

# M.EEC041 - Digital Systems Design

2022/2023

Laboratory project 3 - V1.0  
2 Dec 2022

(due date: Dec 19<sup>th</sup>, 2022)

## PROFIR - Programmable finite response 8 filter bank

Revision history

date	notes	author
Nov 22, 2022	V0.1 - Preliminary version with first spec of the filter bank	<a href="mailto:jca@fe.up.pt">jca@fe.up.pt</a>
Dec 2, 2022	V1.0 - Too many changes added to the initial specification	<a href="mailto:jca@fe.up.pt">jca@fe.up.pt</a>

### 1 - Introduction and functional description

This project will build a custom digital circuit for implementing a bank of eight different finite response digital filters for processing an ultrasonic signal sampled at 1953.1 kHz (250 MHz/128), with 16 bits per sample. Each digital FIR filter can have up to 128 coefficients represented by 18 fixed-point signed numbers, with 2 integer bits and 16 fractional bits.

For each input sample, the system must compute the outputs of the 8 digital filters, producing 8 digital signals with the same sampling frequency as the input signal. The 8 outputs must be updated exactly in the same clock cycle. The output is produced by calculating the discrete convolution between the input signal (represented by samples  $x_j$ ) and the filter impulse response  $h_k$  (the filter coefficients):

$$y_n = \sum_{k=0}^{N-1} h_k \cdot x_{n-k}$$

In this expression, the values  $x_{n-k}$  represent the  $N$  previous input samples,  $h_k$  are the  $N$  filter coefficients and  $y_n$  is the output sample to calculate. The maximum number of coefficients is 128. Shorter filters can be implemented just by filling the non-existent coefficients with zeros. The behaviour of the system, for a single FIR filter can be described by the following pseudo code;

```
Xpast[128] contains the last 128 input samples received at the datain input
Hcoeff[128] contains the 128 filter coefficients
Datain is the input sample
Dataout is the output sample
For each new input sample arriving at Datain (when din_enable == 1)
  For i=0 to 126
    Xpast[ i ] = Xpast[ i+1 ]
  endFor
  Xpast[ i ] = Datain
```

```

Accum = 0
For i=0 to 127
    Accum = Accum + Xpast[ i ] * Hcoeff[ i ]
endFor

Dataout = Accum

```

The filter coefficients  $h_k$  are read from a bank of 8 independent RAM memories. Each memory is organized as 64 words of 36 bits (two 18-bit coefficients per memory location) and is read by a clocked synchronous interface, as described below. The memories will be implemented as dual-port memory blocks, using one write port to upload the memories with the filter coefficients (this part is not included in your project). The low-order filter coefficient will be stored in the 18 least significant bits of each word, as represented below:

RAM address	RAMk[35:18]	RAMk[17:0]
0	$h_1$	$h_0$
1	$h_3$	$h_2$
2	$h_5$	$h_4$
...	...	...
63	$h_{127}$	$h_{126}$

Figure 1 show a simplified block diagram of the system to design.

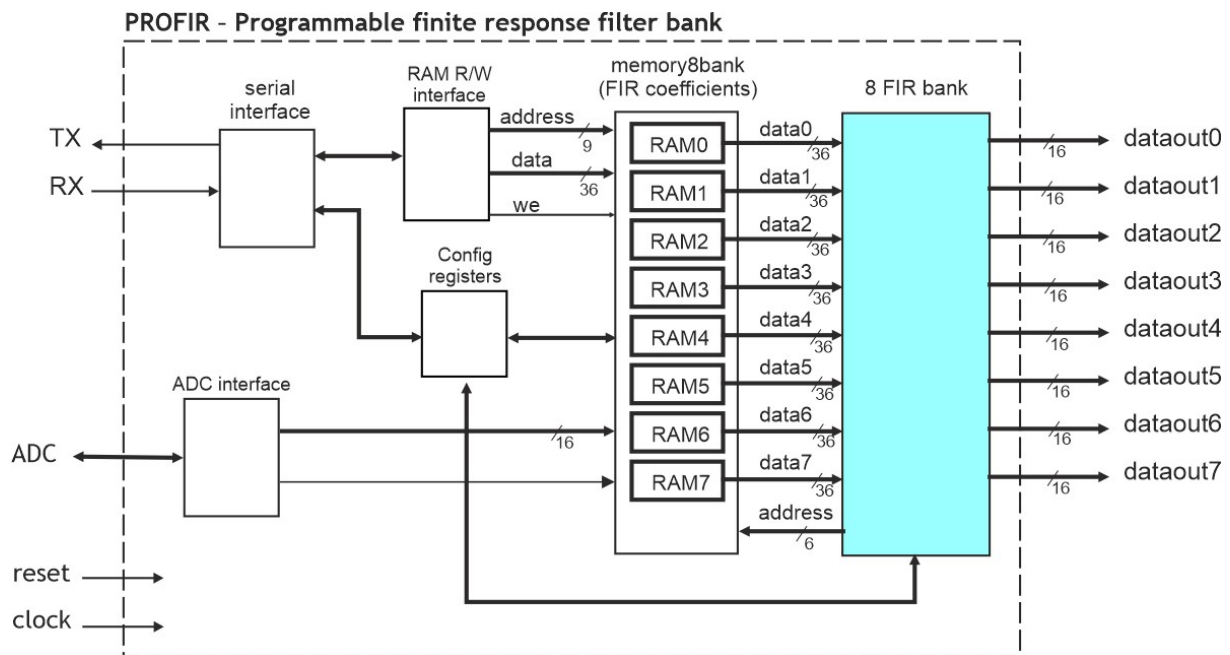


Figure 1 - Block diagram of the digital filter bank

Your design will implement only the 8 digital filters (the blue box) and will be integrated in a complete design that includes the memories holding the filter coefficients, the interface to upload those memories from an external system and the rest of the digital processing system.

## 2 - Implementation

The system must be designed as single clock domain synchronous digital circuit, clocked at 250 MHz. The Verilog code below represents the interface of the system to design:

```
module profir(
    input          clock,          // Master 250 MHz clock, active in the rising edge
    input          reset,          // Master reset, synchronous, active high
    input signed [15:0] datain,     // Input signal sample
    input          din_enable,     // When 1, a new sample is present at datain input (lasts 1 clock)
    output [5:0]    coeffaddress,  // Address to read all the coefficient memories
    input signed [35:0] coeff0,    // Coefficient read data for filter 0
    input signed [35:0] coeff1,    // Coefficient read data for filter 1
    input signed [35:0] coeff2,    // Coefficient read data for filter 2
    input signed [35:0] coeff3,    // Coefficient read data for filter 3
    input signed [35:0] coeff4,    // Coefficient read data for filter 4
    input signed [35:0] coeff5,    // Coefficient read data for filter 5
    input signed [35:0] coeff6,    // Coefficient read data for filter 6
    input signed [35:0] coeff7,    // Coefficient read data for filter 7
    output signed [15:0] dataout0,  // Output data of filter 0
    output signed [15:0] dataout1,  // Output data of filter 1
    output signed [15:0] dataout2,  // Output data of filter 2
    output signed [15:0] dataout3,  // Output data of filter 3
    output signed [15:0] dataout4,  // Output data of filter 4
    output signed [15:0] dataout5,  // Output data of filter 5
    output signed [15:0] dataout6,  // Output data of filter 6
    output signed [15:0] dataout7,  // Output data of filter 7
);
```

### 2.1- System design

The 8 RAMs holding the filter coefficients can be read in parallel with a single address that is connected to the address buses of all the memories. The read ports of these memories are implemented by the following Verilog process (see the Verilog file `./src/verilog-rtl/memory8bank.v`):

```
always @(posedge clock)
    coeffout <= RAMCOEF[ coeffaddress ];
```

The timing of the read operation is as follows: the coefficient memory address **coeffaddress** is set by your circuit in clock cycle K, the memory output data register **coeffout** is loaded in clock cycle K+1 and your system can only use that value in clock cycle K+2. As the memory read access is pipelined, a different memory address can be read at each clock cycle.

The memory bank (module **memory8bank**, Verilog source `./src/verilog-rtl/memory8bank.v`) implements the 8 RAM memories with a common read address and eight output 36-bit data buses. The write port will not be used by your design and will be connected in the top-level design to a low speed interface for loading these memories from an external computer. The memories are pre-loaded for simulation and synthesis with the data read from hex data files using the Verilog system task `$readmemh( )`. These data files are located in folder `./simdata` and contain the filter coefficients generated by the Matlab script `./Matlab/genfilterdata`.

The calculation of all the 8 output samples must be completed before the arrival a new input sample. Thus, the computation of the 8 digital filters must be completed within 128 clock cycles (this is equivalent to a computational complexity of 4 giga integer operations per second). Figure 2 shows a timing diagram illustrating the arrival of the input samples and their relationship with the clock and the **din\_enable** signal.

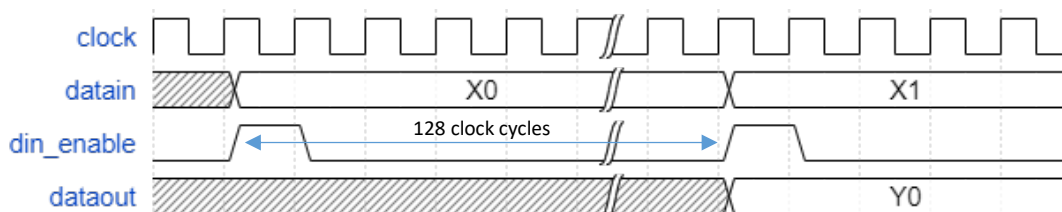


Figure 2 - Timing diagram of the input/output signals.

## 2.2 - Simulation environment

The project kit includes a basic testbench for simulating the module **profir** (the 8 filter bank) and a simulation project already created for QuestaSim. The project kit includes the following folders and files. Note that it is mandatory to keep the projects and files in the correct folders as the Matlab scripts and Verilog files use relative references to some filenames

<code>./src/verilog-rtl</code>	Verilog source files for design implementation
<code>./src/verilog-tb</code>	Verilog source files for design verification/simulation
<code>./src/data</code>	Additional data files required for FPGA implementation
<code>./simdata</code>	Data files used for pre-loading the coefficient memories and for simulation
<code>./Matlab</code>	Matlab scripts for generating the data files
<code>./sim</code>	Simulation project for QuestaSim. Includes a QuestaSim script for configuring the waveform viewer (file <code>./sim/wave_profir.do</code> )

The Verilog source files provided are the following:

<code>./src/verilog-rtl/filterbank.v</code>	The header of the Verilog module to develop. Do not change the signal names and sizes. If needed you can add additional inputs/outputs.
<code>./src/verilog-rtl/memory8bank.v</code>	A synthesisable model of the memory bank with the 8 RAM holding the filter coefficients. These memories are pre-loaded for simulation and synthesis with the data files located in folder <code>./simdata</code> .
<code>./src/verilog-tb/profir_tb.v</code>	A basic testbench for simulating the filter bank. The testbench instantiates the memory bank and the filter bank and applies the input samples to the filter module with the given sampling frequency.

The Matlab scripts are:

<code>./Matlab/genfilterdata.m</code>	creates a data file in <code>./simdata</code> with the 128 coefficients implementing a digital FIR filter. This script can be easily adapted to generate different filters, one for each of the 8 channels. For now, the 8 coefficient datafiles implement the same filter but you should validate your system with 8 different filters.
<code>./Matlab/geninputdata.m</code>	create a datafile in <code>./simdata</code> with the input signal to be applied to used in the simulation of the testbench provided. This script creates a sum of a few sinusoidal waves, with added white noise and modulated in amplitude with another slow sine wave. Feel free to modify the parameters and the signal used.

## 2.3 - Design evaluation and grading

The design must target the Spartan6 FPGA and minimize the global logic complexity, measured as the number of main FPGA resources: lookup-tables, flip-flops and DSP48 blocks.

The project stages to complete and the grading to assign to each stage are:

1. Design of the Verilog HDL code and functional verification with logic simulation [20%]
2. RTL Synthesis and post-synthesis validation with logic simulation [20%]
3. Integration in a FPGA project (a complete project will be provided) [15%]
4. Final RTL synthesis, design optimization and validation post-synthesis [20%]
5. Place and route and post-P&R simulation [10%]
6. Final report [15%]