Hindawi Publishing Corporation
EURASIP Journal on Applied Signal Processing
Volume 2006, Article ID 60613, Pages 1–18
DOI 10.1155/ASP/2006/60613

Multiple-Clock-Cycle Architecture for the VLSI Design of a System for Time-Frequency Analysis

Veselin N. Ivanović, Radovan Stojanović, and LJubiša Stanković

Department of Electrical Engineering, University of Montenegro, 81000 Podgorica, Montenegro, Yugoslavia

Received 29 September 2004; Revised 17 March 2005; Accepted 25 May 2005

Multiple-clock-cycle implementation (MCI) of a flexible system for time-frequency (TF) signal analysis is presented. Some very important and frequently used time-frequency distributions (TFDs) can be realized by using the proposed architecture: (i) the spectrogram (SPEC) and the pseudo-Wigner distribution (WD), as the oldest and the most important tools used in TF signal analysis; (ii) the S-method (SM) with various convolution window widths, as intensively used reduced interference TFD. This architecture is based on the short-time Fourier transformation (STFT) realization in the first clock cycle. It allows the mentioned TFDs to take different numbers of clock cycles and to share functional units within their execution. These abilities represent the major advantages of multicycle design and they help reduce both hardware complexity and cost. The designed hardware is suitable for a wide range of applications, because it allows sharing in simultaneous realizations of the higher-order TFDs. Also, it can be accommodated for the implementation of the SM with signal-dependent convolution window width. In order to verify the results on real devices, proposed architecture has been implemented with a field programmable gate array (FPGA) chips. Also, at the implementation (silicon) level, it has been compared with the single-cycle implementation (SCI) architecture.

1. INTRODUCTION AND PROBLEM FORMULATION

The most important and commonly used methods in TF signal analysis, the SPEC and the WD, show serious drawbacks: low concentration in the TF plane and generation of cross-terms in the case of multicomponent signal analysis, respectively, [1–3]. In order to alleviate (or in some cases completely solve) the above problems, the SM for TF analysis is proposed in [4]. Recently, the SM has been intesively used, [5–8]. Its definition is [4, 9, 10]

$$\mathrm{SM}(n,k) = \sum_{i=-L_d(n,k)}^{L_d(n,k)} P_{(n,k)}(i)\,\mathrm{STFT}(n,k+i)\,\mathrm{STFT}^*(n,k-i), \tag{1}$$

where $\mathrm{STFT}(n,k)=\sum_{i=-N/2+1}^{N/2} f(n+i)w(i)e^{-j(2\pi/N)ik}$ represents the STFT of the analyzed signal $f(n)$, $2L_d(n,k)+1$ is the width of a finite frequency domain (convolution) rectangular window $P_{(n,k)}(i)$ ($P_{(n,k)}(i)=0$, for $|i|>L_d(n,k)$), and the signal's duration is $N=2^m$. The SM produces, as its marginal cases, the WD and the SPEC with maximal ($L_d(n,k)=N/2$), and minimal ($L_d(n,k)=0$) convolution window width, respectively. In the case of a multicomponent signal with nonoverlapping components, by an appropriate convolution window width selection, the SM can produce a sum of the WDs of individual signal components, avoiding cross-terms [4, 10, 11]: $P_{(n,k)}(i)$ should be wide enough to enable complete integration over the auto-terms, but narrower than the distance between two auto-terms. In addition, the SM produces better results than the SPEC and the WD, regarding calculation complexity [4] and noise influence [9]. Note that the essential SM properties are: the high auto-terms concentration, the cross-terms reduction and the noise influence suppression.

Two possibilities for the SM (1) implementation are

(1) with a signal-independent (constant) $L_d(n,k)$, $L_d(n,k)=L_d=\text{const}$, [4, 10], when, in order to get the WD for each component, the convolution window width should be such that $2L_d+1$ is equal to the width of the widest auto-term. For the entire TF plane, except at the central points of the widest component, this window would be too long. This fact might have negative effects regarding cross-terms reduction, [4, 10] and the noise influence suppression, [9]. On the other hand, the shorter window would result in lower concentration;
(2) with a signal-dependent $L_d(n,k)$ (the so-called signal-dependent SM) [11], which may alleviate the

disadvantages of the signal-independent form in the analysis of multicomponent signals having different widths of the auto-terms. In addition, it may further significantly improve the essential SM properties, [9, 11].

In order to improve concentration of highly nonstationary signals, higher-order TFDs can be used [5, 12]. One of them, which can be presented in a two-dimensional TF plane and defined in the same manner as the SM, is the L-Wigner distribution (LWD) [12]:

$$\mathrm{LWD}_L(n,k) = \sum_{i=-L_d}^{L_d} \mathrm{LWD}_{L/2}(n,k+i)\,\mathrm{LWD}_{L/2}(n,k-i), \tag{2}$$

where $\mathrm{LWD}_L(n,k)$ is the LWD of the Lth order, and $\mathrm{LWD}_1(n,k) \equiv \mathrm{SM}(n,k)$. Note that the LWD is implicitly defined based on the SM and the STFT, so it can be implemented in a similar way as the SM.

Definition (1), based on STFT, makes the SM very attractive for implementation. However, all TFDs, beyond the STFT, are numerically quite complex and require significant calculation time. This fact makes them unsuitable for real-time analysis, and severely restricts their application. Hardware implementations, when they are possible, can overcome this problem and enable application of these methods in numerous additional problems in practice. Some simple implementations of the architectures for TF analysis are presented in [10, 13–19]. An architecture for VLSI design of systems for TF analysis and time-varying filtering based on the SM is presented in [16, 17]. However, all these architectures give the desired TFD in one clock cycle. It means that no architecture resource can be used more than once, and that any element needed more than once must be duplicated. Consequently, practical realization of these architectures requires large chips. Besides, just a single TFD—SM with exactly defined convolution window width—can be realized this way.

In this paper, we develop an MCI of a special purpose hardware for TF analysis based on the SM, suitable for the VLSI design. In the proposed implementation, each step in the TFDs execution will take one clock cycle. In the first step, proposed architecture realizes the STFT, as a key intermediate step in realization of the implemented TFDs. In each higher-order clock cycle, different TFD is realized: in the second one—the SPEC, in the third one—the SM with unitary convolution window width, and so on. The WD is realized in the clock cycle when the maximal convolution window width is reached. Note that proposed architecture can realize almost all commonly used TFDs. The MCI design allows a functional unit to be used more than once per TFDs execution, as long as it is used on different clock cycles. This significantly reduces the amount of the required hardware. The ability to allow TFDs to take different number of clock cycles and the ability to share functional units within the execution of a single TFD are the major advantages of the proposed design.

The paper is organized as follows. After the introduction, MCI architectures for the SM realization (in its signal-independent and signal-dependent forms) are designed, the corresponding controls are defined, and the trade-offs and comparisons with the SCI are given. In Section 3, the designed MCI system is used for the real-time realization of the higher-order TFDs. The proposed approaches are verified in Section 4 by designing the FPGA chips. Also, the obtained implementation results at silicon level are compared with SCI architectures.

2. MULTICYCLE HARDWARE IMPLEMENTATION OF THE S-METHOD

2.1. *Signal-independent S-method*

In this section, an MCI system for SM (1) realization, assuming fixed convolution window width ($L_d(n,k) = L_d$), is presented. Since the STFT is a complex transformation, (1) involves complex multiplications. In order to involve only real multiplications in (1), we modify it by using $\mathrm{STFT}(n,k) = \mathrm{STFT}_{\mathrm{Re}}(n,k) + j\,\mathrm{STFT}_{\mathrm{Im}}(n,k)$ ($\mathrm{STFT}_{\mathrm{Re}}(n,k)$ and $\mathrm{STFT}_{\mathrm{Im}}(n,k)$ are the real and imaginary parts of $\mathrm{STFT}(n,k)$, resp.), as

$$\mathrm{SM}_R(n,k) = \mathrm{STFT}^2_{\mathrm{Re}}(n,k) + 2\sum_{i=1}^{L_d} \mathrm{STFT}_{\mathrm{Re}}(n,k+i)\,\mathrm{STFT}_{\mathrm{Re}}(n,k-i), \tag{3}$$

$$\mathrm{SM}_I(n,k) = \mathrm{STFT}^2_{\mathrm{Im}}(n,k) + 2\sum_{i=1}^{L_d} \mathrm{STFT}_{\mathrm{Im}}(n,k+i)\,\mathrm{STFT}_{\mathrm{Im}}(n,k-i), \tag{4}$$

where $\mathrm{SM}(n,k) = \mathrm{SM}_R(n,k) + \mathrm{SM}_I(n,k)$. The kth channel, one of the N channels (obtained for $k = 0, 1, \ldots, N-1$), is described by (3)-(4). Note that it will consist of two identical sub-channels used for processing of $\mathrm{STFT}_{\mathrm{Re}}(n,k)$ and $\mathrm{STFT}_{\mathrm{Im}}(n,k)$, respectively.

The hardware necessary for one channel MCI of the signal-independent SM is presented in Figure 1. It is designed based on a two-block structure. The first block is used for the STFT implementation, whereas the second block is used to modify the outputs of the STFT block, in order to obtain the improved TFD concentration based on the SM. The STFT block can be implemented by using the available FFT chips [20, 21] or by using approaches based on the recursive algorithm [10, 13, 17, 19, 22–24]. Note that, due to the reduced hardware complexity, the recursive algorithm is more suitable for a VLSI implementation, [13]. The second block is designed so that it realizes each summation term from (3)-(4) in the corresponding step of the method implementation.

We break the SM execution into several steps, each taking one clock cycle. Our goal in breaking the SM execution into clock cycles is to balance the amount of work done in each cycle, so that we minimize the clock cycle time. In the first step, the STFT will be executed, in the second step, the SPEC will be executed based on the first step execution, in the third step, the SM with the unitary convolution window width will

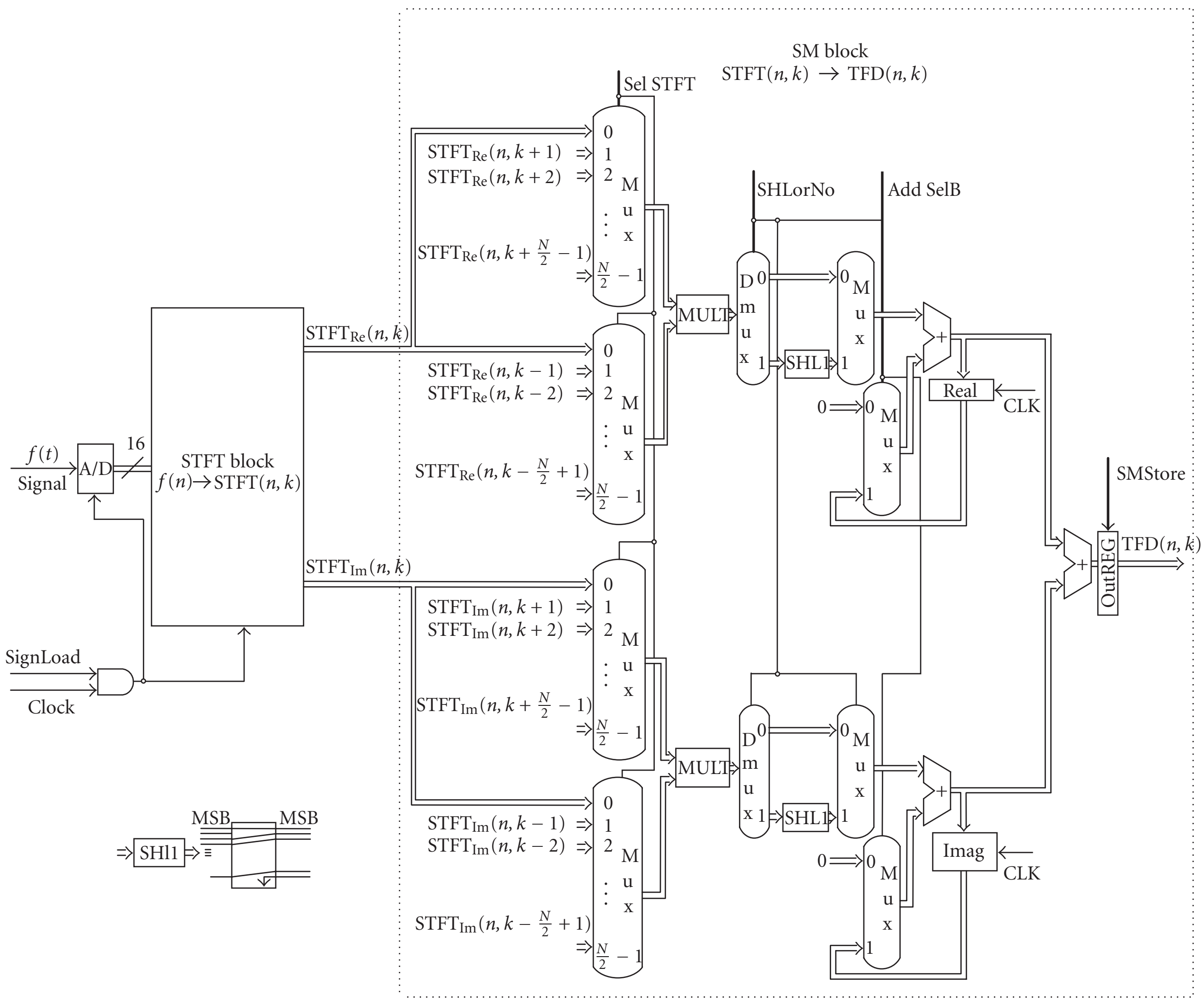

FIGURE 1: MCI architecture for the signal-independent S-method realization.

be executed based on the execution in the first two steps, and so on. With each further step, one realizes the SM with the incremented width of convolution window, based on the preceding steps. This improves the TFD concentration, aiming at achieving the one obtained by the WD.

Proposed hardware has been designed for a 16-bit fixed-point arithmetic. Each subchannel of the second block contains exactly one adder, one multiplier, and one shift left register for implementation of (3)-(4). These functional units must be shared for different inputs in different steps by adding multiplexors and/or a demultiplexor at their inputs. Real and imaginary parts of the SM value, computed in each execution step and based on (3)-(4), are saved into the *Real* and *Imag* temporary registers, respectively. In the first step, only the STFT block of the proposed two-block architecture is used, whereas in the remaining steps only the second block is used. This will be regulated by the set of control signals introduced on temporary registers, and multiplexors and a demultiplexor, see Table 1. Note that control signals *SHLorNo* and *AddSelB* assume unity values in each step of the TFD implementation, except in the second step (SPEC completion step), when they assume zero values. Consequently, these signals can be replaced by one control signal *SPECorSM* that enables the SPEC execution (with its zero value), or execution of the TFDs with the nonzero convolution window widths. Note that the multiplication operation results in a two sign-bit and, assuming Q15 format (15 fractional bit), the product must be shifted left by one bit to obtain correct results. This shifter is included in the multiplier.

The longest path in the second block is one that connects the inputs $\mathrm{STFT}_{\mathrm{Re}}(n,k)$ (or $\mathrm{STFT}_{\mathrm{Im}}(n,k)$), through one multiplier, one shift left register, and 2 adders, with the output of the second block. If the STFT is realized based on a recursive algorithm, than it has the same longest path, [10, 17]. This path determines the clock cycle time and then the fastest sampling rate. This design can be implemented as

TABLE 1: Function of each of the control signal generated by the control logic.

Control signal	Effect
SelSTFT	$(m-1)$-bit signal which controls $N/2$-input multiplexors (two of them per subchannel are introduced to select between the STFT values from different channels)
SHLorNo	1-bit signal which enables use of the shift-left register in the corresponding steps (when we need to implement multiplication by 2), or disables this (in the second step)
AddSelB	1-bit signal which enables use of only one adder per subchannel for implementing sums in (3)-(4) by controlling its second input, which can be either the constant 0 (in the second step) or a register *Real* (or *Imag*) value (in each further step)
SignLoad	1-bit signal which enables sampling of the analyzed analog signal $f(t)$, but only after execution of the desired TFD of the analyzed signal samples from the preceding time instant
SMStore	1-bit write control signal of the *OutREG* temporary register. It should be asserted during the step in which the SM with corresponding convolution window width is computed

an application specified integral circuits (ASIC) chip to meet the speed and performance demands of very fast real-time applications, see Section 4.

Defining the control

From the defined multistep sequence of the multicycle TFDs execution, we can determine what control logic must do at each clock cycle. It can set all control signals, based solely on the distribution code (TFDcode). This code determines TFD which will be implemented by using the proposed architecture. Taking $N = 64$, the TFDcode can be a 6-bit field which determines the convolution window width. An architecture with the control logic and the control signals are shown in Figure 2.

Control for the MCI architecture must specify both the signals to be set in any step and the next step in the sequence. Here we use finite-state Moore machine to specify the multicycle control, Figure 3. Finite-state control essentially corresponds to the steps of desired TFD execution; each state in the finite-state machine will take one clock cycle. This machine consists of a set of states and directions on how to change states. Each state specifies a set of outputs that are asserted or deasserted when the machine is in that particular state. The labels on the arc are conditions that are tested to determine which state is the next one. When the next state is unconditional, no label is given. Note that implementation of a finite-state machine usually assumes that all outputs that are not explicitly asserted are deasserted, and the correct operation of the architecture often depends on the fact that a signal is deasserted. Multiplexors and demultiplexor controls are slightly different, since they select one of the inputs, whether they be 0 or 1. Thus, in the finite-state machine we always specify the settings of all (de)multiplexor controls that we care about.

2.2. Trade-offs and comparisons of the proposed design with the SCI ones

SCI architecture executes desired TFD in one clock cycle. This means that no architecture resource can be used more than once per TFD execution and that any element needed more than once must be duplicated. Then, we can easily conclude that in the case of the considered SM block (3)-(4) implementation we have to use $(2L_d + 1)$ adders, $2(L_d + 1)$ multipliers, and $2L_d$ shift left registers, if we prefer an SCI approach. This can be tested by studying the SCI architectures represented in [16, 17], as well as real-time SCI of the SM with $L_d = 3$ given in Section 4.2.

Comparison of the architectures' resources used in the SCI and MCI designs, as well as comparison of their clock cycle times are given in Table 2. The following advantages of the MCI design, compared with the SCI ones, can be noted:

(i) required reduction of the amount of hardware, achieved by introducing the temporary registers and several multiplexors at the inputs of the functional units. The achieved hardware reduction is significant, and it increases as the convolution window width increases;
(ii) since temporary registers and the introduced multiplexors are fairly small, this could yield a substantial reduction in the hardware cost, as well as in the used chip dimensions;
(iii) the clock cycle time in the MCI design is much shorter.

Finally, the ability to realize almost all commonly used TFDs by the same hardware represents a major advantage of the proposed MCI design.

On the other hand, the fastest sampling rate in the MCI design of the SM with arbitrary L_d is $(L_d+2)\times(T_m+2T_a+T_s)$, see Table 2, while it is equal to the clock cycle time in the corresponding SCI design ($2T_m + (L_d + 3)T_a + T_s$, see Table 2). Then, the SCI approach improves execution time. However, this disadvantage of the MCI approach is significantly alleviated by the fact that the SM with small L_d is usually used,[1] when the execution times in these two cases (the SCI and the MCI approaches) do not differ significantly.

[1] High TFD concentration (almost as high as in the WD case) is achieved even with small L_d [4, 9], whereas the interference effects [10] and the noise influence [9] are more reduced with decreasing of the convolution window width.

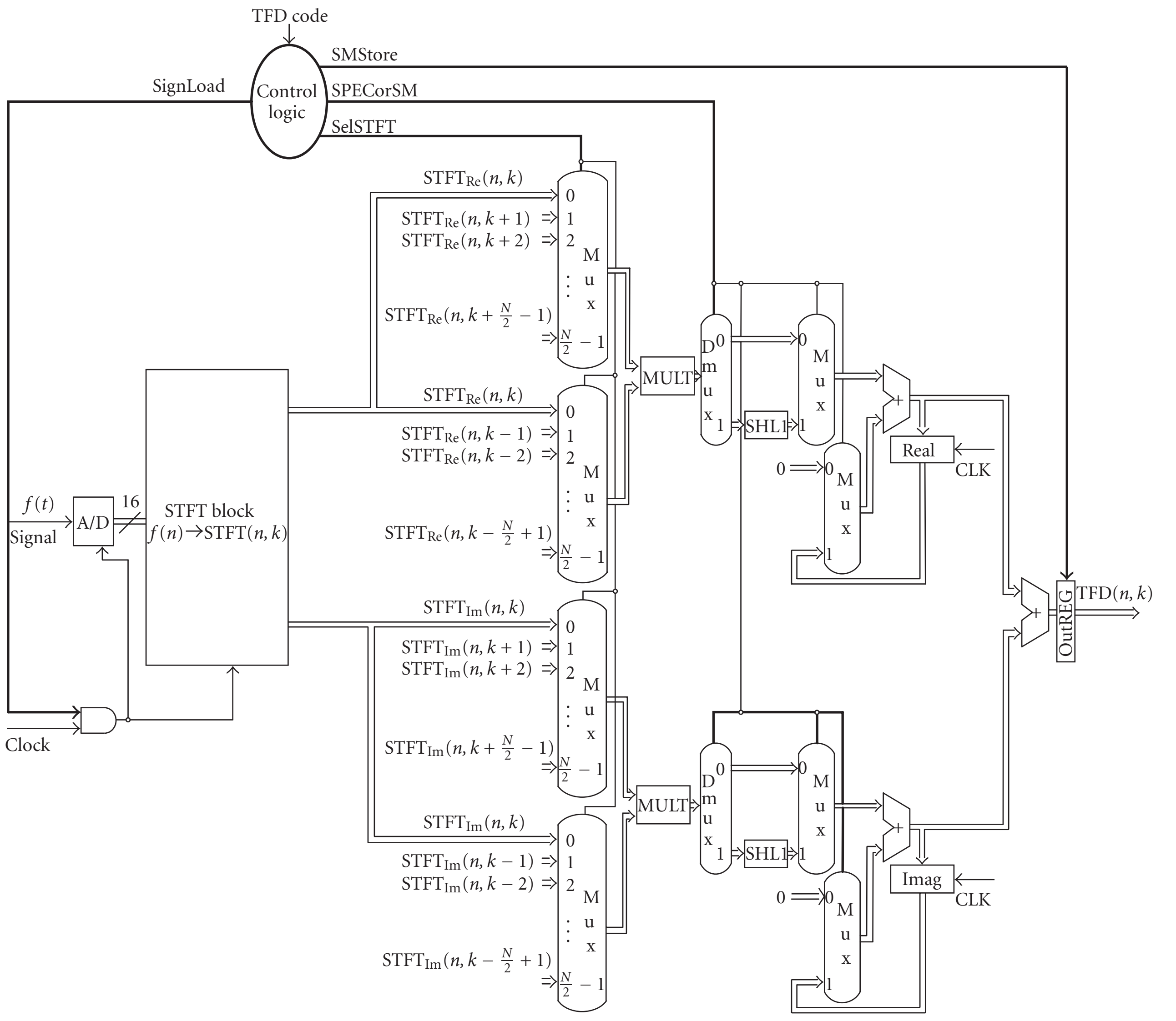
TFD code
Control logic
SMStore
SPECorSM
SelSTFT
SignLoad
$f(t)$
Signal
A/D
16
STFT block
$f(n)\rightarrow$STFT(n,k)
Clock
$\mathrm{STFT_{Re}}(n,k)$
$\mathrm{STFT_{Re}}(n,k+1)$
$\mathrm{STFT_{Re}}(n,k+2)$
$\mathrm{STFT_{Re}}(n,k+\frac{N}{2}-1)$
$\mathrm{STFT_{Re}}(n,k-1)$
$\mathrm{STFT_{Re}}(n,k-2)$
$\mathrm{STFT_{Re}}(n,k-\frac{N}{2}+1)$
$\mathrm{STFT_{Im}}(n,k)$
$\mathrm{STFT_{Im}}(n,k+1)$
$\mathrm{STFT_{Im}}(n,k+2)$
$\mathrm{STFT_{Im}}(n,k+\frac{N}{2}-1)$
$\mathrm{STFT_{Im}}(n,k-1)$
$\mathrm{STFT_{Im}}(n,k-2)$
$\mathrm{STFT_{Im}}(n,k-\frac{N}{2}+1)$
Mux
MULT
Dmux
SHL1
Real
Imag
CLK
OutREG
TFD(n,k)

Figure 2: MCI architecture for the signal-independent S-method realization together with the necessary control lines. Thick solid line highlights the control line as opposed to a line that carries data.

More technical details about practical implementation of the MCI and the SCI architectures can be found in Section 4.

Hybrid implementation

In order to achieve a balance between minimal chip dimensions, hardware consumption and cost from the MCI approach and minimal execution time from the SCI approach, the hybrid implementation approach may be considered. The SM block of this implementation would be based on the SCI design of the SM with exactly defined convolution window width L_d ($L_d \geq 1$). As in the MCI design case, hybrid implementation would give the desired TFD in a few clock cycles: in the second one this architecture could implement the SMs with convolution window widths up to the L_d (up to the SM that is a base for the SM block realization) and in each further step it could realize the SM with the incremental convolution window width. Then, total number of clock cycles would not be greater than the one from the MCI design. In particular, both implementation approaches, hybrid and MCI, use the same number (two) of clock cycles for the SPEC implementation only. In the case of the SM with nonzero convolution window width implementation, total number of clock cycles would be smaller by using hybrid implementation design.

For the SM block implementation one would use $(2L_d+1)$ adders, $2(L_d+1)$ multipliers, and $2L_d$ shift left registers, and the corresponding clock cycle time would be $T_m+(L_d+1)T_a+T_s$. Note that the hybrid implementation (even the one based on the SM with $L_d=1$) increases hardware complexity, chip dimensions, and cost, as well as the clock cycle time from the MCI design. Then, the SM with $L_d=1$ cannot be so useful as a base for the SM block of hybrid

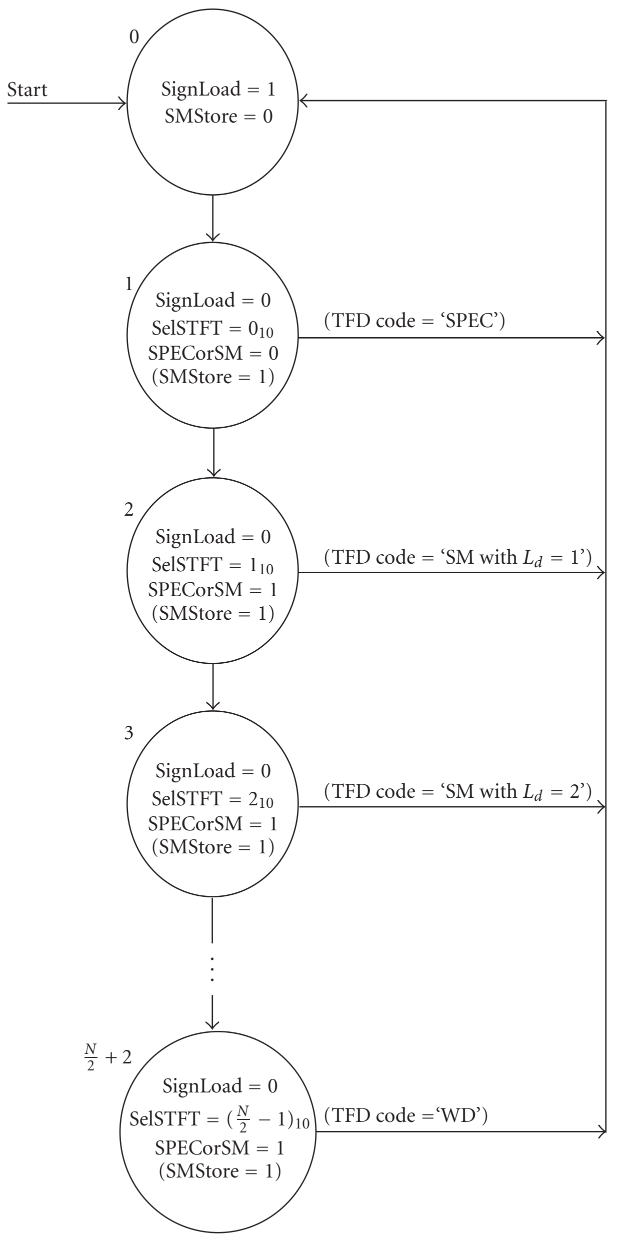

Figure 3: The finite-state machine control for the architecture shown in Figure 2. Output (SMStore = 1) means that the *SMStore* control signal is asserted during only the final step of the corresponding TFD execution.

implementation, since it would only slightly improve the execution time from MCI architecture (it requires only one step—SPEC completion—less than the MCI approach). The SM with $L_d = 2$ would be a reasonable choice for this purpose. However, the hybrid approach would not use the whole SM block in each step. For example, part of the SM block for SPEC implementation (see Figure 12 from Section 4.2) would be used in the second step only. Note that the clock cycle time is determined by the longest possible path in the SM block, which does not have to be used in any step here. Consequently, hybrid architecture could not succeed to balance the amount of work done in each clock cycle, so that we could not minimize the clock cycle time.

Note that the overall performance of the hybrid implementation is not likely to be very high, since all the steps (except, in some cases, the second one) could fit in a shorter clock cycle. The second step is an exception when the SM with convolution window width of at least L_d is implemented by using hybrid design, where L_d is the convolution window width of the SM that is a base for this particular implementation. This fact leads to the dispersion of the hardware resources as well as needed time in almost all steps used in TFD execution. Also, control logic of the hybrid implementation would be similar but, at the same time, more complicated, as compared to the MCI approach case.

Table 2: Total number of functional units per channel in an SM block and the clock cycle time in the cases of (a) single-cycle implementation (SCI) and (b) the multicycle implementation (MCI). T_m is the multiplication time of a two-input 16-bit multiplier, T_a is the addition time of a two-input 16-bit adder, whereas T_s is the time for 1-bit shift. The recursive form of the STFT block implementation is assumed when the clock cycle time in the SCI case is represented.

Implementation	Adders	Multipliers	Shift left registers	Clock cycle time
SCI	$2L_d + 1$	$2(L_d + 1)$	$2L_d$	$2T_m + (L_d + 3)T_a + T_s$
MCI	3	2	2	$T_m + 2T_a + T_s$

2.3. Signal-dependent S-method

Disadvantages of the signal-independent convolution window in the analysis of multicomponent signals, having different widths of the auto-terms, motivates the introduction of a signal-dependent convolution window width. It follows, for each point of TF plane, the widths of the auto-terms excluding the summation in (1) where one or both of the components $\mathrm{STFT}(n, k + i)$ and $\mathrm{STFT}(n, k - i)$ are equal to zero. In addition, it should stop the summation outside a component. Practically, it means that when the absolute square value of $\mathrm{STFT}(n, k + i)$ or $\mathrm{STFT}(n, k - i)$ is smaller than an assumed reference level R_n, the summation in (1) should be stopped. In practice, reference value is selected based on a simple analysis of the analyzed signal and the implementation system [10, 17]. It is defined as a few percent of the SPEC's maximal value at a considered time-instant n, $R_n^2 = \max_k\{\mathrm{SPEC}(n, k)\}/Q^2$, where $\mathrm{SPEC}(n, k)$ is the SPEC of analyzed signal and $1 \le Q < \infty$. In the sequel, the signals that determine nonzero values of $\mathrm{STFT}(n, k \pm i)$ $(i = 0, 1, \ldots, L_d(n, k))$ will be denoted by $x_{\pm i}$: $x_{\pm i} = 1$ if $|\,\mathrm{STFT}(n, k \pm i)|^2 > R_n^2$, and $x_{\pm i} = 0$ otherwise.

Sampling rate of the analyzed analog signal $f(t)$ depends on the clock cycle time T_c and on the number of the executed steps. Consequently, the same number of steps in different time instants must be executed. In that sense, we have to assume maximal possible convolution window width as $2L_{d\max} + 1$ (variable convolution window width approach with the predefined maximal window width), and to define sampling rate by $(2L_{d\max} + 1)T_c$. Since the $\mathrm{SM}(n, k)$ value is calculated in the Lth step, where $L \le L_{d\max} + 1$, it must be saved up to the $(L_{d\max} + 1)$th step into the *OutREG* temporary register.

In order to accommodate hardware from Figures 1 and 2 for signal-dependent window width, we add two $N/2$-input multiplexors to generate *SignDep*(endent) control signal, which determines whether or not the ith term enters the summation in (3)-(4). With the zero value of the *SignDep* control signal, adding the new term to the calculated SM value is disabled, since the additional improvement of the TFD concentration is impossible. It takes different values in different steps defined as

$$SignDep = x_i \cdot x_{-i}, \quad i = 0, 1, 2, \ldots, L_{d\max}. \tag{5}$$

Signals x_i are set in the first step after the STFT calculation. The circuit needed to generate signal x_i is separated within the dashed box and presented in Figure 4.

Multistep sequence of the signal-dependent SM is the same as in the signal-independent case. Two first steps have to be executed, since SPEC value should be forwarded to the output anyway. Namely, even if $|\,\mathrm{STFT}(n, k)|^2 \le R_n^2$, for all k, that is $x_0 = 0$, (practically, these are points (n, k) with no signal) the convolution window width takes zero value, and then the SM takes its marginal form—SPEC [4, 9]. Execution of the second step is provided by setting the unit value instead of x_0 to the first respective inputs of the $N/2$-input multiplexors, so $SignDep \equiv 1$ in the second—SPEC completion step.

Defining the control

Control logic for the MCI realization of the signal-dependent SM can set all but one of the control signals, based solely on the SM enable code (SM_en). Write control signal of the *OutREG* temporary register is the exception. To generate it, we will need to AND together an *SMStoreCond* signal from the control unit, with *SignDep* control signal. The finite-state Moore machine that specifies the multicycle control is presented in Figure 5.

3. MULTICYCLE HARDWARE IMPLEMENTATION OF THE HIGHER-ORDER TFDS

Since the LWD is defined in the same manner as the SM (see the LWD definition (2) and the SM definition (1)), it may be realized by using the same hardware presented in Figures 1 and 2. For that purpose, the SM block of the proposed architecture and the second input of the output adder in the SM block must be shared (by introducing two-input multiplexors) for realization of the LWD with $L = 2$, Figure 6. This must be done since only one subchannel of the SM block is used when the SM block realizes the LWD, [25]. Namely, in that case the SM block always processes the real function $\mathrm{SM}(n, k)$. The function of the proposed hardware is determined by the *SMorLWD* control signal: the SM implementation and the LWD implementation are determined by the *SMorLWD* zero and unit value, respectively, see Figure 7. Note that the *OutREG* temporary register is used for saving the computed SM value when we need to use the SM block for the LWD implementation.

Then, the control logic defined in Section 2 must be expanded with the *SMorLWD* control signal. In the first $L_d + 2$ clock cycles, system realizes $\mathrm{SM}(n, k)$. The calculated SM value, saved in the *OutREG* register, will be used in the next $L_d + 1$ clock cycles, when the LWD with $L = 2$ will be realized. It is done by asserting the *SMorLWD* control

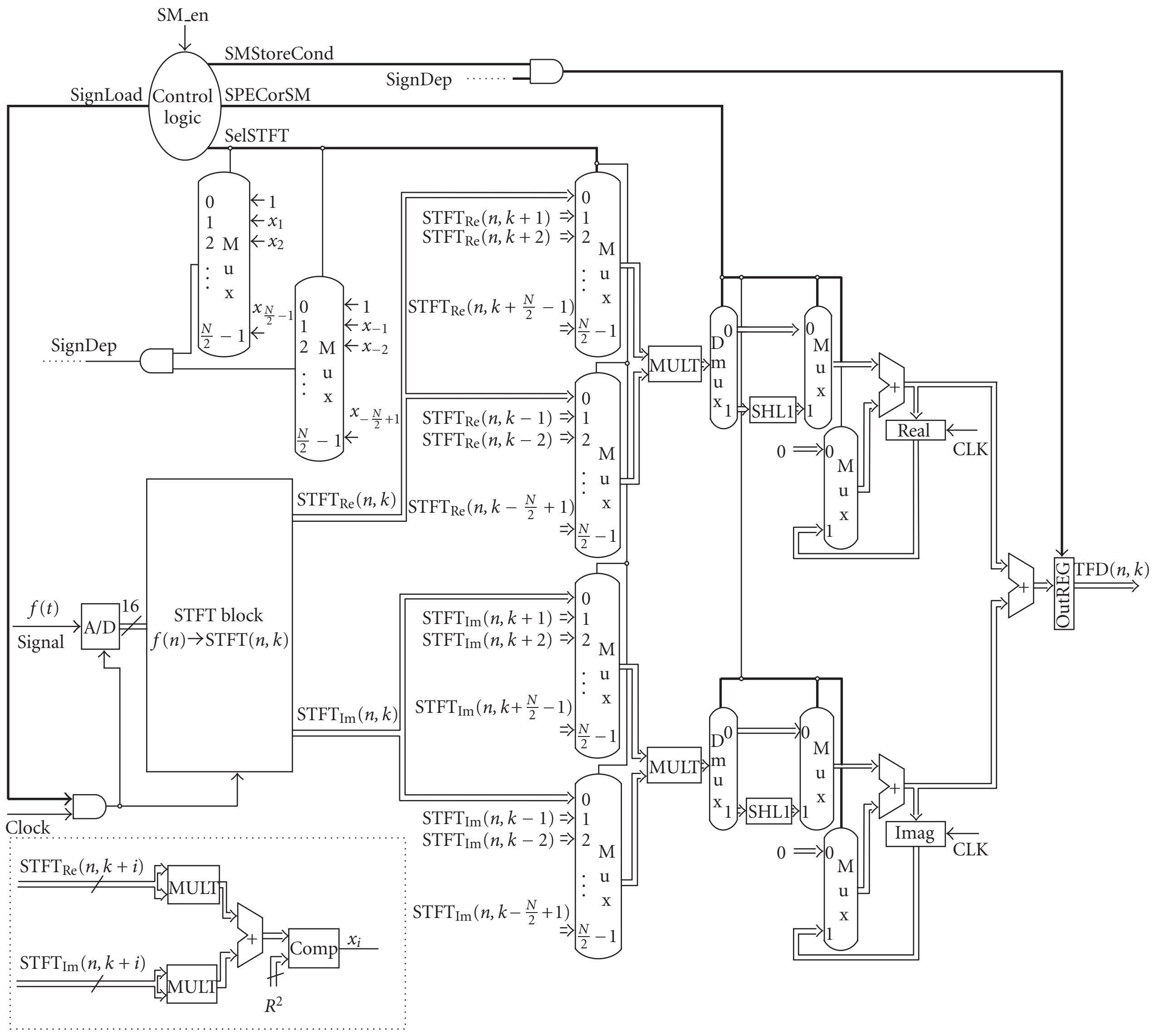

FIGURE 4: MCI architecture for the signal-dependent S-method realization.

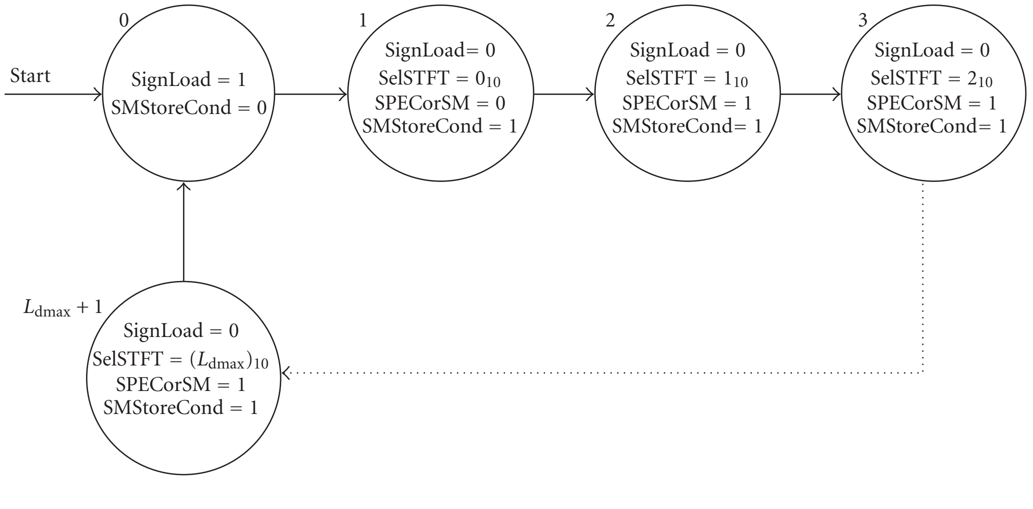

FIGURE 5: The finite-state machine control for the MCI design of the signal-dependent S-method from Figure 4.

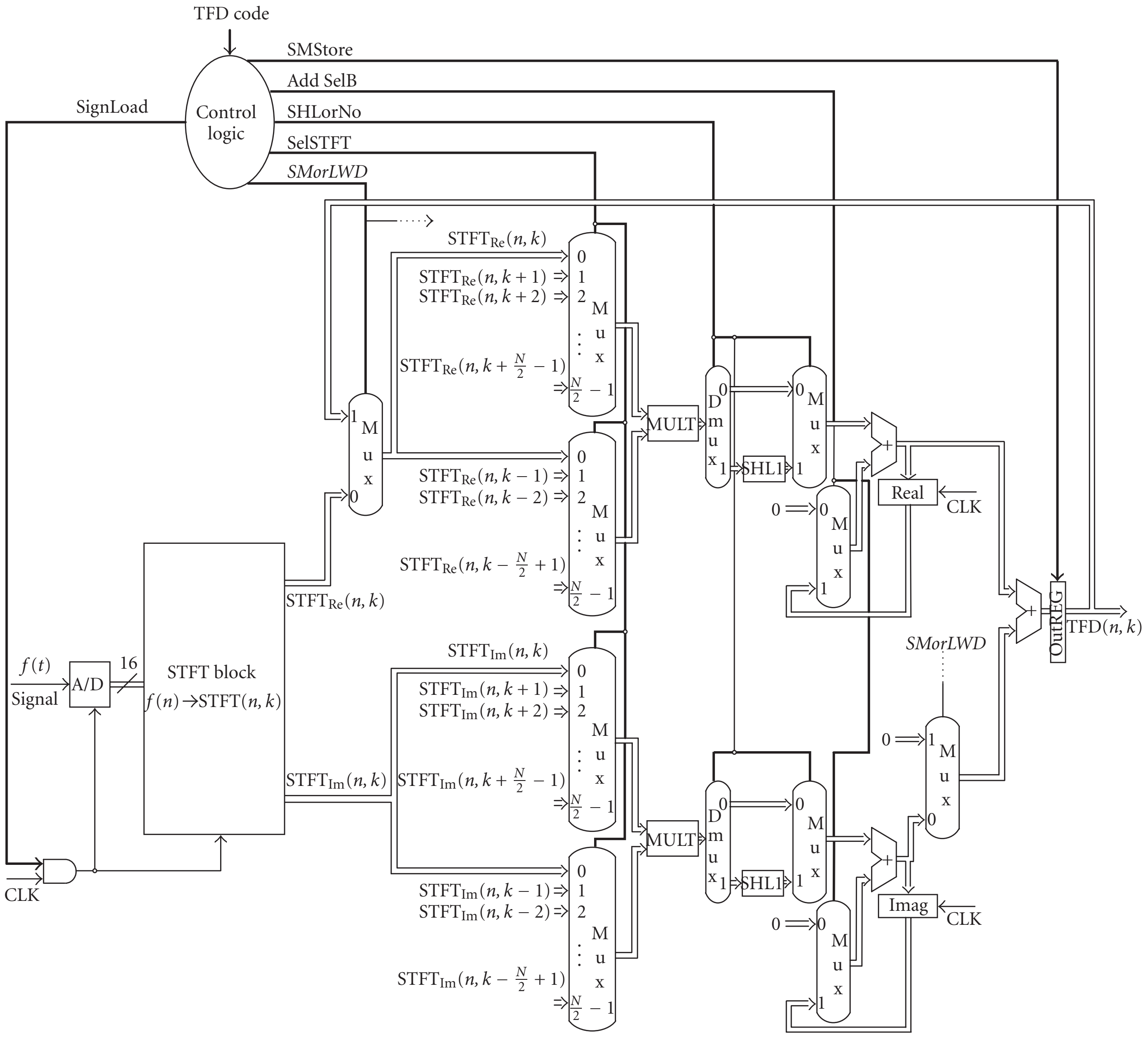

Figure 6: A complete hardware for one channel simultaneous realization of the S-method/L-Wigner distribution.

signal. The finite-state machine control for this system is shown in Figure 7. If we repeat the last $L_d + 1$ steps from Figure 7 (i.e., steps $L_d + 2$ to $2L_d + 2$), together with asserting of the *SMStore* control signal in the $(2L_d + 2)$th step, the LWD with $L = 4$ is implemented by using the proposed architecture.

Here we do not analyze the finite register length influence on the accuracy of the results obtained by the proposed architecture. Its rigorous treatment may be found in [26]. Also, for the numerical illustration we refer the readers to the papers where the theoretical approach for the methods used in this paper is given, [4, 9, 10, 12, 16].

4. PRACTICAL IMPLEMENTATION APPROACH

The architectures for the SM calculation from the STFT samples can be practically realized by using different technologies such as PC- or DSP-based solutions, running special software, or applying specified chips in forms of ASICs or programmable devices (PDs). The first way is not so useful for real-time processing, since it is mostly based on the Von Neumann architecture that significantly reduces the speed performances. Otherwise, a great degree of parallelism at high speed, as well as low power consumption, can be achieved with the chip-based solutions. Using the FPGA chips instead of classical ASICs has numerous advantages, especially in prototype development. Some of them are: (i) reasonable cost for small number of pieces, (ii) in system programming (ISP) possibilities, (iii) availability of software design support provided by different development systems for Windows-based PCs and workstations, and (iv) the developed FPGA's cores and schematics entries can be directly translated to the ASIC's code. In contrast to first families, present FPGAs offer not only a lot of logic cells, but also a huge register

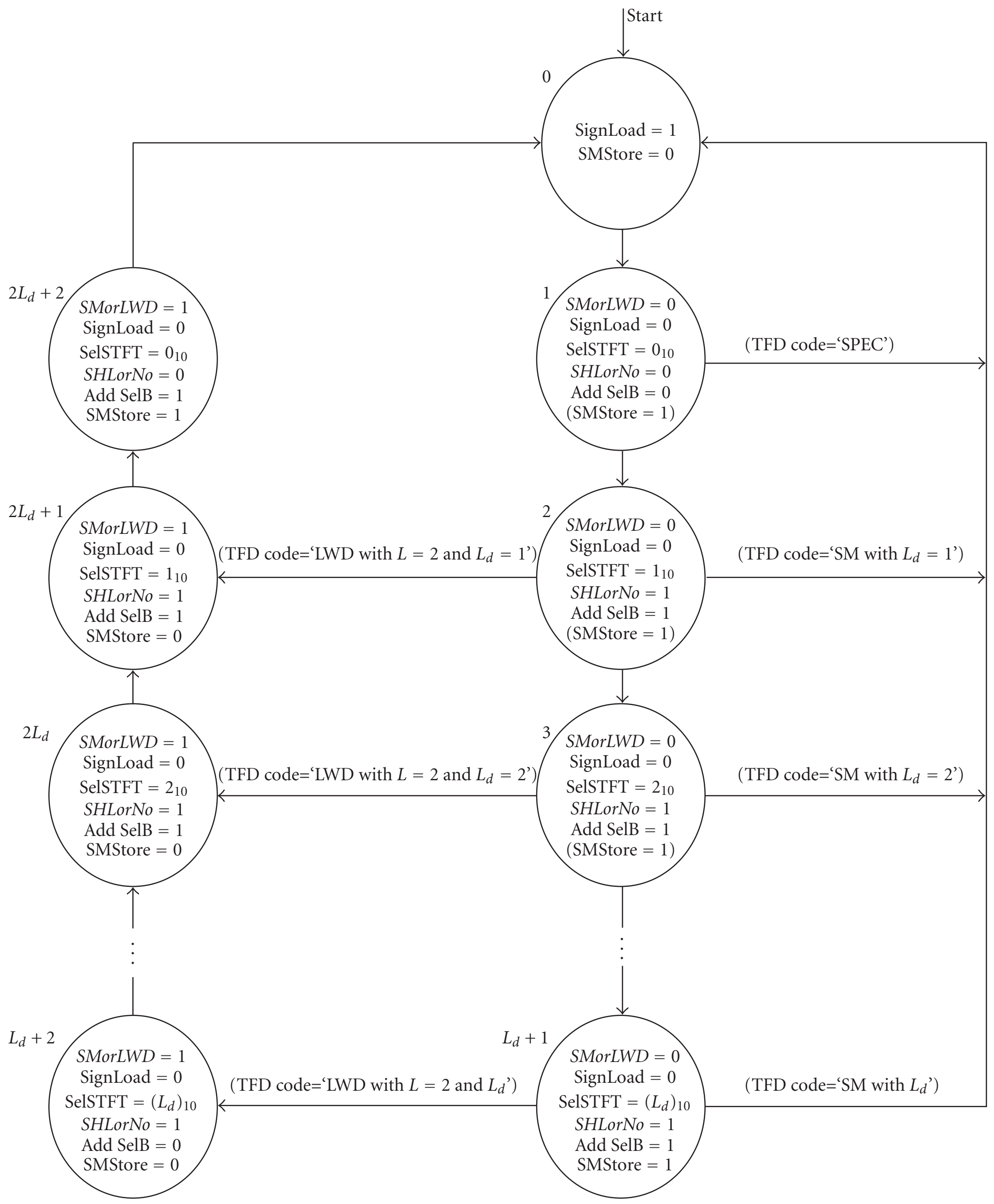

FIGURE 7: The finite-state machine control for the multicycle hardware implementation from Figure 6.

blocks and memory areas. These can be used to built powerful specialized parallel processing units such as adders, multipliers, shifters, and so forth in form of schematic entry or the VHDL code. The internal memory blocks (RAMs, ROMs and FIFOs, etc.) are usable for fast interconnection between parallel structures, as well as to generate the control signals and to configure the system.

In this section, both MCI and SCI architectures are implemented in the FPGA chips. The MCI architecture was implemented following the approach proposed here, whereas the SCI one was implemented following the approach given in [17]. The design was carried out in Altera Max+plus II software. For hardware realization the Altera's FLEX 10 K chips family has been chosen. This family is fabricated in CMOS SRAM technology, running up to 100 MHz and consuming less than 0.5 mA on 5 V. It has a high density of 10,000 to 250,000 typical gates, up to 40,960 RAM bits, 2,048 bits per embedded array block

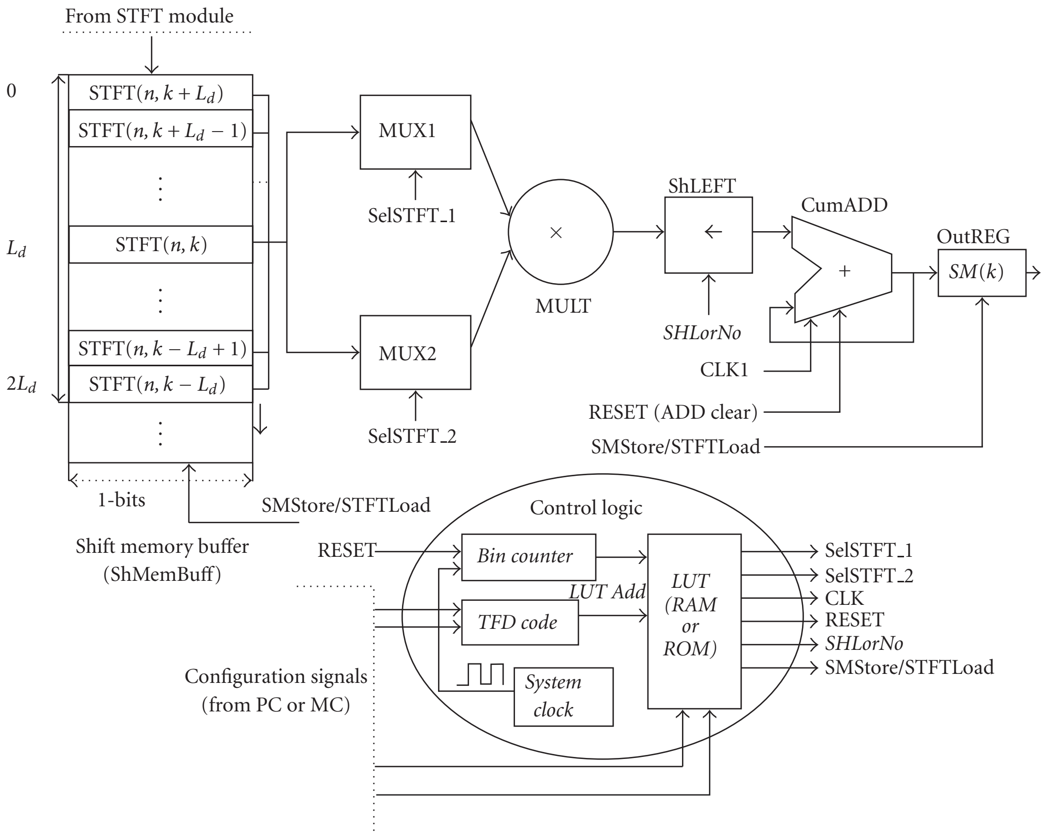

Figure 8: Block diagram of FPGA implementation of the MCI approach.

(EAB), and so on. The computation units are realized by using standard digital components in form of schematics entries or by Altera hardware design language (AHDL)-based mega-functions (library of parametrized modules (LPM)).

The proposed MCI and SCI architectures, implemented in FPGA technology, will be shortly described and compared against usual criteria such as chip capacity, computation speed, power consumption, and cost.

4.1. Implementation of the MCI architecture

The FPGA-based implementation of the MCI architecture follows the design logic given in Figure 8. Since the real and imaginary computation lines are identical, the interpretation will be done through real ones. As seen, it consists of several functional blocks (units). The STFT sample is imported from the STFT module to the *Shift Memory Buffer* (ShMemBuff) that is implemented as an array of parallel-in-parallel-out registers. Their outputs represent the STFT samples in time order $\mathrm{STFT}(n, k + L_d)$, $\mathrm{STFT}(n, k + L_d - 1), \ldots, \mathrm{STFT}(n, k), \ldots, \mathrm{STFT}(n, k - L_d + 1)$, $\mathrm{STFT}(n, k - L_d)$ and due to each *SMStore/STFTLoad* cycle, they have been shifted for one position. These are also fed to the inputs of multiplexors MUX1 and MUX2 and, two-by-two, regarding on multiplexor's addresses *SelSTFT_1* and *SelSTFT_2*, forwarded to the parallel multiplier MULT in order to produce partial product term according to (3). This term is either shifted left or not, depending on the signal *SHLorNo*. This shift is performed by shifter *ShLEFT*, the output of which is connected to the first input of the cumulative pipelined adder *CumADD*. The *CumADD* has been designed to replace an adder and a multiplexor (addressed by the *AddSelB* control signal) from Figures 1 and 2. The time diagram of calculation process is presented in Figure 9. As shown, the multiplying and shifting operations are parallel, while the adding has a latency of one clock. After $L_d + 1$ clocks, the output of the *CumADD* will contain the sum $\mathrm{SM}(n, k)$ that represents the final value of the SM. The next two cycles are used for the signals *SMStore/STFTLoad* and RESET that will store the sum $\mathrm{SM}(n, k)$ in the output register and reset *CumADD* to zero, respectively. Use of the RESET signal will increase the calculation time for one clock. It means that the calculation process takes $L_d + 3$ cycles, one more than is elaborated in Figure 3. Note that the RESET signal can be generated by the signal *SMStore/STFTLoad*, using a short delay, that will reduce the calculation process to $L_d + 2$ cycles. In order to clarify the principle of calculation and simulation (the process of cumulative sums *cumSM* represented in Figure 11), we have used the first variant of RESET generation, with $L_d + 3$ clocks.

Look-up-table (LUT), realized in the form of ROM or RAM memory, manages the computation process. As illustrated in Table 3, its memory location consists of the control

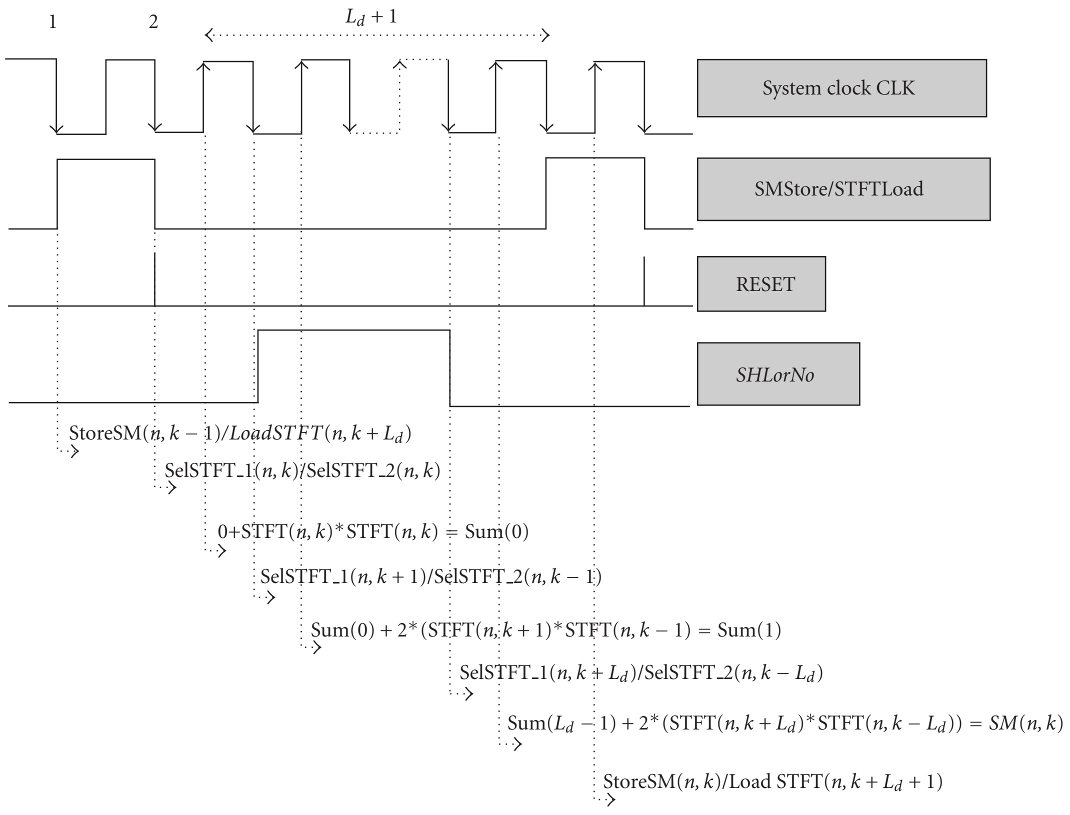
1
2
$L_d + 1$
System clock CLK
SMStore/STFTLoad
RESET
SHLorNo
StoreSM$(n, k-1)$/LoadSTFT$(n, k+L_d)$
SelSTFT_1(n,k)/SelSTFT_2(n,k)
0+STFT(n,k)*STFT(n,k) = Sum(0)
SelSTFT_1$(n,k+1)$/SelSTFT_2$(n,k-1)$
Sum(0) + 2*(STFT$(n,k+1)$*STFT$(n,k-1)$ = Sum(1)
SelSTFT_1$(n,k+L_d)$/SelSTFT_2$(n,k-L_d)$
Sum(L_d-1) + 2*(STFT$(n,k+L_d)$*STFT$(n,k-L_d)$) = $SM(n,k)$
StoreSM(n,k)/Load STFT$(n,k+L_d+1)$

FIGURE 9: The calculation-timing diagram for block diagram from Figure 8.

TABLE 3: LUT's values for given L_d. The ADD(STFT(n, k)) means the address location of the STFT(n, k) sample inside ShMemBuff, whereas m = CEIL($\log_2 N$) = Length(SelSTFT_1). Symbol "≪" denotes logical shift left operation. Note that signals *SHLorNo*, RESET and *SMStore/STFTLoad* make control signals area.

LUT's memory location	*SHLorNo*	RESET	*SMStore/STFTLoad*	*SelSTFT_1* bits	*SelSTFT_2* bits
0	0	0	0	ADD(STFT(n,k)) $\ll m$	ADD(STFT(n,k))
1	1	0	0	ADD(STFT$(n,k+1)$) $\ll m$	ADD(STFT$(n,k-1)$)
—	1	0	0	—	—
L_d	1	0	0	ADD(STFT$(n,k+L_d)$) $\ll m$	ADD(STFT$(n,k-L_d)$)
L_d+1	0	0	1	0	0
L_d+2	0	1	0	0	0

signals area (which consists of signals *SHLorNo*, RESET, and *SMStore/STFTLoad*, resp.) and MUXs' addresses. The binary counter (see Figure 8) generates the low LUT's addresses, while *TFDcode* register sets the high ones. It means that starting address of the running memory block is assigned to the corresponding value L_d stored in *TFDcode* register. At the end of the sequence, the binary counter is cleared by the signal RESET. During system initialization, the memory contents and value of *TFDcode* register are automatically loaded from outside by using PC or general-purpose microcontroller. Of course, these parameters can be permanently stored using ROMs, EEPROMs, and FLASHs instead of RAMs.

Figure 10 shows a schematic diagram for SM calculation from the STFT samples (STFT to SM gateway) using MCI approach. The control logic is realized by using ROM. The maximal register widths for each unit determine the capacity of the assigned chip. The critical point is the width of the *CumADD*. It is a function of both STFT data length and the maximal possible convolution window width $L_{d\max}$ that can be implemented by using proposed architecture. Table 4 shows the relations between minimum widths of units and parameters l (data length) and $L_{d\max}$. In order to verify the chip operation before its programming, the compilation and simulation have been performed by using the various test vectors. An example of simulation is shown in Figure 11.

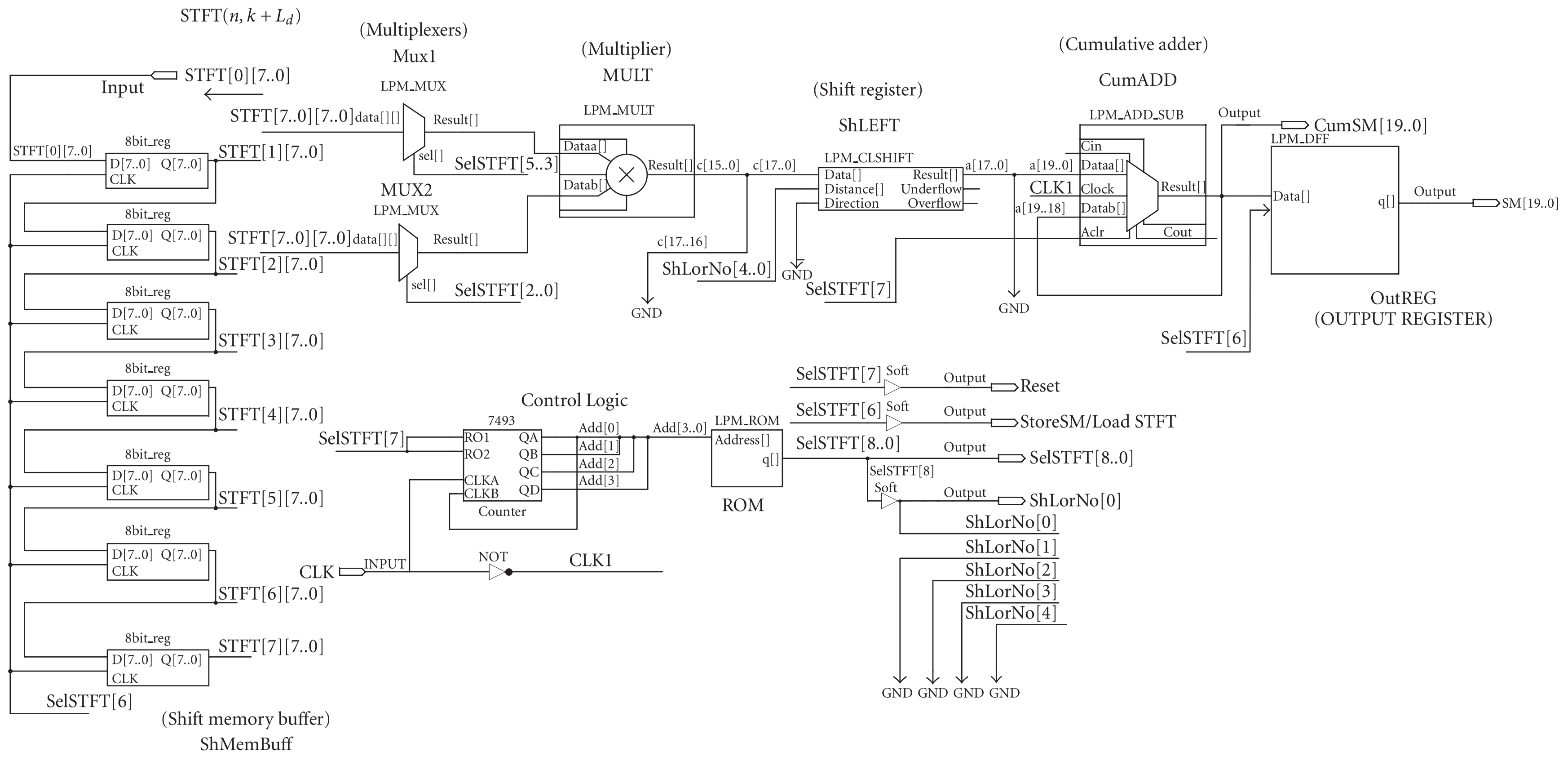

FIGURE 10: The schemastic diagram of the 8-bit STFT to SM gateway implemented in FPGA using MCI approach. It is implemented for $L_d \leq 3$ and $N = 8$.

Table 4: Output register lengths for used digital units depending on the parameters l, $L_{d\max}$.

Length of	MUX1, MUX2	MULT	ShLEFT	*CumADD* and *OutREG*
Parameters l, $L_{d\max}$	l	$2 \cdot l$	$2 \cdot l + 1$	$\mathrm{CEIL}(\log_2((2^{2l+1} - 1) \cdot (L_{d\max} + 1)))$

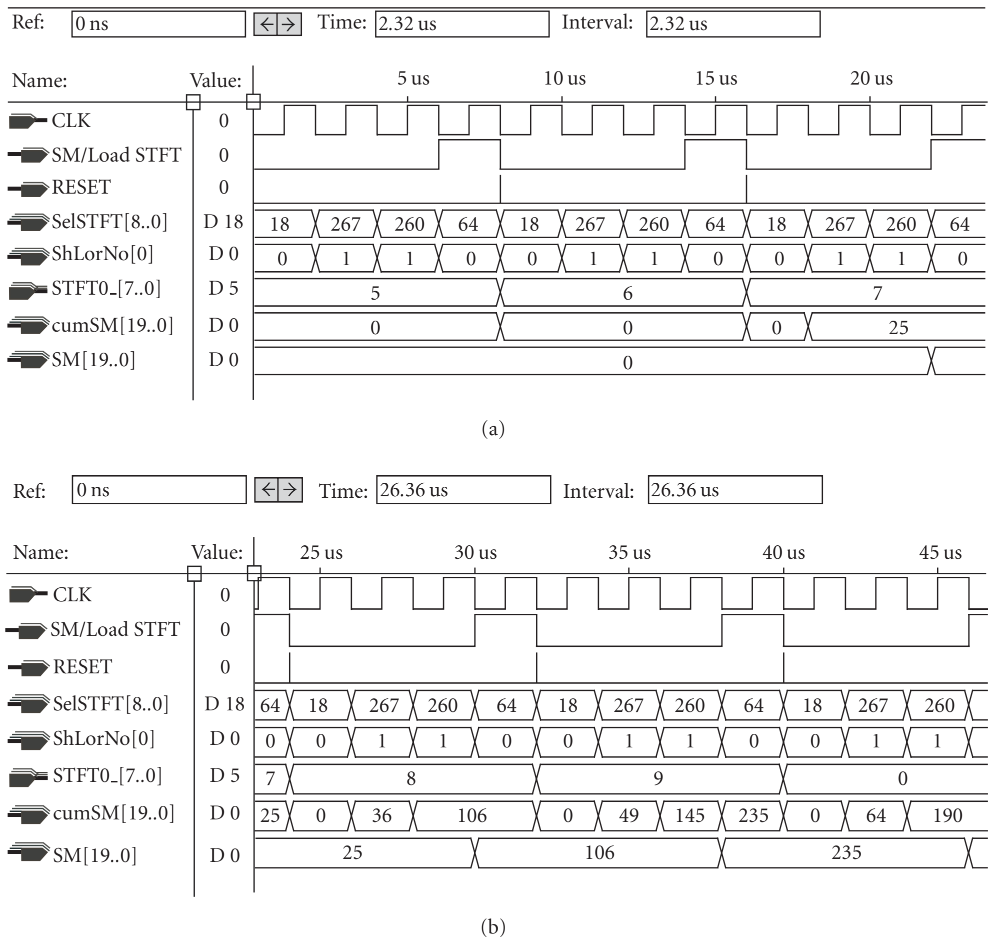

Figure 11: Simulation illustration for test vector $V = \{5, 6, 7, 8, 9, 0, 0, \dots\}$ and $L_d = 3$.

4.2. Implementation of the SCI architecture

As opposite to the MCI architecture, the SCI has no latency [17]. The arithmetic units are realized by using combinational logic, meaning that all calculation operations are performed in parallel. The schematic diagram of its FPGA implementation is given in Figure 12. As seen, there is no need for input multiplexors and control signals such as *SMStore/STFTLoad*, *SelSTFT_1*, *SelSTFT_2*, RESET and *SHLorNo*. Thus, the ROM based generator is needless. At the rising edge of the system clock CLK, the STFT samples are shifted, and due to falling edge, the final result is stored in output register *OutREG*, as shown in the simulation diagram given in Figure 13. One parallel multiplier and one shift register are used for each of product terms from (3), expect for the SPEC term that has no shift register. These terms are added by using cascade network of two-inputs parallel adders, giving the final sum SM[$19 \cdots 0$]. The register widths are the same as in the case of MCI. It should be emphasized that the number of multipliers, shift register, and adders drastically increases with the order of L_d. For example, for $L_d = 3$ we need 4 multipliers (MULT$1 \cdots 4$), 3 shift registers (ShLEFT$1 \cdots 3$), and 3 adders (ParADD$1 \cdots 3$), Figure 12.

4.3. Comparison of MCI and SCI architectures

During the test phase we have implemented 8-bit and 16-bit computation configurations for both architectures MCI and SCI. The different L_ds have been considered. Having in mind the design symmetry, both real and imaginary parts have been developed separately or together. Some implementation details for $L_d = 3$, $N = 8$, and selected real devices from 10 K and 20 K families are summarized in Table 5. In order to generate visual conclusions, the dependence of used logical

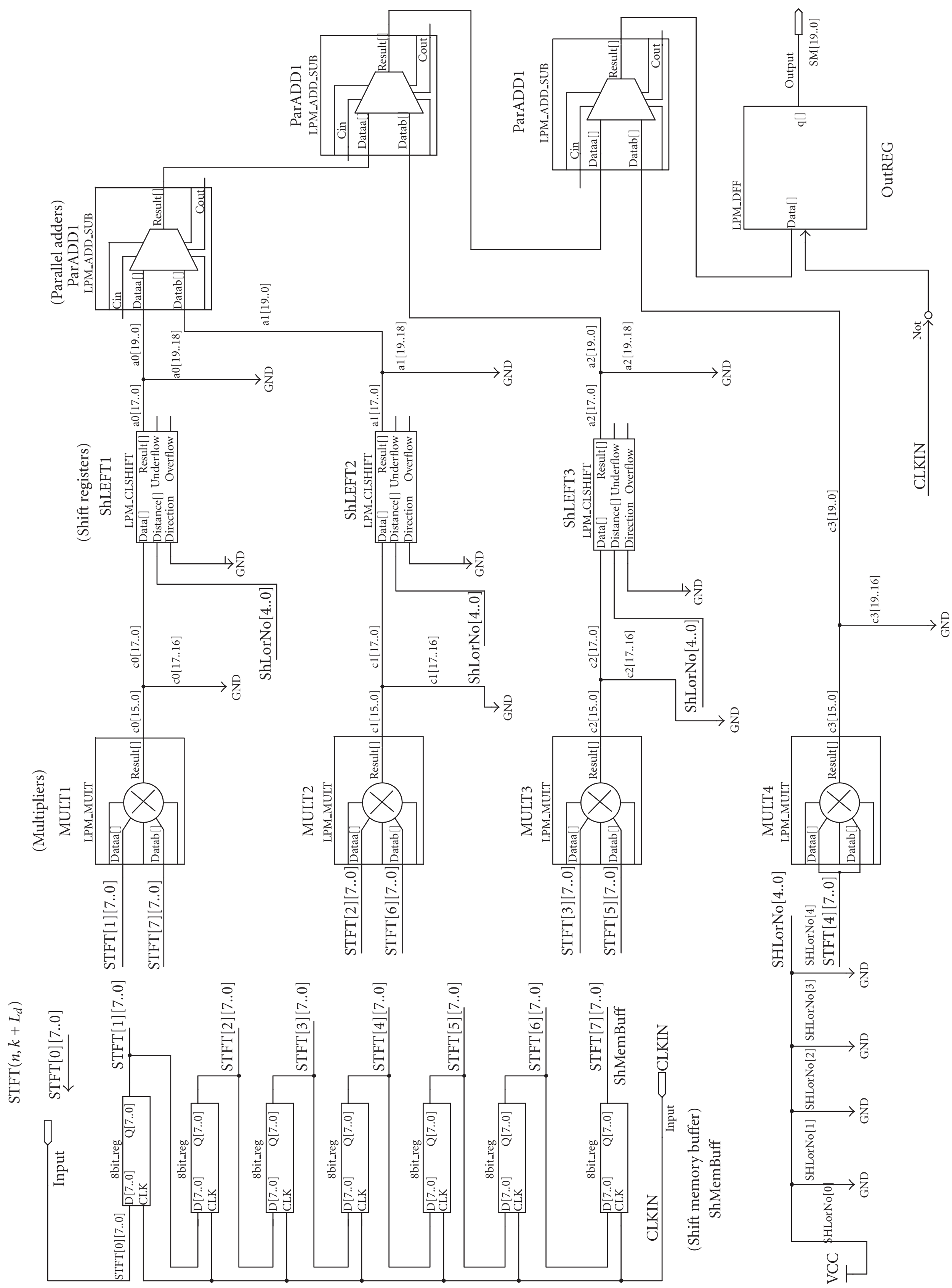

FIGURE 12: FPGA schematic diagram of the 8-bit SCI architecture for $L_d = 3$.

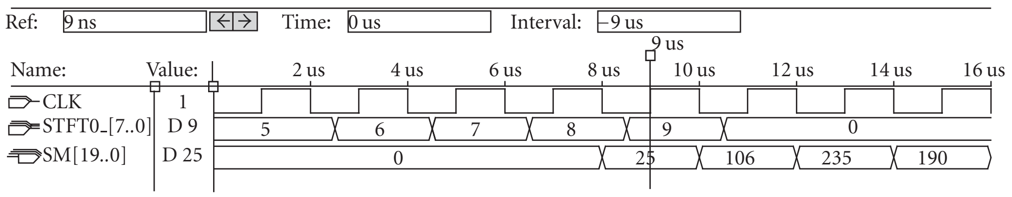

FIGURE 13: Simulation diagrams for SCI architecture. The overall computation process is performed in one clock cycle.

TABLE 5: Summarized implementation utilization for real devices and $L_d = 3$ and $N = 8$ and data lengths $l = 8$ and $l = 16$.

Computation architecture	Total logic cells (LCs) used	Total flip-flops used	Memory bits used	Total I/O pins used	Utilized LCs for recommended device	Recommended device
Real_8-bits MCI	641	101	144	41	55%	EPF10K20TC144-3
Real_8-bits SCI	1728	75	0	29	100%	EPF10K30RC208-3
Real_16-bits MCI	1772	197	144	69	76%	EPF10K40RC208-3
Real_16-bits SCI	5498	147	0	57	No fit	Not fit in the largest of 10 K EPF10K100GC503-3DX4992
					66%	EP20K200
Real + Imag_8-bits MCI	1281	198	144	69	74%	EPF10K30RC208-3
Real + Imag_8-bits SCI	3532	150	0	57	94%	EPF10K70RC240-2
Real + Imag_16-bits MCI	3543	397	144	125	94%	EPF10K70RC248-3
Real + Imag_16-bits SCI	11237	294	0	113	No fit	Not fit in the largest of 10 K EPF10K100GC503-3DX4992
					67%	EP20K400

devices (total logic cells (LCs)) as a function of L_d, for constant $N = 16$, and data length $l = 8$ is illustrated in Figure 14.

As seen, the main advantages of MCI architecture are as follows:

(i) for the same L_d, the MCI architecture needs significantly less LCs for its implementation. It is known that the capacity of chip, that is, the silicon area, is directly proportional to the number of allowed LCs. Since the MCI architecture is structurally identical for different L_ds, the number of LCs could only slightly increase with the increase of N. That is caused by the input span and address lengths of multiplexors (MUX1 and MUX2 from Figure 10);

(ii) the reduced power consumption, which is strongly proportional to the chip capacity; and

(iii) less implementation cost (about 2-3 times).

An advantage of the SCI architecture is the processing speed that is of importance for time-critical applications. The number of LCs significantly varies by L_d (about 400–500 LCs per L_d) that complicates the design and increases the implementation cost and power consumption.

After the simulation, the real FLEX 10 K devices are configured at system power-up using Atlera's UP2 development board with data from ByteBlasterMV. Microcontroller emulated the STFT front end, while the calculated SM was collected and verified by a PC. Because reconfiguration requires less than 320 ms (in case of using external configuration EEPROM), real-time changes can be made during system operation.

5. CONCLUSION

Flexible system for TF signal analysis is proposed. Its MCI design is presented. Proposed architecture can be used for real-time implementation of some commonly used quadratic and higher-order TFDs. It allows a functional unit to be used more than once per TFDs execution, as long as it is used on different clock cycles, and, consequently, enables a significant reduction of hardware complexity and cost. The major advantages of the proposed design are the ability to allow implemented TFDs to take different numbers of clock cycles and to share functional units within a TFDs execution. Finally, proposed architecture is practically verified by

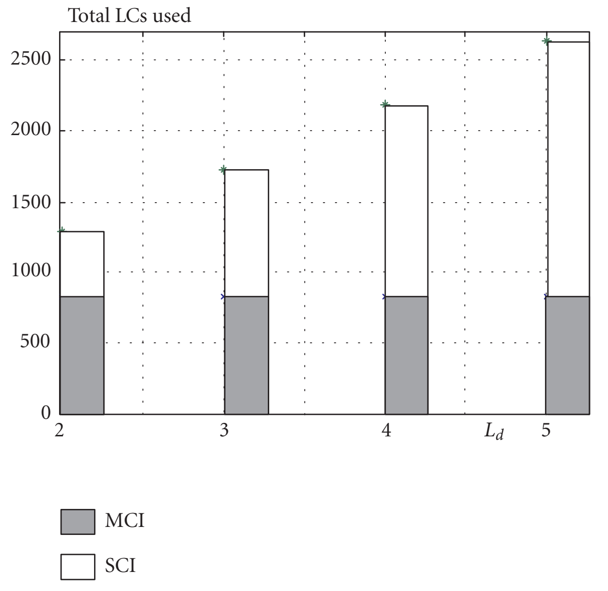

FIGURE 14: The dependance of the LCs used assuming $N = 16$, and data length $l = 8$.

its implementations in FPGA devices and compared with the SCI architecture against usual criteria such as chip capacity, computation speed, power consumption, and cost.

REFERENCES

[1] L. Cohen, "Time-frequency distributions—a review," *Proceedings of the IEEE*, vol. 77, no. 7, pp. 941–981, 1989.

[2] F. Hlawatsch and G. F. Boudreaux-Bartels, "Linear and quadratic time-frequency signal representations," *IEEE Signal Processing Magazine*, vol. 9, no. 2, pp. 21–67, 1992.

[3] L. Cohen, "Preface to the special issue on time-frequency analysis," *Proceedings of the IEEE*, vol. 84, no. 9, pp. 1197–1197, 1996.

[4] LJ. Stanković, "A method for time-frequency analysis," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, pp. 225–229, 1994.

[5] B. Boashash and B. Ristic, "Polynomial time-frequency distributions and time-varying higher order spectra: application to the analysis of multicomponent FM signals and to the treatment of multiplicative noise," *Signal Processing*, vol. 67, no. 1, pp. 1–23, 1998.

[6] P. Goncalves and R. G. Baraniuk, "Pseudo affine Wigner distributions: definition and kernel formulation," *IEEE Transactions on Signal Processing*, vol. 46, no. 6, pp. 1505–1516, 1998.

[7] C. Richard, "Time-frequency-based detection using discrete-time discrete-frequency Wigner distributions," *IEEE Transactions on Signal Processing*, vol. 50, no. 9, pp. 2170–2176, 2002.

[8] L. L. Scharf and B. Friedlander, "Toeplitz and Hankel kernels for estimating time-varying spectra of discrete-time random processes," *IEEE Transactions on Signal Processing*, vol. 49, no. 1, pp. 179–189, 2001.

[9] LJ. Stanković, V. N. Ivanović, and Z. Petrović, "Unified approach to the noise analysis in the spectrogram and Wigner distribution," *Annales des Telecommunications*, vol. 51, no. 11-12, pp. 585–594, 1996.

[10] S. Stanković and LJ. Stanković, "An architecture for the realization of a system for time-frequency signal analysis," *IEEE Transactions on Circuits And Systems—Part II: Analog and Digital Signal Processing*, vol. 44, no. 7, pp. 600–604, 1997.

[11] LJ. Stanković and J. F. Böhme, "Time-frequency analysis of multiple resonances in combustion engine signals," *Signal Processing*, vol. 79, no. 1, pp. 15–28, 1999.

[12] LJ. Stanković, "A method for improved distribution concentration in the time-frequency analysis of multicomponent signals using the L-Wigner distribution," *IEEE Signal Processing Magazine*, vol. 43, no. 5, pp. 1262–1268, 1995.

[13] K. J. R. Liu, "Novel parallel architectures for short-time Fourier transform," *IEEE Transactions on Circuits And Systems—Part II: Analog and Digital Signal Processing*, vol. 40, no. 12, pp. 786–790, 1993.

[14] M. G. Amin and K. D. Feng, "Short-time Fourier transforms using cascade filter structures," *IEEE Transactions on Circuits And Systems—Part II: Analog and Digital Signal Processing*, vol. 42, no. 10, pp. 631–641, 1995.

[15] B. Boashash and P. Black, "An efficient real-time implementation of the Wigner-Ville distribution," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 11, pp. 1611–1618, 1987.

[16] D. Petranović, S. Stanković, and LJ. Stanković, "Special purpose hardware for time-frequency analysis," *Electronics Letters*, vol. 33, no. 6, pp. 464–466, 1997.

[17] S. Stanković, LJ. Stanković, V. N. Ivanović, and R. Stojanović, "An architecture for the VLSI design of systems for time-frequency analysis and time-varying filtering," *Annales des Telecommunications*, vol. 57, no. 9-10, pp. 974–995, 2002.

[18] K. Maharatna, A. S. Dhar, and S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST," *Signal Processing*, vol. 81, no. 9, pp. 1813–1822, 2001.

[19] K. J. R. Liu and C.-T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/Hartley transforms," *IEEE Transactions on Signal Processing*, vol. 41, no. 3, pp. 1357–1377, 1993.

[20] A. Papoulis, *Signal Analysis*, McGraw-Hill, New York, NY, USA, 1977.

[21] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1975.

[22] M. G. Amin, "A new approach to recursive Fourier transform," *Proceedings of the IEEE*, vol. 75, no. 11, pp. 1537–1538, 1987.

[23] M. Unser, "Recursion in short-time signal analysis," *Signal Processing*, vol. 5, no. 3, pp. 229–240, 1983.

[24] M. G. Amin, "Spectral smoothing and recursion based on the nonstationarity of the autocorrelation function," *IEEE Transactions on Signal Processing*, vol. 39, no. 1, pp. 183–185, 1991.

[25] V. N. Ivanović and LJ. Stanković, "Multiple clock cycle real-time implementation of a system for time-frequency analysis," in *Proceedings of 12th European Signal Processing Conference (EUSIPCO '04)*, pp. 1633–1636, Vienna, Austria, September 2004.

[26] V. N. Ivanović, LJ. Stanković, and D. Petranović, "Finite word-length effects in implementation of distributions for time-frequency signal analysis," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 2035–2040, 1998.

Veselin N. Ivanović was born in Cetinje, Montenegro, April 10, 1970. He received the B.S. degree in electrical engineering (1993) and the M.S. degree in electrical engineering from the University of Montenegro (1996). He received the Ph.D. degree in electrical engineering from the same University (2001) in time-frequency signal analysis and architecture design for implementation of time-frequency methods and time-varying filtering. In 2001, he received the Siemens Award for scientific achievements in his Ph.D. research. Dr. Ivanović is an Assistant Professor (Docent) at the Electrical Engineering Department, University of Montenegro. He is also Vice-Dean at the electrical engineering Department, University of Montenegro. His research interests are in the areas of time-frequency signal analysis, hardware/software codesign, computer organization and design, and design with microcontrollers.

Radovan Stojanović was born in Berane, Montenegro, Yugoslavia, November 18, 1965. He received the B.S.E.E. and M.S.E.E. degrees from the University of Montenegro, and the Ph.D. degree from the University of Patras, Greece, in 1991, 1994, and 2001, respectively. From 1990 to 1998, he was at the Electrical Engineering Department, University of Montenegro. From 1998 to 2001, he was a Research Associate at the Department of Electrical Engineering and Computer Technology, University of Patras, Greece. After that, he spent two years as a Senior Researcher in the Industrial System Institute (ISI), Patras, Greece. Currently, he is an Assistant Professor at the University of Montenegro guiding the group of applied electronics. His fields of interest are hardware/software codesign, applied signal and image processing, and industrial and medical electronics.

LJubiša Stanković was born in Montenegro, June 1, 1960. He received the B.S. degree in electrical engineering from the University of Montenegro, in 1982, with the honor "the best student at the University," the M.S. degree in electrical engineering, in 1984, from the University of Belgrade, and the Ph.D. degree in electrical engineering in 1988 from the University of Montenegro. As a Fulbright grantee, he spent the 1984/1985 academic year at the Worcester Polytechnic Institute, Massachusetts. Since 1982, he has been on the faculty at the University of Montenegro, where he now holds position of a Full Professor. Stanković was also active in politics, as a Vice-President of the Republic of Montenegro (1989–1991), and then the leader of democratic (anti-war) opposition in Montenegro (1991–1993). During 1997/1998 and 1999, he was on leave at the Ruhr University Bochum, Germany, with Signal Theory Group, supported by the Alexander von Humboldt foundation. At the beginning of 2001, he spent a period of time at the Technische Universiteit Eindhoven, the Netherlands, as a Visiting Professor. During the priod of 2001–2002 he was the President of the Governing Board of the Montenegrin mobile phone company "MONET." His current interests are in signal processing and electromagnetic field theory. He published about 270 technical papers, more than 80 of them in leading international journals, mainly the IEEE editions. He has published several textbooks about signal processing (in Serbo-Croat) and the monograph *Time-Frequency Signal Analysis* (in English). For his scientific achievements, he was awarded the Highest State Award of the Republic of Montenegro in 1997. Professor Stanković is a Member of the IEEE Signal Processing Society's Technical Committee on Theory and Methods. He is an Associate Editor of the IEEE Transactions on Image Processing. He is a Member of the Yugoslav Engineering Academy, and a Member of the National Academy of Science and Art of Montenegro (CANU). Professor Stanković is the Rector of the University of Montenegro since 2003.