Politecnico di Torino
III Facoltà di Ingegneria

Design and Implementation of a Digital FIR Filter

Master degree in Electrical Engineering

Authors: Group: 41

Nima Kolahimahmoudi, Srinivas Chatrasi

November 21, 2021

Contents

CHAPTER 1

Introduction

This report describes the design and implementation of an 8-bit, 8° order digital FIR Filter in Direct Form.

The filter was developed using two different architectures:

- A basic architecture, realized directly from the Direct-Form of the filter
- An advanced architecture obtained by applying unfolding and pipelining transformation on the previous architecture.

The design of the filter started from the realization of a pseudo-fixed-point reference model in MATLAB and a fixed-point reference model in C, developed starting from the example files provided. From the models, the reference architecture was developed starting from the characteristic difference equation of the FIR Filter and described using VHDL. A simulation phase was then carried out using Modelsim, to verify the correct behaviour of the system. Finally, the design was synthesized, placed, and routed , reporting performances in terms of power consumption and area estimation.

Once the basic architecture of the filter was completed, the development of the advanced architecture was done performing the same design flow described for the basic architecture.

To provide a complete view of the laboratory activities, some of the files produced during the development of the system are included into the appendices of this document.

Github repository

The whole project is uploaded in the Github repository at the following link:
https://github.com/SCHATRASI/LAB1.git

CHAPTER 2

Basic architecture: reference model development

2.1 Main characteristics summary

The main characteristics of the systems to develop are:

- Filter type: 8-bit, 8° order digital Finite Impulse Response (FIR) Filter
- Cut-off frequency at 2 kHz, sampling frequency at 10 kHz
- Total harmonic distortion lower than -30 dB.

2.2 Testing of the filter

The development of the reference model has been carried out realizing and testing two models:

1. A pseudo-fixed-point model using MATLAB
2. A fixed-point C model

2.2.1 MATLAB pseudo-fixed-point model

A first reference model was developed and simulated using the *my_fir_filter.m* script, adapted to our constraint of N = 8 (order of the filter) and nb = 8 (number of bits).
This two values were passed to the *my_fir_design.m* script, which computes the coefficients (represented as integers and quantized) of the transfer function of the filter that will be used both in the MATLAB function filter and then passed as input to the architecture implementation.
The coefficients provided by the function are -1,-2,6,34,51,34,6,-2,-1. The coefficients b_i is the one represented as integers, while b_q is the coefficients normalized with respect to 2^7.

The my_fir_design.m script returns also a graphical representation of the transfer function, that can be seen in Figure 2.1.

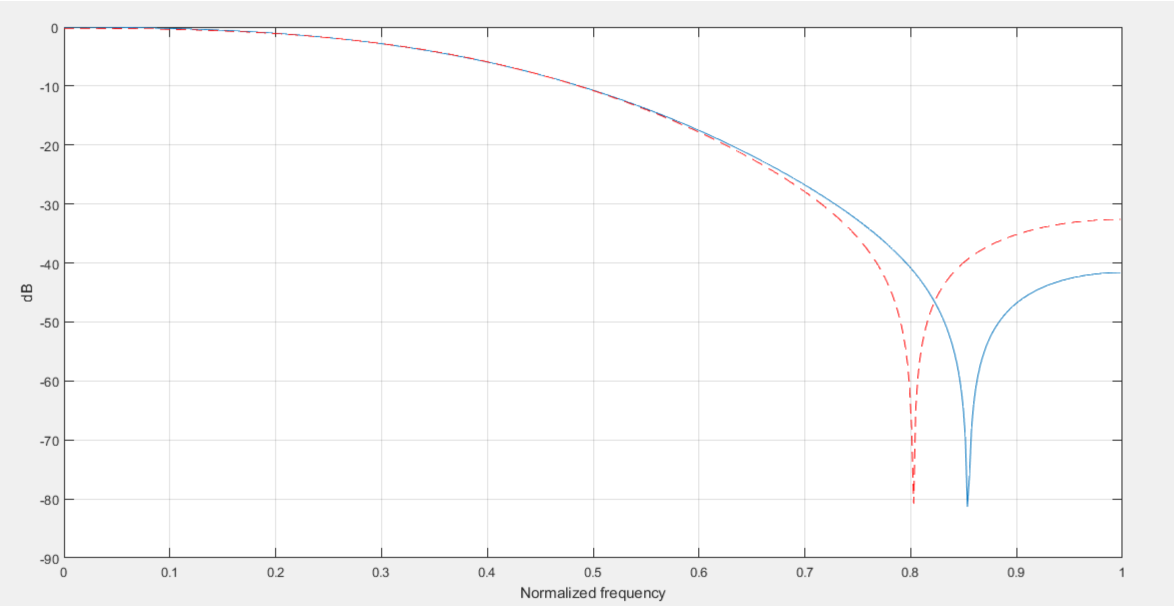

Figure 2.1: Filter transfer function

The *my_fir_filter.m* script applies the filter with the computed coefficients to a signal obtained averaging a sinusoidal wave with $f = 500Hz$ and a second one with $f = 4500Hz$ (out of band). The normalized output wave can be seen in Figure 2.3.

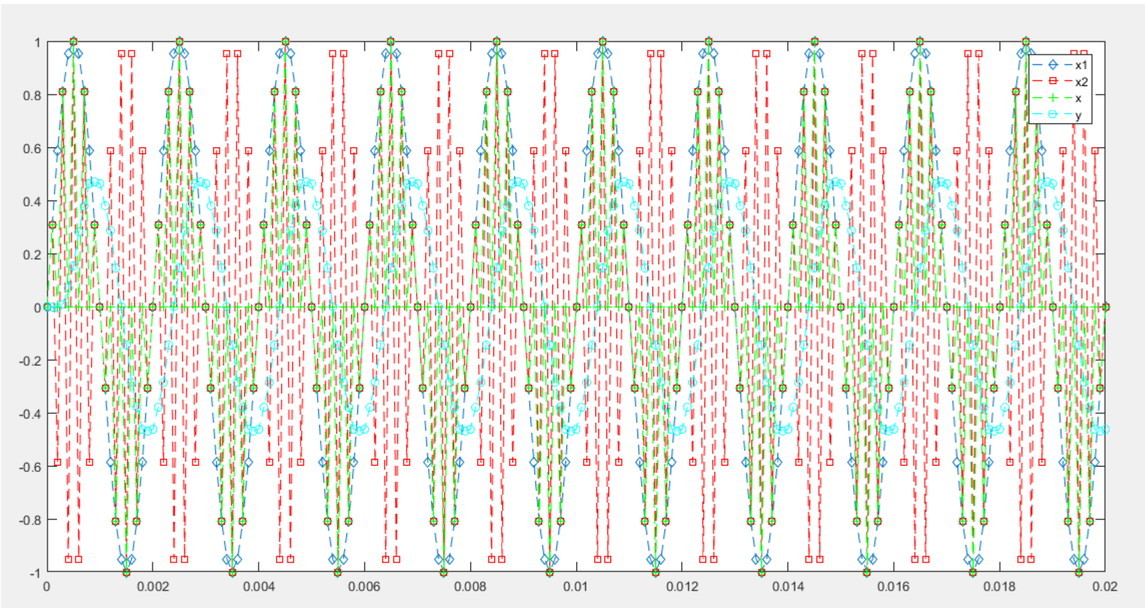

Figure 2.2: Comparison between input and output signals

The samples of the input and output wave were then quantized and saved into two different files, respectively *samples.txt* and *resultsm.txt*.

2.2.2 C fixed-point model

Starting from the example file *myfilterii.c* a C model was written and saved as *myfilter.c*. This model is more specific than the one provided as example and it highly refers to the direct form basic architecture of the filter.

Equation 3 of Section 1.2.2 of the laboratory description document is limited to order 8 and no inclusion of the recursive part:

$$y[n] = b_0{\cdot}x[n]{+}b_1{\cdot}x[n{-}1]{+}b_2{\cdot}x[n{-}2]{+}b_3{\cdot}x[n{-}3]{+}b_4{\cdot}x[n{-}4]{+}b_5{\cdot}x[n{-}5]{+}b_6{\cdot}x[n{-}6]{+}b_7{\cdot}x[n{-}7]{+}b_8{\cdot}x[n{-}8] \tag{2.1}$$

The coefficients b_0, b_1, ... ,b_8 were declared as global constant integer since their value is a characteristic of the filter and never changes during the usage of the system.

The script requires as input two filenames:

- An input file containing the samples of the signal to be filtered
- An output file that will be used to store the data generated by the filter.

The $main()$ reads sequentially all the integer values stored into the input file. For each of them, The function $myfilter$ is called passing the value as input.

In $myfilter$ is described the actual behaviour of the system to be designed. The shift done after every multiplication allows to obtain a final value that is represented on the correct number of bits.

The comparison of the results that we found from the MATLAB and C model can be seen through the following plot:

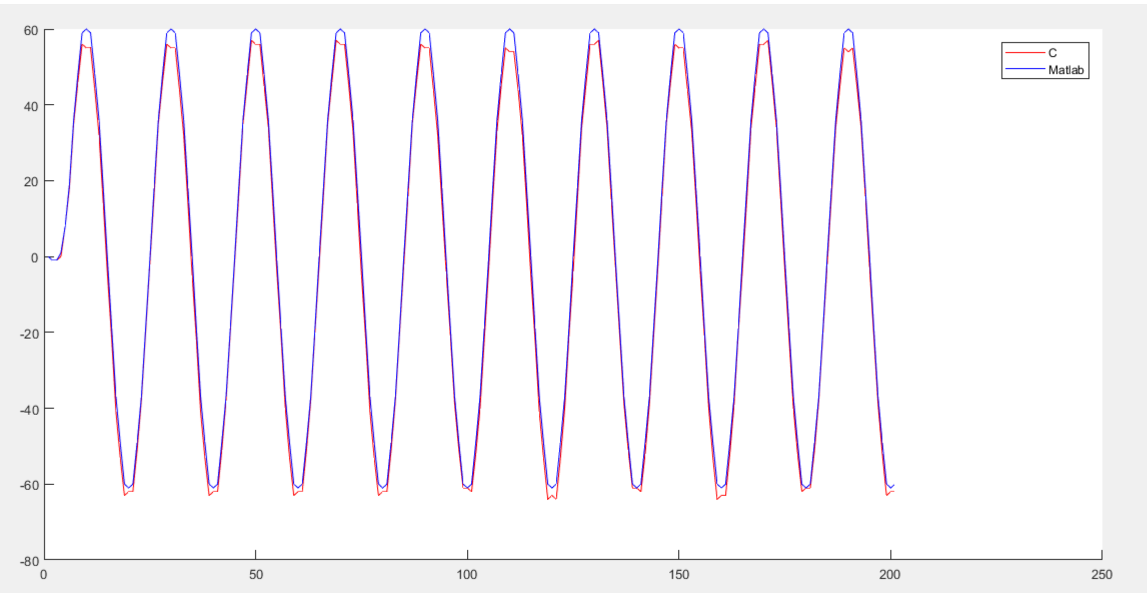

Figure 2.3: Comparison between Outputs of the C and MATLAB codes

CHAPTER 3

Basic architecture: VLSI implementation

3.1 Architecture development

3.1.1 Filter interface

In Figure 3.1 is represented the external interface of the developed system, derived from the one specified by Figure 1.a of laboratory description document.

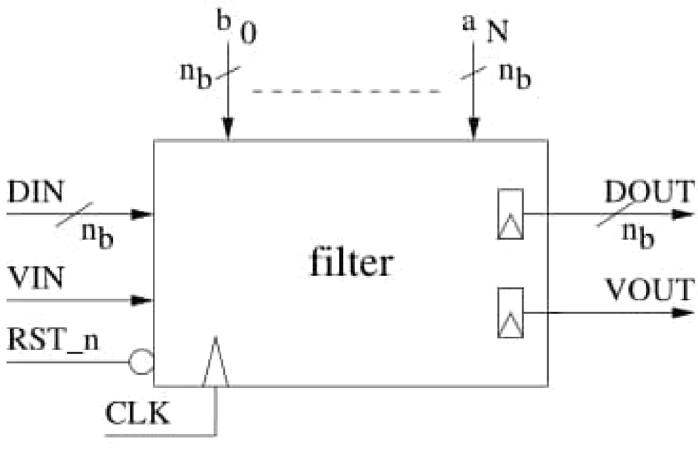

Figure 3.1: External interface of the filter

Whereas nb is of 8 bits as mention previously and the coefficients will from b0 to b8.

3.1.2 Block diagram

From Equation 2.1, it is possible to directly derive the Data-Flow Graph (DFG) of the system in the Direct-Form:.

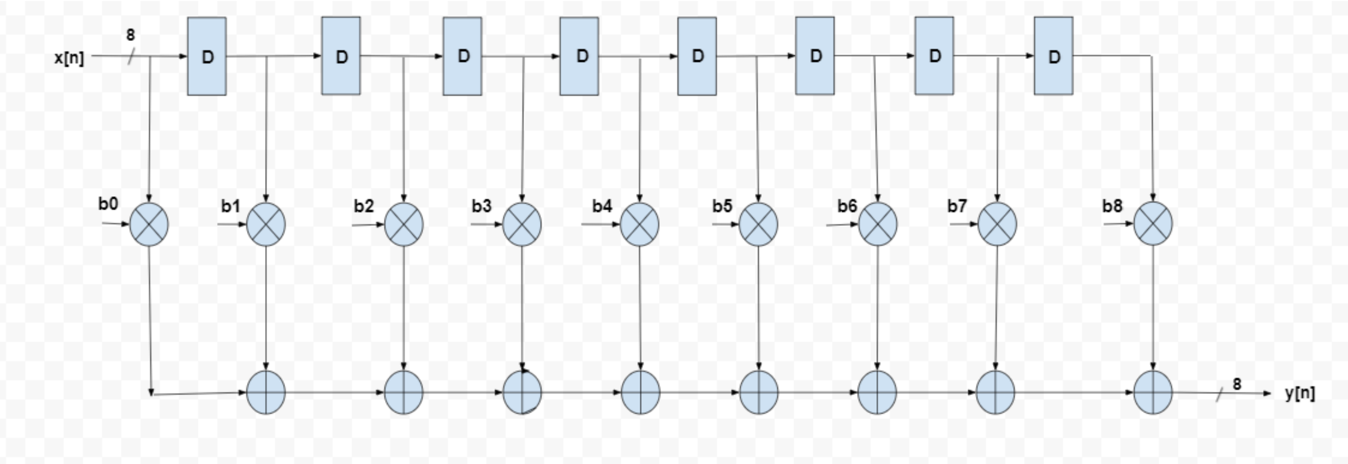

Figure 3.2: FIR Filter DFG - Direct-Form

In Figure 3.2 is also highlighted the critical path of this implementation. Denoting times of multipliers as T_m and adders as T_a, it is possible to conclude that

$$t_{cp} = T_m + 8T_a \tag{3.1}$$

Here we considered the first multiplier with coefficient b0 and all the adders which constitute the critical path for the above mentioned filter, through which we concluded the above critical path delay in the equation.

3.2 VHDL model development

The VHDL description of the FIR filter is reported in Appendix B.1. To summarize, the top level entity is reported here:

```
entity FIR is
      port (
            din:in std_logic_vector(7 downto 0);
            vin:in std_logic;
            rst:in std_logic;
            clk:in std_logic;

            dout:out std_logic_vector(7 downto 0);
            vout:out std_logic
          );
end FIR;
```

Since all data as input as to be read from a register and every output must first be loaded into a register, two registers were added to the system as input and output registers.

In the behaviour part the coefficients were taken of size of 8 bits and also the coefficients were assigned as we obtained these values from the c model.Also we need to have 8 registers to implement the delay from n-1 to n-8 using it. And defined the following signal types and signals that we used in the implementation of the architecture.

```
architecture behavioral of FIR is
type registers is array (7 downto 0) of integer;
type coefficients is array (8 downto 0) of integer;
signal reg: registers;
constant coef : coefficients := (-1,-2,6,34,51,34,6,-2,-1);
signal din_temp : integer :=0;
```

In the process we check for the reset signal , if its '0' then the registers were assigned the zero value and so the "dout" will also be zero. However if the "vin" is '1' during the positive clock edge and no reset signal in action then the filter starts sampling the input and starts the arithmetic calculation as per the architecture arrangement. Here what we need to consider is that as we know that the multiplier result will be of 2n where as the n is the length of the inputs,and we used the temp variable called "prod_temp" to store the multiplier result in 16 bits. But we need the FIR to produce the result of 8 bits so we only consider the 8 bits of MSB of the multiplier result that is "15 downto 7" which is assigned to intermediate variable "prod_shift", which is then fed to the following adder so as to finally obtain the result of desired number of bits from the filter. The vhdl code snippet is as follows:

```
begin
if (rst= '0') then

  for i in 5 downto 0 loop
    reg(i)<= 0;
             dout<= "00000000";
  end loop;
elsif (clk'event and clk='1') then
    if (vin='1') then
                    acc:=0;
```

```
                for i in 7 downto 1 loop
                    reg(i) <= reg(i-1);
                end loop;
                prod := coef(0)*din_temp;
                prod_temp := std_logic_vector(to_signed(prod,16));
               prod_shift := prod_temp(15 downto 7);
               prod := to_integer(signed(prod_shift));
               acc := acc + prod;

    prod := 0;
                for i in 1 to 8 loop
            prod := coef(i)*reg(i-1);
                    prod_temp := std_logic_vector(to_signed(prod,16));
                    prod_shift := prod_temp(15 downto 7);
                    prod := to_integer(signed(prod_shift));
            acc := acc + prod;
                         prod := 0;
                end loop;
                  vout_temp:='1';
                  reg(0) <= din_temp;
     if (vout_temp='1') then
      dout <= std_logic_vector(to_signed(acc,8));
                vout_temp :='0';
     end if;
  end if;
    end if;
    vout<= vout_temp;
end process;
```

3.3 Simulation

3.3.1 Testbench structure

In Figure 3.3 is represented the block diagram of the testbench used to test the system.

It is composed by the following components, described in VHDL:

filter the device under test (DUT).

clk_gen Clock generator for the whole system.

data_maker It generates data to be sent to the device under test by reading each value from the the input file *samples.txt*, generated by MATLAB and used as input by the fixed-point model, too. It also provides the coefficient b_0, b_1,b_3...... and b_8 to the filter. The input signal EN is used to control the simulation from the testbench: this allows to test the functionality of the DIN signal. The output signal END_SIM is raised when the simulation is over.

data_sink It reads data coming from the output of the filter and writes it into an output file, *results_vhd.txt*.

The testbench itself, *tb_fir.v* is written in Verilog and properly connects the component to each other. It also controls the EN and reset signals. At the beginning of the simulation, it sets RST_n

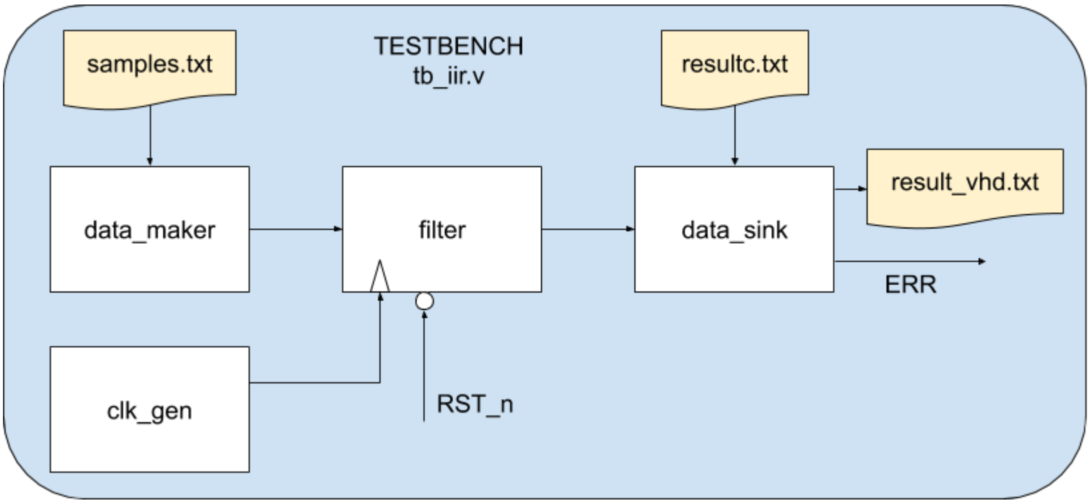

Figure 3.3: Testbench block diagram

to zero forcing to reset the components in the system. Then, the EN signal is raised to start the computation. Setting EN to zero allows to check the correct behaviour of the VIN and VOUT signals, stopping the computation. Raising it again, the computation will start from where it was stopped.

All the files described in this section are included into Appendix C.

3.3.2 Simulation

The simulation was carried out using Modelsim.

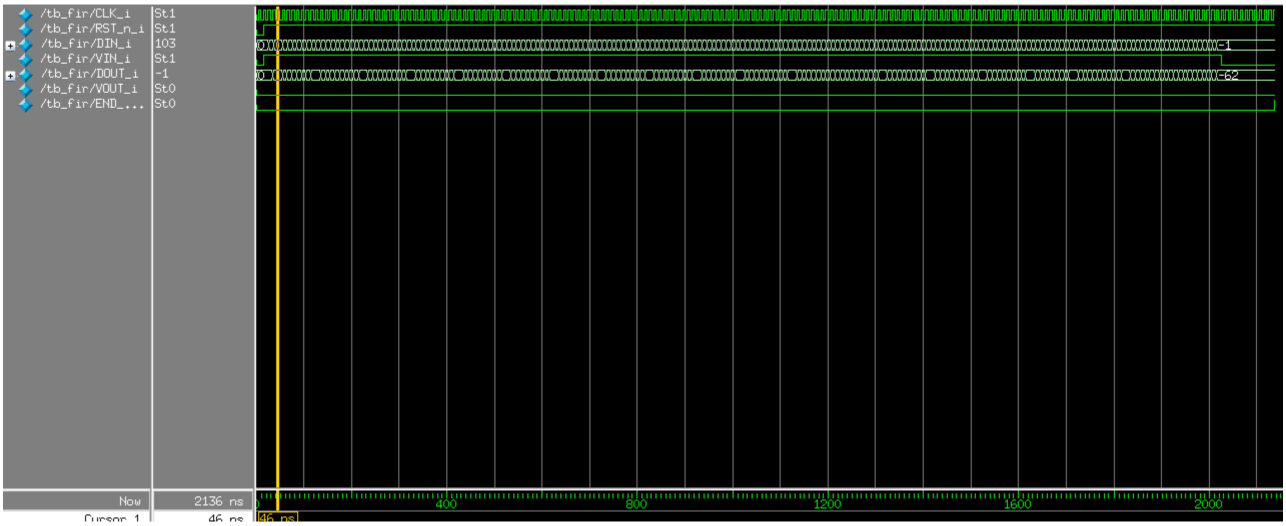

Figure 3.4: Waveforms showing the behaviour of the filter

Figure 3.4 shows the simulation results for 2 us. The input data is sampled and then elaborated in the following clock cycle, as soon as the signal VIN_i goes high, giving the corresponding output result one clock cycle later, from the image we can see that the first output was obtained nearly after 46 ns. And the last sampled input is -1 whereas the the last output is -62. As the simulation ends the END_SIM signal has gone high indicating the completion of the simulation.

3.4 Implementation

3.4.1 Logic synthesis

After verifying the behaviour of the VHDL model, the top-level entity had to be synthesized. The tool that is going to be used to synthesize the components in this project is Synopsys Design Compiler.
In order to correctly launch the Design Compiler, the environment has to be set up into the directory that will be used for logic synthesis. Following the indications given by the documentation, the synthesis was run into the *syn* folder. All the main settings are defined by the file *.synopsys_dc.setup*.
The setup proposed for this lab makes use of the DesignWare library. Thanks to this library the synthesizer detects known basic blocks (e.g. multipliers, adders, . . .). For each of this blocks it can choose different implementations to meet the constraints imposed by the user.
The first step is analyze all the HDL files composing the system, from the smaller component to the top-level entity. This was done with the following commands:

```
analyze -f vhdl -lib WORK ../src/constants.vhd
analyze -f vhdl -lib WORK ../src/fd.vhd
analyze -f vhdl -lib WORK ../src/reg_n.vhd
analyze -f vhdl -lib WORK ../src/filter.vhd
```

In order to preserve RTL names in the netlist to ease the procedure of the power consumption estimation, the following flag is set:

```
set power_preserve_rtl_hie_names true
```

Finally, the design is elaborated.

```
elaborate FIR -arch bhv -lib WORK > elaborate.txt
```

Maximum clock frequency f_M

It was asked to find the maximum frequency at which our design can correctly work. This can be done by finding the clock period that generates a slack equals to 0. In order to determine it, an initial clock signal with period 0 is set: this will force the compiler to optimize the circuit obtaining the best critical path achievable.

```
create_clock -name MY_CLK -period 0.0 CLK
```

By selecting the above mentioned period 0.0 we got the negative slack as mentioned below:

$$slack(t_{clk} = 0) = -1.40ns \tag{3.2}$$

Following the above steps but now with a clock period of -1.40 ns, however, gave again a negative slack of . For this reason, a new synthesis was done (analyzing, elaborating and compiling again with clock period equal to 0) setting the a timing constraint of 1.75 ns on the period. This time, the slack obtained was equal to 0.

To summarize, the following steps have been done:

1)analysis and elaboration of the system, executing ; 2) compilation imposing clock period to 0, in order to find the minimum slack; 3) compilation imposing a time period to 1.75 ns.

The area report of this solution is the following one:

```
****************************************
Report  : area
Design  : FIR
Version: O-2018.06-SP4
Date    : Tue Oct 26 20:47:16 2021
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t

Number of ports:                          322
Number of nets:                          1178
Number of cells:                          807
Number of combinational cells:            659
Number of sequential cells:               128
Number of macros/black boxes:               0
Number of buf/inv:                        148
Number of references:                      31

Combinational area:                979.146004
Buf/Inv area:                       85.386000
Noncombinational area:             657.020012
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  1636.166016
Total area:                 undefined
1
```

As can be seen, this area report misses the net interconnect area since it was not defined in the constraints. To simplify, we can consider a total area equal to the total cell area:

$$Area@1.75ns = 1636.166\mu m^2 \tag{3.3}$$

Design using $f_{clk} = f_M/4$

Once the maximum clock frequency was defined, it was required to go on in the design of the filter relaxing the timing constraint using a clock frequency four times lower than the maximum one, verifying its functionality and evaluating its power consumption.

First, the current designed was then recompiled once again imposing a period given by

$$f_{clk} = f_M/4 \longrightarrow t = 4t_M = 7ns \tag{3.4}$$

We got the following area report:

```
****************************************
Report : area
Design : FIR
Version: O-2018.06-SP4
Date   : Tue Oct 26 20:51:11 2021
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t

Number of ports:                          322
Number of nets:                          1036
Number of cells:                          650
Number of combinational cells:            502
Number of sequential cells:               128
Number of macros/black boxes:               0
Number of buf/inv:                        142
Number of references:                      28

Combinational area:               877.800008
Buf/Inv area:                      81.662000
Noncombinational area:            657.020012
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 1534.820020
Total area:                 undefined
1
```

And, for the same reasons as before, is assumed for result 3.4

$$Area(T_{clk} = 7ns) = 1534.820\mu m^2 \tag{3.5}$$

As can be seen, having relaxed the timing constraint allowed the compiler to generate a solution with lower cost in terms of area.

It is possible now to combine the hierarchy and export a Verilog file combining all the components used in the top-level entity and put them in one file only. This is helpful for simulating the design after synthesizing to check with ModelSim if results haven't changed. Also, this netlist will be used to check the power-switching simulation in ModelSim as well. Moreover, SDF and SDC are important too, since they contain the delays of the netlist and inputs and outputs constraints, respectively. In order to generate those files, it is required to unwrap the cells to flatten the hierarchy, then change the hierarchy to Verilog type, then export. The commands to do the following are:

```
ungroup -all -flatten
change_names -hierarchy -rules verilog
```

```
write_sdf ../netlist/FIR.sdf
write -f verilog -hierarchy -output ../netlist/FIR.v
write_sdc ../netlist/FIR.sdc
```

The verification of the functionality of the generated netlist and the generation of the switching activity information can be obtained at the same time launching a Modelsim simulation in the proper way. As done before, all the stimulus generator and the testbench were compiled by Modelsim; however, instead of the VHDL file describing the filter is loaded the netlist generated by the Design Compiler:

```
vcom -93 -work ./work ../src/constants.vhd
vcom -93 -work ./work ../src/clk_gen.vhd
vcom -93 -work ./work ../src/data_maker.vhd
vcom -93 -work ./work ../src/data_sink_syn.vhd
vlog -work ./work ../netlist/FIR.v
vlog -work ./work ../tb/tb_fir.v
```

To obtain an accurate description, the cell library and also the delay file .sdc have to be loaded using the following commands:

```
vsim -L /software/dk/nangate45/verilog/msim6.2g work.tb_fir
vsim -L /software/dk/nangate45/verilog/msim6.2g -sdftyp /tb_fir/DUT=../netlist/fir_fil
```

Finally, a vcd file containing the switching activity information has to be stored.

```
vcd file ../vcd/FIR_syn.vcd
vcd add /tb_fir/DUT/*
```

Now, the *.saif* file needed to generate the power consumption report is generated by the following command, run in the same environment of Design Compiler:

```
vcd2saif -input ../vcd/FIR_syn.vcd -output ../saif/FIR_fm_over4.saif
```

It is now possible to return to Synopsys Design compiler to generate a power report.

```
read_verilog -netlist ../netlist/FIR.v
read_saif -input ../saif/FIR_fm_over4.saif -instance tb_fir/DUT -unit ns -scale 1
create_clock -name MY_CLK CLK
report_power > power_switching.txt
```

After doing so we get the following power report:

```
Information: Updating design information... (UID-85)
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: There is no defined clock in the design. (PWR-80)

****************************************
Report : power
        -analysis_effort low
Design : FIR
Version: O-2018.06-SP4
Date   : Wed Oct 27 03:12:48 2021
```

```
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t


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

Design          Wire Load Model            Library
------------------------------------------------------
FIR                         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  = 181.1761 uW   (70%)
  Net Switching Power  =  77.6011 uW   (30%)
                         ---------
Total Dynamic Power    = 258.7772 uW  (100%)

Cell Leakage Power     =  28.3818 uW


                   Internal         Switching           Leakage            Total
Power Group        Power            Power               Power              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           107.7475          30.9310          1.1020e+04         149.6984
(   52.13%)
sequential           0.0000           0.0000            0.0000            0.0000
(    0.00%)
```

```
combinational      73.4286        46.6700       1.7362e+04       137.4605
(   47.87%)
------------------------------------------------------------------------
Total             181.1761 uW    77.6011 uW    2.8382e+04 nW    287.1589 uW
1
```

3.4.2 Place & route

After the synthesized netlist verification, the next step is placing and routing the design with Cadence Innovus.
This is done by performing the following steps:

1. Floorplanning
2. Power planning and routing
3. Standard cells placing
4. Signal routing.

After that, an analysis in terms of timing and geometry is performed.

Before all of that, it was necessary to specify the top-level entity and its directory in the *design.globals* file:

```
set IN_DIR "../netlist"
set TopLevelDesign "FIR"
set in_verilog_filename "${IN_DIR}/FIR.v"
set in_sdc_filename "${IN_DIR}/FIR.sdc"
```

Then all the steps described by the design flow document were performed. To summarize, the placement of the standards cells was done with the following settings:

- Core aspect ratio of 1.0, margin from the four sides of the core of 5μm;
- Power rings (VDD and VSS) width and spacing 0.8;
- Placement done in layers 1 to 8;

Then the design was optimized running the Post Clock-Tree-Synthesis (CTS) optimization, selecting both setup and hold type.

Then the filler between cells was added and the routing and post-routing optimization procedure has been done.

In Figure 3.5 is shown the placed and routed design.

3.4.3 Post place and route simulation and switching-activity-based power consumption estimation

To simulate the post place and route netlist, the same procedure done for synthesized netlist was done but compiling ad the top-level entity the verilog file */innovus/FIR.v*, generated by Innovus.

Finally, the total power estimation is performed by Innovus using the vcd file generated by Modelsim. The report is then stored into the innovus folder as *power_report_routed.txt*. The unit used to express power is mW.

The post place and route estimation of the power consumption can be seen in the following report:

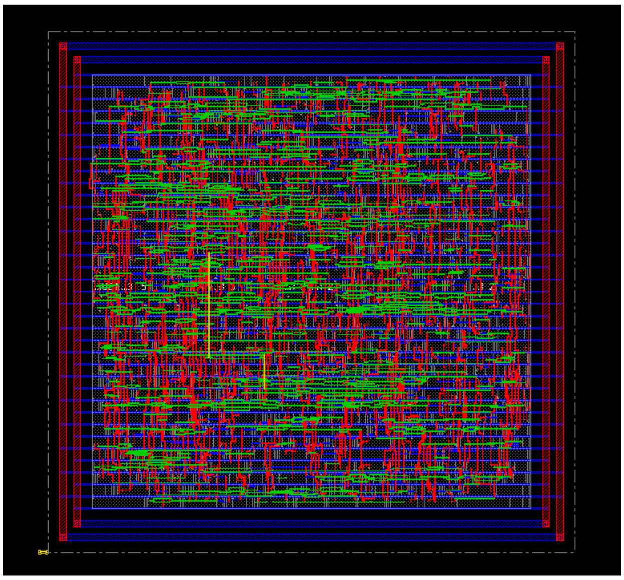

Figure 3.5: Placed and routed design

```
*---------------------------------------------------------------------------
*       Innovus 17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
*
*
*       Date & Time:    2021-Nov-18 20:18:05 (2021-Nov-18 19:18:05 GMT)
*
*---------------------------------------------------------------------------
*
*       Design: FIR
*
*       Liberty Libraries used:
*               MyAnView: /home/isa41_2021_2022/Desktop/lab1_nima/1/new_lab1/lab1.2/innovu
*
*       Power Domain used:
*               Rail:         VDD         Voltage:        1.1
*
*       Power View : MyAnView
*
*       User-Defined Activity : N.A.
*
*       Switching Activity File used:
*               ../vcd/FIR_syn_routed_fm_over4.vcd
*                       Vcd Window used(Start Time, Stop Time):(7.17943e+43, 7.17942e+43)
*                   Vcd Scale Factor: 1
**                  Design annotation coverage: 0/756 = 0%
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
```

```
*         Sequential Element Activity: N.A.
*
*         Primary Input Activity: 0.200000
*
*         Default icg ratio: N.A.
*
*         Global Comb ClockGate Ratio: N.A.
*
*         Power Units = 1mW
*
*         Time Units = 1e-09 secs
*
*         report_power -outfile Power_report_routed -sort total
*
----------------------------------------------------------------------------------------------


Total Power
----------------------------------------------------------------------------------------------
Total Internal Power:        0.32768311          64.1820%
Total Switching Power:       0.15543410          30.4443%
Total Leakage Power:         0.02743563           5.3737%
Total Power:                 0.51055284
----------------------------------------------------------------------------------------------


Group                               Internal    Switching      Leakage        Total
Percentage
                                    Power       Power          Power          Power
(%)
----------------------------------------------------------------------------------------------
Sequential                             0.1531      0.04309      0.01073        0.2069
40.53
Macro                                      0            0            0
0              0
IO                                         0            0            0
0              0
Combinational                          0.1746       0.1123      0.01671        0.3036
59.47
Clock (Combinational)                      0            0            0
0              0
Clock (Sequential)                         0            0            0
0              0
----------------------------------------------------------------------------------------------
Total                                  0.3277       0.1554      0.02744        0.5106
100
----------------------------------------------------------------------------------------------
```

```
Rail                      Voltage   Internal   Switching    Leakage     Total
Percentage
                                    Power      Power        Power       Power
(%)
--------------------------------------------------------------------------------
VDD                          1.1    0.3277     0.1554       0.02744     0.5106
100


--------------------------------------------------------------------------------
*       Power Distribution Summary:
*              Highest Average Power: add_1_root_add_0_root_add_58_I8_U1_3 (FA_X1):
0.003614
*              Highest Leakage Power:            reg_reg_0__0_ (DFFR_X1):
8.565e-05
*              Total Cap:      3.30304e-12 F
*              Total instances in design:   595
*              Total instances in design with no power:     0
*          Total instances in design with no activity:     0
*              Total Fillers and Decap:     0
--------------------------------------------------------------------------------
```

CHAPTER 4

Advanced architecture: reference development model

4.1 Pipelining and Unfolding

Once the design of the FIR filter with the basic architecture carried out now its time to improve the performance using the following methods:

- Pipelining
- 3-level unfolding

To do this, it is needed to begin from Equation 2.1. Using the above equation we formulate 3 new output streams *y[3k],y[3k+1],y[3k+2]* from that we try to elaborate the equation to obtain the elaborated DFG for the 3 level unfolding architecture , which we call as the advance architecture for the betterment of the performance of the filter. The DFG of the 3-level unfolded architecture without pipeling is as shown below:

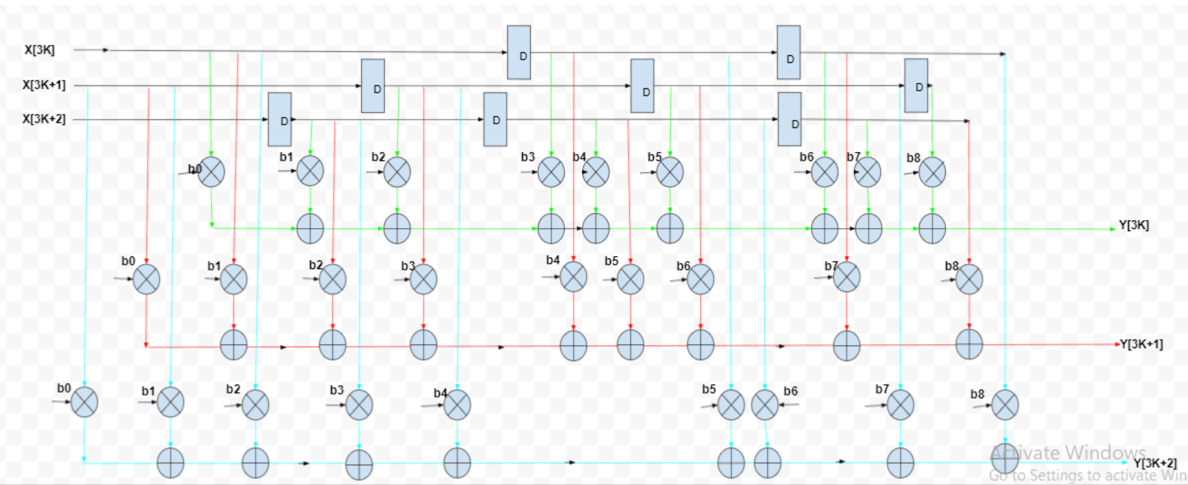

Figure 4.1: 3-Level unfolded DFG without pipelining

This means that there is two way to proceed: applying first the pipelining and then the unfolding, or vice-versa. Our choice was the second one because in this way the procedure is much more straightforward.
The DFG after the implementation of the pipelining is as shown below :

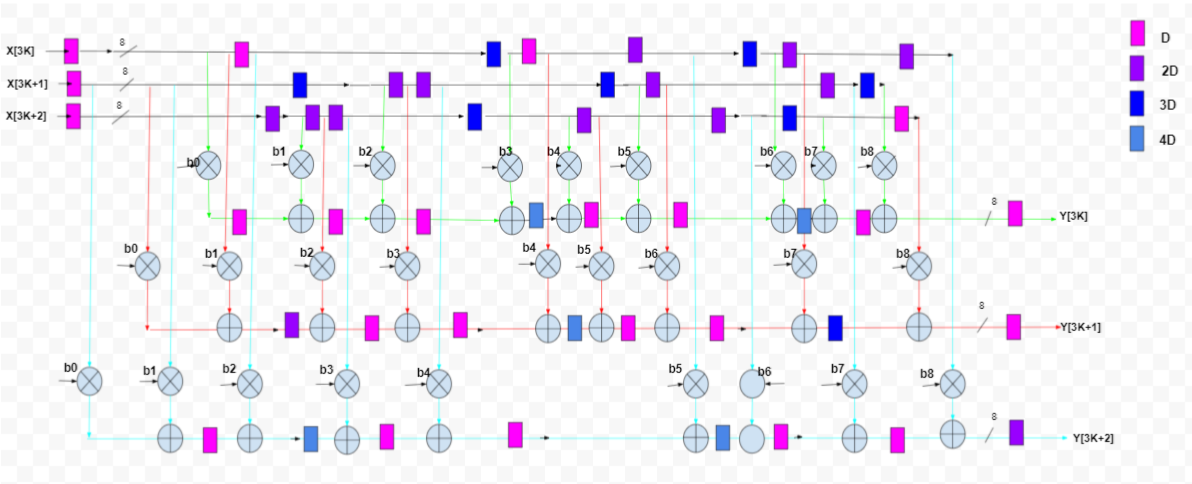

Figure 4.2: 3-Level unfolded DFG with pipelining

The colours actually represent the registers being placed to obtain the particular delay of D which represents one register is being employed and 2D means two register were placed and so on.

Once the 3 level unfolding and pipelining is done thereafter the procedure to simulate, synthesis, and place-and-route the design is equal to the one seen for the basic filter, with the only difference that the testbench is modified to support the unfolding, giving and storing three samples at a time, so here only the final results are reported.

4.2 Modelsim simulation

It can be seen from the simulation result from below shown figure that we got now three input port for sampling *DIN_i,DIN_j,DIN_k* and the same time the output is processed out through the ports *DOUT_i,DOUT_j, DOUT_k*. As mentioned earlier the samples where taken once the VIN goes high which is followed by the DOUT and as we can see from the below simulation result that the out starts appearing at the DOUT ports at 177 ns.

The figure representing the result of the simulation is below:

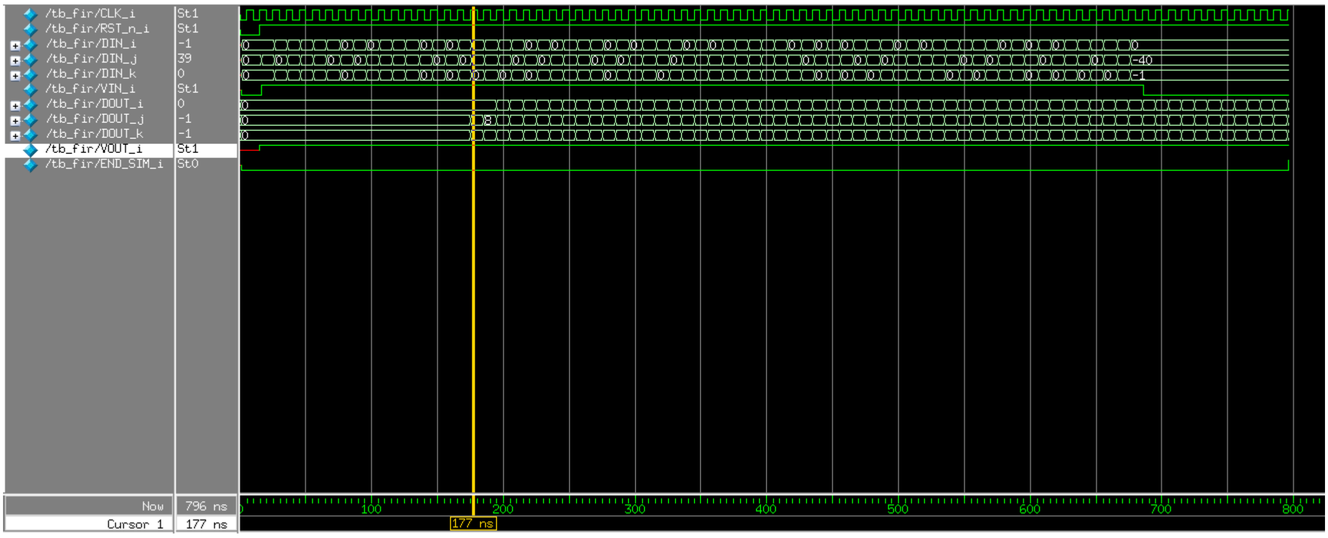

Figure 4.3: Simulation of advance architecture

4.3 Synthesis

The clock period for which the slack of '0' is met at 1.65 ns. And the timing report of the synthesis is as shown below:

```
Information: Updating design information... (UID-85)
Warning: Design 'FIR_unfolded' contains 1 high-fanout nets. A fanout number of 1000 will b

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : FIR_unfolded
Version: O-2018.06-SP4
Date   : Sun Nov 21 05:07:25 2021
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

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

  Startpoint: U2/Q_reg[3]
              (rising edge-triggered flip-flop clocked by MY_CLK)
  Endpoint: U68/Q_reg[8]
            (rising edge-triggered flip-flop clocked by MY_CLK)
  Path Group: MY_CLK
```

```
Path Type: max

Des/Clust/Port       Wire Load Model       Library
------------------------------------------------
FIR_unfolded         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
U2/Q_reg[3]/CK (SDFF_X1)                                0.00 #     0.00 r
U2/Q_reg[3]/Q (SDFF_X1)                                 0.10       0.10 f
U2/Q[3] (oneD_0)                                        0.00       0.10 f
U5/A[3] (mul_0)                                         0.00       0.10 f
U5/mult_20/a[3] (mul_0_DW_mult_tc_0)                    0.00       0.10 f
U5/mult_20/U559/Z (XOR2_X1)                             0.08       0.18 f
U5/mult_20/U437/ZN (NAND2_X1)                           0.04       0.22 r
U5/mult_20/U361/Z (BUF_X1)                              0.05       0.26 r
U5/mult_20/U463/ZN (INV_X1)                             0.03       0.30 f
U5/mult_20/U505/ZN (AOI21_X1)                           0.06       0.35 r
U5/mult_20/U504/ZN (OAI222_X1)                          0.07       0.42 f
U5/mult_20/U426/ZN (NAND2_X1)                           0.04       0.46 r
U5/mult_20/U329/ZN (OAI211_X1)                          0.04       0.50 f
U5/mult_20/U417/ZN (AND2_X1)                            0.04       0.54 f
U5/mult_20/U419/ZN (NOR3_X1)                            0.06       0.60 r
U5/mult_20/U503/ZN (OAI222_X1)                          0.06       0.66 f
U5/mult_20/U393/ZN (NAND2_X1)                           0.04       0.70 r
U5/mult_20/U371/ZN (NAND3_X1)                           0.04       0.74 f
U5/mult_20/U398/ZN (NAND2_X1)                           0.04       0.77 r
U5/mult_20/U401/ZN (NAND3_X1)                           0.04       0.81 f
U5/mult_20/U405/ZN (NAND2_X1)                           0.04       0.85 r
U5/mult_20/U370/ZN (NAND3_X1)                           0.04       0.88 f
U5/mult_20/U410/ZN (NAND2_X1)                           0.04       0.92 r
U5/mult_20/U373/ZN (NAND3_X1)                           0.04       0.96 f
U5/mult_20/U445/ZN (NAND2_X1)                           0.04       0.99 r
U5/mult_20/U446/ZN (NAND3_X1)                           0.04       1.03 f
U5/mult_20/U451/ZN (NAND2_X1)                           0.03       1.06 r
U5/mult_20/U438/ZN (NAND3_X1)                           0.04       1.10 f
U5/mult_20/U459/ZN (NAND2_X1)                           0.04       1.14 r
U5/mult_20/U462/ZN (NAND3_X1)                           0.04       1.17 f
U5/mult_20/U454/ZN (NAND2_X1)                           0.03       1.20 r
U5/mult_20/U456/ZN (NAND3_X1)                           0.03       1.24 f
U5/mult_20/U467/ZN (XNOR2_X1)                           0.05       1.29 r
U5/mult_20/U469/ZN (XNOR2_X1)                           0.06       1.34 r
U5/mult_20/U468/ZN (XNOR2_X1)                           0.06       1.40 r
U5/mult_20/U421/ZN (XNOR2_X1)                           0.07       1.47 r
U5/mult_20/product[15] (mul_0_DW_mult_tc_0)             0.00       1.47 r
U5/U1/Z (BUF_X1)                                        0.06       1.53 r
U5/P[8] (mul_0)                                         0.00       1.53 r
```

```
U68/D[8] (oneD_92)                                  0.00       1.53 r
U68/Q_reg[8]/D (DFF_X2)                             0.01       1.54 r
data arrival time                                              1.54

clock MY_CLK (rise edge)                            1.65       1.65
clock network delay (ideal)                         0.00       1.65
clock uncertainty                                  -0.07       1.58
U68/Q_reg[8]/CK (DFF_X2)                            0.00       1.58 r
library setup time                                 -0.04       1.54
data required time                                             1.54
-----------------------------------------------------------------------
data required time                                             1.54
data arrival time                                             -1.54
-----------------------------------------------------------------------
slack (MET)                                                    0.00


1
```

And the corresponding area report for the above mentioned clock period of 1.65 ns is shown below:

```
****************************************
Report : area
Design : FIR_unfolded
Version: O-2018.06-SP4
Date   : Sun Nov 21 05:07:25 2021
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t

Number of ports:                        17154
Number of nets:                         37214
Number of cells:                        20788
Number of combinational cells:          17429
Number of sequential cells:              3105
Number of macros/black boxes:               0
Number of buf/inv:                       3338
Number of references:                     105

Combinational area:               21869.987907
Buf/Inv area:                      2160.717995
Noncombinational area:            14054.907493
Macro/Black Box area:                 0.000000
Net Interconnect area:       undefined  (Wire load has zero net area)

Total cell area:                  35924.895400
Total area:                  undefined
```

1

From the previous timing report we calculated the relaxed tming constraint that is the above timing period multiplied by 4 , now the clock period becomes 6.60 ns and the timing report is as follows:

```
Information: Updating design information... (UID-85)
Warning: Design 'FIR_unfolded' contains 2 high-fanout nets. A fanout number of 1000 will b

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : FIR_unfolded
Version: O-2018.06-SP4
Date   : Sun Nov 21 05:16:39 2021
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

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

  Startpoint: U4/Q_reg[1]
              (rising edge-triggered flip-flop clocked by MY_CLK)
  Endpoint: U112/Q_reg[31]
            (rising edge-triggered flip-flop clocked by MY_CLK)
  Path Group: MY_CLK
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  FIR_unfolded       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
  U4/Q_reg[1]/CK (DFF_X1)                                 0.00 #     0.00 r
  U4/Q_reg[1]/Q (DFF_X1)                                  0.21       0.21 r
  U4/Q[1] (oneD_95)                                       0.00       0.21 r
  U7/A[1] (mul_25)                                        0.00       0.21 r
  U7/mult_20/a[1] (mul_25_DW_mult_tc_0)                   0.00       0.21 r
  U7/mult_20/U438/Z (XOR2_X1)                             0.12       0.33 r
  U7/mult_20/U324/ZN (INV_X1)                             0.07       0.40 f
  U7/mult_20/U436/ZN (NAND2_X1)                           0.11       0.51 r
  U7/mult_20/U412/ZN (OAI22_X1)                           0.06       0.57 f
  U7/mult_20/U104/S (HA_X1)                               0.08       0.65 f
  U7/mult_20/U329/ZN (INV_X1)                             0.03       0.68 r
  U7/mult_20/U379/ZN (OAI222_X1)                          0.06       0.74 f
```

```
U7/mult_20/U378/ZN (AOI222_X1)                          0.10       0.83 r
U7/mult_20/U323/ZN (INV_X1)                             0.03       0.86 f
U7/mult_20/U377/ZN (AOI222_X1)                          0.11       0.97 r
U7/mult_20/U376/ZN (OAI222_X1)                          0.07       1.03 f
U7/mult_20/U43/CO (FA_X1)                               0.10       1.13 f
U7/mult_20/U42/CO (FA_X1)                               0.09       1.22 f
U7/mult_20/U41/CO (FA_X1)                               0.09       1.31 f
U7/mult_20/U40/CO (FA_X1)                               0.09       1.40 f
U7/mult_20/U39/CO (FA_X1)                               0.09       1.49 f
U7/mult_20/U38/CO (FA_X1)                               0.09       1.58 f
U7/mult_20/U37/CO (FA_X1)                               0.09       1.67 f
U7/mult_20/U36/CO (FA_X1)                               0.09       1.76 f
U7/mult_20/U349/Z (XOR2_X1)                             0.08       1.84 f
U7/mult_20/U348/Z (XOR2_X1)                             0.07       1.91 f
U7/mult_20/U344/Z (XOR2_X1)                             0.07       1.98 f
U7/mult_20/U340/Z (XOR2_X1)                             0.07       2.05 f
U7/mult_20/product[15] (mul_25_DW_mult_tc_0)            0.00       2.05 f
U7/U1/Z (BUF_X1)                                        0.10       2.15 f
U7/P[8] (mul_25)                                        0.00       2.15 f
U111/A[8] (adder_8)                                     0.00       2.15 f
U111/add_20/A[8] (adder_8_DW01_add_0)                   0.00       2.15 f
U111/add_20/U1_8/CO (FA_X1)                             0.13       2.28 f
U111/add_20/U1_9/CO (FA_X1)                             0.09       2.37 f
U111/add_20/U1_10/CO (FA_X1)                            0.09       2.47 f
U111/add_20/U1_11/CO (FA_X1)                            0.09       2.56 f
U111/add_20/U1_12/CO (FA_X1)                            0.09       2.65 f
U111/add_20/U1_13/CO (FA_X1)                            0.09       2.74 f
U111/add_20/U1_14/CO (FA_X1)                            0.09       2.83 f
U111/add_20/U1_15/CO (FA_X1)                            0.09       2.92 f
U111/add_20/U1_16/CO (FA_X1)                            0.09       3.01 f
U111/add_20/U1_17/CO (FA_X1)                            0.09       3.10 f
U111/add_20/U1_18/CO (FA_X1)                            0.09       3.19 f
U111/add_20/U1_19/CO (FA_X1)                            0.09       3.28 f
U111/add_20/U1_20/CO (FA_X1)                            0.09       3.37 f
U111/add_20/U1_21/CO (FA_X1)                            0.09       3.46 f
U111/add_20/U1_22/CO (FA_X1)                            0.09       3.55 f
U111/add_20/U1_23/CO (FA_X1)                            0.09       3.64 f
U111/add_20/U1_24/CO (FA_X1)                            0.09       3.73 f
U111/add_20/U1_25/CO (FA_X1)                            0.09       3.82 f
U111/add_20/U1_26/CO (FA_X1)                            0.09       3.92 f
U111/add_20/U1_27/CO (FA_X1)                            0.09       4.01 f
U111/add_20/U1_28/CO (FA_X1)                            0.09       4.10 f
U111/add_20/U1_29/CO (FA_X1)                            0.09       4.19 f
U111/add_20/U1_30/CO (FA_X1)                            0.09       4.28 f
U111/add_20/U1_31/S (FA_X1)                             0.13       4.41 r
U111/add_20/SUM[31] (adder_8_DW01_add_0)                0.00       4.41 r
U111/S[31] (adder_8)                                    0.00       4.41 r
U112/D[31] (oneD_80)                                    0.00       4.41 r
U112/Q_reg[31]/D (DFF_X1)                               0.01       4.42 r
```

```
  data arrival time                                              4.42

  clock MY_CLK (rise edge)                          6.60         6.60
  clock network delay (ideal)                       0.00         6.60
  clock uncertainty                                -0.07         6.53
  U112/Q_reg[31]/CK (DFF_X1)                        0.00         6.53 r
  library setup time                               -0.03         6.50
  data required time                                             6.50
  -------------------------------------------------------------------
  data required time                                             6.50
  data arrival time                                             -4.42
  -------------------------------------------------------------------
  slack (MET)                                                    2.08


1
```

And the corresponding area report for the clock period of 6.60 ns is shown below:

```
****************************************
Report : area
Design : FIR_unfolded
Version: O-2018.06-SP4
Date   : Sun Nov 21 05:16:39 2021
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t

Number of ports:                        17154
Number of nets:                         28725
Number of cells:                        11359
Number of combinational cells:           8001
Number of sequential cells:              3105
Number of macros/black boxes:               0
Number of buf/inv:                        540
Number of references:                     104

Combinational area:               16660.378057
Buf/Inv area:                       308.560001
Noncombinational area:            14038.947491
Macro/Black Box area:                 0.000000
Net Interconnect area:       undefined  (Wire load has zero net area)

Total cell area:                  30699.325548
Total area:                  undefined
1
```

From the relaxed timing we can see that the area got reduced from 35924.895 sq.micrometers to 30699.325 sq.micrometers. Which both are obviously higher than the area that was reported for the basic architecture , because as we increased the parallelism in the advance architecture. Similarly the post synthesis power switching report suggests that the power consumption of the advance architecture is higher than the basic architecture.And the post synthesis power report is shown below:

```
Information: Updating design information... (UID-85)
Warning: Design 'FIR_unfolded' contains 2 high-fanout nets. A fanout number of 1000 will b
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: There is no defined clock in the design. (PWR-80)

****************************************
Report : power
        -analysis_effort low
Design : FIR_unfolded
Version: O-2018.06-SP4
Date   : Sun Nov 21 05:26:34 2021
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/NangateOpenCellLibrary_t


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

Design        Wire Load Model            Library
------------------------------------------------
FIR_unfolded           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  =   3.0654 mW   (75%)
  Net Switching Power  =   1.0466 mW   (25%)
                         ---------
Total Dynamic Power    =   4.1120 mW  (100%)

Cell Leakage Power     = 579.2385 uW
```

Power **Group** (%) Attrs	Internal Power	Switching Power	Leakage Power	Total Power
io_pad (0.00%)	0.0000	0.0000	0.0000	0.0000
memory (0.00%)	0.0000	0.0000	0.0000	0.0000
black_box (0.00%)	0.0000	0.0000	0.0000	0.0000
clock_network (0.00%)	0.0000	0.0000	0.0000	0.0000
register (57.79%)	2.1969e+03	268.5458	2.4557e+05	2.7110e+03
sequential (0.00%)	3.3022e−03	0.0000	44.4215	4.7724e−02
combinational (42.21%)	868.5141	778.0584	3.3362e+05	1.9802e+03
Total 1	3.0654e+03 uW	1.0466e+03 uW	5.7924e+05 nW	4.6912e+03 uW

4.4 Place and Route

The obtained floorplan in Cadence Innovous graphic envirnoment is shown below:

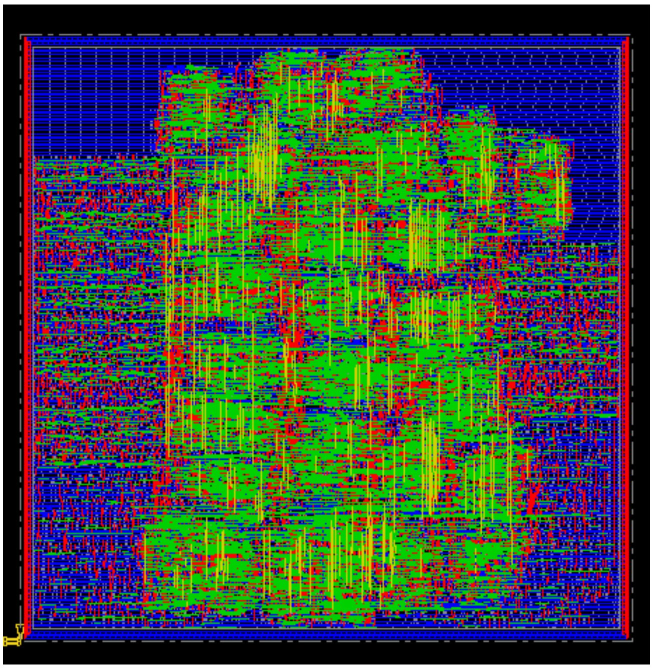

Figure 4.4: Floor plan

The post-place-and-route power consumption is slightly higher when compared to post-synthesis power consumption, however the post place and route power report is shown below:

```
*----------------------------------------------------------------------------------------
*       Innovus 17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
*
*
*       Date & Time:    2021-Nov-18 20:18:05 (2021-Nov-18 19:18:05 GMT)
*
*----------------------------------------------------------------------------------------
*
*       Design: FIR
*
*       Liberty Libraries used:
*               MyAnView: /home/isa41_2021_2022/Desktop/lab1_nima/1/new_lab1/lab1.2/innovu
*
*       Power Domain used:
*               Rail:       VDD         Voltage:        1.1
*
*       Power View : MyAnView
*
*       User-Defined Activity : N.A.
*
*       Switching Activity File used:
*               ../vcd/FIR_syn_routed_fm_over4.vcd
```

```
*                    Vcd Window used(Start Time, Stop Time):(7.17943e+43, 7.17942e+43)
*                    Vcd Scale Factor: 1
**                   Design annotation coverage: 0/756 = 0%
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
*       Sequential Element Activity: N.A.
*
*       Primary Input Activity: 0.200000
*
*       Default icg ratio: N.A.
*
*       Global Comb ClockGate Ratio: N.A.
*
*       Power Units = 1mW
*
*       Time Units = 1e-09 secs
*
*       report_power -outfile Power_report_routed -sort total
*
-------------------------------------------------------------------------------


Total Power
-------------------------------------------------------------------------------
Total Internal Power:        0.32768311          64.1820%
Total Switching Power:       0.15543410          30.4443%
Total Leakage Power:         0.02743563           5.3737%
Total Power:                 0.51055284
-------------------------------------------------------------------------------


Group                              Internal   Switching    Leakage      Total
Percentage
                                   Power      Power        Power        Power
(%)
-------------------------------------------------------------------------------
Sequential                           0.1531     0.04309     0.01073     0.2069
40.53
Macro                                     0           0           0
0             0
IO                                        0           0           0
0             0
Combinational                        0.1746      0.1123     0.01671     0.3036
59.47
Clock (Combinational)                     0           0           0
0             0
```

```
Clock (Sequential)                          0          0          0
0           0
--------------------------------------------------------------------------------
Total                                  0.3277     0.1554    0.02744     0.5106
100
--------------------------------------------------------------------------------


Rail                       Voltage   Internal   Switching    Leakage      Total
Percentage
                                     Power      Power        Power        Power
(%)
--------------------------------------------------------------------------------
VDD                            1.1     0.3277     0.1554    0.02744     0.5106
100


--------------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power: add_1_root_add_0_root_add_58_I8_U1_3 (FA_X1):
0.003614
*               Highest Leakage Power:            reg_reg_0__0_ (DFFR_X1):
8.565e-05
*               Total Cap:      3.30304e-12 F
*               Total instances in design:   595
*               Total instances in design with no power:     0
*          Total instances in design with no activity:     0
*               Total Fillers and Decap:     0
--------------------------------------------------------------------------------
```

The exact parameters of power we are interested in shown below and the units is milliWatt , mW
:

Total Power

Total Internal Power:	0.32768311	64.1820%
Total Switching Power:	0.15543410	30.4443%
Total Leakage Power:	0.02743563	5.3737%
Total Power:	0.51055284	

Figure 4.5: Advanced design post place-and-route switching-activity-based power consumption

APPENDIX A

Basic architecture reference model

A.1 MATLAB: result_analysis.m

```
fs=10000 %% sampling frequency
f1=500;  %% first sinewave freq (in band)
f2=4500; %% second sinnewave freq (out band)

N=8; %% filter order
nb=8; %% number of bits

T=1/500; %% maximum period
tt=0:1/fs:10*T; %% time samples

x1=sin(2*pi*f1*tt); %% first sinewave
x2=sin(2*pi*f2*tt); %% second sinewave

x=(x1+x2)/2; %% input signal

[bi, bq]=myfir_design(N, nb); %% filter design

y=filter(bq, 1, x); %% apply filter

%% plots
figure
plot(tt,x1,'--d');
hold on
plot(tt,x2,'r--s');
plot(tt,x, 'g--+');
plot(tt, y, 'c--o');

legend('x1', 'x2', 'x', 'y')

%% quantize input and output
xq=floor(x*2^(nb-1));
idx=find(xq==2^(nb-1));
```

```
xq(idx)=2^(nb-1)-1;

yq=floor(y*2^(nb-1));
idy=find(yq==2^(nb-1));
yq(idy)=2^(nb-1)-1;

%% save input and output
fp1=fopen('C:\Users\asus\Desktop\Term3\Integrated_systems_architecture\LABS\LAB1\samples.t
fprintf(fp1, '%d\n', xq);
fclose(fp1);

fp=fopen('C:\Users\asus\Desktop\Term3\Integrated_systems_architecture\LABS\LAB1\resultsm.t
fprintf(fp, '%d\n', yq);
fclose(fp);
```

A.2 MATLAB: myfir_design.m

```
function [bi, bq]=myfir_design(N,nb)
%% function myfir_design(N,nb)
%% N is order of the filter
N=8;
%% nb is the number of bits
nb=8;
%% bi taps represented as integers

close all;

f_cut_off = 2000; % 1kHz
f_sampling = 10000; % 10kHz

f_nyq = f_sampling/2; %% Nyquist frequency
f0 = f_cut_off/f_nyq; %% normalized cut-off frequency

b=fir1(N, f0); %% get filter coefficients
[h1, w1]=freqz(b); %% get the transfer function of the designed filter

bi=floor(b*2^(nb-1)); %% convert coefficients into nb-bit integers
bq=bi/2^(nb-1); %% convert back coefficients as nb-bit real values
[h2, w2]=freqz(bq); %% get the transfer function of the quantized filter

%% show the transfer functions
plot(w1/pi, 20*log10(abs(h1)));
hold on;
plot(w2/pi, 20*log10(abs(h2)),'r--');
grid on;
xlabel('Normalized frequency');
ylabel('dB');
```

A.3 MATLAB: results

0
−1
−1
1
8
19
36
48
59
60
59
48
36
18
−1
−19
−37
−49
−60
−61
−60
−49
−37
−19
−1
18
36
48
59
60
59
48
36
18
0
−19
−37
−49
−60
−61
−60
−49
−37
−19
−1
18
36

48
59
60
59
48
36
18
−1
−19
−37
−49
−60
−61
−60
−49
−37
−19
0
18
36
48
59
60
59
48
36
18
0
−19
−37
−49
−60
−61
−60
−49
−37
−19
0
18
36
48
59
60
59
48
36
18
−1
−19

−37
−49
−60
−61
−60
−49
−37
−19
−1
18
36
48
59
60
59
48
36
18
0
−19
−37
−49
−60
−61
−60
−49
−37
−19
0
18
36
48
59
60
59
48
36
18
−1
−19
−37
−49
−60
−61
−60
−49
−37
−19
−1

18
36
48
59
60
59
48
36
18
0
−19
−37
−49
−60
−61
−60
−49
−37
−19
0
18
36
48
59
60
59
48
36
18
−1
−19
−37
−49
−60
−61
−60
−49
−37
−19
0
18
36
48
59
60
59
48
36
18

```
0
-19
-37
-49
-60
-61
-60
```

A.4 MATLAB: comparison_matlab_C

```
fileID = fopen('C:\Users\asus\Desktop\Term3\Integrated_systems_architecture\LABS\LAB1\resu
fileID1 = fopen('C:\Users\asus\Desktop\Term3\Integrated_systems_architecture\LABS\LAB1\res
b = fscanf(fileID1, '%d');
a = fscanf(fileID, '%d');
thd_of_c = thd(a);
thd_of_matlab = thd(b);
figure(1);
hold on
plot(a,'r');
plot(b,'b');
```

A.5 C: myfilter.c

```
#include<stdio.h>
#include<stdlib.h>

#define NT 9 /// number of coeffs
#define NB 8 /// number of bits

const int b[NT]={-1,-2,6,34,51,34,6,-2,-1}; /// b array
//const int a[NT-1]={-147, 52}; /// a array

/// Perform fixed point filtering assming direct form I
///\param x is the new input sample
///\return the new output sample
int myfilter(int x)
{
  static int sx[NT]; /// x shift register
  static int sy[NT-1]; /// y shift register
  static int first_run = 0; /// for cleaning shift registers
  int i; /// index
  int y; /// output sample

  /// clean the buffers
  if (first_run == 0)
  {
    first_run = 1;
    for (i=0; i<NT; i++)
      sx[i] = 0;
```

```
    for (i=0; i<NT-1; i++)
      sy[i] = 0;
  }

  /// shift and insert new sample in x shift register
  for (i=NT-1; i>0; i--)
    sx[i] = sx[i-1];
  sx[0] = x;

  /// make the convolution
  /// Moving average part
  y = 0;
  for (i=0; i<NT; i++)
    y += (sx[i]*b[i]) >> (NB-1) ;

  /// update the y shift register
  for (i=NT-2; i>0; i--)
    sy[i] = sy[i-1];
  sy[0] = y;

  return y;
}

int main (int argc, char **argv)
{
  FILE *fp_in;
  FILE *fp_out;

  int x;
  int y;

  /// check the command line
  if (argc != 3)
  {
    printf("Use: %s <input_file> <output_file>\n", argv[0]);
    exit(1);
  }

  /// open files
  fp_in = fopen(argv[1], "r");
  if (fp_in == NULL)
  {
    printf("Error: cannot open %s\n");
    exit(2);
  }
  fp_out = fopen(argv[2], "w");

  /// get samples and apply filter
  fscanf(fp_in, "%d", &x);
```

```
  do{
    y = myfilter(x);
    fprintf(fp_out,"%d\n", y);
    fscanf(fp_in, "%d", &x);
  } while (!feof(fp_in));

  fclose(fp_in);
  fclose(fp_out);

  return 0;

}
```

A.6 Results: Output of C

```
0
-1
-1
0
8
18
35
46
56
55
55
44
32
14
-5
-22
-41
-53
-63
-62
-62
-51
-38
-20
-2
16
35
46
56
55
55
44
32
14
```

−5
−22
−41
−53
−63
−62
−62
−51
−38
−20
−2
16
35
46
57
56
56
45
34
15
−4
−21
−40
−53
−63
−62
−62
−51
−38
−20
−2
16
35
46
57
56
56
45
33
14
−5
−22
−41
−53
−63
−62
−62
−51
−38

−20
−2
16
35
46
56
55
55
44
32
14
−4
−21
−39
−51
−61
−61
−62
−52
−40
−22
−4
15
33
45
55
54
54
44
32
14
−5
−22
−41
−53
−64
−63
−64
−53
−40
−21
−4
15
34
45
56
56
57
46

34
15
−3
−21
−39
−51
−61
−61
−62
−52
−39
−21
−3
16
35
46
56
55
55
44
33
15
−4
−21
−40
−53
−64
−63
−63
−52
−39
−20
−3
15
34
45
56
56
57
46
34
15
−4
−22
−40
−52
−62
−61
−61

−51
−38
−20
−2
16
34
45
55
54
55
45
34
16
−3
−21
−39
−52
−63
−62
−62

APPENDIX B

VHDL basic architecture description

B.1 FIR.vhd

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
--use ieee.std_logic_signed.all;
use ieee.numeric_std.all;

entity FIR is
  port (
    din:in std_logic_vector(7 downto 0);
    vin:in std_logic;
    rst:in std_logic;
    clk:in std_logic;

    dout:out std_logic_vector(7 downto 0);
    vout:out std_logic
     );
end FIR;

architecture behavioral of FIR is
  type registers is array (7 downto 0) of integer;
  type coefficients is array (8 downto 0) of integer;
  signal reg: registers;
  constant coef : coefficients := (-1,-2,6,34,51,34,6,-2,-1);
  signal din_temp : integer :=0;

begin
  din_temp<= to_integer(signed(din));
  process (clk,rst)
  variable prod_temp : std_logic_vector (15 downto 0);
  variable prod_shift : std_logic_vector (8 downto 0);
  variable acc,prod: integer :=0;
  variable vout_temp : std_logic := '0';
```

```
 begin
   if (rst= '0') then

     for i in 5 downto 0 loop
       reg(i)<= 0;
               dout<= "00000000";
     end loop;
   elsif (clk'event and clk='1') then
       if (vin='1') then
                     acc:=0;
               for i in 7 downto 1 loop
               reg(i) <= reg(i-1);
               end loop;
               prod := coef(0)*din_temp;
               prod_temp := std_logic_vector(to_signed(prod,16));
               prod_shift := prod_temp(15 downto 7);
               prod := to_integer(signed(prod_shift));
               acc := acc + prod;

       prod := 0;
               for i in 1 to 8 loop
         prod := coef(i)*reg(i-1);
                 prod_temp := std_logic_vector(to_signed(prod,16));
                 prod_shift := prod_temp(15 downto 7);
                 prod := to_integer(signed(prod_shift));
         acc := acc + prod;
                     prod := 0;
               end loop;
                     vout_temp:='1';
                     reg(0) <= din_temp;
     if (vout_temp='1') then
     dout <= std_logic_vector(to_signed(acc,8));
             vout_temp :='0';
     end if;
   end if;
       end if;
             vout<= vout_temp;
  end process;
end behavioral;
```

APPENDIX C

Basic architecture testbench

C.1 data_maker.vhd

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
--use ieee.std_logic_signed.all;
use ieee.numeric_std.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

entity data_maker is
  port (
    CLK     : in  std_logic;
    RST_n   : in  std_logic;
    VOUT    : out std_logic;
    DOUT    : out std_logic_vector(7 downto 0);
--      H0      : out std_logic_vector(15 downto 0);
--      H1      : out std_logic_vector(15 downto 0);
--      H2      : out std_logic_vector(15 downto 0);
--      H3      : out std_logic_vector(15 downto 0);
    END_SIM : out std_logic);
end data_maker;

architecture beh of data_maker is

  constant tco : time := 1 ns;

  signal sEndSim : std_logic;
  signal END_SIM_i : std_logic_vector(0 to 10);

begin  -- beh
```

```
--  H0 <= conv_std_logic_vector(286,16);
--  H1 <= conv_std_logic_vector(1571,16);
--  H2 <= conv_std_logic_vector(5374,16);
--  H3 <= conv_std_logic_vector(9151,16);

  process (CLK, RST_n)
    file fp_in : text open READ_MODE is "./samples.txt";
    variable line_in : line;
    variable x : integer;
  begin  -- process
    if RST_n = '0' then                    -- asynchronous reset (active low)
      DOUT <= (others => '0') after tco;
      VOUT <= '0' after tco;
      sEndSim <= '0' after tco;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if not endfile(fp_in) then
        readline(fp_in, line_in);
        read(line_in, x);
        DOUT <= std_logic_vector(to_signed(x, 8)) after tco;
        VOUT <= '1' after tco;
        sEndSim <= '0' after tco;
      else
        VOUT <= '0' after tco;
        sEndSim <= '1' after tco;
      end if;
    end if;
  end process;

  process (CLK, RST_n)
  begin  -- process
    if RST_n = '0' then                    -- asynchronous reset (active low)
      END_SIM_i <= (others => '0') after tco;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      END_SIM_i(0) <= sEndSim after tco;
      END_SIM_i(1 to 10) <= END_SIM_i(0 to 9) after tco;
    end if;
  end process;

  END_SIM <= END_SIM_i(10);

end beh;
```

C.2 data_sink.vhd

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
--use ieee.std_logic_signed.all;
use ieee.numeric_std.all;
```

```
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

entity data_sink is
  port (
    CLK    : in std_logic;
    RST_n  : in std_logic;
    VIN    : in std_logic;
    DIN    : in std_logic_vector(7 downto 0);
        END_SIM : in std_logic);
end data_sink;

architecture beh of data_sink is

begin  -- beh

  process (CLK, RST_n)
    file res_fp : text open WRITE_MODE is "./results.txt";
    variable line_out : line;
  begin  -- process
    if RST_n = '0' then                    -- asynchronous reset (active low)
      null;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if (VIN = '0') then
        write(line_out, to_integer(signed(DIN)));
        writeline(res_fp, line_out);
                if      (END_SIM = '1') then
                        file_close(res_fp);
                end if;
      end if;
    end if;
  end process;

end beh;
```

C.3 clk_gen.vhd

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
--use ieee.std_logic_signed.all;
use ieee.numeric_std.all;

entity clk_gen is
  port (
    END_SIM : in  std_logic;
    CLK     : out std_logic;
```

```
    RST_n    : out std_logic);
end clk_gen;

architecture beh of clk_gen is

  constant Ts : time := 10 ns;

  signal CLK_i : std_logic;

begin  -- beh

  process
  begin  -- process
    if (CLK_i = 'U') then
      CLK_i <= '0';
    else
      CLK_i <= not(CLK_i);
    end if;
    wait for Ts/2;
  end process;

  CLK <= CLK_i and not(END_SIM);

  process
  begin  -- process
    RST_n <= '0';
    wait for 3*Ts/2;
    RST_n <= '1';
    wait;
  end process;

end beh;
```

C.4 tb_fir.v

```
//`timescale 1ns


module tb_fir ();

   wire CLK_i;
   wire RST_n_i;
   wire [7:0] DIN_i;
   wire VIN_i;
   wire [7:0] DOUT_i;
   wire VOUT_i;
   wire END_SIM_i;

   data_sink DS(.CLK(CLK_i),
```

```
                  .RST_n(RST_n_i),
                  .VIN(VOUT_i),
                  .DIN(DOUT_i),
                  .END_SIM(END_SIM_i));

   clk_gen CG(.END_SIM(END_SIM_i),
              .CLK(CLK_i),
              .RST_n(RST_n_i));

   data_maker SM(.CLK(CLK_i),
                 .RST_n(RST_n_i),
                 .VOUT(VIN_i),
                 .DOUT(DIN_i),
                 .END_SIM(END_SIM_i));

   FIR UUT(.din(DIN_i),
           .vin(VIN_i),
                   .rst(RST_n_i),
                   .clk(CLK_i),
           .dout(DOUT_i),
           .vout(VOUT_i));


endmodule
```

APPENDIX D

Advanced architecture description

D.1 FIR_unfolded.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity FIR_unfolded is
  port (
        end_sim_i:in std_logic;
    din:in std_logic_vector(7 downto 0);
    din1:in std_logic_vector(7 downto 0);
    din2:in std_logic_vector(7 downto 0);
    vin:in std_logic;
    rst:in std_logic;
    clk:in std_logic;

    dout:out std_logic_vector(7 downto 0);
    dout1:out std_logic_vector(7 downto 0);
    dout2:out std_logic_vector(7 downto 0);
    vout:out std_logic
     );
end FIR_unfolded;

architecture behavioral of FIR_unfolded is

          type coefficients is array (8 downto 0) of integer;
          constant coef : coefficients := (-1,-2,6,34,51,34,6,-2,-1);
--Components
component fourD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
```

```
end component;

component threeD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

component oneD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

component twoD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

component mul is
        port (
                A: in integer;
                B: in integer;
                P: out integer
        );
end component;

component adder is
        port (
                A: in integer;
                B: in integer;
                S: out integer
        );
end component;
-----------------------------------
--signals
signal D0,D1,D2,P00,P01,P02,P03,P04,P05,P06,P07,P08
                                ,P10,P11,P12,P13,P14,P15,P16,P17,P18
                                ,P20,P21,P22,P23,P24,P25,P26,P27,P28,
                                     Q00,Q01,Q02,Q03,Q04,Q05,Q06,Q07,Q08
                                ,Q10,Q11,Q12,Q13,Q14,Q15,Q16,Q17,Q18
                                ,Q20,Q21,Q22,Q23,Q24,Q25,Q26,Q27,Q28,
```

```
                                    S00 , S01 , S02 , S03 , S04 , S05 , S06 , S07 , S08
                                ,S10 , S11 , S12 , S13 , S14 , S15 , S16 , S17 , S18
                                ,S20 , S21 , S22 , S23 , S24 , S25 , S26 , S27 , S28 ,
                                    SQ00 , SQ01 , SQ02 , SQ03 , SQ04 , SQ05 , SQ06 , SQ07
                                ,SQ10 , SQ11 , SQ12 , SQ13 , SQ14 , SQ15 , SQ16 , SQ17
                                ,SQ20 , SQ21 , SQ22 , SQ23 , SQ24 , SQ25 , SQ26 , SQ27 , PQ00
                                    : integer:=0;

signal vout_temp: std_logic;

-----------------------------------
--assignings

begin

--receiving inputs
D0 <= to_integer(signed(din));
D1 <= to_integer(signed(din1));
D2 <= to_integer(signed(din2));
--multiplying part
U2 : oneD port map(D0, clk ,Q00);
U3 : oneD port map(D1, clk ,Q10);
U4 : oneD port map(D2, clk ,Q20);

U5 : mul port map(Q00, coef(0),P00);
U6 : mul port map(Q10, coef(0),P10);
U7 : mul port map(Q20, coef(0),P20);

U10 : twoD port map(Q20, clk ,Q21);

U11 : mul port map(Q00, coef(1),P01);
U12 : mul port map(Q10, coef(1),P11);
U13 : mul port map(Q21, coef(1),P21);

U14 : oneD port map(Q00, clk ,Q01);
U15 : threeD port map(Q10, clk ,Q11);
U16 : twoD port map(Q21, clk ,Q22);

U17 : mul port map(Q01, coef(2),P02);
U18 : mul port map(Q11, coef(2),P12);
U19 : mul port map(Q22, coef(2),P22);

U20 : threeD port map(Q01, clk ,Q02);
U21 : twoD port map(Q11, clk ,Q12);
U22 : twoD port map(Q22, clk ,Q23);

U23 : mul port map(Q02, coef(3),P03);
U24 : mul port map(Q12, coef(3),P13);
U25 : mul port map(Q23, coef(3),P23);
```

```
U26 : twoD port map(Q02, clk ,Q03);
U27 : twoD port map(Q12, clk ,Q13);
U28 : threeD port map(Q23, clk ,Q24);

U29 : mul port map(Q03, coef(4),P04);
U30 : mul port map(Q13, coef(4),P14);
U31 : mul port map(Q24, coef(4),P24);

U32 : twoD port map(Q03, clk ,Q04);
U33 : threeD port map(Q13, clk ,Q14);
U34 : twoD port map(Q24, clk ,Q25);

U35 : mul port map(Q04, coef(5),P05);
U36 : mul port map(Q14, coef(5),P15);
U37 : mul port map(Q25, coef(5),P25);

U38 : threeD port map(Q04, clk ,Q05);
U39 : twoD port map(Q14, clk ,Q15);
U40 : twoD port map(Q25, clk ,Q26);

U41 : mul port map(Q05, coef(6),P06);
U42 : mul port map(Q15, coef(6),P16);
U43 : mul port map(Q26, coef(6),P26);

U44 : twoD port map(Q05, clk ,Q06);
U45 : twoD port map(Q15, clk ,Q16);
U46 : threeD port map(Q26, clk ,Q27);

U47 : mul port map(Q06, coef(7),P07);
U48 : mul port map(Q16, coef(7),P17);
U49 : mul port map(Q27, coef(7),P27);

U50 : twoD port map(Q06, clk ,Q07);
U51 : threeD port map(Q16, clk ,Q17);
U52 : oneD port map(Q27, clk ,Q28);

U53 : mul port map(Q07, coef(8),P08);
U54 : mul port map(Q17, coef(8),P18);
U55 : mul port map(Q28, coef(8),P28);

-------------------------------------------------
--accumulations
U68 : oneD port map(P00, clk ,PQ00);
U69 : adder port map(PQ00,P21,S00);
U70 : oneD port map(S00, clk ,SQ00);
U71 : adder port map(SQ00,P12,S01);
U72 : oneD port map(S01, clk ,SQ01);
U73 : adder port map(SQ01,P03,S02);
```

```
U74 : fourD port map(S02, clk, SQ02);
U75 : adder port map(SQ02, P24, S03);
U76 : oneD port map(S03, clk, SQ03);
U77 : adder port map(SQ03, P15, S04);
U78 : oneD port map(S04, clk, SQ04);
U79 : adder port map(SQ04, P06, S05);
U80 : fourD port map(S05, clk, SQ05);
U81 : adder port map(SQ05, P27, S06);
U82 : oneD port map(S06, clk, SQ06);
U83 : adder port map(SQ06, P18, S07);
U84 : oneD port map(S07, clk, SQ07);


U90 : adder port map(P10, P01, S10);
U91 : threeD port map(S10, clk, SQ10);
U92 : adder port map(SQ10, P22, S11);
U93 : oneD port map(S11, clk, SQ11);
U94 : adder port map(SQ11, P13, S12);
U95 : oneD port map(S12, clk, SQ12);
U96 : adder port map(SQ12, P04, S13);
U97 : fourD port map(S13, clk, SQ13);
U98 : adder port map(SQ13, P25, S14);
U99 : oneD port map(S14, clk, SQ14);
U100 : adder port map(SQ14, P16, S15);
U101 : oneD port map(S15, clk, SQ15);
U102 : adder port map(SQ15, P07, S16);
U103 : threeD port map(S16, clk, SQ16);
U104 : adder port map(SQ16, P28, S17);
U105 : oneD port map(S17, clk, SQ17);


U111 : adder port map(P20, P11, S20);
U112 : oneD port map(S20, clk, SQ20);
U113 : adder port map(SQ20, P02, S21);
U114 : fourD port map(S21, clk, SQ21);
U115 : adder port map(SQ21, P23, S22);
U116 : oneD port map(S22, clk, SQ22);
U117 : adder port map(SQ22, P14, S23);
U118 : oneD port map(S23, clk, SQ23);
U119 : adder port map(SQ23, P05, S24);
U120 : fourD port map(S24, clk, SQ24);
U121 : adder port map(SQ24, P26, S25);
U122 : oneD port map(S25, clk, SQ25);
U123 : adder port map(SQ25, P17, S26);
U124 : oneD port map(S26, clk, SQ26);
U125 : adder port map(SQ26, P08, S27);
U126 : twoD port map(S27, clk, SQ27);
```

```
dout2<=std_logic_vector(to_signed(SQ27,8));
dout1<=std_logic_vector(to_signed(SQ17,8));
dout<=std_logic_vector(to_signed(SQ07,8));

process (clk,rst)
begin
        if rst='1' then
                vout_temp <= '0';
                if end_sim_i='0' then
                        vout_temp<='1';
                end if;
        end if;
end process;
vout <=vout_temp;
end behavioral;
```

D.2 sum.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity adder is
        port (
                A: in integer;
                B: in integer;
                S: out integer
        );
end adder;

architecture behavioral of adder is

begin



S <= A+B;




end behavioral;
```

D.3 mul.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
entity mul is
        port (
                A: in integer;
                B: in integer;
                P: out integer
        );
end mul;

architecture behavioral of mul is
signal P1:std_logic_vector(15 downto 0);
signal P2:std_logic_vector(8 downto 0);
signal P_temp:integer:=0;
begin

P_temp<=A*B;

P1<=std_logic_vector(to_signed(P_temp,16));
P2<=P1(15 downto 7);
P<=to_integer(signed(P2));

end behavioral;
```

D.4 oneD.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity oneD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end oneD;

architecture behavioral of oneD is
signal D_temp : integer :=0;
begin
        D_temp<= D;
        process(clk)
                begin
                        if rising_edge(clk) then
                                Q<=D_temp;
                        end if;
        end process;
end behavioral;
```

D.5 twoD.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity twoD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end twoD;

architecture behavioral of twoD is

component oneD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

signal Q1,Q2: integer := 0;
signal D_temp : integer :=0;

begin

DFF1: oneD port map(D,clk,Q1);
DFF2: oneD port map(Q1,clk,Q2);
Q<=Q2;

end behavioral;
```

D.6 threeD.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity threeD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end threeD;
```

```
architecture behavioral of threeD is

component oneD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

signal Q1,Q2: integer := 0;

begin

DFF1: oneD port map(D,clk,Q1);
DFF2: oneD port map(Q1,clk,Q2);
DFF3: oneD port map(Q2,clk,Q);

end behavioral;
```

D.7 fourD.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fourD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end fourD;

architecture behavioral of fourD is

component oneD is
        port (
                D: in integer;
                clk: in std_logic;
                Q: out integer
        );
end component;

signal Q1,Q2,Q3: integer := 0;

begin

DFF1: oneD port map(D,clk,Q1);
```

```
DFF2: oneD port map(Q1, clk ,Q2);
DFF3: oneD port map(Q2, clk ,Q3);
DFF4: oneD port map(Q3, clk ,Q);

end behavioral;
```

D.8 tb_fir.v

```
//`timescale 1ns


module tb_fir ();

    wire CLK_i;
    wire RST_n_i;
    wire [7:0] DIN_i;
    wire [7:0] DIN_j;
    wire [7:0] DIN_k;
    wire VIN_i;
    wire [7:0] DOUT_i;
    wire [7:0] DOUT_j;
    wire [7:0] DOUT_k;
    wire VOUT_i;
    wire END_SIM_i;

    data_sink DS(.CLK(CLK_i),
                 .RST_n(RST_n_i),
                 .VIN(VOUT_i),
                 .DIN(DOUT_i),
                 .DIN1(DOUT_j),
                 .DIN2(DOUT_k));

    clk_gen CG(.END_SIM(END_SIM_i),
               .CLK(CLK_i),
               .RST_n(RST_n_i));

    data_maker SM(.CLK(CLK_i),
                  .RST_n(RST_n_i),
                  .VOUT(VIN_i),
                  .DOUT(DIN_i),
                  .DOUT1(DIN_j),
                  .DOUT2(DIN_k),
                  .END_SIM(END_SIM_i));

    FIR_unfolded UUT(.end_sim_i(END_SIM_i),
                     .din(DIN_i),
                      .din1(DIN_j),
                      .din2(DIN_k),
        .vin(VIN_i),
```

```
                         .rst(RST_n_i),
                         .clk(CLK_i),
          .dout(DOUT_i),
          .dout1(DOUT_j),
          .dout2(DOUT_k),
          .vout(VOUT_i));


always @(posedge END_SIM_i) begin
$finish;
end
endmodule
```

D.9 clk_gen.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clk_gen is
  port (
    END_SIM : in  std_logic;
    CLK     : out std_logic;
    RST_n   : out std_logic);
end clk_gen;

architecture beh of clk_gen is

  constant Ts : time := 10 ns;


  signal CLK_i : std_logic;


begin  -- beh

  process
  begin  -- process
    if (CLK_i = 'U') then
      CLK_i <= '0';
    else
      CLK_i <= not(CLK_i);
    end if;
    wait for Ts/2;
  end process;

  CLK <= CLK_i and not(END_SIM);
```

```
  process
  begin  -- process
    RST_n <= '0';
    wait for 3*Ts/2;
    RST_n <= '1';
    wait;
  end process;

end beh;
```

D.10 data_maker_new.vhd

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
--use ieee.std_logic_signed.all;
use ieee.numeric_std.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

entity data_maker is
  port (
    CLK     : in  std_logic;
    RST_n   : in  std_logic;
    VOUT    : out std_logic;
    DOUT    : out std_logic_vector(7 downto 0);
    DOUT1   : out std_logic_vector(7 downto 0);
    DOUT2   : out std_logic_vector(7 downto 0);
    END_SIM : out std_logic);
end data_maker;

architecture beh of data_maker is

  constant tco : time := 1 ns;

  signal sEndSim : std_logic;
  signal END_SIM_i : std_logic_vector(0 to 10);

begin  -- beh

  process (CLK, RST_n)
    file fp_in : text open READ_MODE is "./samples.txt";
    variable line_in : line;
    variable x,y,z : integer;
  begin  -- process
    if RST_n = '0' then                -- asynchronous reset (active low)
      DOUT <= (others => '0') after tco;
```

```
      DOUT1 <= (others => '0') after tco;
      DOUT2 <= (others => '0') after tco;
      VOUT <= '0' after tco;
      sEndSim <= '0' after tco;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if not endfile(fp_in) then
        readline(fp_in, line_in);
        read(line_in, x);
        DOUT <= std_logic_vector(to_signed(x, 8)) after tco;
        readline(fp_in, line_in);
        read(line_in, y);
        DOUT1 <= std_logic_vector(to_signed(y, 8)) after tco;
        readline(fp_in, line_in);
        read(line_in, z);
        DOUT2 <= std_logic_vector(to_signed(z, 8)) after tco;
        VOUT <= '1' after tco;
        sEndSim <= '0' after tco;
      else
        VOUT <= '0' after tco;
        sEndSim <= '1' after tco;
      end if;
    end if;
  end process;

  process (CLK, RST_n)
  begin  -- process
    if RST_n = '0' then                   -- asynchronous reset (active low)
      END_SIM_i <= (others => '0') after tco;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      END_SIM_i(0) <= sEndSim after tco;
      END_SIM_i(1 to 10) <= END_SIM_i(0 to 9) after tco;
    end if;
  end process;

  END_SIM <= END_SIM_i(10);

end beh;
```

D.11 data_sink.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

entity data_sink is
```

```vhdl
  port (
    CLK     : in std_logic;
    RST_n   : in std_logic;
    VIN     : in std_logic;
    DIN     : in std_logic_vector(7 downto 0);
    DIN1    : in std_logic_vector(7 downto 0);
    DIN2    : in std_logic_vector(7 downto 0);
        END_SIM : in std_logic);
end data_sink;

architecture beh of data_sink is

begin  -- beh

  process (CLK, RST_n)
    file res_fp : text open WRITE_MODE is "./results.txt";
    variable line_out : line;
  begin  -- process
    if RST_n = '0' then                 -- asynchronous reset (active low)
      null;
    elsif CLK'event and CLK = '1' then  -- rising clock edge
      if (VIN = '1') then
        write(line_out, to_integer(signed(DIN)));
        writeline(res_fp, line_out);
        write(line_out, to_integer(signed(DIN1)));
        writeline(res_fp, line_out);
        write(line_out, to_integer(signed(DIN2)));
        writeline(res_fp, line_out);
                if      (END_SIM = '1') then
                        file_close(res_fp);
                end if;
      end if;
    end if;
  end process;

end beh;
```