Электронный журнал «Техническая акустика»
http://www.ejta.org

2016, 1

I. V. Ushenina

Penza State Technological University,
440039, Russia, Penza, pr. Baidukova / ul. Gagarina, 1a/11, e-mail: ivl23@yandex.ru

FPGA-based Multi-Channel Adaptive FXLMS Filter Implemented as an Array of Processing Blocks

Received 18.11.2015, published 29.04.2016

The paper analyses the approaches to implementation of a multi-channel adaptive filter used as a controller of multi-channel active noise control systems. The advantages of filter implementation as the array of processing blocks are shown. The filter's processing blocks are based on the DSP slices of field programmable gates arrays (FPGAs). The size of processing blocks array is defined by the number of channels in the filter. The paper also analyzes the dependency of maximum operating frequency on the array size for the presented architecture.

Key words: active noise control, multi-channel adaptive filter, processing block, DSP slice.

INTRODUCTION

To solve some kinds of active noise control (ANC) tasks, such as noise suppression in enclosed space or creating local zone of silence, the multi-channel systems with J reference microphones, K loudspeakers and M error microphones are used. To produce anti-noise signals, such systems use the multi-channel adaptive filters with filtered-reference least mean square algorithm (FXLMS) as controllers [1, 2].

A block diagram of an ANC system controller, under condition that $J=K=M=2$, is shown in Figure 1. Every j-th reference signal is filtered by KJ adaptive filters (AFs) and MK estimations of cancellation path transfer functions (FEs) between the loudspeakers and the error microphones [2]:

$$y_{kj}(n) = \boldsymbol{w}_{kj}^T(n) * \boldsymbol{x}_j(n) = \sum_{i=0}^{N_{kj}-1} x_j(n-i) \cdot w_{kj}^i(n), \quad (1)$$

$$x'_{mkj}(n) = \hat{\boldsymbol{s}}_{mk}^T * \boldsymbol{x}_j(n) = \sum_{i=0}^{L_{mk}-1} x_j(n-i) \cdot \hat{s}_{mk}^i, \quad (2)$$

where n is the sampling interval number, $y_{kj}(n)$ is the kj-th component of the k-th anti-noise signal $y_k(n)$, $\boldsymbol{w}_{kj}(n) = [w_{kj}^0(n) \quad w_{kj}^1(n) \quad \ldots \quad w_{kj}^{N-1}(n)]^T$ is the kj-th AF weight coefficient vector in the n-th sampling interval, $\boldsymbol{x}_j(n) = [x_j(n) \quad x_j(n-1) \quad \ldots \quad x_j(n-N+1)]^T$ is the j-th reference signal samples, N_{kj} is the order of kj-th AF, $x'_{mkj}(n)$ is the result of filtering j-th

reference signal by *mk*-th FE, $\hat{\boldsymbol{s}}_{mk} = [\hat{s}_{mk}^0 \quad \hat{s}_{mk}^1 \quad \ldots \quad \hat{s}_{mk}^{L-1}]^T$ is the *mk*-th FE weight coefficient vector, L_{mk} is the order of *mk*-th FE. The AFs coefficients must be recalculated with respect to all error signals [2]:

$$w_{kj}^i(n+1) = w_{kj}^i(n) + \mu \cdot \sum_{m=1}^{M} e_m(n) \cdot x'_{mkj}(n-i), \tag{3}$$

where $w_{kj}^i(n+1)$ is the new value of the *i*-th coefficient of *kj*-th AF, $w_{kj}^i(n)$ is the current value of the *i*-th coefficient of *kj*-th AF, μ is a convergence coefficient [3], $e_m(n)$ is the *m*-th error signal, $x'_{mkj}(n-i)$ is the *n*-*i*-th sample of the *j*-th reference signal filtered by *mk*-th FE.

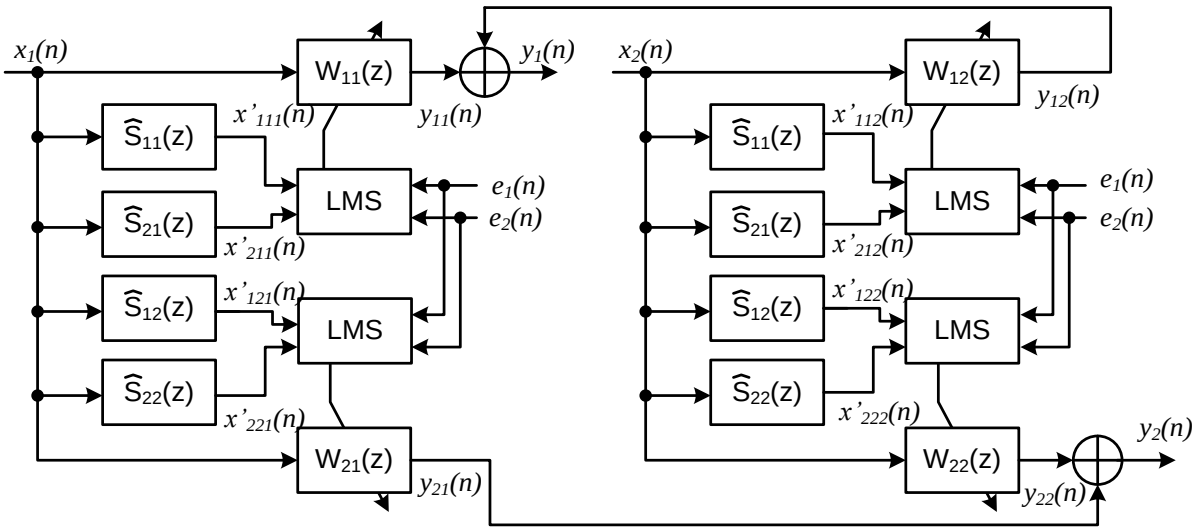

Fig. 1. ANC system controller block diagram, under condition that *J=K=M=2*

The performance needed for ANC system controller is mainly defined by the total number N_{MAC} of multiply-accumulate operations that system's filter must execute per second (this value is usually measured in millions of operations per second – MACs):

$$N_{MAC} = F_s \cdot [(N + L) \cdot M \cdot K \cdot J + N \cdot K \cdot J], \tag{4}$$

where F_s is the ANC system sampling frequency, N_{MAC} is the total number of multiply-accumulate operations. In (4) and further in the paper, the orders of all AFs and all FEs are supposed to be equal to *N* and *L* accordingly.

The highest order AFs (up to some thousands of coefficients) are needed when ANC system is intended to suppress low frequency random noise in a range between approximately 100 Hz and 1 kHz. It is because AF together with FE must simulate the transfer function of the path between the reference microphone and the loudspeaker in a broad band of frequencies [4, 5].

In most cases, the ANC systems' controllers are implemented on DSP processors [1, 2, 6, 7]. Taking into account that *N* and *L* can reach a few thousands of taps [1, 5, 6], *J*, *K*, *M* are typically several tens [1, 2], and F_s should be about ten times higher than the noise frequency

[1], the required processor performance may be up to tens of GMACs or more. This requirement approximately corresponds to the peak performance of the fastest multi-core DSP processors [8]. When a single DSP processor cannot deliver the required performance, several DSPs are combined into an array [1, 6, 7]. However, the implementation of high-performance controllers in multi-channel ANC systems can have an alternative platform - FPGAs. FPGA performance can reach thousands of GMACs due to the fact that thousands of DSP slices are present on a chip [9]. Such quantity of DSP slices and programmable interconnections between FPGA sources make it possible to implement a multi-channel FXLMS filter in a few different ways. Moreover, last-generation FPGAs can work with both the fixed point and floating point data formats [10]. In the paper below, the author compares two approaches to using FPGAs for designing multi-channel FXLMS filters, gives a detailed description of this filter architecture implemented as an array of processing blocks and analyzes this architecture performance.

1. POSSIBLE METHODS OF IMPLEMENTING FPGA-BASED MULTI-CHANNEL FXLMS FILTERS

Specialists may distinguish two main approaches to implementing a multi-channel FXLMS filter on the base of FPGA DSP slices [11]. The first of these supposes modification to the parallel systolic architecture of FIR filter with N taps by addition of an input signal multiplexer, coefficient RAMs attached to each DSP slice and "delay-by-$J(MK+1)$" elements between the filter taps to hold the interleaved streams [12]. The obtained structure is supposed to work in time-multiplexed mode. The disadvantage of this approach is that the required number of DSP slices is equal to N, which can be of large value. Furthermore, it is a great possibility to leave some of the DSP slices unused. That is because the number of DSP slices is approximately equal to the number of 18K block RAMs [9]. When the volume of a single block RAM is not enough to include a delay element between two adjacent taps of the filter, every other (or others) DSP slice is bypassed. The same result is obtained when an FPGA block memory is also used to implement coefficient RAMs. On the other hand, implementing coefficient RAMs on an FPGA distributed memory may result in a significant reduction in performance.

In the second approach, a multi-channel FXLMS filter is implemented as an array of individual processing blocks comprised of one or more DSP slices. Unlike the previous case, the number of processing blocks is proportional to any combination of two parameters out of three – M, K, and J. Each block deals with an individual set of filters' coefficients and input/output signals and implements N or L multiply-accumulate operations sequentially. The increase of any combination of M, K, J, L and N should not lead to changes in the processing block structure; it only affects the total number of processing blocks and the required memory volume. However, the enlargement of an FPGA-implemented structure may result in its performance decrease. In the multi-channel ANC systems, M is often larger than J and K [1, 2], so it is expedient to implement the array with the size of $K \times J$. The next sections examine the architecture of the multi-channel FXLMS filter implemented as a $K \times J$ array of processing blocks.

2. THE ARCHITECTURE OF THE MULTI-CHANNEL FXLMS FILTER

The proposed architecture is shown in Figure 2. Each *kj*-th processing block contains two DSP slices (DSP slice 1, DSP slice 2) and two memory blocks ($\mu X'_{mkj}$_RAM, W_{kj}_RAM).

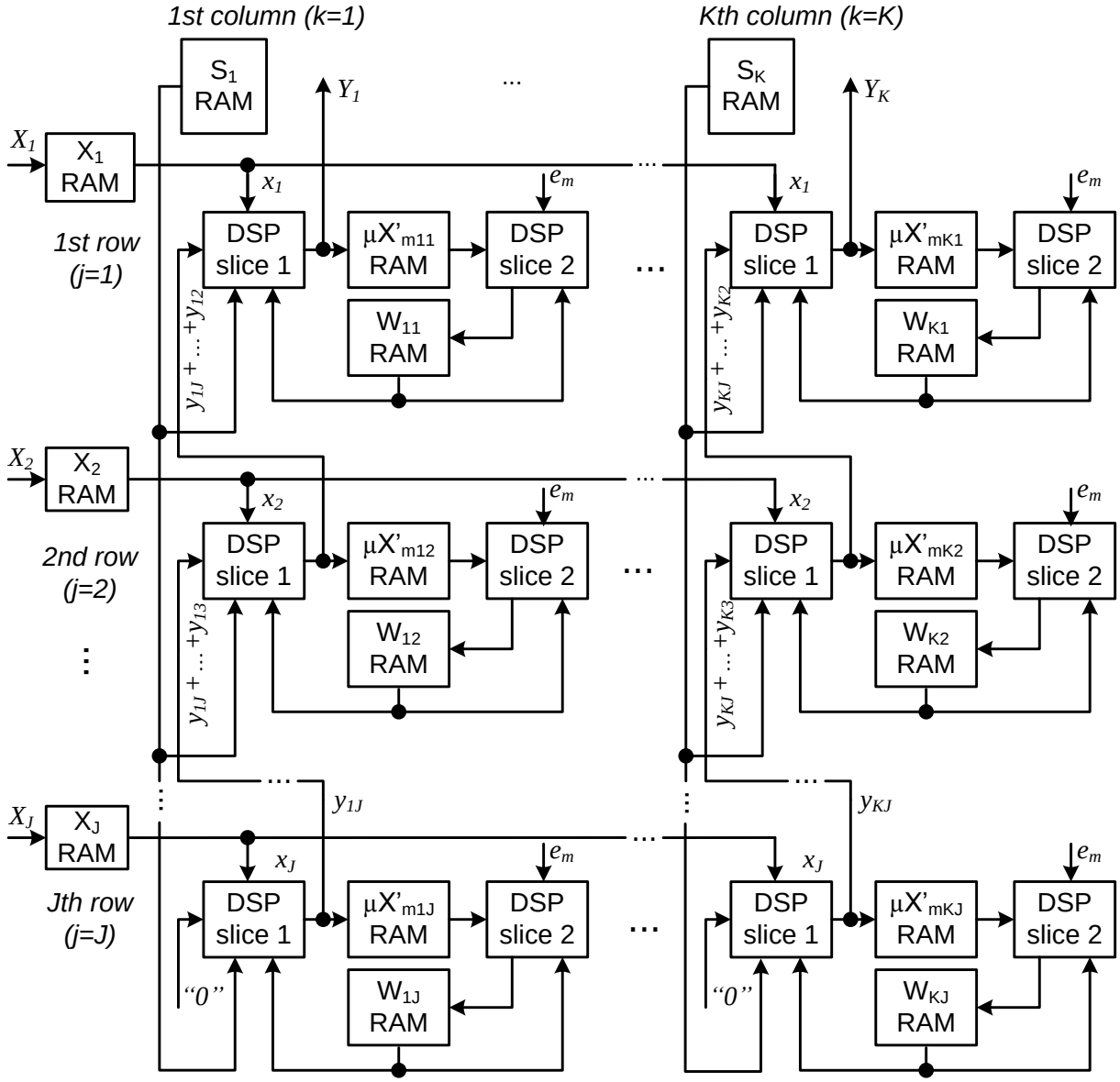

Fig. 2. Multi-channel FXLMS filter as an array of processing blocks

In each *kj*-th block, DSP slice 1 produces the *j*-th part of the *k*-th compensating signal ($y_{kj}(n)$) and *M* samples of filtered reference signal $x'_{mkj}(n)$ - by filtering the *j*-th reference signal by *kj*-th AF and *M mk*-th FEs (1, 2). DSP slice 2 produces a new set of *kj*-th AF coefficients (3). The operation modes of both DSP slices are programmed as shown in Figure 3.

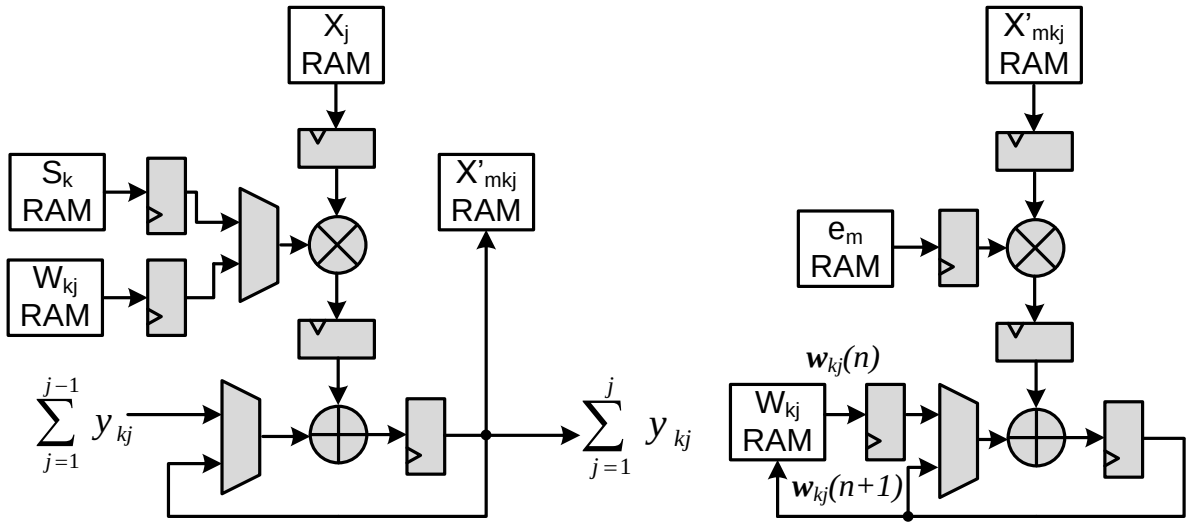

Fig. 3. DSP slice 1 (on the left) and DSP slice 2 operating modes
The elements of DSP slices are grayed

The W_{kj}_RAM memory block contains N AF coefficients, and is fully updated at every sampling interval. The $\mu X'_{mkj}$_RAM memory block is a cyclic buffer containing the $N+1$ sets of M $x'_{mkj}(n-i)$ samples received after filtering the j-th reference signal by M FE_{mk} filters. The organization of memory blocks is illustrated in Figure 4.

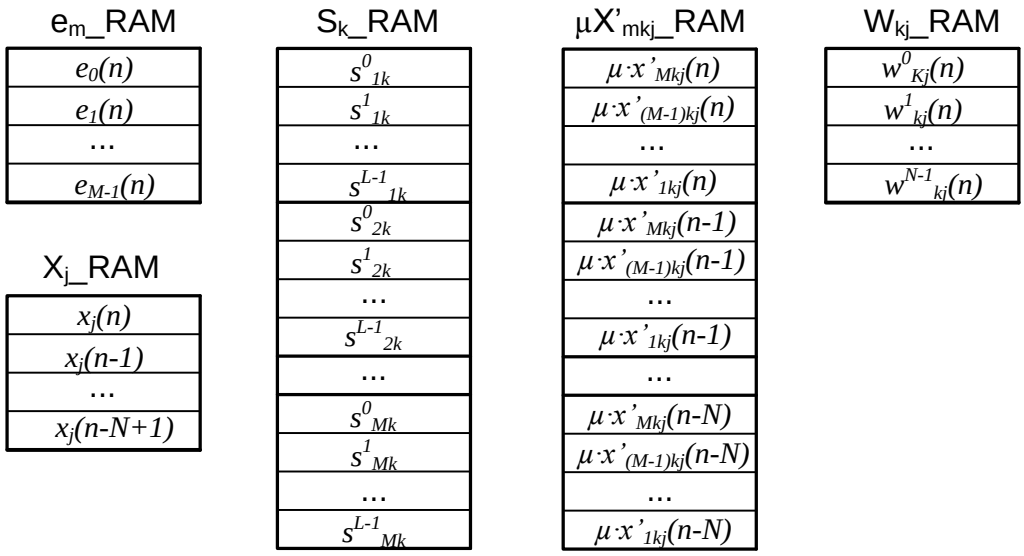

Fig. 4. Organization of memory blocks

In a current sampling interval, N sets of M $x'_{mkj}(n-i)$ values (where $i=1\ldots N$, see Figure 4) are used by DSP slice 2 to produce new coefficients of kj-th AF (3). At the same time, DSP slice 1 filters the reference signal by M FE filters and writes M new values into the

μX'$_{mkj}$_RAM. Following these procedures, DSP slice 1 filters the reference signal by using the new AF coefficients.

Each *j*-th row of processing blocks deals with the *j*-th reference signal, which feeds into DSP slices 1 in *K* processing blocks in the row. *N* samples of the *j*-th reference signal are held in the X_j_RAM memory block. Each *k*-th column of processing blocks generates a compensating signal $y_k(n)$ by adding its parts produced by the *J* individual processing blocks in the column (Figure 2). On calculating $x'_{mkj}(n)$, each processing block in the *k*-th column deals with the shared memory block Sk_RAM containing the coefficients of *M* FEs (Figure 4). *M* error signals feed into all DSP slices 2 from a shared e_m_RAM memory block (not shown in Figure 2).

3. FPGA RESOURCE REQUIREMENTS AND OPERATING FREQUENCY PRE-ESTIMATION

$2KJ$ DSP slices are required to implement the multi-channel FXLMS filter as it is shown in figure 2. Such quantity of DSP slices is usually available even in low-cost FPGAs.

The required memory volume does not dependent on the filter architecture [11]. Under condition of 18-bit precision, the implementation of a multi-channel FXLMS filter requires $18\cdot[KJ\cdot(N+MN+M)+KML+JN+K]$ bits of memory. However, the architecture defines organization and volume requirements for individual memory blocks. For considered architecture, the requirements concerned with the memory blocks' organization are listed in Table 1.

It is obvious from Table 1 that about half of the required total memory volume is assigned to the implementation of μX'$_{mkj}$_RAMs. Meanwhile, the results of $x'_{mkj}(n)$ multiplication by μ saving in μX'$_{mkj}$_RAMs instead of $x'_{mkj}(n)$ may significantly reduce memory volume consumption if μ is equal to the negative power of two. In this case, each memory location in μX'$_{mkj}$_RAM may contain about 5-6 MSBs of $x'_{mkj}(n)$.

Table 1. Memory blocks' organization and volume requirements

Memory block	Number of blocks	Number of memory locations in a block
X_j_RAM	*J*	*N*
S_k_RAM	*K*	*LM*
e_m_RAM (not shown in Figure 2)	*1*	*M*
W_{kj}_RAM	*KJ*	*N*
μX'$_{mkj}$_RAM	*KJ*	*M·(N+1)*

Currently, FPGA block RAM memory volume is up to tens of Mb [9]. Under condition that *J*, *K* and *M* are about several tens, $L \le N$, and μX'$_{mkj}$_RAM memory locations contain 5-6 MSBs of $x'_{mkj}(n)$, it is obtainable to implement filters containing up to a few thousands of taps each.

The operating frequency of the ANC system controller must be at least P times higher than its sampling frequency, where P is a number of clock cycles spent by the processing block in a sampling interval. The numbers of clock cycles required for processing block's operations are shown in Table 2. As new AF coefficients are calculated in parallel with filtering the reference signal by FE, DSP slices are programmed as shown in Figure 3, $J << N$, and $M\cdot(L+3)$ is less than $N\cdot(M+5)$, minimum operating frequency can be estimated as:

$$f_{op} > F_s N(M+6). \quad (5)$$

Table 2. Numbers of clock cycles required to perform processing block's operations

Operation	AF coefficients recalculation and W_{kj}_RAM updating	Filtering the reference signal by FEs and $\mu X'_{mkj}$_RAM updating	Filtering the reference signal by AF and $y_k(n)$ calculation
Number of clock cycles	$N\cdot(M+5)$	$M\cdot(L+3)$	$N+3 + J-1$

The required operating frequency must also be in compliance with the FPGA switching characteristics. Taking into account that F_s is usually equal to a few kHz [1], M can reach values up to a few tens and N can reach values up to a few thousands, the required operating frequency may be within the range of tens to hundreds MHz. Although DSP slices and block RAM primitives individually operate on frequency up to 500-700 MHz [13-15], the performance of the processing block array implemented on FPGA can be sufficiently reduced if the array size increases. This can happen due to the increase of the signals' propagation delay times within the array. The dependency of maximum operating frequency on the array size will be shown and analyzed in the next section.

4. TIMING ANALYSIS RESULTS AND DISCUSSION

The results of the multi-channel FXLMS filter timing analysis presented in this section are obtained for Xilinx 7 series FPGAs [9] under condition of 18-bit data format.

At first, it is necessary to consider the dependency of individual processing blocks performance on M and N. The maximum operating frequency of a DSP slice can be achieved when the internal pipeline registers are used as shown in Figure 3 [16]. A FPGA block memory is available as 36K primitives, which can be connected into a 64K RAM without loss of performance. Further memory block size enlargement leads to a performance decrease caused by the additional propagation delay times between the block RAM primitives. However, the block RAM columns have special routing resources to minimize these delays [17].

The maximum processing block operating frequencies (f_{op_max}) obtained by post-place and route timing analysis for several $M\times N$ ratios are listed in Table 3. The project implemented on Artix 7 XC7A200T-3 FPGA contains a single processing block with a given $M\times N$ ratio and a control unit. The memory locations of $\mu X'_{mkj}$_RAM are of 6 bit width. Table 3 confirms that the maximum performance of processing blocks corresponds to the cases when $\mu X'_{mkj}$_RAM is fully mapped into single or paired block primitives.

Table 3. Processing block maximum operating frequencies achievable on XC7A200T-3 FPGA

$M \times N$ ratio	2×1024	6×1024	12×1024	24×1024	32×1024	64×1024
f_{op_max}, MHz	447	447	447	446	400	337

For the processing block array as a whole, propagation delays affect the operating frequency more significantly. The propagation delay time is a function of signal fanout and a number of switching elements in the interconnection between the logic sources. The first parameter is defined by the total number of processing blocks. The last parameter depends on the distance between the logic sources being connected. Thus, both parameters are proportional to the K and J values. The M increase should affect the performance degradation less than the increase of K and J because of special routing resources used between the memory block primitives, as well as because the increase of M does not cause so much growth of signal fanouts compared to the increase of K and J.

Tables 4-6 show the maximum achievable operating frequencies of processing block arrays with various array sizes and $M \times N$ ratios.

Table 4. Maximum operating frequencies achievable on XC7A200T-3 FPGA for 2×1024 $M \times N$ ratio

	Array size, $K \times J$				
	2×2	4×4	6×6	8×8	10×10
f_{op_max}, MHz (under default maximum signal fanout)	447	406	346	309	255
f_{op_max}, MHz (under reduced maximum signal fanout)	447	440	408	357	312

Tables 4-6 contain the timing analysis results for two cases: default maximum signal fanout of 100000 and reduced maximum signal fanout. The reduced value was selected for each array size individually. It has been shown that signal fanout reduction may cause only insignificant performance gain. The rest part of the propagation delay time is obviously defined by the distance between the logic sources.

Table 5. Maximum operating frequencies achievable on 7 series FPGAs for 12×1024 *M*×*N* ratio

	FPGA	Array size, *K*×*J*				
		2×2	4×4	6×6	8×8	10×10
f_{op_max}, MHz (under default maximum signal fanout)	XC7A200T-3	409	324	268	241	220
	XC7K325T-3	528	421	367	317	289
	XC7VX415T-3	504	421	325	303	304
f_{op_max}, MHz (under reduced maximum signal fanout)	XC7A200T-3	421	339	293	256	227
	XC7K325T-3	528	425	400	333	310
	XC7VX415T-3	524	439	402	335	313

Table 6. Maximum operating frequencies achievable on 7 series FPGAs for 32×1024 *M*×*N* ratio

	FPGA	Array size, *K*×*J*				
		2×2	4×4	6×6	8×8	10×10
f_{op_max}, MHz (under default maximum signal fanout)	XC7A200T-3	393	333	222	-	-
	XC7K325T-3	442	400	303	142	-
	XC7VX415T-3	477	414	313	144	125
f_{op_max}, MHz (under reduced maximum signal fanout)	XC7A200T-3	396	333	222	-	-
	XC7K325T-3	441	400	322	144	-
	XC7VX415T-3	480	416	325	185	147

On the other hand, there are only error, control and address signals that require interconnection with all processing blocks. Pipelining of these signals should be helpful to keep the performance of large-sized arrays high. Table 7 contains the timing analysis results obtained for the arrays of various sizes under condition of pipelining of the error, control and address signals, as well as the FEs coefficients. As a result, each row in an array gets signals delayed in the previous one.

Table 7. Maximum operating frequencies achievable for 32×1024 $M\times N$ ratio with and without signal pipelining on 7 series FPGAs

Array size, $K\times J$	4×4	4×4	4×4	8×8	8×8
FPGA	XC7A200T-3	XC7K325T-3	XC7VX415T-3	XC7K325T-3	XC7VX415T-3
f_{op_max}, MHz (without pipelining)	333	400	414	142	144
f_{op_max}, MHz (with pipelining)	357	425	406	246	295

CONCLUSION

For a multi-channel ANC system, the increase of number of channels and the increase of controller's performance promote the extension of the band of noise reduction, the enlargement of zone of silence created by ANC system and increase of noise reduction level in this zone. The proposed decision allows changing the method of ANC system's controller implementation from creating an array of DSP processors to utilizing a single FPGA chip.

The proposed architecture ensures a balanced combination of parallel and sequential operations. The main advantage of this architecture is that the required number of DSP slices is proportional to the numbers of loudspeakers and reference microphones, which are often less than the number of error microphones and much less than the orders of the filters. Thus, the required number of DSP slices can be available even in low-cost FPGAs.

ACKNOWLEDGMENT

The reported study was supported by RFBR, research project No. 14-07-31091 мол_а.

REFERENCES

1. C.H. Hansen et al, *Active Control of Noise and Vibration*, 2nd ed., CRC Press, 2012.
2. S.M. Kuo and D.R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations*, John Wiley & Sons, 1995.
3. B. Widrow and S.D.Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.
4. Ushenina I.V. Mathematical model of an active noise control system working with bandlimited random noise // http://sibac.info/conf/tech/lii/43390.
5. D.R.Morgan and D.A. Quinlan, "Local silencing of room acoustic noise using broadband active noise control", Applications of Signal Processing to Audio and Acoustics, 1993. Final Program and Paper Summaries., 1993 IEEE Workshop on. IEEE, 1993.
6. X. Qiu, N. Li, G. Chen, and C. H. Hansen, "The implementation of delayless subband active noise control algorithms", in Proceedings of the 2006 International Symposium on Active control of Sound and Vibration, September, 2006.

7. X. Qiu, N. Li, G. Chen, “Multiprocessor DSP Systems for Active Control”, in Proceedings of 18th International Congress on Acoustics, 2004.
8. Texas Instruments Inc, http://www.ti.com/lsds/ti/processors/dsp/overview.page
9. 7 Series FPGAs Overview, http://www.xilinx.com/ support/documentation/ data_sheets/ds180_7Series_Overview.pdf
10. High-Level Implementation of Bit- and Cycle-Accurate Floating-Point DSP Algorithms with Xilinx FPGAs, http://www.xilinx.com/support/documentation/ white_papers/wp409_Floating_Point_DSP_Algorithms.pdf
11. Ushenina I.V. FPGA implementation techniques for multi-channel adaptive FIR filters of active noise control systems // Digital Signal Processing. – 2015. – № 4. – p.49-54.
12. Hawkes, G.C., *DSP: Designing for Optimal Results. High-Performance DSP Using Virtex-4 FPGAs*, Xilinx, 2005.
13. Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics, http://www.xilinx.com/support/documentation/data_sheets/ ds181_Artix_7_Data_Sheet.pdf
14. Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics, http://www.xilinx.com/support/documentation/data_sheets/ ds182_Kintex_7_Data_Sheet.pdf
15. Virtex-7T and XT FPGAs Data Sheet: DC and AC Switching Characteristics, http://www.xilinx.com/support/documentation/data_sheets/ds183_Virtex_7_Data_Sheet.pdf
16. 7 series DSP48E1 Slice User Guide, http://www.xilinx.com/support/documentation/ user_guides/ug479_7Series_DSP48E1.pdf
17. 7 series FPGAs Memory Resources User Guide, http://www.xilinx.com/support/ documentation/user_guides/ug473_7Series_Memory_Resources.pdf