



Politecnico di Torino
Collegio di Elettronica, Telecomunicazioni e Fisica

Report for the course
Integrated Systems Architecture

Lab 1: design and implementation of a digital filter

Master degree in Electrical Engineering

Group: 18

Names: Filippo Rinieri 317894, Mario Porcaro 315888,

Paolo Palma 306046, Sabina Olenic 318768

Signature 1:

Signature 2:

Signature 3:

Signature 4:

Signing this document the authors declare that, the contents and organization of this report constitute their own original work and do not compromise in any way the rights of third parties.

Contents

1 Reference model development	2
1.1 Matlab model	2
1.2 C model and THD	4
1.3 Explanations, comparisons and comments	4
2 VLSI implementation	6
2.1 Architecture	6
2.1.1 Timing diagram	7
2.1.2 VHDL description file	7
2.1.3 Simulation Pre-Synthesis	7
2.2 Logic synthesis	9
2.2.1 Reports without clock gating	9
2.2.2 Reports with clock gating	13
2.2.3 Simulation with clock gating	16
2.3 Place and route	17
2.3.1 Simulation with clock gating after Place and Route	20
2.4 Explanations, comparisons and comments	20
3 Advanced architecture development	22
3.1 Applying unfolding technique	22
3.1.1 VHDL description file	22
3.2 Simulation 3-unfolded architecture pre-synthesis	25
3.3 Pipeline application	26
3.3.1 4-stages pipeline	26
3.3.2 5-stages pipeline	27
3.3.3 10-stages pipeline	28
3.4 Simulation 3-unfolded with 10-pipeline stages pre-synthesis	30
3.5 Logic synthesis	31
3.5.1 Simulation 3-unfolded with 10-pipe without clock gating	34
3.6 Clock gating issue	35
3.6.1 Simulation 3-unfolded with 10-pipe with clock gating	41
3.7 Place and route	42
3.7.1 Simulation 3-unfolded with 10-pipe with clock gating after Place and Route	45
3.8 Explanations, comparisons and comments	45
4 Appendix	47
4.1 Matlab code	47
4.2 VHDL code	49

4.2.1 Standard architecture	49
4.2.2 Unfolded architecture	56
4.3 Text files	75

Appendix

4.1	myfir_design.m	48
4.2	clk_gen.vhd	49
4.3	data_maker_new.vhd	50
4.4	data_sink.vhd	52
4.5	myfir.vhd	53
4.6	tb_fir.v	55
4.7	data_maker_new_unfolded.vhd	56
4.8	data_sink_unfolded.vhd	58
4.9	myfir_unfolded.vhd	59
4.10	myfir_unfolded_pipe_10stages.vhd	62
4.11	myfir_unfolded_pipe_5stages.vhd	67
4.12	myfir_unfolded_pipe_4stages.vhd	71
4.13	tb_filter_unfolded.v	74
4.14	samples.txt	75
4.15	resultsm.txt	79
4.16	resultsc.txt	82
4.17	results_hdl.txt	85

CHAPTER 1

Reference model development

1.1 Matlab model

The first step is to define the characteristics of the finite impulse response filter. Solving the expressions given in the description file, it is obtained an order $N = 10$ and a number of bits $n_b = 8$. The frequency response of such filter is shown in Figure 1.1 where can be compared the behavior of the transfer function of the designed filter (in blue) and the one of the filter with quantized coefficients (in red). The latter is less accurate with respect to the first one where the precision is infinite.

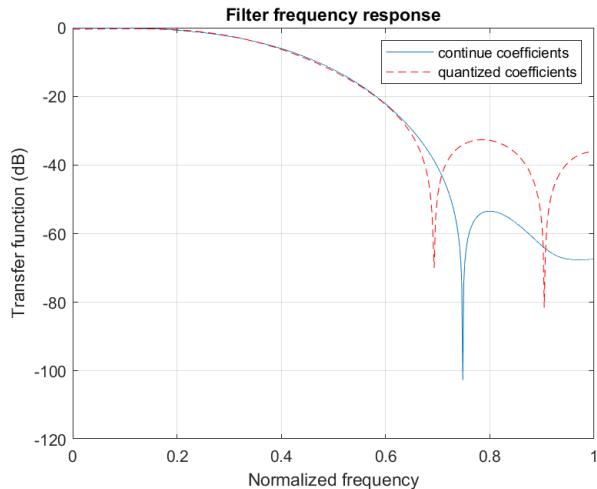


Figure 1.1: Frequency response of the FIR filter designed

Since the cutoff frequency of the filter is $F_c = 2$ kHz, the input signal (shown in Figure 1.2) is obtained by the linear combination of a sinewave x_1 at frequency 500 Hz (which is in-band) and a sinewave x_2 at frequency 3.5 kHz (out-of-band).

Filter coefficients are represented in *1.7 format*: they have one bit for the integer part and seven bits for the fractional part. They correspond to the integer values shown in Table 1.1. It can be observed the symmetric shape of the constants.

Table 1.2 shows the fractional values of the coefficients. As can be seen, $|b_j| < 1$ for each j which means that the quantized values are correctly represented.

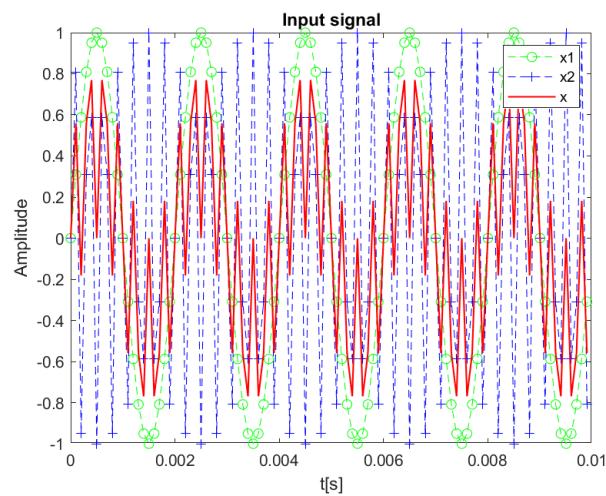


Figure 1.2: Input signal given by the superposition of an in-band and out-of-band signal

Constant	Value
b_0	-1
b_1	-2
b_2	-4
b_3	8
b_4	35
b_5	50
b_6	35
b_7	8
b_8	-4
b_9	-2
b_{10}	-1

Table 1.1: Integer values of coefficients

Constant	Value
b_0	-0.0078
b_1	-0.0156
b_2	-0.0313
b_3	0.0625
b_4	0.2734
b_5	0.3906
b_6	0.2734
b_7	0.0625
b_8	-0.0313
b_9	-0.0156
b_{10}	-0.0078

Table 1.2: Fractional values of coefficients

1.2 C model and THD

Coefficients obtained in the previous section are used to implement the filter in a script written in C language and it is evaluated the Total Harmonic Distortion (THD), varying the shift amount (variable SHAMT in the file .c) in order to get a value not greater than -30 dB. Results are shown in figure 1.3.

Higher the shift amount, lower the precision of multiplication products: with SHAMT equal to 8 the constraint is respected with the less number of bits for multiplications.

Using this value the number of bits after each multiplication is 7 and the THD is -32.29 dB.

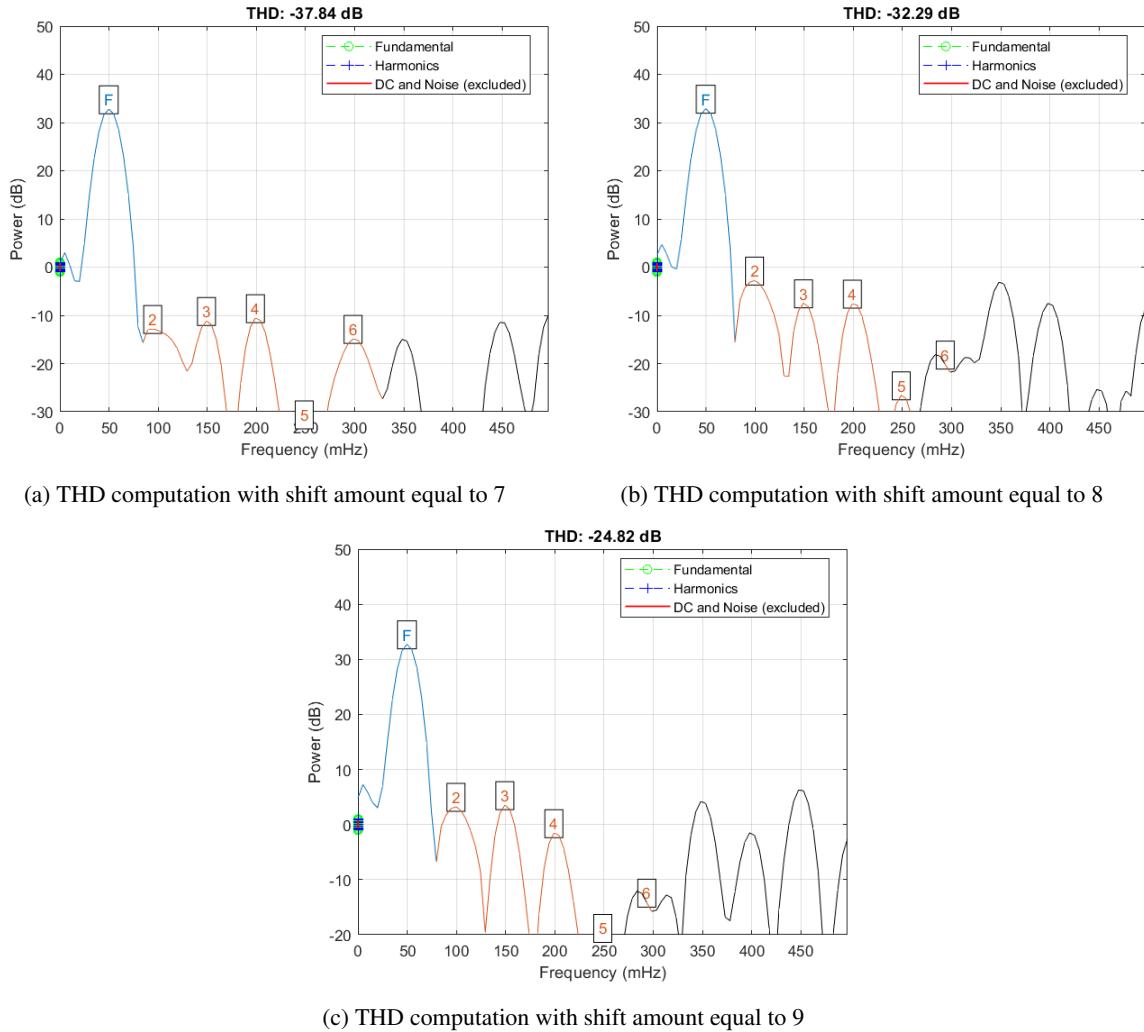


Figure 1.3

1.3 Explanations, comparisons and comments

The graph in Figure 1.4 compares the output values computed with the Matlab code and the ones calculated with the C code. The function obtained with the C values is slightly shifted downwards due to the floor applied at every shift operation.

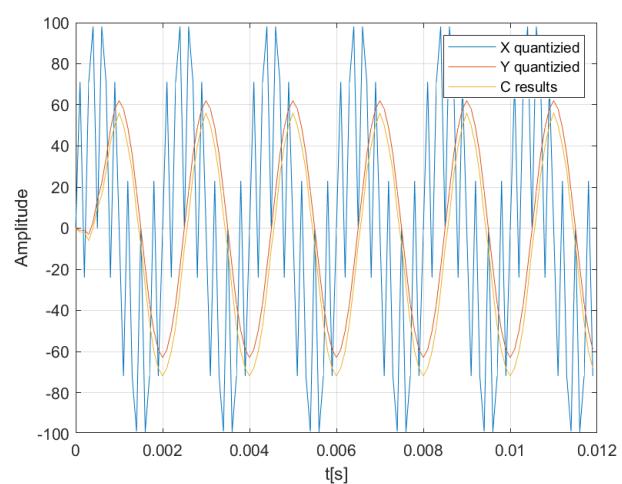


Figure 1.4: Comparison of the output filter in the various cases

CHAPTER 2

VLSI implementation

According to the assignment, the interface (shown in Figure 2.1) consists of an input data vector D_{IN} triggered by a validation signal V_{IN} , an array of eleven constants named from b_0 to b_{10} sampled at each iteration, an output data vector D_{OUT} and the associated validation signal V_{OUT} . The machine is synchronous with an external positive-edge clock CLK and has an asynchronous active-low reset signal.

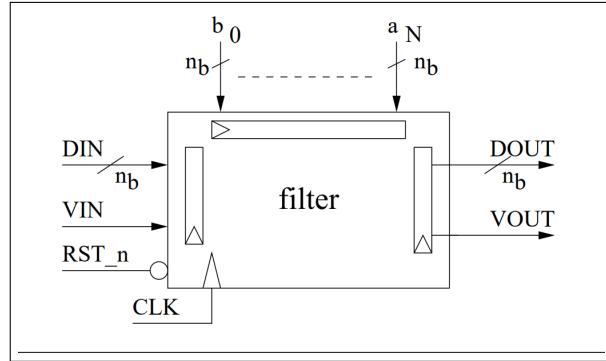


Figure 2.1: Filter interface

2.1 Architecture

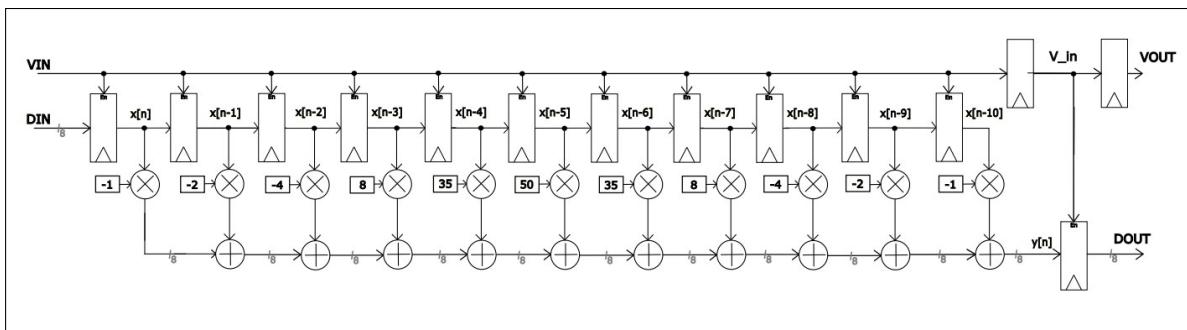


Figure 2.2: Filter architecture in direct form

The architecture is depicted in Figure 2.2. The filter expression is shown in the equation 2.1 and is imple-

mented in the direct-form; the parallelism of data is $n_b = 8$ bits and the order $N = 10$ leads to having eleven b constants; as the filter has a finite response it isn't present any loop feedback.

$$y[n] = \sum_{i=0}^{10} b_i \cdot x[n-i] \quad (2.1)$$

Eleven 8-bit registers named from $x[0]$ to $x[10]$ are placed to store old data samples, each sample is multiplied by the corresponding b constant and the results are summed; the filtered data is stored in an output register and it is available until the next sample is processed. According to this implementation the output is available after one clock cycle, so the validation output signal is obtained delaying by one register V_{IN} .

2.1.1 Timing diagram

The timing diagram is reported in Figure 2.3.

It is shown the one cycle-latency from input sampling to output, both in the case of single execution (D0 sample) and continuous execution (D1 to D3 samples). It can be seen that throughput is $T = \frac{1}{T_c}$ and so the maximum performance can be achieved by reaching the minimum clock period, as it is evaluated during the logical synthesis.

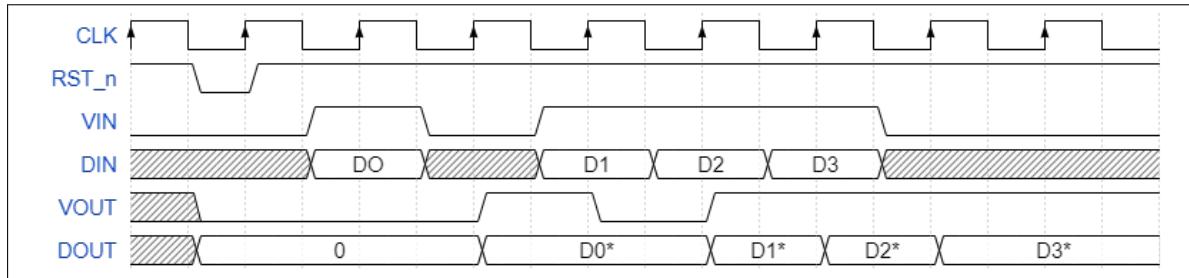


Figure 2.3: Timing diagram of the filter

2.1.2 VHDL description file

The architecture described in VHDL language is reported in the appendix 4.2. The behavioural architecture of the entity works internally with the signed representation of numbers; the precision of multiplication products is truncated at 7 bits and, to obtain the result value in 8 bits-form, a '0' bit is concatenated as least significant bit. Registers are instantiated by the compiler using the construct *process* and they are triggered by V_{IN} .

The filter is now simulated using the testbench and the results are compared to the C-model ones.

2.1.3 Simulation Pre-Synthesis

$CLK = 100\text{ ns}$, $tco = 10\text{ ns}$

To process all the samples the simulation lasts 39100 ns, from 550 ns to 39650 ns (as it can be seen in figure 2.4). A snapshot from 1500 ns to 4000 ns is shown in Figure 2.5 to highlight the behaviour of the filter when V_{IN} moves from 0 to 1 and viceversa. The results of the simulation are coherent with the results of the C-model (appendix 4.3).

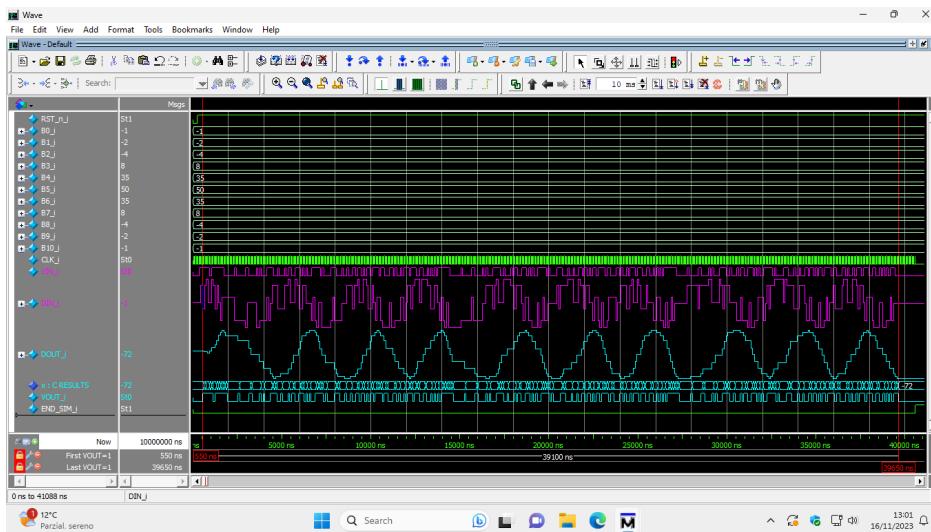


Figure 2.4: Complete simulation

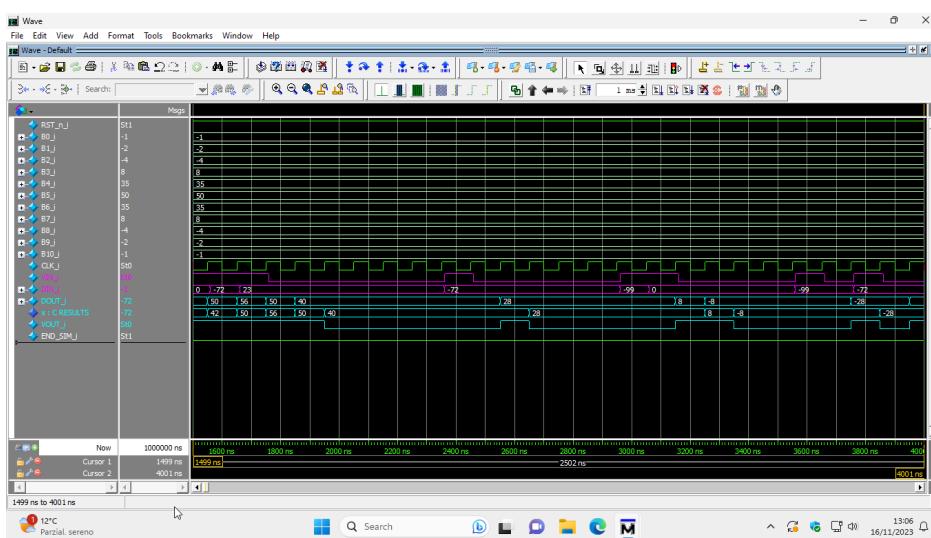


Figure 2.5: Filter behavior

2.2 Logic synthesis

Since the CAD **Synopsys Design Vision** performs always a timing optimization of the hardware described, in the case here presented the chain of adders shown in figure 2.2 is replaced by a 5 level-tree organization of the sums. The architecture, considering this optimization, is depicted in Figure 2.6. This improvement reduces significantly the critical path and it will be taken into account during the timing analysis.

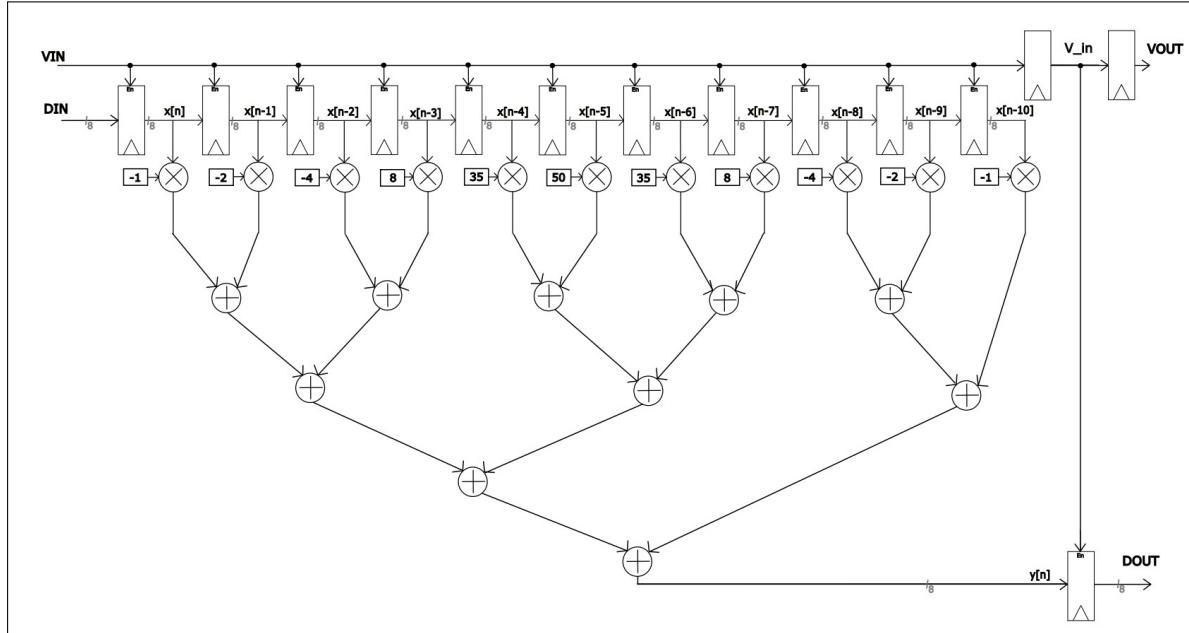


Figure 2.6: Actual filter synthesised by Design Vision, tree of adders

The maximum clock frequency this design can achieve corresponds to the minimum clock period with slack met and equal to zero. According to Design Vision, the minimum period is $T_c = 2.20$ ns and this leads to a $F_M = 454.5$ MHz.

2.2.1 Reports without clock gating

Timing, area and power reports are here shown.

REPORT AREA

```
Report : area
Design : myfir
Version: S-2021.06-SP4
Date   : Mon Oct 23 17:16:40 2023
```

Librarys Used:

```
NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm.db
```

Number of ports:	690
Number of nets:	3070
Number of cells:	2229

```

Number of combinational cells:          2005
Number of sequential cells:           185
Number of macros/black boxes:          0
Number of buf/inv:                  376
Number of references:                32

Combinational area:                 3212.216011
Buf/Inv area:                      215.460001
Noncombinational area:              992.712032
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined Wire load has zero net area

Total cell area:                   4204.928043
Total area:                         undefined

```

REPORT TIMING

```

Report : timing
  -path full
  -delay max
  -max_paths 1
Design : myfir
Version: S-2021.06-SP4
Date   : Mon Oct 23 17:16:40 2023

Operating Conditions: typical    Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Startpoint: x3_reg[1] rising edge-triggered flip-flop clocked by MY_CLK
Endpoint: DOUT_reg[7]
          rising edge-triggered flip-flop clocked by MY_CLK
Path Group: MY_CLK
Path Type: max

Des/Clust/Port      Wire Load Model      Library
-----
myfir             5K_hvratio_1_1      NangateOpenCellLibrary

Point                           Incr      Path
-----
clock MY_CLK rise edge          0.00      0.00
clock network delay ideal       0.00      0.00
x3_reg[1]/CK DFFR_X1           0.00      0.00 r
x3_reg[1]/QN DFFR_X1           0.06      0.06 f
U192/ZN INV_X1                 0.07      0.13 r
mult_100_8/a[1] myfir_DW_mult_tc_1 0.00      0.13 r
mult_100_8/U303/Z XOR2_X1        0.09      0.22 r
mult_100_8/U172/ZN OR2_X2        0.06      0.28 r
mult_100_8/U246/ZN OAI22_X1        0.05      0.33 f
mult_100_8/U37/S HA_X1           0.08      0.40 f
mult_100_8/U181/ZN INV_X1          0.03      0.44 r
mult_100_8/U195/ZN OAI222_X1        0.05      0.49 f
mult_100_8/U176/ZN INV_X1          0.03      0.52 r
mult_100_8/U177/ZN OAI222_X1        0.05      0.57 f
mult_100_8/U274/ZN AOI222_X1         0.10      0.67 r
mult_100_8/U235/ZN INV_X1           0.03      0.70 f
mult_100_8/U273/ZN AOI222_X1         0.09      0.79 r

```

mult_100_8/U236/ZN INV_X1	0.03	0.82 f
mult_100_8/U272/ZN AOI222_X1	0.09	0.91 r
mult_100_8/U237/ZN INV_X1	0.04	0.95 f
mult_100_8/U213/ZN NAND2_X1	0.04	0.99 r
mult_100_8/U215/ZN NAND3_X1	0.04	1.03 f
mult_100_8/U219/ZN NAND2_X1	0.04	1.07 r
mult_100_8/U221/ZN NAND3_X1	0.04	1.10 f
mult_100_8/U207/ZN NAND2_X1	0.04	1.14 r
mult_100_8/U202/ZN NAND3_X1	0.04	1.18 f
mult_100_8/U226/ZN NAND2_X1	0.04	1.22 r
mult_100_8/U228/ZN NAND3_X1	0.04	1.26 f
mult_100_8/U233/ZN NAND2_X1	0.03	1.29 r
mult_100_8/U234/ZN NAND3_X1	0.05	1.34 f
mult_100_8/U167/ZN NAND2_X1	0.04	1.37 r
mult_100_8/U169/ZN NAND3_X1	0.04	1.41 f
mult_100_8/U239/Z XOR2_X1	0.07	1.48 f
mult_100_8/U238/ZN XNOR2_X1	0.06	1.54 f
mult_100_8/product[14] myfir_DW_mult_tc_1	0.00	1.54 f
add_8_root_add_0_root_add_100_10/A[6] myfir_DW01_add_5	0.00	1.54 f
add_8_root_add_0_root_add_100_10/U1_6/S FA_X1	0.15	1.69 r
add_8_root_add_0_root_add_100_10/SUM[6] myfir_DW01_add_5	0.00	1.69 r
add_2_root_add_0_root_add_100_10/B[6] myfir_DW01_add_4	0.00	1.69 r
add_2_root_add_0_root_add_100_10/U1_6/S FA_X1	0.12	1.81 f
add_2_root_add_0_root_add_100_10/SUM[6] myfir_DW01_add_4	0.00	1.81 f
add_1_root_add_0_root_add_100_10/B[6] myfir_DW01_add_1	0.00	1.81 f
add_1_root_add_0_root_add_100_10/U1_6/S FA_X1	0.15	1.96 r
add_1_root_add_0_root_add_100_10/SUM[6] myfir_DW01_add_1	0.00	1.96 r
add_0_root_add_0_root_add_100_10/B[6] myfir_DW01_add_0	0.00	1.96 r
add_0_root_add_0_root_add_100_10/U1_6/S FA_X1	0.12	2.08 f
add_0_root_add_0_root_add_100_10/SUM[6] myfir_DW01_add_0	0.00	2.08 f
DOUT_reg[7]/D DFFR_X1	0.01	2.09 f
data arrival time		2.09

clock MY_CLK rise edge	2.20	2.20
clock network delay ideal	0.00	2.20
clock uncertainty	-0.07	2.13
DOUT_reg[7]/CK DFFR_X1	0.00	2.13 r
library setup time	-0.04	2.09
data required time		2.09

data required time		2.09
data arrival time		-2.09

slack MET	0.00	

[REPORT POWER]

Information: Updating design information... UID-85
 Information: Propagating switching activity low effort zero delay simulation. PWR-6
 Warning: The derived toggle rate value 0.909091 for the clock net 'CLK' conflicts with the annotated value 0.998

Report : power
 -analysis_effort low
 Design : myfir
 Version: S-2021.06-SP4
 Date : Thu Oct 26 12:26:47 2023

Librarys Used:

NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm_nowlm.db

Operating Conditions: typical Library: NangateOpenCellLibrary
 Wire Load Model Mode: top

Design	Wire Load Model	Library
myfir	5K_hvratio_1_1	NangateOpenCellLibrary

Global Operating Voltage = 1.1
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000ff
 Time Units = 1ns
 Dynamic Power Units = 1uW derived from V,C,T units
 Leakage Power Units = 1nW

Cell Internal Power = 1.6276 mW 58%
 Net Switching Power = 1.1630 mW 42%

 Total Dynamic Power = 2.7906 mW 100%

Cell Leakage Power = 86.2046 uW

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	0.00%	
memory	0.0000	0.0000	0.0000	0.0000	0.00%	
black_box	0.0000	0.0000	0.0000	0.0000	0.00%	
clock_network	0.0000	0.0000	0.0000	0.0000	0.00%	
register	638.7037	133.2546	1.6847e+04	788.8048	27.42%	
sequential	0.0000	0.0000	0.0000	0.0000	0.00%	
combinational	988.8500	1.0298e+03	6.9358e+04	2.0880e+03	72.58%	
Total	1.6276e+03 uW	1.1630e+03 uW	8.6205e+04 nW	2.8768e+03 uW		

2.2.2 Reports with clock gating

Clock gating is applied to this architecture (using the command **compile -gate_clock**) and reports are shown below:

REPORT AREA (CG)

```
Report : area
Design : myfir
Version: S-2021.06-SP4
Date   : Thu Oct 26 12:43:46 2023
```

Librarys Used:

```
NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm.db
```

Number of ports:	694
Number of nets:	2822
Number of cells:	1996
Number of combinational cells:	1776
Number of sequential cells:	186
Number of macros/black boxes:	0
Number of buf/inv:	265
Number of references:	27
Combinational area:	3016.972009
Buf/Inv area:	152.418000
Noncombinational area:	995.106033
Macro/Black Box area:	0.000000
Net Interconnect area:	undefined Wire load has zero net area
Total cell area:	4012.078042
Total area:	undefined

REPORT TIMING (CG)

Information: Updating design information... UID-85

```
Report : timing
  -path full
  -delay max
  -max_paths 1
Design : myfir
Version: S-2021.06-SP4
Date   : Thu Oct 26 12:43:46 2023
```

Operating Conditions: typical Library: NangateOpenCellLibrary
Wire Load Model Mode: top

```
Startpoint: x1_reg[1] rising edge-triggered flip-flop clocked by MY_CLK
Endpoint: DOUT_reg[7]
          rising edge-triggered flip-flop clocked by MY_CLK
Path Group: MY_CLK
Path Type: max
```

Des/Clust/Port	Wire Load Model	Library
myfir	5K_hvratio_1_1	NangateOpenCellLibrary
Point	Incr	Path
clock MY_CLK rise edge	0.00	0.00
clock network delay ideal	0.00	0.00
x1_reg[1]/CK DFFR_X1	0.00	0.00 r
x1_reg[1]/Q DFFR_X1	0.16	0.16 r
mult_100_10/a[1] myfir_DW_mult_tc_1	0.00	0.16 r
mult_100_10/U171/ZN XNOR2_X2	0.12	0.28 r
mult_100_10/U169/ZN NAND2_X1	0.06	0.34 f
mult_100_10/U235/ZN OAI22_X1	0.06	0.40 r
mult_100_10/U36/S FA_X1	0.12	0.52 f
mult_100_10/U222/ZN INV_X1	0.03	0.55 r
mult_100_10/U198/ZN OR2_X1	0.04	0.59 r
mult_100_10/U199/ZN NAND3_X1	0.04	0.63 f
mult_100_10/U259/ZN AOI222_X1	0.09	0.71 r
mult_100_10/U223/ZN INV_X1	0.03	0.74 f
mult_100_10/U189/ZN NAND2_X1	0.03	0.77 r
mult_100_10/U191/ZN AND3_X1	0.05	0.83 r
mult_100_10/U185/ZN OR2_X1	0.03	0.86 r
mult_100_10/U187/ZN NAND3_X1	0.04	0.89 f
mult_100_10/U8/CO FA_X1	0.09	0.98 f
mult_100_10/U7/CO FA_X1	0.10	1.08 f
mult_100_10/U175/ZN NAND2_X1	0.05	1.13 r
mult_100_10/U176/ZN NAND3_X1	0.04	1.17 f
mult_100_10/U181/ZN NAND2_X1	0.03	1.20 r
mult_100_10/U182/ZN NAND3_X1	0.04	1.24 f
mult_100_10/U4/S FA_X1	0.14	1.37 r
mult_100_10/product[12] myfir_DW_mult_tc_1	0.00	1.37 r
add_9_root_add_0_root_add_100_10/B[4] myfir_DW01_add_5	0.00	1.37 r
add_9_root_add_0_root_add_100_10/U1_4/S FA_X1	0.12	1.49 f
add_9_root_add_0_root_add_100_10/SUM[4] myfir_DW01_add_5	0.00	1.49 f
add_2_root_add_0_root_add_100_10/B[4] myfir_DW01_add_4	0.00	1.49 f
add_2_root_add_0_root_add_100_10/U1_4/CO FA_X1	0.10	1.60 f
add_2_root_add_0_root_add_100_10/U1_5/S FA_X1	0.14	1.74 r
add_2_root_add_0_root_add_100_10/SUM[5] myfir_DW01_add_4	0.00	1.74 r
add_1_root_add_0_root_add_100_10/B[5] myfir_DW01_add_1	0.00	1.74 r
add_1_root_add_0_root_add_100_10/U1_5/S FA_X1	0.12	1.85 f
add_1_root_add_0_root_add_100_10/SUM[5] myfir_DW01_add_1	0.00	1.85 f
add_0_root_add_0_root_add_100_10/B[5] myfir_DW01_add_0	0.00	1.85 f
add_0_root_add_0_root_add_100_10/U1_5/CO FA_X1	0.10	1.96 f
add_0_root_add_0_root_add_100_10/U1_6/S FA_X1	0.13	2.09 r
add_0_root_add_0_root_add_100_10/SUM[6] myfir_DW01_add_0	0.00	2.09 r
DOUT_reg[7]/D DFFR_X1	0.01	2.10 r
data arrival time		2.10
clock MY_CLK rise edge	2.20	2.20
clock network delay ideal	0.00	2.20
clock uncertainty	-0.07	2.13
DOUT_reg[7]/CK DFFR_X1	0.00	2.13 r

library setup time	-0.03	2.10
data required time		2.10

data required time		2.10
data arrival time		-2.10

slack MET		0.00

REPORT POWER (CG)

Information: Updating design information... UID-85

Information: Propagating switching activity low effort zero delay simulation. PWR-6

Warning: The derived toggle rate value 0.909091 for the clock net 'CLK' conflicts with the annotated value 0.998

Report : power
 -analysiss_effort low
 Design : myfir
 Version: S-2021.06-SP4
 Date : Thu Oct 26 12:55:00 2023

Librarys Used:

NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm_nowlm.db

Operating Conditions: typical Library: NangateOpenCellLibrary
 Wire Load Model Mode: top

Design	Wire Load Model	Library
myfir	5K_hvratio_1_1	NangateOpenCellLibrary

Global Operating Voltage = 1.1
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000ff
 Time Units = 1ns
 Dynamic Power Units = 1uW derived from V,C,T units
 Leakage Power Units = 1nW

Cell Internal Power = 1.2000 mW 60%
 Net Switching Power = 800.3920 uW 40%

Total Dynamic Power = 2.0004 mW 100%

Cell Leakage Power = 82.0502 uW

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	0.00%	
memory	0.0000	0.0000	0.0000	0.0000	0.00%	

black_box	0.0000	0.0000	0.0000	0.0000	0.00%
clock_network	5.8981	47.7682	58.4165	53.7247	2.58%
register	566.1455	122.4126	1.6950e+04	705.5078	33.88%
sequential	0.0000	0.0000	0.0000	0.0000	0.00%
combinational	627.9934	630.2114	6.5042e+04	1.3232e+03	63.54%
<hr/>					
Total	1.2000e+03	uW	800.3922	uW	8.2050e+04
				nW	2.0825e+03
				uW	

Results of the synthesis are summarized in Table 2.1

Architecture	Maximum frequency [MHz]	Area [μm^2]	Power [mW]	Simulation time [ns]
Without Clock Gating	454.5	4204.9	2.87	782
With Clock Gating	454.5	4012.1	2.08	782

Table 2.1: Frequency, area, power consumption and simulation time after synthesis

2.2.3 Simulation with clock gating

CLK = 2.2 ns , tco = 220 ps

To process all the samples the simulation lasts 782 ns, from 9 ns to 791 ns (as it can be seen in figure 2.7).

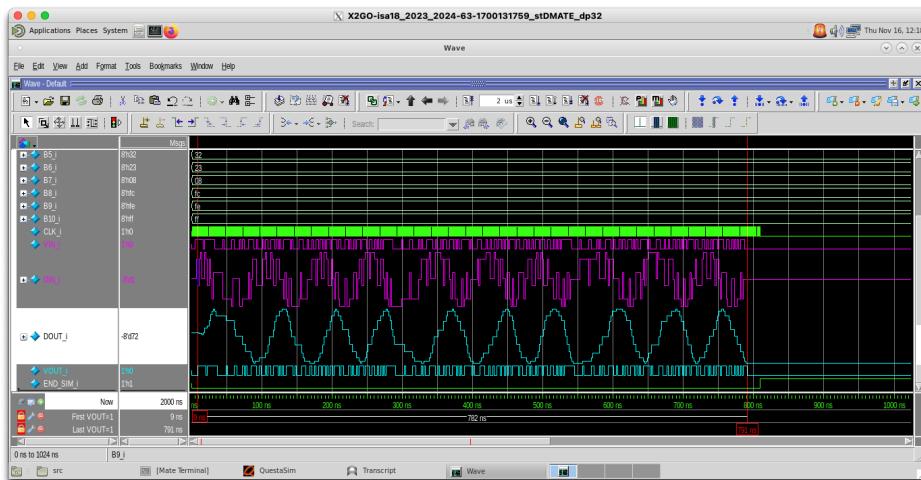


Figure 2.7: Complete simulation

2.3 Place and route

Using **Cadence Innovus** place and route is performed (Figure 2.8).

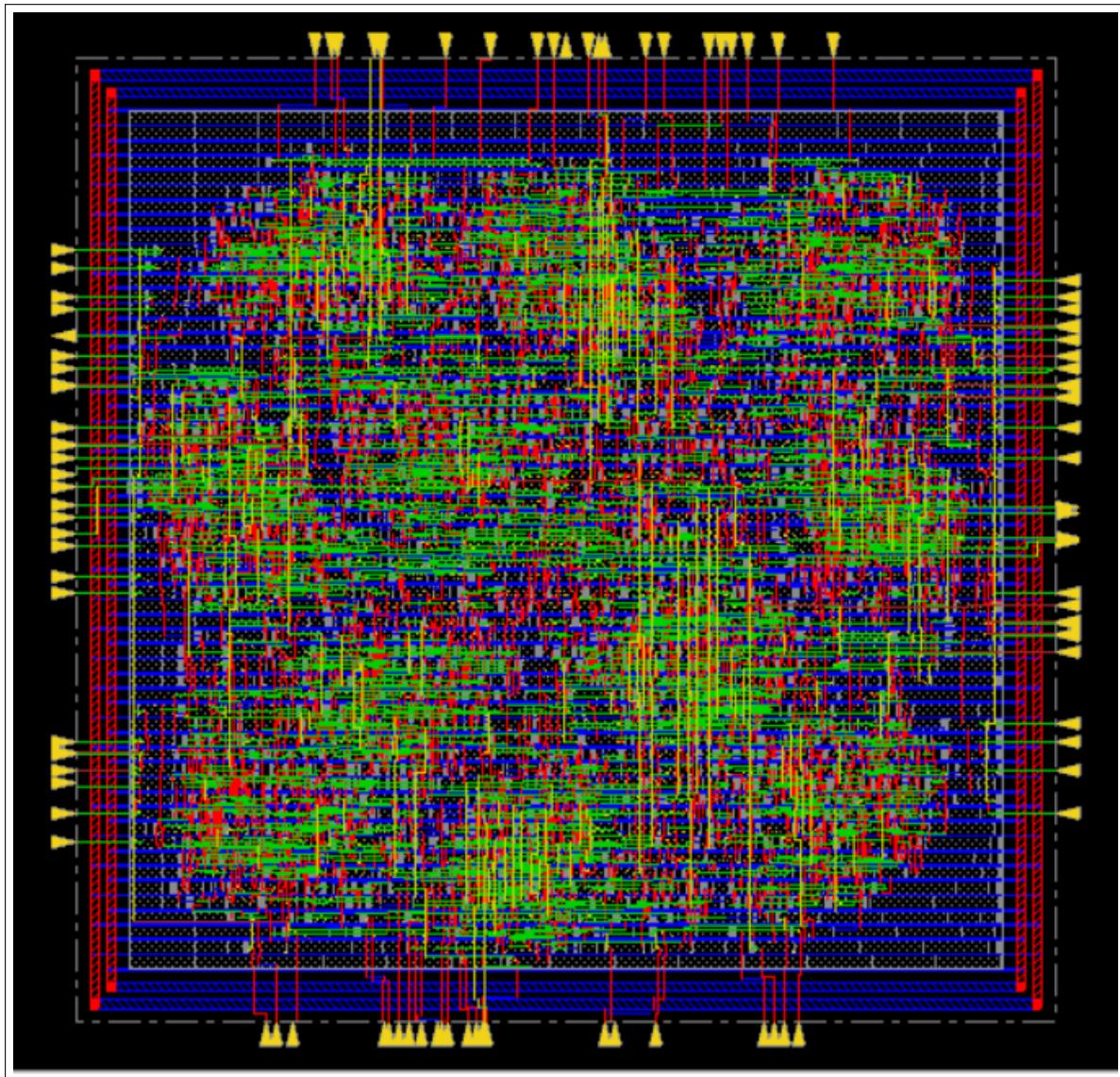


Figure 2.8: Place and Route result

Snapshots showing no timing violation (timeDesign Summary table for both setup and hold modes) and power consumption are shown in the following reports:

Gate Count
Gate area 0.7980 um ²
Level 0 Module myfir
Gates=4944 Cells=1923 Area=3945.8 um ²

```

timeDesign Summary

Setup views included:
MyAnView

+-----+-----+-----+-----+
| Setup mode | all | reg2reg | reg2cgate| default |
+-----+-----+-----+-----+
| WNS (ns):| 2.628 | 2.628 | N/A | 3.619 |
| TNS (ns):| 0.000 | 0.000 | N/A | 0.000 |
| Violating Paths:| 0 | 0 | N/A | 0 |
| All Paths:| 379 | 88 | N/A | 291 |
+-----+-----+-----+-----+

```

Figure 2.9: No timing violation for setup mode

Power Report						
Total Power						
Total Internal Power:	0.61375980		55.2371%			
Total Switching Power:	0.41785297		37.6059%			
Total Leakage Power:	0.07952472		7.1571%			
Total Power:	1.11113750					
Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %	
Sequential	0.2916	0.06242	0.01641	0.3704	33.34	
Macro	0	0	0	0	0	
IO	0	0	0	0	0	
Combinational	0.3089	0.2954	0.06284	0.6672	60.05	
Clock Combinational	0.005315	0.04237	6.113e-05	0.04775	4.297	
Clock Sequential	0.007931	0.01766	0.0002115	0.0258	2.322	
Total	0.6138	0.4179	0.07952	1.111	100	
Rail	Voltage	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %
VDD	1.1	0.6138	0.4179	0.07952	1.111	100
Clock	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %	

```
-----  
timeDesign Summary  
-----  
  
Hold views included:  
MyAnView  
  
+-----+-----+-----+-----+-----+  
| Hold mode | all | reg2reg | reg2cgate| default |  
+-----+-----+-----+-----+-----+  
| WNS (ns):| 0.090 | 0.090 | N/A | 0.000 |  
| TNS (ns):| 0.000 | 0.000 | N/A | 0.000 |  
| Violating Paths:| 0 | 0 | N/A | 0 |  
| All Paths:| 88 | 88 | N/A | 0 |  
+-----+-----+-----+-----+-----+
```

Figure 2.10: No timing violation for hold mode

```
MY_CLK          0.01325    0.06003   0.0002726   0.07354    6.619  
-----  
Total excluding duplicates 0.01325    0.06003   0.0002726   0.07354    6.619  
-----  
Clock: MY_CLK  
Clock Period: 0.004005 usec  
Clock Toggle Rate: 499.3820 Mhz  
Clock Static Probability: 0.2500
```

2.3.1 Simulation with clock gating after Place and Route

CLK = 4.4 ns , tco = 440 ps

To process all the samples the simulation lasts 1564 ns, from 18 ns to 1582 ns (as it can be seen in figure 2.11).

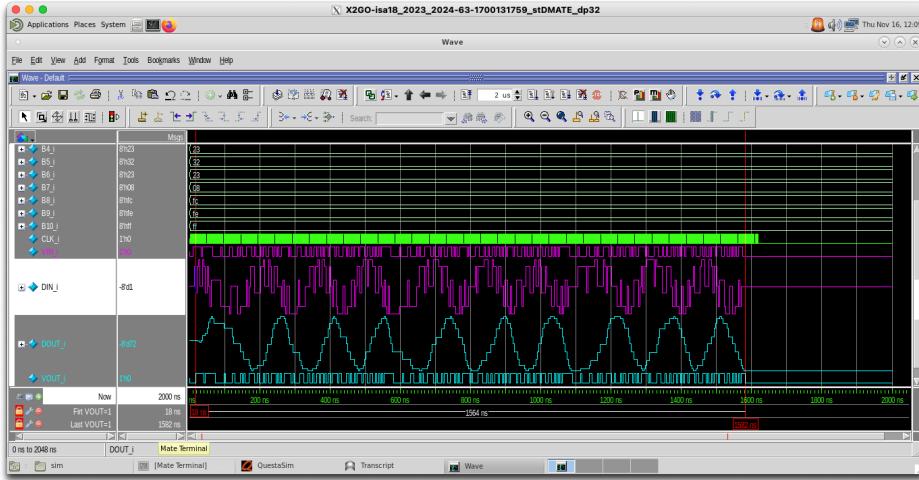


Figure 2.11: Complete simulation

Results of the place and route are shown in Table 2.2

Architecture	Maximum frequency [MHz]	Area [μm^2]	Power [mW]	Simulation time [ns]
After place&route	227.3	3945.8	1.11	1564

Table 2.2: Frequency, area, power consumption and simulation time after place and route

2.4 Explanations, comparisons and comments

As has already been noticed, Design Compiler synthesizes the filter using a tree structure for the additions instead of a chain structure. This implementation reduces the combinatorial logic on the critical path, which decreases from 1 multiplier and 10 adders in the chain structure to 1 multiplier and 4 adders in the tree structure. A proof of this statement is shown in the timing report: the critical path starts from register x3[1], goes through multiplier mult_100_8, adder_8, adder_2, adder_1, adder_0, and ends in register DOUT[7] (so 1 multiplier and 4 adders). From the area report and power report it can be seen that the combinatorial part of the filter occupies the largest portion of the total area (76.43% of total area) and consumes the largest amount of power (72.58% of total power consumption).

After applying the clock gating some changes can be appreciated:

- Area report: the number of sequential cell is increased by one unit, because the clock gating technique is applied using a negative edge-triggered latch. Moreover, the combinatorial area is slightly reduced, due to a reduction of buf/invert cells.

- Power report: there's a new contribution to the total power, related to the clock network consumption. Moreover, the combinatorial power is reduced and represents the 63% of the total. The total power consumption is reduced by the 28%.

At last, it was performed placing and routing using **Cadence Innovus** on the netlist derived from synthesis and obtained applying clock gating. The resulting area is slightly reduced (about a 2 % reduction), while the power consumption is about an half of the one reported before. This happens because this power estimation is done using a clock frequency $f_c = f_M/2$, while the latter was done at $f_c = f_M$.

CHAPTER 3

Advanced architecture development

3.1 Applying unfolding technique

In this chapter the unfolding technique is applied with the purpose of increasing performance of the filter; different numbers of pipeline stages are inserted and results are commented in terms of timing performance, area occupation and power consumption.

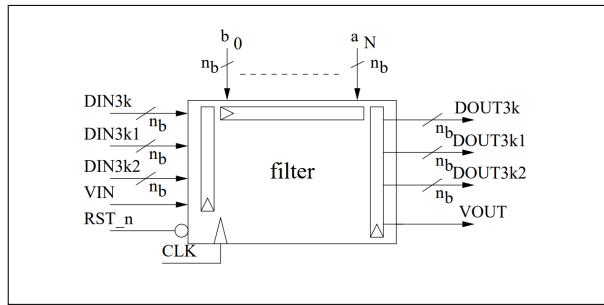


Figure 3.1: 3-Unfolded filter interface

According to the assignment, three-degree unfolding technique is applied to the filter architecture and the interface is shown in Figure 3.1 compared to SISO interface (Figure 2.1) three samples ($DIN3k$, $DIN3k1$ and $DIN3k2$) are processed and three output values ($DOUT3k$, $DOUT3k1$ and $DOUT3k2$) are available at each iteration.

The MIMO data flow graph is obtained applying the graphical method to the SISO DFG and the unfolded architecture is depicted in Figure 3.2.

Also in this case, **Design Vision** during synthesis performs an optimization replacing the chain of adders with a tree-organization; the optimized architecture is shown in Figure 3.3.

It can be seen that the tree-implementation does not allow to apply the pipeline technique, since there are no feedforward cut-sets. Any cut-set tracked in this DFG is not feedforward and for this reason the Design Vision-optimized implementation cannot be considered for pipeline application.

3.1.1 VHDL description file

The VHDL description of the unfolded architecture is reported in 4.2. As in the previous version, the precision of multiplication products is truncated at 7 bits and this leads to -32.29 dB of THD. It can be seen that the

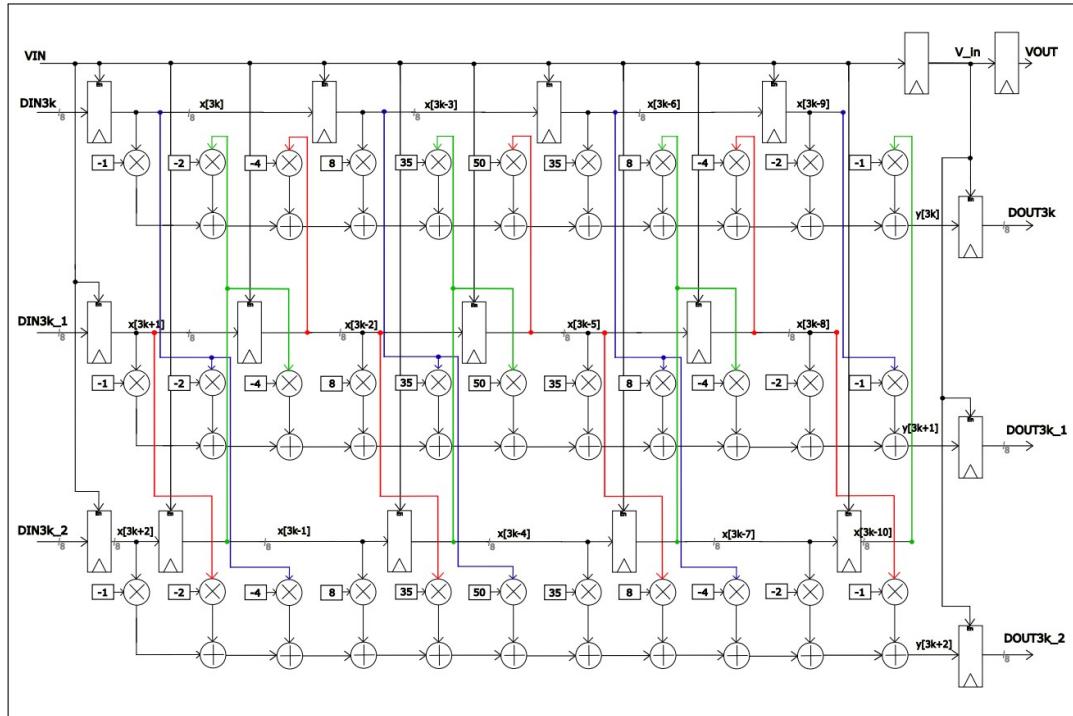


Figure 3.2: 3-Unfolded architecture without pipeline

number of registers for the samples (eleven) is the same as in the previous version; by contrast more adders and multipliers are instantiated according to the parallel execution of the computation.

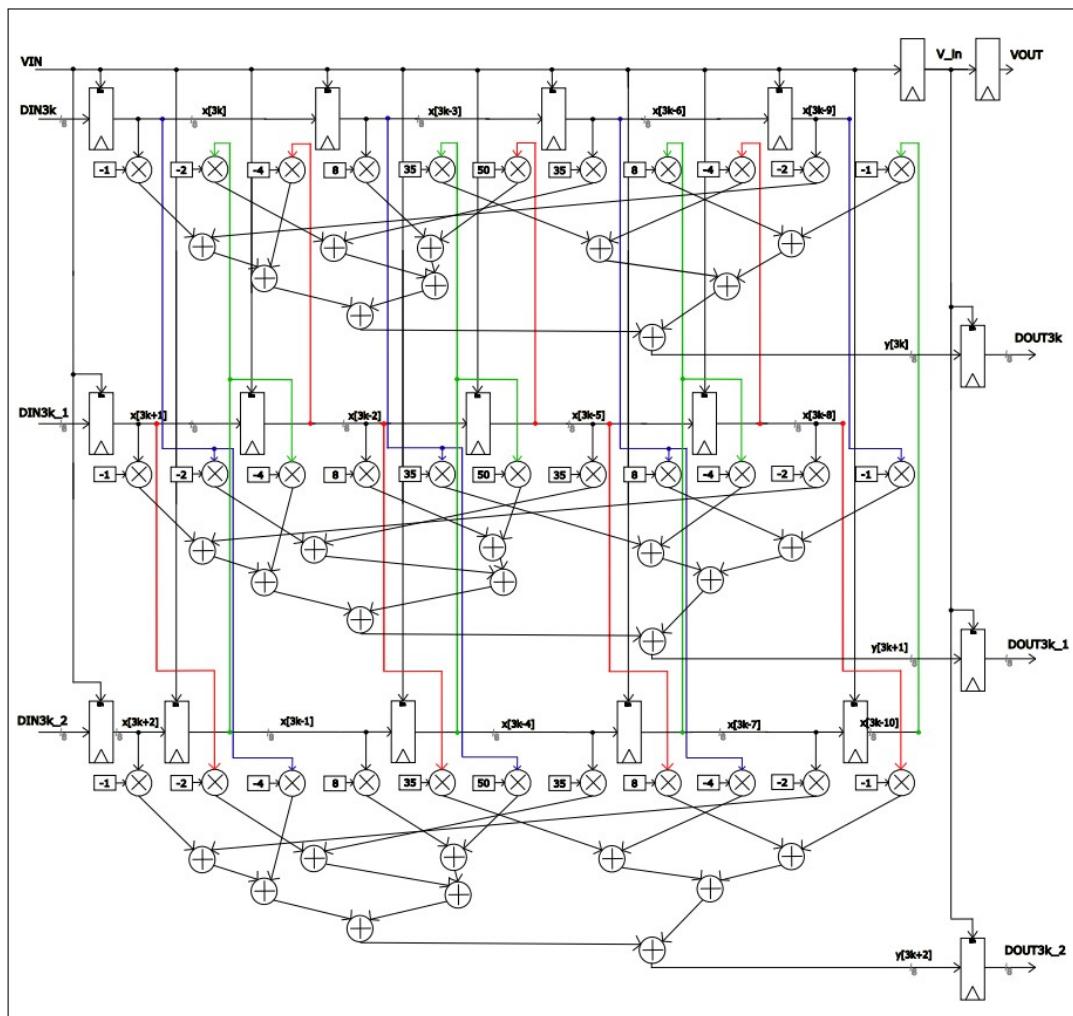


Figure 3.3: Unfolded optimized architecture synthesized by Design Vision

3.2 Simulation 3-unfolded architecture pre-synthesis

CLK = 100 ns , tco = 10ns

To process all the samples, the simulation lasts 12600 ns, from 550 ns to 13150 ns (as it can be seen in figure 3.4). A snapshot from 1500 ns to 4000 ns is shown in Figure 3.5 to highlight the behaviour of the filter when VIN moves from 0 to 1 and viceversa. The results of the simulation are coherent with the results of the C-model (appendix 4.3).

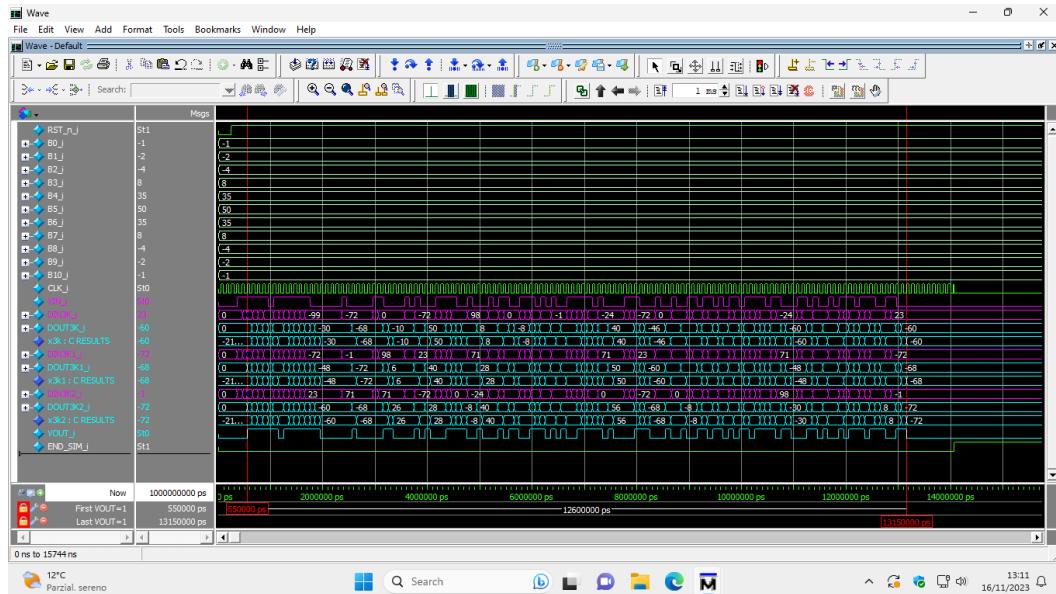


Figure 3.4: Complete simulation

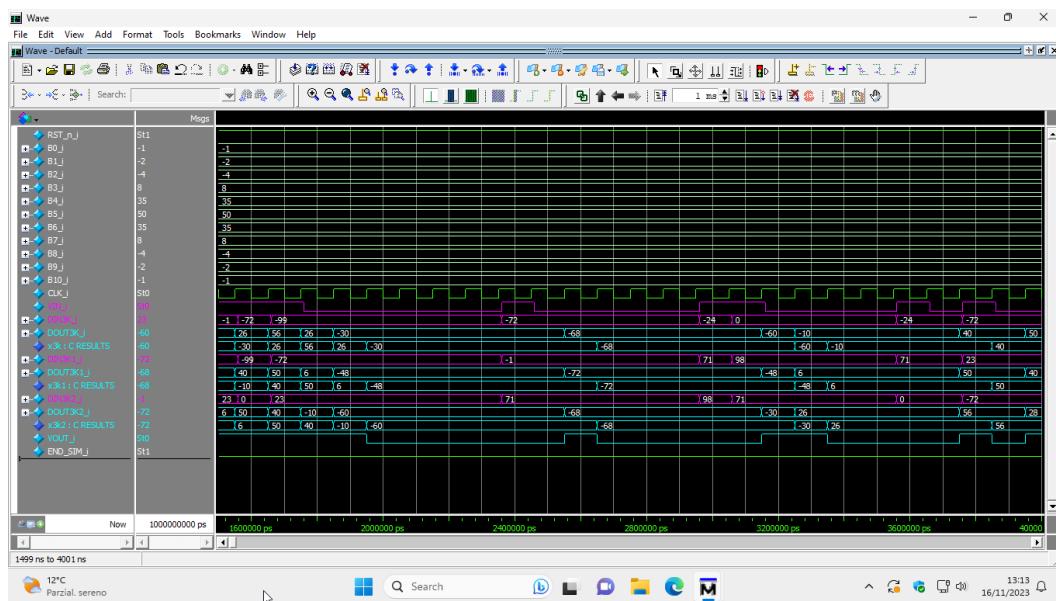


Figure 3.5: Filter behavior

3.3 Pipeline application

Different numbers of pipeline stages are now applied to the unfolded architecture in order to reach the highest throughput. Since the critical path in the basic architecture (Figure 2.6) is composed by one multiplier and four adders, pipeline structures with less than four stages are not taken into account.

3.3.1 4-stages pipeline

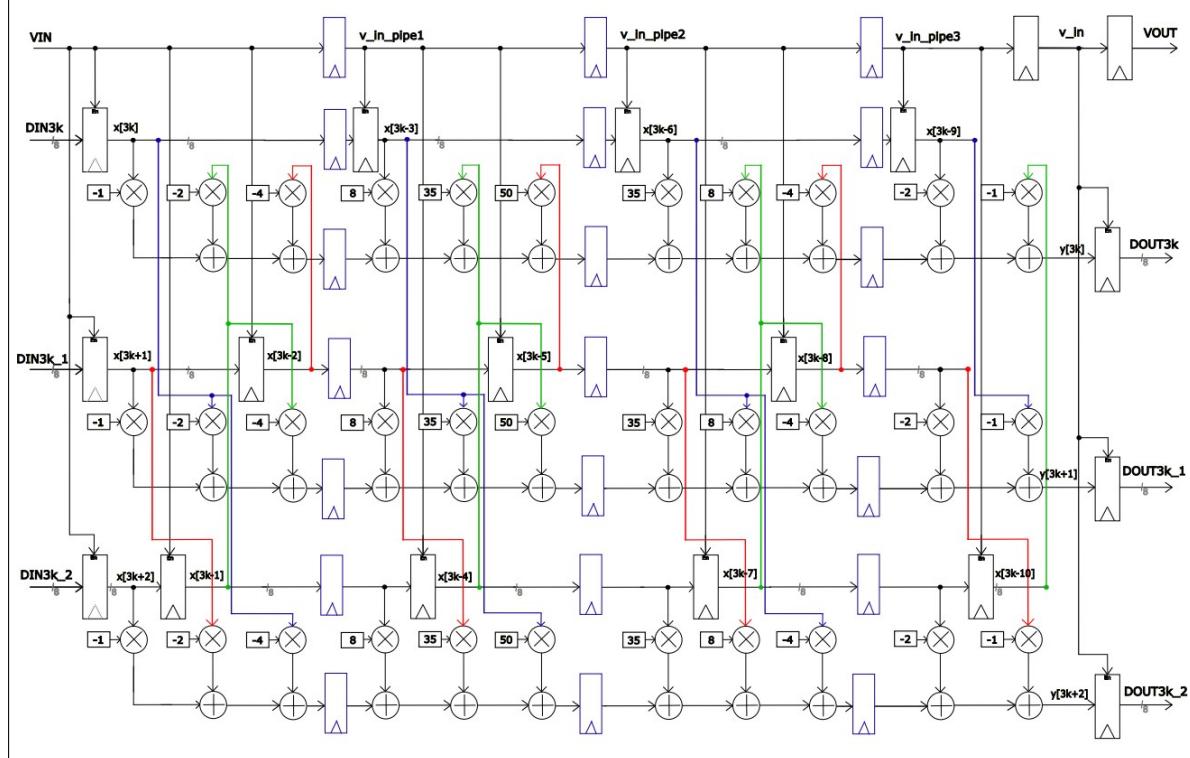


Figure 3.6: 3-Unfolded structure with 4 stages of pipeline

The Figure 3.6 shows the DFG divided in four pipeline stages, according to the rule of feedforward cut-set; each stage has at most three adders and the critical path is reduced to the delay one multiplier and three adders. Otherwise the latency is increased to five clock cycles as reported in the timing diagram in figure 3.7. In this architecture the number of pipeline registers is 21.

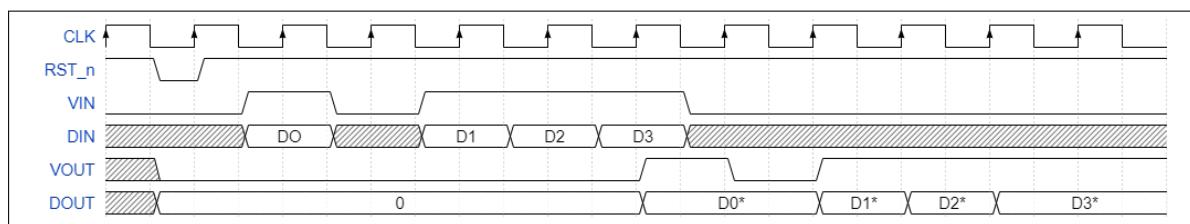


Figure 3.7: 4 stages pipeline timing diagram

3.3.2 5-stages pipeline

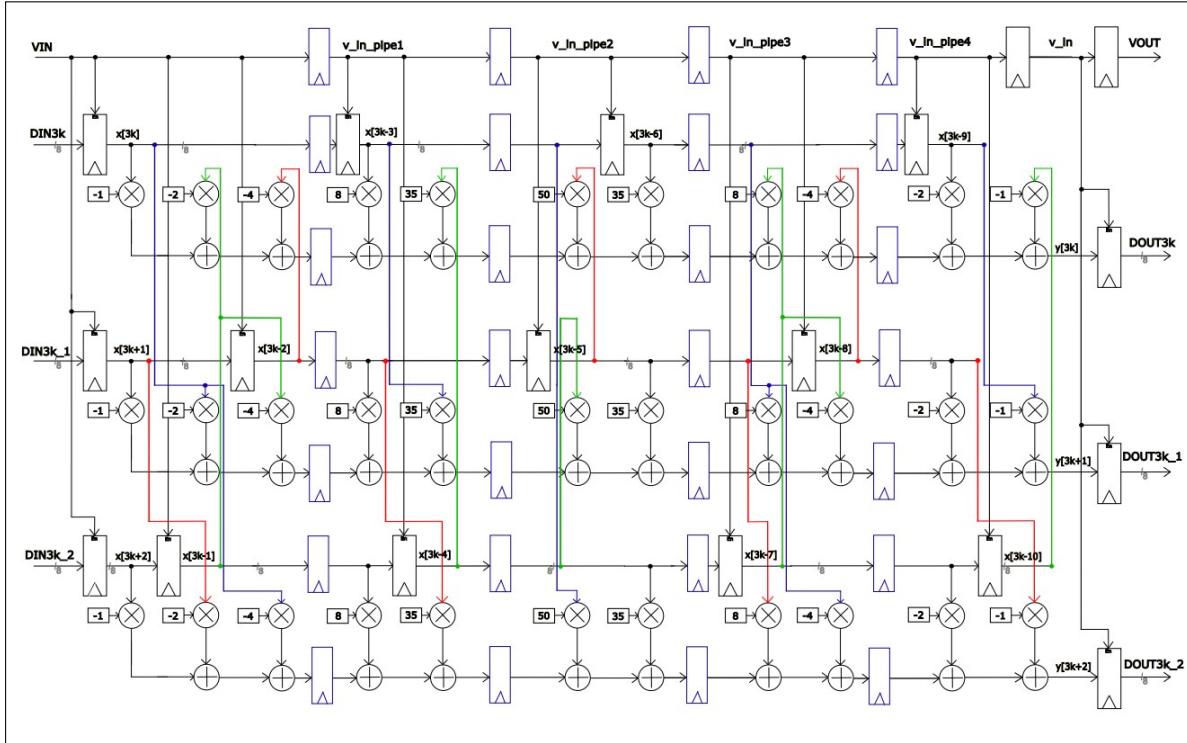


Figure 3.8: 3-Unfold structure with 5 stages of pipeline

The insertion of one more pipe stage allows to reduce the number of adders in the critical path from three to two and so allows to increase more the clock frequency. In figure 3.8 is represented the 5-pipe stages architecture. The latency in this implementation is six cycles as showed in the timing diagram in figure 3.9. pipe registers placed are 28.

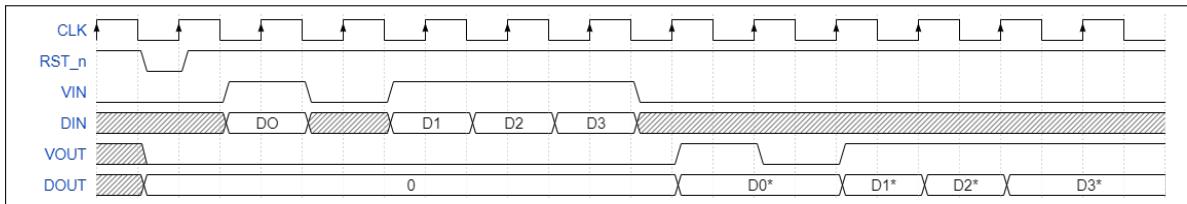


Figure 3.9: 5 stages pipeline timing diagram

3.3.3 10-stages pipeline

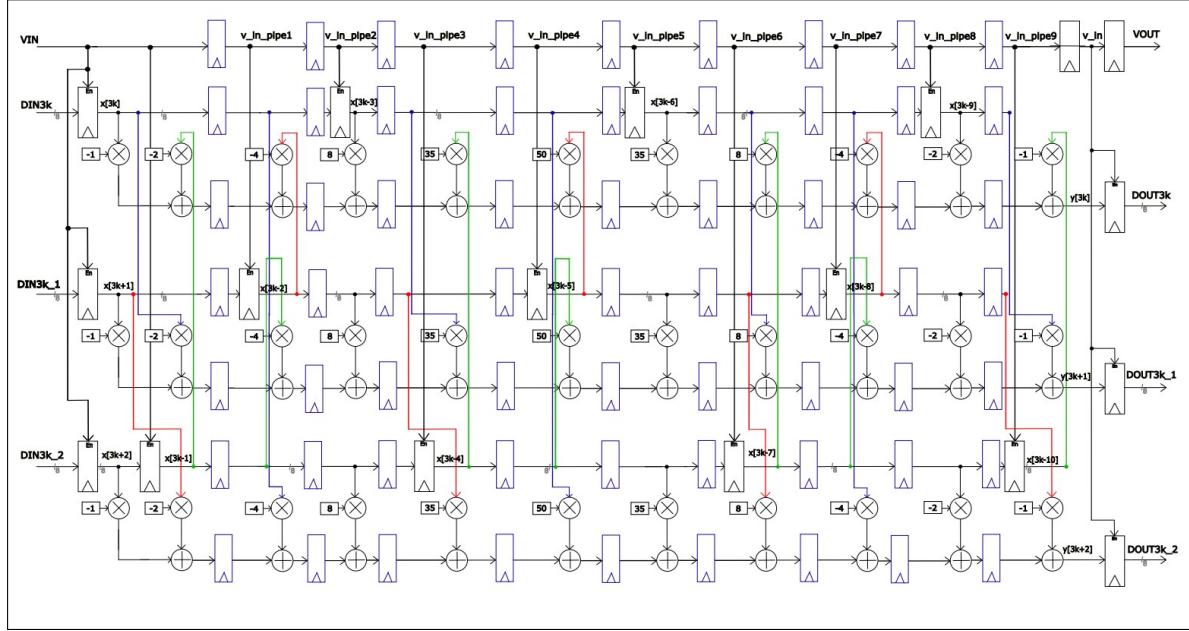


Figure 3.10: 3-Unfold structure with 10 stages of pipeline

Ten is the maximum number of pipeline stages for this filter and allows to reduce the critical path to one multiplier and one adder. In terms of throughput, this is the best implementation but the latency and the number of registers reach the highest values, respectively 11 and 63. The ten stages architecture is depicted in figure 3.10 and the timing diagram is reported in figure 3.11.

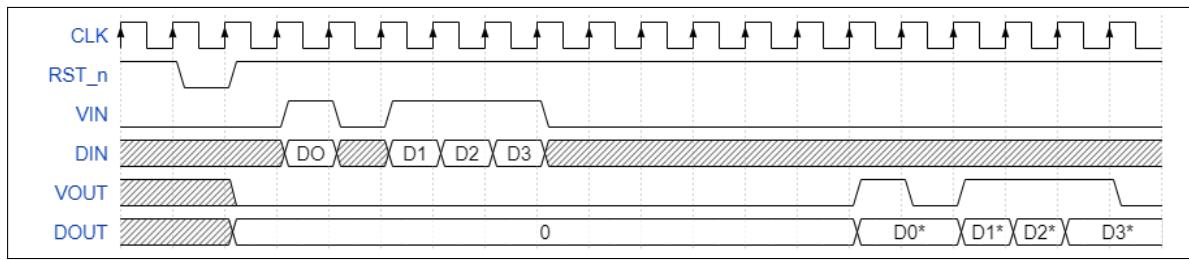


Figure 3.11: 10 stages of pipeline timing diagram

All these architectures have been simulated with **Modelsim** and the obtained results are equals to the ones obtained with the C model.

In order to compare the throughput of different structures, they are synthesized with Design Compiler. It was found the minimum clock period that leads to slack time equal to zero, which corresponds to the maximum operating frequency achievable. The results are shown in table 3.1.

Architecture	Minimum clock period [ns]	Area [μm^2]
UNFOLD + NO PIPE	2.20	10472
UNFOLD + 4 stages pipe	1.73	12658
UNFOLD + 5 stages pipe	1.70	12937
UNFOLD + 10 stages pipe	1.60	13868

Table 3.1: Performance and area comparison of different pipelined architectures

As expected, the implementation with 10 stages of pipeline works at the highest frequency ($f_M = 625$ MHz), so it has been chosen this one for the following analysis. It can be seen that higher the number of pipe stages, higher the area due to the insertion of more registers.

3.4 Simulation 3-unfolded with 10-pipeline stages pre-synthesis

CLK = 100 ns , tco = 10ns

To process all the samples, the simulation lasts 12600 ns, from 1450 ns to 14050 ns (as it can be seen in figure 3.12). A snapshot from 1500 ns to 4000 ns is shown in Figure 3.13 to highlight the behaviour of the filter when VIN moves from 0 to 1 and vice-versa. The results of the simulation are coherent with the results of the C-model (appendix 4.3).

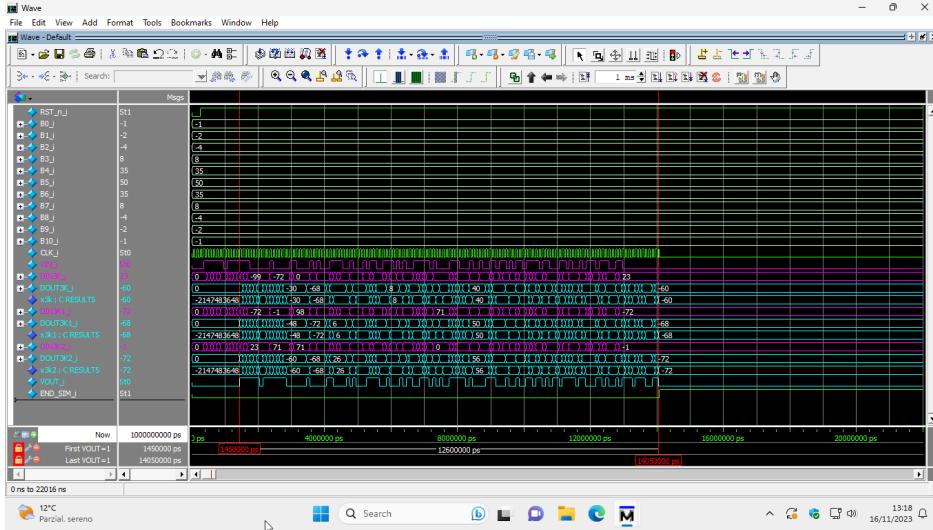


Figure 3.12: Complete simulation

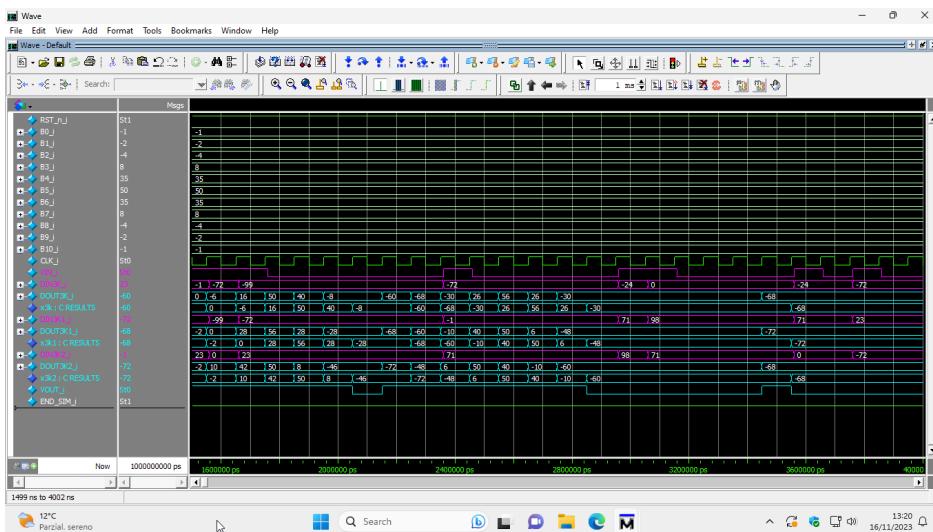


Figure 3.13: Filter behavior

3.5 Logic synthesis

The logic synthesis of this architecture is performed using Design Vision and snapshots showing slack met and power consumption are shown in following reports:

REPORT AREA

```
Report : area
Design : myfir_unfolded_pipe_lsom
Version: S-2021.06-SP4
Date   : Mon Nov  6 15:26:49 2023
```

Librarys Used:

```
NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm.db
```

Number of ports:	1886
Number of nets:	11645
Number of cells:	9417
Number of combinational cells:	8686
Number of sequential cells:	629
Number of macros/black boxes:	0
Number of buf/inv:	1827
Number of references:	75
Combinational area:	11514.873980
Buf/Inv area:	1171.995995
Noncombinational area:	3361.442109
Macro/Black Box area:	0.000000
Net Interconnect area:	undefined Wire load has zero net area
Total cell area:	14876.316089
Total area:	undefined
1	

REPORT TIMING

```
Report : timing
  -path full
  -delay max
  -max_paths 1
Design : myfir_unfolded_pipe_lsom
Version: S-2021.06-SP4
Date   : Mon Nov  6 15:26:49 2023
```

```
Operating Conditions: typical Library: NangateOpenCellLibrary
Wire Load Model Mode: top
```

```
Startpoint: x3k_10_reg[1]
            rising edge-triggered flip-flop clocked by MY_CLK
Endpoint: DOUT3k_reg[7]
            rising edge-triggered flip-flop clocked by MY_CLK
Path Group: MY_CLK
Path Type: max
```

Des/Clust/Port	Wire Load Model	Library
myfir_unfolded_pipe_1som	5K_hvratio_1_1	NangateOpenCellLibrary
Point	Incr	Path
clock MY_CLK rise edge	0.00	0.00
clock network delay ideal	0.00	0.00
x3k_10_reg[1]/CK SDFFR_X1	0.00	0.00 r
x3k_10_reg[1]/Q SDFFR_X1	0.10	0.10 r
mult_227/a[1] myfir_unfolded_pipe_1som_DW_mult_tc_2	0.00	0.10 r
mult_227/U223/ZN XNOR2_X1	0.08	0.18 r
mult_227/U197/ZN NAND2_X1	0.04	0.22 f
mult_227/U196/Z BUF_X1	0.05	0.26 f
mult_227/U278/ZN OAI22_X1	0.05	0.32 r
mult_227/U33/S FA_X1	0.12	0.44 f
mult_227/U32/S FA_X1	0.11	0.55 f
mult_227/U234/ZN NAND2_X1	0.03	0.58 r
mult_227/U235/ZN AND3_X1	0.05	0.64 r
mult_227/U229/ZN OR2_X1	0.03	0.67 r
mult_227/U231/ZN NAND3_X1	0.04	0.70 f
mult_227/U8/CO FA_X1	0.10	0.80 f
mult_227/U204/ZN NAND2_X1	0.05	0.85 r
mult_227/U206/ZN NAND3_X1	0.04	0.89 f
mult_227/U210/ZN NAND2_X1	0.04	0.92 r
mult_227/U212/ZN NAND3_X1	0.04	0.96 f
mult_227/U193/ZN NAND2_X1	0.03	0.99 r
mult_227/U195/ZN NAND3_X1	0.04	1.04 f
mult_227/U261/ZN NAND2_X1	0.03	1.07 r
mult_227/U264/ZN NAND3_X1	0.04	1.10 f
mult_227/U3/CO FA_X1	0.09	1.19 f
mult_227/U225/ZN XNOR2_X1	0.06	1.25 f
mult_227/U265/ZN XNOR2_X1	0.06	1.31 f
mult_227/product[14] myfir_unfolded_pipe_1som_DW_mult_tc_2	0.00	1.31 f
add_227/B[6] myfir_unfolded_pipe_1som_DW01_add_2	0.00	1.31 f
add_227/U1_6/S FA_X1	0.14	1.45 r
add_227/SUM[6] myfir_unfolded_pipe_1som_DW01_add_2	0.00	1.45 r
DOUT3k_reg[7]/D DFFR_X1	0.01	1.46 r
data arrival time		1.46
clock MY_CLK rise edge	1.56	1.56
clock network delay ideal	0.00	1.56
clock uncertainty	-0.07	1.49
DOUT3k_reg[7]/CK DFFR_X1	0.00	1.49 r
library setup time	-0.03	1.46
data required time		1.46
data required time		1.46
data arrival time		-1.46
slack MET	0.00	

REPORT POWER

Information: Updating design information... UID-85
 Information: Propagating switching activity low effort zero delay simulation. PWR-6
 Warning: The derived toggle rate value 1.250000 for the clock net 'CLK' conflicts with the annotated value 0.9964

Report : power
 -analysis_effort low
 Design : myfir_unfolded_pipe_1som
 Version: S-2021.06-SP4
 Date : Tue Nov 14 11:03:31 2023

Librarys Used:

NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm_nowlm.db

Operating Conditions: typical Library: NangateOpenCellLibrary
 Wire Load Model Mode: top

Design	Wire Load Model	Library
myfir_unfolded_pipe_1som	5K_hvratio_1_1	NangateOpenCellLibrary

Global Operating Voltage = 1.1
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000ff
 Time Units = 1ns
 Dynamic Power Units = 1uW derived from V,C,T units
 Leakage Power Units = 1nW

Cell Internal Power = 3.1189 mW 67%
 Net Switching Power = 1.5660 mW 33%

Total Dynamic Power = 4.6849 mW 100%

Cell Leakage Power = 240.9451 uW

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	0.00%	
memory	0.0000	0.0000	0.0000	0.0000	0.00%	
black_box	0.0000	0.0000	0.0000	0.0000	0.00%	
clock_network	0.0000	0.0000	0.0000	0.0000	0.00%	
register	1.8775e+03	309.5158	5.5527e+04	2.2425e+03	45.53%	
sequential	0.0000	0.0000	0.0000	0.0000	0.00%	
combinational	1.2414e+03	1.2565e+03	1.8542e+05	2.6833e+03	54.47%	
Total	3.1189e+03 uW	1.5660e+03 uW	2.4095e+05 nW	4.9258e+03 uW		
1						

Results of the synthesis without clock gating are shown in Table 3.2.

Architecture	Maximum frequency [MHz]	Area [μm^2]	Power [mW]	Simulation time [ns]
Without Clock Gating	625	15063	4.926	252

Table 3.2: Results of synthesis without clock gating: A is the area, P is the power consumption

3.5.1 Simulation 3-unfolded with 10-pipe without clock gating

CLK = 1.6 ns , tco = 160 ps

To process all the samples, the simulation lasts 252 ns, from 29 ns to 281 ns (as it can be seen in figure 3.14).

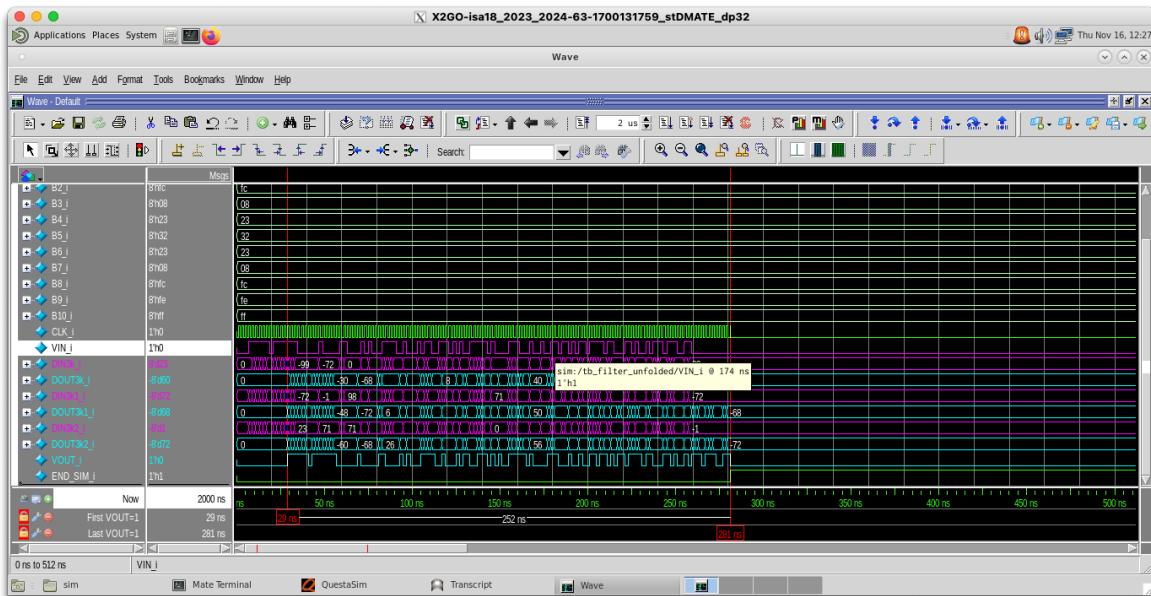


Figure 3.14: Complete simulation

3.6 Clock gating issue

After applying clock gating (using the command **compile -gate_clock**) we encountered some issues: although the report_timing shows no problem (slack MET), the results of the simulated netlist are not coherent with the ones obtained before (figure 3.15).

```
----- REPORT TIMING -----
Information: Updating design information... UID-85

Report : timing
  -path full
  -delay max
  -max_paths 1
Design : myfir_unfolded_pipe_1som
Version: S-2021.06-SP4
Date   : Sat Nov 11 17:02:01 2023

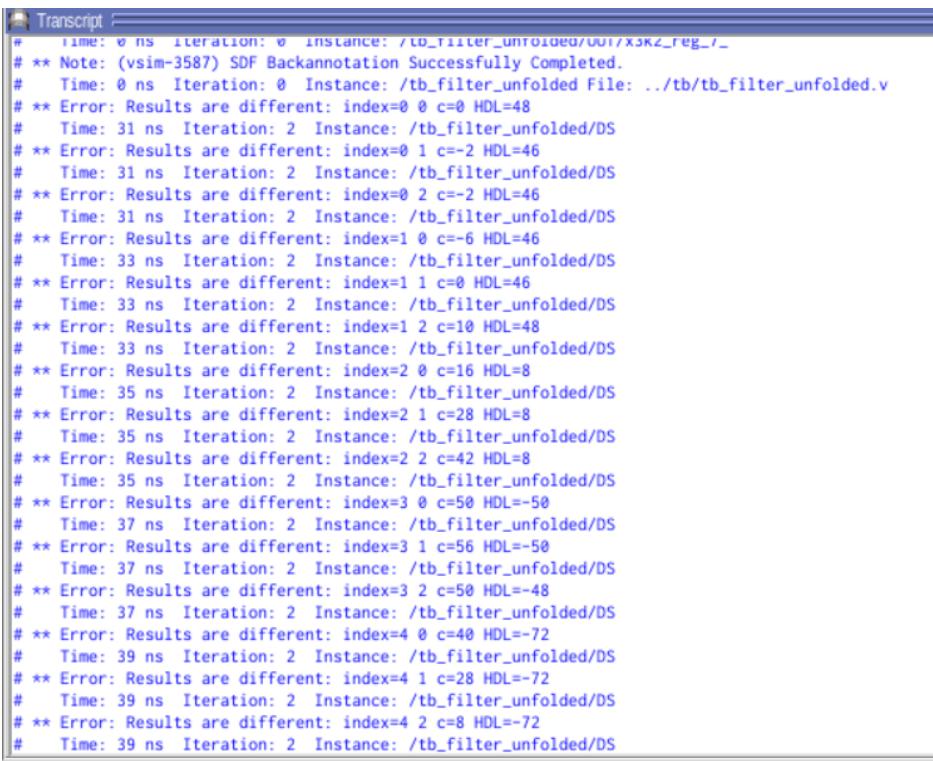
Operating Conditions: typical Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Startpoint: x3k_8_reg[5]
            rising edge-triggered flip-flop clocked by MY_CLK
Endpoint: tmp_pipe8_reg[6]
            rising edge-triggered flip-flop clocked by MY_CLK
Path Group: MY_CLK
Path Type: max

Des/Clust/Port      Wire Load Model      Library
-----
myfir_unfolded_pipe_1som
      5K_hvratio_1_1      NangateOpenCellLibrary

Point          Incr      Path
-----
clock MY_CLK rise edge          0.00      0.00
clock network delay ideal       0.00      0.00
x3k_8_reg[5]/CK DFFR_X1        0.00      0.00 r
x3k_8_reg[5]/QN DFFR_X1        0.07      0.07 r
U70/ZN INV_X2                  0.05      0.12 f
mult_225/a[5] myfir_unfolded_pipe_1som_DW_mult_tc_8
                           0.00      0.12 f
mult_225/U166/ZN INV_X1        0.04      0.16 r
mult_225/U164/ZN NAND2_X1      0.03      0.19 f
mult_225/U165/ZN NAND2_X1      0.03      0.22 r
mult_225/U213/ZN NAND2_X1      0.04      0.26 f
mult_225/U281/ZN OAI22_X1      0.06      0.32 r
mult_225/U35/S HA_X1           0.08      0.40 r
mult_225/U34/S FA_X1           0.11      0.52 f
mult_225/U260/ZN INV_X1        0.03      0.55 r
mult_225/U261/ZN OAI22_X1      0.05      0.60 f
mult_225/U232/ZN INV_X1        0.03      0.64 r
mult_225/U233/ZN OAI22_X1      0.05      0.69 f
mult_225/U239/ZN NAND2_X1      0.04      0.73 r
mult_225/U211/ZN NAND3_X1      0.04      0.77 f
mult_225/U214/ZN NAND2_X1      0.04      0.81 r
mult_225/U217/ZN NAND3_X1      0.04      0.85 f
mult_225/U222/ZN NAND2_X1      0.04      0.89 r
mult_225/U224/ZN NAND3_X1      0.04      0.93 f
mult_225/U227/ZN NAND2_X1      0.04      0.97 r
mult_225/U195/ZN NAND3_X1      0.04      1.01 f
```

mult_225/U199/ZN NAND2_X1	0.04	1.05 r
mult_225/U202/ZN NAND3_X1	0.04	1.09 f
mult_225/U191/ZN NAND2_X1	0.03	1.12 r
mult_225/U193/ZN NAND3_X1	0.05	1.16 f
mult_225/U206/ZN NAND2_X1	0.04	1.20 r
mult_225/U208/ZN NAND3_X1	0.04	1.23 f
mult_225/U231/ZN XNOR2_X1	0.06	1.29 f
mult_225/U262/ZN XNOR2_X1	0.06	1.35 f
mult_225/product[14] myfir_unfolded_pipe_1som_DW_mult_tc_8	0.00	1.35 f
add_225/B[6] myfir_unfolded_pipe_1som_DW01_add_8	0.00	1.35 f
add_225/U1_6/S FA_X1	0.14	1.49 r
add_225/SUM[6] myfir_unfolded_pipe_1som_DW01_add_8	0.00	1.49 r
tmp_pipe8_reg[6]/D DFFR_X1	0.01	1.50 r
data arrival time		1.50
clock MY_CLK rise edge	1.60	1.60
clock network delay ideal	0.00	1.60
clock uncertainty	-0.07	1.53
tmp_pipe8_reg[6]/CK DFFR_X1	0.00	1.53 r
library setup time	-0.03	1.50
data required time		1.50
<hr/>		
data required time		1.50
data arrival time		-1.50
<hr/>		
slack MET	0.00	



```
# Time: 0 ns Iteration: 0 Instance: /tb_filter_unfolded/u01/x3k2_reg_/
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
# Time: 0 ns Iteration: 0 Instance: /tb_filter_unfolded File: ../tb/tb_filter_unfolded.v
# ** Error: Results are different: index=0 0 c=0 HDL=48
# Time: 31 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=0 1 c=-2 HDL=46
# Time: 31 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=0 2 c=-2 HDL=46
# Time: 31 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=1 0 c=-6 HDL=46
# Time: 33 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=1 1 c=0 HDL=46
# Time: 33 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=1 2 c=10 HDL=48
# Time: 33 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=2 0 c=16 HDL=8
# Time: 35 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=2 1 c=28 HDL=8
# Time: 35 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=2 2 c=42 HDL=8
# Time: 35 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=3 0 c=56 HDL=-50
# Time: 37 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=3 1 c=56 HDL=-50
# Time: 37 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=3 2 c=50 HDL=-48
# Time: 37 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=4 0 c=40 HDL=-72
# Time: 39 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=4 1 c=28 HDL=-72
# Time: 39 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
# ** Error: Results are different: index=4 2 c=8 HDL=-72
# Time: 39 ns Iteration: 2 Instance: /tb_filter_unfolded/DS
```

Figure 3.15: After applying clock gating results of simulation of verilog netlist are wrong

Using the schematic view of Design Vision we ensured that the clock gating was being applied in the correct way, using the correct enable signals and driving the correct registers. We also checked that we were using the correct value of T_s in *clk_gen.vhd* and the correct T_{co} in *data_maker.vhd*. After countless tests we discovered that the simulation gives correct results only if the filter is synthesised with a clock period greater than 5.0 ns, without changing anything else in the vhd file.

Therefore we chose this value in order to correct the wrong behaviour. A possible explanation of this phenomenon is that the ports used for clock gating (in this case latches) add an extra delay to the CLK signal.

Using a clock period of $T = 5.0$ ns and applying clock gating we obtain this results:

REPORT AREA

```
Report : area
Design : myfir_unfolded_pipe_lsom
Version: S-2021.06-SP4
Date   : Tue Nov 14 14:09:12 2023
```

Librarys Used:

```
NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm.db
```

Number of ports:	1930
Number of nets:	7876
Number of cells:	5493
Number of combinational cells:	4745
Number of sequential cells:	640
Number of macros/black boxes:	0
Number of buf/inv:	743
Number of references:	76
Combinational area:	8654.310036
Buf/Inv area:	417.354002
Noncombinational area:	3390.170108
Macro/Black Box area:	0.000000
Net Interconnect area:	undefined Wire load has zero net area
Total cell area:	12044.480144
Total area:	undefined
1	

REPORT TIMING

```
Information: Updating design information... UID-85
```

```
Report : timing
-path full
-delay max
-max_paths 1
Design : myfir_unfolded_pipe_lsom
Version: S-2021.06-SP4
Date   : Tue Nov 14 14:09:12 2023
```

```
Operating Conditions: typical Library: NangateOpenCellLibrary
Wire Load Model Mode: top
```

Startpoint: x3k_reg[1] rising edge-triggered flip-flop clocked by MY_CLK
 Endpoint: tmp_pipe1_1_reg[6]
 rising edge-triggered flip-flop clocked by MY_CLK
 Path Group: MY_CLK
 Path Type: max

Des/Clust/Port	Wire Load Model	Library
myfir_unfolded_pipe_1som	5K_hvratio_1_1	NangateOpenCellLibrary
Point	Incr	Path
clock MY_CLK rise edge	0.00	0.00
clock network delay ideal	0.00	0.00
x3k_reg[1]/CK DFFR_X1	0.00	0.00 r
x3k_reg[1]/Q DFFR_X1	0.22	0.22 r
mult_235_2/a[1] myfir_unfolded_pipe_1som_DW_mult_tc_29	0.00	0.22 r
mult_235_2/U249/Z XOR2_X1	0.11	0.34 r
mult_235_2/U163/ZN INV_X1	0.06	0.39 f
mult_235_2/U247/ZN NAND2_X1	0.08	0.48 r
mult_235_2/U191/ZN OAI22_X1	0.06	0.53 f
mult_235_2/U37/S HA_X1	0.08	0.61 f
mult_235_2/U221/ZN AOI222_X1	0.11	0.72 r
mult_235_2/U161/ZN INV_X1	0.03	0.75 f
mult_235_2/U220/ZN AOI222_X1	0.09	0.84 r
mult_235_2/U160/ZN INV_X1	0.03	0.87 f
mult_235_2/U219/ZN AOI222_X1	0.09	0.96 r
mult_235_2/U158/ZN INV_X1	0.03	0.99 f
mult_235_2/U218/ZN AOI222_X1	0.09	1.09 r
mult_235_2/U157/ZN INV_X1	0.03	1.11 f
mult_235_2/U217/ZN AOI222_X1	0.09	1.21 r
mult_235_2/U159/ZN INV_X1	0.03	1.23 f
mult_235_2/U8/CO FA_X1	0.09	1.32 f
mult_235_2/U7/CO FA_X1	0.09	1.41 f
mult_235_2/U6/CO FA_X1	0.09	1.50 f
mult_235_2/U5/CO FA_X1	0.09	1.59 f
mult_235_2/U4/CO FA_X1	0.09	1.68 f
mult_235_2/U3/CO FA_X1	0.09	1.77 f
mult_235_2/U184/Z XOR2_X1	0.07	1.84 f
mult_235_2/U183/ZN XNOR2_X1	0.06	1.90 f
mult_235_2/product[14] myfir_unfolded_pipe_1som_DW_mult_tc_29	0.00	1.90 f
add_235/B[6] myfir_unfolded_pipe_1som_DW01_add_28	0.00	1.90 f
add_235/U1_6/S FA_X1	0.14	2.05 r
add_235/SUM[6] myfir_unfolded_pipe_1som_DW01_add_28	0.00	2.05 r
tmp_pipe1_1_reg[6]/D DFFR_X1	0.01	2.06 r
data arrival time		2.06
clock MY_CLK rise edge	5.00	5.00
clock network delay ideal	0.00	5.00
clock uncertainty	-0.07	4.93
tmp_pipe1_1_reg[6]/CK DFFR_X1	0.00	4.93 r
library setup time	-0.03	4.90
data required time		4.90
data required time		4.90
data arrival time		-2.06

slack MET	2.84
-----------	------

1

REPORT POWER

Information: Updating design information... UID-85
 Information: Propagating switching activity low effort zero delay simulation. PWR-6
 Warning: The derived toggle rate value 0.400000 for the clock net 'CLK' conflicts with the annotated value 0.500000

Report : power
 -analysis_effort low
 Design : myfir_unfolded_pipe_1som
 Version: S-2021.06-SP4
 Date : Tue Nov 14 14:13:23 2023

Librarys Used:

NangateOpenCellLibrary File: /eda/dk/nangate45/synopsys/NangateOpenCellLibrary_typical_ecsm_nowlm.db

Operating Conditions: typical Library: NangateOpenCellLibrary
 Wire Load Model Mode: top

Design	Wire Load Model	Library
myfir_unfolded_pipe_1som	5K_hvratio_1_1	NangateOpenCellLibrary

Global Operating Voltage = 1.1
 Power-specific unit information :
 Voltage Units = 1V
 Capacitance Units = 1.000000ff
 Time Units = 1ns
 Dynamic Power Units = 1uW derived from V,C,T units
 Leakage Power Units = 1nW

Cell Internal Power = 1.5084 mW 67%
 Net Switching Power = 755.1579 uW 33%

 Total Dynamic Power = 2.2635 mW 100%

Cell Leakage Power = 236.0995 uW

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	0.00%	
memory	0.0000	0.0000	0.0000	0.0000	0.00%	
black_box	0.0000	0.0000	0.0000	0.0000	0.00%	
clock_network	32.0402	27.9331	644.0339	60.6173	2.43%	
register	881.4445	130.1666	5.5496e+04	1.0671e+03	42.69%	
sequential	0.0000	0.0000	0.0000	0.0000	0.00%	

combinational	594.9025	597.0595	1.7996e+05	1.3719e+03	54.88%
Total	1.5084e+03 uW	755.1592 uW	2.3610e+05 nW	2.4996e+03 uW	
1					

Results of the synthesis with clock gating are shown in Table 3.3.

Architecture	Maximum frequency [MHz]	Area [μm^2]	Power [mW]	Simulation time [ns]
With Clock Gating	200	12044	2.499	504

Table 3.3: Results of synthesis with clock gating

3.6.1 Simulation 3-unfolded with 10-pipe with clock gating

CLK = 5.0 ns , tco = 500 ps

To process all the samples, the simulation lasts 504 ns, from 62 ns to 566 ns (as it can be seen in figure 3.16).

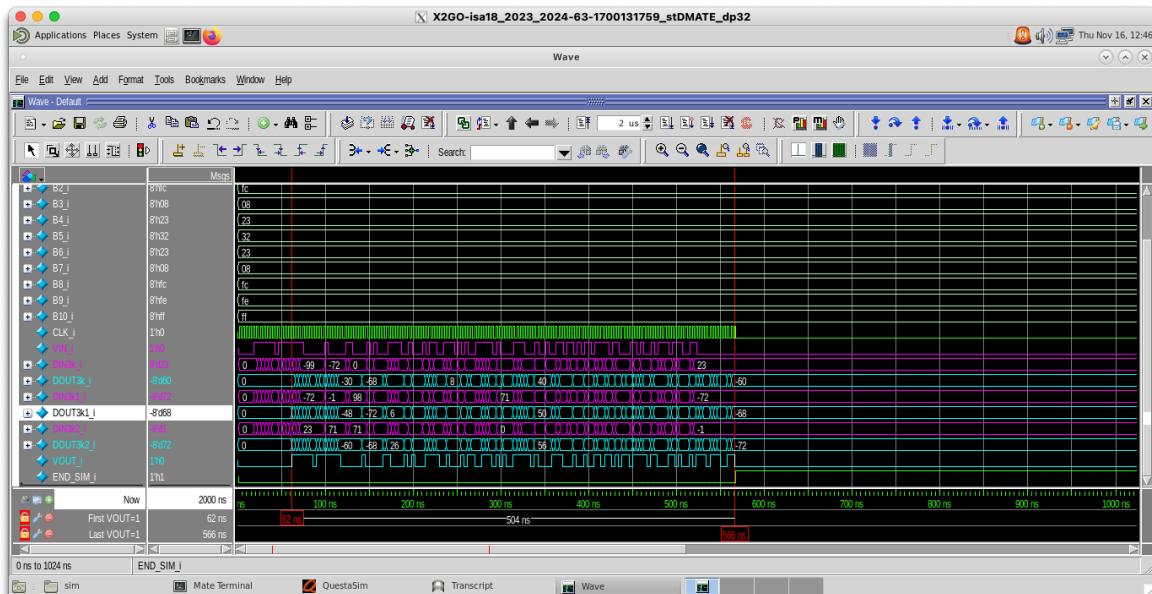


Figure 3.16: Complete simulation

3.7 Place and route

Using **Cadence Innovus** place and route is performed for the 10-pipe architecture already analyzed; in Figure 3.17 the result is shown.

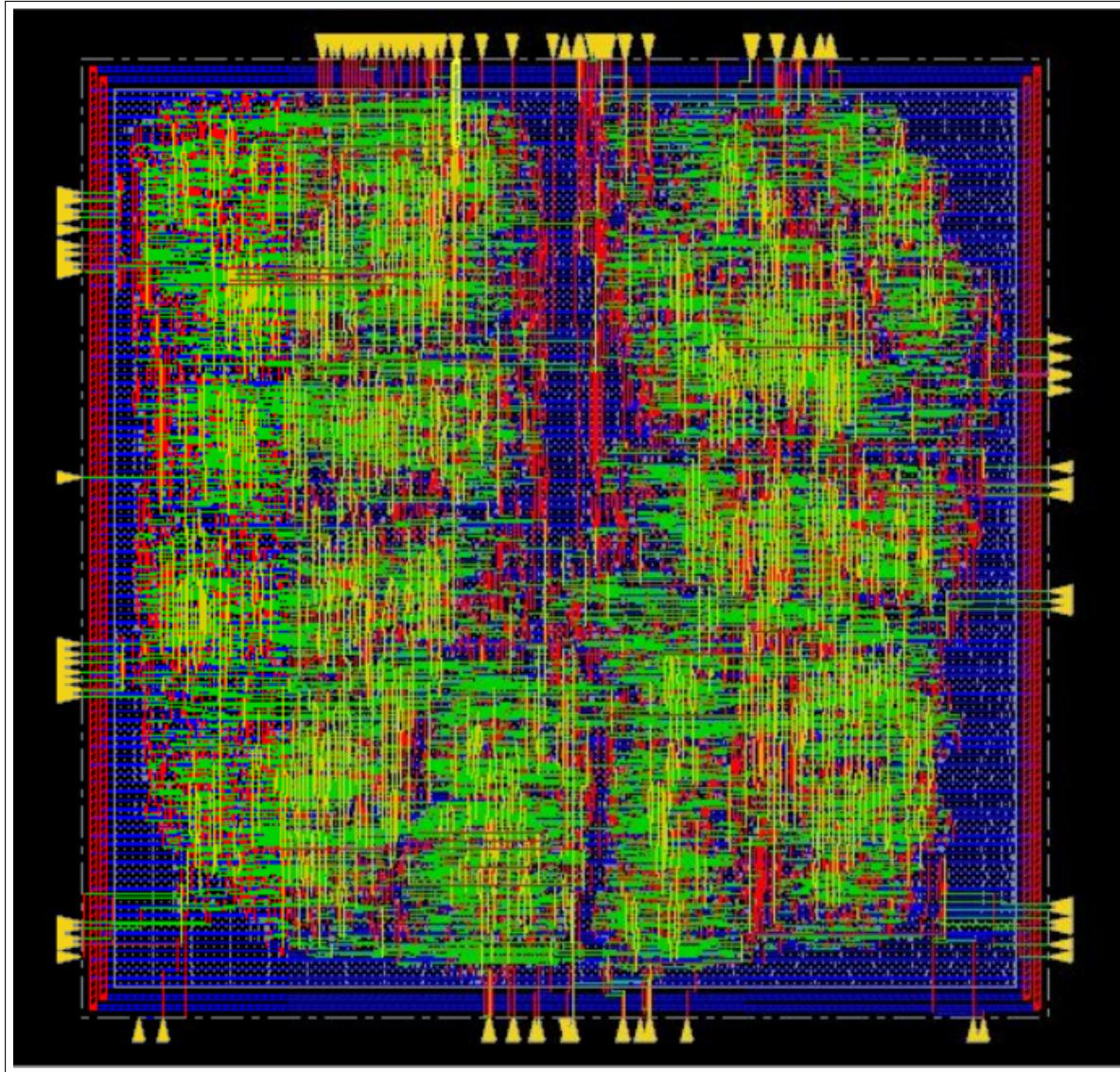


Figure 3.17: Place and Route result

Snapshots showing no timing violation (timeDesign Summary table for both setup and hold modes) and power consumption are shown in the following reports:

timeDesign Summary				
Setup views included: MyAnView				
Setup mode	all	reg2reg	reg2cgate	default
WNS (ns):	8.268	8.268	9.708	9.296
TNS (ns):	0.000	0.000	0.000	0.000
Violating Paths:	0	0	0	0
All Paths:	1291	516	10	765

Figure 3.18: No timing violation for setup mode

timeDesign Summary				
Hold views included: MyAnView				
Hold mode	all	reg2reg	reg2cgate	default
WNS (ns):	0.100	0.100	0.257	0.000
TNS (ns):	0.000	0.000	0.000	0.000
Violating Paths:	0	0	0	0
All Paths:	526	516	10	0

Figure 3.19: No timing violation for hold mode

 Gate Count

Gate area 0.7980 um²
 Level 0 Module myfir_unfolded_pipe_1som
 Gates=15090 Cells=5397 Area=12042.1 um²

 Power Report

 Total Power

Total Internal Power:	0.64443587	51.0026%
Total Switching Power:	0.38350754	30.3519%
Total Leakage Power:	0.23559207	18.6455%
Total Power:	1.26353547	

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %
Sequential	0.3816	0.05904	0.05551	0.4962	39.27
Macro	0	0	0	0	0
IO	0	0	0	0	0
Combinational	0.2436	0.2348	0.1792	0.6576	52.04
Clock Combinational	0.007326	0.08019	0.000214	0.08773	6.943
Clock Sequential	0.0119	0.009492	0.0006604	0.02205	1.745
Total	0.6444	0.3835	0.2356	1.264	100

Rail	Voltage	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %
VDD	1.1	0.6444	0.3835	0.2356	1.264	100

Clock	Internal Power	Switching Power	Leakage Power	Total Power	Percentage %
MY_CLK	0.01922	0.08968	0.0008744	0.1098	8.688
Total excluding duplicates	0.01922	0.08968	0.0008744	0.1098	8.688

Clock: MY_CLK
 Clock Period: 0.009972 usec
 Clock Toggle Rate: 200.5690 Mhz
 Clock Static Probability: 0.2500

Results of the place and route are shown in Table 3.4.

Architecture	Maximum frequency [MHz]	Area [μm^2]	Power [mW]	Simulation time [ns]
After place&route	100	12042	1.264	1260

Table 3.4: Frequency, area, power consumption and simulation time after place and route

3.7.1 Simulation 3-unfolded with 10-pipe with clock gating after Place and Route

CLK = 10.0 ns , tco = 1000 ps

To process all the samples the simulation lasts 1260 ns, from 145 ns to 1405 ns (as we can see in figure 3.20).

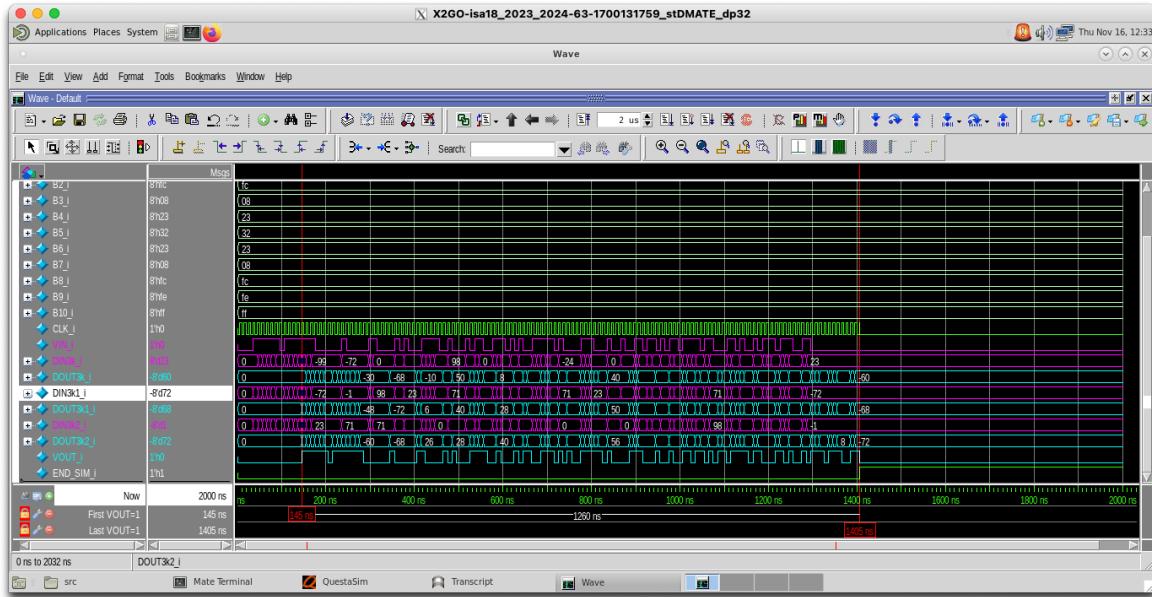


Figure 3.20: Complete simulation

3.8 Explanations, comparisons and comments

The application of clock gating reduces the operating frequency, in particular from 625 MHz to 200 MHz. Moreover, this technique is advantageous since the power consumption obtained is about half that of the initial one. The occupied area is lower than the architecture without the clock gating. The reason could be that at higher frequencies, it is necessary to insert buffers to drive the connections with low delay, while with a clock period of 5 ns less buffers are needed.

Synthesizing the architecture with clock gating, the power obtained is double the one obtained after applying the place and route with half of the initial frequency. Instead, the area remains almost unchanged because

since the architecture has been already synthesized, at a lower frequency, the only operation that could be done is moving the components in order to optimize parameters such as the skew.

The unfolded architecture without pipelining reaches a frequency equal to 454 MHz. As expected, this result is lower than the performance achieved applying pipelining where the maximum frequency reached is 625 MHz. The difference would have been even greater if Synopsys hadn't optimised the filter architecture using the tree-like implementation.

CHAPTER 4

Appendix

4.1 Matlab code

my_fir_filter.m

```
1 clear all
2 close all
3 clc
4
5 fs=10000; %% sampling frequency
6 f1=500; %% first sinewave freq (in band)
7 f2=3500; %% second sinewave freq (out band)
8
9 N=10; %% filter order
10 nb=8; %% number of bits
11
12 T=1/500; %% maximum period
13 tt=0:1/fs:10*T; %% time samples
14
15 x1=sin(2*pi*f1*tt); %% first sinewave
16 x2=sin(2*pi*f2*tt); %% second sinewave
17
18 x=(x1+x2)/2; %% input signal
19
20 [bi, bq]=myfir_design(N, nb); %% filter design
21
22 y=filter(bq, 1, x); %% apply filter
23
24 %% quantize input and output
25 xq=floor(x*2^(nb-1));
26 idx=find(xq==2^(nb-1));
27 xq(idx)=2^(nb-1)-1;
28
29 yq=floor(y*2^(nb-1));
30 idy=find(yq==2^(nb-1));
31 yq(idy)=2^(nb-1)-1;
32
33 %% plots
34 A=readmatrix('resultsc.txt');
35 figure
36 tiledlayout(2,1)
37 nexttile
38 plot(tt(1:120),x(1:120), 'g--+');
```

```

39 xlabel("t[s]")
40 ylabel("Amplitude")
41 hold on
42 plot(tt(1:120), y(1:120), 'c--o');
43 grid on
44 legend('x', 'y')
45 nexttile
46 plot(tt(1:120), xq(1:120));
47 hold on
48 plot(tt(1:120), yq(1:120));
49 hold on
50 plot(tt(1:120), A(1:120));
51 xlabel("t[s]")
52 ylabel("Amplitude")
53 grid on
54
55 %%
56 legend('X quantized','Y quantized','C results')
57 saveas(gcf, 'output_filter.png')
58 figure
59 plot(tt(1:100),x1(1:100), 'g--o')
60 hold on
61 plot(tt(1:100), x2(1:100), 'b--+')
62 hold on
63 plot(tt(1:100),x(1:100), 'r', LineWidth=1)
64 xlabel("t[s]")
65 ylabel("Amplitude")
66 legend('x1', 'x2', 'x')
67 title("Input signal")
68 saveas(gcf,"inputSignalFilter.png")
69 %% save input and output
70 fp=fopen('samples.txt','w');
71 fprintf(fp,'%d\n', xq);
72 fclose(fp);
73
74 fp=fopen('resultsm.txt', 'w');
75 fprintf(fp, '%d\n', yq);
76 fclose(fp);
77 %% test C THD
78 thd(A)
79 %%
80 figure
81 plot(tt(1:120), xq(1:120));
82 hold on
83 plot(tt(1:120), yq(1:120));
84 hold on
85 plot(tt(1:120),A(1:120));
86 xlabel("t[s]")
87 ylabel("Amplitude")
88 grid on
89 legend('X quantized','Y quantized','C results')
90 saveas(gcf, 'comparison.png')

```

Listing 4.1: **myfir_design.m**

```

1 function [bi, bq]=myfir_design(N,nb)
2 %% function myfir_design(N,nb)
3 %% N is order of the filter

```

```

4 %% nb is the number of bits
5 %% bi taps represented as integers
6
7 close all;
8
9 f_cut_off = 2000; % 2kHz
10 f_sampling = 10000; % 10kHz
11
12 f_nyq = f_sampling/2; %% Nyquist frequency
13 f0 = f_cut_off/f_nyq; %% normalized cut-off frequency
14
15 b=firl(N, f0); %% get filter coefficients
16 [h1, w1]=freqz(b); %% get the transfer function of the designed filter
17
18 bi=floor(b*2^(nb-1)); %% convert coefficients into nb-bit integers
19 bq=bi/2^(nb-1); %% convert back coefficients as nb-bit real values
20 [h2, w2]=freqz(bq); %% get the transfer function of the quantized filter
21
22 %% show the transfer functions
23 plot(w1/pi, 20*log10(abs(h1)));
24 hold on;
25 plot(w2/pi, 20*log10(abs(h2)), 'r--');
26 grid on;
27 title("Filter frequency response")
28 legend("continue coefficients","quantized coefficients")
29 xlabel('Normalized frequency');
30 ylabel('Transfer function (dB)');
31 saveas(gcf, "filterFrequencyResponse.png")

```

4.2 VHDL code

4.2.1 Standard architecture

Listing 4.2: **clk_gen.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity clk_gen is
7   port (
8     END_SIM : in std_logic;
9     CLK      : out std_logic;
10    RST_n   : out std_logic);
11 end clk_gen;
12
13 architecture beh of clk_gen is
14
15  constant Ts : time := 100 ns;
16
17  signal CLK_i : std_logic := '1';
18
19 begin  -- beh
20
21  process
22    begin  -- process

```

```

23   CLK_i <= not(CLK_i);
24   wait for Ts/2;
25 end process;
26
27 CLK <= CLK_i and not(END_SIM);
28
29 process
30 begin -- process
31   RST_n <= '0';
32   wait for 5*Ts/2;
33   RST_n <= '1';
34   wait;
35 end process;
36
37 end beh;
```

Listing 4.3: **data_maker_new.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 use ieee.std_logic_textio.all;
6
7 library std;
8 use std.textio.all;
9
10 entity data_maker_new is
11 generic
12 (
13   NBIT : integer := 8);
14 port
15 (
16   CLK      : in std_logic;
17   RST_n   : in std_logic;
18   VOUT    : out std_logic;
19   DOUT    : out std_logic_vector(NBIT - 1 downto 0);
20   B0      : out std_logic_vector(NBIT - 1 downto 0);
21   B1      : out std_logic_vector(NBIT - 1 downto 0);
22   B2      : out std_logic_vector(NBIT - 1 downto 0);
23   B3      : out std_logic_vector(NBIT - 1 downto 0);
24   B4      : out std_logic_vector(NBIT - 1 downto 0);
25   B5      : out std_logic_vector(NBIT - 1 downto 0);
26   B6      : out std_logic_vector(NBIT - 1 downto 0);
27   B7      : out std_logic_vector(NBIT - 1 downto 0);
28   B8      : out std_logic_vector(NBIT - 1 downto 0);
29   B9      : out std_logic_vector(NBIT - 1 downto 0);
30   B10     : out std_logic_vector(NBIT - 1 downto 0);
31   END_SIM : out std_logic);
32 end data_maker_new;
33
34 architecture beh of data_maker_new is
35
36   constant tco          : time    := 10 ns;
37   constant N_CYC_END_SIM : integer := 11;
38   constant LFSR_INIT    : integer := 1365;
39
40   signal sEndSim      : std_logic;
```

```

41 signal END_SIM_i : std_logic_vector(0 to N_CYC_END_SIM - 1);
42
43 signal lfsr : std_logic_vector(11 downto 0);
44 signal valid : std_logic;
45
46 begin -- beh
47
48   B0 <= conv_std_logic_vector(-1, NBIT);
49   B1 <= conv_std_logic_vector(-2, NBIT);
50   B2 <= conv_std_logic_vector(-4, NBIT);
51   B3 <= conv_std_logic_vector(8, NBIT);
52   B4 <= conv_std_logic_vector(35, NBIT);
53   B5 <= conv_std_logic_vector(50, NBIT);
54   B6 <= conv_std_logic_vector(35, NBIT);
55   B7 <= conv_std_logic_vector(8, NBIT);
56   B8 <= conv_std_logic_vector(-4, NBIT);
57   B9 <= conv_std_logic_vector(-2, NBIT);
58   B10 <= conv_std_logic_vector(-1, NBIT);
59
60   process (CLK, RST_n)
61     file fp_in      : text open READ_MODE is "../samples.txt";
62     variable line_in : line;
63     variable x       : integer;
64   begin -- process
65     if RST_n = '0' then -- asynchronous reset (active low)
66       DOUT    <= (others => '0') after tco;
67       VOUT    <= '0' after tco;
68       sEndSim <= '0' after tco;
69     elsif CLK'event and CLK = '1' then -- rising clock edge
70       if not endfile(fp_in) then
71         if (valid = '1') then
72           readline(fp_in, line_in);
73           read(line_in, x);
74           DOUT    <= conv_std_logic_vector(x, 8) after tco;
75           VOUT    <= '1' after tco;
76           sEndSim <= '0' after tco;
77         else
78           VOUT    <= '0' after tco;
79           sEndSim <= '0' after tco;
80         end if;
81       else
82         VOUT    <= '0' after tco;
83         sEndSim <= '1' after tco;
84       end if;
85     end if;
86   end process;
87
88   process (CLK, RST_n) is
89   begin -- process
90     if RST_n = '0' then -- asynchronous reset (active low)
91       valid <= '0' after tco;
92       lfsr  <= conv_std_logic_vector(LFSR_INIT, 12) after tco;
93     elsif CLK'event and CLK = '1' then -- rising clock edge
94       lfsr  <= (lfsr(0) xor lfsr(1)) & sxt(lfsr(11 downto 2), 11) after tco;
95       valid <= lfsr(0) after tco;
96     end if;
97   end process;
98
99   process (CLK, RST_n)
100  begin -- process
101    if RST_n = '0' then -- asynchronous reset (active low)
102      END_SIM_i <= (others => '0') after tco;
103    elsif CLK'event and CLK = '1' then -- rising clock edge

```

```

103     END_SIM_i(0)      <= sEndSim after tco;
104     END_SIM_i(1 to 10) <= END_SIM_i(0 to 9) after tco;
105   end if;
106 end process;
107
108 END_SIM <= END_SIM_i(10);
109
110 end beh;
```

Listing 4.4: **data_sink.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 use ieee.std_logic_textio.all;
6
7 library std;
8 use std.textio.all;
9
10 entity data_sink is
11   generic
12   (
13     NBIT : integer := 8);
14   port
15   (
16     CLK    : in std_logic;
17     RST_n : in std_logic;
18     VIN    : in std_logic;
19     DIN    : in std_logic_vector(NBIT - 1 downto 0));
20 end data_sink;
21
22 architecture beh of data_sink is
23
24 begin -- beh
25
26   process (CLK, RST_n)
27     file res_fp      : text open WRITE_MODE is "../results_hdl.txt";
28     variable line_out : line;
29     file fp_in       : text open READ_MODE is "../results.txt";
30     variable line_in  : line;
31     variable x        : integer;
32     variable cnt       : integer := 0;
33 begin -- process
34   if RST_n = '0' then -- asynchronous reset (active low)
35     cnt := 0;
36   elsif CLK'event and CLK = '1' then -- rising clock edge
37     if (VIN = '1') then
38       write(line_out, conv_integer(signed(DIN)));
39       writeline(res_fp, line_out);
40
41     if not endfile(fp_in) then
42       readline(fp_in, line_in);
43       read(line_in, x);
44       assert conv_integer(signed(DIN)) = x report "Results are different: index=" &
45       integer'image(cnt) & " c=" & integer'image(x) & " HDL=" & integer'image(conv_integer(
signed(DIN))) severity error;
46     else
```

```

46         assert VIN = '0' report "Reached EOF in results_c.txt" severity error;
47     end if;
48     cnt := cnt + 1;
49   end if;
50 end if;
51 end process;
52
53 end beh;
```

Listing 4.5: **myfir.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity myfir is
6   generic
7   (
8     NBIT : integer := 8
9   );
10  port
11  (
12    VIN, RST_n, CLK : in std_logic;
13    B0              : in std_logic_vector(NBIT - 1 downto 0);
14    B1              : in std_logic_vector(NBIT - 1 downto 0);
15    B2              : in std_logic_vector(NBIT - 1 downto 0);
16    B3              : in std_logic_vector(NBIT - 1 downto 0);
17    B4              : in std_logic_vector(NBIT - 1 downto 0);
18    B5              : in std_logic_vector(NBIT - 1 downto 0);
19    B6              : in std_logic_vector(NBIT - 1 downto 0);
20    B7              : in std_logic_vector(NBIT - 1 downto 0);
21    B8              : in std_logic_vector(NBIT - 1 downto 0);
22    B9              : in std_logic_vector(NBIT - 1 downto 0);
23    B10             : in std_logic_vector(NBIT - 1 downto 0);
24    DIN             : in std_logic_vector(NBIT - 1 downto 0);
25    DOUT            : out std_logic_vector(NBIT - 1 downto 0);
26    VOUT            : out std_logic
27  );
28 end myfir;
29
30 architecture beh of myfir is
31
32   signal x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10 : signed(NBIT - 1 downto 0);
33   signal b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_10 : signed(NBIT - 1 downto 0);
34
35   signal v_in          : std_logic;
36
37 begin
38
39   process (CLK, RST_n)
40
41     variable tmp: signed(NBIT - 2 downto 0);
42
43   begin
44     if RST_n = '0' then

```

```

46      x0      <= (others => '0');
47      x1      <= (others => '0');
48      x2      <= (others => '0');
49      x3      <= (others => '0');
50      x4      <= (others => '0');
51      x5      <= (others => '0');
52      x6      <= (others => '0');
53      x7      <= (others => '0');
54      x8      <= (others => '0');
55      x9      <= (others => '0');
56      x10     <= (others => '0');
57      b_0     <= (others => '0');
58      b_1     <= (others => '0');
59      b_2     <= (others => '0');
60      b_3     <= (others => '0');
61      b_4     <= (others => '0');
62      b_5     <= (others => '0');
63      b_6     <= (others => '0');
64      b_7     <= (others => '0');
65      b_8     <= (others => '0');
66      b_9     <= (others => '0');
67      b_10    <= (others => '0');
68      v_in   <= '0';
69      VOUT   <= '0';
70      DOUT   <= (others => '0');

71
72      elsif CLK'event and CLK = '1' then
73          v_in <= VIN;
74          b_0  <= signed(B0);
75          b_1  <= signed(B1);
76          b_2  <= signed(B2);
77          b_3  <= signed(B3);
78          b_4  <= signed(B4);
79          b_5  <= signed(B5);
80          b_6  <= signed(B6);
81          b_7  <= signed(B7);
82          b_8  <= signed(B8);
83          b_9  <= signed(B9);
84          b_10 <= signed(B10);
85          VOUT <= v_in;

86
87      if VIN = '1' then
88          x0  <= signed(DIN);
89          x1  <= x0;
90          x2  <= x1;
91          x3  <= x2;
92          x4  <= x3;
93          x5  <= x4;
94          x6  <= x5;
95          x7  <= x6;
96          x8  <= x7;
97          x9  <= x8;
98          x10 <= x9;

99
100     end if;
101     -- eseguo operazioni troncando moltiplicazione (THD circa -30dB)
102     tmp := "*" (x0 , b_0) (14 downto 8) + "*" (x1 , b_1) (14 downto 8) +
103                  "*" (x2 , b_2) (14 downto 8) + "*" (x3 , b_3) (14 downto 8) +
104                  "*" (x4 , b_4) (14 downto 8) + "*" (x5 , b_5) (14 downto 8) +
105                  "*" (x6 , b_6) (14 downto 8) + "*" (x7 , b_7) (14 downto 8) +
106                  "*" (x8 , b_8) (14 downto 8) + "*" (x9 , b_9) (14 downto 8) +
107                  "*" (x10, b_10) (14 downto 8);

```

```

108      if v_in = '1' then
109          DOUT <= std_logic_vector(tmp) & '0';
110      end if;
111  end if;
112
113  end process;
114 end beh;
```

Listing 4.6: **tb_fir.v**

```

1 //`timescale 1ns
2
3 module tb_fir ();
4
5     wire CLK_i;
6     wire RST_n_i;
7     wire [7:0] DIN_i;
8     wire VIN_i;
9     wire [7:0] B0_i;
10    wire [7:0] B1_i;
11    wire [7:0] B2_i;
12    wire [7:0] B3_i;
13    wire [7:0] B4_i;
14    wire [7:0] B5_i;
15    wire [7:0] B6_i;
16    wire [7:0] B7_i;
17    wire [7:0] B8_i;
18    wire [7:0] B9_i;
19    wire [7:0] B10_i;
20    wire [7:0] DOUT_i;
21    wire VOUT_i;
22    wire END_SIM_i;
23
24 clk_gen CG(.END_SIM(END_SIM_i),
25             .CLK(CLK_i),
26             .RST_n(RST_n_i));
27
28 data_maker_new SM(.CLK(CLK_i),
29                   .RST_n(RST_n_i),
30                   .VOUT(VIN_i),
31                   .DOUT(DIN_i),
32                   .B0(B0_i),
33                   .B1(B1_i),
34                   .B2(B2_i),
35                   .B3(B3_i),
36                   .B4(B4_i),
37                   .B5(B5_i),
38                   .B6(B6_i),
39                   .B7(B7_i),
40                   .B8(B8_i),
41                   .B9(B9_i),
42                   .B10(B10_i),
43                   .END_SIM(END_SIM_i));
44
45 myfir UUT(.CLK(CLK_i),
46            .RST_n(RST_n_i),
47            .DIN(DIN_i),
```

```

48      .VIN(VIN_i),
49      .B0(B0_i),
50      .B1(B1_i),
51      .B2(B2_i),
52      .B3(B3_i),
53      .B4(B4_i),
54      .B5(B5_i),
55      .B6(B6_i),
56      .B7(B7_i),
57      .B8(B8_i),
58      .B9(B9_i),
59      .B10(B10_i),
60      .DOUT(DOUT_i),
61      .VOUT(VOUT_i));
62
63 data_sink DS(.CLK(CLK_i),
64             .RST_n(RST_n_i),
65             .VIN(VOUT_i),
66             .DIN(DOUT_i));
67
68 endmodule

```

4.2.2 Unfolded architecture

Listing 4.7: **data_maker_new_unfolded.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 use ieee.std_logic_textio.all;
6
7 library std;
8 use std.textio.all;
9
10 entity data_maker is
11   generic (
12     NBIT : integer := 8);
13   port (
14     CLK      : in  std_logic;
15     RST_n   : in  std_logic;
16     VOUT    : out std_logic;
17     DOUT3k  : out std_logic_vector(NBIT-1 downto 0);
18     DOUT3k1 : out std_logic_vector(NBIT-1 downto 0);
19     DOUT3k2 : out std_logic_vector(NBIT-1 downto 0);
20     B0      : out std_logic_vector(NBIT - 1 downto 0);
21     B1      : out std_logic_vector(NBIT - 1 downto 0);
22     B2      : out std_logic_vector(NBIT - 1 downto 0);
23     B3      : out std_logic_vector(NBIT - 1 downto 0);
24     B4      : out std_logic_vector(NBIT - 1 downto 0);
25     B5      : out std_logic_vector(NBIT - 1 downto 0);
26     B6      : out std_logic_vector(NBIT - 1 downto 0);
27     B7      : out std_logic_vector(NBIT - 1 downto 0);
28     B8      : out std_logic_vector(NBIT - 1 downto 0);
29     B9      : out std_logic_vector(NBIT - 1 downto 0);
30     B10     : out std_logic_vector(NBIT - 1 downto 0);
31     END_SIM : out std_logic);
32 end data_maker;

```

```

33
34 architecture beh of data_maker is
35
36 constant tco : time := 10 ns;
37 constant N_CYC_END_SIM : integer := 11;
38 constant LFSR_INIT : integer := 1365;
39
40 signal sEndSim : std_logic;
41 signal END_SIM_i : std_logic_vector(0 to N_CYC_END_SIM-1);
42
43 signal lfsr : std_logic_vector(11 downto 0);
44 signal valid : std_logic;
45
46 begin -- beh
47 B0 <= conv_std_logic_vector(-1, NBIT);
48 B1 <= conv_std_logic_vector(-2, NBIT);
49 B2 <= conv_std_logic_vector(-4, NBIT);
50 B3 <= conv_std_logic_vector(8, NBIT);
51 B4 <= conv_std_logic_vector(35, NBIT);
52 B5 <= conv_std_logic_vector(50, NBIT);
53 B6 <= conv_std_logic_vector(35, NBIT);
54 B7 <= conv_std_logic_vector(8, NBIT);
55 B8 <= conv_std_logic_vector(-4, NBIT);
56 B9 <= conv_std_logic_vector(-2, NBIT);
57 B10 <= conv_std_logic_vector(-1, NBIT);
58
59
60 process (CLK, RST_n)
61 file fp_in : text open READ_MODE is "./samples.txt";
62 variable line_in : line;
63 variable x3k : integer;
64 variable x3k1 : integer;
65 variable x3k2 : integer;
66 begin -- process
67 if RST_n = '0' then -- asynchronous reset (active low)
68 DOUT3k <= (others => '0') after tco;
69 DOUT3k1 <= (others => '0') after tco;
70 DOUT3k2 <= (others => '0') after tco;
71 VOUT <= '0' after tco;
72 sEndSim <= '0' after tco;
73 elsif CLK'event and CLK = '1' then -- rising clock edge
74 if not endfile(fp_in) then
75 if (valid = '1') then
76 -----
77 -- NOTE: it works only if samples contains 3k samples
78 -----
79 readline(fp_in, line_in);
80 read(line_in, x3k);
81 readline(fp_in, line_in);
82 read(line_in, x3k1);
83 readline(fp_in, line_in);
84 read(line_in, x3k2);
85 DOUT3k <= conv_std_logic_vector(x3k, 8) after tco;
86 DOUT3k1 <= conv_std_logic_vector(x3k1, 8) after tco;
87 DOUT3k2 <= conv_std_logic_vector(x3k2, 8) after tco;
88 VOUT <= '1' after tco;
89 sEndSim <= '0' after tco;
90 else
91 VOUT <= '0' after tco;
92 sEndSim <= '0' after tco;
93 end if;
94 else

```

```

95      VOUT <= '0' after tco;
96      sEndSim <= '1' after tco;
97  end if;
98 end if;
99 end process;
100
101 process (CLK, RST_n) is
102 begin -- process
103  if RST_n = '0' then                      -- asynchronous reset (active low)
104    valid <= '0' after tco;
105    lfsr <= conv_std_logic_vector(LFSR_INIT, 12) after tco;
106  elsif CLK'event and CLK = '1' then -- rising clock edge
107    lfsr <= (lfsr(0) xor lfsr(1)) & sxt(lfsr(11 downto 2), 11) after tco;
108    valid <= lfsr(0) after tco;
109  end if;
110 end process;
111
112 process (CLK, RST_n)
113 begin -- process
114  if RST_n = '0' then                      -- asynchronous reset (active low)
115    END_SIM_i <= (others => '0') after tco;
116  elsif CLK'event and CLK = '1' then -- rising clock edge
117    END_SIM_i(0) <= sEndSim after tco;
118    END_SIM_i(1 to 10) <= END_SIM_i(0 to 9) after tco;
119  end if;
120 end process;
121
122 END_SIM <= END_SIM_i(10);
123
124 end beh;
```

Listing 4.8: **data_sink_unfolded.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5 use ieee.std_logic_textio.all;
6
7 library std;
8 use std.textio.all;
9
10 entity data_sink is
11 generic (
12   NBIT : integer := 8);
13 port (
14   CLK    : in std_logic;
15   RST_n : in std_logic;
16   VIN   : in std_logic;
17   DIN3k  : in std_logic_vector(NBIT-1 downto 0);
18   DIN3k1 : in std_logic_vector(NBIT-1 downto 0);
19   DIN3k2 : in std_logic_vector(NBIT-1 downto 0));
20 end data_sink;
21
22 architecture beh of data_sink is
23
24 begin -- beh
```

```

26  process (CLK, RST_n)
27    file res_fp : text open WRITE_MODE is "./results_hdl.txt";
28    variable line_out : line;
29    file fp_in : text open READ_MODE is "./results.txt";
30    variable line_in : line;
31    variable x3k : integer;
32    variable x3k1 : integer;
33    variable x3k2 : integer;
34    variable cnt : integer := 0;
35 begin -- process
36   if RST_n = '0' then          -- asynchronous reset (active low)
37     cnt := 0;
38   elsif CLK'event and CLK = '1' then -- rising clock edge
39     if (VIN = '1') then
40       write(line_out, conv_integer(signed(DIN3k)));
41       writeline(res_fp, line_out);
42       write(line_out, conv_integer(signed(DIN3k1)));
43       writeline(res_fp, line_out);
44       write(line_out, conv_integer(signed(DIN3k2)));
45       writeline(res_fp, line_out);
46
47     if not endfile(fp_in) then
48       -----
49       -- NOTE: it works only if samples contains 3k samples
50       -----
51       readline(fp_in, line_in);
52       read(line_in, x3k);
53       readline(fp_in, line_in);
54       read(line_in, x3k1);
55       readline(fp_in, line_in);
56       read(line_in, x3k2);
57       assert conv_integer(signed(DIN3k)) = x3k report "Results are different: index=" &
58       integer'image(cnt) & " 0" & " c=" & integer'image(x3k) & " HDL=" & integer'image(
59       conv_integer(signed(DIN3k))) severity error;
60       assert conv_integer(signed(DIN3k1)) = x3k1 report "Results are different: index=" &
61       integer'image(cnt) & " 1" & " c=" & integer'image(x3k1) & " HDL=" & integer'image(
62       conv_integer(signed(DIN3k1))) severity error;
63       assert conv_integer(signed(DIN3k2)) = x3k2 report "Results are different: index=" &
64       integer'image(cnt) & " 2" & " c=" & integer'image(x3k2) & " HDL=" & integer'image(
65       conv_integer(signed(DIN3k2))) severity error;
66     else
67       assert VIN = '0' report "Reached EOF in results_c.txt" severity error;
68     end if;
69     cnt := cnt + 1;
70   end if;
71 end if;
72 end process;
73
74 end beh;

```

Listing 4.9: **myfir_unfolded.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity myfir_unfolded is
6   generic(NBIT : integer := 8);

```

```

7   port
8     ( VIN, RST_n, CLK
9       B0
10    B1
11    B2
12    B3
13    B4
14    B5
15    B6
16    B7
17    B8
18    B9
19    B10
20   DIN3k,DIN3k1,DIN3k2
21   DOUT3k, DOUT3k1,DOUT3k2
22   VOUT
23 end myfir_unfolded;
24
25 architecture beh of myfir_unfolded is
26
27   signal x3k2, x3k1, x3k, x3k_1, x3k_2, x3k_3, x3k_4,
28     x3k_5, x3k_6, x3k_7, x3k_8, x3k_9, x3k_10 : signed(NBIT - 1 downto 0);
29   signal b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7,
30     b_8, b_9, b_10 : signed(NBIT - 1 downto 0);
31
32   signal v_in : std_logic;
33
34 begin
35
36   process (CLK, RST_n)
37
38     variable tmp3k, tmp3k1, tmp3k2 : signed(NBIT - 2 downto 0);
39
40   begin
41     if RST_n = '0' then
42       x3k2      <= (others => '0');
43       x3k1      <= (others => '0');
44       x3k       <= (others => '0');
45       x3k_1     <= (others => '0');
46       x3k_2     <= (others => '0');
47       x3k_3     <= (others => '0');
48       x3k_4     <= (others => '0');
49       x3k_5     <= (others => '0');
50       x3k_6     <= (others => '0');
51       x3k_7     <= (others => '0');
52       x3k_8     <= (others => '0');
53       x3k_9     <= (others => '0');
54       x3k_10    <= (others => '0');
55       b_0       <= (others => '0');
56       b_1       <= (others => '0');
57       b_2       <= (others => '0');
58       b_3       <= (others => '0');
59       b_4       <= (others => '0');
60       b_5       <= (others => '0');
61       b_6       <= (others => '0');
62       b_7       <= (others => '0');
63       b_8       <= (others => '0');
64       b_9       <= (others => '0');
65       b_10      <= (others => '0');
66       v_in      <= '0';
67       VOUT      <= '0';
68       DOUT3k   <= (others => '0');

```

```

68     DOUT3k1 <= (others => '0');
69     DOUT3k2 <= (others => '0');
70
71     elsif CLK'event and CLK = '1' then
72         v_in <= VIN;
73         b_0  <= signed(B0);
74         b_1  <= signed(B1);
75         b_2  <= signed(B2);
76         b_3  <= signed(B3);
77         b_4  <= signed(B4);
78         b_5  <= signed(B5);
79         b_6  <= signed(B6);
80         b_7  <= signed(B7);
81         b_8  <= signed(B8);
82         b_9  <= signed(B9);
83         b_10 <= signed(B10);
84         VOUT <= v_in;
85
86         if VIN = '1' then
87             x3k  <= signed(DIN3k);
88             x3k1 <= signed(DIN3k1);
89             x3k2 <= signed(DIN3k2);
90             x3k_3 <= x3k;
91             x3k_6 <= x3k_3;
92             x3k_9 <= x3k_6;
93             x3k_2 <= x3k1;
94             x3k_5 <= x3k_2;
95             x3k_8 <= x3k_5;
96             x3k_1 <= x3k2;
97             x3k_4 <= x3k_1;
98             x3k_7 <= x3k_4;
99             x3k_10 <= x3k_7;
100
101        end if;
102        -- eseguo operazioni troncando moltiplicazione (THD circa -30dB)
103        tmp3k := "*" (x3k, b_0) (14 downto 8) +
104            "*" (x3k_1, b_1) (14 downto 8) +
105            "*" (x3k_2, b_2) (14 downto 8) +
106            "*" (x3k_3, b_3) (14 downto 8) +
107            "*" (x3k_4, b_4) (14 downto 8) +
108            "*" (x3k_5, b_5) (14 downto 8) +
109            "*" (x3k_6, b_6) (14 downto 8) +
110            "*" (x3k_7, b_7) (14 downto 8) +
111            "*" (x3k_8, b_8) (14 downto 8) +
112            "*" (x3k_9, b_9) (14 downto 8) +
113            "*" (x3k_10, b_10) (14 downto 8);
114
115        tmp3k1 := "*" (x3k1, b_0) (14 downto 8) +
116            "*" (x3k, b_1) (14 downto 8) +
117            "*" (x3k_1, b_2) (14 downto 8) +
118            "*" (x3k_2, b_3) (14 downto 8) +
119            "*" (x3k_3, b_4) (14 downto 8) +
120            "*" (x3k_4, b_5) (14 downto 8) +
121            "*" (x3k_5, b_6) (14 downto 8) +
122            "*" (x3k_6, b_7) (14 downto 8) +
123            "*" (x3k_7, b_8) (14 downto 8) +
124            "*" (x3k_8, b_9) (14 downto 8) +
125            "*" (x3k_9, b_10) (14 downto 8);
126
127        tmp3k2 := "*" (x3k2, b_0) (14 downto 8) +
128            "*" (x3k1, b_1) (14 downto 8) +
129            "*" (x3k, b_2) (14 downto 8) +

```

```

130      ".*"(x3k_1, b_3)(14 downto 8) +
131      ".*"(x3k_2, b_4)(14 downto 8) +
132      ".*"(x3k_3, b_5)(14 downto 8) +
133      ".*"(x3k_4, b_6)(14 downto 8) +
134      ".*"(x3k_5, b_7)(14 downto 8) +
135      ".*"(x3k_6, b_8)(14 downto 8) +
136      ".*"(x3k_7, b_9)(14 downto 8) +
137      ".*"(x3k_8, b_10)(14 downto 8);

138  if v_in = '1' then
139      DOUT3k <= std_logic_vector(tmp3k) & '0';
140      DOUT3k1 <= std_logic_vector(tmp3k1) & '0';
141      DOUT3k2 <= std_logic_vector(tmp3k2) & '0';
142  end if;
143 end if;
144
145
146 end process;
147 end beh;
```

Listing 4.10: **myfir_unfolded_pipe_10stages.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity myfir_unfolded_pipe_10stages is
6 generic( NBIT : integer := 8);
7 port
8 ( VIN, RST_n, CLK
9      : in std_logic;
10     B0
11      : in std_logic_vector(NBIT - 1 downto 0);
12     B1
13      : in std_logic_vector(NBIT - 1 downto 0);
14     B2
15      : in std_logic_vector(NBIT - 1 downto 0);
16     B3
17      : in std_logic_vector(NBIT - 1 downto 0);
18     B4
19      : in std_logic_vector(NBIT - 1 downto 0);
20     B5
21      : in std_logic_vector(NBIT - 1 downto 0);
22     B6
23      : in std_logic_vector(NBIT - 1 downto 0);
24     B7
25      : in std_logic_vector(NBIT - 1 downto 0);
26     B8
27      : in std_logic_vector(NBIT - 1 downto 0);
28     B9
29      : in std_logic_vector(NBIT - 1 downto 0);
30     B10
31      : in std_logic_vector(NBIT - 1 downto 0);
32     DIN3k,DIN3k1,DIN3k2
33      : in std_logic_vector(NBIT - 1 downto 0);
34     DOUT3k, DOUT3k1,DOUT3k2
35      : out std_logic_vector(NBIT - 1 downto 0);
36     VOUT
37      : out std_logic);
38 end myfir_unfolded_pipe_10stages;
39
40
41 architecture beh of myfir_unfolded_pipe_10stages is
42
43 signal x3k2, x3k1, x3k, x3k_1, x3k_2, x3k_3, x3k_4, x3k_5, x3k_6, x3k_7, x3k_8, x3k_9,
44 x3k_10 : signed(NBIT - 1 downto 0);
45 signal b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_10 : signed(NBIT - 1 downto 0);
46
47 signal v_in, v_in_pipe1, v_in_pipe2, v_in_pipe3, v_in_pipe4, v_in_pipe5, v_in_pipe6,
48 v_in_pipe7, v_in_pipe8, v_in_pipe9 : std_logic;
49 signal reg_pipe1, reg_pipe2, reg_pipe3, reg_pipe4, reg_pipe5, reg_pipe6, reg_pipe7,
50 reg_pipe8, reg_pipe9,
51     reg_pipe1_1, reg_pipe1_2, reg_pipe1_3, reg_pipe1_4, reg_pipe1_5, reg_pipe1_6,
52     reg_pipe1_7, reg_pipe1_8, reg_pipe1_9,
```

```

32      reg_pipe2_1, reg_pipe2_2, reg_pipe2_3, reg_pipe2_4, reg_pipe2_5, reg_pipe2_6,
33      reg_pipe2_7, reg_pipe2_8, reg_pipe2_9: signed(NBIT - 1 downto 0);
34      signal tmp_pipe1, tmp_pipe2, tmp_pipe3, tmp_pipe4, tmp_pipe5, tmp_pipe6, tmp_pipe7,
35      tmp_pipe8, tmp_pipe9,
36      tmp_pipe1_1, tmp_pipe1_2, tmp_pipe1_3, tmp_pipe1_4, tmp_pipe1_5, tmp_pipe1_6,
37      tmp_pipe1_7, tmp_pipe1_8, tmp_pipe1_9,
38      tmp_pipe2_1, tmp_pipe2_2, tmp_pipe2_3, tmp_pipe2_4, tmp_pipe2_5, tmp_pipe2_6,
39      tmp_pipe2_7, tmp_pipe2_8, tmp_pipe2_9 : signed(NBIT - 2 downto 0);
40
41 begin
42
43 process (CLK, RST_n)
44
45 begin
46   if RST_n = '0' then
47
48     x3k2          <= (others => '0');
49     x3k1          <= (others => '0');
50     x3k            <= (others => '0');
51     x3k_1         <= (others => '0');
52     x3k_2         <= (others => '0');
53     x3k_3         <= (others => '0');
54     x3k_4         <= (others => '0');
55     x3k_5         <= (others => '0');
56     x3k_6         <= (others => '0');
57     x3k_7         <= (others => '0');
58     x3k_8         <= (others => '0');
59     x3k_9         <= (others => '0');
60     x3k_10        <= (others => '0');
61
62   reg_pipe1 <= (others => '0'); reg_pipe1_1 <= (others => '0'); reg_pipe2_1 <= (
63   others => '0');
64   reg_pipe2 <= (others => '0'); reg_pipe1_2 <= (others => '0'); reg_pipe2_2 <= (
65   others => '0');
66   reg_pipe3 <= (others => '0'); reg_pipe1_3 <= (others => '0'); reg_pipe2_3 <= (
67   others => '0');
68   reg_pipe4 <= (others => '0'); reg_pipe1_4 <= (others => '0'); reg_pipe2_4 <= (
69   others => '0');
70   reg_pipe5 <= (others => '0'); reg_pipe1_5 <= (others => '0'); reg_pipe2_5 <= (
71   others => '0');
72   reg_pipe6 <= (others => '0'); reg_pipe1_6 <= (others => '0'); reg_pipe2_6 <= (
73   others => '0');
74   reg_pipe7 <= (others => '0'); reg_pipe1_7 <= (others => '0'); reg_pipe2_7 <= (
75   others => '0');
76   reg_pipe8 <= (others => '0'); reg_pipe1_8 <= (others => '0'); reg_pipe2_8 <= (
77   others => '0');
78   reg_pipe9 <= (others => '0'); reg_pipe1_9 <= (others => '0'); reg_pipe2_9 <= (
79   others => '0');
80
81   tmp_pipe1 <= (others => '0'); tmp_pipe1_1 <= (others => '0'); tmp_pipe2_1 <= (
82   others => '0');
83   tmp_pipe2 <= (others => '0'); tmp_pipe1_2 <= (others => '0'); tmp_pipe2_2 <= (
84   others => '0');
85   tmp_pipe3 <= (others => '0'); tmp_pipe1_3 <= (others => '0'); tmp_pipe2_3 <= (
86   others => '0');

```

```

75     tmp_pipe4 <= (others => '0'); tmp_pipe1_4 <= (others => '0'); tmp_pipe2_4 <= (others => '0');
76     tmp_pipe5 <= (others => '0'); tmp_pipe1_5 <= (others => '0'); tmp_pipe2_5 <= (others => '0');
77     tmp_pipe6 <= (others => '0'); tmp_pipe1_6 <= (others => '0'); tmp_pipe2_6 <= (others => '0');
78     tmp_pipe7 <= (others => '0'); tmp_pipe1_7 <= (others => '0'); tmp_pipe2_7 <= (others => '0');
79     tmp_pipe8 <= (others => '0'); tmp_pipe1_8 <= (others => '0'); tmp_pipe2_8 <= (others => '0');
80     tmp_pipe9 <= (others => '0'); tmp_pipe1_9 <= (others => '0'); tmp_pipe2_9 <= (others => '0');

81     b_0 <= (others => '0');
82     b_1 <= (others => '0');
83     b_2 <= (others => '0');
84     b_3 <= (others => '0');
85     b_4 <= (others => '0');
86     b_5 <= (others => '0');
87     b_6 <= (others => '0');
88     b_7 <= (others => '0');
89     b_8 <= (others => '0');
90     b_9 <= (others => '0');
91     b_10 <= (others => '0');

92
93
94     v_in <= '0'; v_in_pipe1 <= '0'; v_in_pipe2 <= '0'; v_in_pipe3 <= '0'; v_in_pipe4 <=
95     '0';
96     v_in_pipe5 <= '0'; v_in_pipe6 <= '0'; v_in_pipe7 <= '0'; v_in_pipe8 <= '0';
97     v_in_pipe9 <= '0';
98     VOUT <= '0';

99
100
101
102    elsif CLK'event and CLK = '1' then
103        b_0 <= signed(B0);
104        b_1 <= signed(B1);
105        b_2 <= signed(B2);
106        b_3 <= signed(B3);
107        b_4 <= signed(B4);
108        b_5 <= signed(B5);
109        b_6 <= signed(B6);
110        b_7 <= signed(B7);
111        b_8 <= signed(B8);
112        b_9 <= signed(B9);
113        b_10 <= signed(B10);

114
115        v_in_pipe1 <= VIN;
116        v_in_pipe2 <= v_in_pipe1;
117        v_in_pipe3 <= v_in_pipe2;
118        v_in_pipe4 <= v_in_pipe3;
119        v_in_pipe5 <= v_in_pipe4;
120        v_in_pipe6 <= v_in_pipe5;
121        v_in_pipe7 <= v_in_pipe6;
122        v_in_pipe8 <= v_in_pipe7;
123        v_in_pipe9 <= v_in_pipe8;
124        v_in <= v_in_pipe9;
125        VOUT <= v_in;

126
127        reg_pipe1 <= x3k;
128        reg_pipe1_1 <= x3k1;

```

```
129     reg_pipe2_1 <= x3k_1;
130
131     reg_pipe2 <= reg_pipe1;
132     reg_pipe1_2 <= x3k_2;
133     reg_pipe2_2 <= reg_pipe2_1;
134
135     reg_pipe3 <= x3k_3;
136     reg_pipe1_3 <= reg_pipe1_2;
137     reg_pipe2_3 <= reg_pipe2_2;
138
139     reg_pipe4 <= reg_pipe3;
140     reg_pipe1_4 <= reg_pipe1_3;
141     reg_pipe2_4 <= x3k_4;
142
143     reg_pipe5 <= reg_pipe4;
144     reg_pipe1_5 <= x3k_5;
145     reg_pipe2_5 <= reg_pipe2_4;
146
147     reg_pipe6 <= x3k_6;
148     reg_pipe1_6 <= reg_pipe1_5;
149     reg_pipe2_6 <= reg_pipe2_5;
150
151     reg_pipe2_7 <= x3k_7;
152     reg_pipe7 <= reg_pipe6;
153     reg_pipe1_7 <= reg_pipe1_6;
154
155     reg_pipe8 <= reg_pipe7;
156     reg_pipe1_8 <= x3k_8;
157     reg_pipe2_8 <= reg_pipe2_7;
158
159     reg_pipe9 <= x3k_9;
160     reg_pipe1_9 <= reg_pipe1_8;
161     reg_pipe2_9 <= reg_pipe2_8;
162
163     if VIN = '1' then
164         x3k  <= signed(DIN3k);
165         x3k1 <= signed(DIN3k1);
166         x3k2 <= signed(DIN3k2);
167
168         x3k_1 <= x3k2;
169     end if;
170     if v_in_pipe1 = '1' then
171         x3k_2 <= reg_pipe1_1;
172     end if;
173
174     if v_in_pipe2 = '1' then
175         x3k_3 <= reg_pipe2;
176     end if;
177
178     if v_in_pipe3 = '1' then
179         x3k_4 <= reg_pipe2_3;
180     end if;
181
182     if v_in_pipe4 = '1' then
183         x3k_5 <= reg_pipe1_4;
184     end if;
185
186     if v_in_pipe5 = '1' then
187         x3k_6 <= reg_pipe5;
188     end if;
189
190     if v_in_pipe6 = '1' then
```

```

191         x3k_7 <= reg_pipe2_6;
192     end if;
193
194     if v_in_pipe7 = '1' then
195         x3k_8 <= reg_pipe1_7;
196     end if;
197
198     if v_in_pipe8 = '1' then
199         x3k_9 <= reg_pipe8;
200     end if;
201
202     if v_in_pipe9 = '1' then
203         x3k_10 <= reg_pipe2_9;
204     end if;
205
206     -- eseguo operazioni troncando moltiplicazione (THD circa -30dB)
207     tmp3k_s1 := "*" (x3k, b_0) (14 downto 8) + "*" (x3k_1, b_1) (14 downto 8);
208     tmp_pipe1 <= tmp3k_s1;
209     tmp3k_s2 := tmp_pipe1 + "*" (x3k_2, b_2) (14 downto 8);
210     tmp_pipe2 <= tmp3k_s2;
211     tmp3k_s3 := tmp_pipe2 + "*" (x3k_3, b_3) (14 downto 8);
212     tmp_pipe3 <= tmp3k_s3;
213     tmp3k_s4 := tmp_pipe3 + "*" (x3k_4, b_4) (14 downto 8);
214     tmp_pipe4 <= tmp3k_s4;
215     tmp3k_s5 := tmp_pipe4 + "*" (x3k_5, b_5) (14 downto 8);
216     tmp_pipe5 <= tmp3k_s5;
217     tmp3k_s6 := tmp_pipe5 + "*" (x3k_6, b_6) (14 downto 8);
218     tmp_pipe6 <= tmp3k_s6;
219     tmp3k_s7 := tmp_pipe6 + "*" (x3k_7, b_7) (14 downto 8);
220     tmp_pipe7 <= tmp3k_s7;
221     tmp3k_s8 := tmp_pipe7 + "*" (x3k_8, b_8) (14 downto 8);
222     tmp_pipe8 <= tmp3k_s8;
223     tmp3k_s9 := tmp_pipe8 + "*" (x3k_9, b_9) (14 downto 8);
224     tmp_pipe9 <= tmp3k_s9;
225     tmp3k_s10 := tmp_pipe9 + "*" (x3k_10, b_10) (14 downto 8);

226
227
228     tmp3k1_s1 := "*" (x3k1, b_0) (14 downto 8) + "*" (x3k, b_1) (14 downto 8);
229     tmp_pipe1_1 <= tmp3k1_s1;
230     tmp3k1_s2 := tmp_pipe1_1 + "*" (reg_pipe2_1, b_2) (14 downto 8);
231     tmp_pipe1_2 <= tmp3k1_s2;
232     tmp3k1_s3 := tmp_pipe1_2 + "*" (reg_pipe1_2, b_3) (14 downto 8);
233     tmp_pipe1_3 <= tmp3k1_s3;
234     tmp3k1_s4 := tmp_pipe1_3 + "*" (reg_pipe3, b_4) (14 downto 8);
235     tmp_pipe1_4 <= tmp3k1_s4;
236     tmp3k1_s5 := tmp_pipe1_4 + "*" (reg_pipe2_4, b_5) (14 downto 8);
237     tmp_pipe1_5 <= tmp3k1_s5;
238     tmp3k1_s6 := tmp_pipe1_5 + "*" (reg_pipe1_5, b_6) (14 downto 8);
239     tmp_pipe1_6 <= tmp3k1_s6;
240     tmp3k1_s7 := tmp_pipe1_6 + "*" (reg_pipe6, b_7) (14 downto 8);
241     tmp_pipe1_7 <= tmp3k1_s7;
242     tmp3k1_s8 := tmp_pipe1_7 + "*" (reg_pipe2_7, b_8) (14 downto 8);
243     tmp_pipe1_8 <= tmp3k1_s8;
244     tmp3k1_s9 := tmp_pipe1_8 + "*" (reg_pipe1_8, b_9) (14 downto 8);
245     tmp_pipe1_9 <= tmp3k1_s9;
246     tmp3k1_s10 := tmp_pipe1_9 + "*" (reg_pipe9, b_10) (14 downto 8);

247
248
249     tmp3k2_s1 := "*" (x3k2, b_0) (14 downto 8) + "*" (x3k1, b_1) (14 downto 8);
250     tmp_pipe2_1 <= tmp3k2_s1;
251     tmp3k2_s2 := tmp_pipe2_1 + "*" (reg_pipe1, b_2) (14 downto 8);
252     tmp_pipe2_2 <= tmp3k2_s2;

```

```

253     tmp3k2_s3 :=tmp_pipe2_2 +    "*" (reg_pipe2_2, b_3) (14 downto 8);
254     tmp_pipe2_3 <= tmp3k2_s3;
255     tmp3k2_s4 := tmp_pipe2_3 + "*" (reg_pipe1_3, b_4) (14 downto 8);
256     tmp_pipe2_4 <= tmp3k2_s4;
257     tmp3k2_s5 := tmp_pipe2_4 + "*" (reg_pipe4, b_5) (14 downto 8);
258     tmp_pipe2_5 <= tmp3k2_s5;
259     tmp3k2_s6 := tmp_pipe2_5 + "*" (reg_pipe2_5, b_6) (14 downto 8);
260     tmp_pipe2_6 <= tmp3k2_s6;
261     tmp3k2_s7 := tmp_pipe2_6 + "*" (reg_pipe1_6, b_7) (14 downto 8);
262     tmp_pipe2_7 <= tmp3k2_s7;
263     tmp3k2_s8 := tmp_pipe2_7 + "*" (reg_pipe7, b_8) (14 downto 8);
264     tmp_pipe2_8 <= tmp3k2_s8;
265     tmp3k2_s9 := tmp_pipe2_8 + "*" (reg_pipe2_8, b_9) (14 downto 8);
266     tmp_pipe2_9 <= tmp3k2_s9;
267     tmp3k2_s10 := tmp_pipe2_9 + "*" (reg_pipe1_9, b_10) (14 downto 8);
268
269     if v_in = '1' then
270         DOUT3k  <= std_logic_vector(tmp3k_s10) & '0';
271         DOUT3k1 <= std_logic_vector(tmp3k1_s10) & '0';
272         DOUT3k2 <= std_logic_vector(tmp3k2_s10) & '0';
273     end if;
274
275 end if;
276 end process;
277 end beh;
```

Listing 4.11: **myfir_unfolded_pipe_5stages.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity myfir_unfolded_pipe_5stages is
6 generic
7 (NBIT : integer := 8);
8 port(
9     VIN, RST_n, CLK
10    : in std_logic;
11    B0
12    : in std_logic_vector(NBIT - 1 downto 0);
13    B1
14    : in std_logic_vector(NBIT - 1 downto 0);
15    B2
16    : in std_logic_vector(NBIT - 1 downto 0);
17    B3
18    : in std_logic_vector(NBIT - 1 downto 0);
19    B4
20    : in std_logic_vector(NBIT - 1 downto 0);
21    B5
22    : in std_logic_vector(NBIT - 1 downto 0);
23    B6
24    : in std_logic_vector(NBIT - 1 downto 0);
25    B7
26    : in std_logic_vector(NBIT - 1 downto 0);
27    B8
28    : in std_logic_vector(NBIT - 1 downto 0);
29    B9
30    : in std_logic_vector(NBIT - 1 downto 0);
31    B10
32    : in std_logic_vector(NBIT - 1 downto 0);
33    DIN3k,DIN3k1,DIN3k2
34    : in std_logic_vector(NBIT - 1 downto 0);
35    DOUT3k, DOUT3k1,DOUT3k2
36    : out std_logic_vector(NBIT - 1 downto 0);
37    VOUT
38    : out std_logic);
39
40 end myfir_unfolded_pipe_5stages;
41
42 architecture beh of myfir_unfolded_pipe_5stages is
43
44 signal x3k2, x3k1, x3k, x3k_1, x3k_2, x3k_3, x3k_4, x3k_5, x3k_6, x3k_7, x3k_8, x3k_9,
45 x3k_10 : signed(NBIT - 1 downto 0);
46 signal b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_10 : signed(NBIT - 1 downto
```

```

0);

30  signal v_in, v_in_pipe1, v_in_pipe2, v_in_pipe3, v_in_pipe4 : std_logic;
31  signal reg_pipe1, reg_pipe2, reg_pipe3, reg_pipe4,
32      reg_pipe1_1, reg_pipe1_2, reg_pipe1_3, reg_pipe1_4,
33      reg_pipe2_1, reg_pipe2_2, reg_pipe2_3, reg_pipe2_4 : signed(NBIT - 1 downto 0);
34  signal tmp_pipe1, tmp_pipe2, tmp_pipe3, tmp_pipe4,
35      tmp_pipe1_1, tmp_pipe1_2, tmp_pipe1_3, tmp_pipe1_4,
36      tmp_pipe2_1, tmp_pipe2_2, tmp_pipe2_3, tmp_pipe2_4 : signed(NBIT - 2 downto 0);
37
38 begin
39
40 process (CLK, RST_n)
41
42 variable tmp3k_s1, tmp3k_s2, tmp3k_s3, tmp3k_s4, tmp3k_s5,
43     tmp3k1_s1, tmp3k1_s2, tmp3k1_s3, tmp3k1_s4, tmp3k1_s5,
44     tmp3k2_s1, tmp3k2_s2, tmp3k2_s3, tmp3k2_s4, tmp3k2_s5 : signed(NBIT - 2
downto 0);
45
46 begin
47     if RST_n = '0' then
48         x3k2      <= (others => '0');
49         x3k1      <= (others => '0');
50         x3k       <= (others => '0');
51         x3k_1    <= (others => '0');
52         x3k_2    <= (others => '0');
53         x3k_3    <= (others => '0');
54         x3k_4    <= (others => '0');
55         x3k_5    <= (others => '0');
56         x3k_6    <= (others => '0');
57         x3k_7    <= (others => '0');
58         x3k_8    <= (others => '0');
59         x3k_9    <= (others => '0');
60         x3k_10   <= (others => '0');
61         reg_pipe1 <= (others => '0'); reg_pipe1_1 <= (others => '0'); reg_pipe2_1 <=
(others => '0');
62         reg_pipe2 <= (others => '0'); reg_pipe1_2 <= (others => '0'); reg_pipe2_2 <=
(others => '0');
63         reg_pipe3 <= (others => '0'); reg_pipe1_3 <= (others => '0'); reg_pipe2_3 <=
(others => '0');
64         reg_pipe4 <= (others => '0'); reg_pipe1_4 <= (others => '0'); reg_pipe2_4 <=
(others => '0');
65
66         tmp_pipe1 <= (others => '0'); tmp_pipe1_1 <= (others => '0'); tmp_pipe2_1 <=
(others => '0');
67         tmp_pipe2 <= (others => '0'); tmp_pipe1_2 <= (others => '0'); tmp_pipe2_2 <=
(others => '0');
68         tmp_pipe3 <= (others => '0'); tmp_pipe1_3 <= (others => '0'); tmp_pipe2_3 <=
(others => '0');
69         tmp_pipe4 <= (others => '0'); tmp_pipe1_4 <= (others => '0'); tmp_pipe2_4 <=
(others => '0');
70         b_0      <= (others => '0');
71         b_1      <= (others => '0');
72         b_2      <= (others => '0');
73         b_3      <= (others => '0');
74         b_4      <= (others => '0');
75         b_5      <= (others => '0');
76         b_6      <= (others => '0');
77         b_7      <= (others => '0');
78         b_8      <= (others => '0');
79         b_9      <= (others => '0');
80         b_10     <= (others => '0');

```

```

81      v_in <= '0';
82      v_in_pipe1 <= '0'; v_in_pipe2 <= '0'; v_in_pipe3 <= '0'; v_in_pipe4 <= '0';
83      VOUT <= '0';
84      DOUT3k <= (others => '0');
85      DOUT3k1 <= (others => '0');
86      DOUT3k2 <= (others => '0');
87
88      elsif CLK'event and CLK = '1' then
89          b_0 <= signed(B0);
90          b_1 <= signed(B1);
91          b_2 <= signed(B2);
92          b_3 <= signed(B3);
93          b_4 <= signed(B4);
94          b_5 <= signed(B5);
95          b_6 <= signed(B6);
96          b_7 <= signed(B7);
97          b_8 <= signed(B8);
98          b_9 <= signed(B9);
99          b_10 <= signed(B10);
100         v_in_pipe1 <= VIN;
101         v_in_pipe2 <= v_in_pipe1;
102         v_in_pipe3 <= v_in_pipe2;
103         v_in_pipe4 <= v_in_pipe3;
104         v_in <= v_in_pipe4;
105         VOUT <= v_in;
106
107         reg_pipe1 <= x3k;
108         reg_pipe1_1 <= x3k_2;
109         reg_pipe2_1 <= x3k_1;
110
111         reg_pipe2 <= x3k_3;
112         reg_pipe1_2 <= reg_pipe1_1;
113         reg_pipe2_2 <= x3k_4;
114
115         reg_pipe3 <= x3k_6;
116         reg_pipe1_3 <= x3k_5;
117         reg_pipe2_3 <= reg_pipe2_2;
118
119         reg_pipe4 <= reg_pipe3;
120         reg_pipe1_4 <= x3k_8;
121         reg_pipe2_4 <= x3k_7;
122
123         if VIN = '1' then
124             x3k <= signed(DIN3k);
125             x3k1 <= signed(DIN3k1);
126             x3k2 <= signed(DIN3k2);
127
128             x3k_1 <= x3k2;
129             x3k_2 <= x3k1;
130         end if;
131
132         if v_in_pipe1 = '1' then
133             x3k_3 <= reg_pipe1;
134             x3k_4 <= reg_pipe2_1;
135         end if;
136
137         if v_in_pipe2 = '1' then
138             x3k_6 <= reg_pipe2;
139             x3k_5 <= reg_pipe1_2;
140         end if;
141
142         if v_in_pipe3 = '1' then

```

```

143     x3k_8 <= reg_pipe1_3;
144     x3k_7 <= reg_pipe2_3;
145 end if;
146
147 if v_in_pipe4 = '1' then
148     x3k_9 <= reg_pipe4;
149     x3k_10 <= reg_pipe2_4;
150 end if;
-- eseguo operazioni troncando moltiplicazione (THD circa -30dB)
151 tmp3k_s1 := "*" (x3k, b_0) (14 downto 8) + "*" (x3k_1, b_1) (14 downto 8) +
152             "*" (x3k_2, b_2) (14 downto 8);
153 tmp_pipe1 <= tmp3k_s1;
154 tmp3k_s2 := tmp_pipe1 + "*" (x3k_3, b_3) (14 downto 8) +
155             "*" (x3k_4, b_4) (14 downto 8);
156 tmp_pipe2 <= tmp3k_s2;
157 tmp3k_s3 := tmp_pipe2 + "*" (x3k_5, b_5) (14 downto 8) +
158             "*" (x3k_6, b_6) (14 downto 8);
159 tmp_pipe3 <= tmp3k_s3;
160 tmp3k_s4 := tmp_pipe3 + "*" (x3k_7, b_7) (14 downto 8) +
161             "*" (x3k_8, b_8) (14 downto 8);
162 tmp_pipe4 <= tmp3k_s4;
163 tmp3k_s5 := tmp_pipe4 + "*" (x3k_9, b_9) (14 downto 8) +
164             "*" (x3k_10, b_10) (14 downto 8);
165
166
167 tmp3k1_s1 := "*" (x3k1, b_0) (14 downto 8) + "*" (x3k, b_1) (14 downto 8) +
168             "*" (x3k_1, b_2) (14 downto 8);
169 tmp_pipe1_1 <= tmp3k1_s1;
170 tmp3k1_s2 := tmp_pipe1_1 + "*" (reg_pipe1_1, b_3) (14 downto 8) +
171             "*" (x3k_3, b_4) (14 downto 8);
172 tmp_pipe1_2 <= tmp3k1_s2;
173 tmp3k1_s3 := tmp_pipe1_2 + "*" (reg_pipe2_2, b_5) (14 downto 8) +
174             "*" (x3k_5, b_6) (14 downto 8);
175 tmp_pipe1_3 <= tmp3k1_s3;
176 tmp3k1_s4 := tmp_pipe1_3 + "*" (reg_pipe3, b_7) (14 downto 8) +
177             "*" (x3k_7, b_8) (14 downto 8);
178 tmp_pipe1_4 <= tmp3k1_s4;
179 tmp3k1_s5 := tmp_pipe1_4 + "*" (reg_pipe1_4, b_9) (14 downto 8) +
180             "*" (x3k_9, b_10) (14 downto 8);
181
182
183 tmp3k2_s1 := "*" (x3k2, b_0) (14 downto 8) + "*" (x3k1, b_1) (14 downto 8) +
184             "*" (x3k, b_2) (14 downto 8);
185 tmp_pipe2_1 <= tmp3k2_s1;
186 tmp3k2_s2 := tmp_pipe2_1 + "*" (reg_pipe2_1, b_3) (14 downto 8) +
187             "*" (reg_pipe1_1, b_4) (14 downto 8);
188 tmp_pipe2_2 <= tmp3k2_s2;
189 tmp3k2_s3 := tmp_pipe2_2 + "*" (reg_pipe2, b_5) (14 downto 8) +
190             "*" (reg_pipe2_2, b_6) (14 downto 8);
191 tmp_pipe2_3 <= tmp3k2_s3;
192 tmp3k2_s4 := tmp_pipe2_3 + "*" (reg_pipe1_3, b_7) (14 downto 8) +
193             "*" (reg_pipe3, b_8) (14 downto 8);
194 tmp_pipe2_4 <= tmp3k2_s4;
195 tmp3k2_s5 := tmp_pipe2_4 + "*" (reg_pipe2_4, b_9) (14 downto 8) +
196             "*" (reg_pipe1_4, b_10) (14 downto 8);
197
198 if v_in = '1' then
199     DOUT3k <= std_logic_vector(tmp3k_s5) & '0';
200     DOUT3k1 <= std_logic_vector(tmp3k1_s5) & '0';
201     DOUT3k2 <= std_logic_vector(tmp3k2_s5) & '0';
202 end if;
203
204

```

```

205      end if;
206   end process;
207 end beh;
```

Listing 4.12: **myfir_unfolded_pipe_4stages.vhd**

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity myfir_unfolded_pipe_4stages is
6   generic( NBIT : integer := 8);
7   port
8     ( VIN, RST_n, CLK
9       : in std_logic;
10      B0
11      : in std_logic_vector(NBIT - 1 downto 0);
12      B1
13      : in std_logic_vector(NBIT - 1 downto 0);
14      B2
15      : in std_logic_vector(NBIT - 1 downto 0);
16      B3
17      : in std_logic_vector(NBIT - 1 downto 0);
18      B4
19      : in std_logic_vector(NBIT - 1 downto 0);
20      B5
21      : in std_logic_vector(NBIT - 1 downto 0);
22      B6
23      : in std_logic_vector(NBIT - 1 downto 0);
24      B7
25      : in std_logic_vector(NBIT - 1 downto 0);
26      B8
27      : in std_logic_vector(NBIT - 1 downto 0);
28      B9
29      : in std_logic_vector(NBIT - 1 downto 0);
30      B10
31      : in std_logic_vector(NBIT - 1 downto 0);
32      DIN3k,DIN3k1,DIN3k2
33      : in std_logic_vector(NBIT - 1 downto 0);
34      DOUT3k, DOUT3k1,DOUT3k2
35      : out std_logic_vector(NBIT - 1 downto 0);
36      VOUT
37      : out std_logic
38    );
39 end myfir_unfolded_pipe_4stages;
40
41 architecture beh of myfir_unfolded_pipe_4stages is
42
43   signal x3k2, x3k1, x3k, x3k_1, x3k_2, x3k_3, x3k_4, x3k_5, x3k_6, x3k_7, x3k_8, x3k_9,
44   x3k_10 : signed(NBIT - 1 downto 0);
45   signal b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_10 : signed(NBIT - 1 downto 0)
46   ;
47
48   signal v_in, v_in_pipe1, v_in_pipe2, v_in_pipe3 : std_logic;
49   signal reg_pipe1, reg_pipe2, reg_pipe3, reg_pipe1_1, reg_pipe1_2, reg_pipe1_3,
50     reg_pipe2_1, reg_pipe2_2, reg_pipe2_3 : signed(NBIT - 1 downto 0);
51   signal tmp_pipe1, tmp_pipe2, tmp_pipe3, tmp_pipe1_1, tmp_pipe1_2, tmp_pipe1_3,
52     tmp_pipe2_1, tmp_pipe2_2, tmp_pipe2_3 : signed(NBIT - 2 downto 0);
53
54 begin
55
56   process (CLK, RST_n)
57
58     variable tmp3k_s1, tmp3k_s2, tmp3k_s3, tmp3k_s4,
59       tmp3k1_s1, tmp3k1_s2, tmp3k1_s3, tmp3k1_s4,
60       tmp3k2_s1, tmp3k2_s2, tmp3k2_s3, tmp3k2_s4 : signed(NBIT - 2 downto 0);
61
62   begin
63     if RST_n = '0' then
64       x3k2      <= (others => '0');
65       x3k1      <= (others => '0');
66       x3k       <= (others => '0');
67       x3k_1     <= (others => '0');
```

```

48      x3k_2      <= (others => '0');
49      x3k_3      <= (others => '0');
50      x3k_4      <= (others => '0');
51      x3k_5      <= (others => '0');
52      x3k_6      <= (others => '0');
53      x3k_7      <= (others => '0');
54      x3k_8      <= (others => '0');
55      x3k_9      <= (others => '0');
56      x3k_10     <= (others => '0');
57      reg_pipe1   <= (others => '0'); reg_pipe1_1 <= (others => '0'); reg_pipe2_1 <= (others => '0');
58      reg_pipe2   <= (others => '0'); reg_pipe1_2 <= (others => '0'); reg_pipe2_2 <= (others => '0');
59      reg_pipe3   <= (others => '0'); reg_pipe1_3 <= (others => '0'); reg_pipe2_3 <= (others => '0');
60      tmp_pipe1   <= (others => '0'); tmp_pipe1_1 <= (others => '0'); tmp_pipe2_1 <= (others => '0');
61      tmp_pipe2   <= (others => '0'); tmp_pipe1_2 <= (others => '0'); tmp_pipe2_2 <= (others => '0');
62      tmp_pipe3   <= (others => '0'); tmp_pipe1_3 <= (others => '0'); tmp_pipe2_3 <= (others => '0');
63      b_0         <= (others => '0');
64      b_1         <= (others => '0');
65      b_2         <= (others => '0');
66      b_3         <= (others => '0');
67      b_4         <= (others => '0');
68      b_5         <= (others => '0');
69      b_6         <= (others => '0');
70      b_7         <= (others => '0');
71      b_8         <= (others => '0');
72      b_9         <= (others => '0');
73      b_10        <= (others => '0');
74      v_in        <= '0';
75      v_in_pipe1  <= '0'; v_in_pipe2 <= '0'; v_in_pipe3 <= '0';
76      VOUT        <= '0';
77      DOUT3k     <= (others => '0');
78      DOUT3k1    <= (others => '0');
79      DOUT3k2    <= (others => '0');

80
81      elsif CLK'event and CLK = '1' then
82          v_in_pipe1 <= VIN;
83          b_0         <= signed(B0);
84          b_1         <= signed(B1);
85          b_2         <= signed(B2);
86          b_3         <= signed(B3);
87          b_4         <= signed(B4);
88          b_5         <= signed(B5);
89          b_6         <= signed(B6);
90          b_7         <= signed(B7);
91          b_8         <= signed(B8);
92          b_9         <= signed(B9);
93          b_10        <= signed(B10);
94          v_in_pipe2 <= v_in_pipe1;
95          v_in_pipe3 <= v_in_pipe2;
96          v_in        <= v_in_pipe3;
97          VOUT        <= v_in;

98          reg_pipe1   <= x3k;
99          reg_pipe1_1 <= x3k_2;
100         reg_pipe2_1 <= x3k_1;
101
102         reg_pipe2   <= x3k_3;

```

```

104    reg_pipe1_2 <= x3k_5;
105    reg_pipe2_2 <= x3k_4;
106
107    reg_pipe3 <= x3k_6;
108    reg_pipe1_3 <= x3k_8;
109    reg_pipe2_3 <= x3k_7;
110    if VIN = '1' then
111        x3k <= signed(DIN3k);
112        x3k1 <= signed(DIN3k1);
113        x3k2 <= signed(DIN3k2);
114
115        x3k_1 <= x3k2;
116        x3k_2 <= x3k1;
117    end if;
118    if v_in_pipe1 = '1' then
119        x3k_3 <= reg_pipe1;
120        x3k_5 <= reg_pipe1_1;
121        x3k_4 <= reg_pipe2_1;
122    end if;
123
124    if v_in_pipe2 = '1' then
125        x3k_6 <= reg_pipe2;
126        x3k_8 <= reg_pipe1_2;
127        x3k_7 <= reg_pipe2_2;
128    end if;
129
130    if v_in_pipe3 = '1' then
131        x3k_9 <= reg_pipe3;
132        x3k_10 <= reg_pipe2_3;
133    end if;
134    -- eseguo operazioni troncando moltiplicazione (THD circa -30dB)
135    tmp3k_s1 := "*" (x3k, b_0) (14 downto 8) + "*" (x3k_1, b_1) (14 downto 8) +
136        "*" (x3k_2, b_2) (14 downto 8);
137    tmp_pipe1 <= tmp3k_s1;
138    tmp3k_s2 := tmp_pipe1 + "*" (x3k_3, b_3) (14 downto 8) +
139        "*" (x3k_4, b_4) (14 downto 8) + "*" (x3k_5, b_5) (14 downto 8);
140    tmp_pipe2 <= tmp3k_s2;
141    tmp3k_s3 := tmp_pipe2 + "*" (x3k_6, b_6) (14 downto 8) +
142        "*" (x3k_7, b_7) (14 downto 8) + "*" (x3k_8, b_8) (14 downto 8);
143    tmp_pipe3 <= tmp3k_s3;
144    tmp3k_s4 := tmp_pipe3 + "*" (x3k_9, b_9) (14 downto 8) +
145        "*" (x3k_10, b_10) (14 downto 8);
146
147    tmp3k1_s1 := "*" (x3k1, b_0) (14 downto 8) + "*" (x3k, b_1) (14 downto 8) +
148        "*" (x3k_1, b_2) (14 downto 8);
149    tmp_pipe1_1 <= tmp3k1_s1;
150    tmp3k1_s2 := tmp_pipe1_1 + "*" (reg_pipe1_1, b_3) (14 downto 8) +
151        "*" (x3k_3, b_4) (14 downto 8) + "*" (x3k_4, b_5) (14 downto 8);
152    tmp_pipe1_2 <= tmp3k1_s2;
153    tmp3k1_s3 := tmp_pipe1_2 + "*" (reg_pipe1_2, b_6) (14 downto 8) +
154        "*" (x3k_6, b_7) (14 downto 8) + "*" (x3k_7, b_8) (14 downto 8);
155    tmp_pipe1_3 <= tmp3k1_s3;
156    tmp3k1_s4 := tmp_pipe1_3 + "*" (reg_pipe1_3, b_9) (14 downto 8) +
157        "*" (x3k_9, b_10) (14 downto 8);
158
159    tmp3k2_s1 := "*" (x3k2, b_0) (14 downto 8) + "*" (x3k1, b_1) (14 downto 8) +
160        "*" (x3k, b_2) (14 downto 8);
161    tmp_pipe2_1 <= tmp3k2_s1;
162    tmp3k2_s2 := tmp_pipe2_1 + "*" (reg_pipe2_1, b_3) (14 downto 8) +
163        "*" (reg_pipe1_1, b_4) (14 downto 8) + "*" (x3k_3, b_5) (14 downto 8);
164    tmp_pipe2_2 <= tmp3k2_s2;
165    tmp3k2_s3 := tmp_pipe2_2 + "*" (reg_pipe2_2, b_6) (14 downto 8) +

```

```

166      "*" (reg_pipe1_2, b_7) (14 downto 8) + "*" (x3k_6, b_8) (14 downto 8);
167      tmp_pipe2_3 <= tmp3k2_s3;
168      tmp3k2_s4 := tmp_pipe2_3 + "*" (reg_pipe2_3, b_9) (14 downto 8) +
169      "*" (reg_pipe1_3, b_10) (14 downto 8);
170
171      if v_in = '1' then
172          DOUT3k <= std_logic_vector(tmp3k_s4) & '0';
173          DOUT3k1 <= std_logic_vector(tmp3k1_s4) & '0';
174          DOUT3k2 <= std_logic_vector(tmp3k2_s4) & '0';
175      end if;
176
177      end if;
178  end process;
179 end beh;
```

Listing 4.13: tb_filter_unfolded.v

```

1 //`timescale 1ns
2
3 module tb_filter_unfolded ();
4
5     parameter NBIT = 8;
6
7     wire CLK_i;
8     wire RST_n_i;
9     wire [7:0] DIN3k_i;
10    wire [7:0] DIN3k1_i;
11    wire [7:0] DIN3k2_i;
12    wire VIN_i;
13    wire [7:0] B0_i;
14    wire [7:0] B1_i;
15    wire [7:0] B2_i;
16    wire [7:0] B3_i;
17    wire [7:0] B4_i;
18    wire [7:0] B5_i;
19    wire [7:0] B6_i;
20    wire [7:0] B7_i;
21    wire [7:0] B8_i;
22    wire [7:0] B9_i;
23    wire [7:0] B10_i;
24    wire [NBIT-1:0] DOUT3k_i;
25    wire [NBIT-1:0] DOUT3k1_i;
26    wire [NBIT-1:0] DOUT3k2_i;
27    wire VOUT_i;
28    wire END_SIM_i;
29
30    clk_gen CG(.END_SIM(END_SIM_i),
31                .CLK(CLK_i),
32                .RST_n(RST_n_i));
33
34    data_maker #(NBIT(NBIT)) SM(.CLK(CLK_i),
35                            .RST_n(RST_n_i),
36                            .VOUT(VIN_i),
37                            .DOUT3k(DIN3k_i),
38                            .DOUT3k1(DIN3k1_i),
39                            .DOUT3k2(DIN3k2_i),
40                            .B0(B0_i),
41                            .B1(B1_i),
```

```

42      .B2(B2_i),
43      .B3(B3_i),
44      .B4(B4_i),
45      .B5(B5_i),
46      .B6(B6_i),
47      .B7(B7_i),
48      .B8(B8_i),
49      .B9(B9_i),
50      .B10(B10_i),
51      .END_SIM(END_SIM_i));
52
53 myfir_unfolded_pipe_10stages UUT(.CLK(CLK_i),
54     .RST_n(RST_n_i),
55     .DIN3k(DIN3k_i),
56     .DIN3k1(DIN3k1_i),
57     .DIN3k2(DIN3k2_i),
58     .VIN(VIN_i),
59     .B0(B0_i),
60     .B1(B1_i),
61     .B2(B2_i),
62     .B3(B3_i),
63     .B4(B4_i),
64     .B5(B5_i),
65     .B6(B6_i),
66     .B7(B7_i),
67     .B8(B8_i),
68     .B9(B9_i),
69     .B10(B10_i),
70     .DOUT3k(DOUT3k_i),
71     .DOUT3k1(DOUT3k1_i),
72     .DOUT3k2(DOUT3k2_i),
73     .VOUT(VOUT_i));
74
75 data_sink #(.NBIT(NBIT)) DS(.CLK(CLK_i),
76     .RST_n(RST_n_i),
77     .VIN(VOUT_i),
78     .DIN3k(DOUT3k_i),
79     .DIN3k1(DOUT3k1_i),
80     .DIN3k2(DOUT3k2_i));
81
82 endmodule
83
84

```

4.3 Text files

Listing 4.14: **samples.txt**

```

1 0
2 71
3 -24
4 71
5 98
6 0
7 98
8 71
9 -24

```

```
10 71
11 0
12 -72
13 23
14 -72
15 -99
16 0
17 -99
18 -72
19 23
20 -72
21 -1
22 71
23 -24
24 71
25 98
26 0
27 98
28 71
29 -24
30 71
31 -1
32 -72
33 23
34 -72
35 -99
36 0
37 -99
38 -72
39 23
40 -72
41 -1
42 71
43 -24
44 71
45 98
46 0
47 98
48 71
49 -24
50 71
51 0
52 -72
53 23
54 -72
55 -99
56 0
57 -99
58 -72
59 23
60 -72
61 0
62 71
63 -24
64 71
65 98
66 0
67 98
68 71
69 -24
70 71
71 0
```

```
72 -72
73 23
74 -72
75 -99
76 0
77 -99
78 -72
79 23
80 -72
81 -1
82 71
83 -24
84 71
85 98
86 0
87 98
88 71
89 -24
90 71
91 -1
92 -72
93 23
94 -72
95 -99
96 0
97 -99
98 -72
99 23
100 -72
101 -1
102 71
103 -24
104 71
105 98
106 0
107 98
108 71
109 -24
110 71
111 0
112 -72
113 23
114 -72
115 -99
116 0
117 -99
118 -72
119 23
120 -72
121 0
122 71
123 -24
124 71
125 98
126 0
127 98
128 71
129 -24
130 71
131 0
132 -72
133 23
```

134 -72
135 -99
136 0
137 -99
138 -72
139 23
140 -72
141 -1
142 71
143 -24
144 71
145 98
146 0
147 98
148 71
149 -24
150 71
151 0
152 -72
153 23
154 -72
155 -99
156 0
157 -99
158 -72
159 23
160 -72
161 -1
162 71
163 -24
164 71
165 98
166 0
167 98
168 71
169 -24
170 71
171 -1
172 -72
173 23
174 -72
175 -99
176 0
177 -99
178 -72
179 23
180 -72
181 0
182 71
183 -24
184 71
185 98
186 0
187 98
188 71
189 -24
190 71
191 0
192 -72
193 23
194 -72
195 -99

```
196 0
197 -99
198 -72
199 23
200 -72
201 -1
```

Listing 4.15: resultsm.txt

```
1 0
2 -1
3 -1
4 -3
5 3
6 14
7 22
8 34
9 48
10 58
11 62
12 58
13 49
14 36
15 18
16 0
17 -19
18 -37
19 -50
20 -59
21 -63
22 -59
23 -50
24 -37
25 -19
26 0
27 18
28 36
29 49
30 58
31 62
32 58
33 49
34 36
35 18
36 0
37 -19
38 -37
39 -50
40 -59
41 -63
42 -59
43 -50
44 -37
45 -19
46 -1
47 18
48 36
49 49
```

```
50 58
51 62
52 58
53 49
54 36
55 18
56 0
57 -19
58 -37
59 -50
60 -59
61 -63
62 -59
63 -50
64 -37
65 -19
66 0
67 18
68 36
69 49
70 58
71 62
72 58
73 49
74 36
75 18
76 0
77 -19
78 -37
79 -50
80 -59
81 -63
82 -59
83 -50
84 -37
85 -19
86 0
87 18
88 36
89 49
90 58
91 62
92 58
93 49
94 36
95 18
96 -1
97 -19
98 -37
99 -50
100 -59
101 -63
102 -59
103 -50
104 -37
105 -19
106 -1
107 18
108 36
109 49
110 58
111 62
```

112 58
113 49
114 36
115 18
116 0
117 -19
118 -37
119 -50
120 -59
121 -63
122 -59
123 -50
124 -37
125 -19
126 0
127 18
128 36
129 49
130 58
131 62
132 58
133 49
134 36
135 18
136 -1
137 -19
138 -37
139 -50
140 -59
141 -63
142 -59
143 -50
144 -37
145 -19
146 -1
147 18
148 36
149 49
150 58
151 62
152 58
153 49
154 36
155 18
156 0
157 -19
158 -37
159 -50
160 -59
161 -63
162 -59
163 -50
164 -37
165 -19
166 -1
167 18
168 36
169 49
170 58
171 62
172 58
173 49

```
174 36
175 18
176 -1
177 -19
178 -37
179 -50
180 -59
181 -63
182 -59
183 -50
184 -37
185 -19
186 0
187 18
188 36
189 49
190 58
191 62
192 58
193 49
194 36
195 18
196 0
197 -19
198 -37
199 -50
200 -59
201 -63
```

Listing 4.16: **resultsc.txt**

```
1 0
2 -2
3 -2
4 -6
5 0
6 10
7 16
8 28
9 42
10 50
11 56
12 50
13 40
14 28
15 8
16 -8
17 -28
18 -46
19 -60
20 -68
21 -72
22 -68
23 -60
24 -48
25 -30
26 -10
27 6
```

28 26
29 40
30 50
31 56
32 50
33 40
34 26
35 6
36 -10
37 -30
38 -48
39 -60
40 -68
41 -72
42 -68
43 -60
44 -48
45 -30
46 -10
47 6
48 26
49 40
50 50
51 56
52 50
53 40
54 28
55 8
56 -8
57 -28
58 -46
59 -60
60 -68
61 -72
62 -68
63 -60
64 -46
65 -28
66 -8
67 8
68 28
69 40
70 50
71 56
72 50
73 40
74 28
75 8
76 -8
77 -28
78 -46
79 -60
80 -68
81 -72
82 -68
83 -60
84 -48
85 -30
86 -10
87 6
88 26
89 40

90 50
91 56
92 50
93 40
94 26
95 6
96 -10
97 -30
98 -48
99 -60
100 -68
101 -72
102 -68
103 -60
104 -48
105 -30
106 -10
107 6
108 26
109 40
110 50
111 56
112 50
113 40
114 28
115 8
116 -8
117 -28
118 -46
119 -60
120 -68
121 -72
122 -68
123 -60
124 -46
125 -28
126 -8
127 8
128 28
129 40
130 50
131 56
132 50
133 40
134 28
135 8
136 -8
137 -28
138 -46
139 -60
140 -68
141 -72
142 -68
143 -60
144 -48
145 -30
146 -10
147 6
148 26
149 40
150 50
151 56

```
152 50
153 40
154 28
155 8
156 -8
157 -28
158 -46
159 -60
160 -68
161 -72
162 -68
163 -60
164 -48
165 -30
166 -10
167 6
168 26
169 40
170 50
171 56
172 50
173 40
174 26
175 6
176 -10
177 -30
178 -48
179 -60
180 -68
181 -72
182 -68
183 -60
184 -46
185 -28
186 -8
187 8
188 28
189 40
190 50
191 56
192 50
193 40
194 28
195 8
196 -8
197 -28
198 -46
199 -60
200 -68
201 -72
```

Listing 4.17: **results_hdl.txt**

```
1 0
2 0
3 -2
4 -2
5 -6
```

```
6 0
7 10
8 16
9 28
10 42
11 50
12 56
13 50
14 40
15 28
16 8
17 -8
18 -28
19 -46
20 -60
21 -68
22 -72
23 -68
24 -60
25 -48
26 -30
27 -10
28 6
29 26
30 40
31 50
32 56
33 50
34 40
35 26
36 6
37 -10
38 -30
39 -48
40 -60
41 -68
42 -72
43 -68
44 -60
45 -48
46 -30
47 -10
48 6
49 26
50 40
51 50
52 56
53 50
54 40
55 28
56 8
57 -8
58 -28
59 -46
60 -60
61 -68
62 -72
63 -68
64 -60
65 -46
66 -28
67 -8
```

68 8
69 28
70 40
71 50
72 56
73 50
74 40
75 28
76 8
77 -8
78 -28
79 -46
80 -60
81 -68
82 -72
83 -68
84 -60
85 -48
86 -30
87 -10
88 6
89 26
90 40
91 50
92 56
93 50
94 40
95 26
96 6
97 -10
98 -30
99 -48
100 -60
101 -68
102 -72
103 -68
104 -60
105 -48
106 -30
107 -10
108 6
109 26
110 40
111 50
112 56
113 50
114 40
115 28
116 8
117 -8
118 -28
119 -46
120 -60
121 -68
122 -72
123 -68
124 -60
125 -46
126 -28
127 -8
128 8
129 28

130 40
131 50
132 56
133 50
134 40
135 28
136 8
137 -8
138 -28
139 -46
140 -60
141 -68
142 -72
143 -68
144 -60
145 -48
146 -30
147 -10
148 6
149 26
150 40
151 50
152 56
153 50
154 40
155 28
156 8
157 -8
158 -28
159 -46
160 -60
161 -68
162 -72
163 -68
164 -60
165 -48
166 -30
167 -10
168 6
169 26
170 40
171 50
172 56
173 50
174 40
175 26
176 6
177 -10
178 -30
179 -48
180 -60
181 -68
182 -72
183 -68
184 -60
185 -46
186 -28
187 -8
188 8
189 28
190 40
191 50

192 56
193 50
194 40
195 28
196 8
197 -8
198 -28
199 -46
200 -60
201 -68