HARDWARE ABILITIES OF LINEAR DECIMATION PROCEDURE IN PRACTICAL APPLICATIONS

Piotr Krzyworzeka, Witold Cioch, Ernest Jamro

University of Science and Technology (AGH-UST)
al. Mickiewicza 30, 30-059 Kraków, Poland
tel.: +48 12 6173622, fax.: +48 12 6332314
e-mail: cioch@agh.edu.pl

Abstract

Procedure of Linear Decimation (PLD) has many practical implementations. The hardware implementation in FPGA (Field Programmable Gate Arrays) of the PLD significantly shortens computation time and allows increasing signal sampling frequency. As a result real-time signal analysis can be obtained which allows for e.g. rotating machine diagnostic during run-up phase. Examples of practical results for the PLD and novel Short-Time PLD are presented.

Keywords*: diagnostics, non-stationary signal, synchronism, decimation,*

1. Introduction

Diagnosing rotary-machine states in variable real-time conditions is vital for their operation safety. Early detection of conditions of incorrect operational processes, as well as of arising damage development can often prevent from failures or serious accidents. The Procedure of Linear Decimation (PLD) [4, 5, 8, 9] has been successfully tested in diagnostic systems for non-stationary cyclical machines. Appling the PLD offers the possibility to increase the energy contributions of time- and frequency- symptomatic components of the signal.

Early detection of failure is determined not solely by a proper detection procedure but also by electronic implementation of diagnostical device. Since the latter may significantly lengthen the calculation time, the failure-detection time may not be achieved within acceptable limits. Consequently, FPGAs (Filed Programmable Gate Arrays) implementation of the PLD is adapted. Hardware (FPGA) implementation of the PLD compared to microprocessors or Digital Signal Processors (DSP) can significantly reduce the calculations time. Furthermore, FPGA implementation of the PLD allows increasing signal sampling frequency, which results in more accurate spectra selectivity.

2. Procedure of Linear Decimation (PLD)

Procedure of Linear Decimation involves the dynamic signal resampling in accordance with rotation speed variations [2]. It assumes linear approximation of cycle variations curbed by values representing its beginning cycle time Θ_p and its end cycle time Θ_k. In other words, it involves the deletion of sample-cluster variations proportional to the cycle and maintaining the constant sample-per-cycle number. The resampling process is shown in fig.1

The key element is to define decimation coefficient Dc_K. This coefficient characterizes the increment of signal resampling [1]. It changes in accordance with the increase or decrease of the rotation speed, i.e. with the cycle variations. By adapting the final decimation coefficient we can determine its variations in accordance with linear signal trend. Selecting final decimation

coefficient enables to reduce the signal to the stationary form in the observation window for the last cycle, which is very convenient in real-time analysis.

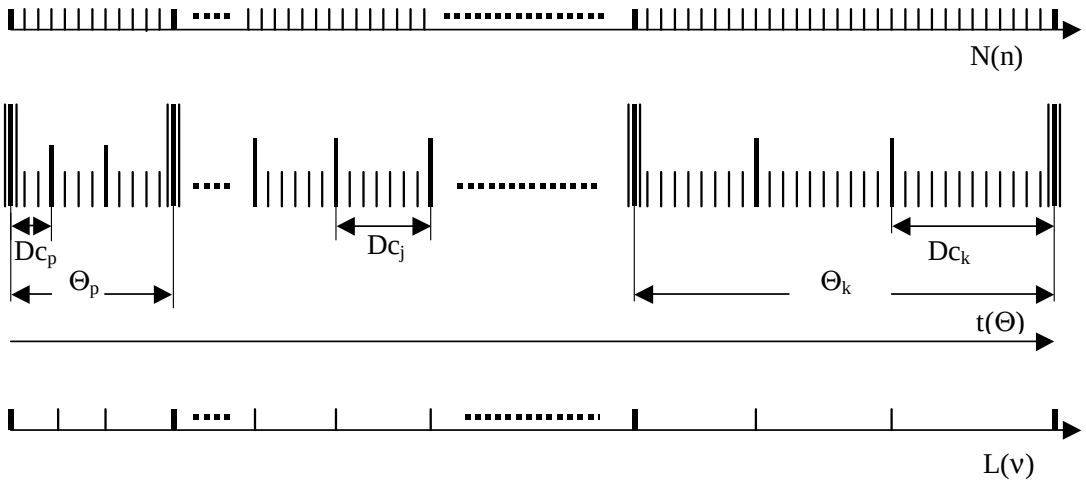

Fig. 1. Procedure of Linear Decimation: Θ - cycle time,(Θ) - observation time, N(n) - primary sample-cluster in the observation window , L(ν) - secondary sample-cluster in the observation window (after LDP), Dc_p - initial decimation coefficient, Dc_k - final decimation coefficient

By selecting a final-decimation-coefficient D_{Ck}, and assuming linear approximation of cycle variations, the decimation coefficient D_{Cn} for *n*-th sample can be obtained by the fallowing formula:

$$Dc_n = Dc_k \frac{N_{\Theta p}}{N_{\Theta k}} + \frac{n}{n_{Dk}}(Dc_k - Dc_p). \tag{1}$$

By applying the above formula, the secondary vector of the signal representing the constant number of samples-per-cycle is given by the formula:

$$w(\nu_k) = u(n_{Dk} + Dc_n) = u(n) \tag{2}$$

where:

u(n) – primary vector,
w(ν)– secondary vector (after resampling).

Fig.2 contains non-stationary signal representing the run-up phase for gear transmission. Cycle change and its linear approximation is shown in Fig. 3. Unfortunately for the given example, the linear approximation error equals 5.1%. Fig. 4 presents signal after applying the PLD. It features the significant improvement of spectrum quality in rotation frequency band. Unfortunately, distinct stripe selectivity in the frequency band of gear meshing and its harmonics was not achieved. It brings the conclusion that approximation by a linear function in the observation window is suitable only for signals with linear cycle-change trend. This made the authors search for new solutions extending the PLD abilities. For this purpose a Programmable Unit for Diagnostic (PUD) was constructed [6]. The PUD is dedicated to non-stationary signal analysis by means of the signal resampling method with a variable increment corresponding with reference cycle changes.

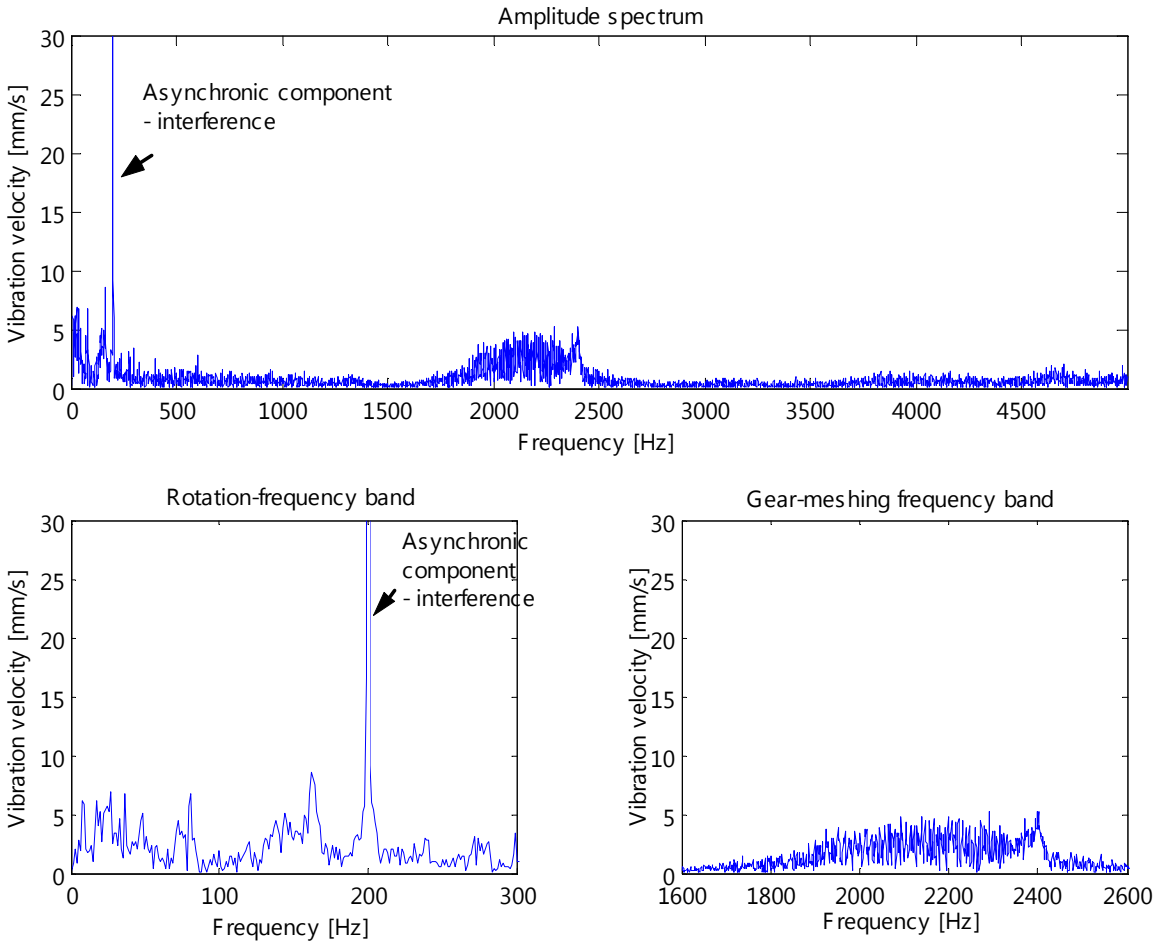

Fig. 2. Amplitude spectrum of vibration velocity of gear-meshing. Average rotational speed increase 0.38 % per cycle

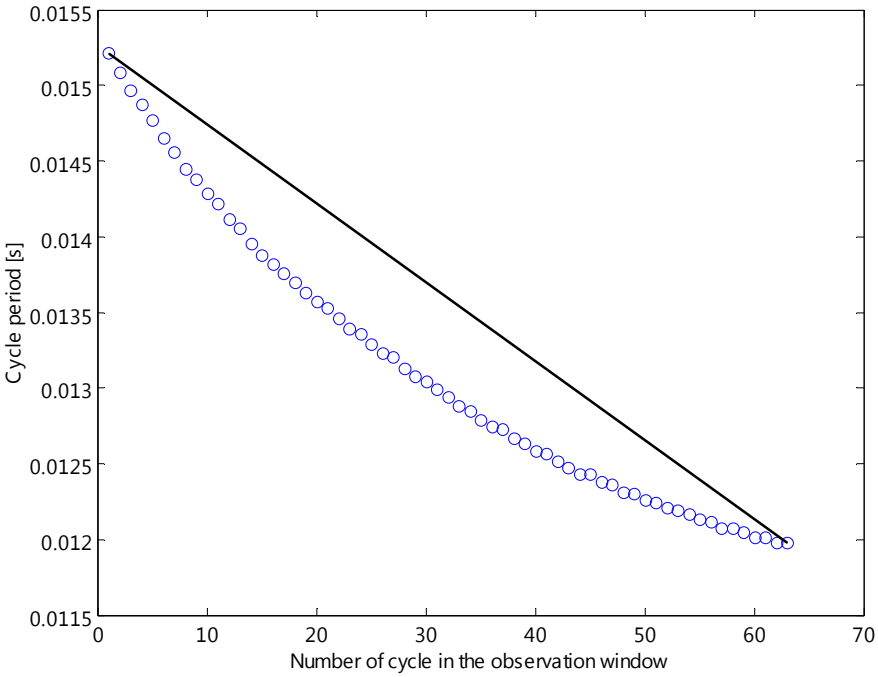

Fig. 3. Cycle length changes within the observation window and their approximation by linear characteristic

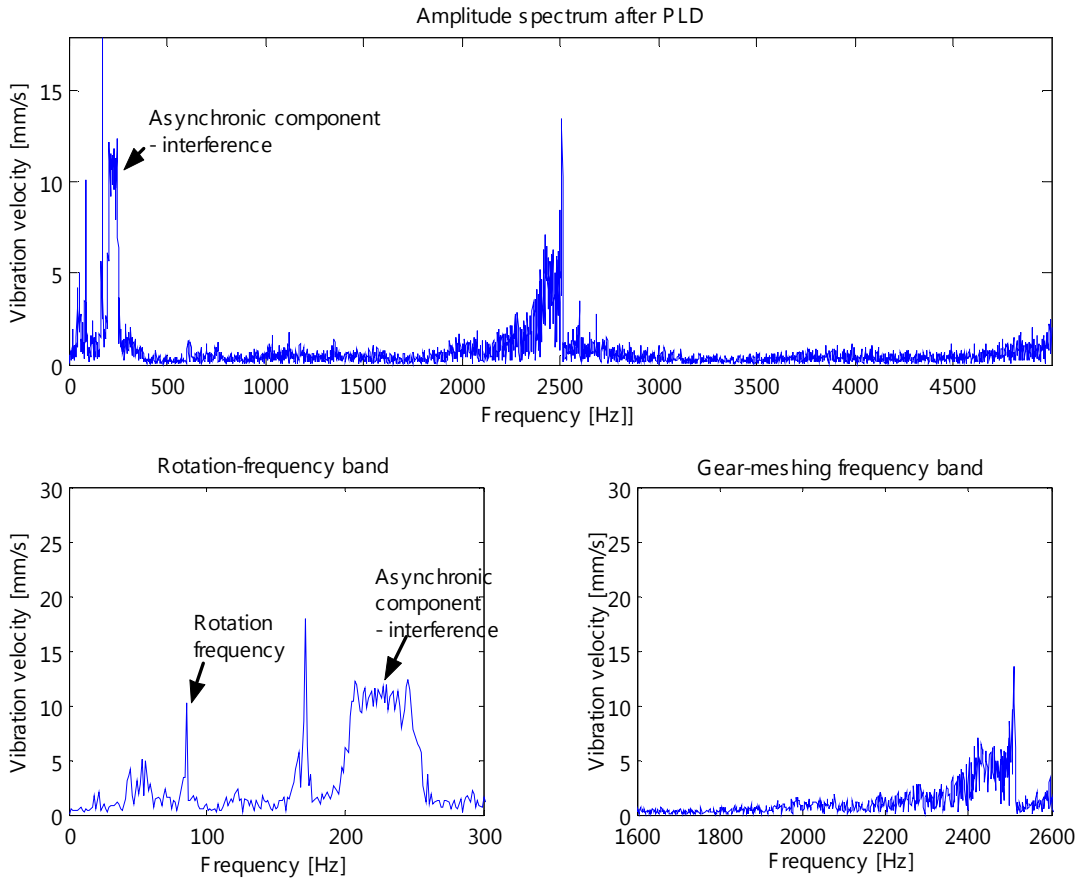

Fig. 4. Amplitude spectrum of vibration velocity of gear-meshing after applying of PLD. Average rotational speed increase - 0.38 % per cycle.

3. Hardware implementation

Electronic implementations of diagnostical procedures are often disregarded in academic discussions. Nevertheless, they can often determine a diagnostic procedure and final results.

The hardware implementation of the PLD can be divided into three separate tasks:

- Marker logic – rotation period indicator,
- Anti-aliasing filter,
- Linear decimation.

3.1 Marker Logic

One ADC channel is dedicated to marker logic (denoted also as rotation period indicator) whose purpose is to synchronise the acquired data with shaft rotation angle. Marker channel is a digital (binary) channel: the marker is detected or not. Nevertheless, the marker signal is usually acquired and transferred to memory as a standard ADC channel e.g. 14-bit channel. This results in inadequate utilization of the resources. Furthermore, in the PLD, the crucial information is not the state of the marker but time-slots between two successive markers. Calculating time periods in a software manner requires significant amount of time, firstly for reading marker state in the external memory and secondly for calculating the time slots by means of microprocessor. This would require tens of clock cycles per marker sample. As a result, dedicated (hardware) marker logic was designed in FPGA, consequently marker states are not transferred to external memory at all. The dedicated marker logic detects markers, then calculates and writes to external memory only time periods between two successive markers.

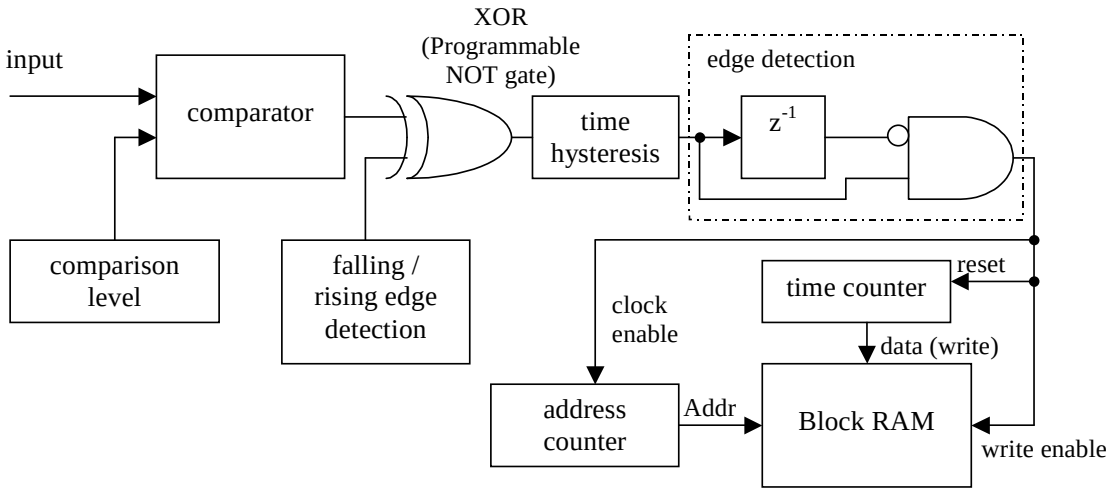

Fig. 5. Block diagram of the marker logic

The block diagram of the marker logic is given in Fig. 5. The input signal from ADC is compared with programmable comparison level (usually middle ADC range) to obtain the binary signal. An alternative solution is that the input signal is already binary, therefore the comparator can be skipped. The binary input is recommended as the number of analog ADC channels is often limited, and marker signal is binary by origin. Then the binary signal may be negated in a XOR gate to detect either rising or falling edge of the marker. Then a special time-hysteresis logic is employed to eliminate input signal glitches. The noise in the input signal is especially destructive when the input signal crosses the comparison level. The hysteresis logic is implemented as a simple up / down counter with saturation. The counter size (the maximum time of eliminated glitches) is programmable. After hysteresis logic, the input signal passes the edge detection logic which produces one clock impulse for every rising edge of the input signal. This impulse initialise the internal or external memory write.

3.2. Finite Impulse Response Filter

One of most remarkable example of employing FPGAs for Digital Signal Processing (DSP) is a FIR filter. Block diagram of FIR filter implemented in FPGAs is given in Fig 6.

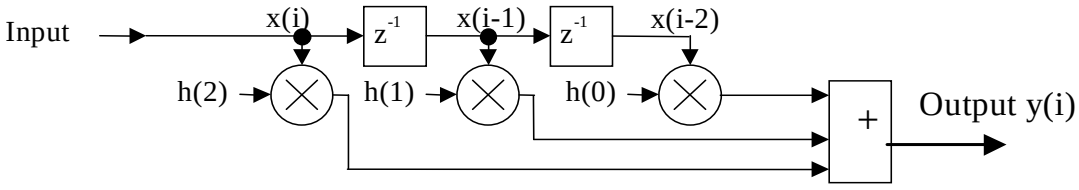

Fig. 6. Block diagram of FIR filter

In Fig. 6, the element z^{-1} denotes one clock delay element and is realized as a flip-flop. Multiplier (*n×n-bit*) occupies less than n^2 Logic Elements (LE) of FPGA, for constant coefficient multipliers the number of occupied resources can be significantly reduced [7]. For example, Xilinx XC3S1500 contains 30 000 LEs. Besides FPGAs incorporate dedicated multipliers, e.g. XC3S1500 contains 32 18×18-bit dedicated multipliers. A *n*-bit adder occupies less than *n* LE. Consequently for symmetric (linear phase) FIR filters with constant coefficient multipliers (for constant FIR filter characteristic) and 12-bit inputs, the number of filter taps is larger than 1000

[7], which means that at least 1000 multiplications are carried out in a single clock cycle. Typical FPGA clock frequency is 50÷400MHz, which is roughly 10 times smaller than for microprocessors. Nevertheless, the number of operations carried out in parallel is significantly larger than that for microprocessors. Summing up, for selected operations, computation power of FPGAs is 10÷1000 (or even more) times greater than for microprocessors.

The above paragraph considered parallel FIR filters implementation which computation power is often far beyond required. Besides, great number of FPGAs resources is occupied by this parallel implementation, and FPGA should also carry out other functions. For example, for analog-digital converter sampling frequency f_S= *100kS/s*, employing parallel FIR filter which can be clocked by *f= 100MHz* would be inefficient. In this case serial FIR architecture is recommended.

For serial FIR architecture only a single multiplication is carried out in a single clock cycle – a similar algorithm as for microprocessors is adopted (see Listing 1). Therefore, *N* clock cycles are required to calculate a single output value (where *N* - the number of filter taps). Consequently, serial architecture clocked by frequency $f= N*f_S$ is equivalent in achievements to parallel architecture clocked by frequency $f=f_S$. Nevertheless, the former occupies only a single multiplier – *N* times less than the parallel counterpart.

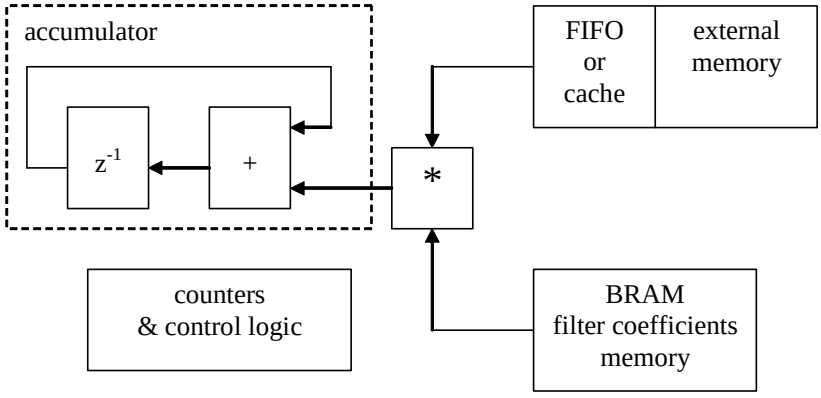

Fig. 7 Block diagram of serial FIR architecture

3.3. Decimation Procedure

In the first approximation, the PLD is a standard decimation procedure for which decimation ratio *D* is not constant. The crucial feature of PLD implementation is high decimation ratio *D* that is roughly $D\approx1000$.

In order to eliminate aliasing effect, a FIR filter is employed before decimation. Unfortunately, the number of the filter taps *N* must be several times greater than the decimation ratio *D* in order to properly filter the input signal. Implementation of such a large filter would require significant FPGA resources. Fortunately for FIR filters, filter calculations may be carried out only for output samples that are not ignored during decimation. This significantly reduces the calculation cost for such a large decimation ratio and is significant advantage of FIR filters over IIR filters.

The FIR filter is implemented employing serial architecture given in Fig. 7. The filter size and filter coefficients are programmable and can be easily changed. The FIR filter employs direct memory access (DMA) to read input data from external memory and to write the resultant data after decimation.

In PLD the decimation ratio *D* linearly increases / decreases as follows:

$$D= D_0 + c \cdot s \tag{3}$$

where:

D_0 – initial decimation ratio,
s – index of the input sample,
c – a constant value derived from the marker time slots.

The decimation ratio *D* is updated according to eq. (3) after each output sample (after decimation). The eq. (3) similarly like filter logic is calculated in hardware, without any co-operation with microprocessors. The microprocessor is only required to calculate values: D_0, *c*, and the total number of input samples. These values are then written to the hardware control registers.

4. Short-Time PLD

With the application of PUD and 10 MS/s sampling, the implementation of the method was created capable of adapting to non-stationary conditions. Furthermore, hardware signal processing enabled the real-time analysis of great number of samples.

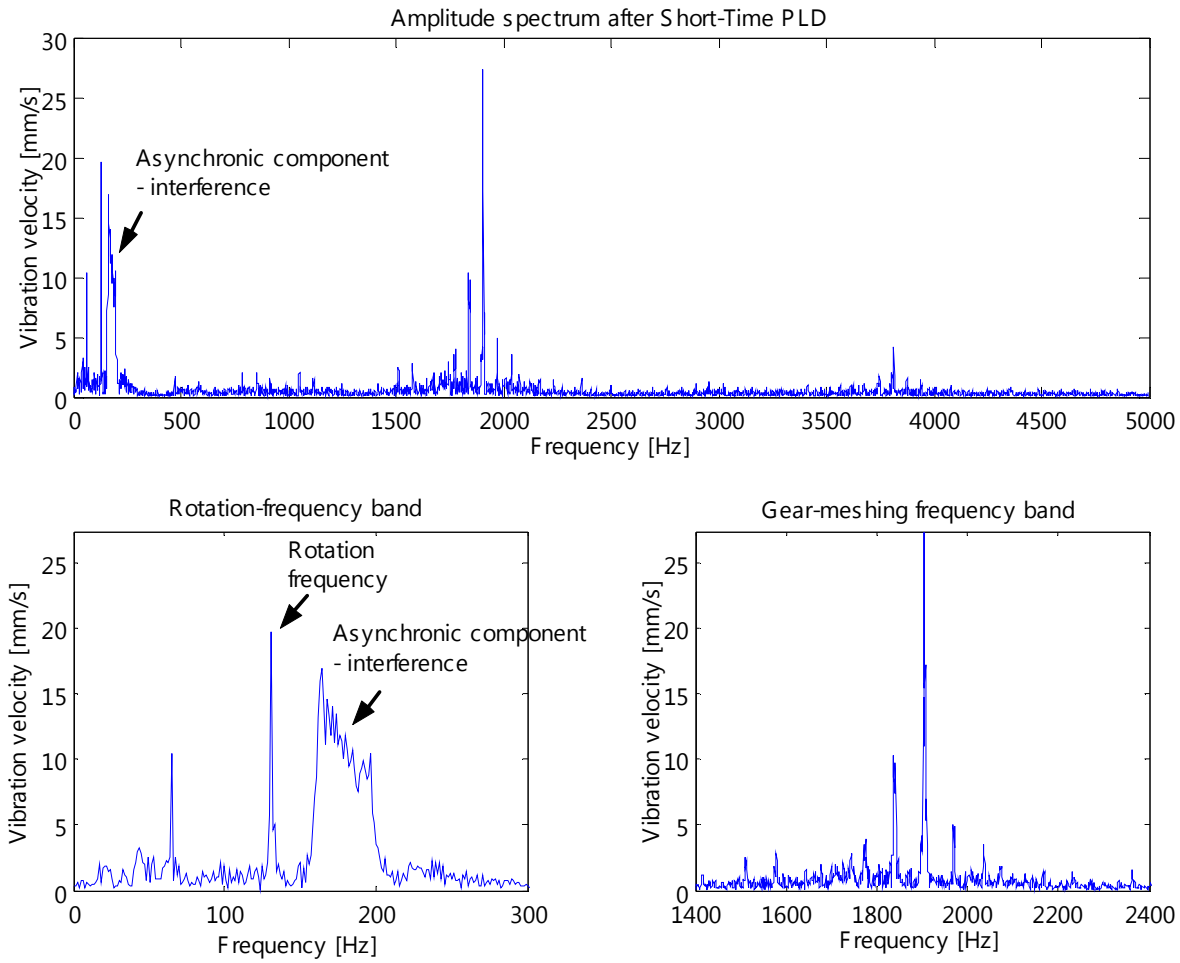

Fig. 8. Amplitude spectrum of vibration velocity gear-meshing after applying Short-Time PLD. Average rotational speed increase 0.38 % per cycle

After taking before-mentioned properties of newly designed device into consideration, a modified PLD was developed, also referred to as Short-Time PLD. Implementation results of the novel method involving the approximation adapting to cycle changes is shown in Fig. 8.

5. Conclusions

Hardware realisation of linear decimation procedure based on FPGA programmable systems gives the possibility of diagnosing cyclical machines at variable operating conditions in real time.

The benefit of applying the method of parallel signal processing is much shorter analysis time that with the processor implementations. It gives great advantage with filtration of signals of great number of samples (millions).

Designed and constructed device (PUD) contains analog-to-digital converters with sampling frequency of 10 MS/s, which enabled to test the developed procedures in high-frequency band at great variations of rotational frequencies. It should be noted that high sampling frequency is crucial for high-speed rotating machines, e.g. gas-turbine machines.

A novel Short-Time PLD method was developed, enabling to adapt the approximation to the cycle changes. This methods solves the problem of non-linear trend change and minimized the error resulted from the original PLD assumption of linear cycle trend in the observation window.

This solution has brought the method nearer to the order analysis without the necessity of applying interpolation filters.

6. References

[1] Adamczyk, J., Cioch, W., Krzyworzeka, P., *Wpływ interpolacji na procedurę liniowej decymacji,* Diagnostyka, vol. 27, 2002.

[2] Adamczyk, J., Cioch, W., Krzyworzeka, P., *Inżynieria Diagnostyki Maszyn. Elementy teorii diagnostyki technicznej* – praca zbiorowa, Roz. 14: *Metody synchroniczne w diagnozowaniu maszyn*, s. 264–278, Radom 2004.

[3] Cempel, Cz., *Diagnostyka wibroakustyczna maszyn*, Wyd. Politechniki Poznańskiej. Poznań, 1985.

[4] Krzyworzeka, P., Cioch, W., *Machine diagnostics in cycle-time scale using linear decimation procedure*, 1st Int. Conf. on Experiments/Process/System/Modelling/Simulation /Optimization, Univ. of Patras. LFME Athens 6–9 July 2005, Greece.

[5] Krzyworzeka, P., Cioch, W., *Dynamiczna kompensacja wpływu zmian długości cyklu na sygnał drganiowy*, Mat XXVII Sympozjum Diagnostyka Maszyn, z. 1, Z.N. Pol. Śl. Katowice 2000.

[6] Jamro, E., Adamczyk, A., Krzyworzeka, P., Cioch, W., *Programowalne urządzenie diagnostyczne stanów niestacjonarnych pracujące w czasie rzeczywistym*, XXXIII Ogólnopolskie Sympozjum Diagnostyka Maszyn, Węgierska Górka, 8.–11.03. 2006 r.

[7] Jamro, E., *Parameterised automated generation of convolvers implemented in FPGAs*, Ph.D. Thesis, University of Science and Technology (AGH-UST), Kraków, 2001.

[8] Krzyworzeka, P., *Wspomaganie synchroniczne w diagnozowaniu maszyn*, Instytut Technologii i Eksploatacji, Radom 2004.

[9] Krzyworzeka, P., Adamczyk, J., Cioch, W., Jamro, E., *Monitoring of nonstationary states in rotating machinery*, Instytut Technologii i Eksploatacji, Radom 2006.

This work has been executed as part of research project at KBN no 6T0720005C/06545.