Acceleration of Digital Stochastic Measurement Simulation Based on Concurrent Programming

Velibor Pjevalica[1], Nebojsa Pjevalica[2], Ivan Kastelan[2], Nenad Petrovic[3]
[1]*JP Srbijagas, Technical Provision Section,*
Novi Sad, Narodnog Fronta 12, Serbia
[2]*Department of Computing and Control Engineering, Faculty of Technical Sciences,*
University of Novi Sad,
Trg D. Obradovica 6, 21000 Novi Sad, Serbia
[3]*School of Electrical Engineering Stari Grad,*
Visokog Stevana 37, 11000 Belgrade
pjeva@uns.ac.rs

***Abstract*—A/D conversion methods improved through stochastic signal superposition, along with oversampling techniques present significant research direction in the area of signal processing and measurement. Concerning that accuracy of those methods rises with length of measurement interval, i.e. integration time; it turns them appropriate for calculation / measurement of the orthogonal transformations. Simulation and validation of above mentioned digital stochastic methods, requires significant computing resource allocation. Long measurement intervals assigned for processing of numerous arithmetic operations over oversampled input signals presents the most demanding computing requirements.**

In this paper, a novel digital stochastic measurement simulation approach is presented and validated. Simulation approach is based on Concurrent Programming technique. General orthogonal transformations are analysed through the stochastic measurement technique. As a reference test case Discrete Fourier Transform is calculated over several periodic input signals converted by the stochastic A/D converter. Time required for a simulation test case accomplishment is analysed as a main performance metric. Final results have proven that Concurrent Programming technique improves simulation speed, without other consequences on measurement performance.

***Index Terms*—Multithreading; Parallel processing; Discrete transforms; Analog-digital conversion; Harmonic analysis.**

I. INTRODUCTION

Orthogonal transforms such as Fourier [1], Hartley [2], Cosine [3], Walsh [4], Wavelet [1], Bilinear [5] and other are very important and utilized in different areas of engineering and information technology. Usage of these transforms is currently mainly in digital and discrete domain. Consequence of this is that all of this transforms are executed on identical hardware: Analog to Digital (A/D) converter, and embedded system with Central Processing Unit (CPU) and a memory. Basic digital stochastic measurements concept, which trades computing speed for A/D converter simplicity is given in [6].

Mathematical definitions of the most used orthogonal transforms are given as a reminder in (1)–(6).

Analog Fourier Transform

$$Y(f) = \int_{-\infty}^{\infty} y(t) e^{-j2\pi ft} dt. \tag{1}$$

Discrete Fourier Transform (DFT)

$$Y_n = \sum_{n=0}^{N-1} y_n e^{-\frac{2\pi j}{N} kn}. \tag{2}$$

Inverse Discrete Fourier Transform

$$y_n = \frac{1}{N} \sum_{k=0}^{N-1} Y_k e^{\frac{2\pi f}{N} kn}. \tag{3}$$

Cosine Transform (applicable to even functions only)

$$F(\omega) = \sqrt{\frac{2}{\pi}} \times \int_{0}^{\infty} y(t) \cos(\omega t) dt. \tag{4}$$

Inverse Cosine Transform (applicable to even functions only)

$$y(t) = \sqrt{\frac{2}{\pi}} \times \int_{0}^{\infty} F(\omega) \cos(\omega t) d\omega. \tag{5}$$

Discrete Hartley Transform

$$H_k = \sum_{n=0}^{N-1} y_n \left(\cos\left(\frac{2\pi}{N} nk \right) + \sin\left(\frac{2\pi}{N} nk \right) \right). \tag{6}$$

In (6), k is the particular sample of Hartley Transform, y_n is particular signal sample, n is counter variable and N is overall number of samples that represents signal. If sampling frequency is at least one order of magnitude higher than the

Manuscript received 8 May, 2018; accepted 23 September, 2018.

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, under grant number TR32014.

frequency of measured signal, we are talking about oversampling method. Throughout design of acquisition system, it is possible to match A/D converter resolution, sampling frequency and data processing speed. In case of orthogonal transforms, there is a need to find average value of integral sum of signal product with corresponding memorized base function for a given coefficient. Basic advantage of the oversampling measurement is that measurement precision increases with the square root of the number of samples.

The rest of the paper is organized as follows. Related work is presented in Section II. Section III describes stochastic processor of the orthogonal transformation. Section IV exposes theory of operation. Section V presents simulation results. Section VI is discussion, followed by the conclusion in Section VII.

II. Related Work

A/D conversion methods enhanced with stochastic signals superposition are well known in the domain of signal processing and measurement.

In general, those methods are derivatives of the oversampling techniques, like for example Delta-Sigma A/D conversion [7] with main difference that stochastic approach excludes feedback loop, which is inevitable in Delta-Sigma A/D conversion. Feedback increases conversion speed and efficiently suppresses quantization noise. However, if the measured input signal is noisy, i.e. if the noise floor is too high in the band of interest, efficiency of the feedback loop drops [8].

It is not easy to determine when significant scientific interest gets focused on A/D conversion systems based on superimposed uniform noise, but the work of Schuchman [9] could be highlighted as an essential from the system level perspective, as the author analyses effects on the generic sinusoidal input signal. Wagdy [10] gives exact fuctional relation between Probability Density Funciton (PDF) and variance of the quantization error if the sinusoidal signal is applied to the uniform quantizer:

$$f_e(e) = 1 + 2\sum_{n=1}^{\infty} J_0(2\pi nA) \times \cos(2\pi ne), \tag{7}$$

$$-\frac{1}{2} \le e \le \frac{1}{2}, \tag{8}$$

where $f_e(e)$ is PDF of the quantization noise e, J_0 is Bessel function of order zero, A is the amplitude of the sinusoidal input signal.

Normalization of the quantization error squared is expressed as

$$\sigma_e^2 = \frac{1}{12} + \frac{1}{\pi^2} 2\sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \times J_0(2\pi nA). \tag{9}$$

Such result matches the most of the theoretical approximations in which uniform distribution of the quantization error is assumed.

If the presented concept is utilized towards True Root Mean Square – True RMS measurement system implementation [11], result brings high performance regarding method robustness and noise immunity.

Further research has brought further generalization of this concept applied on high resolution sampling A/D converters [12]. Practical implementation of prototyping instruments, e.g. harmonic analysers is published in [13] and [14].

Recent results in domain of power measurement [15] expose high accuracy reached with only dual bit A/D converter structure improved with offset error suppression technique. Similar technique is also extended to frequency measurement system [16] where noise immunity creates valuable advantage.

Especially interesting research direction is targeted on biomedical area, where low signal strength, within noisy environment presents demanding measurement task. Typical problems, successfully solved with presented technique are in the domain of electrophysiological monitoring, e.g. Electroencephalography – EEG [17], [18].

III. Stochastic Processor of Orthogonal Transforms

Stochastic additive A/D converter with two noise generators [6] (abbreviated SAADC 2G) is oversampling measurement method. With this method, we digitally measure average value of the integral of the two analog input signals product. Let's denote these signals by $y_1(t)$ and $y_2(t)$. Block scheme of SAADC 2G converter is given in Fig. 1.

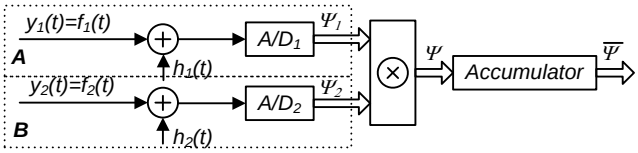

Fig. 1. Block scheme of SAADC 2G converter.

As seen in Fig. 1, it is necessary to perform analog addition of noise sources $h_1(t)$ and $h_2(t)$ to measured signals $y_1(t)$ and $y_2(t)$ respectively. Constraint for operation of SAADC 2G is that noise signals are mutually uncorrelated and that distribution of its amplitudes is uniform within the range of $\pm\Delta_i/2$, i = 1, 2. Δ_i are steps of A/D_1 and A/D_2 from Fig. 1 respectively. If these conditions are fulfilled, numerical accumulator from Fig. 1 contains value $\bar{\psi}$ which is

$$\bar{\psi} = \frac{1}{T}\int_0^T y_1(t)\, y_2(t)\, dt. \tag{10}$$

Upper bound for its absolute measurement error squared is

$$\sigma_s^2 = \frac{1}{T}\frac{\Delta_1^2}{4}\int_0^T y_2^2(t)\, dt + \frac{1}{T}\frac{\Delta_2^2}{4}\int_0^T y_1^2(t)\, dt + \frac{\Delta_1^2\Delta_2^2}{16}. \tag{11}$$

Block *B* in Fig. 1 can be replaced with memory block. This memory block holds dithered samples of basis functions. Such instrument is called stochastic processor of orthogonal transformations (SPOT) [6] and is shown

schematically in Fig. 2.

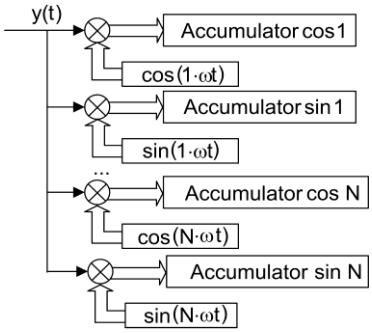

Fig. 2. Block scheme of stochastic processor of orthogonal transformations.

It is shown in [6] that optimal resolution of the stored samples is exactly 2 bits higher than resolutoin of the input A/D converter. In that case, measured signal waveform shape does not influence the measurement accuracy.

If this constraint is satisfied, measurement uncertainty squared is

$$\sigma_s^2 \approx \frac{1}{N}\frac{\Delta_1^2}{8}R^2, \tag{12}$$

where, R is input voltage range in A/D converter, N is overall number of samples in the measurement result and Δ_1 is step of A/D converter.

Equations (10)–(12) are explained in detail in [6] and deeper involvement in metrological part of the problem is out of scope of this paper.

If we take resolution of stored samples of 8 bits, optimal resolution of A/D converter is 6 bits. Input voltage range in A/D converter is R = 2.5 V. Sampling frequency is 1 MHz (20,000 samples per period for 50 Hz mains). Overall duration of the measurement is 2 s (100 periods of mains). With given measurement parameters, upper bound limit for absolute measurement error is

$$\sigma_S = \sqrt{\frac{1}{100\times\left(2\times10^4\right)}\frac{\left(\frac{2,5}{2^{6-1}-1}\right)^2}{8}\ 2.5^2}, \tag{13}$$

which finally gives

$$\sigma_S = 50.4\mu\text{V}. \tag{14}.$$

IV. THEORY OF OPERATION

In simulation, each signal consists of 50 harmonic components. Samples are created from 50 sine and 50 cosine signals with corresponding frequencies. Also, there is a need to calculate noise samples that are with uniform distribution of the amplitudes and with infinite pattern length (infinite period). For that purpose, lagged Fibonacci generator [19] was used. Lagged Fibonacci generator takes another 20 arithmetic operations per sample. According to simulation parameters given, it takes 20,000 samples per one period of mains (50 Hz) and for 2 s of measurements it takes 100 periods of mains. Finally, for reliable statistical result, 50 measurements per one coefficient must be taken into account. Overall, for one coefficient it takes

$$N_C = 20\times50\times20,000\times100 = 2\times10^9, \tag{15}$$

arithmetic operations.

Applying IEEE standard 1547-2003 [20], for complete analysis of the signal in distributive network, 50 harmonics has to be measured, where every harmonic has it's cosine and sine component. Further, it means

$$N_S = 100\times N_C = 2\times10^{11}, \tag{16}$$

i.e. $2\cdot10^{11}$ arithmetic operations. In (16), N_S is overall number of arithmetic operations that must be calculated to obtain all signal spectra. N_C is number of arithmetic operations for one sine or cosine DFT coefficient component.

It is desirable to divide such large number of arithmetic operations among several logical processors inside one physical processor. Utilizing .NET framework and C# programming language, there are three mechanisms for parallel processing SPOT operations. In all of these cases, multithreading can be achieved by putting complete code for one sine or cosine component, including 50 measurement loops into single calculus method. This method has to take information about base function (sine or cosine) as well as harmonic order. The simplest way to pass this data to the method is to send single integer parameter. The calculation method then analyses this integer parameter taken, and if it is even, sine component will be calculated, otherwise if it is odd, cosine component will be calculated. Harmonic order can by calculated from the very same parameter value according to (17) and (18). If parameter P is even number, harmonic order H is calculated as

$$H = \frac{P}{2}+1, \tag{17}$$

and if parameter P is odd number, order of harmonic H is

$$H = \frac{P+1}{2}, \tag{18}$$

where P is the value of the parameter transferred to calculus method, while H is harmonic order. For example, if number zero is transferred, method will simulate measurement of the first sine harmonic. If number one is transferred method will simulate first cosine harmonic.

There are generally several ways to spread execution of a calculating code throughout the logical processors in a single and/or multiple processor cores. Each of these techniques has some advantages and drawbacks that will be discussed in detail in following subchapters A, B and C.

A. Usage of Threads

Threads [21] are independent code sequences that are executed at the same time together with other code sequences on different logical processors of the same or more physical processors. In this way, parallel execution of

the same code with utilization of different parameters (choice of sine or cosine functions and order of harmonics that is taken) does real parallel processing. However, threads are specific objects each of which has its own program counter and stack. The size of a thread in .NET framework is 4 Mb on a 32 bit operating system (OS), or 8 Mb on a 64 bit OS. In multithreading, several threads are part of the same virtual address space and they share code, files and data in both virtual and physical mapping. So, each thread has its own program counter and stack, and shared files, data and code, as it is shown in Fig. 3.

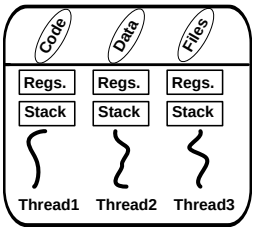

Fig. 3. Block scheme of memory usage by threads in multithreading.

General problem with threads is that threads are "time hungry" objects and operating system (OS) spends significant time during its creation. Second problem is that once started thread can't be started again (unless Thread Pool is used), but still it occupies memory.

B. Usage of ForEach Method of Parallel Class

Inside System.Threading.Tasks namespace of .NET framework, there is a class "Parallel" that allows parallel execution of repeating loop. This is a static class (class that can't instantiate an object of it, but methods of such class are accessible directly via class name), that has "ForEach" method. This method has purpose to start one other method several times with different parameters passed to it in parallel fashion. In case that CPU contains several logical processors, execution of the code is much faster. Opposite to sequential loops execution, in use of "ForEach" method, order of execution can't be guaranteed. For SPOT simulation presented in this paper, order of each coefficient measurement simulation has no influence on overall result, so "ForEach" method is applicable for this purpose. In "ForEach" method programmer can't define level of parallelism. Instead, Run-time environment executes steps of the "ForEach" method in amount that is possible to apply in given moment, as it is shown in Fig. 4.

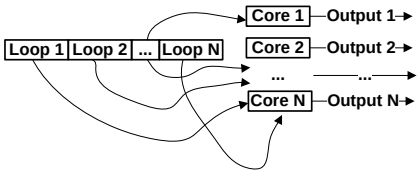

Fig. 4. Block scheme of "ForEach" method execution.

Call of "ForEach" method is made in such a way that in single call, pointer to array with all parameters that has to be processed is sent together with delegate to a method that will be executed in parallel fashion. In particular simulation, array of parameters is array of integers that has values form 1 to 100.

C. Usage of Thread Pooling

Main advantage of Thread Pool concept [22] is that multithreading environment is created and instead of creating new threads over and over again (which is quite time consuming), once formed pool of threads processing new request over and over again. Schematically it is shown in Fig. 5.

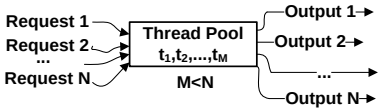

Fig. 5. Block scheme of Thread Pool.

In Fig. 5 we can see N requests and M threads in pool, where $M < N$. Creating new threads burdens CPU and if we have, like in the given simulation 100 requests that has to be fulfilled while each request assumes 2×10^9 arithmetic operations (15), time savings can be significant. The only limitation in Thread Pool usage is that all threads in the pool must be with the same priority and order of thread execution can't be controlled as well as number of threads in the pool, since the Run-time environment creates and controls threads. However, these limitations have no influence on SPOT measurement simulation, and benefit is two order of magnitude shorter simulation time compared with other approaches.

V. Simulation Results

Simulation verification has been performed over a five different test cases. Each test case assumes digital stochastic measurement simulation of DFT transformation applied over a signal which includes significant higher harmonic components. Let's denote given test case signals with s_1, s_2, s_3, s_4 and s_5 respectively.

Waveforms of given signals are presented in Fig. 6 to Fig. 10, while precise mathematical formulas are given in (19)–(23) respectively.

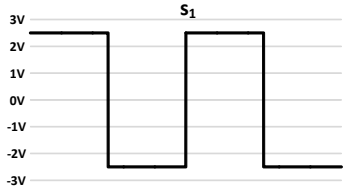

Fig. 6. Signal s_1.

$$s_1 = 2.5\times\frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin\left((2k-1)\omega_0 t\right)}{(2k-1)}. \tag{19}$$

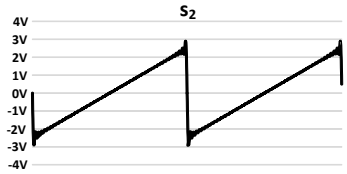

Fig. 7. Signal s_2.

$$s_2 = -\frac{2\times2.5}{\pi}\sum_{k=1}^{50}\frac{(-1)^k}{k}\sin\left(k\omega_0 t\right). \tag{20}$$

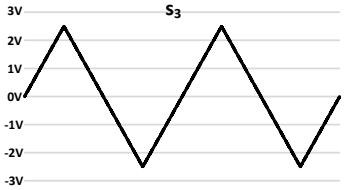

Fig. 8. Signal s_3.

$$s_3 = \frac{4\times 2.5}{\pi^2}\sum_{k=1}^{\infty}\frac{1-(-1)^k}{k^2}\sin(k\omega_0 t), \quad (21)$$

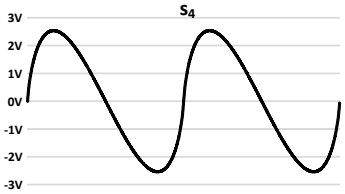

Fig. 9. Signal s_4.

$$s_4 = 2.5\sum_{k=1}^{\infty}\frac{1}{k^2}\sin(k\omega_0 t), \quad (22)$$

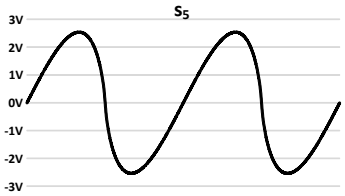

Fig. 10. Signal s_5.

$$s_5 = 2.5\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k^2}\sin(k\omega_0 t). \quad (23)$$

Without any loss of generality and in order to avoid too dense graphics, only initial three (first, second and third) and final three (forty-eight, forty-ninth and fiftieth) harmonic errors are presented for each simulated test case. Figure 11–Fig. 15 correspond to simulated harmonic measurement error levels for signals s_1 to s_5 respectively.

In Table I–Table V results of simulations are presented together with simulation time. In tables, parameter σ_{sA} is averaged standard deviation, obtained as arithmetic mean value of 100 variances square rots. Each of these variance square roots is calculated for corresponding sine or cosine coefficient and for all 50 DFT coefficients it makes one hundred results which are averaged. Among these 100 results value σ_M presents maximum. Fourth column presents simulation time.

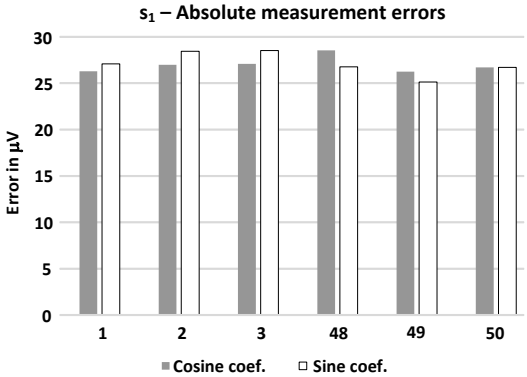

Fig. 11. Signal s_1. Harmonics measurement errors.

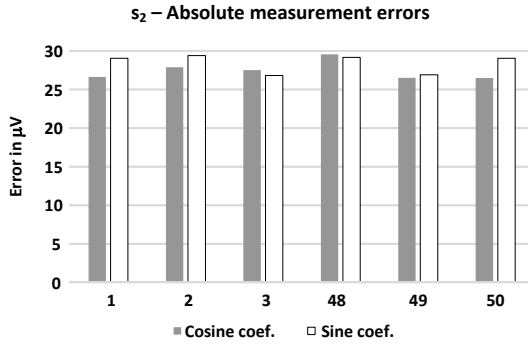

Fig. 12. Signal s_2. Harmonics measurement errors.

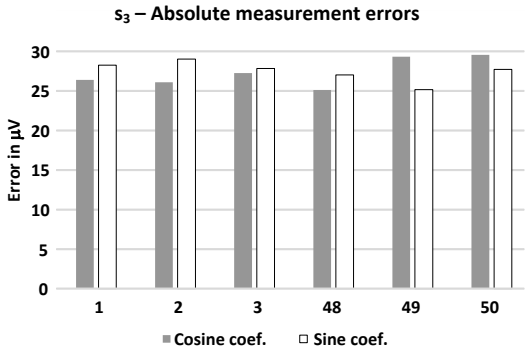

Fig. 13. Signal s_3. Harmonics measurement errors.

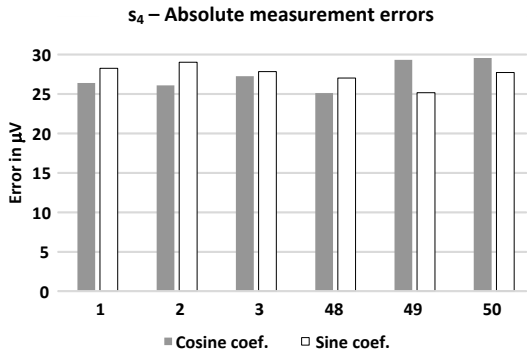

Fig. 14. Signal s_4. Harmonics measurement errors.

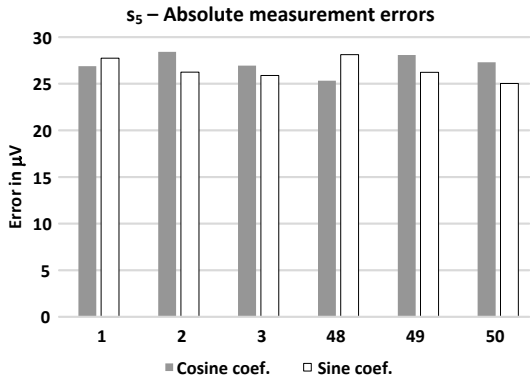

Fig. 15. Signal s_5. Harmonics measurement errors.

TABLE I. SIMULATION RESULTS FOR SIGNAL S_1.

Signal s_1	**σ_{sA}**	**σ_M**	**Simulation time**
No threads	27.41 µV	33.1 µV	10h:41min:31s
Regular threads	27.45 µV	32.5 µV	2h:15min:12s
Parallel.ForEach	27.43 µV	30.5 µV	1h:05min:08s
Thread Pool	27.42 µV	31.7 µV	6min:11s

TABLE II. SIMULATION RESULTS FOR SIGNAL S_2.

Signal s_2	**σ_{sA}**	**σ_M**	**Simulation time**
No threads	28.39 µV	33.3 µV	10h:41min:11s
Regular threads	28.44 µV	32.7 µV	2h:15min:05s
Parallel.ForEach	28.41 µV	31.4 µV	1h:05min:06s
Thread Pool	28.40µV	32.6 µV	6min:13s

TABLE III. SIMULATION RESULTS FOR SIGNAL S_3.

Signal s_3	σ_{sA}	σ_M	Simulation time
No threads	26.39 µV	31.0 µV	10h:41min:07s
Regular threads	26.44 µV	31.7 µV	2h:15min:11s
Parallel.ForEach	26.41 µV	30.6 µV	1h:05min:22s
Thread Pool	26.40 µV	31.4 µV	6min:15s

TABLE IV. SIMULATION RESULTS FOR SIGNAL S_4.

Signal s_4	σ_{sA}	σ_M	Simulation time
No threads	27.55 µV	31.1 µV	10h:41min:13s
Regular threads	26.57 µV	29.4 µV	2h:15min:12s
Parallel.ForEach	26.21 µV	30.5 µV	1h:05min:15s
Thread Pool	26.47 µV	30.2 µV	6min:17s

TABLE V. SIMULATION RESULTS FOR SIGNAL S_5.

Signal s_5	σ_{sA}	σ_M	Simulation time
No threads	27.48 µV	28.9 µV	10h:41min:21s
Regular threads	26.51 µV	29.3 µV	2h:15min:41s
Parallel.ForEach	26.39 µV	31.4 µV	1h:05min:49s
Thread Pool	26.37 µV	29.2 µV	6min:09s

VI. DISCUSSION

According to the results given in tables from I to V, it is obvious that foremost simulation approach is usage of Thread Pool, which accelerates simulation process significantly and generally saves OS resources.

Using Thread Pool in simulations that takes 200 billion arithmetic operations reduces simulation time for two orders of magnitude, compared with simulation on the same CPU without Thread Pool. On CPU with four cores and eight logical processors with 3.3 GHz clock, and PC platform with 16 GB RAM memory; time taken to finish simulation was 2 hours and 15 minutes by using threads, and only 6 minutes by using Thread Pool. Advantage on the system level reveals through the optimal parallel distribution of computing tasks on available logical processors. Threads usage generally offers execution parallelism, while optimal solution within multithreading environment brings Thread Pool.

For simulation of SPOT operations on five proposed test case signals, simulation results are completely inside theoretically predicted boundaries, but simulation time on the same CPU is significantly different depending on the thread technique used.

VII. CONCLUSIONS

Simulation of the electronic measurement systems is generally demanding computing task. It could be solved through the usage of state of the art simulation tools, which offers libraries of predefined models and objects, which mainly shortens the time for system modelling, but on the other hand keeps designers away from the physical implementation and as a drawback in many cases extends execution time.

Opposite approach analysed in this paper assumes creation of local library of models and blocks used for a system set-up which could be precisely tuned for accomplishment of the required measurement simulations in a minimal time frame. Such approach requires significant time needed for initial generation of necessary models, libraries and multithreading environment adjustment. Advantage of presented solution is minimization of the CPU resources, on the first place execution time.

Digital Stochastic Measurement technique generally trades oversampling and large number of arithmetic operations for final accuracy. In applications like orthogonal transformations, numerical complexity rises and it makes simulations long lasting tasks. Presented results have proven that concurrent programming technique offers optimal computing method which successfully bridges complex numerical requirements and shortens execution time.

For implementation of demanding simulation tasks, like those on stochastic implementation of orthogonal transformations, Thread Pool should be considered as regular practice.

From practical engineering point of view, quite promising research direction assumes simulations running on virtual scalable machine in the cloud, where CPU can be defined with a lot more logical processors. This final proposal presents main future study path.

REFERENCES

[1] A. M. Gaonda, M. M. A. Salama, M. R. Sultan, A. Y. Chikhani, "Power Quality Detection and Classification Using Wavelet–Multiresolution Signal Decomposition", *IEEE Transaction on Power Delivery*, vol. 14, no. 4, pp. 1469–1476, 1999. DOI: 10.1109/61.796242.

[2] Wen-Liang Hsue, Wei-Ching Chang, "Real discrete fractional Fourier, Hartley, generalized Fourier and generalized Hartley transforms with many parameters", *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 62, no. 10, pp. 2594–2605, 2015. DOI: 10.1109/TCSI.2015.2468996

[3] Chen Luo, Zhijie Mo, Yijun Zhou, Mengliang Kuang, "Static modeling and simulation of Linear Object based on differential geometry and Discrete Cosine Transform", *IEEE Int. Conf. Mechatronics and Automation (ICMA 2017)*, Takamatsu, Japan, 2017, pp. 1342–1347. DOI: 10.1109/ICMA.2017.8016012.

[4] A. M. A. Bin Ateeq, S. A. Abbasi, A. R. M. Alamoud, "Hardware Realization of Walsh Functions and Their Applications Using VHDL and Reconfigurable Logic", *ICM*, 2002, pp. 58– 61. DOI: 10.1109/ICM-02.2002.1161496.

[5] Xiaoli Xi, Zhengwei Li, Jiangfan Liu, Jinsheng Zhang, "FDTD simulation for wave propagation in anisotropic dispersive material based on bilinear transform", *IEEE Trans. Antennas and Propagation*, vol. 63, no. 11, pp 5134–5138, 2017. DOI: 10.1109/TAP.2015.2475629.

[6] V. Pjevalica, V. Vujicic, "Further generalization of the low-frequency true-RMS instrument", *IEEE Trans. Instrumentation and Measurement*, vol. 59, no. 3, pp. 736–744, 2010. DOI: 10.1109/TIM.2009.2030874.

[7] S. R. Norsworthy, R. Schreier. G. C. Temes, *Delta-Sigma Data Converters: Theory, Design, and Simulation*. New York: Wiley-IEEE Press, 1997, pp. 14–16.

[8] J. Ireland, A. Cryer, J. M. Williams, E. Houtzager, R. Hornecker, H. E. van den Brom, "Design of delta-sigma feedback loop for quantum voltage digitizer", *Conf. Precision Electromagnetic Measurements (CPEM 2016)*, 2016, Ottawa, Ontario Canada DOI: 10.1109/CPEM.2016.7540457.

[9] L. Schuchman, "Dither signals and their effect on quantization noise", *IEEE Trans. Communication Technology*, vol. 12, no. 4, pp. 162–165, 1964. DOI: 10.1109/TCOM.1964.1088973.

[10] M. F. Wagdy, W. M. Ng, "Validity of uniform quantization error model for sinusoidal signals without and with dither", *IEEE Trans. Instrumentation and Measurement*, vol. 38, no. 3, 1989, pp. 718–722. DOI: 10.1109/19.32180.

[11] V. Vujicic, S. Milovancev, M. Pesaljevic, D. Pejic, I. Zupunski, "Low-frequency stochastic true RMS instrument", *IEEE Trans. on Instrumentation and Measurement*, vol. 48, no. 2, 1999, pp. 467–470. DOI: 10.1109/19.769630.

[12] V. Vujicic, "Generalized low-frequency stochastic true RMS instrument", *IEEE Trans. Instrumentation and Measurement*, vol. 50, no. 5, pp. 1089–1092, 2001. DOI: 10.1109/19.963164.

[13] J. Tomic, M. Kusljevic, V. Vujicic, "A new power system digital harmonic analyzer", *IEEE Trans. Power Delivery*, vol. 22, no. 2, pp. 772–780, 2007. DOI: 10.1109/TPWRD.2007.893372.

[14] B. Santrac, M. Sokola, Z. Mitrovic, I. Zupunski, V. Vujicic, "A novel method for stochastic measurement of harmonics at low signal-to-noise ratio", *IEEE Trans. Instrumentation and Measurement*, vol. 58, no. 10, 2009, pp. 3434–3441, DOI: 10.1109/TIM.2009.2017661.

[15] M. Urekar, D. Pejic, V. Vujicic, S. Avramov-Zamurovic, "Accuracy improvement of the stochastic digital electrical energy meter", *Measurement*, vol. 98, pp. 139–150, 2017. DOI: 10.1016/j.measurement.2016.11.038.

[16] A. Radonjic, P. Sovilj, V. Vujicic, "Stochastic measurement of power grid frequency using a two-bit A/D converter", *IEEE Trans. Instrumentation and Measurement*, vol. 63, no. 1, pp. 56–62, 2014. DOI: 10.1109/TIM.2013.2277515.

[17] P. Sovilj, S. Milovancev, V. Vujicic, "Digital stochastic measurement of a nonstationary signal with an example of EEG signal measurement", *IEEE Trans. Instrumentation and Measurement*, vol. 60, no. 9, pp. 3230–3232, 2011. DOI: 10.1109/TIM.2011.2128670.

[18] M. Urekar, P. Sovilj, "EEG dynamic noise floor measurement with stochastic flash A/D converter", *Biomedical Signal Processing and Control*, vol. 38, pp. 337–345, 2017. DOI: 10.1016/j.bspc.2017.07.006.

[19] Hui Li, Zhonghua Liu, Junkai Yi, "Fast Elliptic scalar multiplication using lagged Fibonacci generator", *IEEE 5th Int. Conf. Software Engineering and Service Science*, Being, China, 2014, pp. 488–491. DOI: 10.1109/ICSESS.2014.6933612.

[20] IEEE standard for interconnecting distributed resources with electric power systems, IEEE standard 1547-2003

[21] Zhong Cheng, Ke Qi, Liu Jun, Huang Yi-Ran, "Thread-level parallel algorithm for sorting integer sequence on multi-core computers", *Fourth Int. Symposium on PAAP*, Tianjin, China, 2011, pp. 37–41. DOI: 10.1109/PAAP.2011.57.

[22] Shuai Zhang, Tao Li, Qiankun Dong, Xuechen Liu, Yulu Yang, "CPU-assisted GPU thread pool model for dynamic task parallelism", *IEEE Int. Conf. Networking, Architecture and Storage (NAS 2015)*, Boston, USA, 2015, pp. 135–140. DOI: 10.1109/NAS.2015.7255234.