Politecnico di Torino

III Facoltà di Ingegneria

Laboratory 2
Digital arithmetic

Master degree in Electrical Engineering

Authors: Group 21

Nicola Dilillo, Stefano Moncalvo, Lorenzo Carrano

November 20, 2021

Many thanks to Prof. Mariagrazia Graziano for providing us with this template.

Contents

CHAPTER 1

Reference model development

1.1 Introduction

The goal of this laboratory is to design a Finite Impulse Filter filter (FIR) with a cut frequency of 2 kHz. Filter has is design according two parameter: order and number of bits. The order employ for the following filter is 10 and the number of bits are 9.

Before starting with filter design it's needed to develop a prototype version that ensure the final result of request implementation.

1.2 Design the filter with Matlab/Octave

First step is the generation of coefficients. To do this Matlab function fir1 has been used. The coefficients are shown in table 1.1.

Number	Quantize	Normalize
0	-1	1
1	-2	1
2	-4	1
3	8	0
4	35	1
5	50	1
6	35	1
7	8	1
8	-4	1
9	-2	1
10	-1	1

Table 1.1: All coefficients.

Always staying in Matlab and using the previous coefficients, a further Matlab script is executes in order to perform different simulation with prototype filter with a cut-off frequency of 2 kHz and a sampling frequency of 10 kHz. The input signal used is an average between two sinusoidal waves respectively at 500 HZ and 4.5 kHz. After this execution two files have been generated:

1. *sample.txt*, which contains the sample values that have fed the input of FIR;

2. *result.txt*, which contains the output values that has been elaborated from our FIR.

1.3 C prototype

A C program language has been written to have a fixed point implementation of FIR that use the following formula:

$$y_i = \sum_{n=0}^{10} x_{i-n} \cdot b_n$$

Thanks this program is possible to evaluate the performance of fixed version respect to Matlab execution.

1.3.1 Evaluate the THD

The purpose of this script is to evaluate the Total Harmonic Distortion (THD) trying to react a value that is maximum -30dB. If an amount of tollerance is avaiable maybe will be possible to reduce the bit numbers and so to reduce the size of FIR design.

In first hand, with 9 bits used for data, the THD is -40 dB. Trying to reduce the number of bits to 8 the value of THD obtain is still acceptable, it's -33 dB. When a further reduction has been applied the value of THD a not allowed value is returned, with 7 bits THD is -27dB.

In the end, for the final implementation of FIR 8 bits have been used in order to achieve the THD request and to reduce the area.

CHAPTER 2

VLSI implementation

2.1 Starting architecture development

The purpose of this section is to develop in VHDL the architecture of the previously designed filter. The architecture of the filter is composed by four elements:

- Adders
- Multipliers
- Flipflops
- Registers

The 8-bit input is recived and then propagated through a chain of 10 registers; the output of each register is multiplied by the corresponding coefficient, and the results are summed together to form the filter's output. All registers use VIN as an enable signal, in order to avoid unwanted propagation of data. The VIN signal, delayed of two clock cycles, is also used to drive VOUT. Every input and output signal is loaded or produced by registers or flipflops, to reduce the risk of interference from external signals.

2.2 Simulation

The design was simulated using a testbench written in both Verilog and VHDL. The testbench is composed of four disinct entities:

- **clk_gen:** generates a clock signal of the specified frequency, and a reset signal.
- **data_maker:** reads the samples.txt file and provides an input every clock cycle and its validity using the VIN signal.
- **data_sink:** recives the outputs of the filter every clock cycle and writes them in the output.txt file if VOUT is equal to 1.
- **tb_fir:** is the testbench top entity written in Verilog.

At the end of the simulation the values stored in Output.txt were compared with the ones produced by the C prototype. The two files are equal, which means that the filter is behaving correctly.

2.3 Logic synthesis

After the simulation the design must be synthetized. To estimate the maximum working clock frequency of the filter, the clock period in the design compiler is set to 0 ns. In this way the compiler optimizes the circuit as much as possible, and the negative slack of the timing report corresponds to the maximum clock frequency.

After running the synthesis at the computed frequency the area is evaluated.

It is requested to set the frequency to 25% of the maximum value. After the synthesis a new area estimation is produced.

The constraints on the clock have a significant role in the estimation of the area: by allowing the frequency to be lower, the size of the circuit will be smaller.

The Design Compiler produces the Verilog netlist of the synthetized circuit and a .sdf file containing the circuit's delays. Those files are used by Modelsim to simulate the netlist with the correct timing parameters and obtain the switching activity of the nodes, which is saved in a .vcd file. The results of this simulation have been checked to assure that they are coherent with the ones of the original circuit.

This file is converted to .saif and used by Design Compiler to generete a power report.

2.4 Place & Route

The last section requires to perform the place and route on the synthetized circuit to obtain the switching activity and power report. To do that using Innovus several steps are necessary:

- **Structuring the floorplan**, where Innovus allocates the area for the cells;
- **Inserting power rings**, two rings for power (VDD) and ground (VSS) are inserted around the floorplan;
- **Standard cell power routing**, horizzontal wires for power and ground are prepared for the cells;
- **Placement**, the cells are placed in the floorplan but are still to be connected between them;
- **Post Clock-Tree-Synthesis optimization**, the design is optimized to achieve the required timing constraints;
- **Place filler**, filler cells are placed to ensure continuity in n+ and p+ wells in each row;
- **Routing**, the cells are connected among each other;
- **Post routing optimization**, the design is optimized again to achieve the timing constraints. After this step, the design is saved as a .enc file;
- **Parasitics extraction**, Innovus extracts the parasitic values of resistencies and capacitances;
- **Timing analysis**, the performance of the circuit is evaluated, if the slack is negative the constraints are violated;
- **Design analysis and verification**, Innovus checks for the presence of floating wires and violations on the constraints on the geometric features of the circuit. Finally the area and gate count, the netlist and a file with delay annotations are saved.

Since the slack values for setup and hold are positive and the verification on connectivity and geometry returned no errors, the final step is to simulate the produced netlist with Modelsim, to check if the circuit behaves correctly and to calculate the switching activity. The following results have been obtained:

CHAPTER 3

Advanced architecture development

In this section the purpose is to improve the FIR performance. In first and unfolding technique has been used to improve the throughput and then pipeline technique has been used to reduce the longest path and improve maximum frequency.

3.1 Unfolding

Unfolding of order 3 has been applied to FIR filter (N = 3) and the equations derived to build the new system are the following:

$$\begin{aligned} y[3n] = a_0 \cdot x[3n] + a_1 \cdot x[3(n-1)+2] + a_2 \cdot x[3(n-1)+1] + \\ a_3 \cdot x[3(n-1)] + a_4 \cdot x[3(n-2)+2] + a_5 \cdot x[3(n-2)+1] + a_6 \cdot x[3(n-2)] + \\ a_7 \cdot x[3(n-3)+2] + a_8 \cdot x[3(n-3)+1] + a_9 \cdot x[3(n-3)] + a_{10} \cdot x[3(n-4)+2] \end{aligned} \tag{3.1}$$

$$\begin{aligned} y[3n+1] = a_0 \cdot x[3n+1] + a_1 \cdot x[3n] + a_2 \cdot x[3(n-1)+2] + \\ a_3 \cdot x[3(n-1)+1] + a_4 \cdot x[3(n-1)] + a_5 \cdot x[3(n-2)+2] + a_6 \cdot x[3(n-2)+1] + \\ a_7 \cdot x[3(n-2)] + a_8 \cdot x[3(n-3)+2] + a_9 \cdot x[3(n-3)+1] + a_{10} \cdot x[3(n-3)] \end{aligned} \tag{3.2}$$

$$\begin{aligned} y[3n+2] = a_0 \cdot x[3n+2] + a_1 \cdot x[3n+1] + a_2 \cdot x[3n] + a_3 \cdot x[3(n-1)+2] + \\ a_4 \cdot x[3(n-1)+1] + a_5 \cdot x[3(n-1)] + a_6 \cdot x[3(n-2)+2] + a_7 \cdot x[3(n-3)+1] + \\ a_8 \cdot x[3(n-2)] + a_9 \cdot x[3(n-3)+2] + a_[10] \cdot x[3(n-3)+1] \end{aligned} \tag{3.3}$$

Using this method of optimization the two more input and output port have been added becouse now 3 inputs are processed and produce, at the same time, 3 output. To overall throughput has been triplicate.

3.2 Pipeline

A further optimization has been applied. This method allows to reduce the size of critical path.

From schematic of previous FIR is possible to see that the critical path is the chain off adders. To achieve the best result is not necessary split each of them, but it is needed to group them in order to have a total time path that is less than the amount of time for performed a multiplication.