XAPP366 (v1.0) November 7, 2001

Handheld Musical Instrument Tuner

Summary

This document describes the implementation of the Musical Instrument Tuner design submitted to the recent "Cool Module Design Contest". All development for this contest was performed using the Insight Springboard™ development platform which allows for rapid development of Handspring modules. This development platform incorporates the reprogrammable Xilinx CoolRunner™ XPLA3 CPLD and uses the Handspring Visor™ PDA expansion slot. Low power CoolRunner CPLDs are the ideal programmable logic solution for portable, handheld applications.

The Musical Instrument Tuner employs the Handspring microphone to acquire a tone from an instrument and then displays the measured tone and its variance from a desired frequency on the Handspring LCD. This application also provides a fixed tone output, thereby functioning as an electronic pitch pipe. The CPLD and Handspring design files for the Musical Instrument Tuner are available and can be found at **Download Pack**, page 10.

A visual presentation of the Musical Instrument Tuner is provided to show operation and functionality. This on demand video is available on the Xilinx website at: **http://www.xilinx.com/apps/video.htm**.

Introduction

In January of 2001, Xilinx, Handspring and Portable Design Magazine announced the "Cool Module Design Contest," a design contest that highlights the use of Xilinx CoolRunner XPLA3 CPLDs to quickly develop Springboard modules for the Handspring Visor PDA. About 250 contest registrations were received and nearly 100 contest design ideas were submitted. From these submissions, ten were chosen to receive a Handspring Visor and an Insight Springboard development kit. These ten finalists were given three months to complete their designs in order to compete for a "winner takes all" grand prize of $10,000. All finalists were required to included a written description of the project, all design files, and the necessary software to make their Springboard module prototype operate. All the finalists did a great job. This application note is derived from the submission supplied by Steve Haynal of Hillsboro, Oregon. Refer to Figure 1 for a photograph of the Handheld Musical Instrument Tuner.

Appendix A outlines existing Xilinx application notes that are appropriate for understanding this application note. These are available on the Xilinx website at: **http://www.xilinx.com/**.

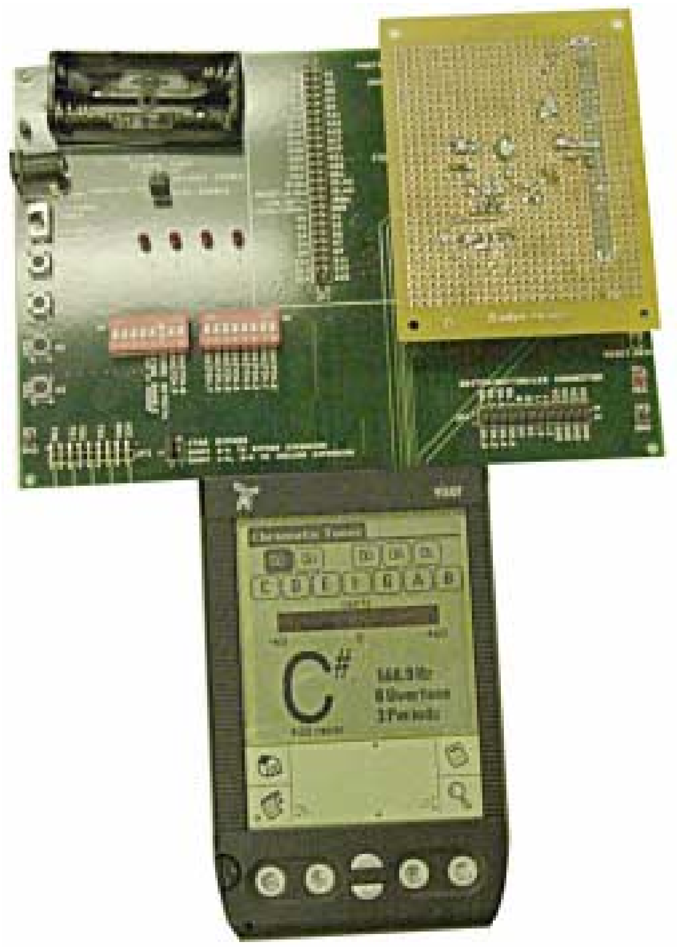

Figure 1: **Photograph of Musical Tuner in operation**

Design Description

This Springboard module transforms a Handspring Visor PDA into a precision musical instrument tuner. A musician tunes an instrument by sounding a note, determining the current pitch from the Visor and making appropriate tuning adjustments to the instrument. This design is a chromatic tuner and is comparable to portable chromatic tuners made by Korg, Yamaha and BOSS, yet it takes advantage of the graphical user interface and computing capabilities provided by the Visor. A chromatic tuner determines the fundamental frequency and chromatic semitone name (A through G#) of a single sounding musical pitch. The Visor's built-in microphone interfaces to low-power audio filtering, amplification and comparator circuitry. A Xilinx CoolRunner device precisely measures time between successive audio signal zero crossings and implements Springboard bus and FLASH memory glue logic. The Visor software analyzes the samples and determines the fundamental musical note present with near real-time responsiveness even when overtones are present. Finally, the pitch is displayed as a raw frequency and by semitone name (12 semitones, A through G# of the equal tempered scale) and cents (100 cents per semitone) flat or sharp from the displayed semitone.

Features and Specifications

Features of the Handheld Instrument Tuner include:

- 1 of 12 semitone buttons reverses video to indicate detected note.
- Semitone buttons double as a pitch pipe. Tapping a button cause the note to sound.
- Cents meter graphically displays how flat (-1 to -60 cents) or sharp (+1 to +60 cents) the detected note is.

- Current detected note (A through G#) appears in large print in lower left corner for distance reading.
- Numerical cents value (-60 to +60) displays below large printed note.
- Lock indicator appears when detecting a valid note.
- Raw frequency of detected note is displayed.
- Highest overtone (0-7) present in detected note is displayed.
- Number of periods in successful detected note sample is displayed.
- Reference frequency for A4 (410.0-470.0 Hz) may be changed for calibration.
- Sample error margin is parameterized.
- Filter error margin, depth, agreement, and rate are parameterized.
- Detection of individual overtones (0-7) may be specified.
- Preference values are saved.
- About, Info, and Help menu options.
- External time reference is powered down when not in use.

The Musical Instrument Specifications are as follows:

Tuning range:	C1 (65.4 Hz) to G#7(6644.9 Hz) with no overtones A0 (27.5 Hz) to G#7 (6644.9 Hz) with overtones greater than 2
Accuracy:	Frequency ± 0.15% Hz (low ambient noise) Cents ± 1 Cent (low ambient noise)
Calibration:	A4 410.0 Hz to 470.0 Hz
Sampling rate:	4.000 MHz
Adjustable sampling error:	± 0.1% to ± 5% Hz
Filter sampling rate:	1 to 20 Hz
Filter depth:	2 to 15 samples
Filter agreement:	1 to 15 samples in consensus
Filter agreement error:	± 0.1% to ± 5.0% Hz
Overtone detection:	0 to 7 Overtones

System Design Overview

The Handheld Instrument Tuner design consists of both hardware and software components. The necessary hardware is comprised of a simple and efficient microphone interface which utilizes the Handspring microphone. The microphone signal is amplified and filtered before being applied to a comparator to detect zero crossings.

Editor's note: Although the finished module works correctly, the output of the comparator would best be followed by a Schmitt trigger buffer as the output of the comparator will not meet the required edge rate as required by the clock inputs on the XPLA3 CPLD.

End Module Applications

The primary application for this device is that of an instrument tuner, or to provide an audible reference frequency to be used when performing instrument tuning.

Operating Instructions

The Handheld Musical Instrument Tuner design can be downloaded from the Xilinx website. The download pack includes the necessary files needed to run the Handheld Musical Instrument Tuner, see **Download Pack**, page 10.

Starting the Application

To start the Handheld Musical Instrument Tuner:

1. Insert Module and turn on Visor (if Visor does not auto detect card).
2. Start Handheld Musical Instrument Tuner program.
3. Sounds note on instrument.
4. Tuner detects, measures, and displays pitch.
5. Adjust instrument tune to desired pitch.

Handspring Display

The tuner display provides a number of measurements for the user. At the top are 12 buttons representing the 12 semitones in a chromatic scale. The appropriate button is displayed in reverse video if a pitch closest to that semitone is detected. For example, the #/*b* button above the C will display reversed if a pitch close to C# is detected. These 12 semitone buttons also double as a pitch pipe. For instance, touching the G button will cause the Visor to sound the note G. The pitch pipe note is always in tune (e.g. changing the flat/sharp cents control does not change the pitch pipe note). The tuner does not detect a pitch pipe note. The pitch pipe does respect the A4 calibration frequency specified in the preferences.

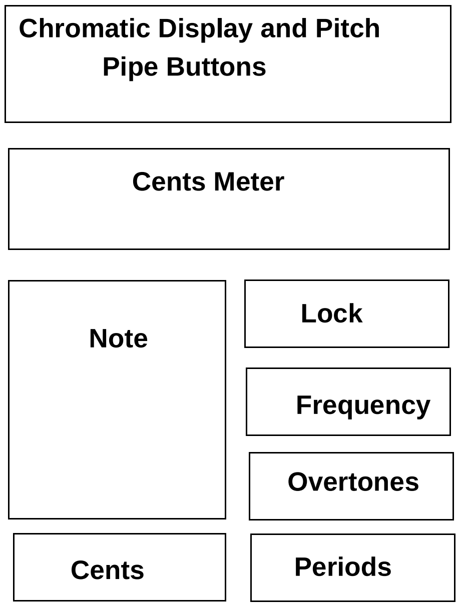

Figure 2: **Handspring Tuner Display**

Below the 12 semitone buttons is the flat/sharp cents meter. This displays how sharp or flat the detected note is when compared to the closest perfectly tuned semitone. For example, if the F semitone button is reverse video and the cents meter registers +20 cents, the detected note is

20 cents sharp of the note F. Although there are 100 cents between semitones, the cents meter ranges from -60 to +60 cents. This allows an overlap of 20 cents between notes in order to keep rapid semitone switching from occurring at -50 or +50 cents. To obtain the highest degree of accuracy, the user should tune an instrument with a steady (pitch and volume) tone in a place that has little or no ambient noise. The detected semitone and numeric flat/sharp cents value are also displayed in the lower left corner. This display is in large print to facilitate distance reading.

In the right bottom corner, the tuner displays a lock indicator, raw frequency, overtone count and period count. The tuner displays "Lock" whenever it detects notes meeting error and filter constraints. When a valid note is detected, the tuner updates the raw frequency value. This is the tuner's frequency measurement in Hertz of the detected note. The overtone count displays the number of overtones detected in the input note. In an orchestra, every instrument may be playing the note A but each instrument sounds different. It is different overtone content that makes each instrument sound unique. (See **Handspring Software Design**, page 8 for additional overtone discussion). For example, a trombone will have a strong second overtone that the tuner must detect and ignore while a slide whistle will have no strong overtones. The final measurement is the number of periods over which the sample tone remained within the error bounds. A higher number implies a more accurate measurement. For instance, if "8" (the maximum) is displayed, then eight consecutive period measurements were within the error bounds for the detected note.

Preferences

The chromatic tuner exposes many configurable parameters to the user. The chromatic tuner menu opens when a user taps either the menu icon or the "Chromatic Tuner" title at the top of the screen. Selecting the "Preferences" menu item opens the "Tuner Preferences" dialogue box. The user may calibrate the tuner to a specific A4 frequency in this dialogue. Also, the user may select which overtones the tuner should look for or ignore. There are configurable sample and filtering parameters in this dialogue as well. These are intended for advanced users and are described in the **Handspring Software Design** section. Most users will use the default settings, which may be recovered by tapping the "Defaults" button. Tapping the "OK" button saves all current selections and leaves the dialogue box. "Cancel" leaves the dialogue box without saving any changes.

Hardware Description

Analog Interface

Refer to the **Handheld Musical Instrument Tuner schematic** which may be downloaded from the Xilinx website. See **Download Pack**, page 10, for more information.

The chromatic tuner uses external hardware for low-power audio filtering, amplification and comparator functions. Note that the two active components consist of a 4.00 MHz clock and a dual, rail to rail, operational amplifier. Two separate discrete networks provide microphone biasing and a virtual ground.

One stage of the operational amplifier pair is used to provide gain, along with high and low pass filtering. This stage is then followed by another amplifier section that functions as a comparator.

At a high level, the intended purpose of this hardware is illustrated in Figure 3:

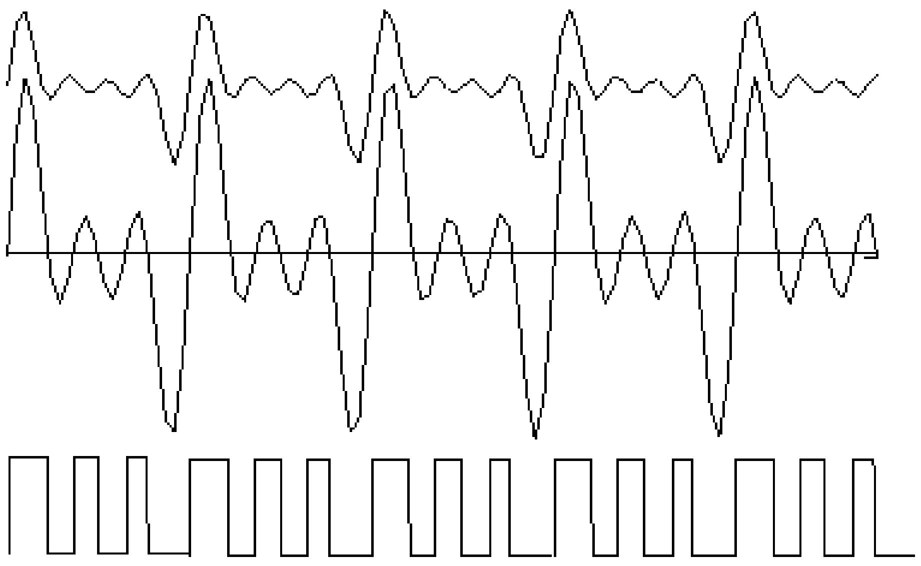

Figure 3: **Waverforms for Handheld Musical Instrument tuner**

The Visor built-in microphone is biased to provide a low-level audio signal, as shown in the top plot. A trombone note is shown in this plot. A low-power op amp substantially amplifies the signal and provides high frequency roll off. Finally, an op amp comparator converts the amplified signal into a square wave, where each rising edge of the square wave corresponds to a rising edge zero crossing of the amplified signal.

Signal Conditioning

The first schematic block shows how the Visor microphone is biased with a filtered power supply. As is, the Springboard module prototyping board already biases the microphone. A removal of C18, C19 and R21 from the prototyping board and change to this biasing scheme, satisfies the recommendations of the Visor Microphone application note (AN-08) from Handspring.

The virtual ground schematic block shows how a virtual ground voltage is generated for single supply operation of the op amp circuits. It is slightly less than half of V_{CC}.

The X1 time reference schematic block is a crystal oscillator. The A/D converter on the Springboard module may be used as a clock but is not accurate enough due to process and temperature variations (varies 50-100 kHz for temperature changes, varies 100-150 kHz between parts). X1 is used instead as an accurate 4.000 MHz time base. If the tuner is not being used, X1 is disabled via IO14 to conserve power.

The U1 OPA2340PA schematic block describes how the power is connected and filtered for the op amp. This Burr Brown op amp is a double version of the OPA340 that is recommended for the Springboard module prototyping board.

The gain block amplifies the input signal and provides high frequency roll off. The DC gain is roughly R8/R1=1000. C5 decreases the gain as the signal frequency increases. C4 is the decoupling capacitor from the microphone interface

The comparator block has an additional low pass filter at its input and compares this signal with virtual ground, VG. A square wave, SIGCLK, identifies zero crossings and is connected to the CoolRunner CPLD via IO15.

CPLD Design

The Xilinx CoolRunner XPLA3 CPLD software serves two purposes. First, it provides the interface between the FLASH memory and the Springboard bus. Second, the CPLD times and collects zero-crossing samples of the input signal, SIGCLK. Figure 4 below illustrates typical samples that may be collected. Registers ’R0’ through ’R6’ plus ’T’ (Timer) collect period times for eight consecutive periods. Refer to Figure 4 for an example of the sample periods. These are 16-bit values with count resolution of 250 ns (4.000 MHz clock). The Visor software initiates a sampling sequence, waits, and then collects and processes these eight samples to determine the fundamental frequency.

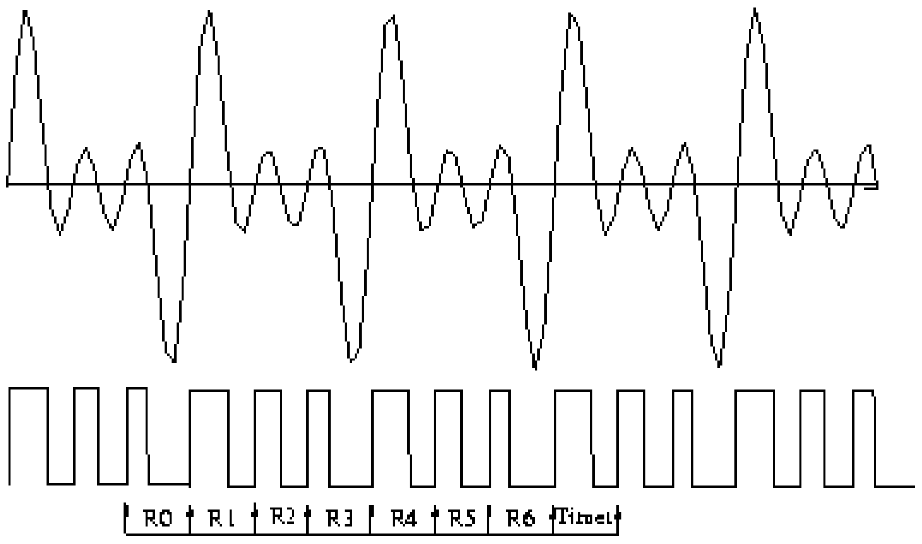

Figure 4: **Zero Cross Sample Periods**

The source code and implementation results for the CPLD design are included in the available download pack (see **Download Pack**, page 10). The CPLD implementation is written in Verilog with commented source files included with this submission.

Figure 5 is a block diagram that shows the basic structure for period sampling. To initiate a sampling sequence, the Visor program must write 0x0001 to 0x29000000. The least significant bit enables the 4.000 MHz clock. The act of writing to this address triggers a finite state machine reset and thus a new sampling sequence. The finite state machine holds the 16-bit timer in reset until the first rising edge of SIGCLK. Once the first rising edge is detected, the timer is enabled and counts until the second rising edge. At this point, the count value is latched into R0 and the timer is cleared and the process repeats for R1 through R6. After R6 is latched, the timer is cleared and counts until the final rising edge of the sampling sequence. At that time, the timer is disabled to preserve the final period count. All eight count values are accessible through the Springboard bus as memory mapped registers. The Visor program polls address 0x29000000 for two flags. The least significant bit, bit 0, is cleared once the finite state machine has completed a sampling sequence. It is set while the finite state machine is busy. The next bit, bit 1, is set if an overflow occurs during any period count.

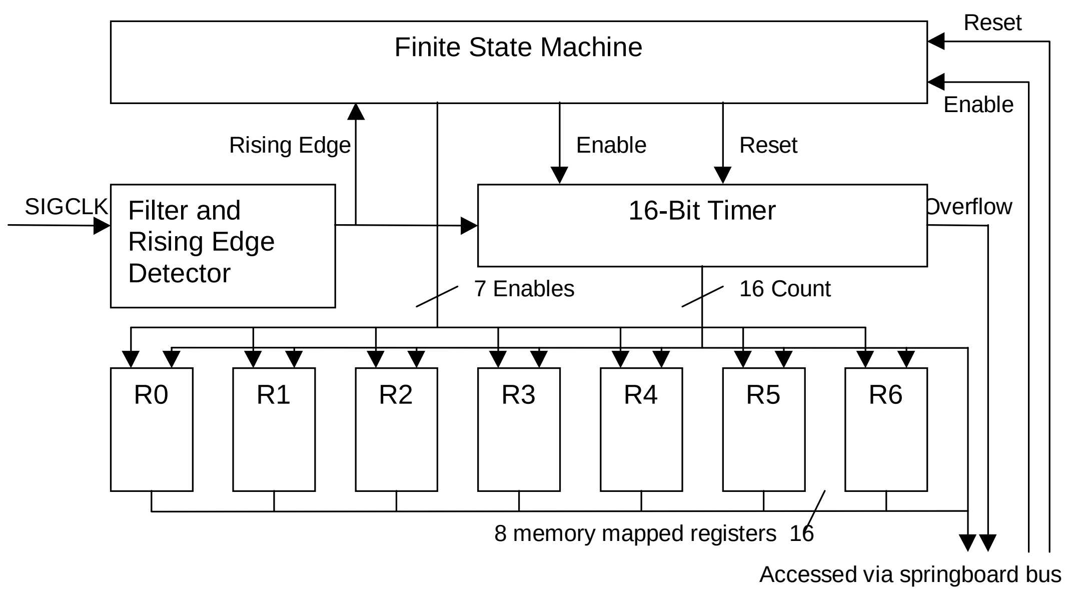

Figure 5: **Software Block Diagram**

The filter and rising edge detector requires that two consecutive SIGCLK values agree before the internal copy of SIGCLK is updated. This eliminates noise and correctly synchronizes this asynchronous signal with the internal synchronous design. Finally, this module asserts rising edge for any clock cycle following a rising edge on the internal copy of SIGCLK.

Handspring Software Design

The tuner application was created with GNU Tools and PilRC running under Linux. Commented C code is included with this submission.

Tuner Algorithm

Before describing the software, it is helpful to understand the implemented tuning algorithm. It is not sufficient to simply count time between rising edge zero crossings and consider this the period of the fundamental frequency. The example of a trombone note, shown above in Figure 3, demonstrates that this does not capture a true period. On the other hand, it is too costly to compute a full-fledged DFT (Discrete Fourier Transform) with high accuracy for a single musical note. The content of the entire frequency spectrum is not needed, but rather just the precise fundamental frequency of a sounding musical note. Such a note always has the form,

$$f(F) = A1\sin(F) + A2\sin(2F) + A3\sin(3F) + \ldots + An\sin(nF)$$

where F is the fundamental frequency, and Ai is the amplitude of each overtone, or frequency component that is a whole multiple of F. Because all tones present are whole multiples of the fundamental frequency, the fundamental frequency zero crossings will not be altered by any overtones, whereas zero crossings of overtones will be shifted in time due to the presence of the fundamental frequency or other lower overtones. Thus, if we can find the least period, spanning perhaps several zero crossings, that consistently repeats, we have found the fundamental frequency. The preceding discussion is illustrated in the example below. We see that once we span two zero crossings, we obtain a regular repeating period that is the fundamental period, even though two overtones are present.

(Http://www.izzy.net/~jc/PSTInfo/Temper.html summarizes music theory, tuning and overtones. Http://sparky.ls.luc.edu/sharc/ contains a web accessible database of musical instrument spectral signatures.)

The tuner algorithm looks for regular repeating periods as described. First, it tries the no-overtone case (a period that spans no zero crossing or the time between two consecutive zero crossings). It compares this period with the next period and calculates to determine whether they agree (within the sample error margin). The sample error margin is the allowed percent difference between the two samples and is configurable in the preferences dialogue. If at least three agree, this is considered the fundamental frequency and the algorithm is done. If not, the algorithm looks at periods spanning from one to seven zero crossings (one to seven overtones) and sees if they agree with a period of the same span shifted by one zero crossing. If so, a match is found and this sample is considered the fundamental frequency.

The software also implements a low frequency filter. For the default filter rate, 10 per second, the software initiates and evaluates a hardware sample set 10 times per second. If a valid note detect occurs, it stores the note in a time-indexed circular queue. The default depth of this queue is 5. Only if n samples agree ("n" is filter agreement parameter) to within the filter error margin (frequency difference by percent) is a sample considered valid for display. At that point, all display parameters are updated. All filter parameters are configurable in the preferences dialogue (Filter Error Margin 0.1% to 5.0%, Filter Depth 2 to 15, Filter Agreement 1-15, Filter Rate 1-20). This parameterization allows an advanced user to customize the tuning algorithm.

Software Flow

A flow chart for the heart of the tuning algorithm is shown in Figure 6. First, the software checks for a completed hardware sampling sequence. If this has completed, it collects the eight zero crossing times from the CPLD. It then compares consecutive periods spanning from zero to seven zero crossings to see if any fall within the sample error bounds as described previously in the algorithmic section. If a detect occurs, it checks if the detected note passes the software filter as described earlier. If so, the screen variables are updated, the note is entered into the filter queue and another hardware sampling sequence is initiated. This main loop is called filter_rate times per second by the main application event loop. There is also standard code for the preference dialogue, screen updates and other normal events.

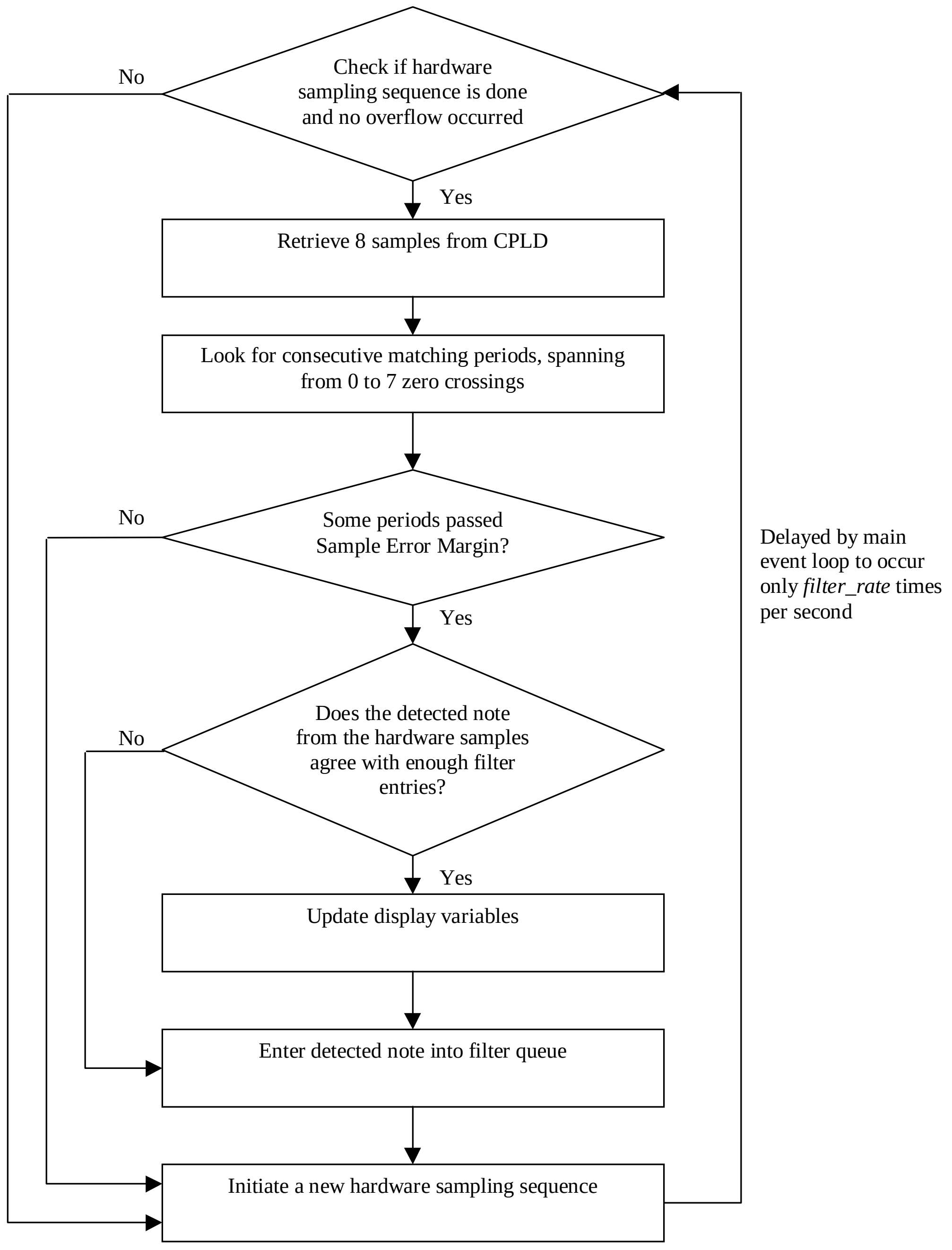
Check if hardware sampling sequence is done and no overflow occurred
No
Yes
Retrieve 8 samples from CPLD
Look for consecutive matching periods, spanning from 0 to 7 zero crossings
Some periods passed Sample Error Margin?
No
Yes
Does the detected note from the hardware samples agree with enough filter entries?
No
Yes
Update display variables
Enter detected note into filter queue
Initiate a new hardware sampling sequence
Delayed by main event loop to occur only filter_rate times per second

Figure 6: **Software Flow for Handheld Musical Instrument Tuner**

Download Pack

THIRD PARTIES MAY HAVE PATENTS ON THE HANDHELD MUSICAL INSTRUMENT TUNER. BY PROVIDING THIS HDL CODE AS ONE POSSIBLE IMPLEMENTATION OF THIS DESIGN, XILINX IS MAKING NO REPRESENTATION THAT THE PROVIDED IMPLEMENTATION OF THIS DESIGN IS FREE FROM ANY CLAIMS OF INFRINGEMENT BY ANY THIRD PARTY. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, AND XILINX SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE, THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OR REPRESENTATION THAT THE IMPLEMENTATION IS FREE FROM CLAIMS OF ANY THIRD PARTY. FURTHERMORE, XILINX IS PROVIDING THIS REFERENCE DESIGNS "AS IS" AS A COURTESY TO YOU.

XAPP366 - http://www.xilinx.com/products/xaw/coolvhdlq.htm

Conclusion

This design submission for the Cool Module Design Contest is a perfect application of a CoolRunner CPLD in a low power Springboard module. This design illustrates how designers can combine the low power consumption of CoolRunner CPLDs with the programming flexibility of the Handspring in a minimal parts count design solution.

The chromatic tuner performs as expected, but has potential for improved functionality. Most chromatic tuners have an input jack for electric instruments. Such a direct connection would bypasse noise introduced by an audio link and would increase the accuracy. Other higher quality accoustic input devices (microphones, etc.) could also be accommodated. Second, adding an AGC circuit and improving the amplification and filter design at the front-end would reduce the impact of ambient noise. Third, the amplification and filtering do not shut down when the tuner is inactive. These could be shut down to improve power management. Finally, algorithmic changes that look at both rising edge zero crossings and falling edge zero crossings would help to better distinguish between overtones. Although it is possible to use the SRAM to save a large number of samples, a goal of this design was to use as few components as possible to reduce potential production costs.

Appendix A

Appendix A lists appropriate Xilinx CoolRunner CPLD application notes. These application notes can be found by searching the Xilinx website and keying on the specific XAPP#. Many include appropriate driver software along with high level design code. All have been constructed and work.

PDA Springboard Design

XAPP147: Low Power Handspring Springboard Module Design with CoolRunner CPLDs

XAPP359: Understanding the Insight Springboard Development Kit

XAPP357: CoolRunner Visor Springboard LED Test

XAPP355: Serial ADC Interface Using a CoolRunner CPLD

XAPP146: Designing an Eight Channel Digital Volt Meter with the Insight Springboard Kit

XAPP149: Designing an Oscilloscope for the Insight Springboard Development Kit

XAPP349: Pocket Oscilloscope

References

Note: Xilinx does not maintain the below links, and as such they may be invalide subsequent to the publishing of this document.

Handspring website: http://www.handspring.com/

Additional music theory: http://www.izzy.net/~jc/PSTInfo/Temper.html

Musical instrument signatures: http://sparky.ls.luc.edu/sharc/

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/07/01	1.0	Initial Xilinx release.