**IIR Filter Implementation**

For Celero Telecoms Inc.  
Addressed to Carlo Condo

William Magnan

3/28/2025

**Implementation definition**

The IIR filter implemented in this document is given by the following:

A close up of text

AI-generated content may be incorrect.

Expanding the equation using the given parameter definitions:

Rearranging:

Factoring out **d**:

**Algorithm Implementation**

To compute y[n], we use the following steps:

1. Compute the difference:
2. Multiply by the decay factor **d**:
3. Compute the new output value:

This structure allows for efficient recursive implementation of the low-pass filter using simple arithmetic operations.

**RTL representation of the IIR algorithm**

The goal of using this algorithm is to minimize the critical path and implement pipelining to maximize system performance. Additionally, by factoring the multiplication with the d value, we reduce the process to a single multiplication, yielding significant gains in both speed and resource efficiency. Note that this introduces a latency to the output (3), and since the system operates in a closed loop, this latency is present with each new computation. The new result is required to calculate the next one, so every calculation is affected by this latency. (Note that a way of reducing this latency is presented later in this document).

A screenshot of a computer

AI-generated content may be incorrect.

**Fixed-Point Multiplication and Width Calculation**

In this section, the method for calculating the decay value and the width of each signal is defined.

**1. Fixed-Point Representation**

A two’s complement fixed-point number in the format <1.15> consists of 1 sign bit and 15 fractional bits. We define:

Since we round to the nearest integer, we obtain:

In binary, this corresponds to:

**2. Multiplication with an 8-bit Integer**

Let’s consider an 8-bit two’s complement integer:

Since the integer format is effectively <8.0>, we need to align it with <1.15> by shifting left by 15 bits:

This results in:

Now, we multiply:

Multiplying <8.15> by <1.15> produces a <9.30> vector.

**3. Scaling the Result Back**

Since the multiplication results in <9.30>, we must shift right by 30 bits to restore the correct scale:

This value is confirmed by analyzing the binary representation of the result:

Interpreting this in fixed-point format:

which evaluates to:

**4. General Formula for Result Width**

Given:

* x[n] with XWI integer bits (assumed <XWI.0> before alignment),
* d with DW total bits formatted as <1.(DW-1)>,

the resulting fixed-point format width YW is given by:

This accounts for the increase in integer and fractional bit widths due to multiplication.

**5. Generalize scaling the Result Back**

Since the number of fractional bits in the result is given by , the scaling factor needed to restore the correct value is .

In the design presented in the next section, the output signal can be parametrized to extract only YWO (output width) bits from the least significant bits of YW. To correctly scale back the value, we need to divide the result by .

**RTL representation of the IIR algorithm with bus sizes**

With the values obtained in the last section, we have the following RTL system:

A screenshot of a computer program

AI-generated content may be incorrect.

**First implementation results**

The following picture shows the result of the system's first iteration. We observe that it takes three clock cycles for the first output to appear on the yn signal. Additionally, the output remains unchanged for three clock cycles

A screenshot of a computer

AI-generated content may be incorrect.

**Speed optimization**

A straightforward method to accelerate the calculation is by performing part of it on the falling edge of the clock. This technique allows us to overlap operations, effectively reducing the latency from 3 to 2 clock cycles. The improved performance is illustrated in the following diagram, where the timing differences are clearly visible.

A screenshot of a computer

AI-generated content may be incorrect.