International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-2 Issue-5, April 2013

# Design of A Embedded Ethernet Packet Sniffer

**Rahul Pal, Rahul Gotiya, Pankaj Singh, Amit Agrawal**

***Abstract- In this paper we are proposing a brief description about embedded Ethernet based event controller Packet sniffers. These are devices or programs capable of intercepting and logging network traffic for which they were not the intended recipient. Their ability to eavesdrop on network traffic has made them indispensable tools for IT administrators. In modern IP networks, packet sniffers are often used to determine the source of network problems, detect intrusions and locate vulnerabilities. Sniffers can also be used for covert surveillance of users internet activities. Ethernet operates at higher bit rate than slow-speed embedded protocols.***

***Keywords— embedded, packet sniffer, collision domain.***

## I. INTRODUCTION

Computer communication systems and especially the Internet are playing an important role in the daily life. Using this knowledge many applications are imaginable. Home automation, utility meters, appliances, security systems, card readers, and building controls, which can be easily, controlled using either special front-end software or a standard internet browser client from anywhere around the world. Web access functionality is embedded in a device to enable low cost widely accessible and enhanced user interface functions for the device. With the growing complexity of IP networks, it has become increasingly difficult to determine the source of network problems. Packet loss can occur due to any number of sources, from congestion to poorly written firewall rules or routing problems. It can at times be difficult to determine where in the network packets are lost, and system administrators will often find it helpful to view the traffic going over the line. This is the job of a packet sniffer[1]. The following sections will explain the operation of a packet sniffer, the inadequacies of current packet-sniffing technologies, and the motivation for the E-Sniff project.

### *1.1 THE ETHERNET NETWORK TOPOLOGY*

Most modern computer networks follow the Ethernet specification, which defines the physical interface used to transmit data between hosts on the network. In Modern Ethernet networks, computers are linked by Category 5 UTP cabling to a central hub, which serves as a connection point for the different hosts on the network. [2] As can be seen from figure 1, this topology resembles a star, with each point on the star representing a single networked computer.

**Manuscript published on 30 April 2013.**
***Correspondence Author(s)**
**Rahul Pal,** Department of Electronics & Communication BE (VII SEM) Takshshila Institute of Engineering & Technology, Jabalpur, (M.P)
**Rahul Gotiya,** Department of Electronics & Communication BE (VII SEM) Takshshila Institute of Engineering & Technology, Jabalpur, (M.P)
**Pankaj Singh,** Department of Electronics & Communication BE (VII SEM) Takshshila Institute of Engineering & Technology, Jabalpur, (M.P)
**Amit Agrawal,**HOD Electronics And Communication department Takshshila Institute of Engineering & Technology, Jabalpur (M.P)



Hence Ethernet networks are said to use a physical star topology, where physical refers to the actual spatial arrangement of the
networked computers.[3]

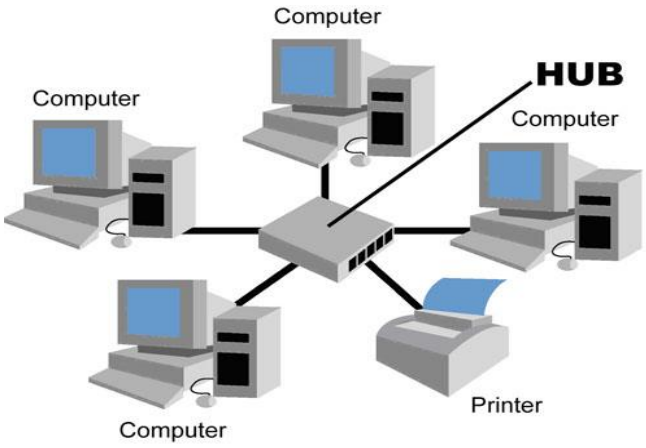


**Fig. 1. Physical Star topology**

The physical topology of an Ethernet network is misleading, because it gives the impression that each host has a dedicated line to the hub that is not shared with the rest of the network. In reality, the line attaches the host to a bus segment hidden inside the hub, to which every host has equal access. This arrangement is depicted in figure 2. Thus Ethernet networks are said to have a logical bus topology, where logical refers to the behavior of the electronics that connect the computers.

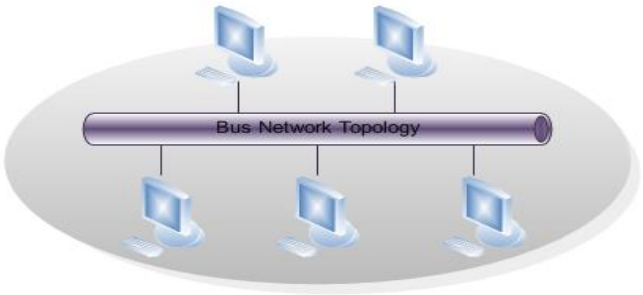


**Fig.2. Bus Network Topology**

To differentiate each computer from its neighbors, each computer is given a unique 48-bit MAC address, which is hardwired into the network interface hardware. Transmitted data is split into frames, tagged with the MAC address of the sender and intended recipient, and sent out over the bus. Because every host has equal access to the bus, every host receives all data transmitted onto the network, even if it was not the intended recipient. A typical network host ignores data intended for other computers, by comparing the recipient MAC address in the Ethernet frame to its own and discarding the frame if thetwo do not match. [8]

## II. WHAT IS A PACKET SNIFFER

A packet sniffer, in essence, is any device that does not discard frames in the manner described above. This is referred to as putting the interface into promiscuous mode. [1] By processing and logging all the packets going over the bus, a packet sniffer can eavesdrop on communications between other hosts on the network. This enables interested parties to view any and all network traffic, including emails, instant messages, remote logins, etc. Thus, a packet sniffer can be used to verify network performance, troubleshoot network errors, and perhaps most importantly, intercept "confidential" communications between other users of the network. The computers transmitting and receiving give no indication that the traffic has been intercepted, making a properly implemented packet sniffer undetectable to the average user. [1]. Packet Sniffers are devices or programs capable of intercepting and logging data traveling over a computer network. Sniffers are used for a variety of purposes, including monitoring, network troubleshooting, intrusion detection and surveillance. The typical packet sniffer is a software program that runs on a PC. This wastes CPU time that could be used for other tasks. Also, PCs are bulky and difficult to hide, making them less practical for surveillance purposes.

## III. DESIGN AND IMPLEMENTATION

The design and implementation of the E-Sniff system took over 6 months and hundreds of man-hours of work to complete. The completed system hardware and software consists of approximately 3000 lines of VHDL code and 4000 lines of C. Given the complexity and sheer size of the code, covering it in any great detail is beyond the scope of this paper; hence, only a high-level software flowchart and hardware schematic will be discussed here. Interested readers may consult http://cegt201.bradley.edu/projects/proj2007/dsniff/, which contains complete downloadable C and VHDL source code listings for the project. The next several sections will outline the design requirements for the system, the development platform used, the overall structure and function of the system hardware, and an explanation of the main routines in the system software.

E-Sniff is a small special-purpose embedded system for capturing and logging network data. It is implemented in an FPGA, and uses a soft processor and custom hardware to capture packets and display them on a VGA monitor. The user can also enter commands using a PS/2 keyboard to control packet processing. The E-sniff project is implemented in an Altera DE2 board. This board contains numerous peripherals.

The ones used for this project are: a Cyclone II FPGA, an 8MB SDRAM, 4MB CFI flash, an 8MB Serial Flash, a 16x2 Optrex LCD, a high-speed Video DAC, a PS/2 serial port and a DM9000A 10/100 Mbps Ethernet Controller. These are highlighted below in red as shown in fig.3.

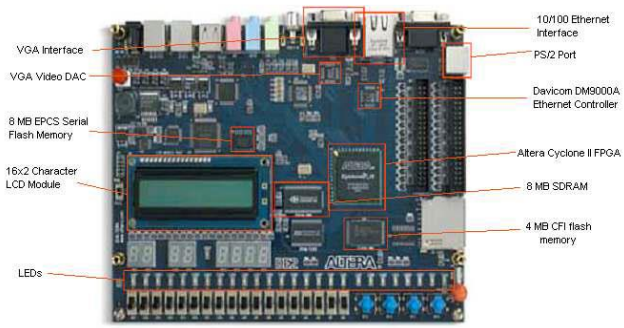


**Fig.3. Altera DE2 development board. Peripherals used for the project are highlighted in red.**

## IV. WORKING

Ethernet networks use a physical bus topology. All devices on the same bus segment are said to be in the same "collision domain."All hosts in a collision domain can see all the data on the bus segment, i.e. they can intercept traffic meant for other hosts. This means our packet sniffer can intercept traffic going to/from any host in the same collision domain. In modern networks, switches are used to break up collision domains. Each port on a switch corresponds to a single collision domain. This means our sniffer cannot see any traffic if it is plugged into the switch!!! To sidestep this problem, E-Sniff is plugged into a hub, which is inlined between the host and the switch. This puts the sniffer on the host's collision domain. Now we can read our coworker's email!.

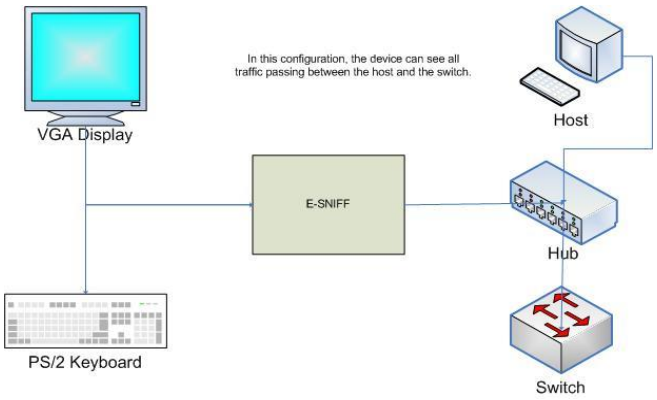


**Fig.4. Working of a Packet sniffer**

## V. HARDWARE DESIGN

The CPU is an Altera Nios II customizable soft processor. It is clocked at 100 MHz and is implemented in the Cyclone II FPGA. The CPU was generated with Altera's SOPC builder tool. The Nios II can be completely customized, and can interface directly with any memory or peripheral device on the board. The E-sniff CPU interfaces with several chips on the DE2 and has access to over 20 MB of memory, as shown below:

1.8MB EPCS serial flash - holds software image and FPGA config data

2.8MB SDRAM - main program memory, holds heap, stack, text, etc.


*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

3.4MB CFI Flash - non-volatile packet storage and software config data
4.4KB On-chip memory - fast memory on the FPGA, used to hold the exception stack/interrupt service routines.

**The CPU also interfaces with the following onboard peripherals:**
1.Davicom DM9000A Fast Ethernet MAC/PHY
2.Keyboard/VGA on-chip text memory
3.Keyboard receive module
4.Optrex LCD
5.25 LEDs, 17 switches, 4 buttons

**It contains the following peripheral modules:**
1.Hardware system ID module
2.JTAG Debug/Trace module
3.1s interval timer - generates interrupts for per-second tasks
4.1ms interval timer - generates interrupts for CPU usage monitoring

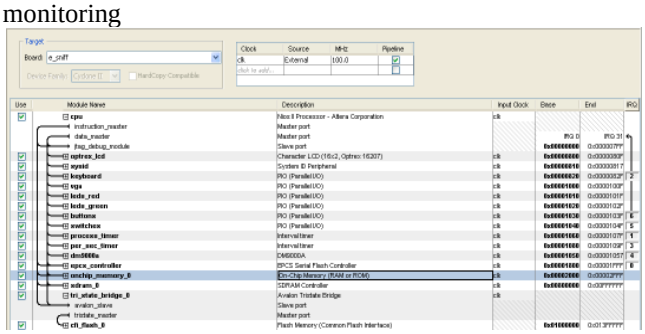

**Fig.5. Layout of the E-Sniff processor in SOPC Builder.**

As shown in the fig5. To the right of each peripheral are an input clock and a memory address. The E-Sniff processor uses only one clock; hence it is used as the input clock for every peripheral.
Memory addresses can be assigned manually by the user or automatically by SOPC Builder. To have SOPC builder lay out components in memory, select 'Auto-Assign Base Addresses' from the 'System' menu. Some peripherals can also be assigned interrupt requests. The Nios II has 32 nonvectored
interrupt requests available, which can be assigned in the rightmost column of the memory map. Designers should carefully consider which peripherals should have higher IRQs, as this will affect the precedence of one external interrupt over another.The highest priority IRQ is IRQ 0, and IRQ 31 has the lowest priority.
The remaining peripherals are interfaces to various memory devices on the DE2 board.The first one added to the project was an SDRAM controller, which interfaces to the DE2's 8 MB SDRAM chip. SDRAM timings such as CAS latency, setup times and hold times were adjusted to the datasheet values by double-clicking on the component in SOPC Builder. Directly below the SDRAM in memory is a 4 kB on-chip memory peripheral, which is a fast memory implemented inside the FPGA. This was added to provide a dedicated memory from which to execute exception handlers.

### *5.1 VGA CONTROLLER*

The VGA controller is a custom hardware module that enables the E-Sniff system to interface with any standard VGA monitor. The module displays text at a 640x480 resolution, and can display up to 53 lines of 76 characters apiece. The screen is divided into three sections - an output section, which can display 51 lines of text, a status line, which displays one line of status information, and a keyboard input line, which displays characters typed on the PS/2 keyboard. Program output is written at the bottom of the output section, then shifted upwards when a line return is signaled.


**Fig. 5 - Typical screenshot of the VGA controller output. The status line is the portionthat reads "Initializing system." Above this is the output section, and at the bottom is the keyboard input line, which contains the "e-sniff>" prompt.**

The VGA controller was designed to require minimal use of the processor to update the display. In furtherance of this goal, sync signal generation and line returns are handled in hardware, and the processor can write a character or line return in only three instruction cycles. An input port from the keyboard receiver enables the keyboard to write directly to the bottom line of the display without interrupting the processor, and a keyboard input13 read port allows the processor to read typed characters off of the display and clear the keyboard input line.

### *5.2 KEYBOARD MODULE*

The keyboard hardware receives scan codes from a PS/2 keyboard, translates them into ASCII and writes them to the screen. The first module in the keyboard controller receives 11-bit keyboard messages, checks them for parity and alignment errors, and outputs the 8-bit scan codes contained in them. The second module is a hardware look-up table, which takes the scan codes and translates them into ASCII. The ASCII characters are then written to an on-chip memory in the vga module, which displays them on the screen. This is all done without interrupting the processor. When the user has typed an entire line of text, they press the enter key. This interrupts the processor, which reads the line of text out of the input memory, then clears it, then processes the command.

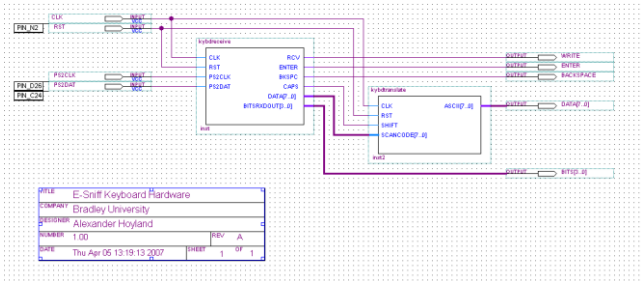


**Fig.6. Schematic diagram of the keyboard controller module.**


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

## VI. SOFTWARE DESIGN

The software for the E-Sniff system consists of approximately 4000 lines of C code distributed over 23 source files, and was developed using the Nios II EDS software from Altera. The software was implemented as a single-tasking bareboard application; no underlying system services or Real-Time Operating System were used. This was done to keep the project software as simple as possible, and to avoid the overhead of a "thick" operating system library. In retrospect, the system may have benefited from the use of an RTOS kernel, since there are many points where the system blocks while waiting for an external event to occur. Future work on this project might include the integration of the E-Sniff system software into a multitasking RTOS environment, so that the processor can be used more efficiently. The software can be divided into five main routines, which are discussed in the following sections. In figures 3-5, a red box indicates that the process alters its behavior based on user input from the keyboard.

The E-Sniff software was built using the Altera Nios II EDS, an Eclipse-style IDE for writing Nios II software.

The software consists of five main routines distributed over 21 source files. The source files contain over 3000 lines of C code.

1.The first routine initializes the system.

2.The second routine monitors CPU usage.

3.The third routine updates the status bar, system run timer and LEDs.

4.The fourth routine handles user input from the keyboard.

5.The fifth routine handles interrupts from the Ethernet controller.

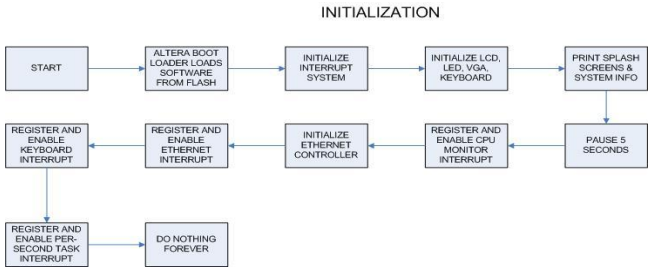


**Fig.7. E-Sniff system initialization routine.**

## VII.CONCLUSION

### *7.1 INTERPRETATION OF TEST RESULTS*

The results above indicate that a small embedded system such as E-Sniff could potentially perform as reliably as PC-based sniffing software; however, there is a lot of work yet to be done. The reader should bear in mind that E-Sniff is a proof-of-concept 26 prototype. If refined, clocked faster, and run under an RTOS, E-Sniff could potentially be as efficient as its PC predecessors, and would have none of the disadvantages presented by the PC platform. In the future work section, a number of refinements are suggested which, the author believes, would result in a marked performance improvement over the current alpha version.

### *7.2 FUTURE WORK*

While essentially functional, the E-Sniff system is far from complete. This section lists a number of changes that could be made in order to improve the performance of the E-Sniff system to more acceptable levels.

- The system software could be run in a multitasking RTOS environment to ensure more effective use of the processor. The current bareboard app has a number of sections where execution is blocked, wasting CPU time that could be used for other tasks. An RTOS would avoid this ineeficiency.
- The flash memory might be replaced with a faster form of non-volatile memory, such as battery-backed RAM. Alternative NVM schemes should be investigated to improve the system's performance while packets are being logged.
- A performance increase might be achieved by rewriting some of the Altera device drivers to make more efficient use of the processor.
- The current keyboard driver is somewhat buggy and does not allow transmission to the keyboard; it also does not support all the keys. This should be corrected in a future version by rewriting the keyboard hardware.
- The E-Sniff system can currently interpret about fifteen protocols out of the thousands of standard internet protocols. This number must increase drastically if the E-Sniff system is to be made commercially viable. The changes above are aimed at improving the existing features of the system. However, there are a number of new features that could potentially be added and should be investigated. Among these are:
- Add the ability to identify gateways and nameservers
- Add the ability to cache routing information and build a route table
- Add the ability to log and display current DHCP leases
- Add the ability to track website usage by each host on the network
- Allow audio eavesdropping on VoIP phone calls One of the requirements for the system was that it not transmit any data onto the network. However, E-Sniff may also be useful for IT administrators who do not need to worry about being detected. By relaxing the "no transmission" requirement, a plethora of newand useful features could be added:
- Add a web interface for issuing commands and reading the log
- Add the ability to flash in new firmware over the network
- Add DNS name resolution to packet dissectors 27
- Add support for basic network testing tools - ping, tracert, nslookup, port scanning, etc.
- Identify network servers and determine their weaknesses

## REFERENCES

1. Wikipedia,"Packet Sniffer," [Online Document], 2007 3 April, [Cited 2007 11 May], Available HTTP: http://en.wikipedia.org/wiki/Packet_sniffer
2. Wikipedia, "Ethernet," [Online Document], 2007 8 May, [Cited 2007 11 May], Available HTTP: http://en.wikipedia.org/wiki/Ethernet
3. Michael Myers, Network+ Certification All-In-One Exam Guide, 3rd Edition, McGraw-Hill Osborn Media, 2004.
4. Linh Trinh, "TCP/IP Sniffer Designs Teaches Basics of Embedded Ethernet," [Online Document], 2002 15 April, [Cited 2007 11 May], Available HTTP: http://www.elecdesign.com/Articles/Index.cfm?AD=1&ArticleID=2099
5. Wikipedia, "Request For Comments", [Online Document], 2006 June 9, [Cited 2007 11 May], Available HTTP: http://en.wikipedia.org/wiki/Request_For_Comments


*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

6. Altera Corporation, “Quartus II Handbook vol. 5: Altera Embedded Peripherals,” [Online Document], 2007 May, [Cited 2007 11 May], Available HTTP: http://www.altera.com/literature/hb/nios2/n2cpu_nii5v3.pdf
7. Altera Corporation, “Nios II Software Developer’s Handbook,” [Online Document], 2007 May, [Cited 2007 11 May], Available HTTP:http://www.altera.com/literature/hb/nios2/n2sw_nii5v2.pdf
8. Altera Corporation, “Nios II Flash Programmer User Guide,” [Online Document], 2007 May, [Cited 2007 11 May], Available HTTP:http://www.altera.com/literature/ug/ug_nios2_flash_programmer.pdf
9. Altera Corporation, “Nios II Processor Reference Handbook,” [Online Document], 2007 May, [Cited 2007 11 May], Available HTTP: http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf
10. Altera Corporation, “DE2 Development and Education Board User Manual,” [Online Document], 2007 May, [Cited 2007 11 May], Available HTTP: http://www.altera.com/education/univ/materials/boards/DE2_UserManual.pdf
11. Jon Postel, “RFC 791 – Internet Protocol,” [Online Document], 1981 September,[Cited 2007 11 May] Available HTTP: http://www.ietf.org/rfc/rfc0791.txt
12. Jon Postel, “RFC 793 - Transmission Control Protocol,” [Online Document],1981 September, [Cited 2007 11 May] Available HTTP:http://www.ietf.org/rfc/rfc0793.txt
13. Jon Postel, “RFC 768 – User Datagram Protocol,” [Online Document], 1981 September, [Cited 2007 11 May] Available HTTP: http://www.ietf.org/rfc/rfc0768.txt
14. David C. Plummer, "RFC 826 - Address Resolution Protocol," [OnlineDocument], 1982 November, [Cited 2007 11 May], Available HTTP:http://www.ietf.org/rfc/rfc0826.txt.
15. Jon Postel, "RFC 792 - Internet Control Message Protocol," [Online Document],1981 September, [Cited 2007 11 May], Available HTTP:http://www.ietf.org/rfc/rfc0792.txt
16. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners- Lee, “RFC 2616 – HyperText Transfer Protocol Version 1.1,” [Online Document], 1999 June, [Cited 2007 11 May], Available HTTP: http://www.ietf.org/rfc/rfc2616.txt
17. M. Crispin, “RFC 3501 – Internet Mail Access Protocol,” [Online Document],2003 March, [Cited 2007 May 11], Available HTTP:http://www.ietf.org/rfc/rfc3501.txt
18. M. Rose and J. Meyers, “RFC 1939 – Post Office Protocol Version 3” [Online Document], 1996 May, [Cited 11 May 2007], Available HTTP:http://www.ietf.org/rfc/rfc1939.txt
19. Jon Postel, “RFC 821 – Simple Mail Transfer Protocol,” [Online Document], 1982 August, [Cited 2007 11 May], Available HTTP: http://www.ietf.org/rfc/rfc0821.txt
20. Wikipedia, “OSCAR Protocol,” [Online Document], 2007 8 May, [Cited 2007 11 May] Available HTTP: http://en.wikipedia.org/wiki/OSCAR_protocol.



Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)