Politecnico di Torino

Collegio di Elettronica, Telecomunicazioni e Fisica

Design, implementation and optimization of a FIR filter - Laboratory 1

Integrated Systems Architecture

Master degree in Electrical Engineering

Group 26

Gianpietro Noto, Dario Greto

November 21, 2022

Contents

CHAPTER 1

Reference model development

1.1 Filter Design

The aim of this laboratory is to design a Finite Input Response (FIR) digital filter. The design of the filter started by applying the algorithm described in the LAB assignment in order to compute some parameters useful for the design. The following results have been derived:

- Order: 8
- Resolution: 12 bit

The other imposed requirements are:

- Cutoff frequency: $2kHz$
- Sampling frequency: $10kHz$
- $THD < -30dB$. Area must be reduced as much as possible by finding the number of shift to perform on the multiplication and so lowering the resolution of the results.

1.1.1 Matlab Design

The obtained Order and the Resolution of the filter have been passed to the matlab scripts (my-fir_design.m and my_fir_filter.m); they provide the quantized coefficients of the filter.

b0	b1	b2	b3	b4	b5	b6	b7	b8
-13	-28	104	544	830	544	104	-28	-13

Table 1.1: b coefficients.

These scripts generate two sine waves (500Hz and 3.5kHz), which then are then added together and halved in amplitude. The meaning behind this process is to generate a signal that has both high (out of band) and low (in band) frequencies and to test if the the FIR correctly filters them according to specifications. The plot of the result is shown in Figure 1.1 .

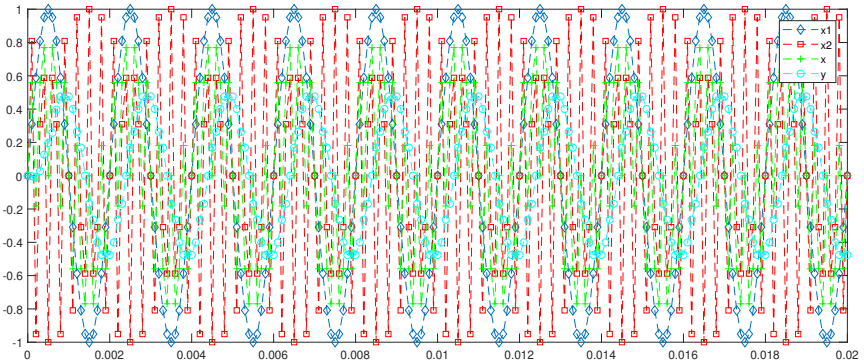

Figure 1.1: Input and output waveforms

As said before the matlab script has the role of designing the filter, a comparison between the Designed filter and the Quantized filter can be observed in Figure 1.2 .

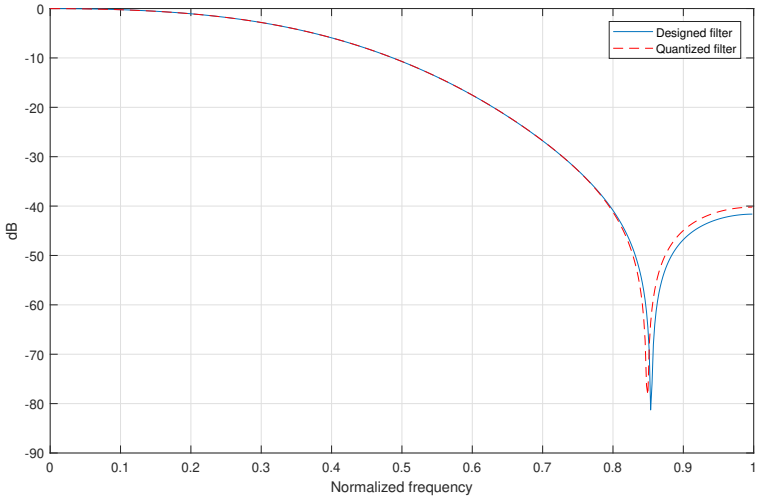

Figure 1.2: FIR transfer functions

Furthermore the matlab script writes the samples used to plot the output signal y of Figure 1.1 in the *"samples.txt"* file so they can be used for testing the c and VLSI implementation.

The Total Harmonic Distortion (THD) for this step has been evaluated with the quantized output data using the thd function of the DSP toolbox. Figure 1.3 shows the frequency spectrum of the signal obtained interpolating the output samples; as shown also in the figure, the THD obtained by the Matlab function is $THD = -72.8759dB$.

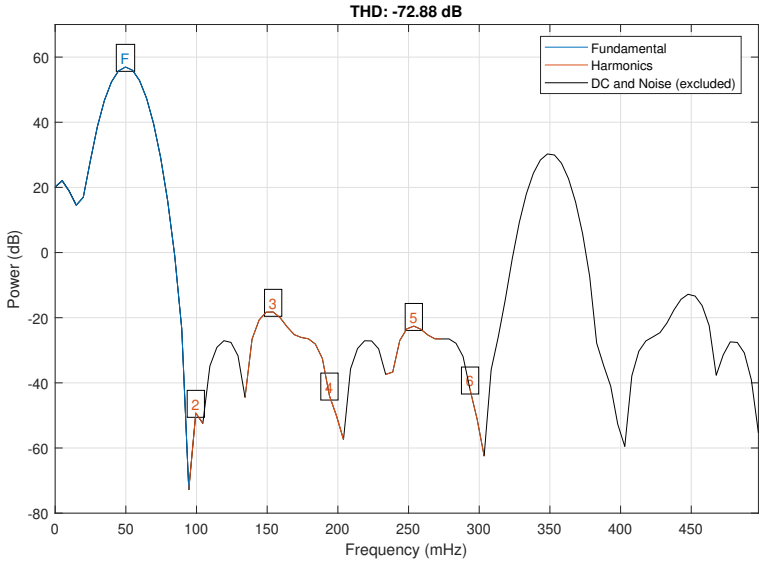

Figure 1.3: FIR transfer functions

1.1.2 Fixed-point C Model

A fixed-point model of the filtering operation has been implemented by means of a program written in C. Direct form I (canonical direct form) has been chosen for the FIR filter architecture. Hence, the main formula for this structure is:

$$y[n] = b_0 * x[n] + b_1 * x[n-1] + b_2 * x[n-2] + b_3 * x[n-3] + b_4 * x[n-4] + ...$$

$$... + b_5 * x[n-5] + b_6 * x[n-6] + b_7 * x[n-7] + b_8 * x[n-8]$$

The provided C program *myfilter.c* takes the samples from the file *samples.txt*, generated previously by the matlab implementation, as input and saves in the file *results_c.txt* the quantized output. This model is necessary as it is programmed to behave exactly like the VLSI implementation.
In a loop the program reads all the samples generated by the matlab script and applies the filtering function to it. At this step the multiplication output precision has been reduced performing the following steps:

1. The samples are multiplicated with the corresponding coefficient at full precision
2. The result is then shifted right to preserve only the desired number of MSB
3. A left shift is then performed to ensure that the output is on the desired (12) number of bits.

This algorithm works by preserving the sign of the shifted output sign, assuming that the integers in C are represented in 2's complement. It's important to note that at this step, to obtain the correct

result, the left shift had to be one bit less than the proposed one.
Again, the total harmonic distortion for this step has been calculated using the thd function of the matlab DSP toolbox, and by iteration it has been found that the minimum amount of bits at the output of the multiplier is $8bits$ with $THD = -33.0825dB$

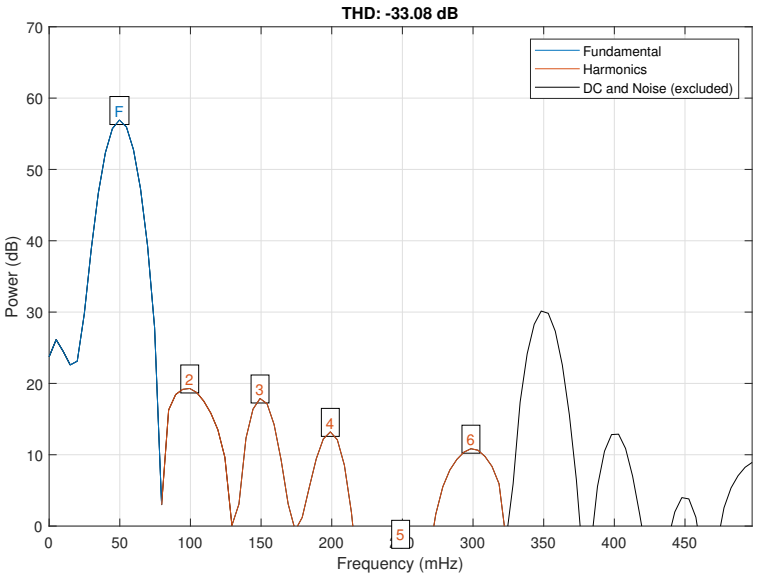

Figure 1.4: FIR transfer functions

In the two models used, Matlab and C, different results have been retrieved. This can be by analyzing the values present in the two output files (*results_m.txt* and *results_c.txt*) but also by looking at their THDs. This is due to the fact that a fixed-point approach has been used for the C model, while the Matlab one uses floating point representation.

CHAPTER 2

VLSI Basic Implementation

This chapter provides an explaination of the design choices and steps performed to obtain a FIR with the exact behaviour of the C program developed in the previous step.

2.1 Design

As first step a block scheme and a timing diagram of the desired architecture have been drawn and they are shown in Figure 2.1 and 2.2.

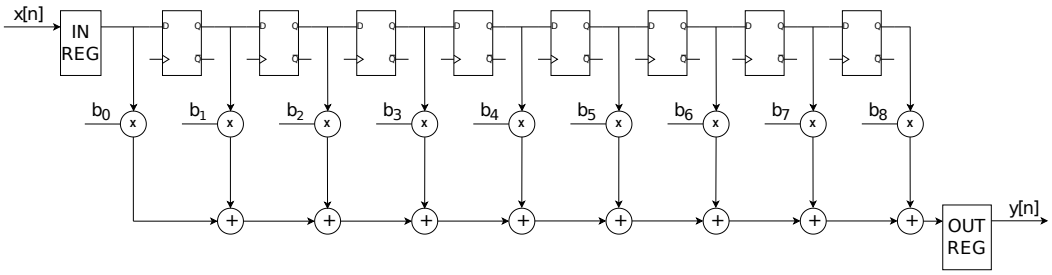

Figure 2.1: FIR filter

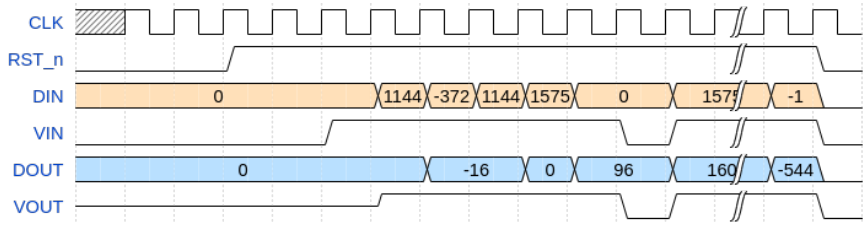

Figure 2.2: FIR timing

The involved input/output signals of the datapath are:

- DIN is the sample input
- VIN validates the input sample
- RST_n resets the FIR to the starting condition

- CLK is the clock signal
- DOUT is the filtered output sample
- VOUT asserts the validity of the output signal

The circuit is mostly designed in a single VHDL file, since the architectures of the adder and multiplier are considered as combinatory blocks. The only behavioural part is the description of the DOUT signal, for which, after the synthesis it has been noted that 2 flip-flops have been inferred. A custom register component, with a load signal, triggered by the VIN, has been used.

2.2 Simulation

The testbench setup consists in a few components that:

- Read the samples and feed it to the newly developed FIR architecture
- Verify that the output generated by the VHDL are the same as the C program and write the results to a file. At this step an error is reported if there are any difference in the files.
- Perform the actual testbench analyzing the UUT. This part has been written in verilog.

A bash script has been developed to prepare the simulation directories, dynamically compile all the needed files and launch vsim and source a tcl script with the actual simulation steps.
All the simulation parts run as expected. No Asserts have been triggered, *results_c.txt* and *results_vhdl.txt* are exactly the same. For reference and comparison with the design timing diagram Questa Sim screenshot have been included in figure

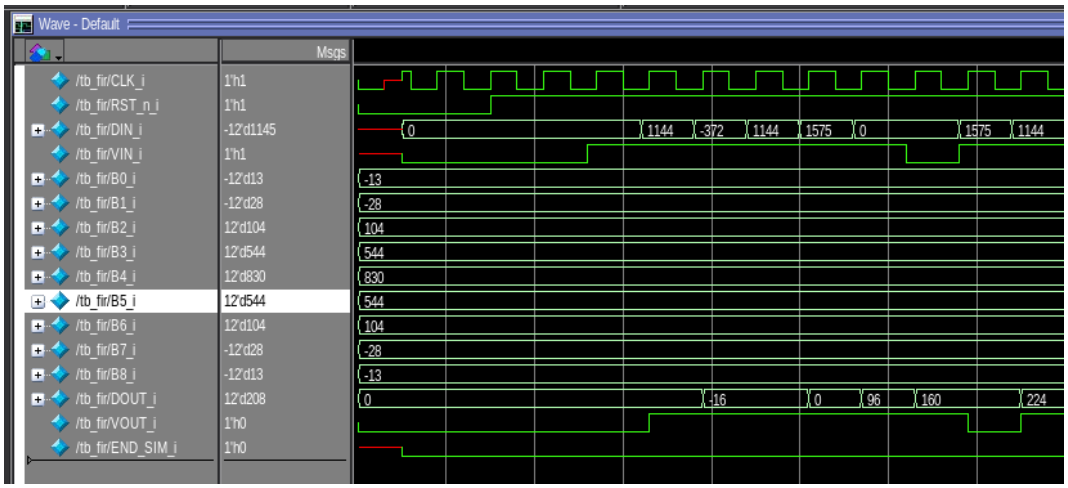

Figure 2.3: FIR simulation

2.3 Implementation

2.3.1 Synthesis

For the synthesis step another set of scripts has been designed. As in the previous step a bash script prepares and cleans up the directories and a tcl script is sourced into Design Compiler to perform the

actual synthesis steps.

The first request was to find the maximum clock frequency f_M that our design achieves with slack met and not equal to zero, by iteration of the synthesis script the minimum clock frequency achieved is $3.1ns$, so the maximum frequency is $f_M = 322.58MHz$.

Then the following request was to set the clock period multiplied by 4, so $12.4ns$, and the corresponding area and power reports.

Area, and switching activity are reported in the Table 2.1 and Table 2.2:

CLK	Combinational area	Buf/Inv area	Noncombinational area	Total cell area
$3.1ns$	5922.49 μm^2	279.56 μm^2	813.16 μm^2	6735.65 μm^2
$12.4ns$	5441.29 μm^2	209.60 μm^2	813.16 μm^2	6254.45 μm^2

Table 2.1: Area report

As expected the area is smaller with the more relaxed constraint. This may be related to the less strict limit on the CLK speed of the circuit, which for instance allows mapping the multiplier and adder operations on smaller blocks.

2.3.2 Switching-activity-based power consumption estimation

The switching-acticity-based power consumption estimation have been produced.

CLK	Cell internal power	Net switching power	Total dynamic power	Cell leakage power
$12.4ns$	203.4030 uW	117.8275 uW	321.2305 uW	130.9395 uW

Table 2.2: Power report

After the syntesis a verilog netlist referring to the 12.4 ns circuit synthesized circuit, has been generated. After a questa sim simulation it has been found that the waveforms were the same as the VHDL one and no asserts have been triggered.

2.3.3 Place & Route

This section is related to the Place and Route of the Fir filter using Cadence Innovus. A new directory called Innovus has been created in order to store all the results. Finally the obtained netlist produced by Innovus has been simulated in order to estimate the power consumption based on the switching activity.

In Figure 2.4 we can observe a graphical representation of the result of the place and route procedure.

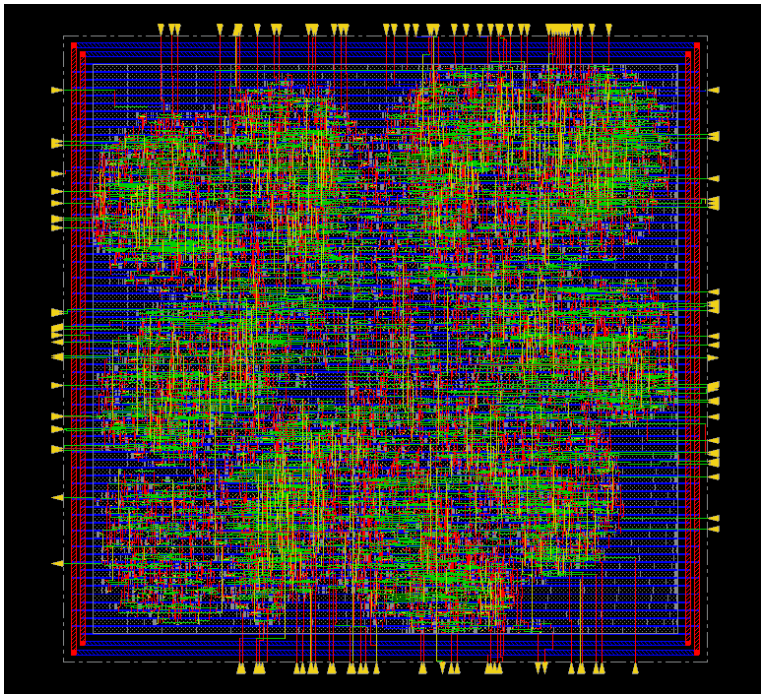

Figure 2.4: FIR place and route

The new netlist has been placed and routed using Innovus and following the assigned procedure. A check concerning the violation of the setup and hold times has been performed after the route phase by generating the corresponding reports. No negative slack has been found.
No errors have been reported throughout the whole procedure and after the connectivity and geometry check.
The generated area report has been found consistent (slightly different, but in an acceptable range) with the synthesis step; the obtaine area is $6219.612\mu m^2$.

The verilog netlist generated at this step has been once again exported and most of the steps done after the synthesis have been repeated. The simulation has been found compliant with all the others.

2.3.4 Switching-activity-based power consumption estimation

Finally the switching activity has been performed in order to estimate the power consumption of our design. As before a new script called *sw_act.sh* has been created in order to automate everything. The goal of this last step is to perform a switching-activity-based power estimation for the $f_{clk} = \frac{f_M}{4}$ circuit and verify that it behaves as expected using QuestaSim. A QuestaSim simulation has been launched using scripts saved in the folder *"sim"*. A .vcd file has been created in order to monitor all the signals inside our unit-under-test *myfir_post_route.v*, then the simulation has been started and we observed the generated waves, after that we checked the correctness of the results and at the end

the switching activity power estimation has been performed. After the simulation the folder syn has been selected as working directory, the previously generated .vcd has been converted into a .saif file and saved inside the saif directory. The following steps have been performed in the Synopsys Design Compiler where some commands have been inserted in the shell. The file .saif together with the post-synthesis netlist produces a power report named basing on the string that it receives in input. In this case, the output file has been called *report_power_sw_act_post_route.txt*. The main elements to be observed in this report are the one showed in table 2.3:

CLK	Cell internal power	Net switching power	Cell leakage power	Total dynamic power
12.4ns	245.9783 uW	179.0333 uW	1.2845e+05 nW	553.4636 nW

Table 2.3: Post Place & Route power report

As expected the post place and route power estimation is higher than the post synthesis one

CHAPTER 3

VLSI Advanced Implementation

In this chapter are detailed the design choices for the redesign of the filter after the following specification have been added:

- Reduce the critical path by adding pipelining
- Unfold the circuit with a degree of 3

3.1 Design

The formal unfolding method has been applied to the the graph in Figure 2.2 by using the equations method. The equations used to derive the new DFG are reported below.

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] + b_3 \cdot x[n-3] + b_4 \cdot x[n-4] + ...$$
$$... + b_5 \cdot x[n-5] + b_6 \cdot x[n-6] + b_7 \cdot x[n-7] + b_8 \cdot x[n-8]$$

The three different equations executed in parallel to perform the evaluations are:

- $y[3k] = b_0 \cdot x[3k] + b_1 \cdot x[3k-1] + b_2 \cdot x[3k-2] + b_3 \cdot x[3k-3] + b_4 \cdot x[3k-4] + ...$
$$... + b_5 \cdot x[3k-5] + b_6 \cdot x[3k-6] + b_7 \cdot x[3k-7] + b_8 \cdot x[3k-8]$$
- $y[3k+1] = b_0 \cdot x[3k+1] + b_1 \cdot x[3k] + b_2 \cdot x[3k-1] + b_3 \cdot x[3k-2] + b_4 \cdot x[3k-3] + ...$
$$... + b_5 \cdot x[3k-4] + b_6 \cdot x[3k-5] + b_7 \cdot x[3k-6] + b_8 \cdot x[3k-7]$$
- $y[3k+2] = b_0 \cdot x[3k+2] + b_1 \cdot x[3k+1] + b_2 \cdot x[3k] + b_3 \cdot x[3k-1] + b_4 \cdot x[3k-2] + ...$
$$... + b_5 \cdot x[3k-3] + b_6 \cdot x[3k-4] + b_7 \cdot x[3k-5] + b_8 \cdot x[3k-6]$$

The following step is to rewrite the equations in the form compliant with our needs:

- $y[3k] = b_0 \cdot x[3k] + b_1 \cdot x[3(k-1)+2] + b_2 \cdot x[3(k-1)+1] + b_3 \cdot x[3(k-1)] + b_4 \cdot x[3(k-2)+2] + ...$

  $... + b_5 \cdot x[3(k-2)+1] + b_6 \cdot x[3(k-2)] + b_7 \cdot x[3(k-3)+2] + b_8 \cdot x[3(k-3)+1]$

- $y[3k+1] = b_0 \cdot x[3k+1] + b_1 \cdot x[3k] + b_2 \cdot x[3(k-1)+2] + b_3 \cdot x[3(k-1)+1] + b_4 \cdot x[3(k-1)] + ...$

  $... + b_5 \cdot x[3(k-2)+2] + b_6 \cdot x[3(k-2)+1] + b_7 \cdot x[3(k-2)] + b_8 \cdot x[3(k-3)+2]$

- $y[3k+2] = b_0 \cdot x[3k+2] + b_1 \cdot x[3k+1] + b_2 \cdot x[3k] + b_3 \cdot x[3(k-1)+2] + b_4 \cdot x[3(k-1)+1] + ...$

  $... + b_5 \cdot x[3(k-1)] + b_6 \cdot x[3(k-2)+2] + b_7 \cdot x[3(k-2)+1] + b_8 \cdot x[3(k-2)]$

Then the architecture has been derived directly from the equations and an image of the filter architecture is shown in figure 3.1.

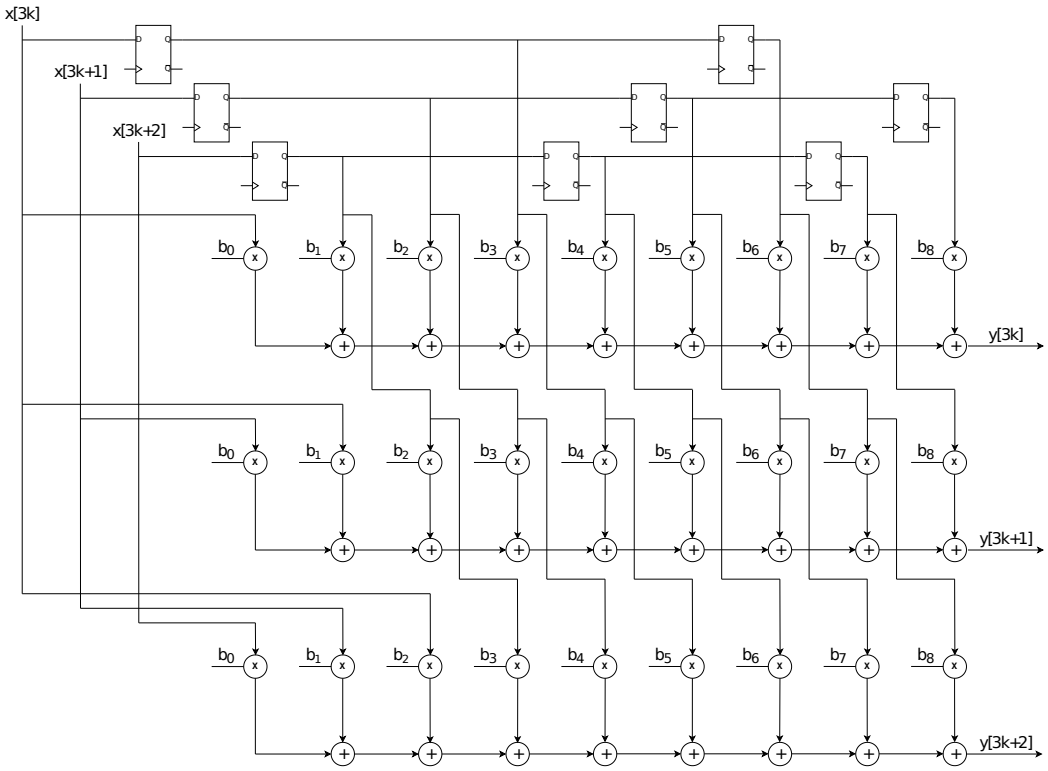

Figure 3.1: FIR Unfolded

After applying the unfolding, to get the maximum throughput possible, the critical path has been reduced introducing pipelining registers on the most critical feedforward cutsets. Since no description of delays of the components of the cell library in use has been found, a delay of 2 ut an 1 ut has been assigned to the multipliers and the adders respectively.
The throughput of the new architecture is then calculated as:

$$th = \frac{N}{T_{CP}} = \frac{3}{T_m + 2T_a} = 1$$

With the previous assumptions made about the delay of the multiplier and the adder this is the highest possible throughput with this level of unfolding. The full circuit is in figure 3.2:

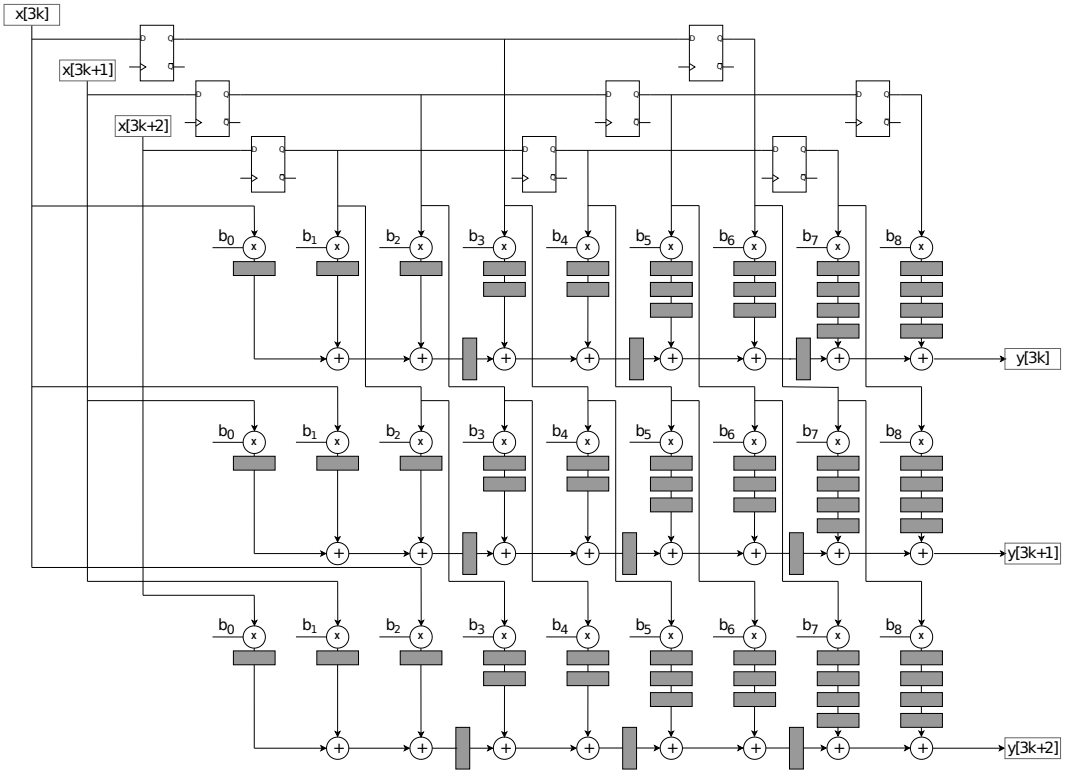

Figure 3.2: FIR Unfolded Pipelined

A new vhdl code, with a similar architecture to the previous one, has been written.

3.2 Simulation

The testbench files of the previous implementation have been sligtly edited to work with a MIMO unfolded implementation, but the architecture has been kept as close as possible to the original one. The code has been then tested in Questa Sim again following the steps done before and the values stored in the new file *resuls_vhd.txt* are exactly the same as the one produced by the standard filter implementation without unfolding and the one produced by the c program.
A screenshot of the simulation can be found in Figure 3.3 where we can observe the input values entering in the circuit 3 at a time along with a validation signal VIN and exiting from the circuit along with the validation signal VOUT:

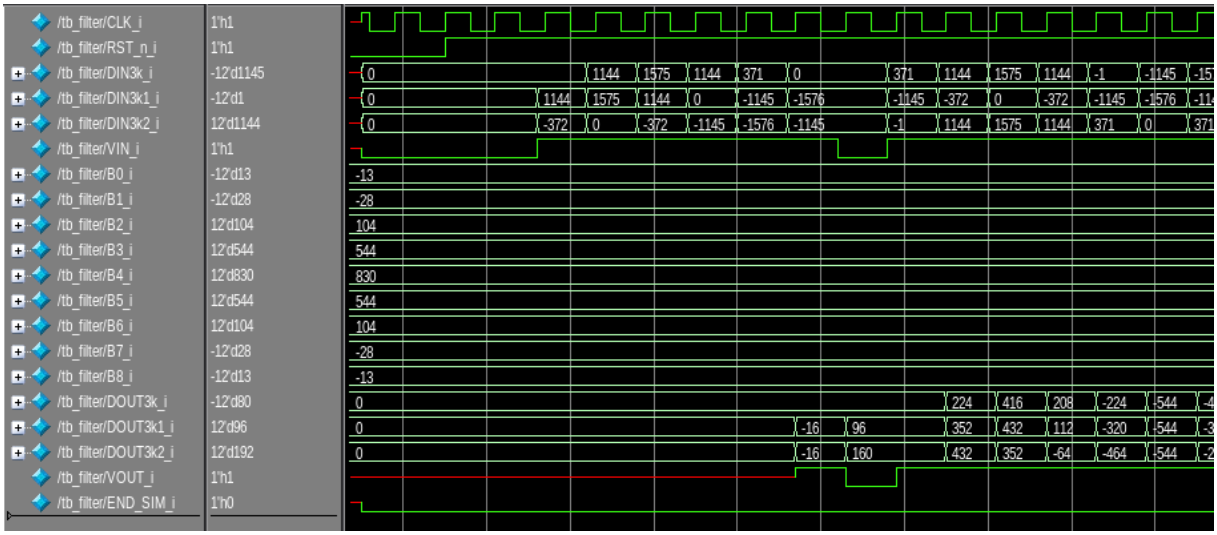

Figure 3.3: FIR Unfolded Pipelined Simulation

3.3 Implementation

3.3.1 Synthesis

Similarly to the previous chapter a set of scripts has been written to perform the synthesis steps.

The first request, again, was to find the maximum clock frequency f_M that our design achieves with slack met and not equal to zero, by iteration of the synthesis script the minimum clock frequency achieved is $2.5ns$, so the maximum frequency is $f_M = 400MHz$.

Then the following request was to set the clock period multiplied by 4, so $10ns$, and to find the corresponding area and power reports.

Area, and switching activity are reported in the following tables:

CLK	Combinational area	Buf/Inv area	Noncombinational area	Total cell area
$2.5ns$	19690.12 μm^2	1089.53 μm^2	5705.17 μm^2	25395.29 μm^2
$10ns$	17929.73 μm^2	765.02 μm^2	5703.57μm^2	23633.30 μm^2

Table 3.1: Area report

As expected the area is almost 3 times as the standard implementation. With the order 3 unfolding we have tripled the number of multipliers, adders and we increased the number of registers because of the pipelining.

3.3.2 Switching-activity-based power consumption estimation

The switching-acticity-based power consumption estimation have been produced by using the netlist produced by Synopsis in the previous step.

CLK	Cell internal power	Net switching power	Cell leakage power	Total dynamic power
$10ns$	989.19 uW	351.84 uW	4.7939e+05 nW	1.8204e+03 uW

Table 3.2: Power report

3.3.3 Place & Route

All the place and route steps done in the previous chapter have been performed again with the new implementation. No errors have been found at this time.

In figure 3.4 we can observe a graphical representation of the result of the place and route procedure.

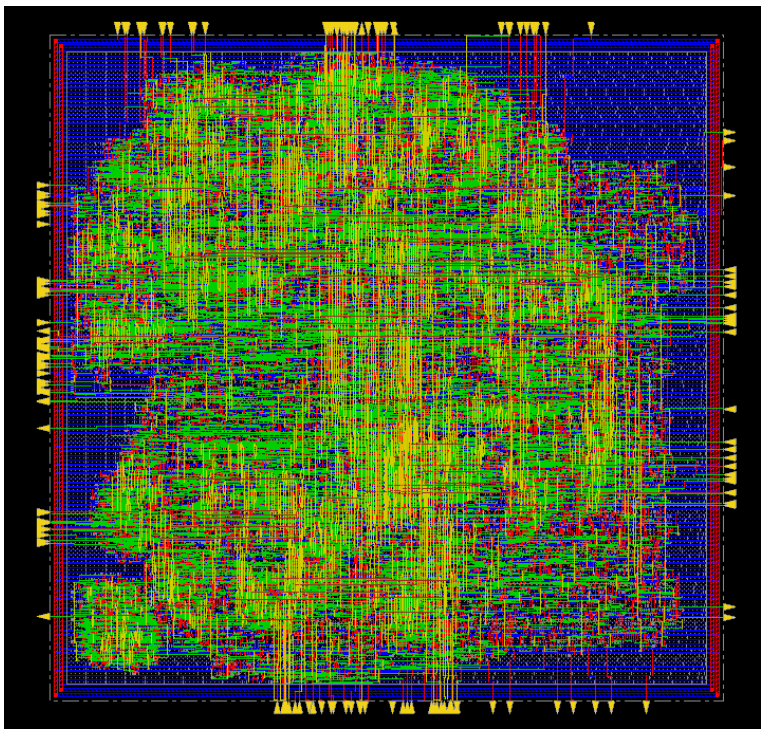

Figure 3.4: FIR place and route

The new netlist has been placed and routed using Innovus and following the assigned procedure. A check concerning the violation of the setup and hold times has been performed after the route phase by generating the corresponding reports. No negative slack has been found.
No errors have been reported throughout the whole procedure and after the connectivity and geometry check.
The generated area report has been found consistent (slightly different, but in an acceptable range) with the synthesis step; the obtained area is $23426.62 \mu m^2$.

The verilog netlist generated at this step has been once again exported and most of the steps done after the synthesis have been repeated. The simulation has been found compliant with all the others.

3.3.4 Switching-activity-based power consumption using QuestaSim

Finally the switching activity has been performed in order to estimate the power consumption of our design using the new netlist produced by Innovus. As before a new script called *sw_act.sh* has been created in order to automate everything. The goal of this last step is to perform a switching-activity-based power estimation for the $f_{clk} = \frac{f_M}{4}$ circuit and verify that it behaves as expected using QuestaSim. A QuestaSim simulation has been launched using scripts saved in the folder *"sim"*. A .vcd file has been created in order to monitor all the signals inside our unit-under-test *myfir_post_route.v*, then the simulation has been started and we observed the generated waves, after that we checked

the correctness of the results and at the end the switching activity power estimation has been performed. After the simulation the folder syn has been selected as working directory, the previously generated .vcd has been converted into a .saif file and saved inside the saif directory. The following steps have been performed in the Synopsys Design Compiler where some commands have been inserted in the shell. The file .saif together with the post-synthesis netlist produces a power report named basing on the string that it receives in input. In this case, the output file has been called *report_power_sw_act_unfolded_pipelined.txt*. The main elements to be observed in this report are:

The most significant power report are reported in the following table 2.3

CLK	Cell internal power	Net switching power	Cell leakage power	Total dynamic power
10.0ns	995.4624 uW	539.1526 uW	4.7164e+05 nW	2.0063e+03 uW

Table 3.3: Post Place & Route power report

3.3.5 Switching-activity-based power consumption using Innovus

The last request was performing another power estimation, using the vcd file produced by Questa Sim and the netlist produced by Innovus, it has been found that the values were slightly lower, probably because of the further optimization introduced by Innovus. The new values can be found in the following table 3.4

CLK	Total internal power	Total switching power	Total leakage power	Total power
10.0ns	720.31 uW	503.17 uW	441.27 uW	1.664e+03 uW

Table 3.4: Post Place & Route power report from Innovus

As expected the post place and route power estimation is higher than the post synthesis.

A notable difference with the simple implementation is that the power is around 4 times more. As explained above this is to be expected due to the triplication of most of the logic.