Correctly Sizing FIR Filter Architecture in the Framework of Non-uniform Sampling

Jean Simatic
and Laurent Fesquet
Univ. Grenoble Alpes, TIMA,
46 rue Felix Viallet,
38031 Grenoble Cedex, France
Email: Jean.Simatic@imag.fr
Laurent.Fesquet@imag.fr

Brigitte Bidegaray-Fesquet
Univ. Grenoble Alpes, LJK,
BP. 53, 38041 Grenoble Cedex 9, France
Email: Brigitte.Bidegaray@imag.fr

***Abstract*—Based on non-uniform sampling techniques and event-driven logic, signal processing is evolving to integrate new demands such as power consumption. As power is mainly connected to the processing activity and data volume, the level-crossing sampling scheme offers a simple way to reduce data volume and consequently processing activity. Nevertheless, these good properties could be constraining for the designers because of the non-predictable sample number that can be involved in the processing. In this paper, we target a FIR filter architecture and show how to correctly size its input shift-register. This paper shows a strategy to choose the shift-register depth but also a way to dynamically adapt the computation to an heterogeneous data flow.**

I. Context and Introduction

Today, our digital society exchanges data flows as never it has been the case in the past. The amount of data is incredibly large and the future promises that not only humans will exchange digital data but also technological equipments, robots, etc. We are close to open the door of the Internet of Things. This data deluge will waste a lot of energy. There already exists a lot of design solutions to enhance the energetic performances of the electronic systems and circuits [5]. Nevertheless, another way to reduce energy is to rethink the sampling techniques and digital processing chains [8]. In order to mitigate the power consumption, a simple idea based on [1] proposes to design digital filters with a non-uniform level-crossing sampling scheme and event-driven logic. It has been shown in [7] that this approach allows to gain up to two orders of magnitude when filtering sporadic signals, compared to a uniform filtering technique. When designing the hardware, one of the challenges is to correctly size the architecture of the filters because of the non-predictable number of samples produced by the non-uniform Analog-to-Digital conversion. This paper takes FIR filters as an example to show how to adapt the FIR architecture to obtain an optimal behavior of the filters.

Designing FIR filters with a non-uniform sampling scheme is not a new technique ([4], [6], [9], [10], [11]) but this implies to carefully choose the variable parameters of the filter architecture. Indeed, the input samples are stored in a shift register which, contrarily to the classical sampling, has not a fixed size when a non-uniform sampling is used. The consequence on the filter design is obvious: a correct sizing will guarantee a perfect behavior but an incorrect will show degraded filtering performances. In the sequel, the impact on the filtering is shown and a method is proposed to size the shift register.

II. Principles

A. NUS FIR algorithm

The principles presented here refer to the paper of Aeschlimann et al. [1]. Let x be the input signal and h be the impulse response with support $[0, T_h]$. For continuous times, the output y is the convolution product of x and h:

$$y(t) = \int_0^{T_h} h(s)x(t-s)\ ds. \tag{1}$$

If the signals are uniformly sampled with a period $T_s = T_h/N_h$, we have

$$y_k = \sum_j T_s h_j x_{k-j},$$

where y_k, h_k and x_k are the values of the signals sampled at kT_s. This classic formula can be rewritten using piecewise constant interpolation. Indeed, signals u (h or x) are interpolated to a piecewise constant continuous time functions $\tilde{u}$ and equation (1) reads at times kT_s

$$y_k = \sum_j T_s h_j x_{k-j} = \int \tilde{h}(s)\tilde{x}(kT_s - s)\ ds. \tag{2}$$

Figure 1 gives a graphical view of the computation of Equation (2) where the signal $\tilde{x}$ has been reversed in time, yielding $R_k(\tilde{x})$, to write the convolution as a scalar product.

For non-uniform sampled signals, we extend the convolution product using the interpolated versions of the signals. More precisely, a the non-uniformly sampled input signal x is described by a sequence of couples $(x_i, \delta t_i)$ where $x_i = x(t_i)$ is the sampled amplitude at time t_i of the signal and $\delta t_i = t_i - t_{i-1}$ is the delay between the previous sample and the current sample. A right piecewise constant interpolation is

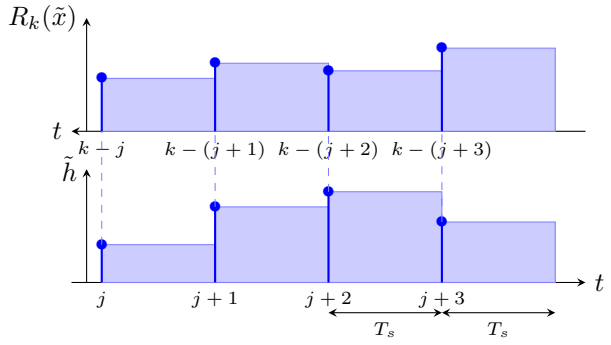

Fig. 1. Computation of an output sample in the uniform-sampling scheme.

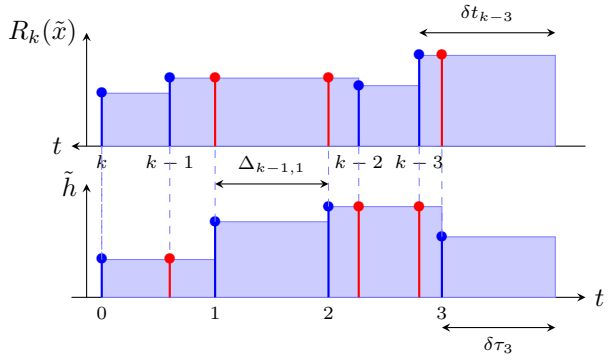

Fig. 2. Computation of an output sample in a non-uniform sampling scheme. Blue bins describe initial input samples. Red bins describe recomputed input signal to match interpolation times of both signals.

used for the input signal ($\tilde{x}(t) = x_i$ if $t_{i-1} < t \leq t_i$). Similarly the impulse response is described by samples $(h_j, \delta\tau_j)$, and a left piecewise constant interpolation is used for h, in order that both signal are interpolated on the same side once the input signal x is reversed in time to compute the convolution. Contrarily to uniform sampling, the signal and the impulse response are not defined at the same times and Figure 2 presents graphically the convolution computation in the nonuniform context. For this computation, the integral occurring in Equation (1) is decomposed on domains of width $\Delta_{i,j}$ on which both x and h are constant:

$$\Delta_{i,j} \stackrel{\text{def}}{=} \max(0, \min(t_{i-1}, \tau_j) - \max(t_i, \tau_{j-1}))$$

Then, the amplitude y_k of y at time t_k (same as for the input signal) is

$$y_k = \sum_{i,j} \Delta_{k-i,j}\ x_{k-i} h_j.$$

In [2] an iterative algorithm is presented to perform this computation only using delays. Figure 3 describes this algorithm which after the capture of the k-th sample of x, namely $(x_k, \delta t_k)$, returns the k-th sample of y, namely $(y_k, \delta t_k)$. We re-index the x samples: x'_0 indicates the latest sample amplitude (originally x_k), x'_1 indicates the latest but one and so on. This transformation, which corresponds to the actual

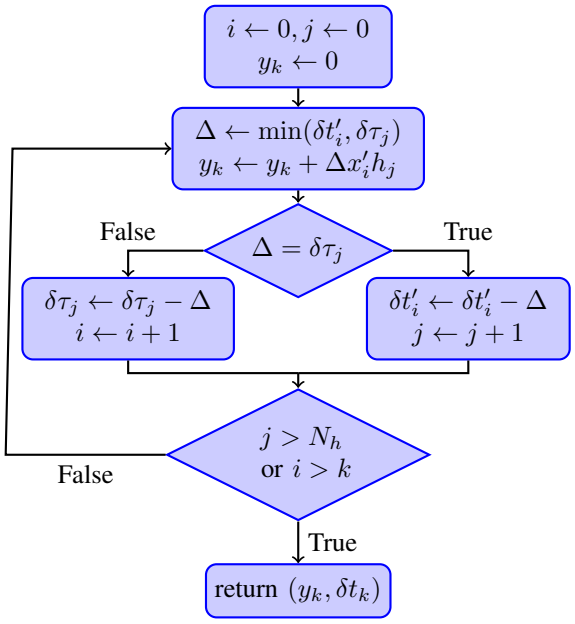

Fig. 3. Algorithm to compute an output sample.

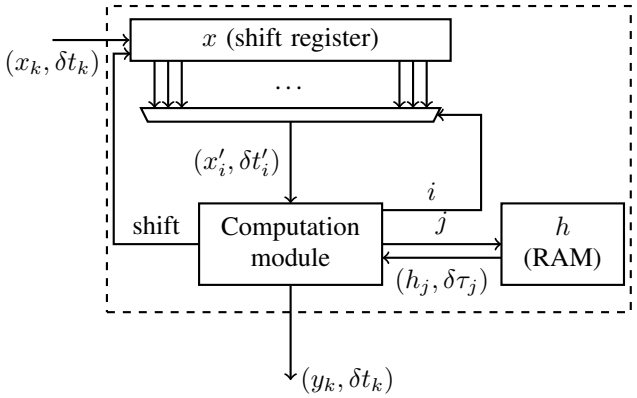

Fig. 4. High-level view of the filter architecture.

storage in the shift register, allows to remove all reference to k in the algorithm meaning that the implementation can be independent of the number of received samples.

We can obtain a more precise computation if the samples are interpolated at higher order, leading to an algorithm which structure is very similar to the previous one [2]. It is implemented in the SPASS toolbox [3], for interpolation up to order 3.

B. FIR Architecture

We implement this algorithm using event-based logic to match the event-based nature of the sampling. Figure 4 gives a high level view of the filter. The signal x is stored in a shift register and h is stored in a RAM. Control signal i (resp. j) grants access to $(x'_i, \delta t_i)$ (resp. $(h_j, \delta\tau_j)$). When $i = 0$, the computation module accesses the left-most element of the shift register which is the latest sample.

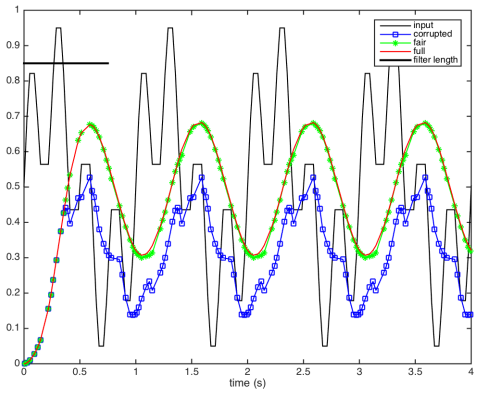

Fig. 5. FIR filtering degradation when the register number is not sufficient.

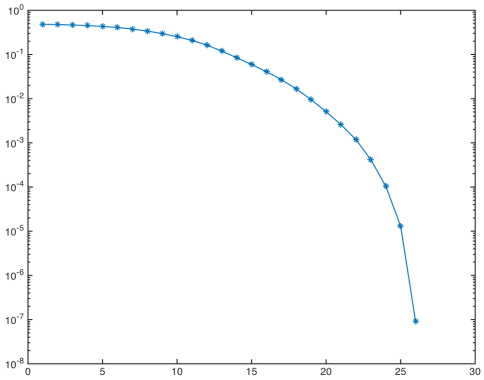

Fig. 6. ℓ^1 error between full filtered signal y and y^M.

The designer chooses the function h and its samples according to the desired transfer function. In the following sections, we discuss the impact of the size of the shift register on the filtering quality.

III. Filtering degradation

We first choose a sum of two sinusoids as input signal

$$x(t) = \frac{1}{2} + \frac{1}{4}\sin(2\pi t) + \frac{1}{4}\sin(4 \cdot 2\pi t). \tag{3}$$

The A-ADC is setup to sample data in the range 0 to 1 V with 8 equally spaced thresholds between 0.05 and 0.95 V. We choose a filter with a cut frequency of 2 Hz and order 5. The horizontal straight line on Figure 5 represents T_h, the temporal expansion of the filter. We started from the algorithm initially developed by Aeschlimann et al. [1] where all the necessary samples are kept for the processing. In our case, we take into account the shift register size M which limits the number of samples of x that can be used. This amounts to replace the test "$i > k$" by "$i > \min(k, M)$" in Figure 3. Then the computation results are evaluated for different values of M:

- M is large. The computation is identical to the theoretical convolution formula because the sample storage is not limited by the shift register depth (red curve on Figure 5).
- M is sufficient to guarantee in most cases a number of samples approximately covering the filter temporal expansion (green curve on Figure 5).
- M is too small. The filter temporal expansion is larger than that covered by the samples. This is due to a limited depth of the shift register. In this case, we clearly observe the output signal degradation (blue curve on Figure 5).

IV. Correctly sizing the architecture

The input signal (3) is described by 127 non-uniform samples stemming from the level crossing sampling of the signal. The algorithm in [1] has not addressed the problem of the number of samples really used that need to be stored in each single convolution computation.

For a periodic input signal such as (3), the delay between two samples is not varying much. Therefore we can think that a good approximation of M is the ratio of the length of the filter ($T_h = 0.75\,\mathrm{s}$) and the average sampling interval dt_{ave} which is the ratio of the total duration of the signal by the number of samples: $dt_{\mathrm{ave}} = 4/127 \simeq 0.03$. This leads to a value of $M_{\mathrm{ave}} = 24$ for signal (3). This is an average value, which means that the actual needed value to obtain the accurate filtering computation is larger. An upper bound would be given by the smallest delay between two samples ($9.2\,\mathrm{ms}$ here) which would lead to $M_{\max} = 82$.

In fact the values taken in Figure 5 are 12 for the corrupted result, but only 20 for the fair result given by the green curve, which is less than $M_{\max}$ (which we of course awaited) but also than M_{ave}. Figure 6 displays the error between the filtered signal y with the asynchronous FIR filter and the filtered signal y^M using M samples of the input signal. The norm used for the error is a ℓ^1 error

$$\frac{1}{T}\sum_k \delta t_k |y_k - y_k^M|,$$

where T is the total duration of the input signal.

In the former example we see that we can do better than the worst and also the average value of M. We therefore may ask what is an optimal value of M to obtain fair results. To do this perform an analysis on a frequency sweeping signal, keeping the other parameters unchanged:

$$x(t) = \frac{1}{2} + \frac{1}{2}\sin\left(\frac{\pi t^2}{2}\right). \tag{4}$$

This will allow us to increase progressively the number of samples needed to perform the computation and quantify the quality degradation and size the shift register accordingly.

Since we have seen in the previous test that we can use a smaller value of M than M_{ave}, we use $M = 0.9M_{\mathrm{ave}}$ for

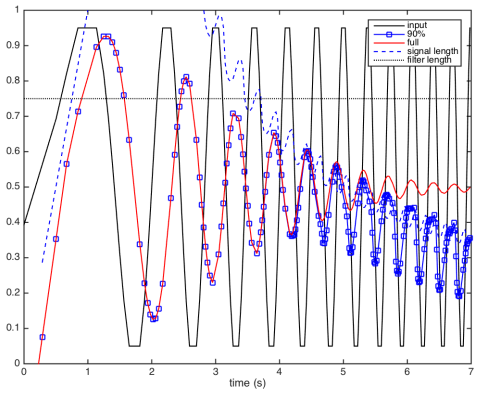

Fig. 7. FIR filtering of a sliding frequency signal with $M = 0.9M_{\text{ave}}$.

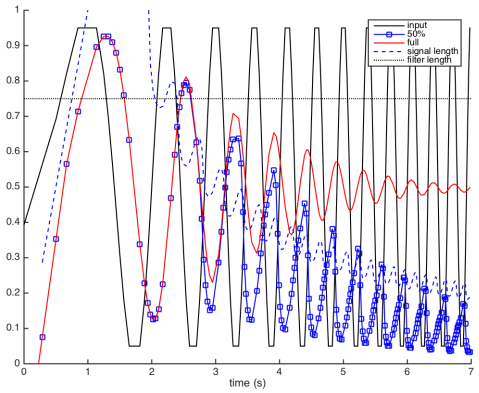

Fig. 8. FIR filtering of a sliding frequency signal with $M = 0.5M_{\text{ave}}$.

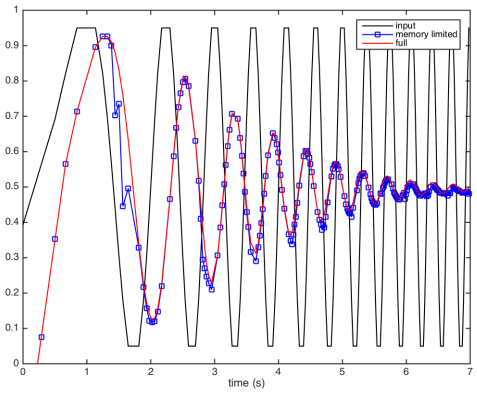

Fig. 9. Adaptive FIR filtering of a sliding frequency signal.

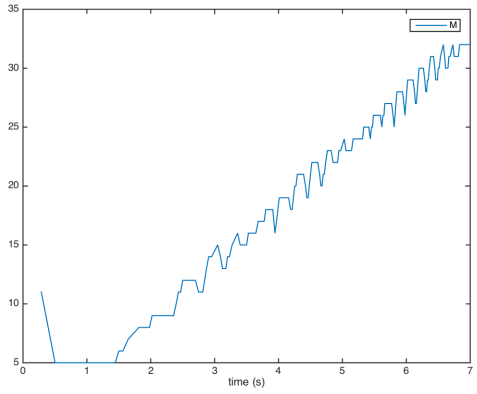

Fig. 10. Memory used for the adaptive FIR filtering test of Figure 9.

the test displayed on Figure 7 and $M = 0.5M_{\text{ave}}$ for the test displayed on Figure 8.

On both figures we display the input signal with a solid black curve, the filtered signal with a blue curve with square markers, the reference filtered signal (with all samples) in red. We also display the duration of the signal used for the convolution computation (dashed curve) and the length of the filter (dotted curve). We see in both cases that the filtering quality is altered when the length of the signal is lower than the length of the filter, but that we have some margin since its still works when the signal length is slightly below the filter length.

This leads to a refinement of the optimization of the memory size. We indeed want to use as few samples as possible at each step of the computation for consumption reduction purposes. We therefore propose an adaptive algorithm.

Instead of using a percentage of an averaged value of M, we use a percentage r of an M computed with the length of the previous computation. In Figures 7 and 8, we can compare the filter length and the length of the signal which is needed to yield a fair value of the filtered signal. We therefore choose $r = 80\%$. In order to be able to implement the algorithm on a hardware architecture, we also impose the number M not to increase by more than 1 at each step (otherwise we would have to use samples already out of the shift register). Hence at step k

$$M(k) = \min\left(r\frac{T_h}{dt_{\text{ave}}(k)}, M(k-1)+1\right). \qquad (5)$$

The filtered signal (as well as the input and reference filtered signals) is displayed on Figure 9. We see that this allow to recover a good result even for the higher frequencies.

The values of $M(k)$ are displayed on Figure 10. We see that memory is not always increasing although the frequency is, and that the constraint of not increasing more than by a unit at each step does not alter the result. The maximum value, obtained for the highest frequency, is $M = 32$. The average value is 21.38 which is less than if we had to filter a signal with this highest frequency as done in Section III.

Non-uniform sampling is especially interesting when the signal is sporadic, with large parts with no or little time evolution. Therefore the input signals used in this paper are not the ones for which the situation is the most advantageous.

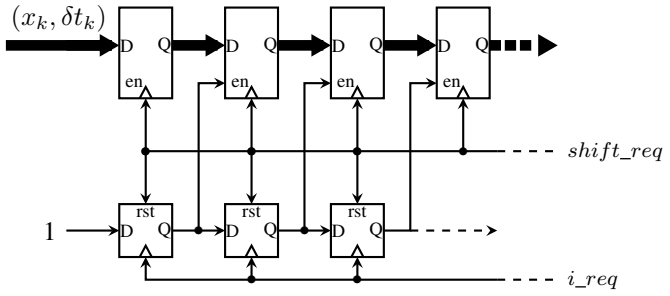

Fig. 11. Implementation details of the shift register.

Indeed, very few samples are used in the less active part of the input signal (4) which is very active compared to quiet parts of sporadic signals.

V. Implementation

In Equation (5), we compute dt_{ave} as the local mean over the $M(k)$ last samples.

$$dt_{\text{ave}}(k) = \frac{1}{M(k)} \sum_{i=0}^{M(k)-1} \delta t_{k-i}. \tag{6}$$

Equations (5) and (6) give the following inequality:

$$\sum_{i=0}^{M(k)-1} \delta t_{k-i} \leq rT_h. \tag{7}$$

It means that the computation can ignore all samples that are older than rT_h compared to the latest sample. In other words, the signal h can be truncated beyond this point. Thus we need no additional hardware material to implement the proposed sizing, it is sufficient to adequately setup h in the RAM.

However, without additional hardware, the FIR ignores the unused samples but those samples still propagate along the shift register and use power. We want to avoid keeping in the shift register samples that are too old to be useful. After an output sample is produced, only the input samples that were used to produce the output may be needed for computing the next output sample. We add to the shift register that stores the input samples a new shift register to indicate which samples are used or not. Figure 11 shows the corresponding circuit. The added shift register has a small footprint. It is only 1-bit wide.

On a rising edge on the $shift\_req$ signal, only the samples that are marked as used ("1") are shifted and all the marks are reset to unused ("0"). Each mark value must be used before it is reset but some can be reset before others are used. The shift signal drawing from right to left suggests an ordering.

Then, each time that we fetch a sample $(x'_i, \delta t'_i)$, there is a rising edge on i_req that shifts the marks so that one additional sample is marked as used.

This mechanism avoids activity on the unused parts of the shift register but it adds activity on the used part. The interest of the system depends on the usual occupation rate of the register. A rough approximation of the filter consumption is proportionnal to the algorithm iteration number. The latter is also approximatively proportionnal to T_h [2]. Thus, if $r = 0.8$ as presented above we can expect a 20% decrease of the power.

VI. Conclusion

The non-uniform sampling techniques coupled to event-driven logic is a key approach when data volume or power consumption have to be reduced. Nevertheless, the associated signal processing requires to rethink the processing architecture and to take into account the signal characteristics to customize the signal processing chain which becomes more application specific. In this paper, we showed, through a non-uniform FIR filtering example, that the standard processing chain can take advantage of a level-crossing sampling scheme. We are able to size the shift register depth according to the average number of samples, related to the activity of the signal. Moreover, we have been able to dynamically adapt the number of samples used in the shift register in order to mitigate the data volume. This is useful for reducing the power because, thanks to event-driven logic, the unused registers are not activated. Indeed, only the processing part of the event-driven logic is consuming power.

Acknowledgment

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01).

References

[1] F. Aeschlimann, E. Allier, L. Fesquet, and M. Renaudin, *Asynchronous FIR filters: Towards a new digital processing chain*, 10th International Symposium on Asynchronous Circuits and Systems (Async'04) (Hersonisos, Crete), IEEE, April 2004, pp. 198–206.

[2] F. Aeschlimann, *Traitement du signal échantillonné non uniformément : algorithme et architecture*, Ph.D. thesis, INP Grenoble, February 2006.

[3] B. Bidégaray-Fesquet and L. Fesquet, *SPASS 2.0: Signal Processing for ASynchronous Systems*, Software, May 2010. http://ljk.imag.fr/membres/Brigitte.Bidegaray/SPASS/

[4] B. Bidégaray-Fesquet and L. Fesquet, *Non-uniform filter interpolation in the frequency domain*, Sampling Theory in Signal and Image Processing **10**, 17–35, 2011.

[5] A. Chandrakasan and R.W. Brodersen (eds.), *Low-power CMOS design*, Wiley-IEEE Press, 1998.

[6] L. Fesquet and B. Bidégaray-Fesquet, *IIR digital filtering of non-uniformly sampled signals via state representation*, Signal Processing **90**, 2811–2821, 2010.

[7] Laurent Fesquet, Gilles Sicard, and Brigitte Bidégaray-Fesquet, *Targeting ultra-low power consumption with non-uniform sampling and filtering*, IEEE International Symposium on Circuits and Systems (ISCAS2010) (Paris, France), IEEE, May 2010, pp. 3585–3588.

[8] F.A. Marvasti, *Nonuniform sampling. theory and practice*, Information Technology: Transmission, Processing and Storage, Springer, 2001.

[9] S.M. Qaisar, L. Fesquet, and M. Renaudin, *Adaptive rate filtering: a computationally efficient signal processing approach*, Signal Processing, **94**, 620–630, 2014.

[10] B. Schell and Y. Tsividis, *A clockless ADC/DSP/DAC system with activity-dependent power dissipation and no aliasing*, IEEE International Solid-State Circuits Conference (ISSCC 2008) (San Francisco), IEEE, February 2008, pp. 550–551.

[11] C. Vezyrtzis, W. Jiang, S.M. Nowick, and Y. Tsividis, *A flexible clockless digital filter*, Proceedings of the ESSCIRC (Bucharest), IEEE, September 2013, pp. 65–68.