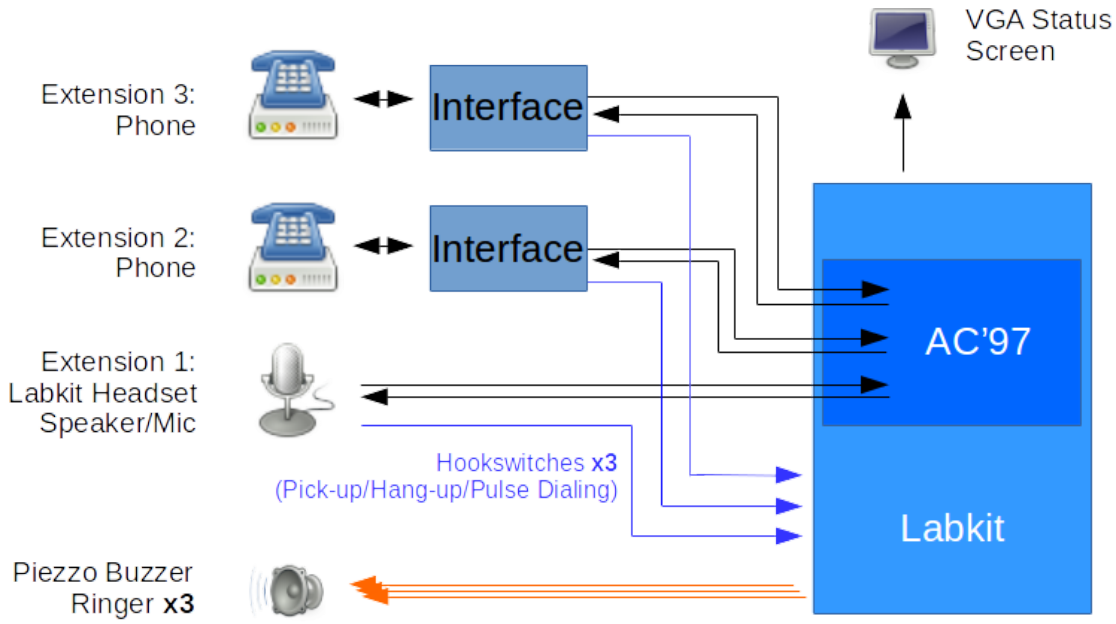
Tristan Honscheid
6.111 Project Proposal
1 Nov 2016

Introduction

This project will aim to construct an FPGA-based telephone exchange. Two landline telephones, combined with custom interfacing hardware, and the labkit-provided headset will be connected to the labkit to function as three phone lines. The headset line cannot dial. The FPGA will operate a state machine capable of decoding dialed numbers, playing dial and busy tones, and connecting calls between the attached lines. Each line has one input and one output channel on the labkit's AC'97 codec so that two phones in a call can be digitally connected by passing audio samples accordingly.



Sample Usage

A sample use case begins with a user lifting the receiver on an attached telephone. He or she hears begins pulse-dialing a number. The FPGA counts the pulses and displays the complete number on the VGA screen. If the number is a valid extension, the FPGA begins signalling the called party's line with a piezo buzzer (actual ringing involves messy high-voltage electronics). Once the called party picks up, the call gets connected. The FPGA begins routing audio between the phones: `phone[1][in]` ➡`phone[2][out]` ; `phone[1][out]` ⬅`phone[2][in]`, where `phone[n][in]` and `phone[n][out]` refer to the AC`97 input and output channels for extension `n`. Once either party hangs up, audio is no longer routed between the phones.

Throughout this process, the user will hear call-progress tones (the familiar dial tone, busy tone, and ringing tone). These signals can be played into the caller's phone line by simply routing the samples of these tones to `phone[n][out]`.

Stretch Goals

If time permits, several expansion features can be considered.

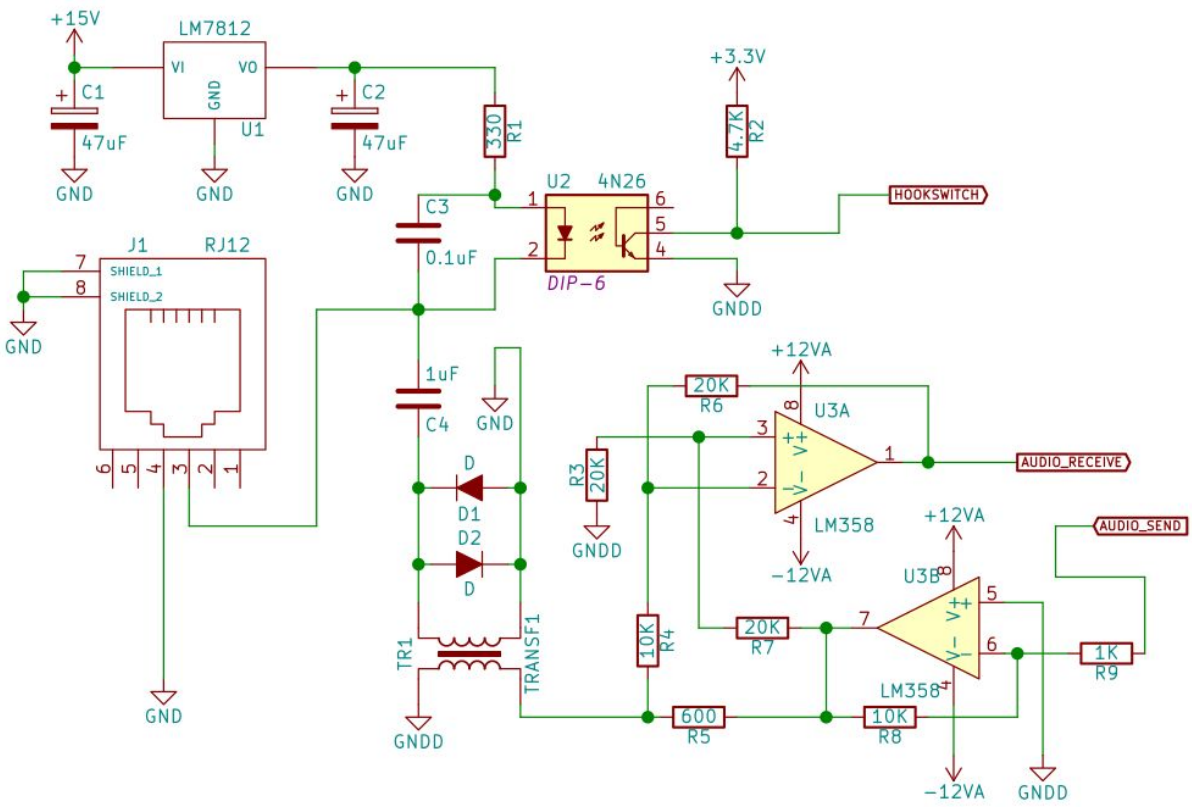
1. **Voicemail:** If the called party does not answer, allow the caller to record a brief voicemail message instead. This message will be stored in the labkit ZBT memory.
2. **DTMF Dialing**: Instead of (older) pulse dialing, modern telephone use DTMF dialing, where each digit (0-9, A-D, *, #) is encoded as a superposition of two voiceband sine waves. For example, a "5" is 770Hz+1336Hz. The FPGA would have to perform some audio signal processing to detect these frequencies and decode the dialed digit.

Special Hardware

Landline telephone interfaces have not changed noticeably in the past 100 years. In other words, there is no straight-forward way to interface a landline telephone to modern digital logic. Special circuitry, shown in the draft schematic below is needed.

To power the phone, a DC talking current must be passed through the phone. Voltage regulator U1 provides a clean DC source, and R1 sets a suitable current. Optoisolator U2 is the hookswitch detector. When the phone is picked up, U2 activates, pulling signal `HOOKSWITCH` low. U2 also flashes on and off during pulse dialing. Therefore, the dialed number can be decoded by counting pulses on `HOOKSWITCH`. C3 couples audio around the non-linear diode (U2 only needs to detect the DC supply current).

C4 couples audio to and from an isolation transformer. D1 and D2 are diode clamps that keep transients (such as those caused by picking- and hanging-up) out of the audio circuitry. On the other side of the transformer is an arrangement of op-amps that perform a 2- to 4-wire conversion (this circuit is called a telephone hybrid). This splits the phone line audio into a separate send and receive signal that can be connected to the labkit AC`97. Level adjustment can be achieved by adjusting the op-amp gains. This hybrid circuit was borrowed from a TI datasheet [http://www.ti.com/product/LMV822-N-Q1/datasheet]



Modules

Several modules are needed to implement the exchange. Below is a listing of anticipated modules, a brief functional description, and rough list of inputs and outputs. (some I/O is omitted to reduce clutter, or vague because exact specifications are to-be-determined)

Parameterized Clock Divider

Ports: divided clock (out), system clock (in)
Several clock dividers will be needed to produce timing signals. For example, the busy tone has a cadence of 0.5s on and 0.5s off. Another clock will be needed for audio processing matching the 48Khz sample rate.

AC`97 Interface

Ports: 6x 20-bit audio channels (one input and one output per phone line), clock (in)
This module will be similar to the AC`97 interface module in Lab 5. It will perform the serialization and AC`97 interfacing. It will also have to perform multiplexing of the audio streams so that audio from multiple input channels can be simultaneously recorded and played back. The module aims to abstract the AC`97 hardware completely, so that the telephone application only needs to interact with the (6) 20-bit audio I/O ports.

Audio Router

Ports: 6x 20-bit audio channels to/from the AC`97 interface, mode select inputs for each line (in), clock (in).
The audio router patches the phone connections together and also generates call progress tones with the help of a sub-module (see Tone Generator). Each phone line has a 6-bit mode select input that describes what its input and output audio connections should attach to:

Mode[2:0] - Destination of nput connection (audio from mouthpiece)	0 - No Connection 1 - Extension 1 audio out 2 - Extension 2 audio out 3 - Extension 3 audio out 4 - Voicemail recorder (stretch goal)
Mode[5:3] - Source of output connection (audio sent to earpiece)	0 - No Connection (silence) 1 - Dial tone 2 - Busy tone 3 - Ringing tone 4 - Extension 1 audio in 5 - Extension 2 audio in 6 - Extension 3 audio in 7 - Voicemail playback (stretch goal)

The router uses conditional statements to copy samples between the requested connections.

Parameterized Tone Generator

Ports: system clock (in), 20-bit audio samples (out)
This module generates call progress tones (busy, dial, ringing) by recalling audio samples stored in memory and outputting them. The module accepts parameters that specify the location of the samples in memory and the cadence of the tone (for example, on for 2 sec, off for 4 sec). This allows one type of module to be instantiated for each of the three tones within the Audio Router.

Line Interface

Ports: Hookswitch (in), Ring Piezzo (out), Ring enable (in), Dialed digit (out), Digit available (out), Phone state (out), Clock (in)
The line interface interacts with the landline phones. It will:

- Monitor the hookswitch for brief dialing pulses. If a number is decoded, it is outputted.
- Monitor the hookswitch for pickup/hangup conditions. Phone state is updated accordingly with 1=picked up, 0=phone down.

- Drive the piezzo ringing buzzer with a tone, at a cadence of 2s on and 4s off when Ring enable goes high.

A similar module, but without dialing features, will exist for the headset-based line as it can only receive calls.

PBX Core

Ports: all Line Interface connections, phone mode select signals to Audio Router, clock (in), phone state signals (out)

The PBX core is a state machine that operates the exchange. It monitors all line interfaces and track each phone's state (e.g. Hung-up, dialing, call connected) through the life cycle of a call. It interprets dialed numbers and directs the Audio Router to make connections. It also outputs information about each phone (e.g. dialed number, call state) for display on the VGA screen (see VGA module)

VGA / Graphics Module

Inputs: phone state inputs from PBX core (in), clock (in), VGA signals (in), pixel (out).

This module, very similar to Lab 3's graphics, will draw text and symbols onto the screen, likely referencing a character generator ROM with raster data for each symbol. Some additional modules are needed to drive the VGA DAC, but these have already be written during Lab 3.