Field-Programmable Gate Array Telephony

6.111 Final Project Proposal

Kiarash Adl
Nandi Bugg
Sachin Shinde

I. Overview

With a wide array of applications ranging from telecommunications to distributed computing, networking has greatly enhanced our ability to share information and resources. In integrated circuit (IC) design, emulating hardware that interfaces with a network may in some circumstances benefit from emulating the corresponding network as well.

In this project, we plan to use standard networking design principles to create a system for sharing data between many Field-Programmable Gate Arrays (FPGAs) running in parallel. Specifically, we shall implement a fast and reliable method of packet transfer between FPGAs through wired networks with multipoint links. The functionality of this method shall be validated through the development of a telephony system equipped with features such as Call Rejection, Call Blocking, Call Forwarding, Call Waiting, Caller ID, Conference Call, and Voicemail.

To facilitate independence between modules and manage complexity efficiently, we shall base the network component of our system on the Open Systems Interconnection (OSI) model. This model traditionally separates the functionality of a communications system into seven layers of abstraction: in order from highest to lowest, Application, Presentation, Session, Transport, Network, Data Link Control (DLC), and Physical Interface (PHY). We adopt the convention of assigning each of these layers to single Verilog modules, except for the bottom two layers. For each of these layers, a copy is instantiated for each transceiver on the FPGA.

II. Implementation

At the end of the document is a high-level block diagram illustrating data flow between modules. Each of the OSI layer modules can be thought of as providing services to the OSI layer modules above it. For the sake of simplicity, we'll start our description of functionalities with the lowest layer module and work upwards.

Physical Interface (PHY) [Adl]

This module can be thought to provide a virtual bit pipe to the DLC module above it, not immune to errors or collisions.

At the front end, the module has a full-duplex data link with the DLC module, with which it receives outgoing packets and sends incoming packets. The back end of this module, although not depicted in the block diagram, has a serial full-duplex link with a transceiver. It also controls the driver enable (DE) bit of the transceiver, but not the receiver enable (RE) bit, as the latter is always active low so that the module may always read the physical data on the link.

When the PHY module receives packets to send, it uses an 8b/10b encoding to achieve DC balance and facilitate clock recovery. It will also be responsible for adding headers and footers to signal the beginning and ending of a frame. The transceiver will read a logical high when the bus is not driven; it is the job of the PHY module to interpret from this when the bus can be assured to be idle, and relay that data as a signal to the DLC module. For the raw signal coming in from the transceiver, the PHY must perform clock recovery by sampling the signal at a rate much faster than the data rate, and aligning the clock using the transitions in the signal. Upon detecting the beginning of a frame, it will then attempt to use 8b/10b decoding on the synchronized signal until the end of the frame is detected. If the signal is too degraded to recover or the 8b/10b decoding encounters an unused codeword, the remainder of the signal is discarded and transmission to the DLC module is properly halted if it has been initiated. The PHY module must also detect data collisions when driving a signal by comparing the received bit stream to the driven bit stream, and send the result to the DLC module so that it may cease transmission.

Data Link Control (DLC) [Shinde]
This module can be thought to provide a point-to-point virtual link for reliable transmission of error-free packets (in order) for the Network module above it.

At the front end, the DLC module has a full-duplex data link with the Network module, with which it receives outgoing packets and sends error-free incoming packets. The back end of the DLC module is as described in the previous section. Operations on outgoing packets will only involve adding headers and/or footers and will not depend on the actual packet data, in accordance with the layering abstraction. These headers and footers will be removed from incoming packets.

Because links in our system are multipoint, we must divide the DLC layer into two sublayers: a Logical Link Control (LLC) sublayer on top, and a Media Access Control (MAC) sublayer on bottom. The latter has the purpose of controlling access to the network to avoid collisions, while the former has the purpose of providing the error-free, reliable virtual link. Both functionalities are implemented within each DLC module.

The MAC protocol will be Carrier Sense Multiple Access with Collision Detection (CSMA/CD), utilizing truncated binary exponential backoff. The bounds for truncation will be determined from testing. The probabilistic nature of this protocol necessitates Pseudorandom Number Generators (PRNGs), which shall be implemented with Linear Feedback Shift Registers (LFSRs). For the LLC protocol, error-detection shall be performed with a Cyclic Redundancy Check (CRC), with generator polynomial to be determined from testing. Packet reliability shall be guaranteed with Selective Repeat Automatic Repeat Request (Selective Repeat ARQ) with piggybacking. This ARQ Protocol will also be responsible for managing multipoint link initialization and disconnect.

Network [Shinde]
This module can be thought to provide an end-to-end virtual link for reliable transmission of error-free packets for the Transport module above it.

At the front end, the Network module has a full-duplex data link with the Transport module, with which it receives outgoing packets and sends error-free incoming packets. The front end will also have a simplex data link for transmission of addresses of outgoing packets, as well as a full-duplex data link for control signals. The back end of the Network module is as described in the previous section. Operations on outgoing packets will only involve adding headers and/or footers and will not depend on the actual packet data, in accordance with the layering abstraction. These headers and footers will be removed from incoming packets headed toward the Transport module, or possibly modified for incoming packets headed toward another DLC module.

The Network module has two major functions: datagram routing and flow/congestion control. If the physical link proves significantly noisy, another layer of error-detection will be also performed at this stage with a CRC and Selective Repeat ARQ with piggybacking. The Network module is also responsible for initializing the network. When links are dropped, the Network module must retrieve the packets in its downstream DLC modules and reroute them. An appropriate flow control technique shall be decided once further information is gained regarding memory and latency constraints.

Since the Xilinx Virtex-II XC2V6000 FPGAs do not come equipped with hardware-encoded identification strings, there is no immediate way to distinguish between FPGAs. To resolve this, PRNGs implemented with LFSRs will run immediately after power-on reset. After the links initialize, the network will send network control packets using a reserved address that sends to all nodes on a link. Upon the first reception of an error-free packet, a node will store whatever value its PRNG generates at that moment as its initial address. It will then compare with other nodes in the network to manage address contention, after which the addresses will be mapped to simpler bit strings to reduce overhead. This address will then be available to higher layers through the control signal link.

Transport [Adl]

This module can be thought to provide a virtual link for end-to-end transmission of error-free messages for the Session module above it.

At the front end, the Transport module has a full-duplex data link with the Session module, with which it receives messages to be disassembled into packets and sends messages assembled from received packets. The front end will also have a simplex data link for transmission of addresses of outgoing messages, as well as a full-duplex data link for control signals. The back end of the Transport module is as described in the previous section. Operations on outgoing messages will only involve regrouping of data into packets as well as the addition of headers and/or footers, in accordance with the layering abstraction. These headers and footers will be removed from incoming packets, and the packets regrouped into their original messages.

The Transport module has the main purpose of partitioning outgoing messages into packets and merging incoming packets (which may arrive out of order) into messages. It may also need to guarantee reliability in the case that the network becomes disconnected, and provide end-to-end flow control functionality (e.g. limiting incoming packet rates) if Network module flow control cannot handle congestion within the network. If multiple sessions exist with the same source and destination, then the Transport module may multiplex the sessions so that they appear as a single session to the Network module, which reduces packet overhead and increases throughput.

When packetizing speech, we shall attempt to adhere to Real-Time Transport Protocol (RTP) standards [1] and the RTP payload format for the Internet Low Bitrate Codec (iLBC) [2], which is used in the Presentation module to compress speech. However, in the context of this design problem some features offered by RTP are superfluous, and are granted at the expense of packet overhead. We will thus only use RTP as a rough base upon which to construct our Transfer Protocol.

Session [Adl]
This module can be thought to provide a virtual communication session for the Presentation module above it. A session may be thought of as a correspondence of messages between two nodes in the network.

At the front end, the Session module has a full-duplex data link with the Presentation module, with which it receives service requests and sends service responses. The back end of the Session module is as described in the previous section.

The Session module is tasked with the purpose of starting, maintaining, and ending a session. For instance, the Session module is responsible for building-up a session at the start of a telephone call, maintaining that session for the duration of the call, and breaking it down at the end of a telephone call. The Session module becomes crucial for tasks that require synchronization, and can be used to limit access rights.

Presentation [Shinde]
This module converts data sent from the Application module into a format suitable for the network and sends it to the Session module (and vice-versa). In particular, it is tasked with compressing voice samples to a lower bit-rate for packet transfer and decompressing compressed voice samples to a higher bit-rate for playback. If time permits and we are able to develop a public-key encryption for speech, it will occur after compression for voice data being sent, and decryption will occur before decompression for voice data being received.

The specific audio codec used for speech compression is the Internet Low Bitrate Codec (iLBC), detailed extensively in [3]. It processes 8 kHz 16-bit audio in frames of 20 ms, compressing each frame into 304 bits. This results in a roughly eight-fold reduction in bit rate from 128 kbps to 15.2 kbps, while also reproducing sound fairly well with a Mean Opinion Score (MOS) [4] of 4.14 out of 5 using the Perceptual Speech Quality Measure (PSQM) algorithm [5]. The independence between data in compressed frames results in graceful speech quality degradation if frames aren’t received in time due to packet loss, making

the algorithm suitable for packet-based voice-communication networks in which packet delay can be large. The other major benefit of iLBC is that the algorithm is open-source and royalty-free, licensed under the BSD 3-Clause License.

The algorithm itself is fairly complex, requiring several floating point operations. We will perform these operations as accurately as necessary to produce acceptable speech reproduction. To minimize the use of resources, all steps of the algorithm will utilize a limited set of fixed-width multipliers and adders, allocated through an access control mechanism. The size of this set shall be determined near the end of the project, when better estimates of resource consumption on the FPGA can be made.

Application [Bugg & Shinde]
This module can be thought to provide common network services for the User Interface module above it.

In general, the front end of this module will take as input service requests from the UI module and output service responses to the UI module. The format of a service request consists of a request control signal, with possible 16-bit 8 kHz audio input and/or address input(s). A service response consists of a response control signal, with possible 16-bit 8 kHz audio output and/or address output(s). The basic service requests that the Application module can receive will be:

- get your own telephone number
- dial a call
- end a call
- accept an incoming call
- reject an incoming call
- place a call on hold
- forward an incoming call
- call another party while in the middle of a call (conference call)
- accept an incoming call while in the middle of a call (call waiting)

User Interface (UI) [Bugg]
The functionality of the User Interface is multi-faceted. On one end, it manages communications with the user, with user inputs being read from the FPGA's switches and buttons and user outputs being fed to the FPGA ASCII 16-character string display (using the 6.111 display_string module). On another end, it interfaces with the FPGA AC '97 Audio Codec LM4550 chip, sampling 16-bit audio at 8 kHz from the microphone and sending the same-quality audio to the headphones. At a third end, it interfaces with the Application module, issuing service requests and receiving service responses.

The UI module will specifically implement the components of UI features that don't rely on the network. For instance, the mixing between voices during conference calling specifically occurs in the Application module, with the Application's audio output as the mixed output. However, when a customer is in the middle of a call, and he receives another call (which will result in the Application module of the customer's FPGA issuing an appropriate service response to the UI module), it is the job of the UI module to mix another tone into the audio output from the Application module to alert the customer that another call is on the line. It is also job of the UI module to manage local storage, such as voicemail on Compact Flash, address books, and speed dials.

The ASCII 16-character string display will be the primary visual output device. Since messages from the UI will not always fit completely on the ASCII display, they will scroll across the display. There will be menus for users with options that change depending on call status. The buttons on the FPGA will be used for menu navigation as well as volume adjustment during calls. Dial tones and notification sounds (as well as non-personalized voicemail greetings) will be stored in BRAM on FPGA power-on.

III. External Components

Each physical link in the network consists of a serial twisted-pair cable, with transceivers tapping into the cable in a daisy-chain configuration to minimize driver signal reflections at the stubs. Differential signaling in the twisted-pair cable greatly attenuates the effects of electromagnetic interference, and both ends of

the cable are terminated by 100 Ohm resistors for impedance matching.

The transceiver chips are Texas Instrument SN75HVD10P DIP8 half-duplex transceivers that conform to or exceed TIA/EIA 485-A standards [6], with maximum signaling rates of 32 Mbps and open-, short-, and idle-bus internal failsafe. In particular, the idle-bus failsafe permits the detection of an idle bus in the MAC layer of the design, which allows for more sophisticated MAC protocols to increase throughput.

Even though the transceiver's maximum signaling rate is specified as 32 Mbps, the Xilinx Virtex-II XC2V6000 FPGA can only sustain signaling rates at its physical user ports of approximately 20 Mbps without significant signal degradation. Furthermore, operating the SN75HVD10P at maximum signaling rate places a limit on cable length before timing jitter renders the signal unusable (about 40 feet). To alleviate both issues, we will operate the physical link at signaling rates not exceeding 20 Mbps, with possibly stricter limits pending further testing and validation.

IV. Testing

Verilog testbenches for behavioral simulation will be developed for each module in the system, with the exceptions of those modules that interface with complex IC chips (such as the LM4550). For such modules, separate testing techniques will be used to ensure proper functionality. Once individual module functionality is verified, the OSI layer modules will be stacked from the bottom up incrementally, with testbenches being developed after each increment, to ensure that the layer modules interact as they should.

Once all behavioral simulations are completed, real-time tests will begin on the FPGAs. UI features will be extensively tested, and three separate network topologies will be used: a single multipoint link (i.e. a shared bus), multiple point-to-point links, and multiple multipoint links.

V. Extensions

Should time permit, there are extra features that we would attempt to implement.

- As this project is proof of concept, secure voice-communication is not a necessity. However, should the system be marketed as a consumer product, security protocols must be established to prevent unauthorized third parties from wire-tapping. We would use public-key encryption (e.g. RSA) to keep calls private.
- Data network statistics are useful to network designers, as it allows them to redistribute network resources and alter protocol to manage load efficiently. Thus, collecting such statistics and storing them at each node in Compact Flash or displaying them in real-time may be a worthwhile endeavor.
- While efforts will be made for the UI to be fairly intuitive, first-time users might still have issues adjusting to the system as designed. Therefore, if possible, a graphical user interface will be implemented for display on a VGA monitor.
- More features may be added to the UI, including
    - Personalized Voicemail greetings
    - Personalized address books
    - Anonymous Calling
    - Do Not Disturb (DND)

VI. Citations

[1] *RTP Profile for Audio and Video Conference with Minimal Control,* IETF RFC 3551, 2003.
[2] *Real-time Transport Protocol (RTP) Payload Format for internet Low Bit Rate Codec (iLBC) Speech*, IETF RFC 3952, 2004.
[3] *Internet Low Bit Rate Codec (iLBC)*, IETF RFC 3951, 2004.
[4] *Methods for subjective determination of transmission quality*, ITU-T P.800, 1996.
[5] *Objective quality measurement of telephone-band (300-3400 Hz) speech codecs*, ITU-T P.861, 1996.
[6] *Electrical Characteristics of Generators and Receivers for Use in Balanced Multipoint Systems*, TIA/EIA 485-A, 2003.

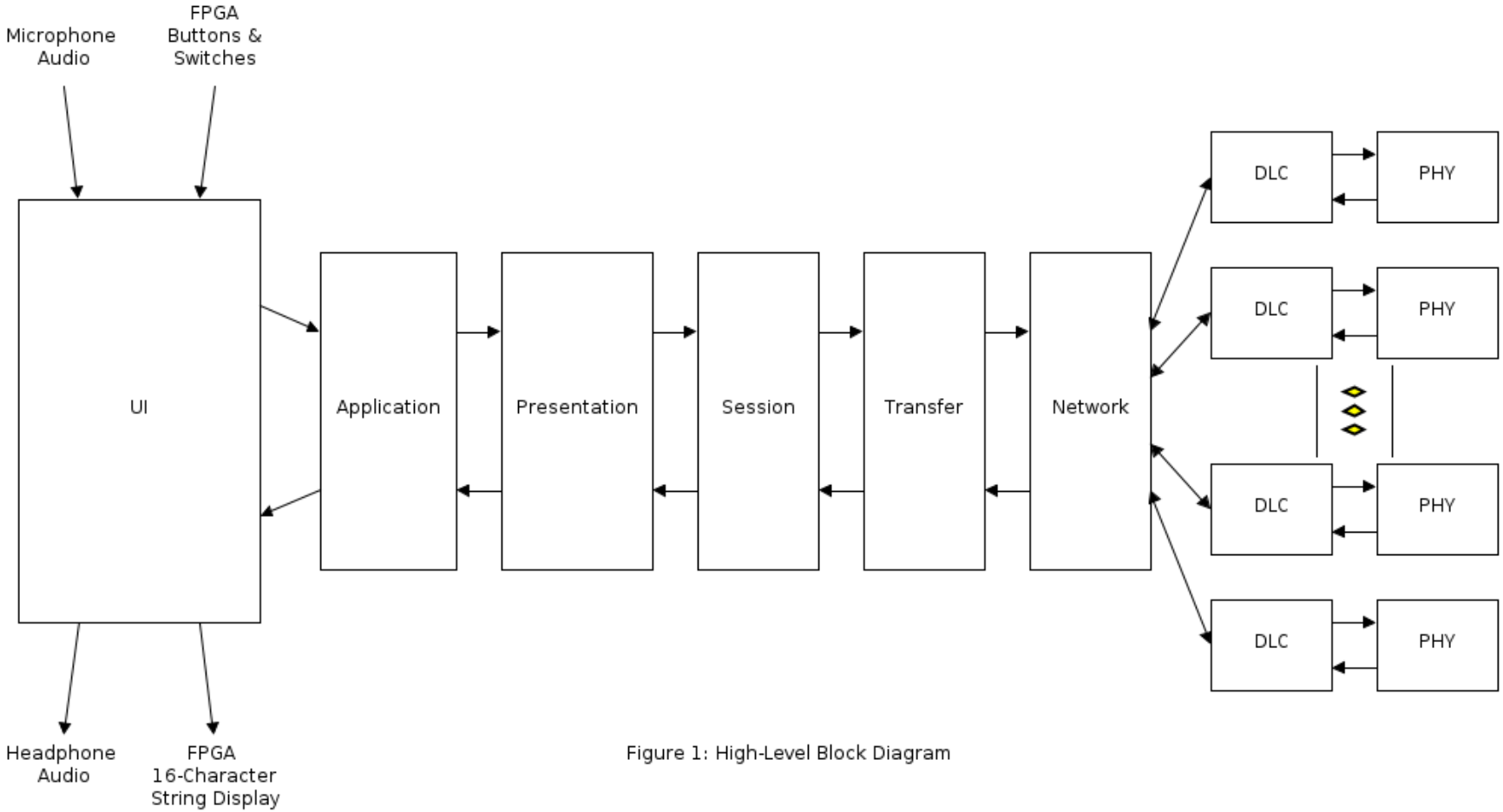

Figure 1: High-Level Block Diagram