Instantaneous frequency estimation and data conversion for IEPE-compatible transducers

Łukasz Huchel

12.11.2020

Contents

1 Introduction

An IEPE accelereometer is a *de-facto* standard for industrial vibration sensing. Several accelerometer manufacturers use a different proprietary names, however, the name "Integrated Electronics Piezo-Electric (IEPE)" is commonly accepted to characterize a family of sensors and the corresponding analog interface. A typical IEPE interface is illustrated in Fig. 1. As seen in Fig. 1, a current source provides a dc current component and establishes a dc bias Vdc across the impedance of the sensor. The piezoelectric material, being an active transducer, in response to dynamic excitation alters the output impedance of the sensor. The AC component of the output voltage conveys information about the mechanical excitation. The IEPE sensors provide high bandwidth, high fidelity vibration data over a two-wire interface, which is simultaneously

providing power to the sensor. Despite the simplicity and robustness of the interface, there are relatively few IEPE standard solutions for other types of sensors.

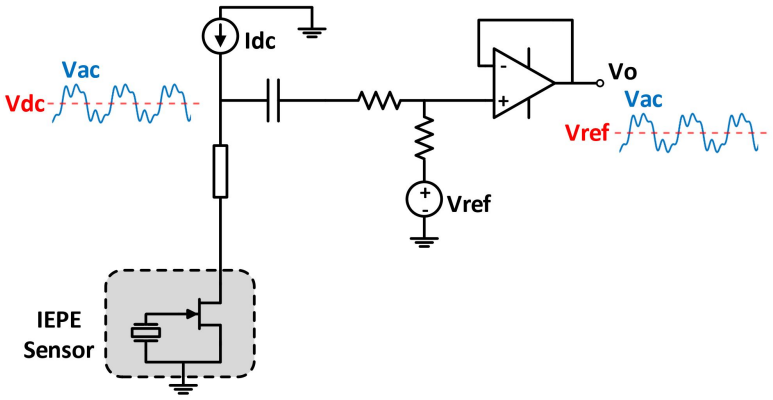

Figure 1: The IEPE sensor and interface.

To address this need, an extension of the IEPE interface is proposed here. This extension enables new measurements to be acquired over the IEPE interface. The new sensor solution utilizes frequency modulation for signal transmission; thus, it presents high noise immunity. Fig. 2 presents a view of the proposed strategy. Due to the modulation technique it is possible to transmit a DC measurements, despite the inherent AC-coupling of the IEPE interface. Static measurements like temperature and pressure are possible in this configuration.

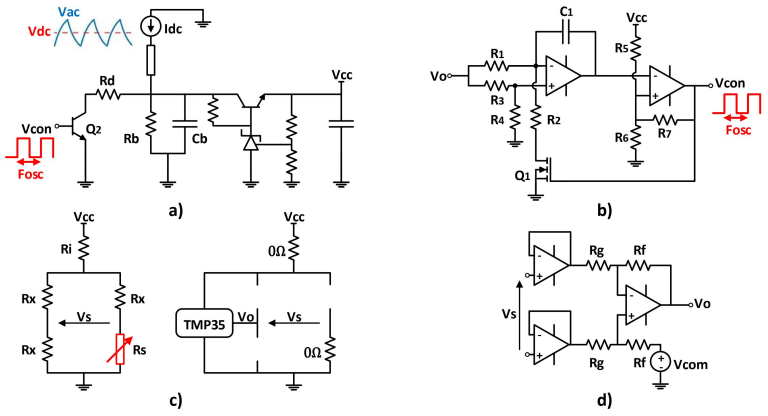

Figure 2: The view of proposed sensing solution.

So far, the IEPE transducer signal was acquired in parallel to vibration

measurements and frequency content was converted off-line to the measurand domain using Hilbert transform. This approach provides the instantaneous phase of the signal which can be mapped for instance to temperature (in case of sensing element reacting to temperature changes). This report demonstrates a hardware support for this new sensing approach which enables real-time signal conversion. It uses a Field Programmable Gate Arrays (FPGA) to perform real-time computations and provide a measured quantity to the user. As an example application, the temperature measurements are used in this report.

Further sections present details of signal processing and implementation of the processing pipeline. The final result of conversion is displayed over the VGA interface to the user. The results presented in this report were obtained with Nexys 4 DDR evaluation platform that is equipped with the XC7A100T-1CSG324C FPGA from Xilinx.

2 Processing Pipeline

The data processing requires several stages. The main parts of the pipeline are summarized below. The pipeline is realized in two clokc-domains. The 100 MHz domain for signal processing and computations and 65 MHz for VGA related logic (with 1024x768 resolution).

- Sampling of the frequency modulated signal
- Digital band-pass filtering
- Calculation of the magnitude frequency spectrum
- Detection of the fundamental frequency
- Frequency estimate to voltage conversion
- Voltage estimate to resistance of sensing element conversion
- Resistance to measurand conversion
- Display

The schematic view of the pipeline is given in Fig. 3. Synchronization between modules is performed with Advanced eXtensible Interface (AXI), however, custom modules do not necessarily strictly follow the AXI standard. Further sections provide details of the particular blocks.

2.1 Sampling and Filtering

The sampling and quantization of the input signal is performed with the xadc module. Physically sampling is performed at 961.54 kHz, decimation provides the desired signal rate. The range of frequency modulation is between 8-19 kHz. The sampling frequency was designed as 48 kHz. In order to minimize

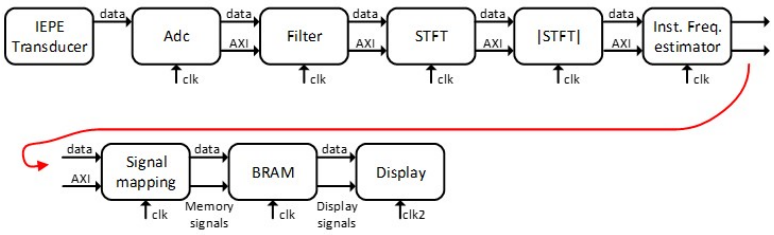

Figure 3: A high-level block diagram of data pipeline.

the interference of undesirable content the sample signal is filtered with a band-pass FIR filter. The magnitude frequency response of the 128 coefficient FIR filter is presented in Fig. 4. The filter coefficients were designed with the Parks–McClellan algorithm.

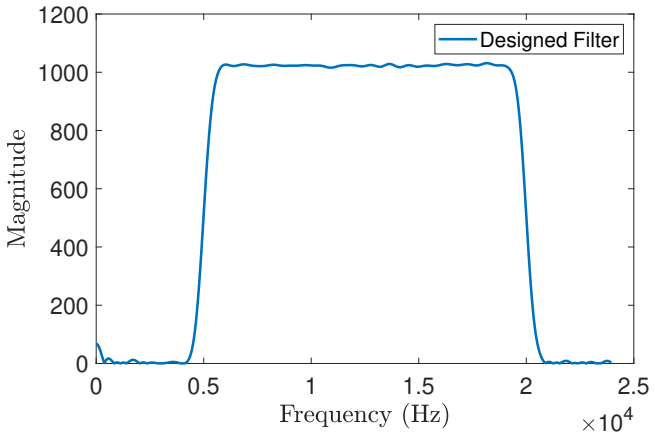

Figure 4: The magnitude frequency response of the band-pass filter.

2.2 Time-frequency Conversion

Filtered signal is an input to the Xilinx IP subsystem performing FFT. The architecture of the FFT IP core is designed as fixed point pipelined streaming I/O. It uses the highest number of resources but provides highest throughput for continuous processing which is essential in this application. The Fast Fourier Transform v9.1 was used in this project.

The continuous data processing in blocks is essentially a short-time Discrete Fourier Transform (STFT). There is no overlap and adjacent blocks are processed sequentially. Data blocks are extracted with basic rectangular window.

Frequency resolution is a key factor in this application, while magnitude of the content is of no interest. Thus, rectangular window with smaller mainlobe width than many popular windowing functions is a good candidate. The size of window was designed as 32768. Consequently, with a sampling frequency of 48 kHz the frequency resolution is $\Delta f = \frac{F_s}{2^{15}} = 1.46\,Hz$. A sample view of the FFT module output is presented in Fig. 5, output for 1024 long FFT. It should be noted with the desired FFT size output data in not available in one burst. The AXI bus with indication of last output element is necessary to synchronize the following modules.

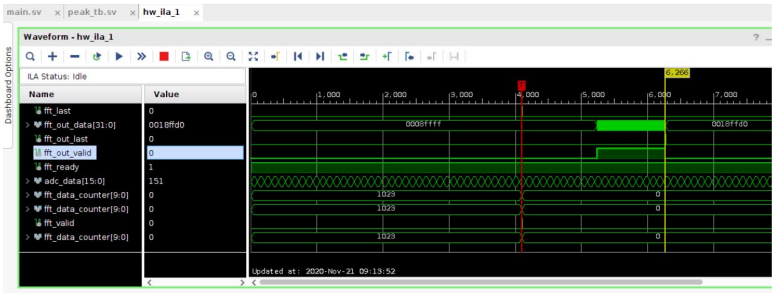

Figure 5: The sample output of a 1024 long FFT.

The following modules in the pipeline extract the magnitude spectrum. The square root of the final output is a vector of magnitude values which is used as an input to a peak extraction module. Peak is localized using properties of real input signals and only half of the spectrum is investigated. A simple approach iterating over all discrete frequencies between DC and $\frac{F_s}{2}$ is applied. All of the above modules are fully pipelined to maximize the throughput. Modules relevant to this section are reported in Table 1 with a corresponding latency.

Table 1: Module Latency

Module	**Latency [clk]**
xfft_0	65060(last in to last out)
square_and_sum	2
cordic_0	17
peak_search	1

The latency reported for the FFT module corresponds to a number of clock cycles between last input sample and last output sample. At 100 MHz it corresponds to approximately 650 ms.

2.3 Frequency to Resistance Conversion

The output of the peak_search module is a 16 unsigned bit integer that needs to be converted into frequency value in *Hz*. The frequency estimate $\hat{F}_{est} =$

$\frac{PeakIndex}{N_{FFT}} F_s$ isn't in general an integer; thus, the following operations require fixed point arithmetic. In the expression for $\hat{F}_{est}$ the FFT size N_{FFT} is a power of two number and bit shifting can replace the division operation. The fixed point data format is designed as Q16.16 format.

The frequency value is mapped to the amplifier input voltage, see Fig. 2b. Mapping is done according to (1).

$$\hat{F}_{est} = \frac{V_{ref} - G \times V_{amp}}{4 \times V_{pp} R_1 C_1}, \tag{1}$$

where V_{ref}, G, V_{pp} and R_1C_1 are constants defined by circuit parameters and scaled to Q16.16 format. The V_{amp} signal is the unknown input signal. Inversion of this equation results in a multiplication by a constant, subtraction and division operations.

Table 2: Module Latency

Module	**Latency [clk]**
center_freq_est	2
center_freq_2_voltage	3
sensing_resistance	37

The voltage estimate allows to calculate the resistance estimate, i.e., sensing resistance, see Fig. 2c. Resistance is estimated with the following equation:

$$\hat{R}_{est} = \frac{V_{cc} R_x + V_{amp}(R_i + 2R_x)}{-V_{amp}(\frac{R_i}{R_x} + 2) + V_{cc}}, \tag{2}$$

where R_i, V_{cc}, R_x are constants defined by circuit parameters and scaled to Q16.16 format. The above calculations are performed in one sensing_resistance module. It performs necessary fixed point multiplications, scaling and division. Latency of the module equals to 37 clock cycles as reported in Table 2. Latency is mostly driven by the divider generator from Xilinx library. The divisor and dividend arguments are 32 bit numbers, output of the divider is a 64 bit number scaled to the original format. The divider is implemented with a non-blocking Radix-2 architecture with reminder output.

2.4 Resistance to Temperature Conversion

The characteristic of the sensing element is strongly nonlinear [1]. Equation (3) is used to calculate temperature estimate based on the value of $\hat{R}_{eq}$.

$$\hat{T}_{est} = \frac{T_0 \beta}{T_0 ln(\frac{\hat{R}_{est}}{R_0}) + \beta}, \tag{3}$$

where T_0, R_0 and β are constants defined by thermistor characteristic. Output of (3) is an unsigned number in Kelvin scale.

It can be seen that evaluation of natural logarithm is required to calculate temperature value. Unfortunately, Xilinx IP library provides only solution for natural logarithm with floating point numbers as a part of the Floating-Point Operator core. Thus, in order to complete the calculations a custom implementation of natural logarithm for fixed point numbers is necessary. Due to the fixed point nature, a fast binary logarithm algorithm is used [2, 3].

The main idea behind the algorithm is based on the Al Kashi's method. It uses sequential squaring and division, which in radix-2 representation becomes division by two and can be efficiently implemented with bit shifting. In order to obtain natural logarithm of x, the following representation is used: $x = 2^y$. Because of radix-2 representation $x = 2^y$ can be written as:

$$x = 2^{2^{-1}(y_1 + 2^{-1}y_2 + \ldots)} \tag{4}$$

Squaring both sides of (4) provides:

$$x^2 = 2^{y_1} 2^{2^{-1}(y_2 + 2^{-1}y_3 + \ldots)} \tag{5}$$

It can be recognized that $2^{2^{-1}(y_2 + 2^{-1}y_3 + \ldots)}$ is a number greater or equal to one and less than two. Thus, (5) is greater than two only when y_1 is one, i.e., bit at position one of number x has value one.

$$y_1 = \begin{cases} 1, & \text{if } x^2 \geq 2 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

If y_1 is one, (5) is divided by 2; if not, there is no change to (5). In both cases, the reminder $2^{2^{-1}(y_2 + 2^{-1}y_3 + \ldots)}$ has the same form as (4), i.e., it can be squared and analyzed in the same fashion. This is the main idea behind fast binary logarithm method. It should be noted that algorithm relies on x values between one and two. Thus, the input number has to be scaled accordingly before first iteration. If we define N as the number of initial divisions by two and M as the number of initial multiplications by two, the final mantissa $y^{'}$ equals to:

$$y^{'} = y + N - M \tag{7}$$

The described algorithm allows to find a base 2 logarithm of x, however, the property in (8) can be used to convert it to a desired base; conversion can be efficiently implemented by multiplying with a constant $\frac{1}{log_2(b)}$.

$$log_b(x) = \frac{log_2(x)}{log_2(b)} \tag{8}$$

With the fixed point implementation of natural logarithm final evaluation of (3) is possible. Table 3 presents latency of the modules involved in conversion to temperature. The division in (3) is included in the convert_temperature module and contributes to its latency. This module uses the sane architecture of the divider generator as sensing_resistance module. High latency of base_2_log

module is due to its iterative nature. Look-up table could be an alternative for logarithm evaluation, however, with 32 bit variables this would be considerable memory effort. Additionally, this algorithm can be scaled for operations with words of different size.

Table 3: Module Latency

Module	**Latency [clk]**
base_2_log	170
natural_logarithm	3
convert_temperature	41

2.5 Temperature Conversion

Current unsigned result is converted to Celcius scale by subtracting 273.4. The result is 64 bit signed number in (°C). The data width is a result of divider block in convert_temperature module which generates 64 bit result. Since this is last step in the processing pipeline, the result isn't scaled back to 32 bit format to avoid potential precision loss.

$$\hat{T}_{est}^{C} = \hat{T}_{est} - 273.4 \tag{9}$$

3 Memory

A dual port BRAM memory is used to buffer the temperature data. Width of the memory word is designed as 8 bytes (final temperature result). Depth of the memory is designed as 128. This corresponds to temperature data for equivalent time window of 87 seconds, $\frac{N_{FFT}}{F_s}128 \approx 87\ s$.

Additional role of the dual port memory is to enable safe clock domain crossing. The signal processing pipeline operates at 100 MHz while display pipeline operates at 65 MHz

4 Display

The temperature measurement is displayed on the monitor over the VGA interface. The basic functionality displays temperature measurements corresponding to 64 measurement windows, i.e., approximately 43 seconds. Horizontal dimension of the display is discretized into 64 segments. During each monitor refresh cycle 64 segments are read from the BRAM starting at the most current measurement.

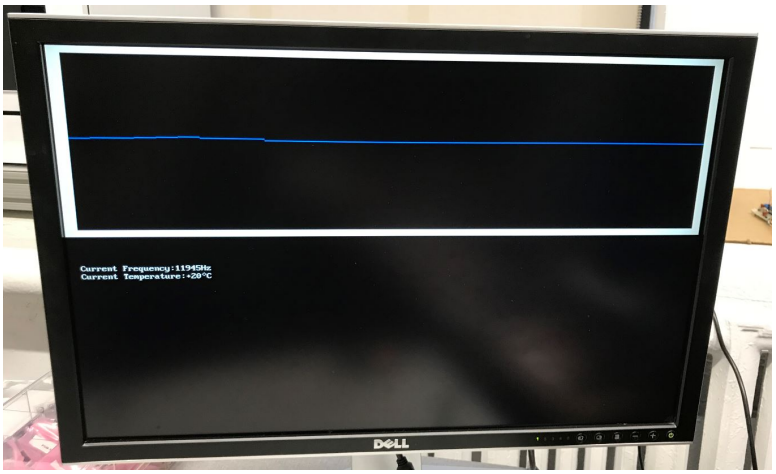

Figure 6: The graphical interface.

4.1 VGA Text Controller

Besides waveform display, graphical interface provides numerical value of current frequency and temperature estimates $\hat{F}_{est}$ and $\hat{T}^C_{est}$ [4]. To enable this feature, a coe file with ASCII symbols is stored in ROM memory. Symbols stored in the ROM are 8x16 pixels; thus, the ROM memory size is $16 \times 8 \times 128 = 16384$. Standard ASCII set was modified to include a degree symbol.

Additional RAM memory is used to store ASCII encoded text representation. Value of the encoded text is used as an input address to the ROM memory storing bit values for particular pixel. Every refresh cycle the new value of $\hat{F}_{est}$ and $\hat{T}^C_{est}$ is written to the RAM. This provides a way of updating the displayed text.

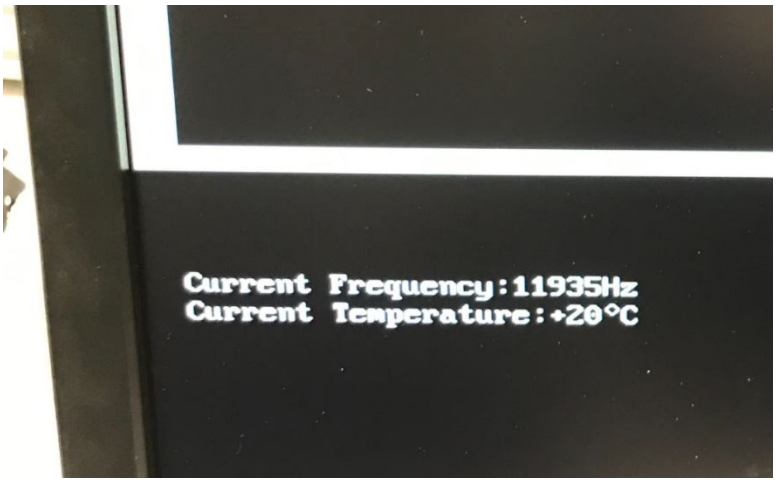

Figure 7: The graphical interface with text.

The latency of the memory read cycle equals to two clock cycles. It is critical to synchronize the output of RAM meory with the input to the ROM memory. Otherwise at a given screen coordinate pipeline will read from the wrong address in the ROM memory. Without synchronization the following text overlap can occur:

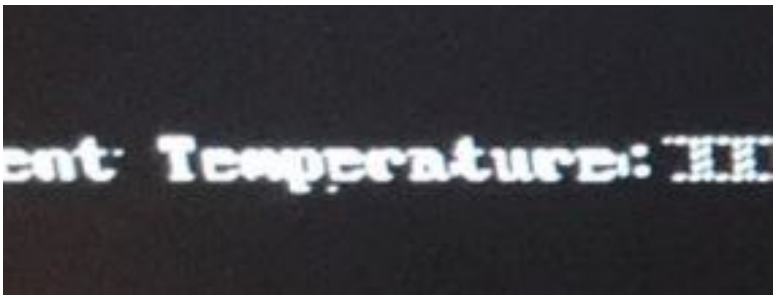

Figure 8: The lack of synchronization between RAM and ROM memories.

The block diagram view of the text controller is given in Fig. 9.

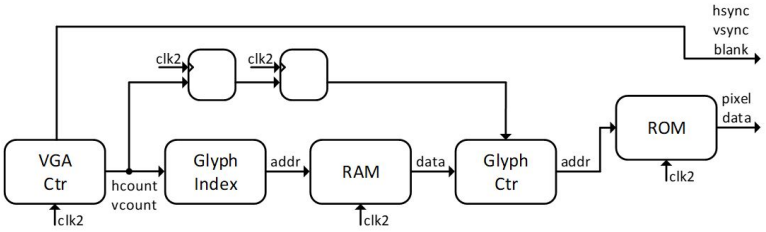

Figure 9: The block diagram of the VGA text controller.

5 Measurement Resolution and Challenges

The graphical text controller was implemented to display tenths and hundredths places. However, due to the limited frequency resolution precision of the final temperature value is limited. The sensitivity is determined by (1)-(3).

The signal processing pipeline presented several challenges. The major challenges were:

- Fixed point implementation of natural logarithm without look-up table
- Implementation of the mapping between frequency and temperature using fixed point arithmetic

The major challenges in the graphical part were:

- Implementation of the VGA text controller
- Synchronization between ROM and RAM memories in the text controller

6 Future Work

Future work includes extending the resolution of the measurement result by improving frequency resolution. There are two potential approaches:

- Bigger size FFT in order to improve frequency resolution
- Digital PLL for frequency extraction

Rest of the modules in the pipeline do not require modifications when resolution is improved.

References

[1] "NTC thermistors for temperature measurement, B57164K". In: *EPCOS datasheet* (2009).

[2] Clay S. Turner. “A Fast Binary Logarithm Algorithm”. In: *IEEE SIGNAL PROCESSING MAGAZINE, dsp TIPSTRICKS* (2010).

[3] *Binary Logarithm SystemVerilog implementation of the Hardware Algorithm*. `https://sistenix.com/logarithm.html`.

[4] *Implementing Text mode for a VGA controller in Verilog*. `https://ktln2.org/2018/06/17/vga-text-mode-verilog/`.