Load-Balanced Implementation of a Delayless Partitioned Block Frequency-Domain Adaptive Filter

M. Fink, S. Kraft, M. Holters, U. Zölzer

Dept. of Signal Processing and Communications
Helmut-Schmidt-University
Holstenhofweg 85
22043 Hamburg
mfink, skraft, hol@hsu-hh.de

Abstract: In this paper, a load balanced implementation of a delayless FxLMS algorithm for the purpose of active noise cancellation is proposed. Frequency-domain adaption algorithms using FFT's are well-known for their efficiency. However, their block-based character will lead to a delay in the order of the used block length. A hybrid adaption approach is chosen in this study to combine the advantages of frequency- and time-domain processing. To allow the usage of the proposed system on various platforms, including single-threaded embedded and real-time systems, a manual yet flexible approach of load-balancing the block-based operations is demonstrated. This successfully reduces the peak-to-average computational load and allows an optimized utilization of the available processing power.

1 Introduction

The environmental noise polluting the life of humans has been increasing for decades and impacts the living quality by causing stress or even serious illness. For the purpose of attenuating such kind of noise the technique of *active noise cancellation* (ANC) is known. The cancellation is achieved by emitting phase-inverse anti-noise to eliminate the actual noise by destructive interference at a specific position, called sweet spot. To generate the anti-noise, the noise has to be captured, inverted, phase shifted, and amplified according to the position of the noise source and the anti-noise actuator.

An exemplary ANC system is shown in Fig. 1. The noise source on the left is captured by a reference microphone. The acoustic channel between the reference microphone and the sweet spot is defined as primary path $P(s)$ in the following, whereas $S(s)$ describes the corresponding acoustic channel between the anti-noise actuator and the sweet spot. The transfer function of $P(s)$ and $S(s)$ have to be measured in advance and are used to model the stationary acoustical environment. With the knowledge of both transfer functions the frequency response $W(z)$ of the digital FIR filter can be tuned to create an anti-noise signal which is then usually emitted by a loudspeaker system. This yields a cancellation of the noise source at the sweet spot. An optional error microphone can be used as a feedback control to automatically adjust the filter over time.

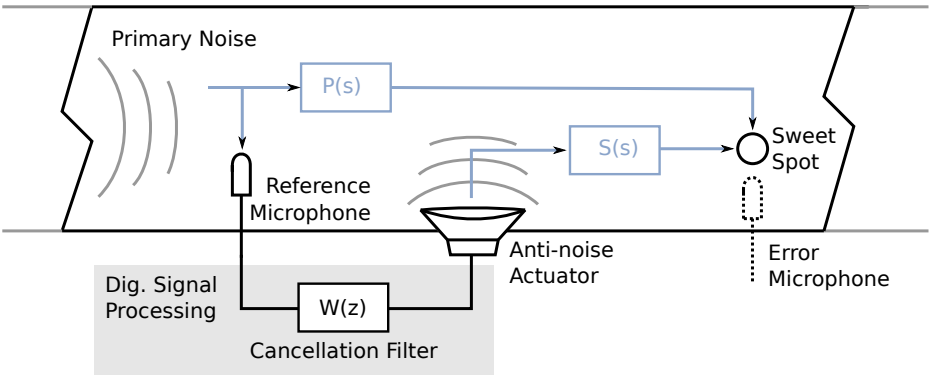


Figure 1: Example of noise cancellation in an air condition system.

The cancellation filter $w(a)$ can easily reach a length of several thousand samples and a direct form FIR filter implementation would need an unreasonable amount of processing power. The broadly-used fast convolution technique [OS09] could drastically reduce the computational demand but leads to a block delay of the same length as the filter. This delay and phase shift badly affects the success of the noise cancellation at the sweet spot. Although several techniques are known to partly reduce the delay of frequency-domain convolution including the Overlap-Add and Overlap-Save partitioning schemes [OS09], all these approaches will still introduce a delay of one partial block. To finally achieve a fast and efficient implementation but also to retain zero input-output delay, a hybrid convolution approach for the filter operation has to be used [ZCL07, BBSW01].

Measurements of the algorithm's runtime reveal a strong peak load being up to 200 times higher at the first sample of each block than the mean load. This requires the selection of a strong processor which however will be fully loaded only for a small fraction of time. In the case of multi-channel ANC systems with tens of convolutions it may even be impossible to find a processor which is capable to run the program in real-time.

To solve this problem and to efficiently use the available processing power, a state machine based scheduler is proposed in this paper to uniformly distribute the load of the block-based frequency-domain calculations.

The hybrid fast convolution is described in Sec. 2 and the *Filtered x Least Mean Squares* (FxLMS) adaption algorithm is explained in Sec. 3. Combining the FxLMS and hybrid convolution yields a *Delayless Partitioned Block Frequency-Domain Adaptive Filtering* (DPBFDAF) system which is illustrated in Sec. 4. Further steps to achieve the load-balanced implementation are explained in Sec. 5 and Sect. 6 concludes this study and shows possible extensions of the system.

2 Hybrid Fast Convolution with Zero Delay

Gardner first presented a real zero delay fast convolution approach [Gar95]. It is the combination of a time-domain filter and an overlap-save convolution in the frequency-domain. The basic block diagram is depicted in Fig. 2. Gardner partitioned the frequency coefficients in nonuniform blocks of increasing length to achieve the best possible performance gain. In the following we decided to use slightly less efficient uniform block lengths for simplicity and easier integration in the desired ANC application.

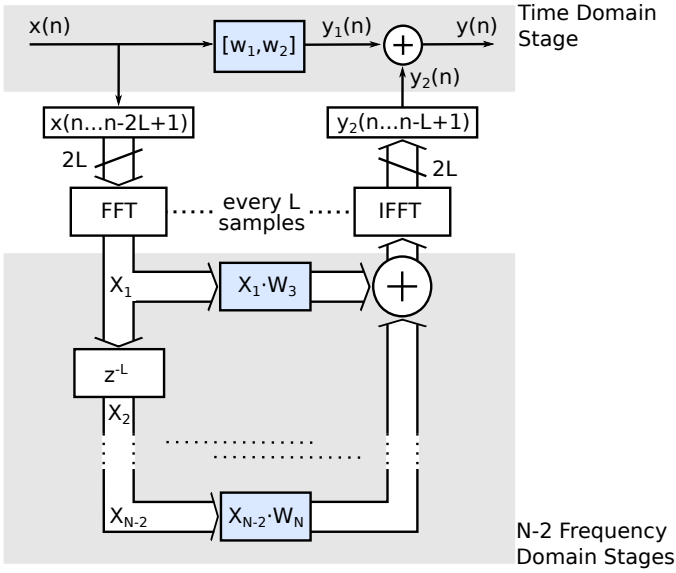


Figure 2: Hybrid Fast Convolution: Partition of filter coefficients $w(a)$ into the time domain parts $[w_1, w_2]$ and the frequency domain stages $[W_3, \ldots, W_N]$. Note, that the frequency and filter index parameters are omitted to improve clarity

The A filter coefficients $w(a)$ are partitioned into $N = \lceil A/L \rceil$ blocks $[w_1, \ldots, w_N]$, where each block is of length L. Zero-padding is optionally applied to completely fill the last block. The first two blocks $[w_1, w_2]$ are concatenated to form the coefficients of the time-domain filter. The remaining fractions $[w_3, \ldots, w_N]$ are zero-padded to twice their length and are individually transformed to the frequency-domain yielding the spectra $[W_3, \ldots, W_N]$.

The input sequence $x(n)$ is fed to the resulting separate processing paths. On the one hand, $x(n)$ passes the time-domain filter. On the other hand, every L samples the last $2L$ buffered samples of $x(n)$ are transformed into the frequency-domain. In the first frequency domain filter stage the resulting spectrum X_1 is multiplied by the third partition W_3. Furthermore, the input spectrum X_1 is continuously delayed by L samples and fed to

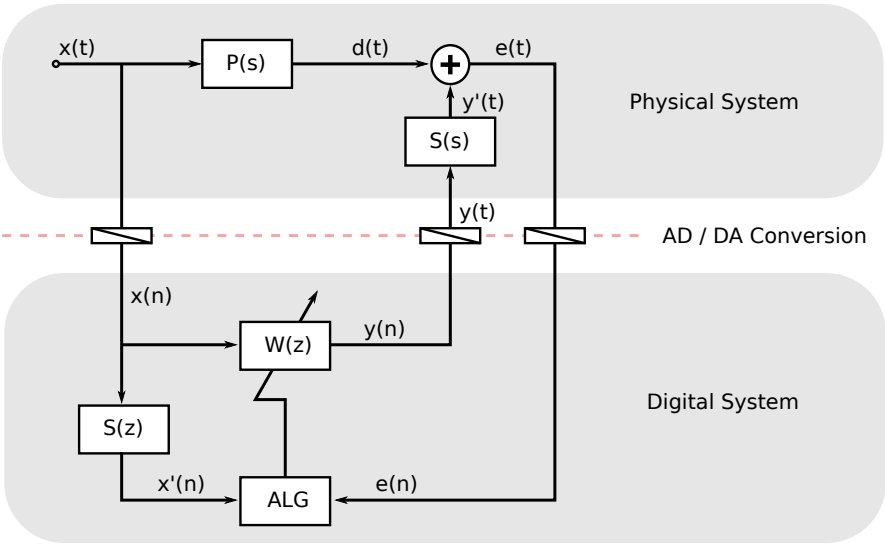


Figure 3: Single channel FxLMS divided into the continuous physical system and the discrete digital adaption algorithm.

the following frequency-domain stages where it is multiplied with the remaining frequency domain blocks of the filter.

The output of all frequency-domain stages is summed up and inversely transformed back to the time-domain. Since the fast convolution is implemented with the overlap-save method, the first half of the resulting time-domain of length $2L$ is omitted, resulting in $y_2(n, \ldots, n - L + 1)$. This block of time-domain samples is serialized and summed with the output samples of the time-domain filter $y_1(n)$ to compute the overall result $y(n)$.

3 FxLMS

The *Filtered x Least Mean Squares* (FxLMS) method is a well-known algorithm to perform *active noise cancellation* (ANC) with an adaptive filter [KM99, Dou99]. A single channel FxLMS system is illustrated in Fig. 3 and can be divided in two subsystems. The upper part is the physical system which is modeled by the continuous transfer functions $P(s)$ and $S(s)$ of the primary and secondary path. In the lower part the processing is performed in the digital domain with the sampled noise source $x(n)$ and error signal $e(n)$ as input and anti-noise $y(n)$ as output. Also the z-transform $S(z)$ of the secondary path transfer function is taken into account for the adaption algorithm. Physical and digital system components are separated by an AD/DA conversion layer.

3.1 Physical System

The noise at the sweet spot

$$d(t) = x(t) * p(t) \tag{1}$$

should be eliminated and can be described by convolving the noise source $x(t)$ with the impulse response $p(t)$ of the acoustic channel between source and sweet spot. The cancellation of $d(t)$ is achieved by emitting a phase-inverse anti-noise signal $y(t)$ by a cancellation actuator, which is a loudspeaker in the most cases. The anti-noise traverses the acoustic channel between the position of the cancellation actuator and the sweet spot — described by the secondary path transfer function $S(s)$ — and so the effective anti-noise at the sweet spot

$$y'(t) = y(t) * s(t) \tag{2}$$

is the actuator signal $y(t)$ convolved by the impulse response of the secondary path $s(t)$. The overall goal is to achieve a complete elimination of the residual signal

$$e(t) = d(t) + y'(t) = x(t) * p(t) + y(t) * s(t) \tag{3}$$

which is the sum of the noise $d(t)$ and anti-noise signal $y'(t)$, by destructive interference.

3.2 Digital System

The digital anti-noise signal

$$y(n) = x(n) * w(a) \tag{4}$$

is the result of the convolution of the digitized noise source $x(n)$ with the coefficients $w(a)$ of the filter $W(z)$. The A digital filter coefficients $w(a)$ have to be tuned in such a way that $y(n)$ minimizes the error energy at the sweet spot. Looking at Eq. (3) reveals that $e(t)$ directly depends on $y(n)$ and therefore is indirectly influenced by the filter operation. To adapt to changing characteristics of the noise source or the room it is necessary to have a time-variant set of filter coefficients $w(a, n)$.

To update the filter coefficients $w(a, n)$ for the next time step $n + 1$, the method of steepest descent is applied. The negative gradient of the mean square cost function $J(n) = E[e^2(n)]$ — approximated by the instantaneous error energy $e^2(n)$ — is weighted with the gradient step size $\frac{\mu}{2}$ and yields the new coefficient set

$$w(a, n+1) = w(a, n) - \frac{\mu}{2} \frac{\partial J(n)}{\partial w(a, n)}. \tag{5}$$

Solving the derivative using Eq. (1,2,3,4) yields

$$w(a, n+1) = w(a, n) - \mu\, x'(n-a+1)\, e(n), \quad a \in [1, \ldots, A]. \tag{6}$$

It becomes apparent, that only the digitized noise convolved with the impulse response of the secondary path $x(n)' = x(n) * s(n)$ and the digitized error signal $e(n)$ are inputs of the actual adaption algorithm. To improve the robustness of $w(a, n)$ against variations of the signal energy, a power normalized μ is used

$$\mu' = \frac{\mu}{\sum_{a=1}^{A} x'(n-a+1)^2}. \tag{7}$$

Note, that the speed of convergence of the adaption mainly depends on the gradient step size μ. However, choosing μ inappropriately high leads to instability of the algorithm.

4 DPBFDAF

For an efficient implementation, the manifold convolutions and the adaption of the FxLMS in Sec. 3 can be performed in frequency-domain. By the use of a common fast convolution an adverse delay of one block length on the path through the filter $W(z)$ is introduced. Using the described hybrid convolution of Sec. 2 for this certain operation removes the delay and basically yields the *Delayless Partitioned Block Frequency-Domain Adaptive Filtering* (DPBFDAF) algorithm.

The filter adaption is also moved from the time- into the frequency-domain and is now similar to the one described in [SP90].

Every L samples, a block of $2\,L$ last values of the input signal $x'(n)$ is transformed into the frequency-domain and shifted to a buffer $X'(k, b)$, where k denotes the frequency index and b the block index. Furthermore, the result from the frequency domain convolution parts

$$Y_2(k) = \sum_{b=1}^{N-2} X'(k, b) \cdot W(k, b+2) \tag{8}$$

is transferred back to time-domain and serialized for summation with the time domain part $y_1(n)$.

For the filter adaption, the energy computation of $x'(n)$ in Eq. (7) is approximated in the frequency-domain with a recursive average, as shown by

$$P(k) = \lambda\, P(k) + (1-\lambda)\, |X'(k, 1)|^2, \quad k \in [1, \ldots, 2L], \tag{9}$$

where λ controls the influence of prior values.

The adaption Eq. (6) to yield the updated filter coefficients $W'(k,b)$ in the frequency-domain is described as

$$W'(k,b) = W(k,b) + \mu \cdot \frac{X'(k,1)^* \cdot E(k)}{P(k)}, \tag{10}$$

where $E(k)$ is the spectrum of the internally buffered last $2\,L$ error samples and $X'(k,1)^*$ is the complex conjugate of $X'(k,1)$. To remove the cyclic fraction of the convolution, resulting in time-domain aliasing, in the second half of the spectra in $W'(k,b)$, the whole matrix is transformed back to the time-domain

$$w'(a,b) = \mathcal{F}^{-1}\{W'(k,b)\}, \quad a \in [1,\ldots,2L], \tag{11}$$

and the upper L samples of every block are set to zero

$$w'(a,b) = 0, \quad a \in [L+1,\ldots,2L]. \tag{12}$$

Reapplying the FFT yields the filter spectra $W(k,b)$ for the filtering of the next block. To update the time-domain coefficients, the lower L entries of $w(a,1)$ and $w(a,2)$ are directly assigned to the filter coefficient sets w_1 and w_2

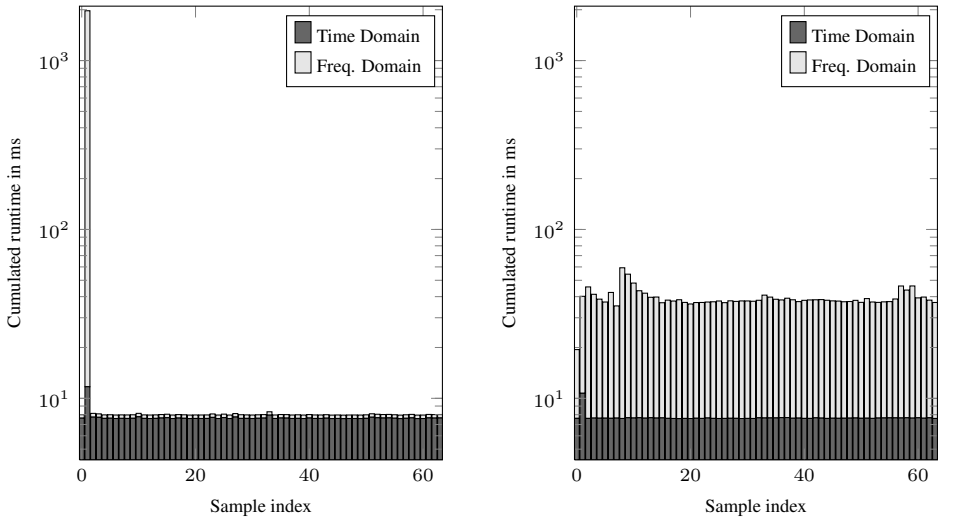
$$w_1(a) = w(a,1), \quad a \in [1,\ldots,L] \tag{13}$$
$$w_2(a) = w(a,2), \quad a \in [1,\ldots,L]. \tag{14}$$

5 Load-Balanced Implementation

The described implementation successfully implements a delayless partitioned frequency-domain adaptive filter. One remaining problem for a real-time implementation is the highly unbalanced processing load. The cumulated processing time per sample with a block length of $L = 64$ is shown in Fig. 4 a). An overall time of about $2000\,\mathrm{ms}$ is spent on the first sample of each block, whereas the other samples only occupy a runtime below $8\,\mathrm{ms}$ each. Applying the latterly explained load-balancing will compensate the vast runtime difference between the first sample of a block and the rest, yielding a nearly constant load as depicted in Fig. 4 b). The mean runtime per sample is around $40\,\mathrm{ms}$ and the biggest peak with $60\,\mathrm{ms}$ is just slightly above. This allows the use of a much cheaper processor or alternatively running more channels in parallel to enhance the ANC system to multi-channel variants.

Taking a closer look at the algorithm from the previous section, one can see that the result of the filter adaption is not needed immediately at the beginning of a block but just right before the next block starts. So the result of the frequency-domain filters and the adaption of the filter W can be performed in parallel while the time-domain filter processes the next N samples.

A more detailed and implementation based view of this relationship inside the DPBFDAF system can be seen in Fig. 5. Obviously, only the input x, the partial and overall results



(a) non-balanced implementation

(b) load-balanced implementation

Figure 4: Accumulated processing time spent per sample index with a length $L = 64$ of the blocks.

y_1, y_2, y, and the error signal e are processed sample-wise. The remaining signals are processed in the frequency-domain while the time-domain filtering takes part and are only connected by the `BLOCK_SYNC` event. Since the DPBFDAF system is supposed to work on several platforms, including embedded systems or DSP's, the simultaneous execution of time-domain and frequency-domain parts can't be inherently established by using threads. Instead, the load-balancing was achieved with the help of a state-machine and a predefined distribution of the processing load. Therefore, the complete frequency-domain processing was manually divided into M_S subtasks and every single subtask was assigned to a state. The guide line for the division into subtasks was to avoid subtasks requiring more than L operations of any kind. Whenever this criteria was not met, the corresponding subtask was further divided.

The state machine implementation of the frequency-domain processing is illustrated with the pseudo code in Alg. 1. Depending on the current state, the corresponding subtask in the switch statement (line 1) is processed. A subtask itself could be further split into smaller chunks which are then processed by repeated calls of the state, or case respectively, until it reports its completion (see lines 8-10). The state counter is incremented subsequently (line 4 and line 9). For example, a matrix multiplication is handled as a single task which is split up in various vector products. A single call of the adapt function then only computes a single vector product. Also the most prominent digital signal processing algorithm, the *Fast Fourier Transform*, is similarly partitioned. Instead of computing the complete spectrum at once - requiring the complexity $O(L \log L)$ - the proposed implementation

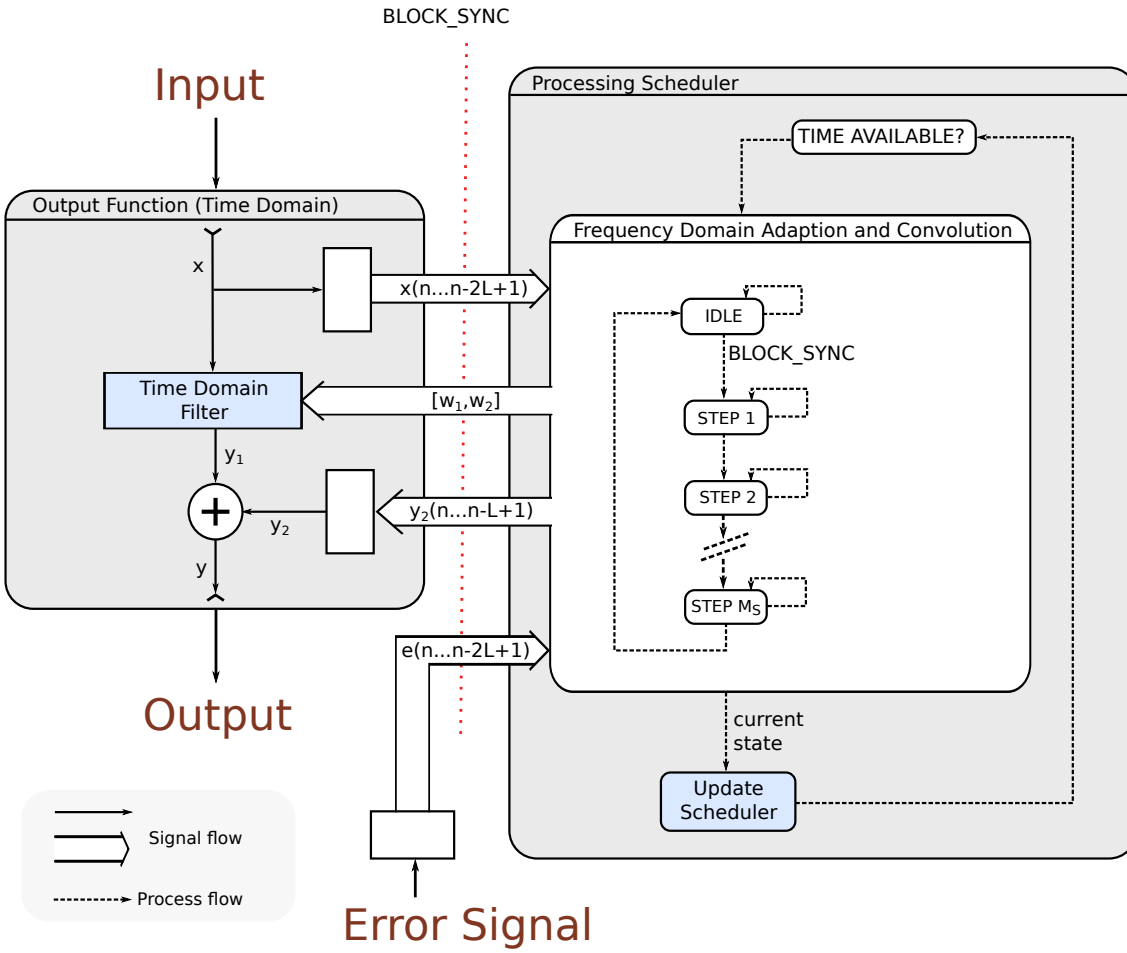


Figure 5: Signals and processes of the overall DPBFDAF system.

Algorithm 1 function adapt:

```
switch (currentState)
case STATE 1:
    processingStep1();
    currentState++;
    break;
case STATE 2:
    processingStep2();
    if Step2Finished() == true then
        currentState++;
    end if
    break;
    ⋮
case STATE M:
    processingStepM();
    currentState = IDLE;
    break;
case IDLE:
    break;
end switch
```

Algorithm 2 function scheduler:

```
while currentState ≠ IDLE and timeRemaining > 0 do
    adapt(currentState);
    updateTime(currentState);
end while
```

divides the FFT in $\log L$ subtasks. So every call only executes one subtask or chunk before it passes the control back to a scheduler.

The scheduler is aware of the overall time available for frequency-domain operations per sample and the current time already spent for a sample. The basic scheduler mechanism is depicted in Alg. 2. As long as the current state is valid and there is processing time available (line 1), the scheduler calls the adaption function which performs the instructions for the corresponding state (line 2). Afterwards, the scheduler refreshes the internal variable *timeRemaining* with an update function (line 3). The time update is done according to the computational costs of the lastly processed state. Therefore, the varying costs of the different states have to be known within the scheduler. It turned out that it was quite easy to divide the overall algorithm in subtasks and chunks with near equal processing time per single step. So we could simply assume a virtual cost of 1 time unit for a single call to the state machine. The number of samples per block L and the number of necessary calls to run the whole state machine M_C is known in advance. From these values the amount of available time units per sample $T = \lceil {}^{M_C}/_{L} \rceil$ can be retrieved and is used to initialize the scheduler. Fig. 4 b) demonstrates that even such a simple determination of the costs yields a uniform distribution with a small variance.

Whenever a state is successfully processed, the state machine falls into the next state and a call of the adapt function yields in computing the instructions of the next state. When the processing time for the current sample is expired, the next sample is processed with the time-domain filter and afterwards, the scheduler is called again, as illustrated in Fig. 5. This procedure repeats until all states are processed and the state machine remains in the `IDLE` state (see, Alg. 1 line 16).

Every L samples the so-called `BLOCK_SYNC` event is triggered. This event causes a synchronization of all signal blocks required in the different processing domains. For example, the updated filter coefficients $[w_1, w_2]$ are copied from the frequency-domain to the time-domain part of the DPBFDAF system. Besides, the current state is set from `IDLE` to the first actual state to reset the state machine and trigger the calculation of the next block in frequency-domain.

The overall system was not directly evaluated with measurements, but the simulation of the presented system still holds the test requirements of a previous system without load-balancing and based on a pure fast convolution. Hence, the new system still shows the same performance of noise cancellation and is additionally improved in terms of efficiency and delay.

6 Conclusions

An exemplary implementation of a delayless frequency-domain adaptive filter for the purpose of ANC was presented in this study. It is based on the FxLMS adaption algorithm and hybrid convolution to perform delayless, yet fast convolution. The computational load of the overall system was equally distributed over the samples of a block with the help of a state machine approach. The available time per sample is retrieved dynamically depending on the block length and partitioning scheme under the assumption of equal costs per state machine call. A possible extension would be a profiling step in the initialization to measure the exact cost distribution for an optimal allocation of the computational resources and an extension towards multichannel processing to allow the application of the proposed system in more complex scenarios.

References

[BBSW01] Y. Bendel, D. Burshtein, O. Shalvi, and E. Weinstein. Delayless frequency domain acoustic echo cancellation. *Speech and Audio Processing, IEEE Transactions on*, 9(5):589–597, 2001.

[Dou99] S.C. Douglas. Fast implementations of the filtered-X LMS and LMS algorithms for multichannel active noise control. *Speech and Audio Processing, IEEE Transactions on*, 7(4):454–465, 1999.

[Gar95] William G. Gardner. Efficient convolution without input/output delay. *Journal of the Audio Engineering Society (JAES)*, 43(3):127–136, 1995.

[KM99] S.M. Kuo and D.R. Morgan. Active noise control: a tutorial review. *Proceedings of the IEEE*, 87(6):943–973, 1999.

[OS09] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[SP90] J.-S. Soo and K.K. Pang. Multidelay block frequency domain adaptive filter. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(2):373–376, 1990.

[ZCL07] Yin Zhou, Jialu Chen, and Xiaodong Li. A Time/Frequency-Domain Unified Delayless Partitioned Block Frequency-Domain Adaptive Filter. *Signal Processing Letters, IEEE*, 14(12):976–979, 2007.