Configurable Ethernet Frame Generator

CprE 583 Project Final Report

Adam Pfab & Tim Polehna

Introduction

Our project was meant to provide the ability to generate Ethernet traffic at very high, constant rates for testing hardware and software network stacks. A Linux PC is able to generate random Ethernet frames with specific data, but many of the PCs available for testing like this are not able to produce the data at high and consistent rates. FPGAs offer a highly parallel architecture and no interference in processing from other tasks running on the system. This makes them a good candidate to achieve the goals of the project.

This report is meant to be supplemental to the project presentation and slides. The information contained in the presentation was primarily focused on the details in the implementation and what pieces of the design were meant to do. Shortcoming and final project status are covered in this report.

Design/Architecture

Our design approach had the following in mind:

- Keep it simple
- Use standard approaches/concepts
- Allow for easy expansion
- Provide a way for the user to interact
- Use as little resources as possible for storage

By keeping our design simple and standard concepts, such as state machines and register banks, we were avoiding errors that can happen due to complex architectures. Because this project was meant to be the starting point for a more complex Ethernet generator, one with more types of packets, we decided that several of the components needed to be easily reproducible. The multiplexing frame generator was designed using a component generic. This allows it to multiplex any number of configurations.

We had discussed a few ways to configure the device. It could have been done with a special configuration pack or via RS-232. The RS-232 interface was chosen to eliminate some of the complexity involved with buffering the configuration information and having to create a program that would generate the configuration message. RS-232 provided the quickest way for getting the configuration information onto the device.

A main bank of block RAM was desired to store the configuration information. That information could then be directly used by the user interface and the frame generator. Individual registers would only then be needed as utilities for the implementation in the different modules. In the frame generation, this method could be referred to as on-the-fly creation.

Shortcomings

With the amount of that could be dedicated to this project, certain features had to be left out. The largest piece of the project that was left out was the checksum generator. We set the checksums to values that disabled their usage, but ideally the checksums would have been valid. Originally, we had planned to use checksums generators that were available through Xilinx or VHDL algorithms freely available on the internet. Since the proper functionality of project did not depend on having checksums enabled in the headers, it was decided to remove them. This saved time by removing complexity from the frame generator.

The device currently only supports IP packets with UDP or TCP data. For this to be a true Ethernet frame generator it should support other protocols, such as ICMP and RTP. Ideally it would even support custom data frames where the payload for the Ethernet frame is completely hardcoded by the user. This was something that was mentioned in the presentation as a possible feature if time allowed. Along these lines, it would be nice to allow the user to change any field in any header without having to hardcode the entire frame. This is a very complex feature to implement so it was scrapped for the current project, but would definitely be something with added benefit later. Also mentioned in the presentation, we had wanted to possibly make an ICMP echo configuration. This would have been nice for determining if the device was properly connected to the network.

The user interface is quick, but requires that user enter the information in a very specific format. It would be nice to either make the serial output more of a menu type setup. Another possibility would have been to create a serial application that can control the device using the input method currently employed.

Enhancements

As mentioned in the shortcomings, it would be ideal to add the CRC generator, have a completely user defined frame, allow the user to change any field in any header, and also have an easier to use user interface. Beyond that, it may be nice to increase the operating frequency of the design to allow for shorter packet transmission times. Another enhancement that could be made to improve output rate consistency would be to adjust the round-robin scheduling algorithm used in the shift-register of the frame generator.

Currently the shift register output algorithm only allows one instance of the frame configuration to be in the queue at a given time. If it the configuration expires before the frame is added to the local link FIFO, the specified rate is lost. As mentioned in the presentation this could be fixed by increasing the operating frequency without decreasing the frame generation interval, but it would be better to expand the shift register so that if each configuration was at the fastest rate all configurations could always be added. This involves a fairly complex computation because of the variable number configurations so it was not even attempted.

Final State of Project

Even with the modifications to the project to remove things like the CRCs, time that could be allocated to complete the project was still small. We met a few times to discuss the challenges we were having when coming up with the implementation for the modules that each of us had decided to take on. We

also let each other know of implementation restrictions that may impact how the other person was implementing their module. In the end, we were not able to produce a fully functioning and tested bit file that could be loaded on the FPGA. Getting closer to the end of the project, our main concern was having all the pieces coded and compiled so they could be tested individually. Once each piece of the design was functioning, then they could be pulled together into a top-level design and tested. Many of the modules were tested, but a large one, the frame generator, was not tested.