Integrated System Architecture

Lab 1 Report

Academic Year 2020-2021

Professors:	Masera Guido	
	Martina Maurizio	
	Valpreda Emanuele	
	Walid Walid	
Students:	Lagostina Lorenzo	288019
	Lanza Mauro	290766
	Tudisco Antonio	285433
Group Number:	5	

Contents

1 Reference Model Development

1.1 Overview

The aim of the whole laboratory was to create a filter, in the case of our group, a Finite Impulse Response filter with a cut-off frequency of 2 kHz and a sampling frequency of 10 kHz.

1.2 Matlab pseudo-fixed-point

The first thing to do was to order the surnames of the group alphabetically and check the number of characters in the first two surnames. Since the order is:

Name	Number of Characters
Lagostina	9
Lanza	5
Tudisco	7

After that we had to apply the following formulas to obtain the order of the filter (N) and the number of bits (n_b).

$$N = 2^p * [(x \bmod 2) + 1] + 6 * p$$

$$n_b = (y \bmod 7) + 8$$

With $x = 9$, $y = 5$ and $p = 1$ we get the following value as N and n_b

$$N = 10$$

$$n_b = 13$$

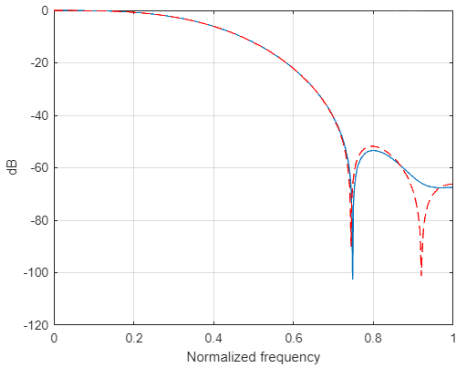
Now we adapted the Matlab script *my_fir_filter.m* with the values found for N and n_b.

And we found the following values as results:

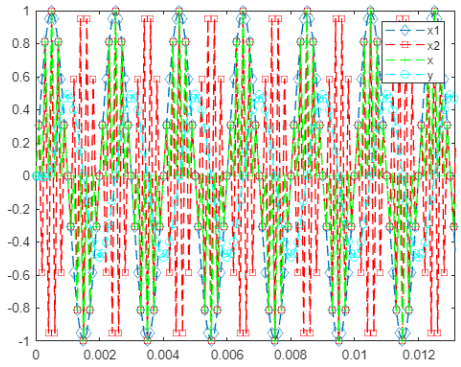
0	-1	-17	-33	38	264	662	1154	1616	1920
2018	1921	1633	1187	623	-1	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	-1	623	1187	1633	1921
2018	1921	1633	1187	623	0	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	-1	623	1187	1633	1921
2018	1921	1633	1187	623	-1	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	0	623	1187	1633	1921
2018	1921	1633	1187	623	0	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	0	623	1187	1633	1921
2018	1921	1633	1187	623	-1	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	-1	623	1187	1633	1921
2018	1921	1633	1187	623	0	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	0	623	1187	1633	1921
2018	1921	1633	1187	623	-1	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	-1	623	1187	1633	1921
2018	1921	1633	1187	623	0	-624	-1188	-1634	-1922
-2019	-1922	-1634	-1188	-624	0	623	1187	1633	1921
2018	1921	1633	1187	623	-1	-624	-1188	-1634	-1922
-2019	-	-	-	-	-	-	-	-	-

And the following samples:

0	1265	0	3313	0	4095	-1	3313	-1	1265
-1	-1266	-1	-3314	-1	-4096	0	-3314	0	-1266
0	1265	0	3313	0	4095	-1	3313	-1	1265
-1	-1266	-1	-3314	-1	-4096	0	-3314	0	-1266
0	1265	0	3313	0	4095	0	3313	-1	1265
-1266	-1	-3314	-1	-4096	0	-3314	0	-1266	0
1265	0	3313	0	4095	0	3313	-1	1265	-1
-1266	-1	-3314	-1	-4096	0	-3314	0	-1266	0
1265	0	3313	0	4095	-1	3313	-1	1265	-1
-1266	0	-3314	0	-4096	0	-3314	-1	-1266	-1
1265	0	3313	-1	4095	-1	3313	-1	1265	-1
-1266	-1	-3314	-1	-4096	-1	-3314	-1	-1266	0
1265	-1	3313	0	4095	0	3313	0	1265	-1
-1266	0	-3314	0	-4096	0	-3314	-1	-1266	0
1265	0	3313	0	4095	-1	3313	-1	1265	0
-1266	-1	-3314	-1	-4096	-1	-3314	0	-1266	0
1265	-1	3313	0	4095	0	3313	0	1265	-1
-1266	-1	-3314	0	-4096	0	-3314	0	-1266	0
1265	0	3313	-1	4095	-1	3313	0	1265	0
-1266	-1	-3314	0	-4096	-1	-3314	0	-1266	0
-1	-	-	-	-	-	-	-	-	-



(a) Normalized Frequency



(b) Sine waves

Always with the same script we also found the two graphs shown above.

1.3 C model fixed-point

For the C program we had to implement the following operation:

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

The first thing that done was setting the number of coefficients $NT = 11$ and of bits $NB = 13$. Then we put the coefficients in the constant arrays a and b

$$a[NT-1] = \{6, 26, 46, 66, 86, 106, 126, 146, 166, 186\}$$

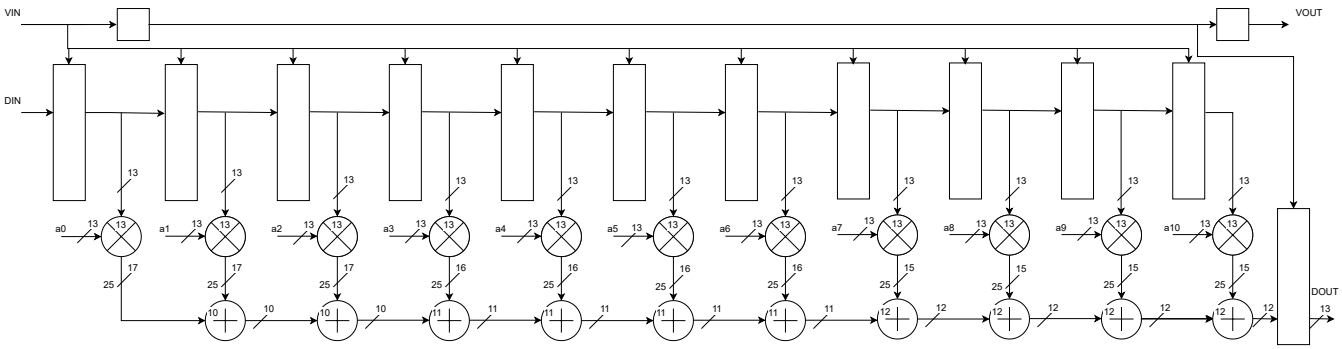
$$b[NT] = \{-1, -52, -102, -106, 1125, 1630, 1125, 260, -102, -52, -1\}$$

We then checked the results of the C program finding out that was the one we were expecting. The last step was to use the *thd* function in Matlab to get the Total Harmonic Distortion. Our aim was to get as close as possible to $-30dB$ without going above it. We managed to find a $THD = -32.08dB$.

2 VLSI Implementation

2.1 Starting architecture development

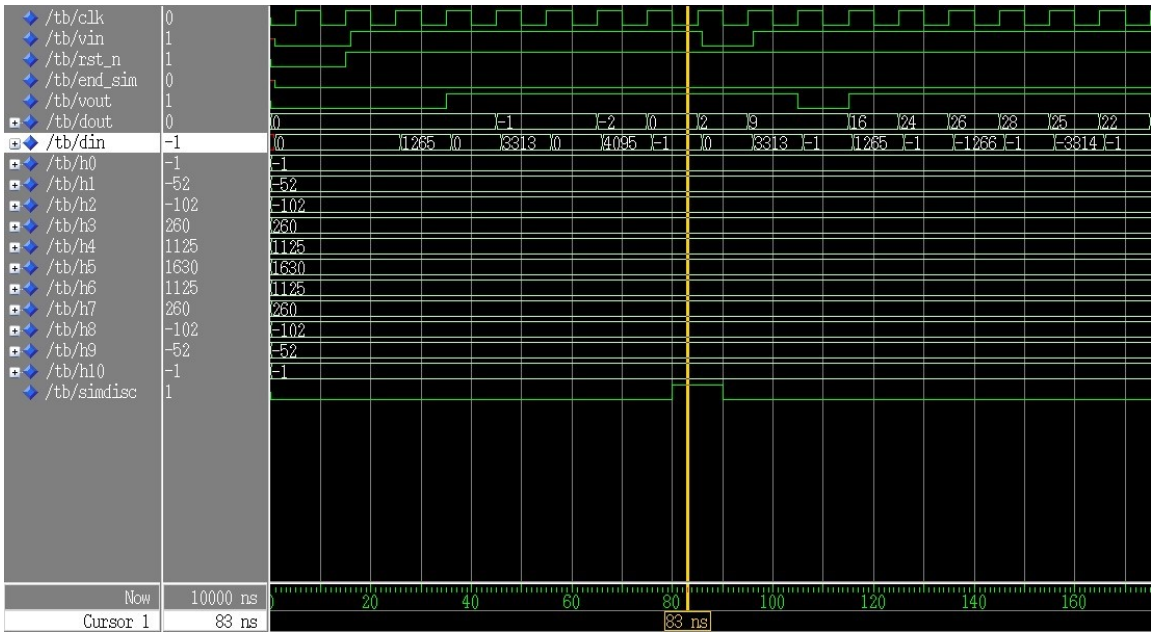
We realized the architecture using the block diagram described in the lectures. We truncated the results coming from the multipliers, keeping just the 9 MSBs.
In order to further reduce the area of our filter, we realized the adders with different parallelism. In order to avoid overflow the parallelism increases with the logarithm of 2. This choice is due to the fact that when adding two numbers on N bits, the maximum result is equivalent to a multiplication by 2, i.e. a result on N+1 bits. For the case of 4 added numbers we will have N+2 bits, and so on...
A generic loop was used to implement the filter, and this allowed for the use of the log_2 of the index inside the generic parameter of the adder which decides its parallelism.
Follows the block diagram of the FIR filter.



First implementation of the filter

2.2 Simulation

We wrote a test-bench in order to verify the correctness of the system. We verify that the results obtained are the same with respect the C model and also the behaviour of the system if a not valid signal is applied (VIN equals to 0).



Result of the simulation

2.3 Logic synthesis

Synopsis Design Compiler was used for the synthesis with 45nm CMOS library. After the synthesis the back-annotation procedure with Modelsim allowed for a more precise power estimation.
After finding the critical path t_{CP}, the circuit was resynthesized with a clock period of $4t_{CP}$. The table below shows the results in terms of timing, area and power

t_{CP}	f_M	T_{CK}	f_{CK}
3.84ns	260.427MHz	15.36ns	65.104MHz
t_{CP} post syn	4.99ns	Area	$9855.831\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$771\mu W$	$579.3\mu W$	$198.36\mu W$	1.549mW

Table 2.1: Synthesis results

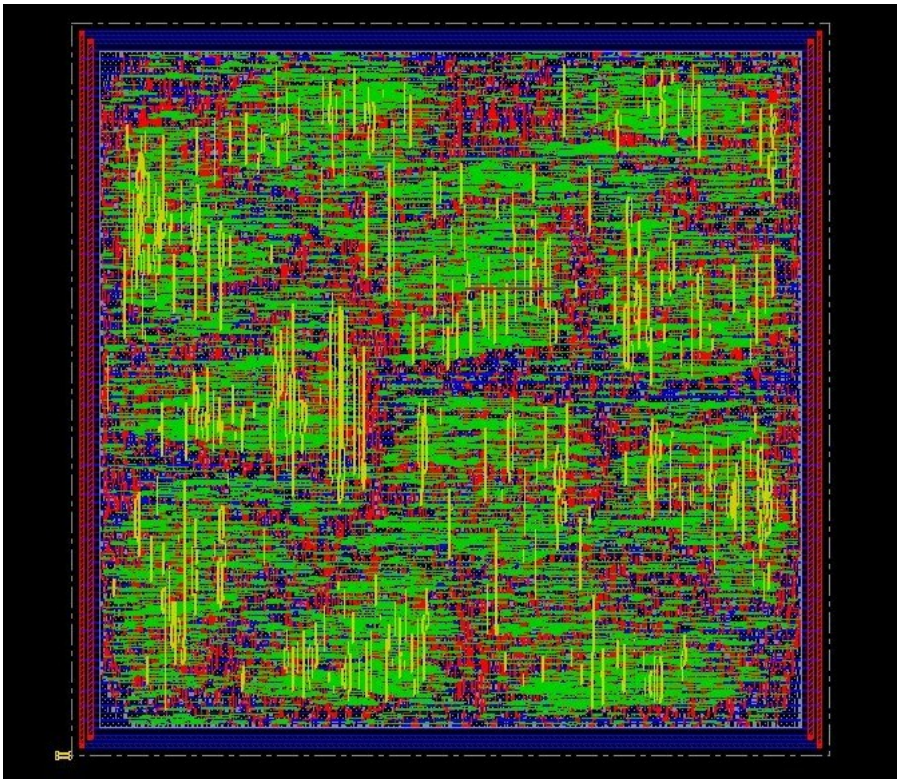


Figure 2.1: Innovus Post-CTS Snapshot

2.4 Place & Route

The first thing done in the Place $ Route operation is to create a *.globals* file that loads into innovus all the necessary libraries. Then we followed the inctructions to create the netlist and the perform all the operations. After setting $f_{clk} = f_M/4$ we tested the script on Innovus and generated the following results

t_{CP}	f_M	T_{CK}	f_{CK}
4.99ns	200.4MHz	15.36ns	65.1MHz
t_{CP} post P&R	11.3ns	Area	$9793.3\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$942.47\mu W$	$729.55\mu W$	$198.29\mu W$	1.84mW

Table 2.2: Place & Route results

We can observe how the internal and dynamic power of the circuit significantly increased because of the interconnections inside of the IC.

3 Advanced Architecture Development - Unfolding & Pipelining

In order to obtain the unfolded configuration of the filter, it was necessary to design it on paper, understanding how the different signals are connected.
From the theory, it's possible to understand that the component at the level i is connected to the component at the level j with w by using those formulas:

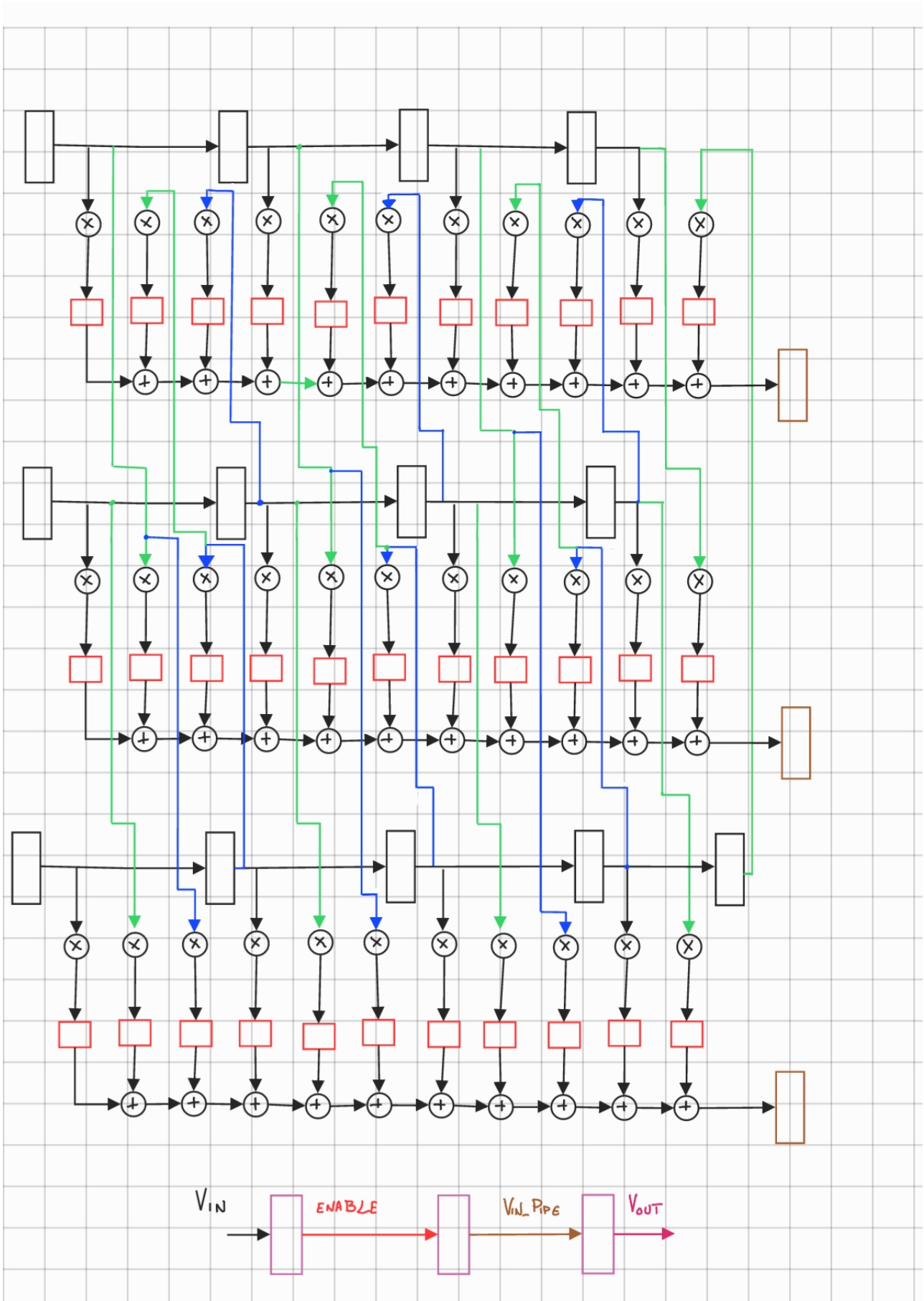
$$j = (i + w)\%N \tag{3.1}$$

$$w_i = \frac{i + w}{N} \tag{3.2}$$

So, analyzing just the edge that connects the input data to each multiplier, it's possible to obtain the following results:

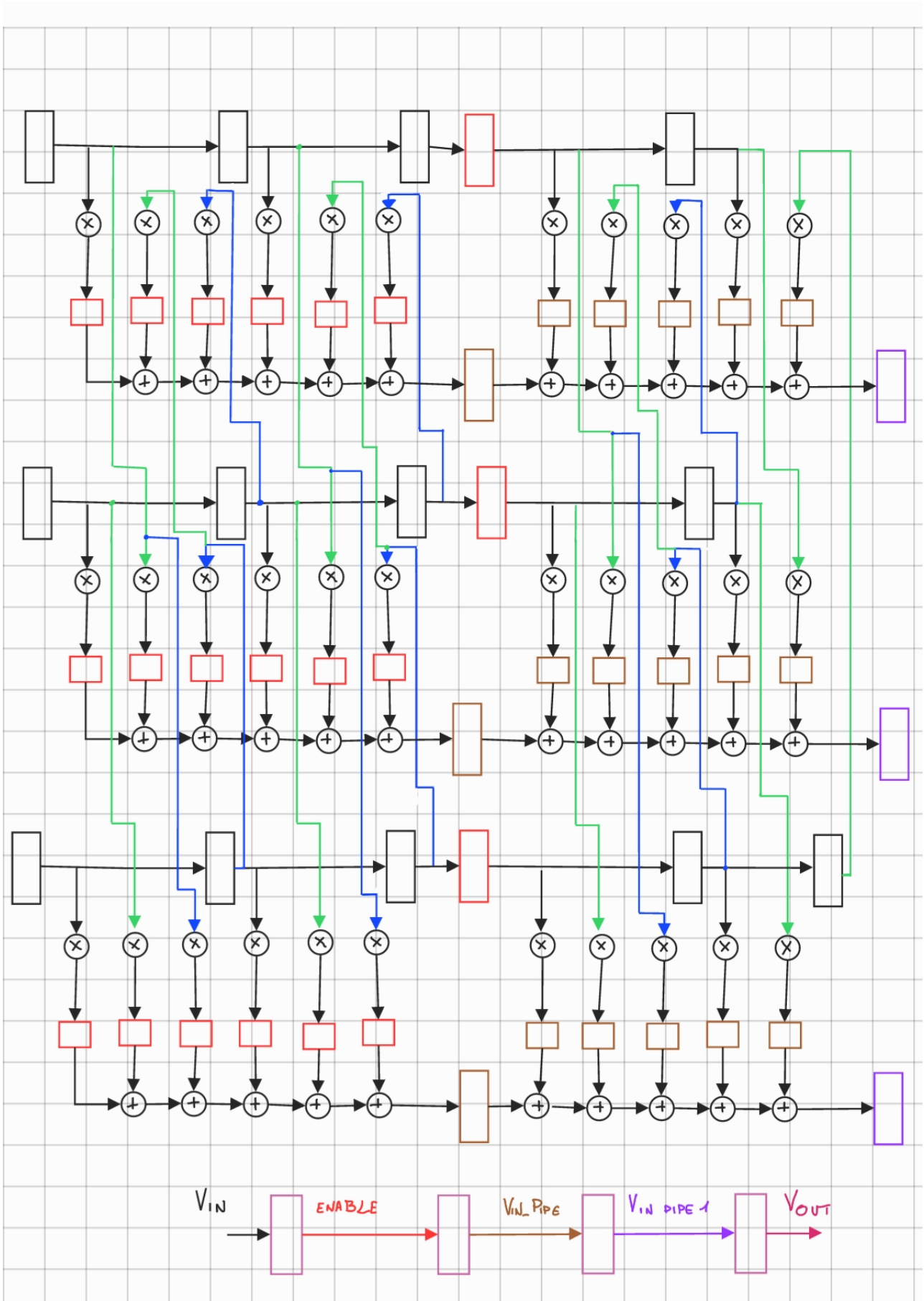
Edge	j(i = 0)	w_i (i = 0)	j(i = 1)	w_i (i = 1)	j(i = 2)	w_i (i = 2)
IN → MUL0	0	0	1	0	2	0
IN → MUL1	1	0	2	0	0	1
IN → MUL2	2	0	0	1	1	1
IN → MUL3	0	1	1	1	2	1
IN → MUL4	1	1	2	1	0	2
IN → MUL5	2	1	0	2	1	2
IN → MUL6	0	2	1	2	2	2
IN → MUL7	1	2	2	2	0	3
IN → MUL8	2	2	0	3	1	3
IN → MUL9	0	3	1	3	2	3
IN → MUL10	1	3	2	3	0	4

Then, we know that the pipeline can be applied for the **feed forward cut-set**. We have firstly identified 3 feed forward cut-set that divides the result of the multiplier from the chain of adders.



Scheme of the first version of unfolding and pipeline

In this scheme the enable signal of the black registers come from the VIN signal, for the red ones is the enable signal while for the brown one come VIN_PIPE. Then the VIN_PIPE pass through a register in order to obtain VOUT.
We have also identified an other feed-forward cutset, that cross the divides the first 5 adders from the others 5.

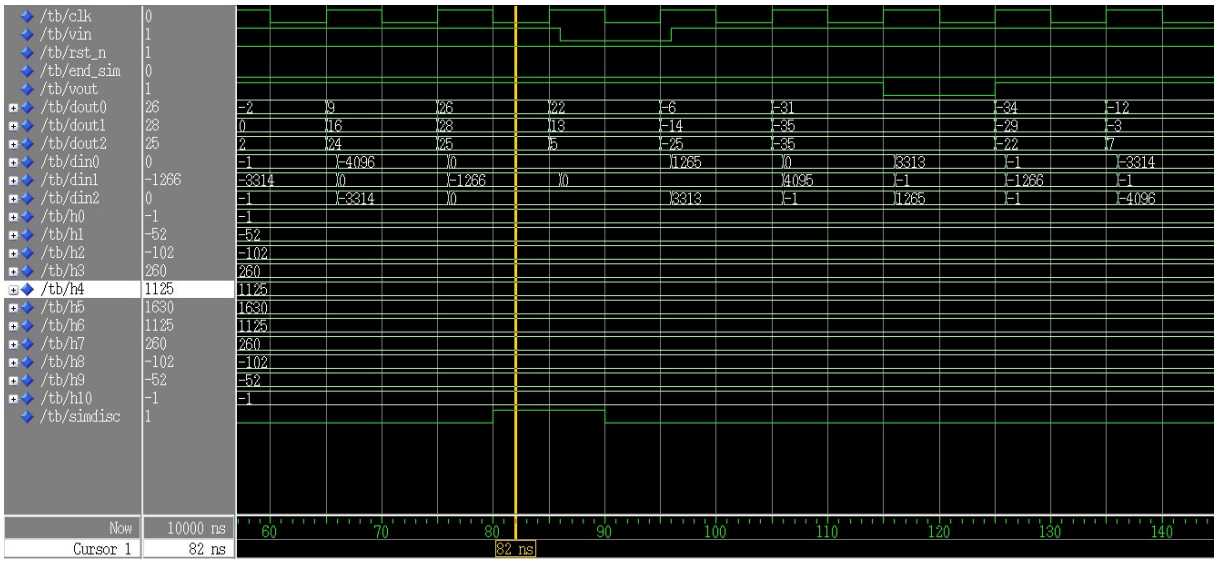


Scheme of the second version of unfolding and pipeline

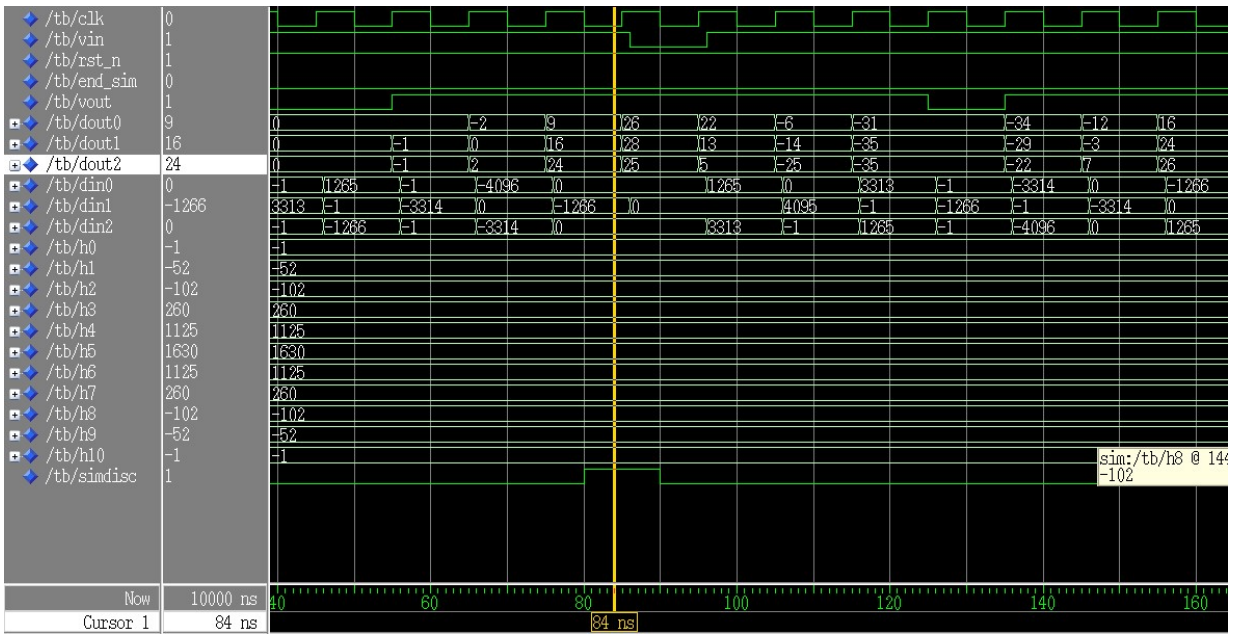
3.1 Starting architecture development

3.2 Simulation

By means of our test-bench, we verify the correctness of the circuit. As can be seen from the comparison of the results obtained from the simulation and the results of the C program, there is no difference. We also verify the correctness of the system if the VIN signal change.



Simulation first version of unfolding and pipeline



Simulation second version of unfolding and pipeline

3.3 Logic synthesis

Following are displayed the results of the logic synthesis, performed as described before.

First version

t_{CP}	f_M	T_{CK}	f_{CK}
2.4ns	416.7MHz	9.6ns	104.17MHz
t_{CP} post syn	3.42ns	Area	$28844.51\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$850.93\mu W$	$325.15\mu W$	$589.8\mu W$	1.766mW

Table 3.1: Synthesis results

Second version

t_{CP}	f_M	T_{CK}	f_{CK}
1.8ns	555.55MHz	7.2ns	138.9MHz
t_{CP} post synthesis	3.42ns	Area	$29412.15\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$918.3\mu W$	$339.15\mu W$	$524.6\mu W$	1.858mW

Table 3.2: Synthesis results

We can observe the drastic improvement in the critical path, which in the second version drops to almost half of the original one. The area is roughly triplied, as we would expect from a triplication of the allocated hardware. The leakage power also follows this trend, while the other power metrics do not. We expect the P&R procedure to deliver much higher values for overall power dissipation.

3.4 Place & Route

The first thing done in the Place $ Route operation is to create a *.globals* file that loads into innovus all the necessary libraries. Then we followed the inctructions to create the netlist and the perform all the operations. After setting $f_{clk} = f_M/4$ we tested the script on Innovus and generated the following results

First version

t_{CP}	f_M	T_{CK}	f_{CK}
2.4ns	416.67MHz	9.6ns	104.17MHz
t_{CP} post P&C	6.63ns	Area	$28650.3\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$4.333mW$	$3.385mW$	$579.7\mu W$	8.3mW

Table 3.3: Place & Route results

Second version

t_{CP}	f_M	T_{CK}	f_{CK}
1.8	555.55MHz	7.2ns	138.89MHz
t_{CP} post	4.25ns	Area	$29240\mu m^2$
Int. Power	Sw. Power	Leak. Power	Tot. Power
$5.92mW$	$4.52mW$	$591.1\mu W$	11.03mW

Table 3.4: Place & Route results

The final result is a drastical increase of power compsumtion from the first architecture, which is explained both by the increase in hardware and interconnection, and both by the increase of the clock frequency. The pipeline version deliver another increment of internal and dynamic power, with a negligible increase of the leakage power.

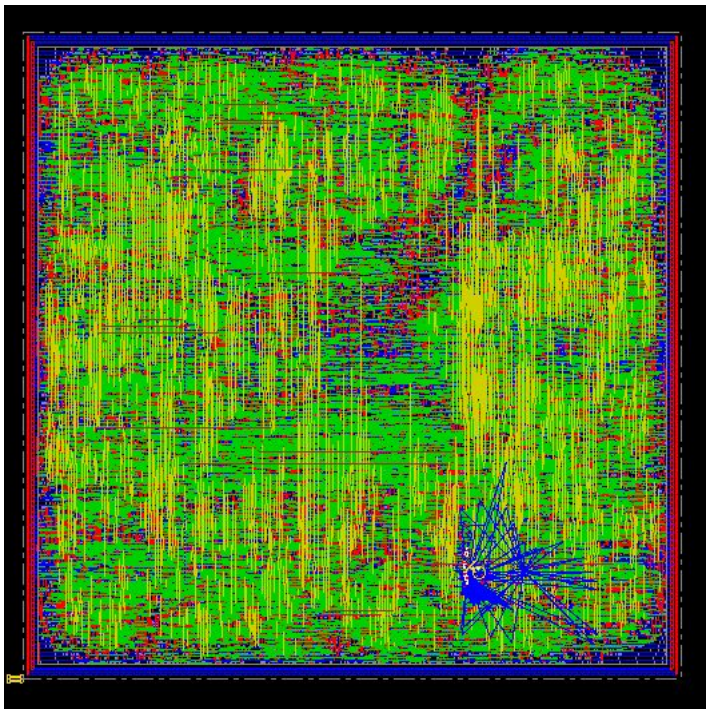


Figure 3.1: Innovus Pipelined First version

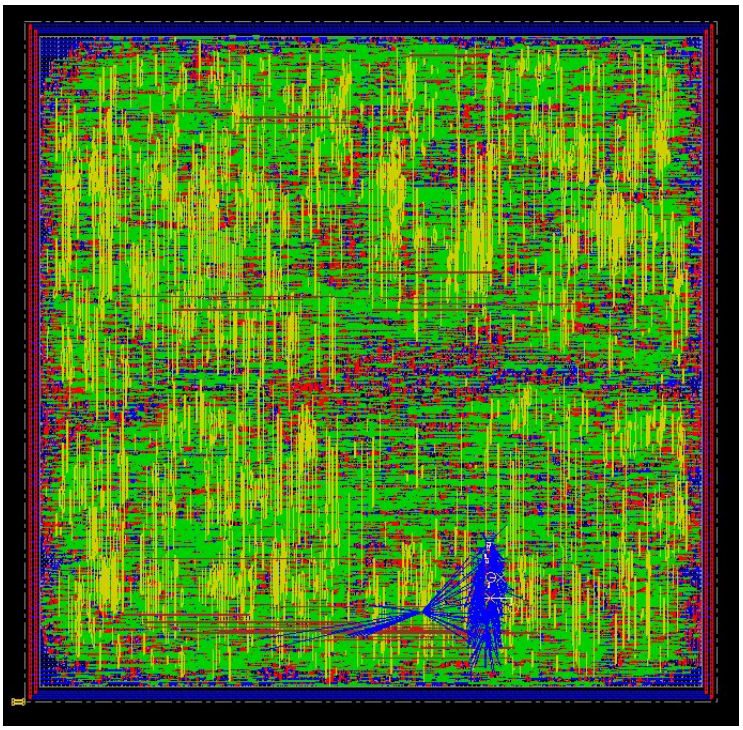


Figure 3.2: Innovus Pipelined Second version