Abstract

This report details the design and implementation of a digital guitar tuner using an Actel ACT1 FPGA. This tuner is intended to encompass both accuracy and information exchange with the user. Several distinct conceptual blocks were derived, from which more detailed design of both the hardware and software was conducted. Each component was thoroughly debugged and tested with special regard to implementation on an Actel FPGA throughout. Once the simulation performed as expected, final preparations were made to burn an Actel FPGA and to connect it to the system. Detailed documentation of the VHDL code, the schematic used for the prototype system, and a data sheet were prepared for possible future use and as good engineering practice.

Overview

This project implements a digital guitar tuner with microphone or line input, and displays the frequency difference with additional information to the user so that they may tune the guitar accordingly. The tuner should be accurate enough to tune within half a cycle.

The user is presented with a six-key keypad, corresponding to the standard tuning on a six-string guitar (Low E, A, D, G, B, and High E), in addition to a reset. A two-digit LED display with three additional labeled LEDs – Ready, Hi/Low, and Error – exist to guide the user in the use of the guitar tuner. In addition, a microphone or line input (if the guitar has string pickups) is available for the user to use with the guitar.

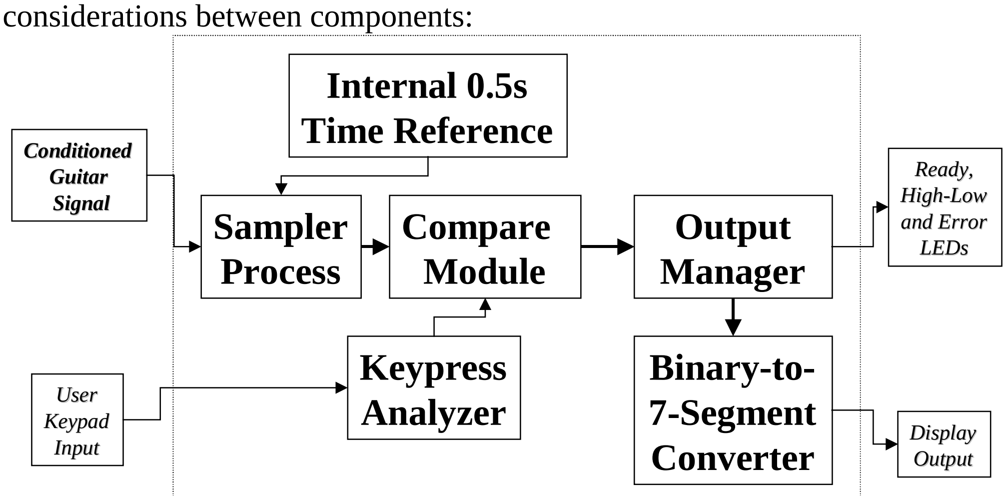
The system sits in the Ready state and displays “00” until a string button is pressed. The tuner will then begin to sample the signal every second. The ready light will go off, and the user will pluck the string manually at which point there are three possible outcomes:

1. The tuner will detect the correct frequency and display “00” with the Ready LED on.
2. If the frequency is different, the tuner will indicate the difference in frequency from the known correct value, and turn the Hi/Low LED either red or green, respectively. Another sample is taken iteratively until in tune or out of range (see below).
3. If the frequency is out of range, or the guitar has not been plucked loud enough or at all, the display will show “99” and turn the Error LED on. The tuner will continue to sample iteratively until the correct frequency is achieved.

The guitar signal must be preconditioned to a digital signal by overamplification and Schmitt triggering as described below.

Design Details

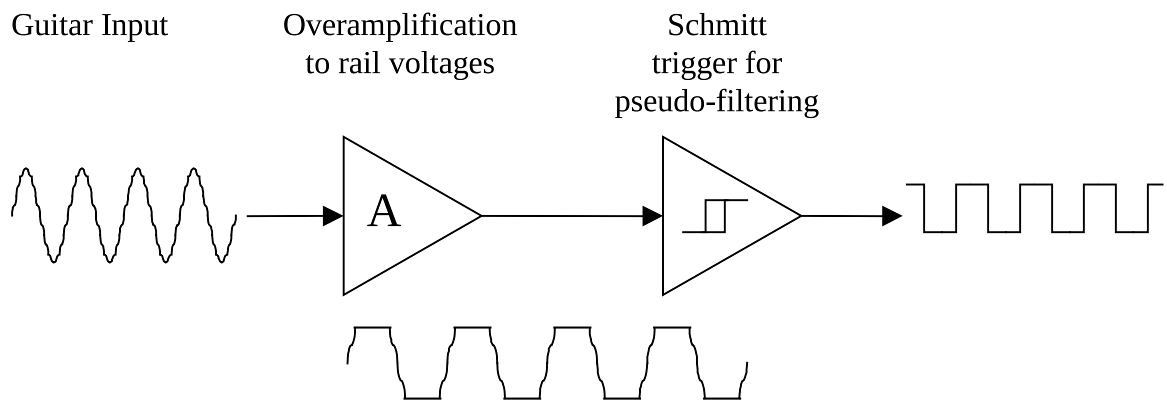
A block diagram of the system is shown below to indicate the design flow and the interface considerations between components:



The general design methodology was to partition the design into as many independently-functioning component processes as possible. (For specific hardware, refer to Appendix B – Hardware Configuration.) The keypad routine detects the particular key being pressed and loads the corresponding reference frequency. The sample routine counts the number of transitions of the input guitar signal when the timing reference is high and initializes the count to zero when the timing reference is low. The compare routine finds the difference of the input and the reference frequency and updates the ready, error and high/low flags when the timing reference is high. The output routine updates the display and sets/resets the rdy, err and lohi LEDs when the timing reference remains low.

Conditioned Guitar Signal (External Hardware)

A guitar signal is essentially a sinusoidal waveform with smaller-amplitude harmonics mixed in. By exploiting several features of analog hardware and discrete components, the guitar signal can be converted into a square wave normalized to 5V logic levels. Consider the following block diagram of the guitar conditioning circuit:



The guitar input comes in to the amplifier (a standard LM741 op-amp configured with a gain of A=-Rf/Rin gain). Though inherently it requires amplification, by setting the op-amp gain much higher than what it would normally be able to accommodate, the guitar signal literally runs into the positive and negative rails and clips the rounded portions of the waveform. There are still harmonics that need to be removed before the signal can be fed to the FPGA. This is accomplished by the TTL Schmitt trigger, whose hysteresis prevents the smaller-amplitude harmonics from retriggering the output either high or low. Instead, the fundamental dominates and produces the digital output at levels compatible with the FPGA.

User Keypad Input (External Hardware)

As stated previously, the user keypad input consists of six buttons that correspond to each of the six guitar strings found on a guitar, in addition to a reset button. None of these buttons are debounced because signal debouncing is not an issue in this design. Essentially, when the user selects a string, that user needs to select it only once. Subsequent presses of the same button (as may be registered with an undebounced switch) will not affect the system. The keypad was also not multiplexed due primarily to ease of internal design, but also because of the availability of user I/O on the FPGA. A 10k resistor was used to limit the current into the FPGA pad from the 5V supply when the switch was closed.

For the reset circuit, a "power-on" reset was implemented in two steps. First, the input had a first-order RC circuit with the reset button normally open to ground. The value of RC was selected to give a long enough reset time for the internal logic to be defined. A value in the millisecond range was chosen, and with a time constant of RC = (1k)(1uF) = 1ms, the value would settle close to 5V in about 4 time constants (or around 4ms). Thus, while the rest of the circuit was powered to 5V, the reset input was effectively held low for the 4ms period. The second step was to normalize it to digital 5V logic levels, which involved the use of cascaded inverter gates from a 7404 TTL chip. One first inverter was used to condition the input, while another inverter was to re-invert the signal to normal levels.

Keypress Analyzer

The keypress analyzer is an important component of this system. It interprets user keypad I/O, and loads the appropriate reference frequency for comparison with the sampled signal. There are specific frequencies associated with each string, which are tabulated below along with the corresponding binary equivalents:

String Name	Scale Reference	Frequency (Hz)	Binary Equivalent (nearest integer)
Low E	E3	82.4	001010010
A	A4	110	001101110
D	D4	146.8	010010001
G	G4	196.0	011000100
B	B5	246.9	011110111
High E	E5	329.6	101001010

The VHDL code that implements the assignments for each of the buttons pressed simply assigns the corresponding reference frequency to a latched variable. This latched variable is passed on to the difference routine for processing. The code for this and other components of the FPGA tuner may be found in Appendix A – Code Documentation for Design.

Internal Half-Second Time Reference

In order to sample the frequency properly, a time reference is required for the sample circuit. This is provided by the internal half-second time reference. Because most logic devices work with binary representations of numbers, it is easiest to divide by two (since this implies a simple bit shift). In order to get an accurate reference to the half-second frequency required, the system clock frequency must be dividable by two – in other words, it must be an integer power of two. For the purposes of the project, a system clock frequency of 16384 Hz was selected because:

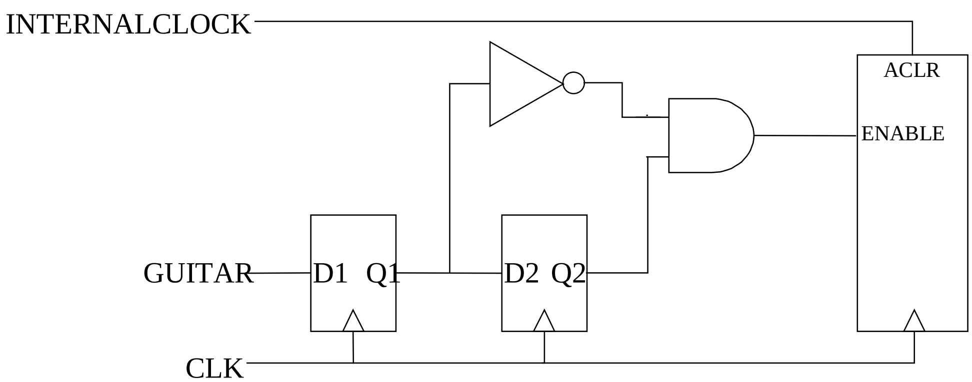
- It is evenly divisible by two successively down to one second.
- It was more than one order of magnitude faster than the fastest expected guitar frequency, which accommodates proper sampling of the guitar signal.
- It prevented timing problems associated with gate delays and setup times.

- It was easily implemented using an astable and adjustable 555 timer circuit.

In the VHDL code, a counter was used whose bit width was determined by taking the base-2 logarithm of the clock frequency,
i.e. log(16384) / log(2) = 13 implies a 13 bit wide counter.
The counter is set to increment continuously on each system clock rising edge. The derived signal will simply be the most significant bit of the counter, or bit 13, because it will change every second as the counter increments, overflows, and resets.

Sampler Process

The algorithm for the sample routine is shown below in schematic form. The inverter-AND gate combination detects low-to-high transitions of the guitar signal and enables a 14-bit counter. The counter increments when the timing reference signal - internal clock is high, otherwise it resets the count to zero when the timing reference is low. The sample routine sets or resets an internal flag called the init_flag_sample depending on whether the count is zero (no guitar signal) or not. This flag is used by the output routine to update the displays and the output LEDs.



Compare Process

The compare routine essentially finds the absolute difference of the input and the reference frequency and sets/resets the output LED flags. If the difference is greater than 99 then the error flag is set else it is reset. If the difference is less than 3 then the ready flag is set indicating that the input frequency and the reference frequency almost match. An internal flag called the init_flag_compare is also set or reset at this point. If the input frequency is greater than the reference frequency then the lohi flag is set to 01 else 10.

Output Manager Process

The output routine OR's the init_flag_sample and the init_flag_compare flags from the sample and compare routines respectively with the reset signal and generates the init flag. The init flag is checked before updating the output LEDs and the display. If the init flag is set, it indicates that either a reset is in process or no guitar signal is present or the guitar signal frequency is matched to the reference frequency selected. In either case the display shows "00", the ready signal goes high, the error signal goes low as do the lohi signals. The output routine sends the 7 most significant bits to the display routine.

Binary-to-Seven-Segment Converter

The conversion of binary data (the conditioned “difference” from the output manager component) was conceptually a difficult task. Initially, combinational logic equations and conversion processes based on state machines were considered. However, using a case statement with multiple “when” conditions proved to be the best conversion. The input is a seven-bit binary number from the output manager, and the outputs are the MSB and LSB (in decimal) for the seven-segment displays. The VHDL code compiled in only 64 modules, due in large part to the Actel synthesis tool’s ability to minimize logic equations but also to the repetitive nature of the output (0 to 9 LED values are reused throughout). Since the display could only show numbers up to 99, a case for values outside of this range was implemented to blank the display.

Ready, High-Low and Error LEDs and Display Output

This was the simplest portion of the hardware design. The ready and error LEDs were single-color LEDs connected to sink current through the FPGA. For the High-Low indicator LED, a multi-color LED was chosen for indicator flexibility and future expansion of functions. Only one color at a time was used in this design as is evident from the VHDL code for the output manager. For each seven-segment display, a common-anode LED array is used with no multiplexing at the outputs. This is again due to the availability of I/O pins on the FPGA. All LEDs have resistor values of 330 ohms due to I/O maximum current constraints of 20mA per pin, as well as part availability. A maximum current of about 10 mA was used and a 1.5 V drop through the LED assumed. Using Ohm’s Law and assuming a supply voltage of 5 V,
i.e. R(LED) = V(net) / I(constraint) = (5 – 1.5) V / 10 mA = ~350 ohms (so use 330 ohms).

Design Verification

To verify the design, a simulation of each portion of the circuit was carried out and debugged individually. This entailed compilation using the Actmap tools rather than Design Architect to ensure that only designs that were synthesized properly were used. The design was then tested using a top-level VHDL file linking each of the components together, and found to perform as expected. A listing of the waveforms may be found in Appendix C – Design Verification Waveforms.

Circuit Test

The circuit was built as per the schematic in Appendix B. At this time, there is no report on how the circuit itself operates; however, all of the hardware external to the FPGA has been fully tested and its operation verified. In addition, a fuse file has been prepared for burning of the FPGA which represents the verified design as described above. As such, there is little doubt that the entire circuit will perform to the intended specifications.

Final Thoughts

This was an extremely challenging project due to the fact that a novel design idea (i.e. VHDL) was the primary tool used. Extensive debugging of the code was required throughout, and even the hardware presented some challenges. Overall, the learning curve for FPGAs was essentially (but not completely) overcome, and will contribute to the betterment of the designers' skills in the future.

References

1. Terry Downs, *T's Technical Notes* – http://pages.prodigy.com/nightshift/freq.htm.
2. David Sutherland, Thomas Serink, Daniel Onischuk and Michael Daskalopoulos, *EE 482 Project – Guitar Tuner,* April1995.
3. Texas Instruments, *TTL Data Book*, 1993.
4. Robert Coughlin and Frederik Driscoll, *Operational Amplifiers and Linear Integrated Circuits*, 3rd edition, 1987.
5. Actel, *ACT1 Series Data Sheets*, http://www.actel.com/docs/databook96/s01_07.pdf, 1996.

Appendix A –
Code Documentation
for Design

Appendix B –
Hardware Configuration

Appendix C – Design Verification Waveforms

Appendix D –
Data Sheet for FPGA