Integrated System Architecture
Lab session 1 report

Marco Andorno (247222)
Michele Caon (253027)
Matteo Perotti (251453)
Giuseppe Sarda (xxxxxx)

December 18, 2018

This report along with all the source files, scripts, reports and diagrams for the project can be found on GitHub at https://github.com/mksoc/ISA-filter-design.

1 Filter design

Following the rules given in the assignment, the main specifications were derived:

- Filter type: IIR
- Filter order: $N = 2$
- Cutoff frequency: $f_c = 2\,\text{kHz}$
- Sampling frequency: $f_s = 10\,\text{kHz}$
- Data parallelism: $n_b = 12$

Then, using the provided example MATLAB script, the filter coefficients were found by means of the `butter` function. Real coefficients are then quantized as fixed point fractional number in the format $Q1.(n_b - 1)$ ($Q1.11$ in our case) and expressed as integers on n_b bits for the future C model and hardware filter. We will discover later that some care has to be taken when performing operations on the integer representation of fixed point numbers.

Quantization is performed by truncation (`floor` function), so that the maximum error is equal to:

$$\varepsilon_{max} = 2^{-(n_b-1)} = 2^{-11} = 0.049\% \tag{1}$$

We accept that this error could be reduced by rounding and that truncation introduces a negative bias by approximating always towards $-\infty$, because on the other hand truncation is much easier to implement in hardware, where it just represent an arithmetic shift.

The resulting difference equation is in the end:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

where

$b_0 = 0.20654 = 423$

$b_1 = 0.41309 = 846$

$b_2 = 0.20654 = 423$

$a_1 = -0.36963 = -757$

$a_2 = 0.19580 = 401$

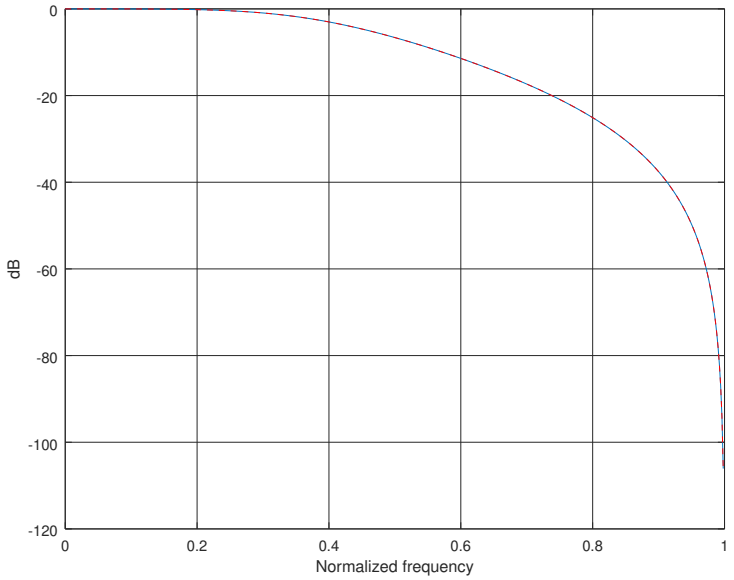

Figure 1: Bode plot of the filter frequency response

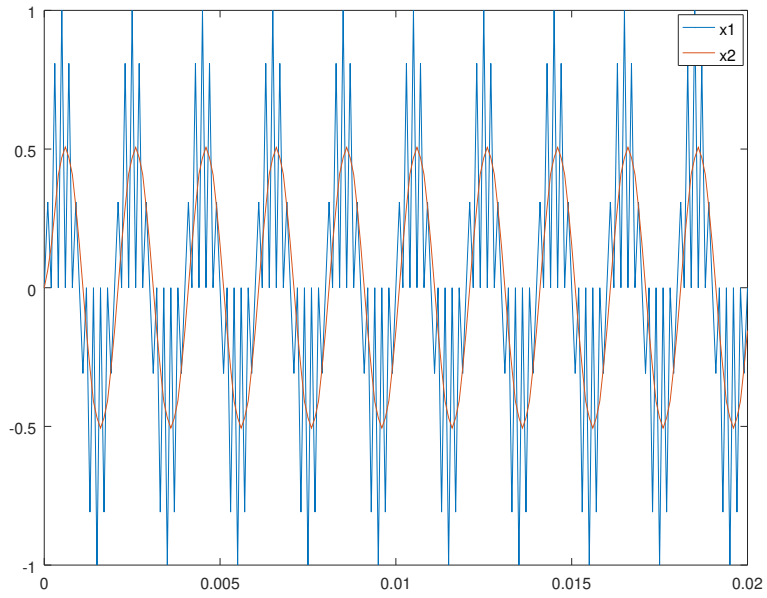

Figure 2: Input and output waveforms

Figure 1 shows the transfer function of the filter computed from both the original coefficients and the quantized ones. The two curves cannot be told apart because the error is too small ($5.42 \cdot 10^{-7}$ in the worst case).

Figure 2 on the other hand shows the time domain waveforms of an input signal $x_1(t)$, in blue, containing two frequency components one in band and one out of band, and the corresponding output $x_2(t)$, in red, where only the in-band component survives.

2 Fixed-point C model

The next step consists of writing a software model of the filter in C, which mirrors the behavior of the hardware architecture to be designed next. In this regard, the main difference between the MATLAB model and this one is that the former uses quantized coefficients but performs the internal computation using the maximum precision allowed by the machine, while the latter performs computation always resorting to the original fixed parallelism of data (12 bits here).

The development of this software model started with the example provided in the assignment, tailored to our specifications. A Python script was developed and used to compare the results file of the two models. As expected, the comparison shows that the two models differ at most of one unity, that is the previously computed ε_{max} (1) in fractional form. Furthermore, results from the MATLAB model, when different, are always greater than the results from the C model, as the latter performs multiple truncations (rounding towards $-\infty$) in its computations.

3 Base architecture design

The development of the first architecture began with the definition of the Data Flow Graph of the filter, shown in figure 3, from which the number of hardware resources needed can be derived (i.e. two delay elements, four adders and five multipliers).

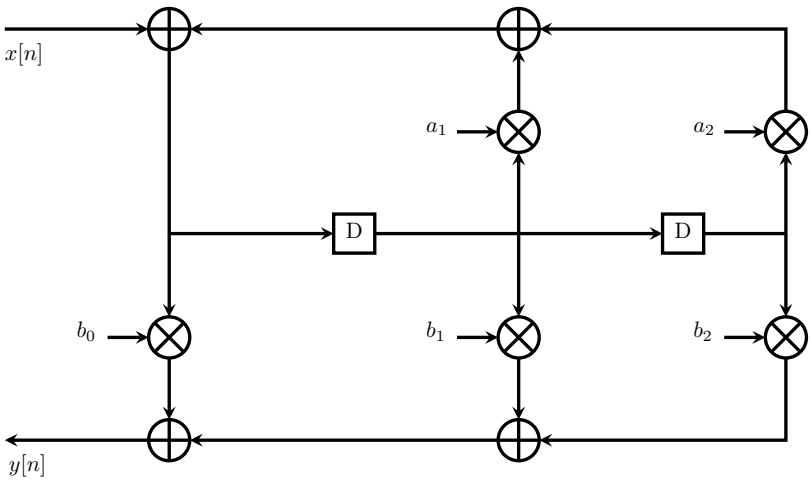

Figure 3: DFG of the filter

3.1 Datapath

From the generic DFG, a full datapath architecture was designed, shown in figure 4. Compared to the simple DFG of figure 3, this datapath explicitly shows all signal names used throughout the design and their parallelism, along with additional interface registers required by the specifications. The following paragraphs detail the key points of the design.

3.1.1 Color legend

All datapath schemes used in this report follow the same color coding:

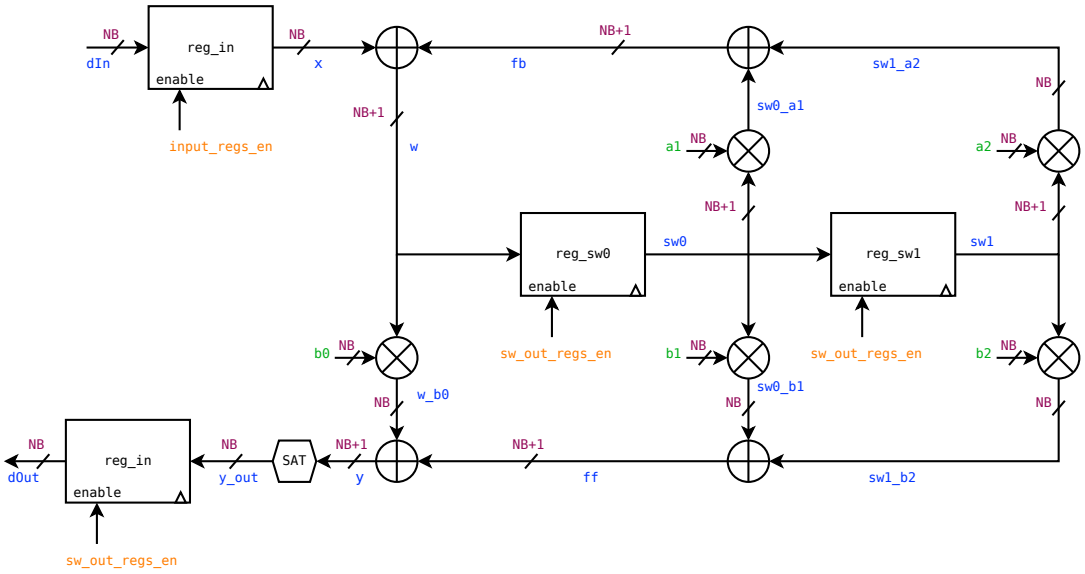

Figure 4: Datapath

- Blue signals are data signals
- Green signals are external coefficients
- Purple labels define the parallelism of the corresponding signal
- Orange signals are control signals coming from the Control Unit

3.1.2 Registers

In addition to the two core delay elements of the filter DFG, the complete datapath includes also border registers for each piece of data coming in or out of the architecture, specifically the input and output sample stream and all filter coefficients. Input registers for the latter are not actually shown in figure 4 to avoid too much visual clutter, but assume that each coefficient in green is actually the output of a register.

Each register has implicit clock and asynchronous reset signals, as well as individual enable signals. No synchronous clear is provided as it is of no use for the purposes of this design.

3.1.3 Arithmetic operators

As per specifications,