A step towards real-time time-frequency analyses with varying time-frequency resolutions: Hardware implementation of an adaptive S-transform

Nevena Radović[1a], Veselin N. Ivanović[1b], Igor Djurović[1c], Marko Simeunović[2d], Ervin Sejdić[3,4,5e]

[1]: *University of Montenegro, Department of Electrical Engineering, Cetinjski put bb, 81000 Podgorica, MONTENEGRO*
[2]: *University of Donja Gorica, Faculty for Information Systems and Technologies, Oktoih 1, 81000 Podgorica, MONTENEGRO*
[3]*: Department of Electrical and Computer Engineering, Swanson School of Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA*
[4]*: The Edward S. Rogers Department of Electrical and Computer Engineering, Faculty of Applied Science and Engineering, University of Toronto, Toronto, ON M5S 2E4, CANADA*
[5]*: North York General Hospital, Toronto, ON M2K 1E1, CANADA*

e-mail: *nevenar@ucg.ac.me*

[a]: ORCID ID: 0000-0002-0214-883X
[b]: ORCID ID: 0000-0002-3422-5019
[c]: ORCID ID: 0000-0003-3856-2706
[d]: ORCID ID: 0000-0001-8006-4466
[e]: ORCID ID: 0000-0003-4987-8298

Abstract— This paper addresses the rarely considered issue of hardware implementation of the S-transform, a very important time-frequency representation with many practical applications. Various improved, adaptive, and signal-driven versions of the S-transform have been developed over the years, but only its basic (non-adaptive) form has been implemented in hardware. Here, a novel hardware implementation of the adaptive S-transform is proposed extending the previously developed design. To minimize hardware demands, the proposed approach is based on an appropriate approximation of the frequency window function considered in the S-transform. The adaptivity of the transform to the signal is achieved by an optimal choice of a window parameter from the set of predefined values, meaning that for each window parameter the S-transform is calculated. To additionally save hardware resources, the proposed design does not require storing all calculated values, but only two in each iteration. The proposed multiple-clock-cycle architecture is developed on the field-programmable gate array device and its performance is compared with other possible implementation approaches such as the hybrid and single-clock-cycle ones. It is demonstrated that the developed design minimizes hardware complexity and clock cycle time compared to alternative approaches, and is significantly more efficient than the software realization. Both noiseless and noisy multicomponent highly nonstationary signals were considered. An excellent match between the results of the hardware and the "exact" adaptive S-transform evaluation obtained through the MATLAB implementation is demonstrated. Lastly, the execution time that can be estimated in advance is also an important practical feature of the developed design.

Keywords— Adaptive S-transform, Time-frequency representation, Hardware implementation, FPGA.

1. INTRODUCTION

The short-time Fourier transform has a constant resolution along both time and frequency. However, for nonstationary signals with the fast spectrum and instantaneous frequency variations, the constant time-frequency resolution can be a limiting factor. For example, a time-frequency resolution can be adjusted for allowing separation of close signal components at low frequencies, but this can be unfeasible for separating components at high frequencies since they will overlap in the time-frequency plane. The wavelet and other time-scale transforms can partially resolve this issue by producing varying time-frequency resolution [1]. However, experts in many fields have difficulties interpreting the time-scale representations leading to difficulties when it comes to the design of algorithms for detection and feature extraction.

The S-transform has been developed as an attempt to overcome pointed drawbacks of classical time-frequency and time-scale transforms [2, 3]. Moreover, the wavelet transforms require phase correction which is not the case when considering the S-transform [2]. Therefore, the S-transform is used in the time-frequency analysis as an alternative to the short-time Fourier and wavelet transforms [2-5]. It can be considered as a combination of classical time-frequency and time-scale representations sharing many properties with the Gabor transform [6]. The S-transform is very popular in the biomedical field since these applications require varying time-frequency resolution (better frequency resolution at low frequencies and better time resolution on higher frequencies) and keeping a common frequency domain [7-13]. The nonstationary signals with rapid changes in the spectral content and in the instantaneous frequency appear in numerous other fields. For example, hyperbolic frequency modulation appears in geological and geophysical systems, radars, underwater acoustics, etc [14, 15]. The S-transform and its improved versions are proven to be excellent tools for the analysis of these and similar signals. Stockwell noted in his seminal paper [3] that in years to come it can be expected significant developments in the S-transform alleviating its limitations. Driven by a wide application area, it was fulfilled through the enormous development of various adaptive, signal-driven, modified, and upgraded S-transforms over the past 20 years, [16-20]. Among them, the adaptive S-transforms have attracted significant attention, and need for its efficient hardware implementation was imposed.

A fast and reliable evaluation of underlying time-frequency representations is very important in practical applications. However, the time-frequency-based transforms have large computational demands since the corresponding transform functions are calculated in a large number of time-frequency points. This can limit their real-time applicability. Often such fast response/evaluation is not possible with software realization, but a hardware implementation of time-frequency techniques can overcome this issue. The development of fast and low-cost hardware systems for evaluation of the S-transforms and their advanced forms able to rapidly respond to changing signal (frequency and spectral) content is very important for sensor nodes, drones, and other devices. There are two common approaches to accomplish this:

1) *single-clock-cycle implementation* [21-23] - provides excellent execution time, but at the expense of the hardware complexity and
2) *multiple-clock-cycle implementation* [24-27] - reduces hardware complexity providing a larger execution time than the single-clock-cycle approach.

To combine desired properties of approaches, low hardware complexity, and execution time, the signal adaptive and pipelined hardware implementations of time-frequency methods are recently proposed [28-32]. Hardware realization of the standard (non-adaptive) S-transform is considered in [33], as well as its implementation on the FPGA Cyclone III EP3C16E144C7 circuit. This paper extends research toward the S-transform with a data-driven window function [16]. The concentration of the S-transform in the time-frequency plane is improved by evaluating several S-transforms calculated for different window functions and selecting the best one based on the concentration measure [34]. This leads to an improved, adaptive S-transform, [16-18]. However, compared with its basic form, the adaptive S-transform has a significantly increased time consumption and calculation complexity leading to a limited number of applications. Hence, creating an efficient hardware design with reasonable (minimal possible) hardware demands (to be suitable for on-a-chip implementation and consequently to enable the deployment of the transform in various applications) is the main challenge in the hardware development of this transform. Our approach relies on the idea that we have to approximate frequency window function with simpler realization than in [33] and to address the evaluation of multiple S-transforms within the same memory resources by developing a mechanism not requiring the simultaneous presence of all calculated values. The main contribution of the paper is development of the design satisfying the following requirements: conditions of efficiency (regarding execution and clock cycle times, and the hardware complexity), operation in real-time, suitability for on-a-chip implementation, and high accuracy.

The paper is organized as follows. Theoretical background related to the S-transform, adaptive S-transform, concentration measure, and frequency window function approximation is given in Section 2. Section 3 provides the hardware implementation, while Section 4 presents testing, verification, and results of the proposed design. Discussion about the design including possibilities for further development is outlined in Section 5, while concluding remarks are given in Section 6.

2. S-TRANSFORM AND ITS ADAPTIVE FORMS

The S-transform of a signal, $x(t)$, is defined as

$$S_x(t, f) = \int_{-\infty}^{+\infty} x(\tau) w(t-\tau, f) e^{-j2\pi f \tau} d\tau, \quad (1)$$

where the window function is given by

$$w(t,f)=\frac{|f|}{\sqrt{2\pi}}e^{-\frac{t^2|f|^2}{2}}. \tag{2}$$

At low frequencies, the S-transform has a higher frequency resolution compared to time resolution, while at high frequencies it can better separate close components in time than in frequency domain. Such a behavior is desirable in numerous practical applications, and it resembles the wavelet analysis while maintaining phase information which is important for signal reconstructions. However, numerous signal types appear in the time-frequency analysis, and a "one-size-fits-all" approach is not possible. A potential solution is the development of time-frequency representations having design parameter(s) that can be selected considering some appropriate criterion [34-36]. The time-frequency concentration measures are the common criterion for the selection and adjustment of time-frequency representations.

To obtain an optimal time-frequency representation with the S-transform, consider the following window function:

$$w_p(t,f)=w(t,|f|^p)=\frac{|f|^p}{\sqrt{2\pi}}e^{-\frac{t^2|f|^{2p}}{2}}, \tag{3}$$

where $S_x^{(p)}(t,f)$ denotes the S-transform evaluated with $w_p(t,f)$. As demonstrated in [16], larger p value provides a narrower window (better time resolution), while for a smaller p value, a wider window that can improve frequency resolution is obtained. An appropriate range of p values proposed in [16] is $p\in[0,1]$.

Our goal is to design an objective function to lead an optimal p value yielding $S_x^{(p)}(t,f)$. This can be done using the proposed three steps procedure given in the following:

Step 1: Normalize the energy of all S-transforms from the set:

$$\tilde{S}_x^{(p)}(t,f)=\frac{S_x^{(p)}(t,f)}{\sqrt{\iint_{(t,f)}|S_x^{(p)}(t,f)|^2\,dtdf}}. \tag{4}$$

Step 2: Calculate the concentration measure[1] $CM(p)$ for all p values and their corresponding S-transforms [16, 34]:

$$CM(p)=\frac{1}{\iint_{(t,f)}|\tilde{S}_x^{(p)}(t,f)|\,dtdf}. \tag{5}$$

[1] There are several concentration measures for time-frequency representations. The commonly used measures are based on the Renyi entropy. However, all of them have drawbacks analyzed in [34]. Nevertheless, the measure used in this paper avoids some of the drawbacks outlined in the previous publication. In general, by keeping constraint (4), this measure minimizes the region of the time-frequency plane covered by signal components that correspond to high resolution. It is already used in various research papers including consideration of the adaptive S-transform in [16]-[20].

Step 3: Select an optimal p value by maximizing (5):

$$p_{opt} = \arg\max_{p} CM(p). \tag{6}$$

Note that p_{opt} represents the global maximum of all $CM(p)$, (5). Precisely, within the adaptive S-transform calculation, Np S-transforms are evaluated, CM is calculated for each of them, and p_{opt} corresponds to the S-transform reaching the highest CM. Simply, the S-transform calculated with p_{opt} represents an adaptive S-transform, commonly referred to as the S-transform with a data-driven window function:

$$AS(t, f) = \tilde{S}_x^{(p_{opt})}(t, f). \tag{7}$$

For the fast realization of the S-transform, calculating the convolution integral in equation (1) should be avoided. Instead, the frequency domain expression can be used [2, 3]:

$$S_x^p(t, f) = \int_a X(f + a) W_p(f, a) da \tag{8}$$

where $W_p(f, a) = e^{-\pi a^2 / |f|^{2p}}$ is the window function expressed in the frequency domain. Hardware implementation of $W_p(f, a)$ is not a simple task since its Taylor series expansion requires a large number of terms to converge at higher f. An efficient evaluation of the window function is studied in [33] where two Taylor series expansions around f=0 and f=f_1 are performed. Then, the frequency window function is approximated considering the first three terms of the selected Taylor series expansion (around f=0 or f=f_1). The selection of expansion series is done by the switching rule depending on f.

The data-driven S-transform requires multiple S-transform calculations, and it is imperative for these calculations to be carried out in an efficient manner. Now, our goal is to find an efficient approximation of $W_p(f, a)$ by considering two Taylor series expansions with two terms. We propose the following approach. As in [33], the first Taylor series expansion is evaluated around f=0. The second expansion point f=f_1 and parameters of the switching rule are determined by simulations. The optimization criteria considered minimization of the mean squared difference between the approximated and true $W_p(f, a)$ values and resulted by the following approximation:

$$\hat{W}_p(f, a) = \begin{cases} \hat{W}_{p1}(f, a), & |a| \le \left(\sqrt{2} - \frac{1}{2}\right) \frac{|f|^{2p}}{2\pi} \\ \hat{W}_{p2}(f, a), & \left(\sqrt{2} - \frac{1}{2}\right) \frac{|f|^{2p}}{2\pi} \le |a| \le \left(\sqrt{2} + 1\right) \frac{|f|^{2p}}{2\pi} \\ 0, & |a| > \left(\sqrt{2} + 1\right) \frac{|f|^{2p}}{2\pi}. \end{cases} \tag{9}$$

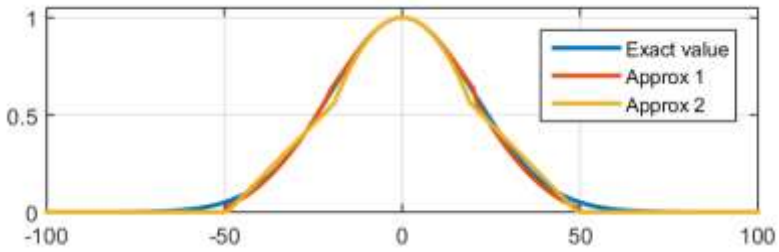


Fig. 1. Approximation of the frequency window function: Exact value – blue line, approximation from [33] – red line, and approximation (9) – yellow line

Table 1. List of abbreviations and notations used within the presentation of the hardware implementation.

Abbreviation/notation	***Description***
CLK	Clock cycle
Register file	One-dimensional structure of M registers
Register block	Two-dimensional structure of $M \times N$ registers
CumADD	Cumulative pipelined adder
CM1	Concentration measure of S-transform from Register block_1
CM2	Concentration measure of S-transform from Register block_2

where:

$$\hat{W}_{p1}(f,a) = 1 - \frac{2\pi^2 a^2}{|f|^{2p}},$$
$$\hat{W}_{p2}(f,a) = \exp\left[-\frac{(3-2\sqrt{2})}{2}\right]\left[1-\left(\sqrt{2}-1\right)\frac{2\pi a}{|f|^{p}}\right]. \quad (10)$$

Fig. 1 depicts the exact value of the frequency window function (blue color line), approximation from [33] (red line), and the proposed approximation (9) (yellow line). It can be observed a good match between these functions. In the case of the approximation from [33], the mean squared difference between the exact and approximated value is $4.81 \cdot 10^{-4}$ while for approximation (9) it is slightly higher, $5.26 \cdot 10^{-4}$, but with a reduced number of terms in the Taylor series expansion.

Note that within the adaptive S-transform calculation, the multiple S-transform evaluations with different p and hence multiple evaluations of the frequency window function are performed. Approximation (9) provides both savings in hardware resources and adequate precision.

3. HARDWARE IMPLEMENTATION OF THE ADAPTIVE S-TRANSFORM

The proposed hardware implementation of the adaptive S-transform is shown on Fig. 2. The system inputs are stored in 3 ROM blocks, named ROM_1, ROM_2, and ROM_3.

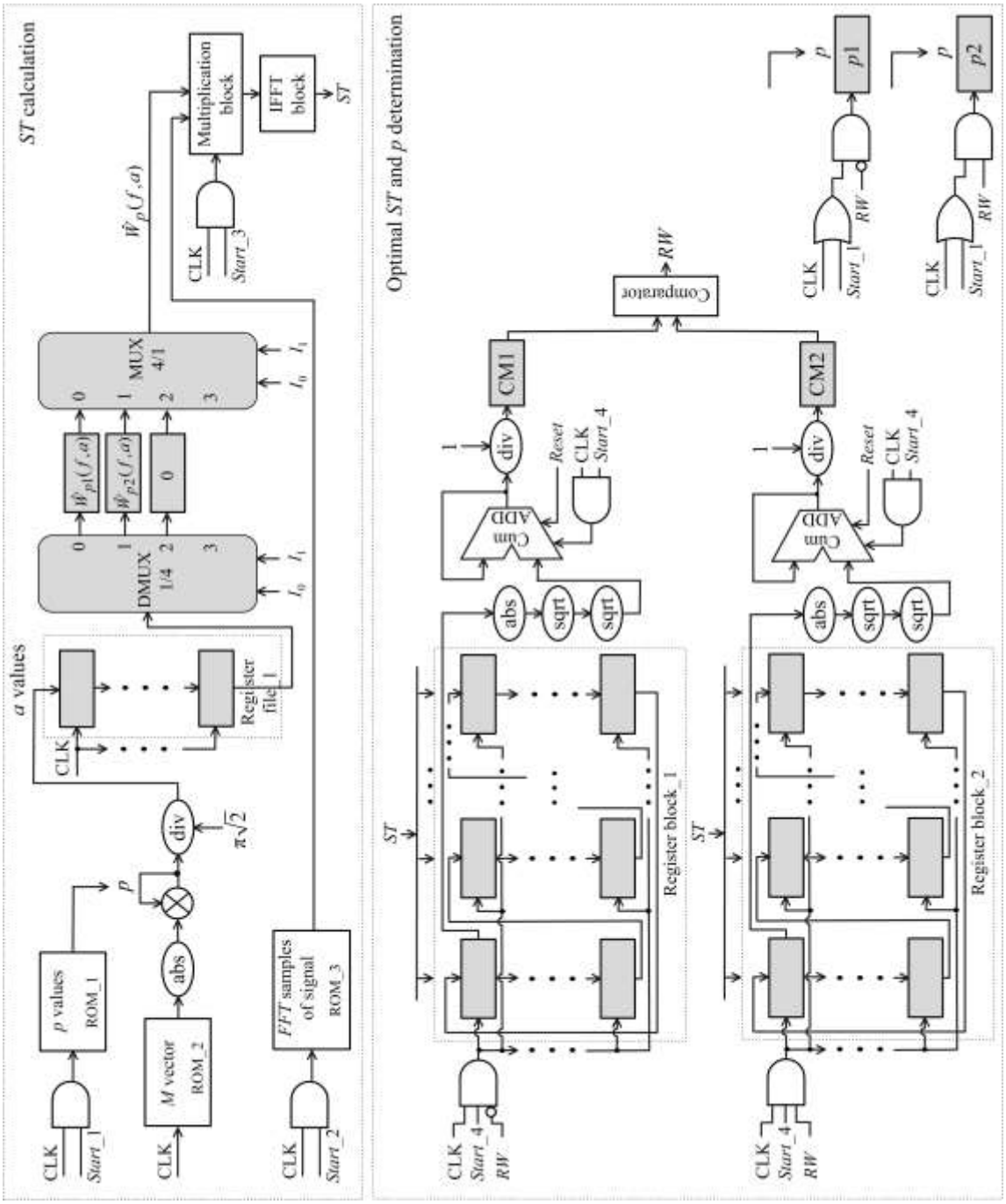


Fig. 2. Proposed hardware design of the adaptive S-transform. Unit denoted by “abs” calculates absolute value of the input value, unit denoted by “div” performs division operation, unit denoted by “sqrt” performs square root operation, whereas the Comparator checks whether the first input value is greater than the second one.

These ROMs are used as follows:

- ROM_1 contains a set of parameter p values. Owing to each M-th clock cycle (CLK), determined by the control signal *Start*_1, a new parameter p value is read. Within the M CLKs in which parameter p takes the same value, a particular window function $\hat{W}_p(f,a)$ is calculated. Note that parameter p values are sequentially read from ROM_1.
- ROM_2 contains the vector of values $(-M/2) : (M/2+1)$ stored N times (one after another, in a way that after each particular vector (i.e. after its last element $(M/2+1)$), the first element $(-M/2)$ of the next vector follows), where M and N represent the analyzed signal duration and the S-transform window width, respectively. For simplicity reasons, the content of ROM_2 will be referred to as M-vector in the rest of the paper. With each CLK, a value from M-vector is read;
- ROM_3 contains samples of the Fourier transform of the analyzed signal. These samples can be created in a standard module for the Fourier transform realization [22–24]. The same module, but intended for the inverse Fourier transform realization, is used at the output for the creation of a particular S-transform. Here, the standard fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) modules are used. Within the M CLKs, determined by the *Start*_2 control signal, M Fourier transform samples (FTs) are read from the ROM_3, but in following M CLKs, reading from ROM_3 is disabled (by the zero value of the *Start*_2 control signal). Note that FTs are sequentially read from ROM_3.

Outputs of the system are:

- A value of the parameter p corresponding to the adaptive S-transform,
- The adaptive S-transform.

The implementation includes 4 Register files (Register file_1, 2, 3, 4, where Register file_2, 3, 4 are used within the Multiplication block) and 3 Register blocks (Register block _1, 2, 3, where Register block_3 is used within Multiplication block). Each Register file contains M memory locations, while Register blocks contain $M \times N$ memory locations. The initial values of all these locations are zeros. Register file_1 denotes the used FIFO delay block, whereas Register files_2, 3, 4 are shift memory buffers.

Based on the parameter p value read from the ROM_1 and the value of the M-vector element read from the ROM_2, the corresponding a value is calculated and stored in an appropriate register of the Register file_1. Note that within M CLKs, determined by the control signal *Start*_1, the same parameter p value is used. In each of these CLKs (once per a CLK), this parameter participates within the calculation of different values of a. Further, the obtained a values are used within the calculation of the window function $\hat{W}_p(f,a)$.

The de-multiplexor DMUX 1/4 in combination with the multiplexor MUX 4/1 is used to implement the window function in the frequency domain $\hat{W}_p(f,a)$, based on its approximation from equation (9). To this end, DMUX 1/4 and MUX 4/1 are controlled by the same selection signals I_1 and I_0. With each CLK, a values are led one by one to the input of DMUX 1/4. Depending on a,

control signals I_1 and I_0 select one of 3 possible options of $\hat{W}_p(f,a)$ (see eq. (9)). Calculated samples of $\hat{W}_p(f,a)$ are then stored in Register file_2 (Fig. 3). Simultaneously, Fourier transforms samples of the analyzed signal are read from ROM_2 and are stored in the Register file_3, and Register file_4 (Fig. 3). As shown in Fig. 3, the corresponding samples of $\hat{W}_p(f,a)$ and Fourier transforms samples of the analyzed signal are multiplied in parallel, and the obtained products are stored in the parallel-in-parallel-out Register block_3 (also in parallel and managed by the *Start*_3 control signal). Products saved in the Register block_3 are inputs of the IFFT block, which produces the S-transform at the system output.

Control signals I_1 and I_0 are created in a way presented in Fig. 4. With each CLK, *a* values are led to a set of 4 comparators that produce control signals *A*, *B*, *C*, and *D* at their outputs. As seen, each of these comparators is used to compare *a* value with the corresponding limit defined in $\hat{W}_p(f,a)$ approximation (9). Finally, based on the values of the produced signals *A*, *B*, *C*, and *D*, the signals I_1 and I_0 are defined in a way shown by the functional table given in Fig. 4 (b).

Samples of the calculated S-transform are imported to the Register block_1 or the Register block_2. During ($N\times M$) CLKs determined by the *Start*_4 control signal, these samples are propagated through the Register block in which they are imported, such that each sample propagates through each location of the Register block (in the corresponding CLK). In this way, all S-transform samples are used in the concentration measure calculation (Fig. 2). In our design, S-transform samples are taken from the first location of the Register block in the corresponding CLK. Note that the storage location (Register block_1 or _2) in the *n*-th time instant depends on the control signal *RW*, created in the previous, (*n*-1)th, time instant. The control signal *RW*, created by the Comparator (Fig. 2) checks whether the concentration measure of the S-transform from the Register block_1 (CM1) is greater than or equal to the concentration measure of the S-transform from the Register block_2 (CM2), where the concentration measures CM1 and CM2 are calculated according to equation (5). The following cases can be distinguished:

- CM1≥CM2, when the control signal *RW* takes a unit value that provides the S-transform samples to be imported to the Register block_2;
- CM1<CM2, when the control signal *RW* takes zero value that provides the S-transform samples to be imported in the Register block_1;

In this way, the S-transform with the greater concentration measure is preserved and the S-transform with a lower concentration measure will be rewritten by the S-transform calculated for the next parameter *p* value. Note that the memory resources of the system are saved by using only two Register blocks. Precisely, *Np* S-transforms are evaluated, but only two of them are stored: the currently calculated S-transform and the S-transform with the highest concentration measure. After calculating all S-transforms, the optimal S-transform is stored in one of two Register blocks (_1 or _2), while the control signal *RW* value describes in which of these blocks it is placed.

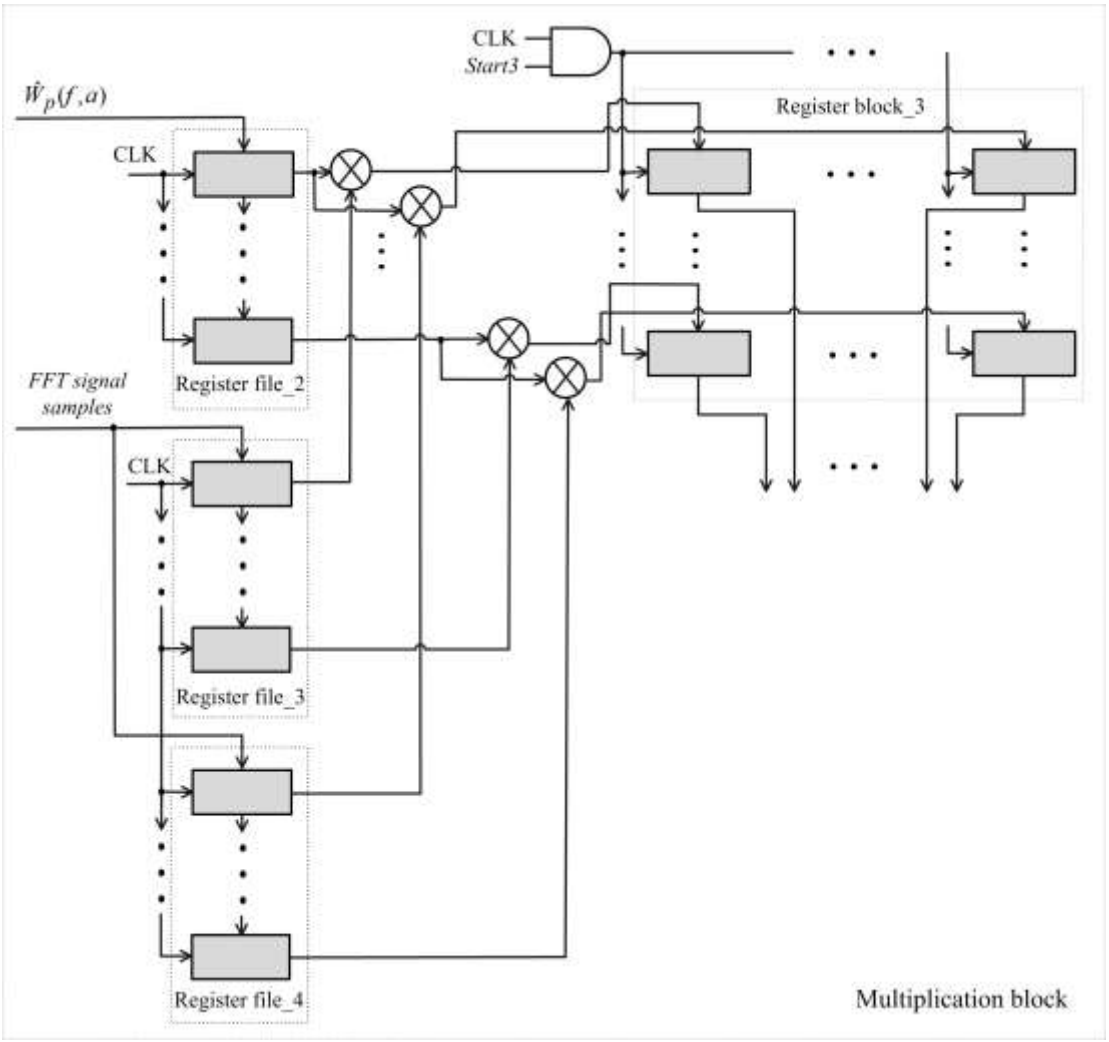
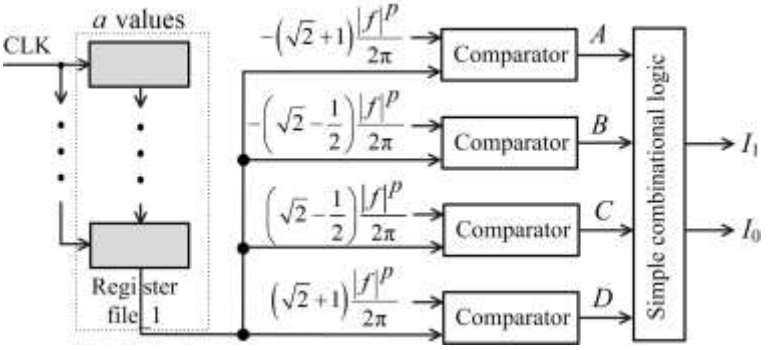


Fig. 3. Multiplication block from Fig. 2.



A	B	C	D	I_1	I_0
0	0	0	0	1	0
1	0	0	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1
1	1	1	1	1	0

(*a*) (*b*)

Fig. 4. (a) Unit that creates multiplexer/de-multiplexer selection signals I_1, I_0. (b) Functional table determining input-output dependence of the simple control logic from graphic (a).

Similar logic is used to preserve the value of the parameter p corresponding to the optimal S-transform. To this end, registers $p1$ and $p2$ are used to store parameter p values corresponding to S-transforms with greater and lower concentration measures,

where the register containing the parameter p corresponding to the S-transform with the lower concentration measure is rewritten. The register with the optimal parameter p value is also determined by the control signal *RW*.

The concentration measure, based on (5), is implemented by using units that perform the absolute value, square root, and the division operations, as well as the cumulative adder CumADD. The CumADD operation is managed by the control signals *Start*_4 and *Reset*. The CumADD starts its calculation after $2M$ CLKs (the time consumed within the calculation of the S-transform) and performs calculations during ($N \times M$) CLKs. Within these CLKs, all S-transform samples calculated for a particular parameter p value are included in addition. The CumADD is reset after ($2M+N \times M$) CLKs by using the control signal *Reset*. After that, the process is repeated: the CumADD waits for the next $2M$ CLKs (thanks to the control signal *Start*_4) until the S-transform corresponding to the next parameter p value is obtained and the addition during the next ($N \times M$) CLKs is performed. This procedure ends with the processing of the S-transform corresponding to the last parameter p value.

The unity value of the control signal *End* terminates the calculation. At that moment, the optimal S-transform is placed in either Register blocks_1 or _2, while the corresponding p value is stored in one of the registers $p1$ or $p2$:

- if the final value of control signal *RW* is one, the optimal S-transform is stored in Register block_1, while the corresponding parameter p value is stored in register $p1$;
- otherwise, the optimal S-transform is stored in Register block_2, while the corresponding parameter p value is stored in register $p2$.

Control signals *Start*_1, *Start*_2, *Start*_3, *Start*_4, *Reset*, and *End* are determined by the parameters from the Configuration registers given on Fig. 5(a). These signals are generated by modules consisting of binary counters (of variable lengths) combined with binary magnitude comparators whose references are corresponding parameters from Configuration registers. The timing of these signals is presented on Fig. 5(b).

As it can be noted, Fig. 5(b), control signals can disable the functioning of some parts of the system for specific periods of time. It can appear that latency is introduced in the system in this way, making its real-time execution questionable. However, there are parts of the system that are working during these periods of time since they are not controlled with the same signals as the disabled parts. For example, the control signal *Start*_2 enables importing of the FT samples of the analyzed signal in the Multiplication block within M CLKs, but in the following M CLKs the same control signal disables that importing (and so on). However, while the importing of the FT samples is disabled, samples $\hat{W}_p(f,a)$ are calculated. In this way, the latency in the specific part of the system does not cause the latency of the whole system and therefore does not endanger the real-time execution.

To complete the calculation, the developed design takes a fixed number (per S-transform sample and parameter p) of CLKs. Calculation of each S-transform and its concentration measure takes the same number of CLKs, that is ($2M+N \times M$) CLKs. Since

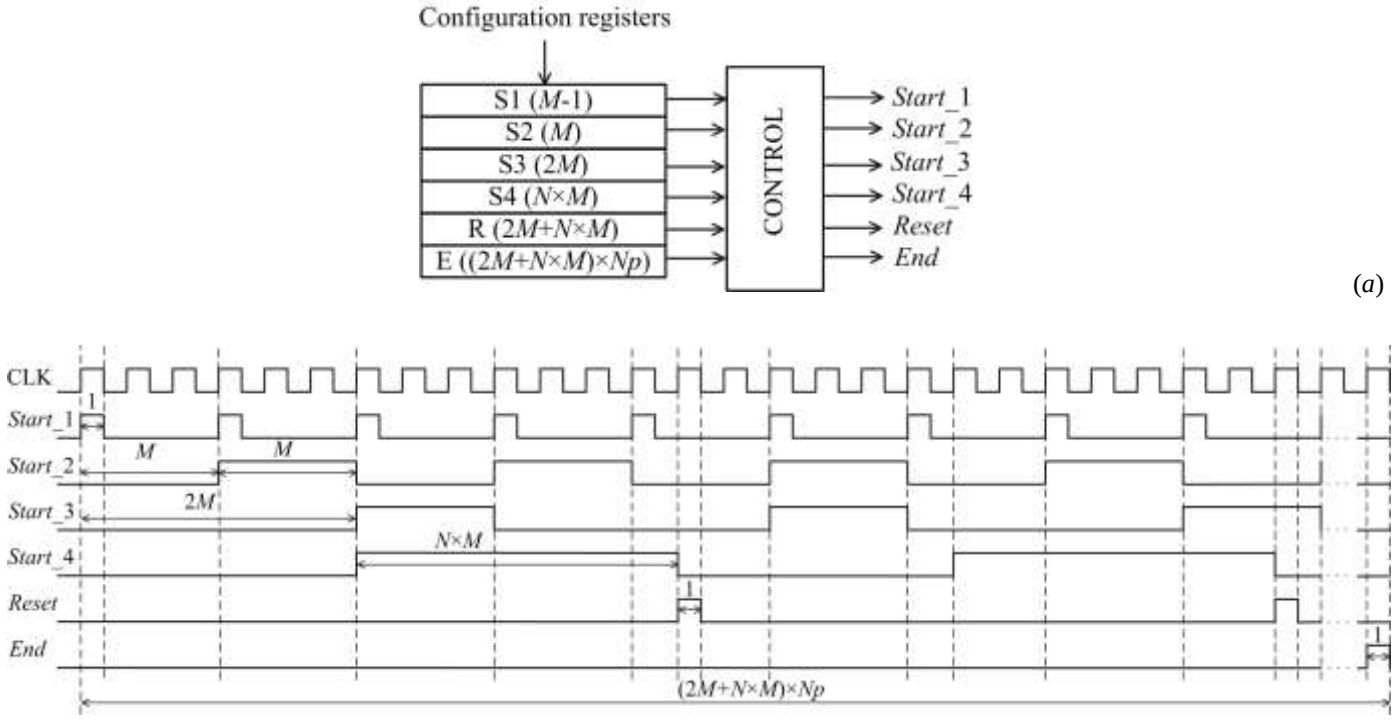


Fig. 5. (a) Control unit for determination of control signals *Start*_1, *Start*_2, *Start*_3, *Start*_4, *Reset*, and *End*, (b) Timing of these signals.

this calculation is repeated as many times as there are different values of parameter *p* (*Np*), $(2M+N\times M)\times Np$ CLKs are required to complete the overall calculation process. As it can be seen, the total number of required CLKs depends on *a priory* known algorithm parameters *Np*, *N* and, *M*. Therefore, the time required for the execution of the developed system can be estimated in advance. Note that this can be very important in practice.

4. TESTING, VERIFICATION, AND RESULTS OF THE PROPOSED APPROACH

The proposed hardware design is implemented on the FPGA EP4CGX150CF23C7 device from the Cyclone IV GX family. The testing of the implementation is done considering the following two-component signal:

$$x(t) = A(\cos(2\pi(20\log(5t+1))) + \cos(2\pi(5t) + 2\pi(45t^2))) \quad (11)$$

observed within the range $t\in[0,1)$. The first signal component has hyperbolic modulation while the second one is the linear FM signal. Both components are of practical importance. The hyperbolic modulated signal is sensitive to time-frequency resolution since it can rapidly change within short time interval. Within the testing, the unity signal amplitude, A=1, sampling rate of T_s=1/256, and the window width of N=128 are selected. The signal is further corrupted with the additive white Gaussian noise and different

Table 2. Utilization of hardware resources of the FPGA Cyclone IV GX EP4CGX150CF23C7 device used within the implementation of the developed design. The design is determined with the following parameters M=256, N=128, Np=100, and input data length l=32 for test signal 1, (11).

Resources	***Utilization***
Total Logic Elements	99,348/149,760 (66%)
Combinational Functions	13/149,760 (<1%)
Dedicated Logic Registers	99,335/149,760 (66%)
Total Pins	65/287 (23%)
Total Virtual Pins	0
Total Memory Bits	1,641,600/6,635,520 (25%)
Embedded Multiplier 9-bit elements	257/720 (36%)
Total GXB Receiver Channel PCS	0/4 (0%)
Total GXB Receiver Channel PMA	0/4 (0%)
Total GXB Transmitter Channel PCS	0/4 (0%)
Total GXB Transmitter Channel PMA	0/4 (0%)
Total PLLs	0/6 (0%)

input signal-to-noise ratios (SNRs) are considered, where $SNR_{in}=10\log_{10}(A^2/\sigma_n^2)$ and σ_n^2 is the variance of the additive noise. Input data, as well as data obtained within the implementation, are written in the signed 32-bit fixed-point notation including an 8-bit fraction. All details regarding the on-a-chip implementation are given in Table 2. Note also that within the execution, the CLK rate of 100MHz is achieved. The software implementation was achieved in MATLAB.

Three different cases of the input signal are observed: the noiseless signal (11), the noisy signal (11) with SNR_{in}=10dB, and the noisy signal (11) with SNR_{in}=5dB. For each of these cases, the standard (non-adaptive) S-transform, the adaptive S-transform (8) obtained by using the software (MATLAB) simulation, and the adaptive S-transform (8) obtained by using the real-time hardware implementation, are presented in Fig. 6. As can be noted, among the considered time-frequency representations and in all observed cases, the adaptive S-transform provides significantly improved signal concentration and substantially higher signal representation quality in comparison to the standard (non-adaptive) S-transform. This is particularly noticeable at lower SNR as shown in Fig. 6 (c). The results obtained by the developed real-time hardware implementation of the adaptive S-transform (Fig. 6, right-hand column graphics), correspond well to the software simulations (Fig. 6, central column graphics). This statement can visually be noted from Fig. 6, but also can numerically be proven by presenting both the hardware- and the software-based results regarding the optimal parameter p-value and differences between corresponding S-transforms (Table 3).

To measure similarities between two representations, the following difference measures between the adaptive S-transform obtained by hardware ($S_{x,H}^{p}(t,f)$) and software ($S_{x,S}^{p}(t,f)$) implementations are used:

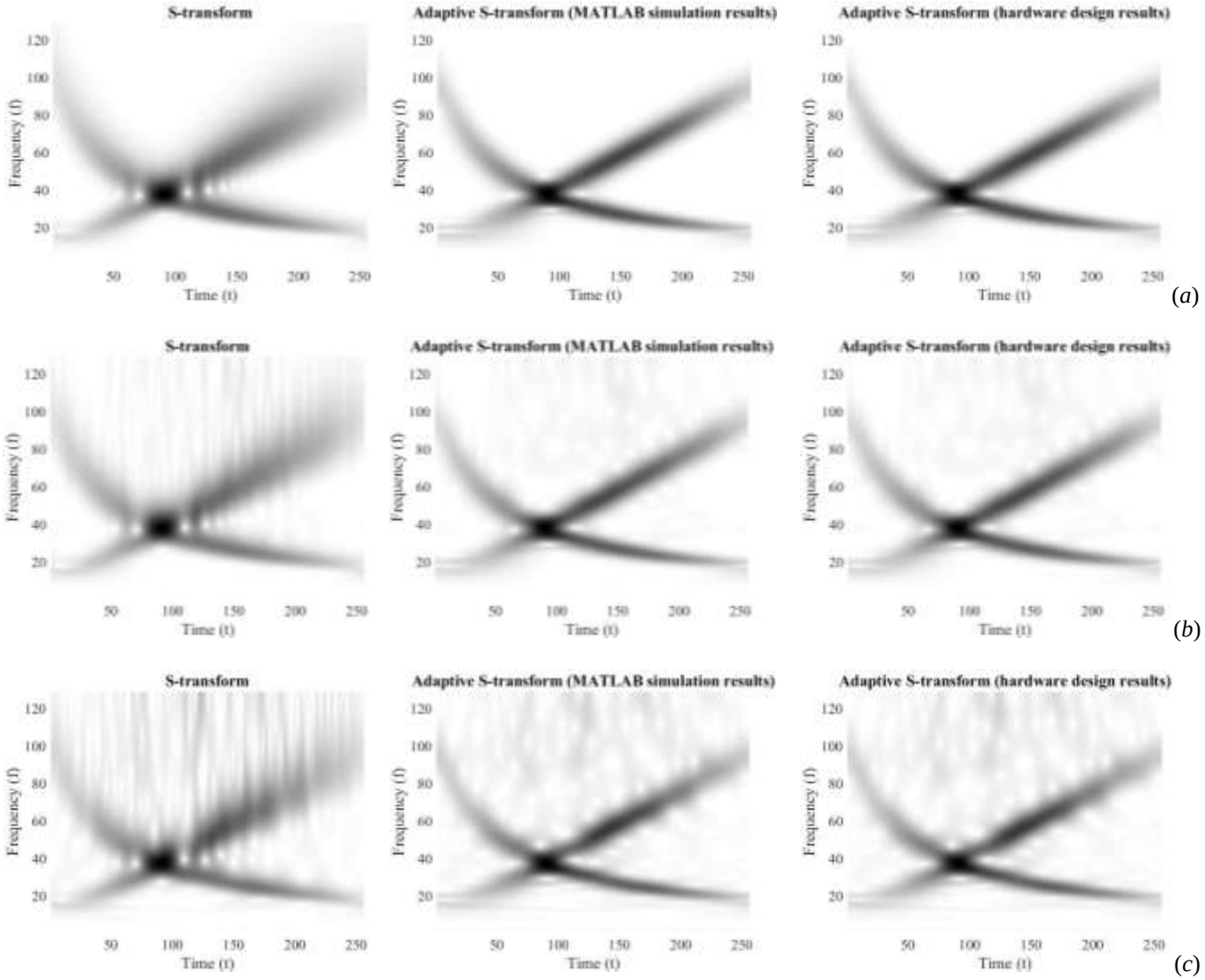


Fig. 6. (a) Noiseless test signal 1 $x(t)$, (b) Noisy test signal 1 $x(t)$ with SNR=10dB, (c) Noisy test signal 1 $x(t)$ with SNR=5dB. Left-hand graphics present the standard (non-adaptive) S-transform, central graphics present the adaptive S-transform obtained by using the software simulation, whereas right-hand graphics present the adaptive S-transform obtained by using the developed hardware implementation.

Maximal difference (MAX): $\mathrm{MAX} = \max_{(t,f)} \left| S^{p}_{x,H}(t,f) - S^{p}_{x,S}(t,f) \right|,$

Mean absolute difference (MEAN): $\mathrm{MEAN} = \operatorname{mean}_{(t,f)} \left| S^{p}_{x,H}(t,f) - S^{p}_{x,S}(t,f) \right|,$ and

Pseudo-signal to noise ratio (PSNR): $\mathrm{PSNR} = \dfrac{\max_{(t,f)} \left| S^{p}_{x,S}(t,f) \right|^2}{\operatorname{mean}_{(t,f)} \left| S^{p}_{x,H}(t,f) - S^{p}_{x,S}(t,f) \right|^2} [\mathrm{dB}].$

Table 3. Hardware-based and software-based results regarding the optimal parameter p value, and differences between adaptive S-transforms.

Considered cases	Optimal parameter p		MAX	MEAN	PSNR[dB]
	Hardware	Software			
Noiseless test signal 1	0.7663	0.7657	0.0018	$1.8 \cdot 10^{-4}$	38.53
Noisy test signal 1, SNR_{in}=10dB	0.3297	0.3293	0.0028	$7.0 \cdot 10^{-4}$	30.83
Noisy test signal 1, SNR_{in}=5dB	0.2242	0.2240	0.0071	$1.6 \cdot 10^{-3}$	22.63
Noiseless test signal 2	0.6364	0.6359	0.0025	$3.8 \cdot 10^{-4}$	44.56
Noisy test signal 2, SNR_{in}=10dB	0.6040	0.6036	0.0042	$9.2 \cdot 10^{-4}$	37.68
Noisy test signal 2, SNR_{in}=5dB	0.5475	0.5471	0.0089	$2.1 \cdot 10^{-3}$	29.61

An excellent match can be observed for all considered measures. It can readily be concluded that the software-based results coincide with the hardware-based ones. Precisely, PSNR, as probably the most reliable measure in this case, is above 22dB even for relatively strong noise of SNR=5dB. Small differences are the consequence of the used approximation (9)-(10), as well as the different lengths of registers used in the hardware development (32-bit notation) and the software simulations (64-bit notation).

Obtaining time-frequency representations with varying time-frequency resolution is recently addressed with numerous techniques developed for biomedical signals. In particular, it is recognized as a growing need for these representations in the EEG signal analysis [37], [38], but also in other fields [39–41]. Accordingly, it is interesting to test the proposed design on real-life signals. To this end, the famous bat signal recorded by a team from Beckman Institute of the University of Illinois is also considered here. We have considered 310 samples of the bat signal sampled with a sampling interval of 7ms.

As in the case of test signal 1, three different cases of the input bat signal (test signal 2) are observed: the noiseless bat signal, the noisy bat signal with SNR_{in}=10dB, and the noisy bat signal with SNR_{in}=5dB. For each of these cases, the standard (non-adaptive) S-transform, the adaptive S-transform (8) obtained by using the software (MATLAB) simulation, and the adaptive S-transform (8) obtained by using the real-time hardware implementation, are presented in Fig. 7. As expected, excellent agreement between "exact" results (obtained by using MATLAB simulations) and output of the proposed hardware can be visually observed. Excellent agreement is supported by the numerical results given in Table 3 (last three rows).

After comparing results achieved with the software simulation and the proposed hardware implementation, these two approaches are additionally compared regarding the execution times. As considered in Section 3, the developed hardware implementation requires multiple and fixed number of CLKs to complete the calculation and to create signals at the output (see Table 4). For parameters M=256, N=128, Np=100, and the CLK time of 10ns (corresponds to the maximum CLK rate of 100MHz), the execution time of the developed hardware implementation coincides with 32.7ms. On the other hand, the MATLAB simulation, performed on a high-performance computer (32GB RAM, i7 2.60 GHz Intel processor) requires the execution time of 0.539s, i.e., about 16.5 times more than the hardware implementation.

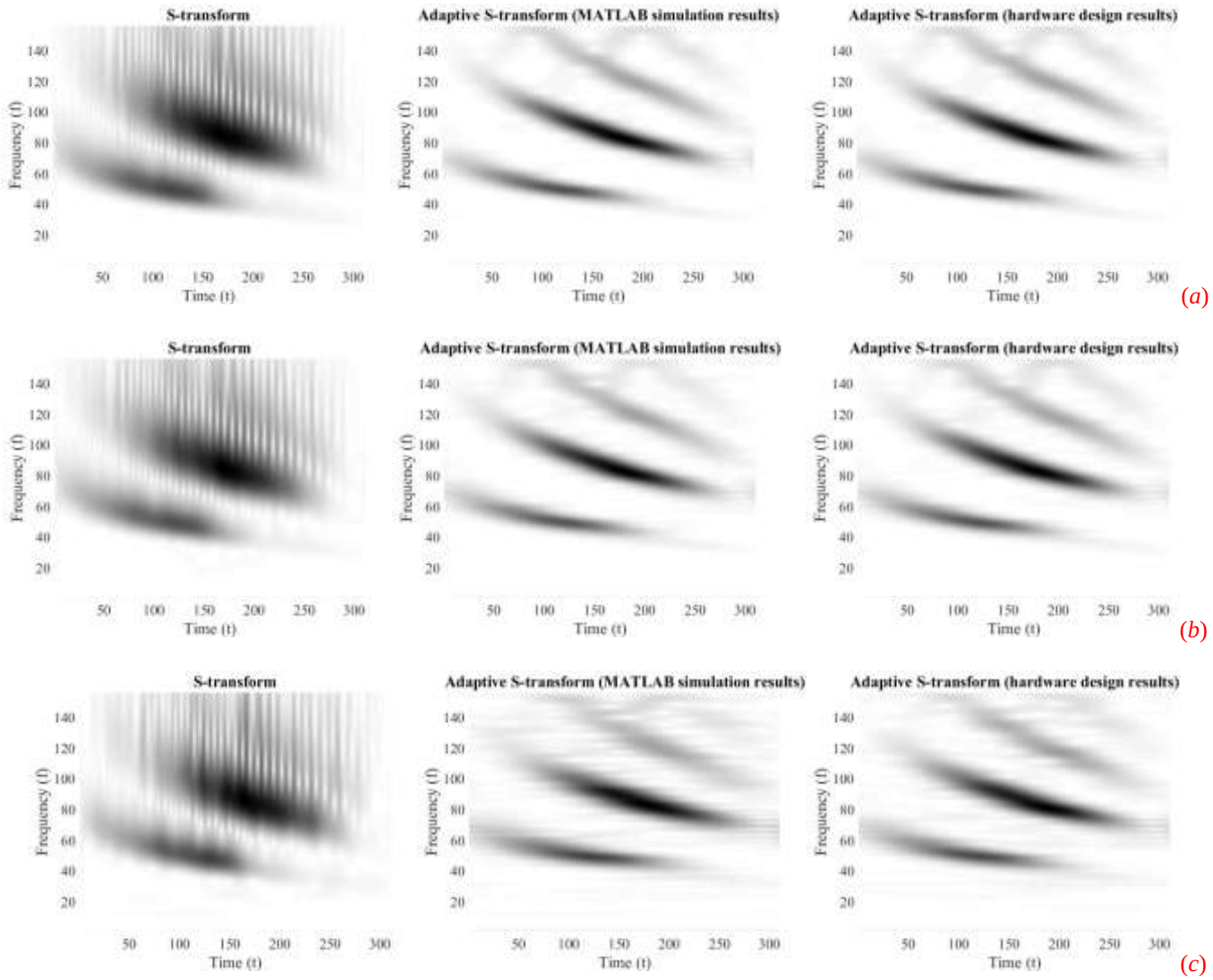


Fig. 7. (a) Noiseless test signal 2, (b) Noisy test signal 2 with SNR=10dB, (c) Noisy test signal 2 with SNR=5dB. Left-hand graphics present the standard (non-adaptive) ST, central graphics present the adaptive ST obtained by using the software simulation, whereas right-hand graphics present the adaptive ST obtained by using the developed hardware implementation.

Lastly, we will compare the computational complexities of various implementations of the S-transform. Specifically, calculating a set (*Np*) of the particular S-transforms increases the execution time of a system *Np* times in comparison to the systems that implement the single time-frequency transformation, [22–24], [33]. Hence, comparing the development proposed here with the development of single time-frequency transformations, such as the STFT, is not meaningful [22–24]. It is better to compare the developed design with the other possible designs of the adaptive S-transform. To this end, the multiple-clock-cycle implementation of the adaptive S-transform developed in this paper is compared with the other possible implementations such as the single-clock-

Table 4. The developed multiple-clock-cycle implementation design vs other possible (single-clock-cycle implementation and hybrid) designs of the adaptive S-transform regarding the hardware complexity, CLK cycle time, and the execution time. The given number of functional units corresponds to the total number of used adders, multipliers, sqrt units, division units, units for absolute value calculation, and comparators, respectively separated by square brackets, whereas the given number of memory locations corresponds to the total number of used registers and locations contained within the integrated memories used in implementations, also respectively separated by squared brackets. T_a, T_m, T_{div}, and T_{comp} are the addition, multiplication, division, and comparison times, T_{abs} and T_{sqrt} are times required by the absolute value and sqrt functional units, whereas CLK_{SCI}, CLK_{Hybrid}, and CLK_{MCI} denote CLK cycle times in the cases of single-clock-cycle implementation, Hybrid and the proposed multiple-clock-cycle implementation designs, respectively. SCI denotes the single-clock-cycle implementation, while MCI denotes the multiple-clock-cycle implementation.

Approach	Hardware complexity		CLK time	Execution time
	Functional units	Memory locations		
SCI	$[3MNp] + [Np\times(MM+7M)] +$ $+ [2MNp] + [4MNp] +$ $+ [2MNp] + [4MNp+1]$	$[MN+1] +$ $+ [2MN+Np]$	$CLK_{SCI}=2T_a+6T_m+$ $+ 2T_{sqrt}+3T_{div}+2T_{abs}+T_{comp}$	CLK_{SCI}
Hybrid	$[4M+2] + [16M] +[4] +$ $+ [6M+2] + [2M+2] + [4M+1]$	$[2MN+4] +$ $+ [2MN+Np]$	$CLK_{Hybrid}=T_a+6T_m+2T_{div}+T_{abs}$	$(Np/2)\times CLK_{Hybrid}$
Developed MCI	$[4] + [M+7] + [4] + [5] +$ $+ [3] + [5]$	$[3MN+4M+13]+$ $+ [2MN+Np]$	$CLK_{MCI}=T_a+4T_m+T_{div}$	$(2M+NM)Np\times CLK_{MCI}$

cycle and hybrid implementations. The corresponding single-clock-cycle implementation design assumes simultaneous single-clock-cycle implementations of all considered S-transforms with different p values, their comparison by a set of comparators, and, following the performed comparison, the adaptive S-transform selection. The hybrid implementation would correspond to the implementation from this paper, but only if it includes the single-clock-cycle implementation of each of two S-transforms with different p values that are compared within the process of determining the optimal p value.

The proposed system provides a one-by-one calculation of a set (Np) of the S-transforms with different parameter p values, as well as a selection of the desired S-transform based on the concentration measure criterion. The bordered part of the architecture shown in Fig. 2 provides a calculation of the S-transforms with different parameter p values. The rest of the architecture given in Figs. 2-4, managed by the appropriate control signals, provides a selection of the desired S-transform and the corresponding parameter p value based on the concentration measure criterion. This part of the architecture increases hardware complexity of the developed system for 11 functional units (2 adders, 4 sqrt units, 2 division units, 2 units for absolute value calculation, 1 comparator) and ($2MN$ + 6) memory locations. This is the difference between the architecture developed here from developments of other common time-frequency transformations like the S-method, [32], or the short-time Fourier transform [22–24]. The

comprehensive comparative analysis concerning the hardware complexity, CLK cycle time, and execution time is presented in Table 4.

As it can be expected, the developed multiple-clock-cycle implementation design optimizes hardware complexities of the considered implementations and minimizes the CLK cycle time. The single-clock-cycle implementation optimizes the execution time, but due to the complexity caused by the parameter *Np*, the single-clock-cycle implementation is not always suitable for implementation on an integrated chip. Finally, the developed multiple-clock-cycle implementation and the hybrid design provide comparable execution times. However, the developed multiple-clock-cycle implementation design uses a larger number of memory locations with respect to counterparts, but memory capacities required by the considered designs are fairly small and do not represent the critical design performance (only 25% of memory capacity is used for the proposed implementation, Table 2). Accordingly, the multiple-clock-cycle implementation of the adaptive S-transform is realized in this paper as the optimal solution among the considered designs. In addition, its complexity does not depend on *Np*. Therefore, it is always suitable for the implementation on an integrated chip. Note that the similar challenges are solved by developing multiple-clock-cycle implementations in the case of systems providing the advanced one-dimensional, [24, 26–28], and two-dimensional, [25], signal representations, or the efficient parameter estimation of nonstationary signals, [29–32].

5. DISCUSSION

Our aim to develop a hardware solution able to evaluate the adaptive S-transform is achieved. The developed solution is suitable for an on-a-chip implementation, such as the implementation on an FPGA device, and it provides improved execution times regarding the software-based solution. These improved results were demonstrated via several test examples. Our results have also demonstrated that the hardware output is almost identical to the software (simulation) one despite employing approximation of the frequency window function needed to facilitate the hardware implementation. These findings clearly demonstrate that FPGA devices provide a suitable (pseudo) real-time framework for the analysis of non-stationary signals, as such analyses have been typically carried out via offline means using personal computers.

Following recent trends in design of the signal processing systems, [24–33], the multiple-clock-cycle implementation of the proposed hardware solution that optimizes complexities of the other corresponding designs is developed in this paper. In this way, our initial hypothesis that it is possible to implement an adaptive time-frequency representation, such as the adaptive S-transform, and therefore to extend its applicability in practice is proven, i.e. the adaptive S-transform can be implemented on an inexpensive hardware device. All the challenges we have faced within the process of the hardware design, including replacement of the frequency window function with appropriate approximation, and providing evaluation of *Np* different S-transforms, but storing

only two of them (the currently evaluated and the one which currently has the best concentration measure) to save hardware resources, are resolved.

The main advantage of the proposed hardware approach is the significantly reduced execution time with respect to the software-based realization. At the same time, the proposed solution provides an ability to implement computationally demanding time-frequency representation on a simple device that can be deployed on low-cost sensor nodes and remote devices processing nonstationary and in particular frequency modulated signals. This is critical for many practical applications that depend on real-time decision support systems such as biomedical applications, speech processing, and radar processing, just to name a few.

The next step in our study could be an implementation of the S-transform with a "greater level of adaptivity", or, precisely, the hardware implementation able to produce adaptivity on a local level (for the particular time instants, or frequency range, or a part of the time-frequency plane). Such S-transform, with possibilities for time or frequency adaptivity, would enable the design to change parameter *p* in time and frequency domains and to produce optimized time-frequency representations in both domains. This opens an entirely new area of practical applications of time-frequency representations, specially optimized time-frequency representations, as a major drawback of time-frequency representations in real-life applications is their lack of adaptability. Our proposed solution provides a framework that can be adopted for any time-frequency representation. The second important development is the improvement of scalability of the hardware for time-frequency representations. Namely, a well-established recursive relationship exists to evaluate the short-time Fourier transform in the current instant based on its previous values. Similar recursive or iterative relationships are available for other time-frequency representations, as well. Such connections are also useful for the adaptive S-transform facilitating processing of long signals that are common for medical and other recordings.

6. CONCLUSION

An efficient multiple-clock-cycle implementation of the adaptive S-transform was developed, tested, and verified. The developed design was verified by implementation on an FPGA device and the achieved results were compared with the results obtained by a MATLAB simulation. It was shown that those results fully correspond to each other, which proves the high accuracy of the developed design. In terms of the execution time, which can be calculated in advance, the proposed hardware implementation significantly outperforms the software simulation. The proposed design has been compared with the single-clock-cycle and hybrid hardware implementations. It was shown that a developed design is an optimal solution among the considered implementations.

ACKNOWLEDGMENT

The authors wish to thank Curtis Condon, Ken White, and Al Feng of the Beckman Institute of the University of Illinois for the

bat data and for permission to use it in this paper.

REFERENCES

[1] S. G. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed., San Diego, USA: Academic Press, 1999.

[2] R. Stockwell, L. Mansinha, R. Lowe, "Localization of the complex spectrum: The S-transform," *IEEE Transactions on Signal Processing*, Vol. 44, No. 4, Apr. 1996, pp. 998-1001.

[3] R. G. Stockwell, *S-transform analysis of gravity wave activity from a small scale network of airglow imagers*, PhD thesis, University of Western Ontario, London, Ontario, Canada, 1999.

[4] R. G. Stockwell, "A basis for efficient representation of the S-transform," *Digital Signal Processing*, Vol. 17, No. 1, Jan. 2007, pp. 371-393.

[5] E. Sejdić, I. Djurović, J. Jiang, "Time-frequency feature representation using energy concentration: An overview of recent advances," *Digital Signal Processing*, Vol. 19, No. 1, pp. 153-183, Jan. 2009.

[6] C. Conru, I. Djurović, C. Ioana, A. Quinquis, and Lj. Stanković, "Time-frequency detection using Gabor filter bank and Viterbi based grouping algorithm," in *Proc. of IEEE ICASSP*, 2005, doi: 10.1109/ICASSP.2005.1416054.

[7] M. V. Reddy and R. Sodhi, "An Open-Loop Fundamental and Harmonic Phasor Estimator for Single-Phase Voltage Signals," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4535-4546, July 2020, doi: 10.1109/TII.2019.2950404.

[8] A. G. Rehorn, E. Sejdić, and J. Jiang, "Fault diagnosis in machine tools using selective regional correlation," *Mechanical Systems and Signal Processing*, Vol. 20, No. 5, pp. 1221-1238, July 2006.

[9] R. d. A. Coelho and N. S. D. Brito, "Power Measurement Using Stockwell Transform," *IEEE Transactions on Power Delivery*, vol. 36, no. 5, pp. 3091-3100, Oct. 2021, doi: 10.1109/TPWRD.2020.3033403.

[10] P. Kijanka and M. W. Urban, "Phase Velocity Estimation With Expanded Bandwidth in Viscoelastic Phantoms and Tissues," *IEEE Transactions on Medical Imaging*, vol. 40, no. 5, pp. 1352-1362, May 2021, doi: 10.1109/TMI.2021.3054950.

[11] S. D. Sharma, R. Saxena and S. N. Sharma, "Identification of Microsatellites in DNA Using Adaptive S-Transform," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 3, pp. 1097-1105, May 2015, doi: 10.1109/JBHI.2014.2330901.

[12] G.A. Jones, A.M.G. Ferreira, B. Kulessa, et al., "Uppermost crustal structure regulates the flow of the Greenland Ice Sheet", *Nature Communications*, vol. 12, Art. No. 7307, 2021. doi: https://doi.org/10.1038/s41467-021-27537-5.

[13] A. Moukadem, Z. Bouguila, D. O. Abdeslam, A. Dieterlen. "A new optimized Stockwell transform applied on synthetic and real non-stationary signals", *Digital Signal Processing*, vol. 46, Nov. 2015, pp 226-238.

[14] P. Liu and C. Song, "SCH: A Speed Measurement Method of Combined Hyperbolic Frequency Modulation Signals," *IEEE Access*, vol. 9, pp. 95986-95993, 2021, doi: 10.1109/ACCESS.2021.3094540.

[15] N. Mousavi, V. E. Ardestani, "Application of Hyperbolic S-transform in Environmental Gravity Investigation," *Journal of Environmental and Engineering Geophysics*, vol. 21, no. 2, 2016, pp. 47-90.

[16] N. Liu, J. Gao, B. Zhang, Q. Wang and X. Jiang, "Self-Adaptive Generalized S-Transform and Its Application in Seismic Time–Frequency Analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 10, pp. 7849-7859, Oct. 2019, doi: 10.1109/TGRS.2019.2916792.

[17] I. Djurović, E. Sejdić, J. Jiang, "Frequency based window width optimization for S-transform," *AEU – International Journal of Electronics and Communications*, Vol. 62, No. 4, pp. 245-250, Apr. 2008.

[18] E. Sejdić, I. Djurović, and J. Jiang, "A Window Width Optimized S-Transform," *Journal of Advances in Signal Processing*, Vol. 2008, Article ID 672941, Feb. 2008.

[19] R. Kamalakkannan, R. Roopkumar, "Two-dimensional fractional Stockwell transform," *Circuits, Systems, and Signal Processing*, vol. 41, pp. 1735–1750, 2022, doi: https://doi.org/10.1007/s00034-021-01858-8.

[20] B. Li, Z. Zhang, X. Zhu, "Adaptive S-transform with chirp-modulated window and its synchroextracting transform," *Circuits, Systems, and Signal Processing*, vol. 40, pp. 5654–5681, 2021, doi: https://doi.org/10.1007/s00034-021-01740-7

[21] F. Hlawatsch, G. Matz, H. Kirchauer, W. Kozek, "Time-frequency formulation, design and implementation of time-varying optimal filters for signal estimation," *IEEE Transactions on Signal Processing*, vol. 48, no. 5, May 2000, pp. 1417-1432.

[22] K. Maharatna, A.S. Dhar, S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST," *Signal Processing*, vol. 41, no. 3, 2001, pp. 1357-1377.

[23] S. Stanković, LJ. Stanković, V.N. Ivanović, R. Stojanović, "An architecture for the VLSI design of systems for time-frequency analysis and time-varying fitering," *Annales des Telecomm.*, vol. 57, no. 9/10, .Sept/Oct. 2002, pp. 974-995.

[24] V.N. Ivanović, R. Stojanović, LJ. Stanković, "Multiple clock cycle architecture for the VLSI design of a system for time-frequency analysis," *EURASIP Jou. on Applied Signal Process., Special Issue Design Methods for DSP Syst.*, 2006, pp. 1-18.

[25] V.N. Ivanović, R. Stojanović, "An efficient hardware design of the flexible 2-D system for space/spatial-frequency analysis," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, June 2007, pp. 3116-3126.

[26] N.R. Brnović, I. Djurović, V.N. Ivanović, M. Simeunović, "Hardware implementation of the quasi maximum likelihood estimator core for polynomial phase signals," *IET Circuits, Devices & Systems*, vol. 13, no. 2, March 2019, pp. 131-138.

[27] N.R. Brnović, V.N. Ivanović, I. Djurović, M. Simeunović, "Multi-core hardware realization of the QML PPS estimator," *IET Computers & Digital Techniques*, vol. 14, issue 5, Sept. 2020, pp. 187–192.

[28] V.N. Ivanović, S. Jovanovski, "A signal adaptive system for time-frequency analysis," *IET Electronics Letters*, vol. 44, no. 21, Oct. 2008, pp. 1279-1280.

[29] S. Jovanovski, V.N. Ivanović, "Signal adaptive pipelined hardware design of time-varying optimal filter for highly nonstationary FM signal estimation," *J Signal Process. Systems*, 62(3), 2011, pp. 287-300.

[30] V.N. Ivanović, N. Radović, "Signal adaptive hardware implementation of a system for highly nonstationary two-dimensional FM signal estimation," *AEU – Int. J Electronics & Communications*, vol. 69, no. 12, Dec. 2015, pp. 1854–1867.

[31] V.N. Ivanović, N.R. Brnović, "Superior execution time design of a space/spatial-frequency optimal filter for highly nonstationary 2D FM signal estimation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, issue 10, Oct. 2018, pp. 3376–3389.

[32] V.N. Ivanović, S. Jovanovski, "Advanced superior execution time optimal time-frequency filter suitable for non-linear FM signals estimation," *Microprocessors and Microsystems*, vol. 74, April 2020, pp. 1-11.

[33] N. Radović, V.N. Ivanović, I. Djurović, M. Simeunović, "System for S-transform realization," in *Proc. of 10th Mediterranean Conference on Embedded Computing (MECO 2021)*, Budva, Montenegro, June 7-10, 2021, pp. 348-351, doi: 10.1109/MECO52532.2021.9460253.

[34] LJ. Stanković, "Measuring time-frequency distributions concentration," in Time-Frequency Signal Analysis & Processing (B. Boashash, ed.), Elsevier, 2016, pp. 401-408.

[35] H. Zhang, G. Hua and Y. Xiang, "Enhanced time-frequency representation and mode decomposition," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4296-4311, 2021, doi: 10.1109/TSP.2021.3093786.

[36] B. Boashash, S. Ouelha, "An improved design of high-resolution quadratic time–frequency distributions for the analysis of nonstationary multicomponent signals using directional compact kernels," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2701–2713, 2017.

[37] V. V. Moca, H. Bârzan, A. Nagy-Dăbâcan, R. C. Mureșan, "Time-frequency super-resolution with superlets", *Nature Communications Science*, vol. 12, Art. no. 337, 2021, doi: https://doi.org/10.1038/s41467-020-20539-9.

[38] L. P. A. Arts, E. L. van den Broek, "The fast continuous wavelet transformation (fCWT) for real-time, high-quality, noise-resistant time–frequency analysis," *Nature Communications*, vol. 2, pp. 47–58, 2022, doi: https://doi.org/10.1038/s43588-021-00183-z.

[39] M. Mazelanik, A. Leszczyński, M. Parniak, "Optical-domain spectral super-resolution via a quantum-memory-based time-frequency processor", *Nature Communications*, vol. 13, Art. no. 691, 2022, doi: https://doi.org/10.1038/s41467-022-28066-5.

[40] M.S.E. Abadi, H. Mesgarani, S.M. Khademiyan, "Two improved wavelet transform domain LMS sign adaptive filter algorithms against impulsive interferences," *Circuits, Systems, and Signal Processing*, vol. 40, pp. 958–979, 2021, doi: https://doi.org/10.1007/s00034-020-01508-5

[41] R.N. Vargas, A.C.P. Veiga, "Empirical mode decomposition, Viterbi and wavelets applied to electrocardiogram noise removal," *Circuits, Systems, and Signal Processing*, vol. 40, pp. 691–718, 2021, doi: https://doi.org/10.1007/s00034-020-01489-5

Data availability statement: This manuscript has no associated data.