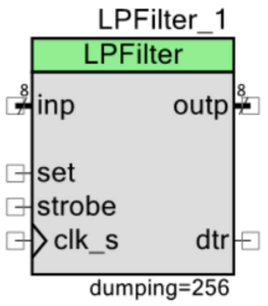
LPFilter: Exponential Moving Average Digital Filter

0.0

Features

- Implements exponential moving average digital filter.
- Arbitrary input and output bus width.
- Variable dumping length.
- Set (track) control option.
- Output decimator.
- Strobe sync.



General description

The LPFilter component represents standard exponential moving average filter[*] implemented in hardware. It can smooth stream of data coming from any digital source using exponential moving average algorithm [1, 2]. The filter output bus carries smoothed data, which can be fed to other components. Adjustable input and output bus width and several output options (single or dual bus) allow for easy interfacing with other PSoC components (ADC_SAR, VDAC8, Status register, BasicCounter, FIFOin[†], DDS24, etc.) or filter cascading.

When to use LPFilter component

Component was developed for conditioning of high-speed data coming from ADC_SAR, but can be used anywhere digital data require noise filtering or improving bit resolution. Demo project is provided.

[*] Also known as “recursive average”, “α-filter”, “leaky integrator”, etc.
[†] FIFOin is a part of the PSoC Sensei library by Brad Budlong.

Input-output connections

inp – digital input bus

Input bus for sampled digital stream. The bus width is user-selectable, valid range is 1 to 32. This pin is always visible. The pin doesn't have to be connected.

set – digital set input

Digital input pin for asynchronous set (track) control. Positive edge on this pin primes filter accumulator forcing filter output to instantly track the input. Use this pin to force faster settling of the output when the set cause event is clearly defined (for example when sequencing MUX has been switched). Visibility of this pin is controlled by the **set_enable** option in the Advanced dialog. If visible, the pin has to be connected to valid digital source.

strobe – digital strobe input

The input bus is sampled on the rising edge of the strobe signal. Valid digital source must be connected to this input. If source is asynchronous, then **sync_enable** option must be selected for internal synchronization of the strobe. This pin is always visible.

clk_s – sync clock input

Signal that strobe source is to be resynchronized against. This pin is visible when **sync_enable** option is selected. When visible, the pin must be connected to clock. The frequency of the clock source must be at least twice that of expected strobe frequency (higher is better).

outp – digital output bus

Filter output bus. Bus appearance is controlled by option **bus_type**. Depending on selection, the output can be either a single bus of variable width or fixed 16-bit (2x8-bit) bus. For **single_bus** option the width is user-selectable, valid range is [1 to 32]. For 2x8-bit bus type out0 represent the LSBs and out1 represents the MSBs. The output pin is always visible. The pin doesn't have to be connected.

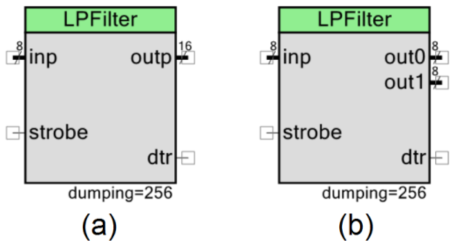


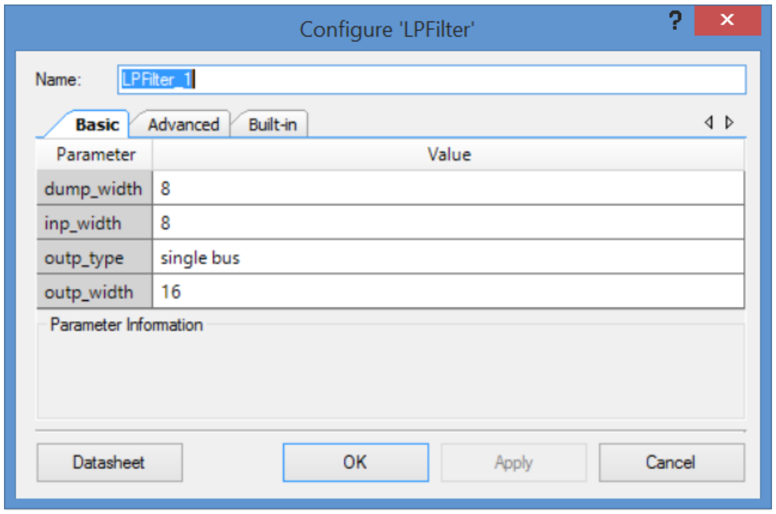
Figure 1. Output type: (a)-single bus of variable width, (b)-fixed 2x8-bit bus.

dtr – data ready digital output

Rising edge on this pin indicates that output data is ready. The pin is used to sync with other components or to generate an interrupt. When decimator is not used, the pin pulses on each strobe. If decimator is enabled, this pin pulses whenever decimator counter rolls over. The pin is always visible. The pin doesn't have to be connected.

Parameters and Settings

Basic dialog provides following parameters:



Name: LPFilter_1

Parameter	Value
dump_width	8
inp_width	8
outp_type	single bus
outp_width	16

dump_width (uint8)

A short name for the $dumping\_width$ defined as exponential factor in the moving average exponential weight: $\alpha = 1/2^{dumping\_width}$ (see Functional Description section for details). For example, values of 0, 1, 2,... 8 produce exponential weights α = 1, ½, ¼, ... 1/256, etc. Valid range is 0 to 32. Value of 0 causes no averaging (filter output follows the input).

inp_width [1…32]

Input bus width. Valid range is from 1 to 32.

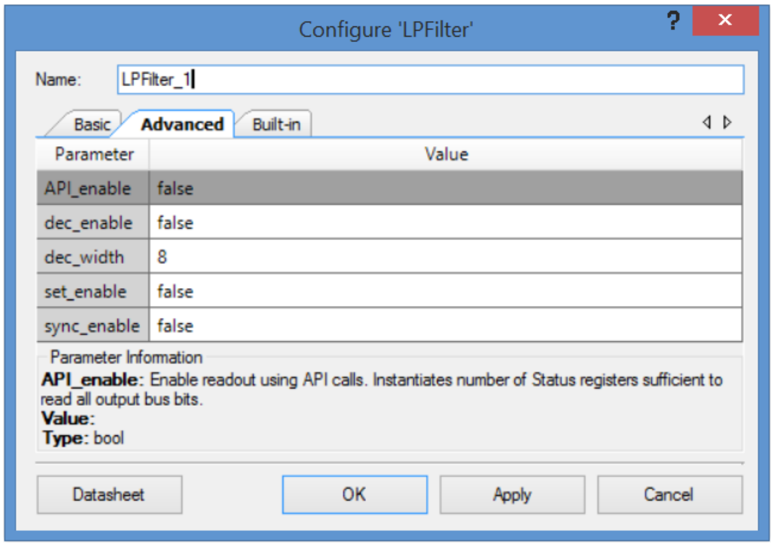
outp_type [singe bus | 2x8-bit bus]

Output bus type: single bus with variable width or fixed 2x8-bit bus (Figure 1). For 2x8-bit bus type out0 and out1 represent LSBs and MSBs. Splitting the output into 8-bit buses is optional and helps interfacing the component with some external components, such as FIFOin(*).

outp_width [1…32]

Output bus width. For **single_bus** option the valid range is from 1 to 32. For fixed 2x8-bit bus the **outp_width** parameter must be set exactly to 16 bits.

Advanced dialog provides following parameters:



Name: LPFilter_1

Parameter	Value
API_enable	false
dec_enable	false
dec_width	8
set_enable	false
sync_enable	false

Parameter Information
API_enable: Enable readout using API calls. Instantiates number of Status registers sufficient to read all output bus bits.
Value:
Type: bool

API_enable (bool)

Enables reading filter output using API functions. Default value is False. If enabled, this feature consumes from 1 to 4 status registers, depending on output bus width. See Application Programming Interface section for details.

[*] FIFOin is a part of the PSoC Sensei library by Brad Budlong.

dec_enable (bool)

Enables decimator counter for **dtr** pin. If enabled, the **dtr** output pulses whenever decimator counter rolls over. If disabled, the **dtr** output pulses on each strobe signal. If **set_enable** option is also selected, the positive edge on set control resets decimator counter to “0”, starting a new period. Decimator has no effect on the output bus, which continues updating on each strobe signal. Using decimator helps to reduce sampling rate of output data stream.

dec_width [1..32]

Decimator counter bit width. Valid range is [1 to 32]. Corresponding decimator period length is 2^{dec_width}. For example, values of 1, 2,... 8 produce period length: 2, 4, ... 256, etc. Parameter has effect only when **dec_enable** option is selected.

set_enable (bool)

Select this option to enable hardware set (track) control. If enabled, **set** terminal appears on the symbol.

sync_enable (bool)

Select this option for synchronization of the strobe signal when it comes from another clock domain. If enabled, **clk_s** terminal appears on the symbol.

Application Programming Interface

Function	Description
`LPFilter_Read8()`	Returns 8-bit filter output
`LPFilter_Read16()`	Returns 16-bit filter output
`LPFilter_Read32()`	Returns 32-bit filter output

uint8 LPFilter_Read8()

Description: Reads filter output. Has effect only if **API_enable** option is selected. This function is available only for output bus width range [1 to 8]. The internal Status registers are sticky(*), and clear upon read.

Parameters: none

Return Value: Filter output, right-aligned with filter output bus.

uint16 LPFilter_Read16()

Description: Reads filter output. Has effect only if **API_enable** option is selected. This function is only available for output bus width range [9 to 16].

Parameters: none

Return Value: Filter output, right-aligned with filter output bus.

uint32 LPFilter_Read32()

Description: Reads filter output. Has effect only if **API_enable** option is selected. This function is only available for output bus width range [17 to 32].

Parameters: none

Return Value: Filter output, right-aligned with filter output bus.

* For transparent mode change SR_mode localparam in the Verilog code.

Functional Description

Component implements first-order low-pass IIR filter that applies weighting factors which decrease exponentially [1, 2]:

$$y_n = (1 - \alpha)y_{n-1} + \alpha x_n \qquad \text{(Eq. 1)}$$

where x_n and y_n are filter input and output, and α is weighting factor defined as negative power of two: $\alpha = 1/2^{dumping\_width}$. The filter output has exponential response to step input; the smaller the weighting factor, the longer it gets the filter output to reach a steady value. The relationship between exponential time constant (τ), the weighting factor (α), and the sampling time interval (dt) simplifies when the sampling is fast compared to the time constant

$$\alpha^{-1} \cong \tau/dt \qquad (dt \ll \tau) \qquad \text{(Eq. 2)}$$

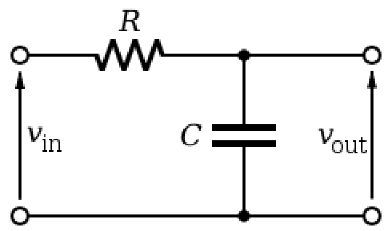
Thus, for discrete digital implementation α^{-1} is approximately equals a number of sampling clocks necessary to reach exponential value (the $dumping\_length$):

$$dumping\_length \equiv \alpha^{-1} = 2^{dumping\_width} \qquad \text{(Eq. 3)}$$

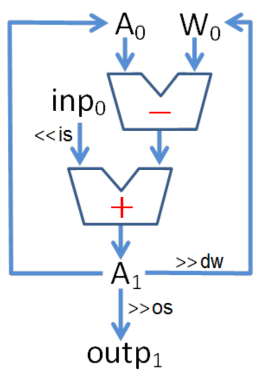
The filter cut-off frequency can be controlled by the sampling frequency, $F_{clock} = (dt)^{-1}$, and the dumping length

$$f = (2\pi\tau)^{-1} = \frac{1}{2\pi} \cdot \frac{F_{clock}}{dumping\_length} \qquad \text{(Eq. 4)}$$

Effect of exponential moving average filter on digital data is easy to understand and predict as it is similar to effect of integrating RC circuit on the analog signal [2].



Implementation



The component is implemented in Verilog as a state machine. Data flow diagram of the algorithm depicts filter accumulator previous value (A_0), next value (A_1), weight-factor-scaled accumulator (W_0), input (inp_0) and next output ($outp_1$). Accumulator bit width is selected based on the input and output bus width and dumping width. Using the weighting factor value as a power of 2 simplifies calculations, as multiplication in this case can be replaced with shift operation by $dumping\_width$ (***dw***) bits.

Output width of the filter can be set either larger than the input (output resolution increased), same width, or smaller width (data is averaged, but resolution is lost). To account for input and output bus width difference, the filter input (inp_0) is scaled up by input scale (***is***) bits prior to addition to accumulator. The cycle repeats on each rising edge of the strobe input. Filter has latency of one cycle.

The output ($outp_1$) is extracted from the accumulator using appropriate output scaling (***os***)[*]. The output is always left-aligned to maximize signal range. For example, if input bus has 8-bit width with input range [0 to 255], then 10-bit output bus shall have output signal range [0 to 1020][(†)].

Set and decimation options

Filter has optional set (track) control and output decimator. Their effect on filter operation is shown on Figures 2-4.

If enabled, the set signal forces filter output to instantly track the input. The filter response to sudden input change is exponentially slow[‡]. Using **set** control may improve settling by priming accumulator with new start value. The **set** signal also resets decimator counter (if available).

Decimator option enables decimator counter on **dtr** pin. This option is useful to reduce output data rate when smoothing high-speed data stream. If **set** control is also enabled, the set signal resets decimator counter; otherwise the decimator counter is free-rolling.

[*] Please refer to component's Verilog code for definitions of input and output scale factors.
[†] Notice the upper range is 1020 ($255\times2^{10-8}$), not 1023 (2^{10}-1). The range shrinkage is negligible for wide inputs, but becomes noticeable for narrow-bit inputs: e.g. for 1-bit input the output range would be [0 to 512].
[‡] It takes longer than 6τ to settle output down to 1-bit from 8-bit amplitude swing.

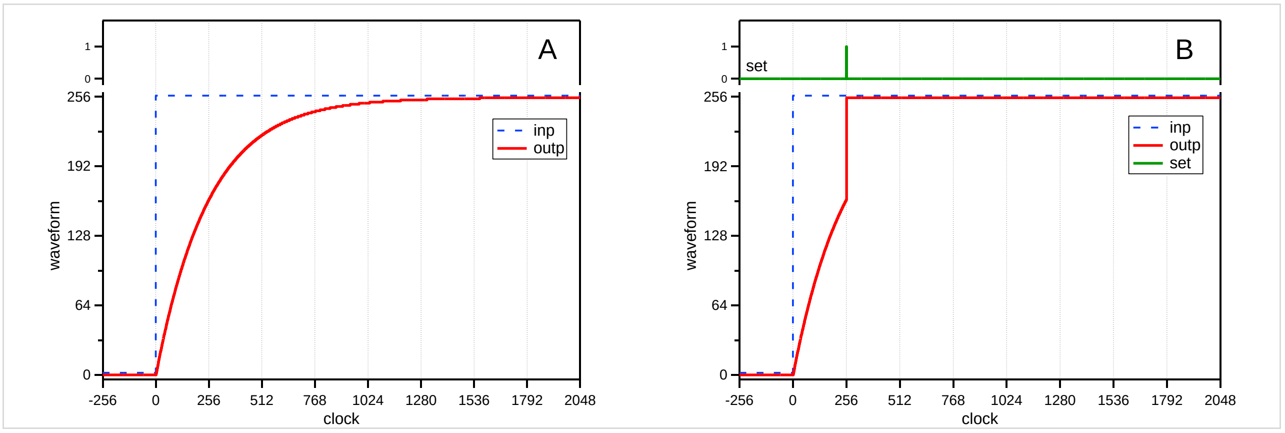


Figure 2. Effect of the set control. (A) filter response to the 0 to 255 step input. Input bus width: 8-bit, output width: 8-bit, dumping length: 256 (8-bit). (B) same but with set pulse applied at time 256, causing instant update of the output. The dotted line is offset by few pixels for clarity.

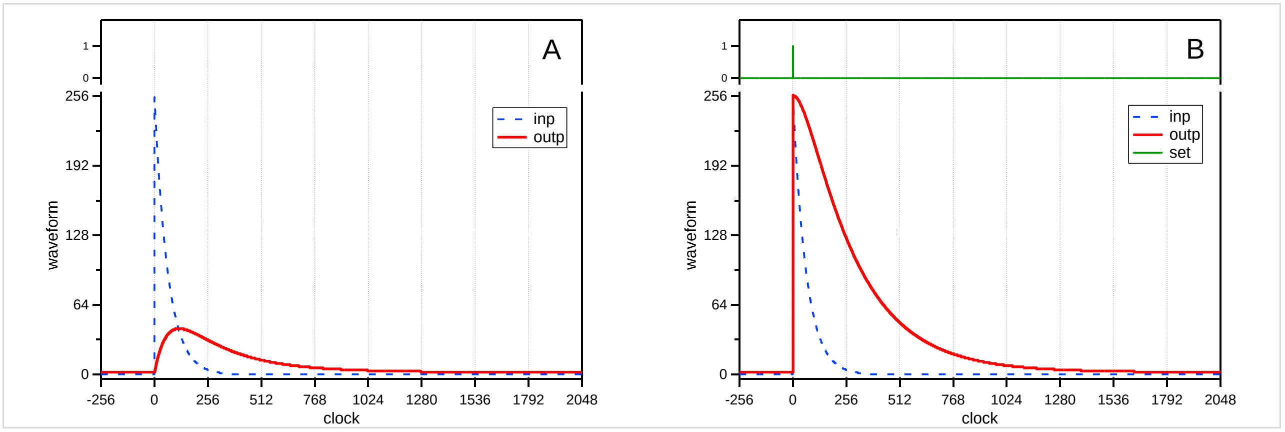


Figure 3. Effect of the set control: (A) filter response to short pulse of exponential decay length 64. Filter input width: 8-bit, output width: 8-bit, dumping length 256 (8-bit). (B) same but with set pulse applied at time 0, causing instant update of filter output.

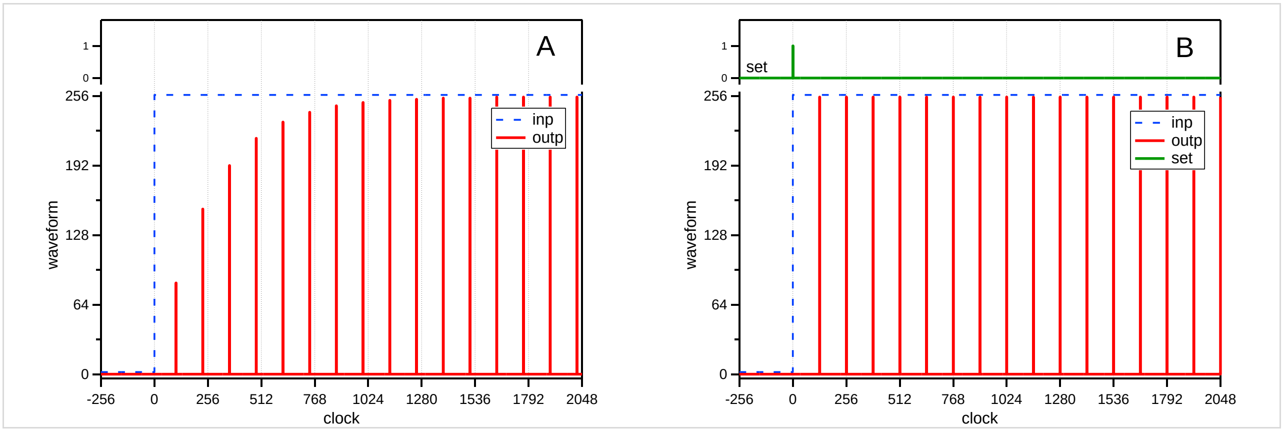


Figure 4. Effect of decimator: (A) filter response to the 0 to 255 step input. Input bus width 8-bit, output width 8-bit, dumping length 256 (8-bit). Decimator is free-rolling. Decimator counter length 128 (7-bit). (B) same but with set pulse applied at time 0, causing instant update of filter output and reset of the decimator counter.

Averaging for noise filtering

Filter response to noisy signal is shown on Figures 5-7.

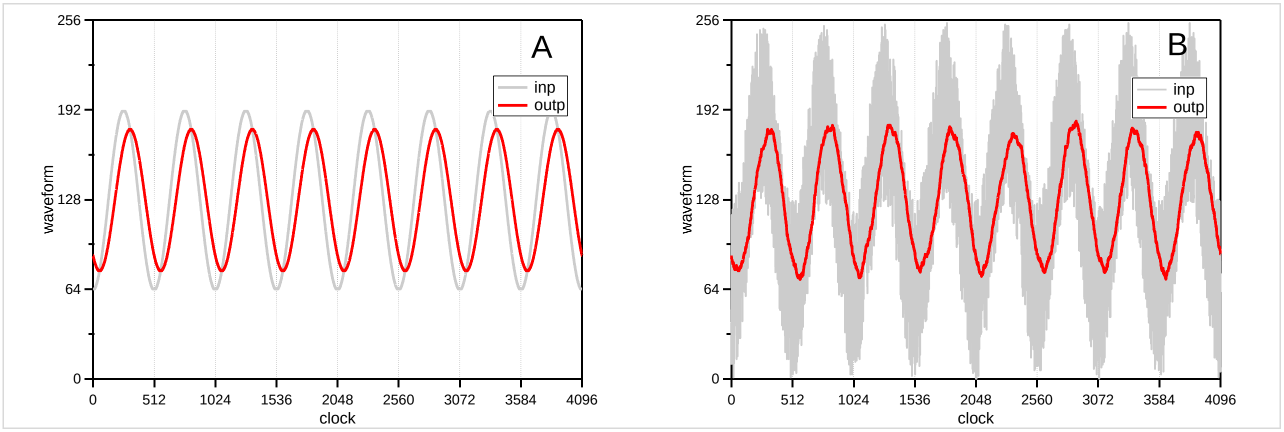


Figure 5. (A) filter response to the sine wave of amplitude 128 and period 512. Input width: 8-bit, output width: 8-bit, dumping length 64 (6-bit). (B) same but in the presence of random noise of amplitude 128.

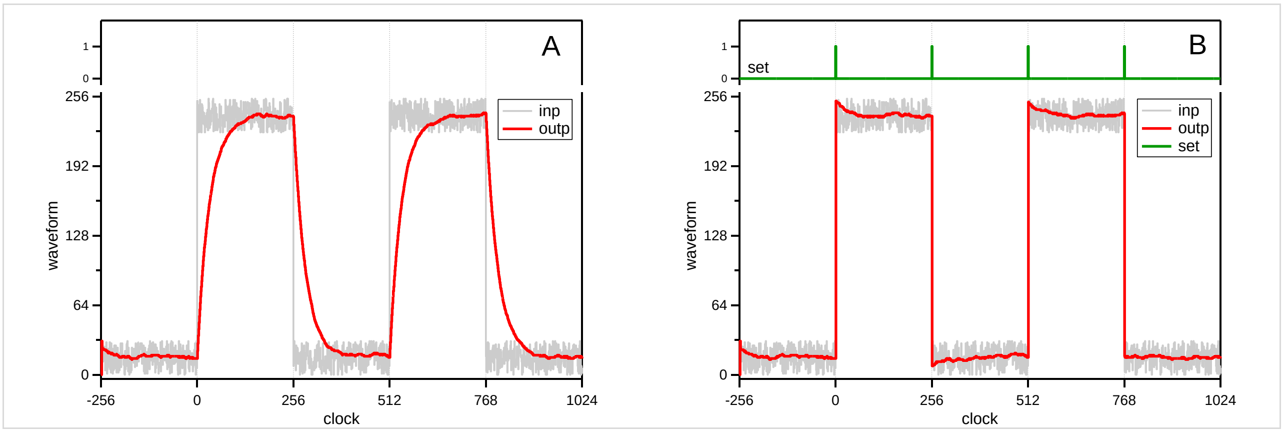


Figure 6. (A) filter response to square wave in the presence of the white noise. Input width: 8-bit, output width: 8-bit, dumping length 32 (5-bit). (B) same but with set pulses applied synchronously with data change.

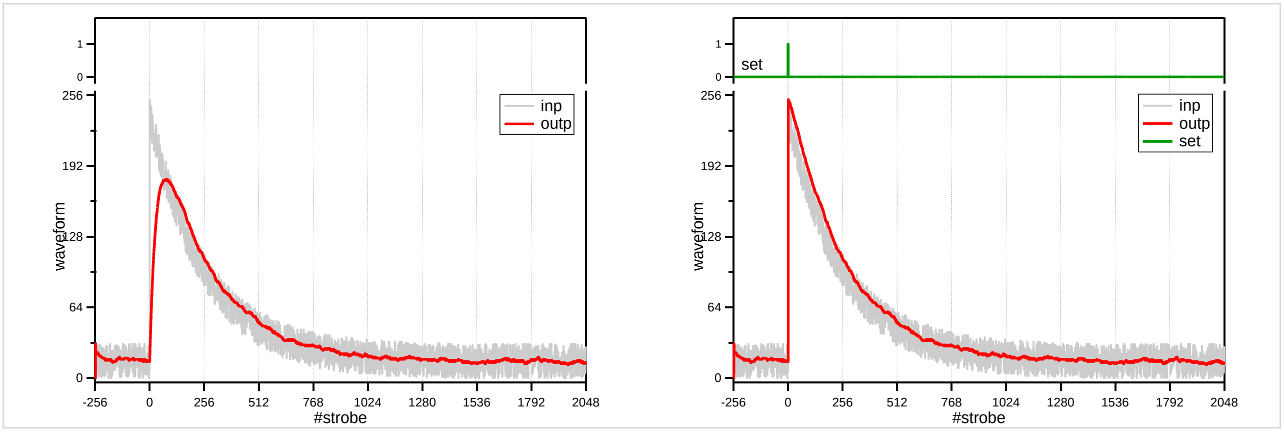


Figure 7. (A) filter response to exponential pulse in the presence of the noise. Input width: 8-bit, output width: 8-bit, dumping length 32 (5-bit). (B) same but with set pulse applied at time 0, causing instant update of output.

Averaging for better resolution

Under certain conditions, the filter can improve finesse of the sampled signal beyond ADC resolution[*] using dithering. It requires the sampled signal of having uniform noise of about 1-bit amplitude [3, 4]. The simplest method is to utilize noise already present in the analog signal. Often analog noise is artificially added to ADC input; such dithering option is present in many commercially available ADC cards. Each additional bit of resolution requires 4x oversampling (“1-bit” = $\sqrt{4}$), which, according to Eq. 3, need two extra bits of the ***dumping_width***. For example, increasing ADC resolution from 12-bit to 16-bit requires ***dumping_width*** of 8-bits.

Effect of the ADC_SAR 12-bit output being oversampled to nearly 16-bit resolution is shown on Figure 8.

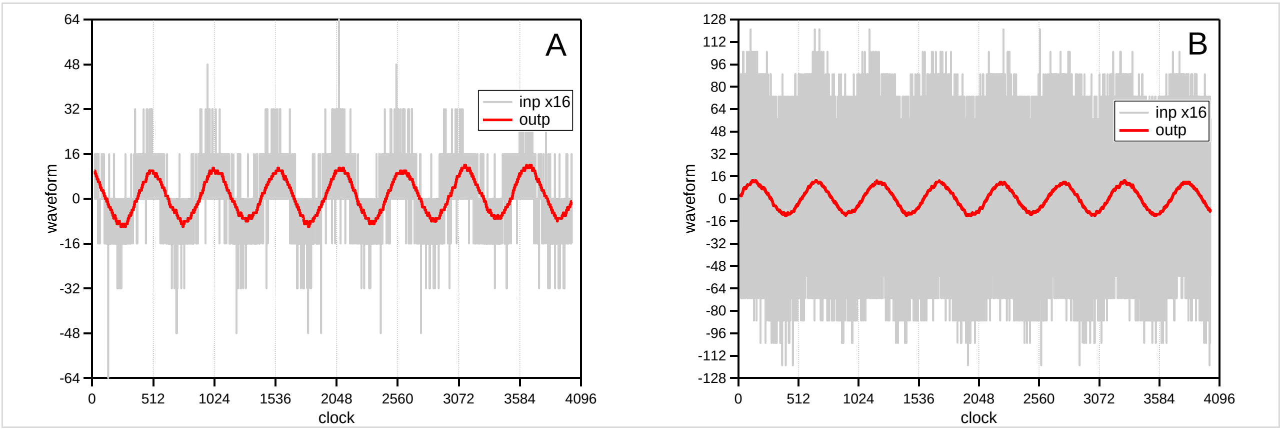


Figure 8. (A) filter response to ~0.7 mV (p-p), 3.9 Hz sine signal, sampled by 12-bit ADC_SAR at 16382 Hz. Filter input width: 12-bit, output width: 16-bit, dumping length 256 (8-bit), decimation length 8 (3-bit). Input signal is scaled by 16; constant offset is subtracted from both waves for clarity. ADC_SAR quantization steps of 0.5mV are clearly observed on the input. Output resolution is about 16-bits. (B) same but in the presence of external AC noise of amplitude 5 mV (nearby scope turned on).

[*] Filtering does not improve ADC’s INL performance.

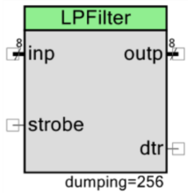
Performance

The component is written in Verilog and does not consume CPU resources. Component does not have built-in[*] DMA capabilities.

Resources

The component is written in Verilog and does not utilize UDB Datapath resources. The component was tested using PSoC5LP (CY8C5868-LP035). Component's footprint greatly varies depending on the option selected, summary for base configuration is provided below.

Table 1. PSoC5LP resource usage.

Options selected	Configuration
inp_width	8
outp_width	8
dump_width	8
API_enable	☑
dec_enable	☐
dec_width	—
set_enable	☐
	 LPFilter 8 inp outp 8 strobe dtr dumping=256

UDB Resource Type	Used	% Used
Macrocells	33	17.19%
Unique Pterms	49	12.76%
Datapath Cells	0	0.00%
Status Cells	1	4.17%
Control Cells	0	0.00%

DC and AC Electrical Characteristics

Table 2. Maximum sampling clock frequency vs. operation temperature range (for base configuration).

Parameter	Description	-40 °C – 85 °C	0 °C – 85 °C
F_{STROBE}, Max.	Maximum allowed sample clock	27.6 MHz	33.2 MHz

[*] DMA transfer from the filter can be achieved using the FIFOin component.

Sample Firmware Source Code

Basic application example shows filtering of the ADC_SAR[*] data stream (Figure 9). Several demo projects are provided showing various use of the component.

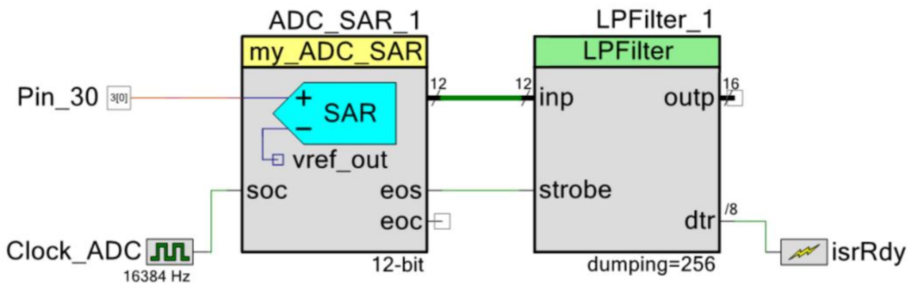


Figure 9. Basic application example.

Component Changes

Version	Description of changes	Reason for changes/impact
0.0	Version 0.0 is the first beta release of the LPFilter component	

References

1. Wikipedia. Exponential smoothing. https://en.wikipedia.org/wiki/Exponential_smoothing
2. Wikipedia. Low-pass filter. https://en.wikipedia.org/wiki/Low-pass_filter
3. Silicon Labs AN118. Improving ADC Resolution by Oversampling and Averaging. http://www.silabs.com/support%20documents/technicaldocs/an118.pdf
4. R. Lyons, R. Yates, Reducing ADC Quantization noise. http://mwrf.com/components/reducing-adc-quantization-noise

[*] Stock ADC_SAR component customized with hardware bus output.