Boston University
Electrical & Computer Engineering

EC463 Capstone Senior Design Project

First Prototype Testing Plan

CyberTap

by

Team 2
CyberTap

Team Members:
Felipe Dale Figeman fdale@bu.edu
Alex Fatyga afatyga@bu.edu
Evan Lang evanlang@bu.edu
Noah Malhi malhin@bu.edu
Justin Morgan justinfm@bu.edu

Required Materials:

Hardware:

- 2x Raspberry Pi 3 (with Ethernet cable)
- TP-LINK TL-SG105E 5-Port Gigabit Easy Smart Network Switch
- Desktop PC
- Nexys A7 FPGA

Software:

- Xilinx C/C++ SDK 2019.1
- Node.js Web Client
    - main.html
        - Front end that receives data from server.js and puts it into the table
    - testUart.js
        - Back end that receives output through serial output, sends it to front end
- VHDL/Verilog
    - Microblaze server and other required Vivado modules such as AXI ethernet, clk wizard, and MII_to_RMII
    - Utilized Vivado IP's for the above

Setup:

The overall test is separated into two parts. The first part will involve testing the end to end functionionality of the current networking set up with the FPGA Microblaze echo server. After loading the Microblaze echo server onto the FPGA, the desktop will be connected to the FPGA via Ethernet and an app called Tera Term will be used to send packets through the connection to the Microblaze server's IP address, 192.168.1.10 (on port 7). The packet payload is then not only echoed back to the source IP (Tera Term) but is also sent to the serial port COM6 (UART). The serial port is then read by the node.js file testUart.js, which displays the data in the web app. The success of this test will be determined by whether or not all packets sent through Tera Term are correctly echoed by the FPGA (the payloads should match) and received by the web app interface via UART communication.

The second part will involve creating the simulated network by having one Raspberry Pi send TCP packets to the other through the network switch, and then monitoring the SPAN port on the switch (port 4 will be the port we set up to mirror all traffic coming out of port 3 on the switch) via Wireshark. The test's success metric will be quantifiably measured by comparing the packets sent and received by the Pi's against the packets shown in Wireshark.

Pre-Testing Setup Procedure:

Raspberry Pi Side:

1. Add client and server c socket programming files to two different SD cards via an SD card reader (these will be used in the two Raspberry Pis)
2. Connect the Raspberry Pi's to monitor with mouse and keyboards
3. On one Raspberry Pi go into server directory and compile with gcc server_pit.c -o server
4. On the other Pi go into client directory and compile with gcc client_pi.c -o client
5. When running the test first initialize the server by running ./server
6. Once the server is up, run ./client

FPGA Side:

1. Set up FPGA to be connected to the desktop via Ethernet and UART (USB)
2. Open .xpr project on Vivado
3. File -> Export -> Export Hardware…
    a. Check include bitstream, click OK
4. File -> Launch SDK -> Local to Project on both settings
    a. Window pops up with echoserv_bsp and echoserv
    b. Click Xilinx -> Program FPGA -> click program
    c. Right click echoserv -> Run as -> Launch on Hardware (System Debugger)
5. Open Tera Term and configure the settings to monitor IP 192.168.1.10 (Microblaze server IP on the FPGA) on port 7
    a. In Tera Term, click Setup -> Terminal
    b. Check local echo and change transmit to be CR+LF
6. Run node testUart.js in command prompt on the desktop connected to the FPGA

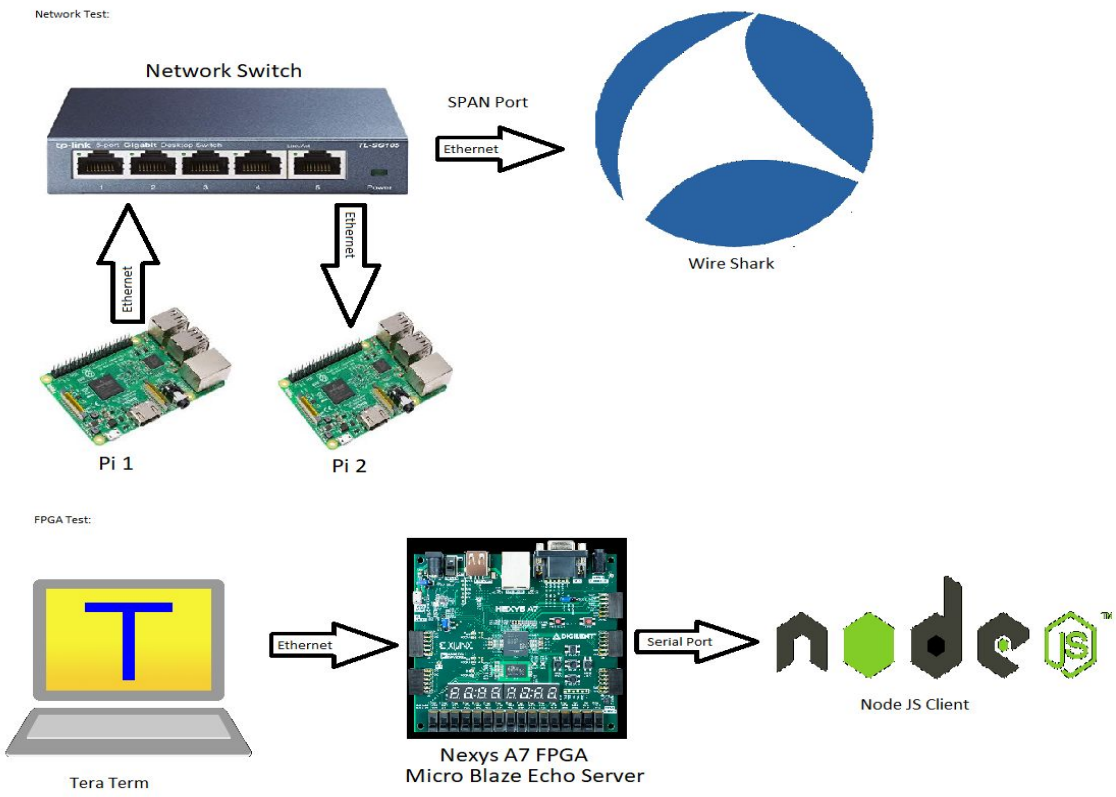

Figure 1: Illustration of Setup and System Flow

Testing Procedure:

Part 1: Echo

1. Go to localhost:8001:/main.html
2. Send message of max length 3 in Tera Term at a max speed of 1 message per sec
3. FPGA blinks upon receiving and echoing back data to the SPAN port
4. Show packets being displayed on web client
5. Repeat steps 2 to 4 to send messages

Part 2: Test Server Proof of Concept

1. Connect both Raspberry Pi's to the network switch via ethernet cables (server PI to port 2 and client PI to port 3)
2. Connect a laptop with Wireshark via ethernet to the network switch on port 4 (mirroring port/SPAN port)
3. Setup Raspberry Pi's as done in the above Raspberry Pi server setup
4. Run wireshark on laptop and select ethernet adapter as the interface

a. Able to verify TCP packets being sent from test server which will be utilized to prove the success of our FPGA tap
b. This shows that protocols are able to viewed via the span port, which our FPGA will be able to eventually read and parse

Measurable Criteria:

The criteria for successful running and output is as follows:

a. For Part 1: Echo
    1. FPGA is receiving the network activity, shown through its blink on the board
    2. Network activity is sent to the Node.js, shown through the web page
    3. There is less than 10% packet loss from the desktop to the FPGA to the Web Page in x amount of samples
b. For Part 2: Test Server Proof of Concept
    1. The Raspberry Pis are sending network activity to one another
    2. Can connect to and monitor this network activity using Wireshark
    3. Checking to make sure all packets sent by the Pis are being caught/monitored by Wireshark through the network switch's SPAN port (configured as port 4 on switch).
    4. There is 0% packet loss from pi-to-pi and all packets are captured in Wireshark.

Score Sheet:

Part 1: Echo

Packet Sent (Payload)	FPGA blink? (Y/N)	Match on Web Client? (Y/N)
Result: _____%		

Part 2: Test Server Proof of Concept

Packet Sent (Payload)	Matched on Client Pi? (Y/N)	Shown on Wireshark? (Y/N)
Result: _____%		