source port	118/2360	92/1840
Source IP address	122/2440	96/1920
Destination IP address	122/2440	96/1920
Source IP and Destination port	153/3060 to 169/3380	127/2540 to 143/2860
MAC address	122/2440	96/1920

5. CONCLUSION

In addition to high speed, this hardware network firewall also provides many practical features such as real-time configuration, real-time status report, transport and physical layer packet/frame processing, and IP address blocking range capability. The complete design uses a small portion of the Altera StratixII-2S60 FPGA, 11% of the logic blocks and 6% of the memory blocks. This makes an ideal size for the firewall system on a network gateway. The processing speed can be improved further if the design is to be combined with a faster system clock, faster Ethernet module or eliminating waiting cycles (due to Altera board design and connection between the FPGA and Ethernet module). This firewall can be used for various purposes in a network with several applications. The source or destination port filtering is the useful feature for controlling the traffic to a dedicated server.

5. REFERENCES

[1] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls*, O'Reilly Media, 2nd Edition, 2000.

[2] Jedhe, G.S.; Ramamoorthy, A.; Varghee, K., "*A Scalable High Throughput Firewall in FPGA*," The 16th International Symposium on Field-Programmable Custom Computing Machines, FCCM'08, Palo Alto, CA, USA, April 14-15, 2008, pp. 43-52.

[3] Kayssi, A.; Harik, L., Ferzli, R., Fawaz, M., "*FPGA-based Internet protocol firewall chip*," The 7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000, Jounieh, Lebanon, December 17-20, 2000, pp. 316-319, vol.1.

[4] D. Laturnas, R. Bolton, "*Dynamic silicon firewall,*" Canadian Conference on Electrical and Computer Engineering, CCECE 2005, Saskatoon, Canada, May 1-4, 2005, pp. 304-307.

[5] *NIOS II Processor Reference Handbook*, Altera Coproration.

[6] J. L.Brelet, "Using Block RAM for High Performance Read/Write CAMs," http://www.xilinx.com/support/documentation/application_notes/xapp204.pdf, 2000. Accessed Nov 2010.

[7] J. Postel, "RFC:792, Internet Control Message Protocol," 1981.

[8] E. Nemeth, G. Snyder, and T. R. Hein, *Linux Administration Handbook*, Prentice Hall, 2nd Ediion, 2006.