Politecnico di Torino

Integrated Systems Architectures

Design and implementation of a digital filter
Laboratory 1 report

GROUP13

Naouras Latiri Hugo Bouisson Zeineb Ben Othmen

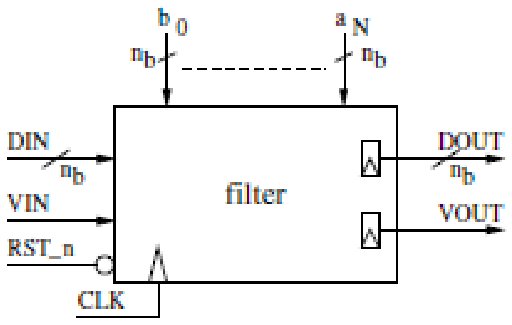
November 21, 2021

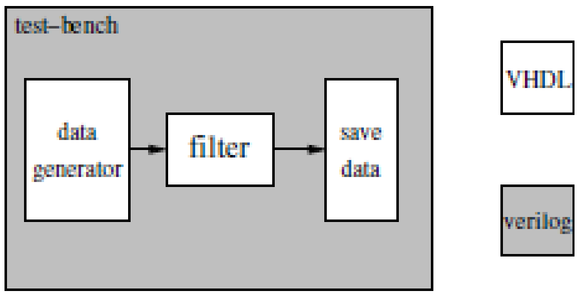
Contents

1 Introduction

In this first laboratory we have designed and implemented a digital filter of a Finite Impulse Response ”FIR”.
To start with the Reference model development, we have used a Matlab script implemented in pseudo fixed point in order to test and get quantified coefficients of the low-pass filter of 9 bit-width and an order equal to 10 with a cut-off frequency of 2 kHz. Then we have developed a fixed point C model that is evaluating the performance of the pseudo fixed point implementation.

For the second part which concerns the VLSI implementation, we have developed in VHDL the architecture of the FIR filter according to these requirements specified in the following figures:





We have used Modelsim, in order to do the simulation and check the correctness of our designed system. Then we have synthesized the filter using Synopsys Design Compiler which allowed us to find the maximum frequency that our design may achieves,the corresponding area and the power consumption. For the place and route phase, we have used Cadence SOC Innovus.

Last and not least, we have improved our architecture by applying the unfolding and pipelining techniques. For this advanced design architecture we repeated all the steps that we did for the base design.

2 Assignment

2.1 Reference model development

2.1.1 First step: Matlab/Octave pseudo-fixed implementation

The first step was to test our filter implemented in pseudo fixed point in the Matlab environment, using these files :

- my_fir_filter.m
- myfir_design.m

The first file contains a sequence of samples corresponding to a signal composed by the mean of two sine-waves of 500 Hz (in pass-band) and [4:5] kHz (outpass-band) and sampled with a frequency of 10 kHz.

It simulates the filter with given specification and give the quantized input samples and output samples in different text files. These different files were used after for testing the C program implementation and then the VHDL implementation.

The input samples are the same for both of them and the outputs are compared to the ones obtained using Matlab.

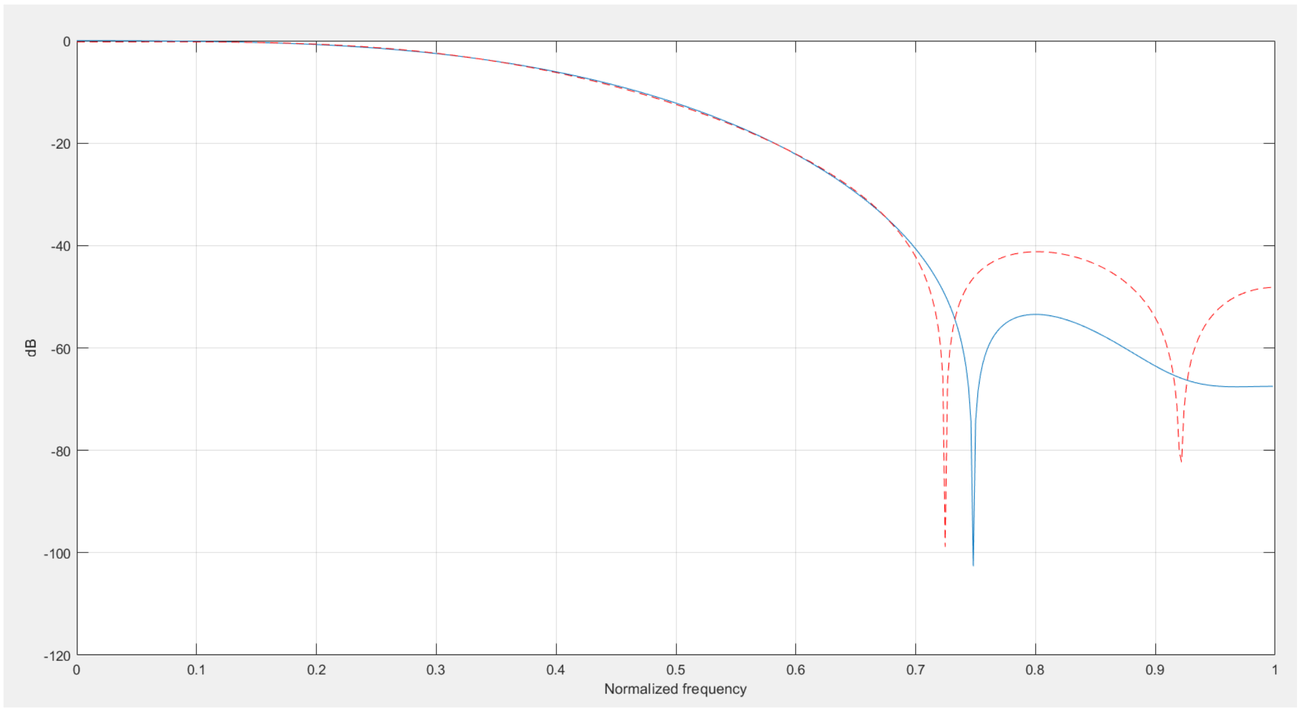


Figure 2.1: Filter response obtained in Matlab environment. The blue curve is the ideal response, the red curve is the response of the quantized filter

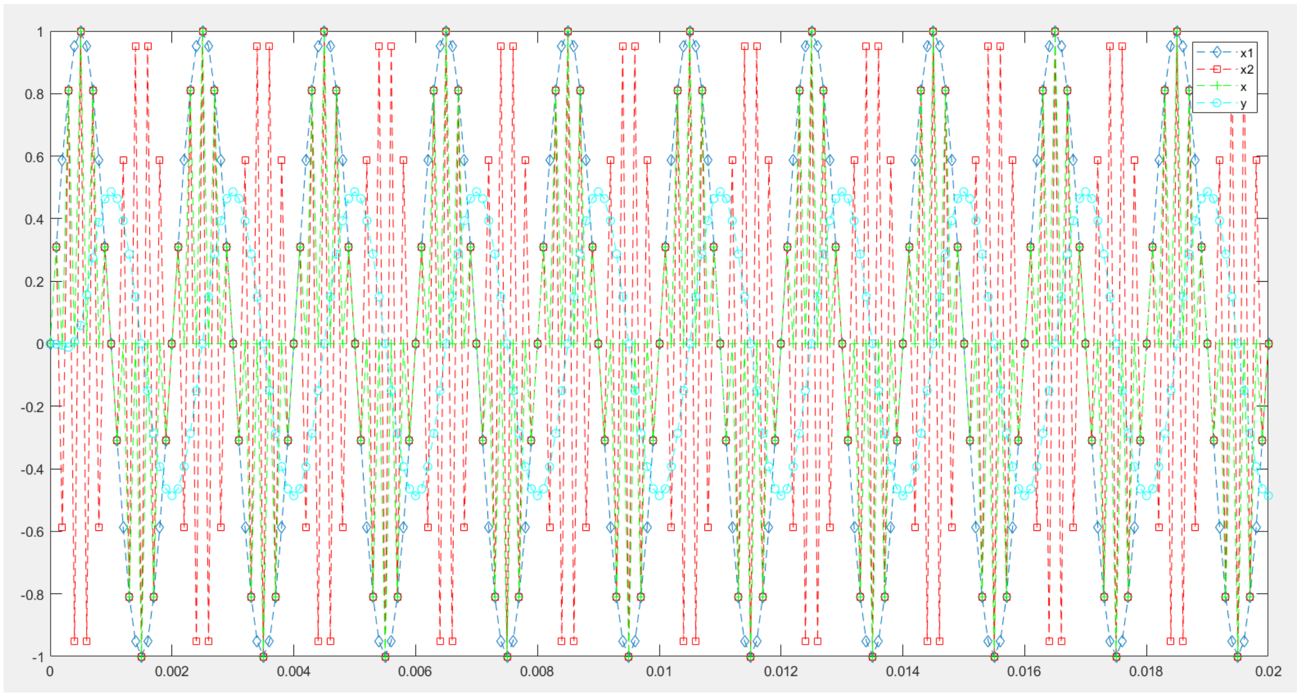


Figure 2.2: Blue: samples of the first input component. Red: samples of the second input component. Green: resulting input samples. Cyan: output samples

2.1.2 Second step : fixed-point C model

As we have to design a FIR filter, the recursive part will disappear and so we will have as fixed point implementation of the filtering operation:

$$y_i = \sum_j x_{i-j} b_j - \sum_k y_{i-k} a_k \ = \sum_j x_{i-j} b_j$$

2.2 VLSI implementation

2.2.1 Starting architecture development

Firstly we have to implement the previous filter form in VHDL. The top entity is shown bellow in the next figure and those are the ports used by it:

- DIN : data input
- VIN : high when valid input data
- RST n : active low synchronous reset
- CLK : input clock
- $b_0, ..., b_n$: coefficients
- DOUT : data output
- VOUT : high when valid output data

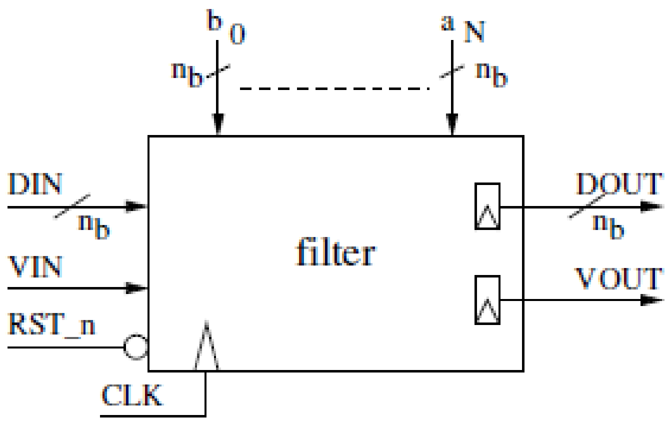


Figure 2.3: Top level entity

The VHDL code is divided into four blocks : adder, multiplier, register and multiplexer.
The adder performs the addition of two numbers, the multiplication performs the multiplication of two numbers and then the MSB is taken, to obtain a result on 10 bits (9+1).
The register updates the output by clearing or writing on it. It is a synchronous reset active low.
Then, the multiplexer is used to select the correct output which is a value containable on 9 bits (between [-256 - 255]). It can be the maximum value that could be written on 9 bits, if the the sample that has been computed is higher than [255]. Or it can be the minimum value that could be written on 9 bits, if the computed sample is lower than [-256].

2.2.2 Simulation

The testbench is divided into three blocks, a clock generator, a data generator which takes its inputs from the samples file and writes the outputs in an other file, and a data synchronizer.

The clock generator generates the clock for the entire circuit and especially the clock which drives the data generator.

The data generator reads the samples generated by the first part in order to feed the filter. Moreover a validation signal is implemented in order to test the correct behavior of VIN and VOUT.

The data synchronizer manages the signal VIN, in order to control the sequential output samples and so the correct writing of the result.

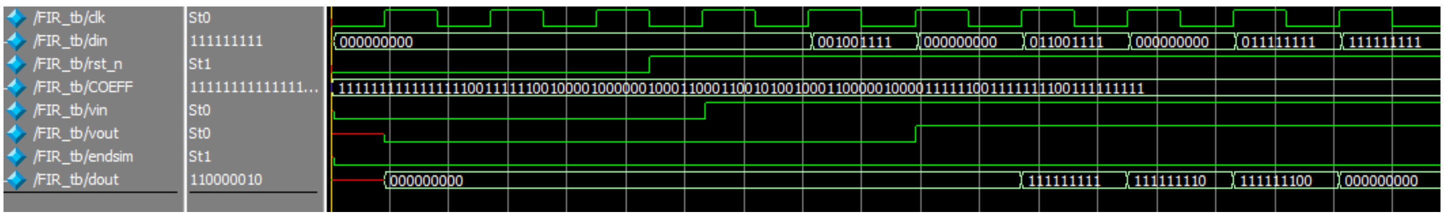


Figure 2.4: Filter's behavior at the beginning

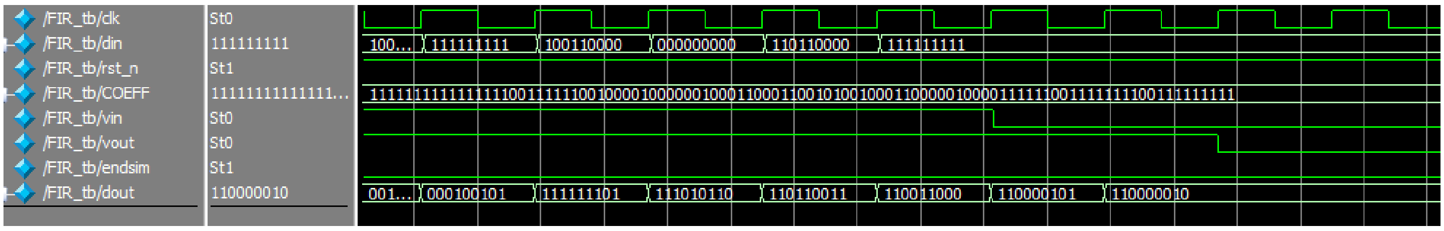


Figure 2.5: Filter's behavior at the end

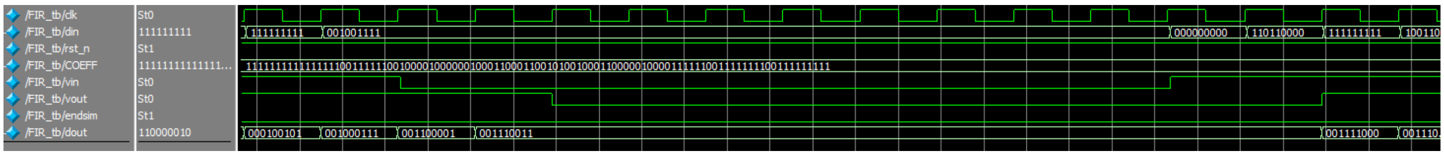


Figure 2.6: Correctness of the circuit even when VIN goes to 0

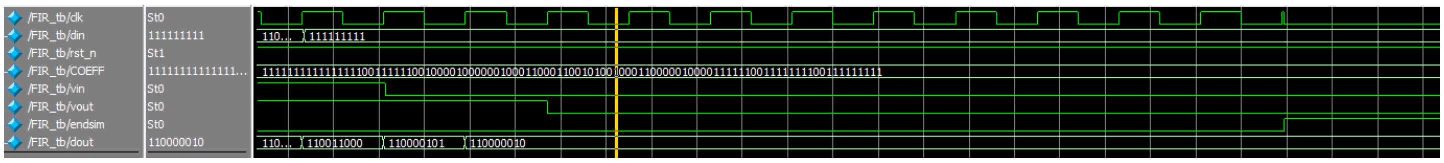


Figure 2.7: End of the simulation, the signal "endsim" goes to 1 after a delay

Looking to the Figure 2.4 we can see the computation of the first samples. The validation signals VIN and VOUT allow to load the next value for DIN and DOUT respectively.

The Figure 2.5 shows the behavior of both VIN and VOUT signals (equal to 0) at the end.

On the Figure 2.6 we are shown the test for the correctness of the Design. When VIN goes to zero, the process is paused until VIN is set again at the high level. Here we performed the test during 10 clock cycles.
On the Figure 2.7 we can see the behavior of the signal "endsim" which goes to 1 after a fixed delay.

2.2.3 Implementation

- Logic Synthesis

Max frequency [MHz]	Max area μm^2
$302,11$	$5670,587982$

Table 1: Post synthesis report

	Internal Power	Switching Power	Leakage Power	Total Power
µW	179.8025	152.7318	95.395	427.9295
%	42.01	35.69	22.29	100

Table 2: Post synthesis report

- Place and Route

Max area μm^2
4752.4

Table 3: Post synthesis report

	Internal Power	Switching Power	Leakage Power	Total Power
µW	409.71061	310.48660	94.32820	814.52542
%	50.3	38.12	11.58	100

Table 4: Post synthesis report

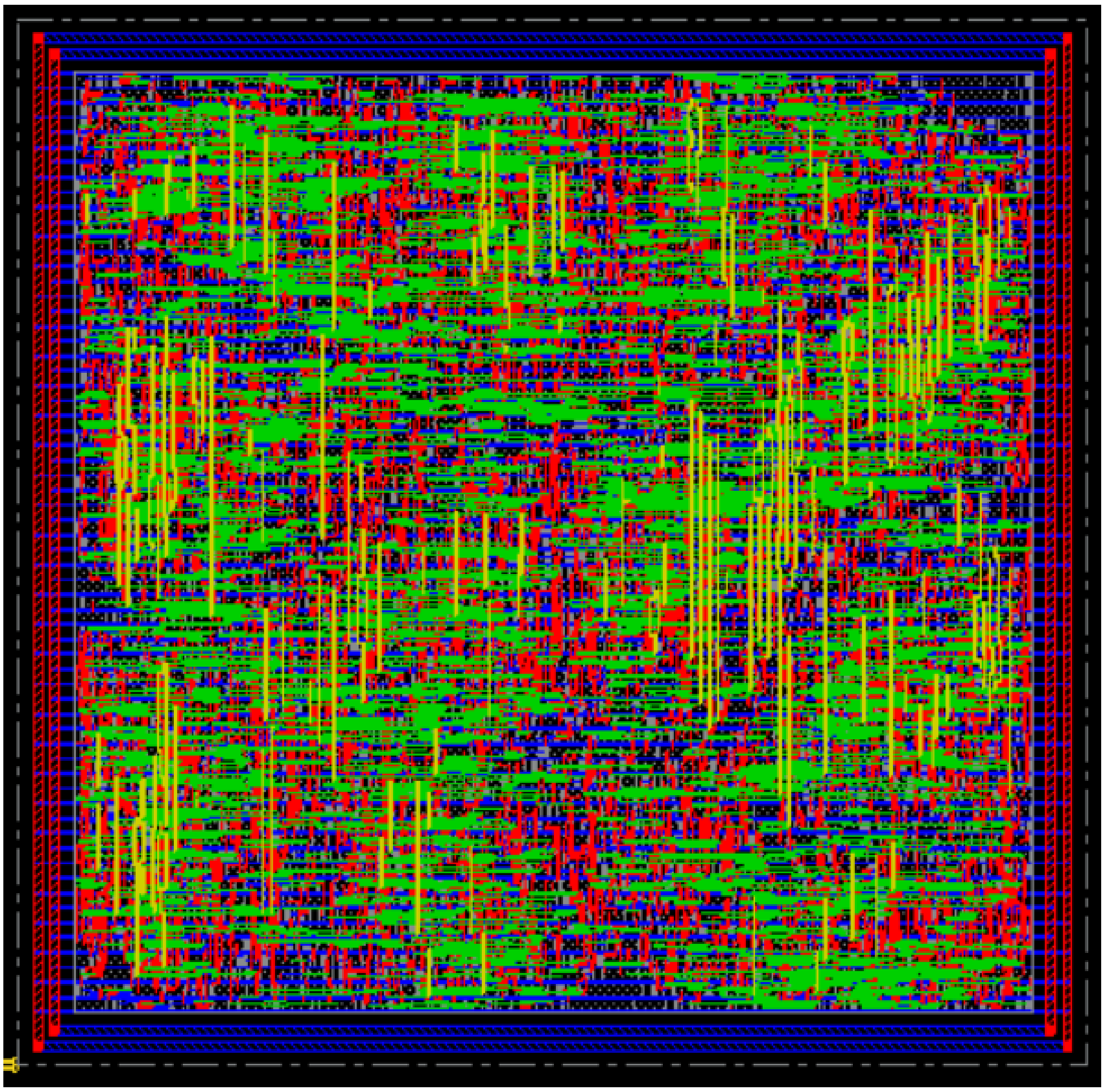


Figure 2.8: Innovus floorplan overview

2.3 Advanced architecture development

For this part we have improved our design by using both 3-unfolding and pipelining techniques. We decided first to unfold the system and after pipeline it, because it seemed to be easier.

Then, since we have the same architecture for the three path, we can apply pipelining only on one path and copy it for the others. We first use an additional register for each multiplier. Then, as we have 10 adders, we made the choice to make groups of 3 adders in cascade before reaching the critical path of the multiplier. This configuration allows us to have the minimum critical path and also the minimum delay.

Looking at the VLSI implementation, the procedure to simulate, synthesis and place and route our design is the same as before.

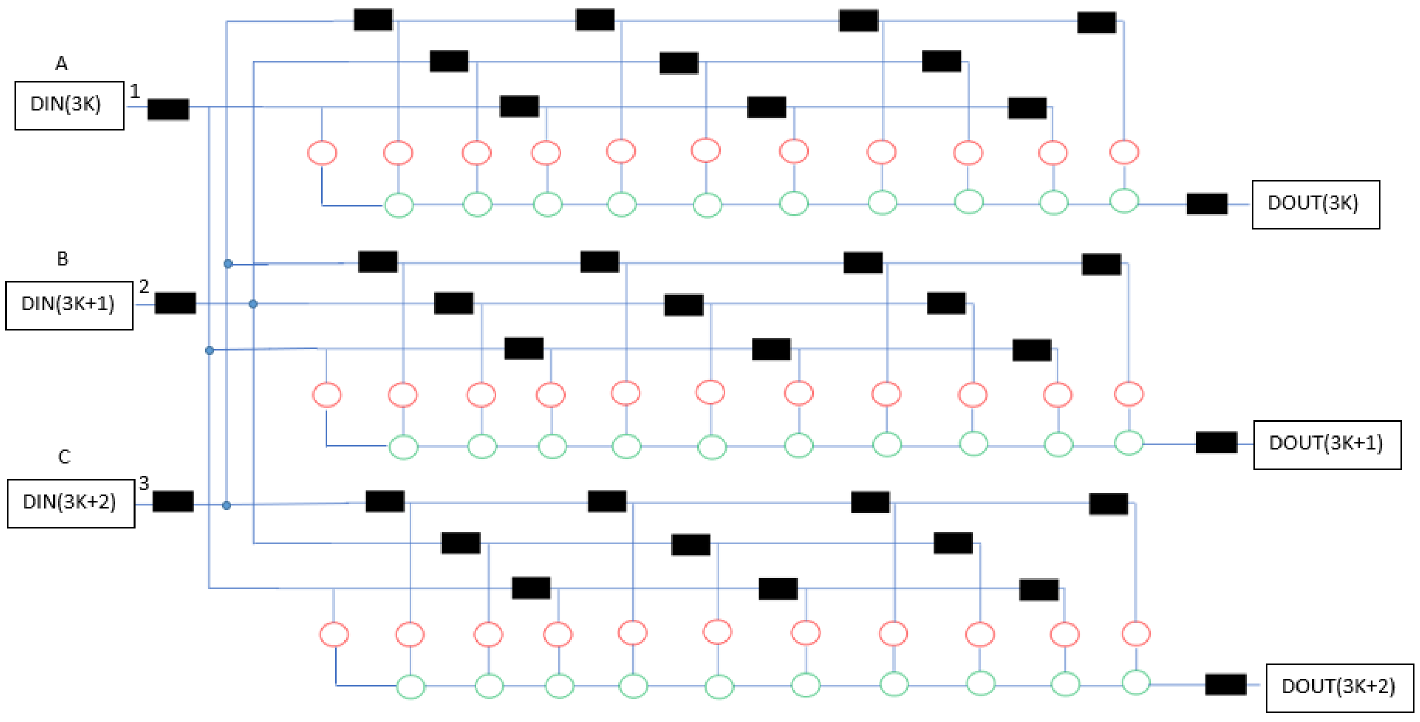


Figure 2.9: Unfolded architecture (black box = register, red circle = multiplier, green circle = adder)

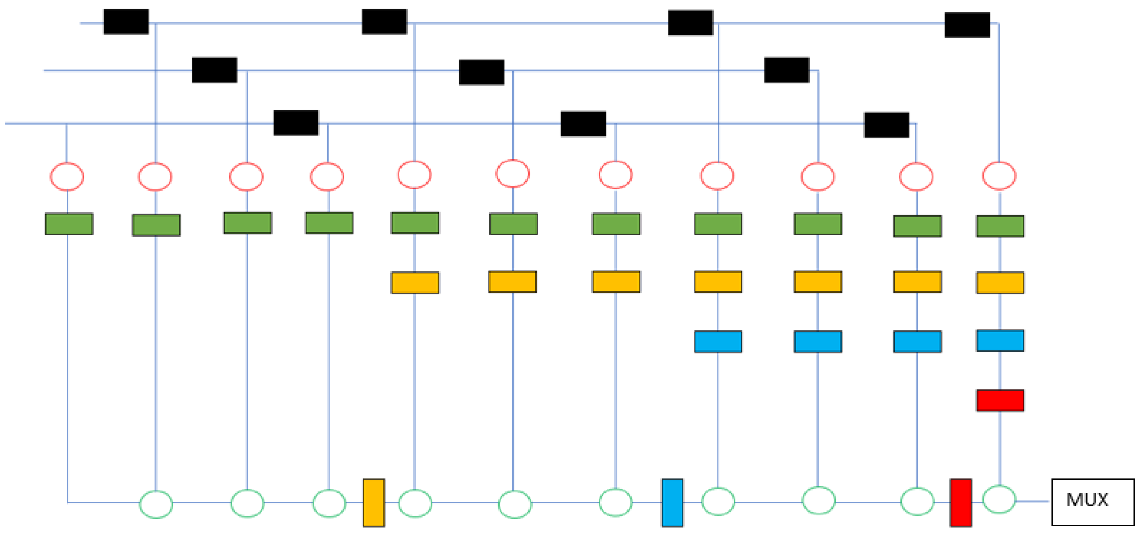


Figure 2.10: Pipelined architecture with different stages of registers

2.3.1 Starting architecture development

We kept the same logic as before considering now 3 inputs/outputs. The critical part is to connect each input to a multiplier by applying a certain delay depending on the unfolding circuit. Then, for the pipelining method, we have to add several signals in order to manage the different stages of registers at the correct timing.

2.3.2 Simulation

Regarding the simulation part, the programs are almost the same, we have just changed the data generator and data synchronizer files in order to manage 3 inputs/outputs.

2.3.3 Implementation

- Logic Synthesis

As expected, the maximum frequency is higher (more than doubled), and the area is almost three times bigger with respect to the previous architecture.

Max frequency [MHz]	Max area μm^2
653.59	21755.873870

Table 5: Post synthesis report

	Internal Power	Switching Power	Leakage Power	Total Power
µW	824.7650	478.7255	363.3365	1303.5
%	63	37		100

Table 6: Post synthesis report

- Place and Route

Max area μm^2
18195.2

Table 7: Post synthesis report

	Internal Power	Switching Power	Leakage Power	Total Power
µW	824.76501	478.7255	363.3365	1303.5
%	63	37		100

Table 8: Post synthesis report

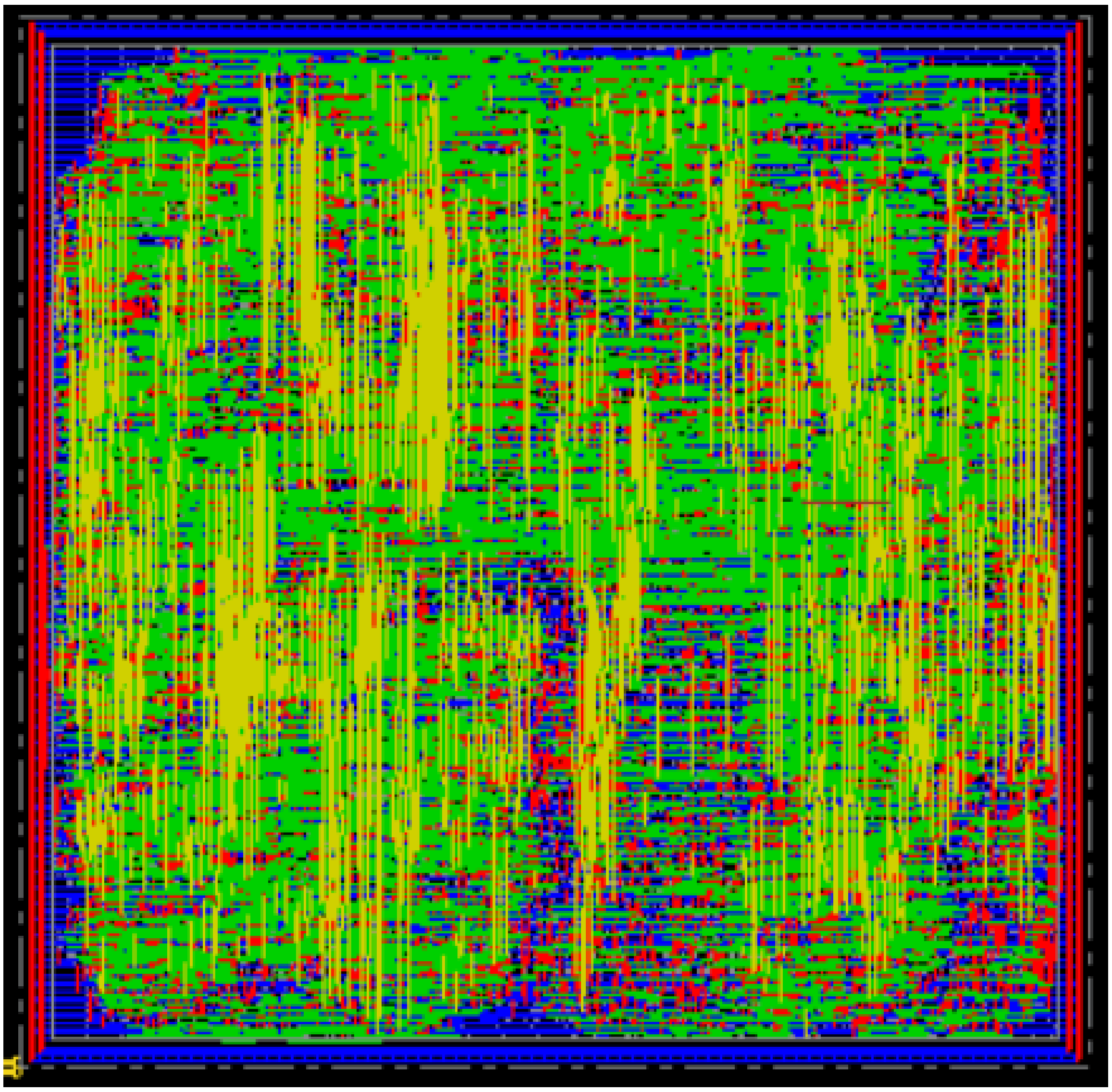


Figure 2.11: Innovus floorplan overview

3 Conclusion

During this laboratory, we have learnt how to implement a digital filter using: C language, matlab, VHDL and Verilog testbenches.
By using the softwares Synopsys Design Compiler and Cadence SOC Innovus we have simulated the real behavior of our basic and advanced filter. Synopsys allows as to estimate the achievable maximum frequency, the area and the power consumption.

Last and not least, by using the unfolding and pipelining techniques we were able to improve our filter and we have seen the impact of the changes on our design and its the previous parameters(Maximum frequency, area and power consumption).