Kyle Keislar

ECE524/L FPGA/ASIC Design and Optimization Using VHDL Lab



Lab 2

Writing VHDL Testbenches Using File IO

09/22/2020

## 

## Introduction & Problem Statements

In this experiment the response of a CIC (Cascaded Integrator Combination) filter was observed for a set of frequencies. To observe this response, three sine waves will be passed through the filter at or below the Integrator block clock rate. The high level block diagram of the CIC filter is shown below in Fig 2.0.A. The filter’s frequency response can be seen in Figure 2.0.B

Fig 2.0.A: High Level CIC Block Diagram

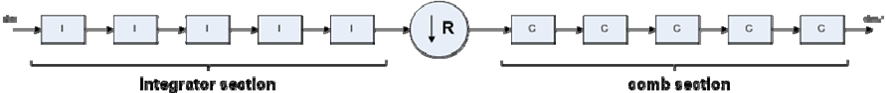
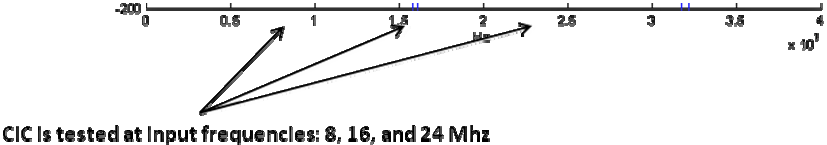
****

Fig 2.0.B: CIC Filter Response



The CIC filter will either suppress or amplify the input sine wave based on its frequency. To demonstrate the filter’s behavior, its response at 8, 16 and 24 MHz must be observed. The CIC response must be compared to each input to determine if the input is passed, suppressed or amplified.

The synthesized design of the CIC filter utilizes a provided design file. This filter design can be seen in the Appendix at item A.4 *cic.vhd*. The filter’s operating parameters were stipulated to be an integrator clock of 80 MHz and a differentiator clock at a fifth of this clock at 16MHz.

An implemented design of this filter was not realized using the Zedboard for this experiment.

## Procedure:

When simulating the CIC design, some constant operating parameters were used for this experiment. The filter clock was a constant 80 MHz, the differentiator section of the CIC filter required at decimated clock by a factor of 5 and was simulated with a 16 MHz clock accordingly.

Task 1: Prepare detailed block diagram of the design. A top level block design of the CIC filter can be seen in Appendix items A.1. Further levels of detail are represented in Appendix items A.2-A.5.

Task 2: Prepare input data value samples for all three cases done using Excel application software. These input data files were constructed by discretizing a sinusoid of frequency 8, 16 or 24 MHz over its defined period of 125ns, 62.5ns or 41.67ns respectively. For all default cases, the default input size was used of 18 bits. Each input and output *cic.vhd*

Task 3: Provide VHDL testbench for the design for all three cases:

3a) 8 MHz Sine Input

3b) 16 MHz Sine Input

3c) 24 MHz Sine Input

The VHDL test benches for each of these input frequencies can be seen below in the Appendix as items A.6 tb\_CIC\_8MHz.vhd, A.7 tb\_CIC\_16MHz.vhd and A.8 tb\_CIC\_24MHz.vhd. Each VHDL testbench instantiated an instance of the design found in *cic.vhd*.

## Results (Data):

## *Waveforms graphed from generated & simulated data:*

Fig 2.1: 8 MHz CIC Filter Response: Impulse & Band

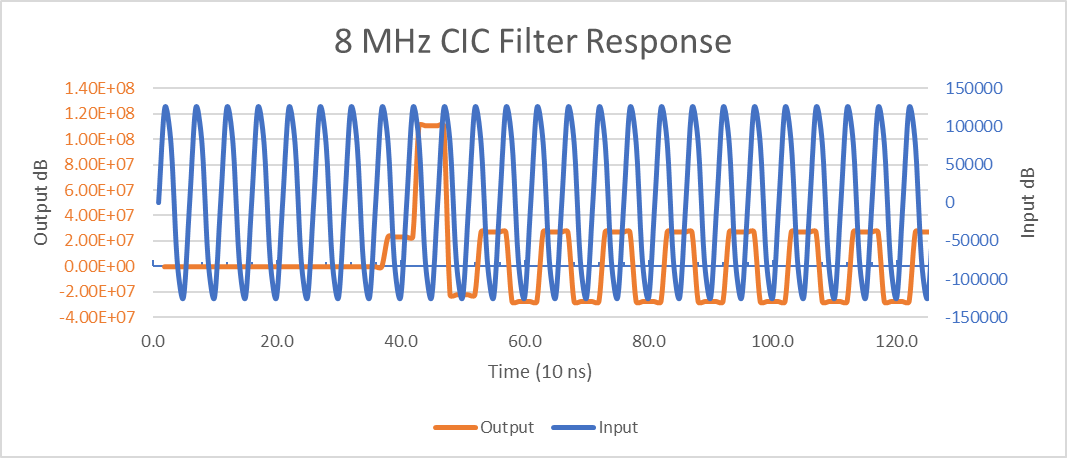


Fig 2.2: 16 MHz CIC Filter Response: Impulse

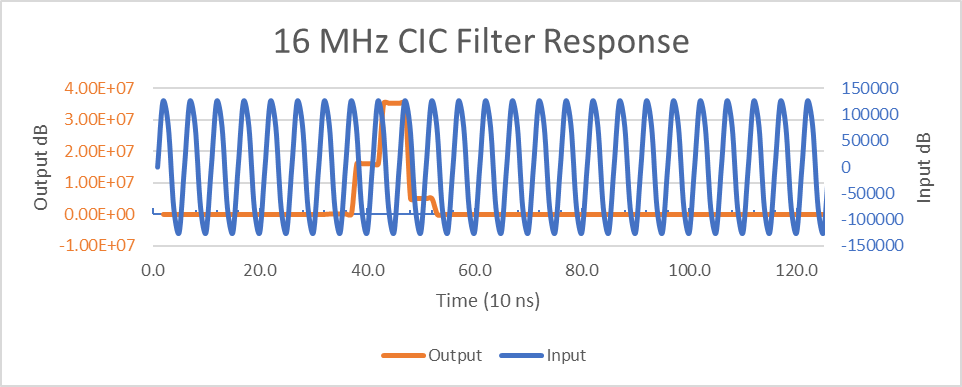


Fig 2.3: 16 MHz CIC Filter Response: Band

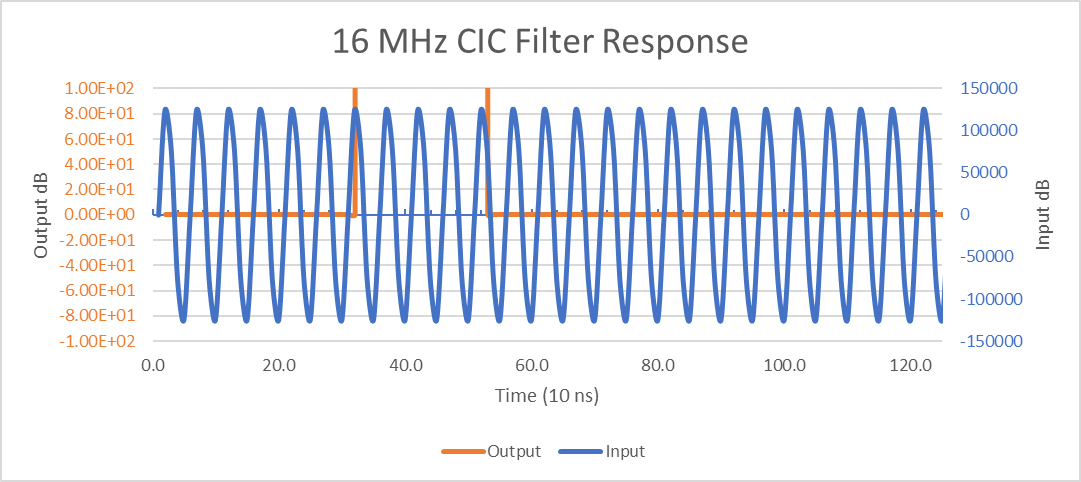


Fig 2.4: 24 MHz CIC Filter Response: Impulse

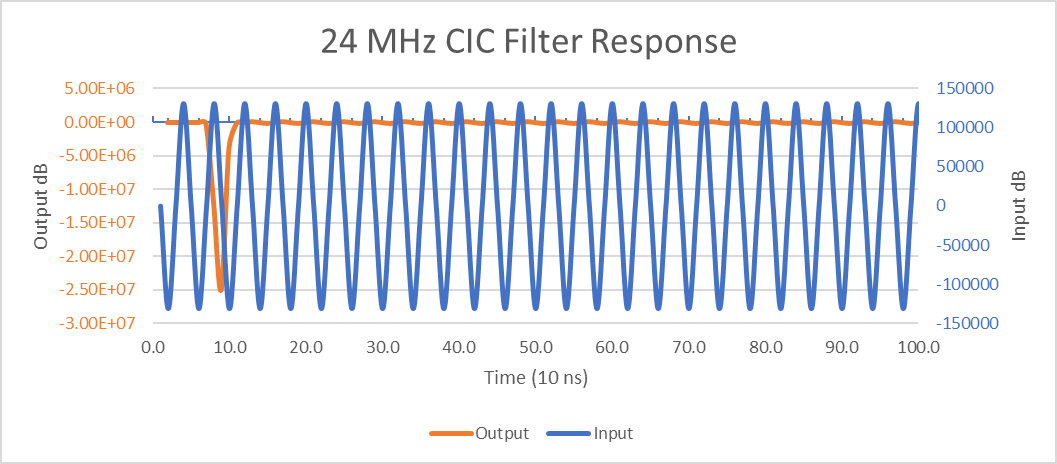
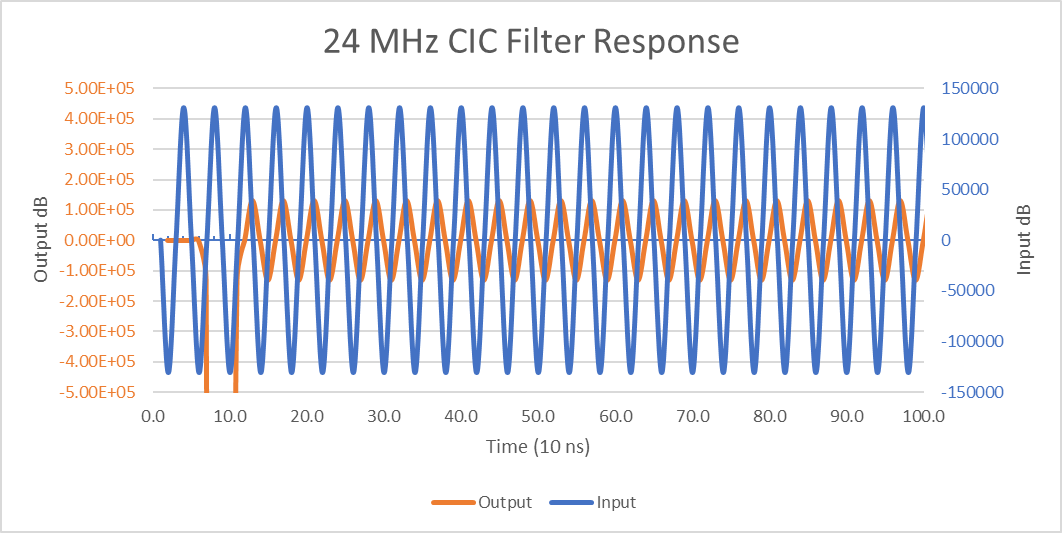


Fig 2.5: 24 MHz CIC Filter Response: Band



## Analysis:

The simulated design for this experiment successfully demonstrates the functionality of the proposed CIC filter described by Figure 2.0.A and is concurrent with the expected frequency response shown in Figure 2.0.B for all three cases.

Figure 2.1 shows the frequency response of the filter with a sinusoid input of 8 MHz frequency. The input waveform is in blue with its axis located on the left side of the graph, the output waveform is orange with its axis located on the right side of the graph. The CIC band output for the 8 MHz is sampled at such a rate that the sinusoid appears to be “clipped”. However the input is still passed through the CIC filter at the common input frequency. The 8 MHz impulse response can be seen at approximately 420ns. The impulse response does not appear to impact the waveform beyond the first input pulse. At this frequency, the output is delayed by approximately 400ns, this time reflects clocking parameters in the testbench as well as the filter clocking frequencies.

Figure 2.2 and 2.3 show the frequency response of the filter with a sinusoid input of 16 MHz frequency. In both graphs, the input waveform is in blue with its axis located on the left side of the graph, the output waveform is orange with its axis located on the right side of the graph. The CIC band output for the 16 MHz is sampled at such a rate that the sinusoid would be passed through the CIC filter at the common input frequency. However this output is suppressed from the input. The graphing tool used for this analysis was unable to detect any trace of sinusoidal behavior in the output. However, in realty no input can be perfectly suppressed. This infinitesimal change in output amplitude is cohesive with the expected frequency response shown in Figure 2.0.B. The 16 MHz impulse response can be seen at approximately 340ns. The impulse response does not appear to impact the waveform beyond the first input pulse. At this frequency, the output is delayed by approximately 340ns, this time reflects clocking parameters in the testbench as well as the filter clocking frequencies.

Finally, figure 2.4 shows the frequency response of the filter with a sinusoid input of 24 MHz frequency. The input waveform is in blue with its axis located on the left side of the graph, the output waveform is orange with its axis located on the right side of the graph. The CIC band output for the 24 MHz is sampled at such a rate that the sinusoid appears to be passed through the CIC filter at the common input frequency. This waveform resembles the input waveform amplified and out of phase. The 24 MHz impulse response can be seen at approximately 100ns. The impulse response does not appear to impact the waveform beyond the first input pulse. At this frequency, the output is delayed by approximately 100ns, this time reflects clocking parameters in the testbench as well as the filter clocking frequencies.

In each of these cases, the impulse response of the filter was many order of magnitude larger than the input waveform’s amplitude. The largest impulse response by magnitude was the 8 Mhz response of ~18 Million dB while the smallest by magnitude was the 24 MHz response of ~2.5 Million dB. From these observations it appears the impulse response is correlated to the input frequency and irrespective of the filter’s frequency response. In the 24 MHz case, the impulse response is 180◦ out of phase with those of the 8 and 24 MHz.

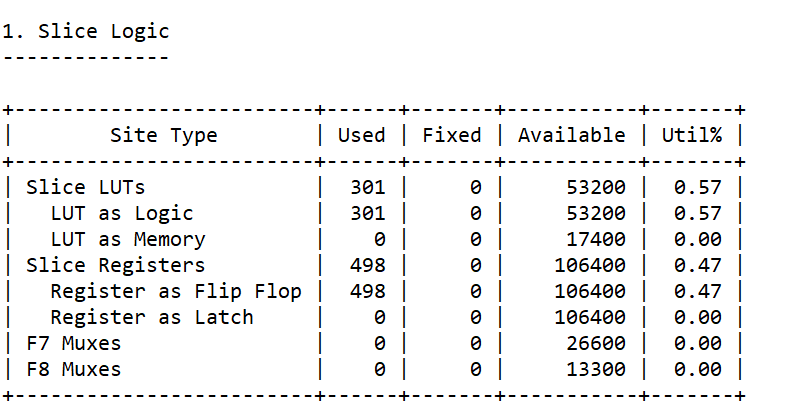
For the band pass of the filter, the output amplitude appears to be directly tied to the filter’s frequency response. At 8 MHz and 24 MHz, a sinusoidal input resulted in a sinusoidal output with exclusion of oversampling. However at 16 MHz, the sinusoidal input did not result in any detectable output. From the filter’s frequency response shown in Figure 2.0.B, the output characteristic of the filter can be extrapolated to suppress any inputs at a frequency range of ~16 MHz while passing those at 8 or 24 MHz. At 8 MHz, the filter input and output have nearly the same phase when accounting for the operational delay. At 24 MHz, there is less total operational delay due to more accurate sampling, resulting in both output and input waveforms having nearly the same phase.

APPENDIX

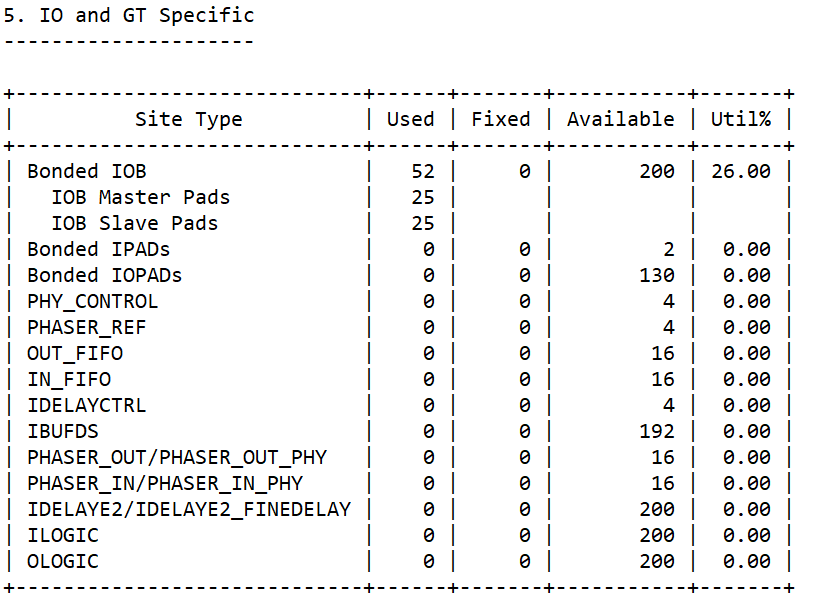
Questions

* 1) Obtain the resource utilization of the design and performance from the reports generated by Xilinx tools.

**Utilization report**

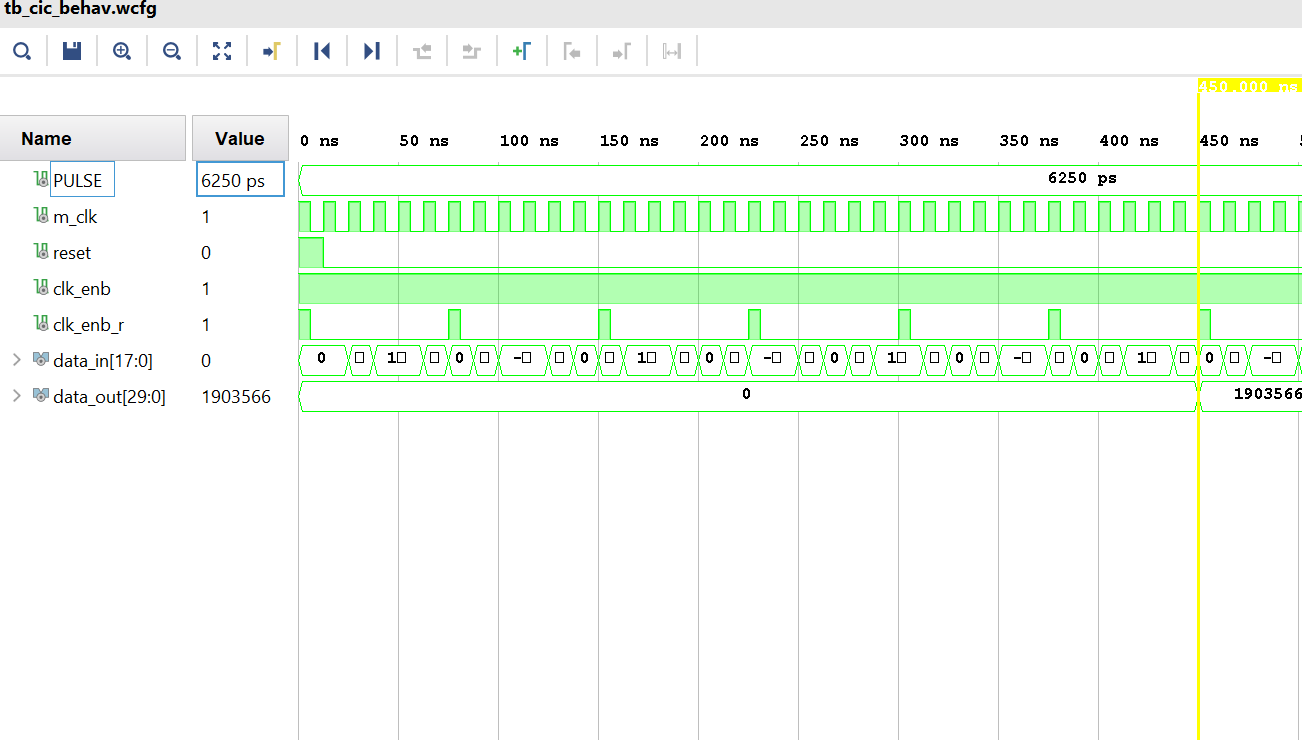


**I/O report**



* ~~2) What is the critical path delay for your design using timing simulation? Compare the value from the report with the one that results from the simulation waveform.~~
* 3) What is the latency of this design (number of clock cycles to the first output)? Prove your answer by showing your answer on the functional simulation waveform.

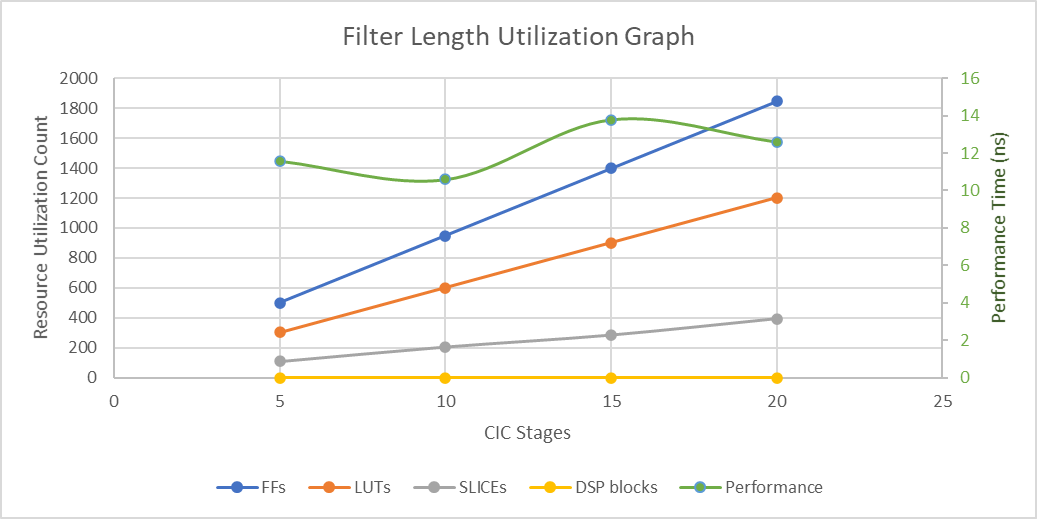
The maximum latency of this design is approximately 450 ns. This value changed based on the input waveforms frequency.



* 4) Change the parameters and fill out the following table for different cases. Alternatively you can provide Excel graphs that shows this information.

*Stages:*

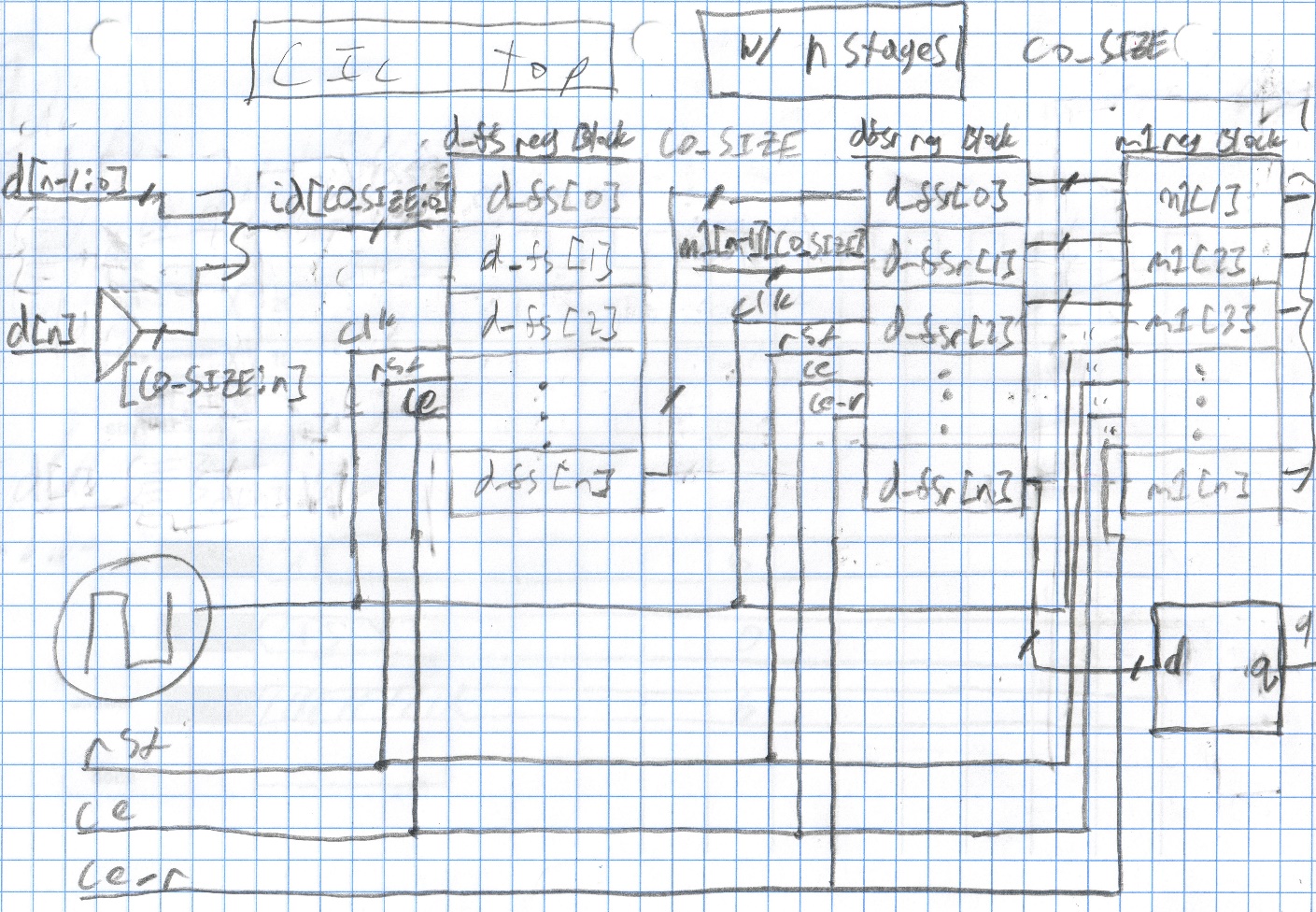
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Filter Length (STAGES)** | **FFs** | **LUTs** | **SLICEs** | **DSP blocks** | **Performance (ns)** |
| 5 | 498 | 301 | 107 | 0 | *11.587* |
| 0 | 948 | 601 | 204 | 0 | *10.594* |
| 15 | 1398 | 901 | 284 | 0 | *13.786* |
| 20 | 1848 | 1201 | 393 | 0 | *12.604* |



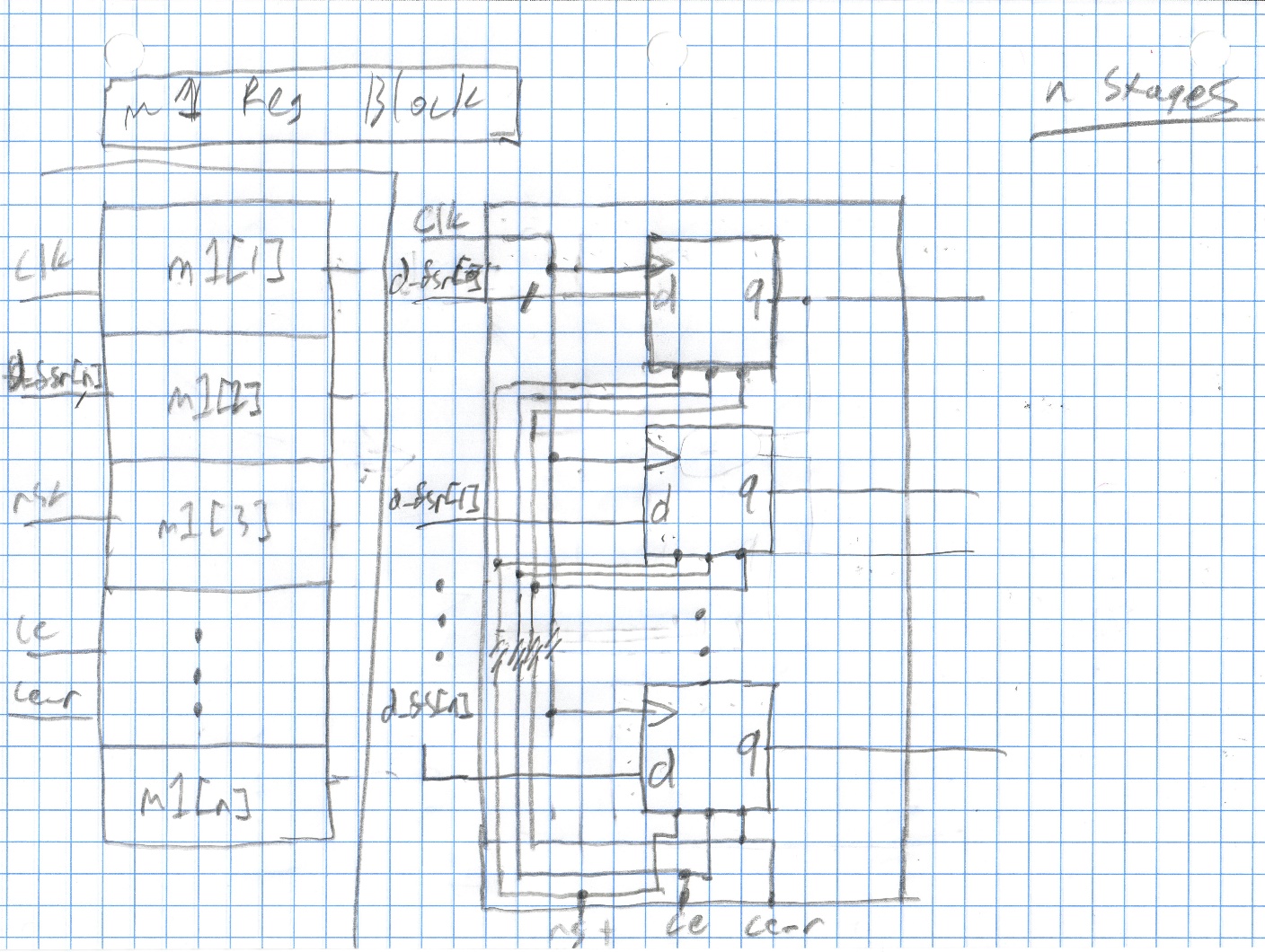
*Input Size:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input\_Width (CI\_SIZE)** | **FFs** | **LUTs** | **SLICEs** | **DSP blocks** | **Performance (ns)** |
| 18 | 498 | 301 | 107 | 0 | *11.587* |
| 20 | 500 | 301 | 112 | 0 | *9.012* |
| 22 | 502 | 301 | 118 | 0 | *8.982* |
| 24 | 504 | 301 | 140 | 0 | *10.31* |
| 26 | 506 | 301 | 117 | 0 | *9.44* |
| 28 | 508 | 301 | 118 | 0 | *10.139* |

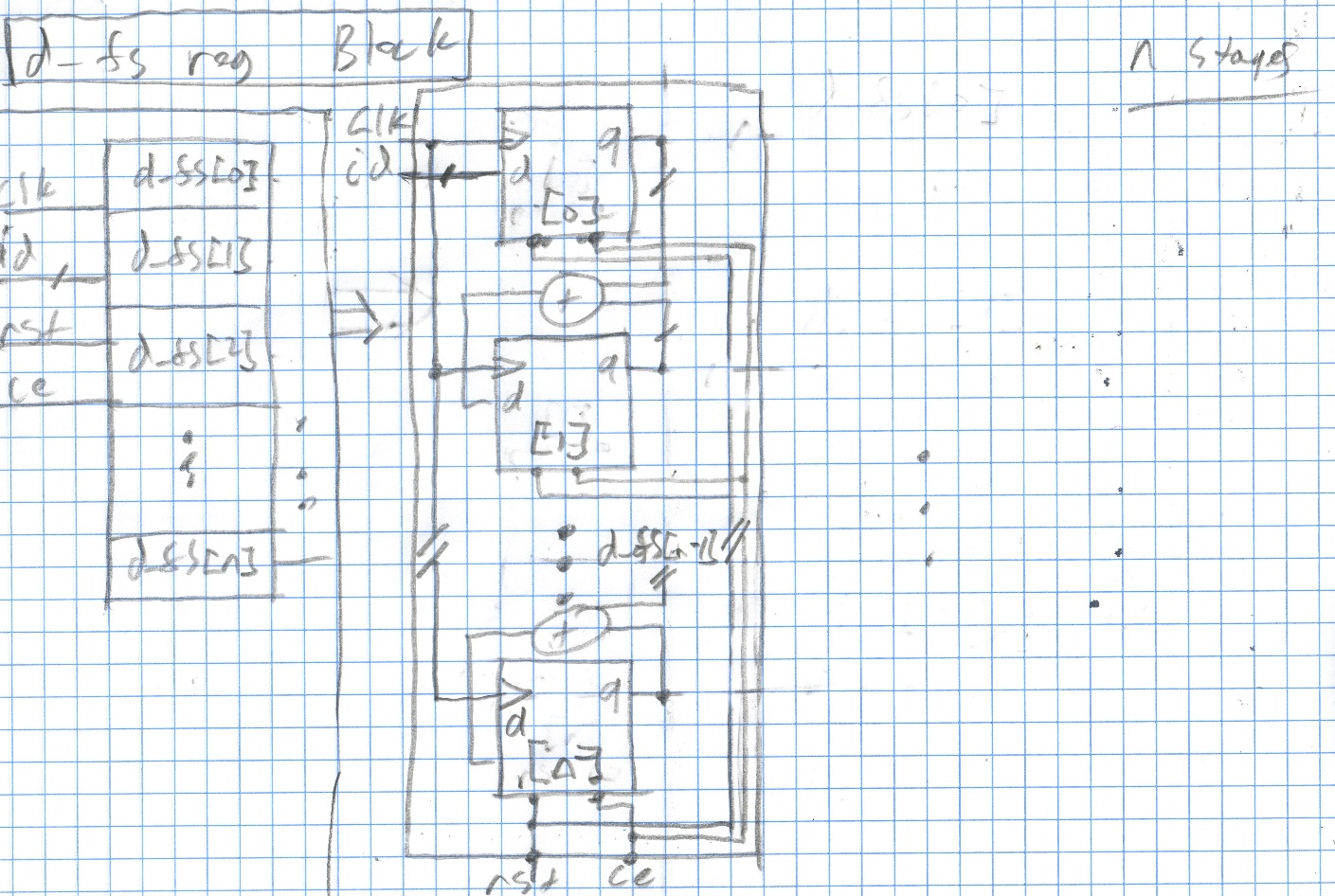
1. *CIC\_top\_Level block diagram*



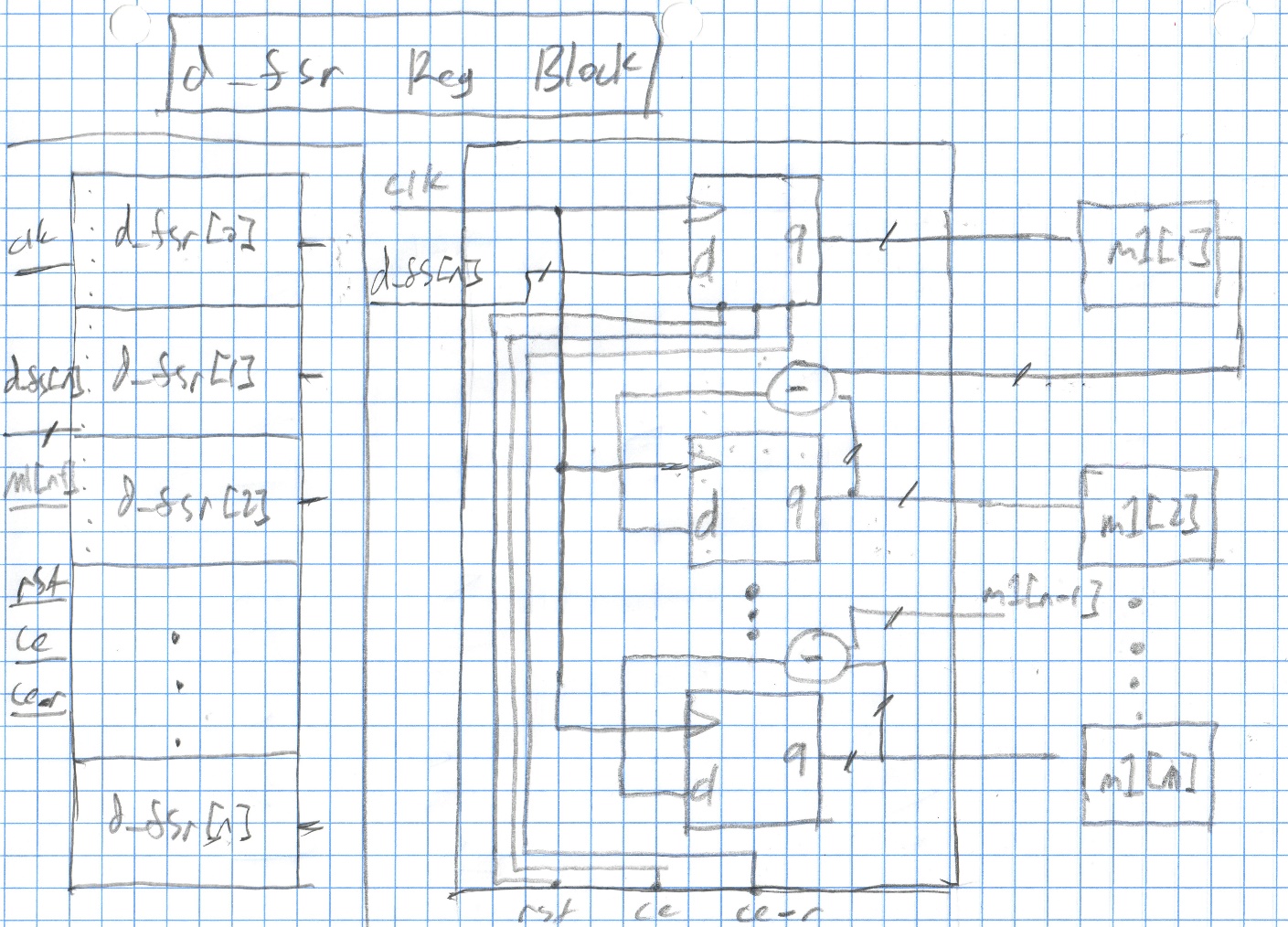
1. *M1\_Reg\_block diagram*



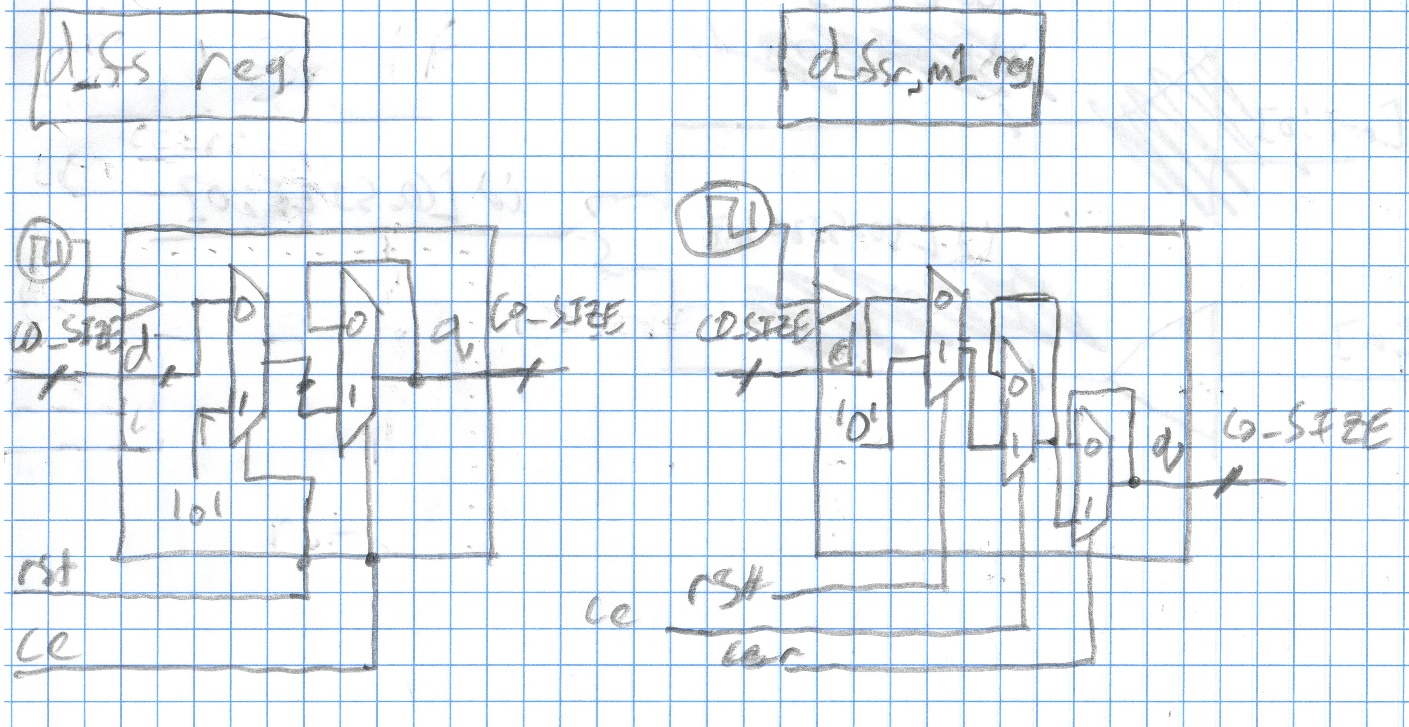
1. *D\_fs\_Reg\_block diagram*



1. *D\_fsr\_Reg\_block diagram*



1. *D\_fsr & D\_fs FF diagrams*



1. *Tb\_CIC\_8Mhz.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 09/18/2020 07:25:36 PM

-- Design Name:

-- Module Name: tb\_cic - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**use** std**.**textio**.all;**

**entity** tb\_cic **is**

-- Port ( );

**end** tb\_cic**;**

**architecture** Behavioral **of** tb\_cic **is**

--Constant Definitions

**CONSTANT** CP**:** TIME **:=** 12.5ns**;**--clock period

--8 MHz (80 MHz Sample) : 125 ns sample period (12.5 ns clock)

--16 MHz (80 MHz Sample) : 62.5ns sample period (12.5 ns clock)

--24 MHz (80 MHz Sample) : 41.67ns sample period (12.5 ns clock)

**CONSTANT** STAGES**:** INTEGER **:=** 5**;**--number of Filter Stages

--CONSTANT SAMP: INTEGER := 10;

--CONSTANT R\_VAL: INTEGER := SAMP/STAGES; --clock decimator, number of pulses to wait until ce\_r is high

**CONSTANT** CI\_SIZE **:** INTEGER **:=** 18**;**--filter data input size

**CONSTANT** CO\_SIZE**:** INTEGER **:=** 30**;** --filter data output size

--Signal Definitions

**SIGNAL** PULSE**:** TIME**:=**CP**\***0.5**;**--pulse width

**signal** m\_clk**,**reset**:** std\_logic **:=** '0'**;** --master clock, filter reset

**signal** clk\_enb**:** std\_logic **:=** '1'**;** --tied to 1 so that filter is always enabled

**signal** clk\_enb\_r**:** std\_logic **:=** '0'**;** --enables decimated clock pulse

**signal** data\_in**,**temp**:** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);**

**signal** data\_out**:** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**);**

**signal** r\_val\_cnt**:** integer **:=** 1**;**

**component** cic **is**

**generic(**CI\_SIZE **:** integer **:=** CI\_SIZE**;** -- cic input data width

CO\_SIZE **:** integer **:=** CO\_SIZE**;** -- cic output data width

STAGES **:** integer **:=** STAGES**);**

**port** **(** clk **:** **in** std\_logic**;** -- system clock (80 Mhz)

ce **:** **in** std\_logic**;** -- clock enable

ce\_r **:** **in** std\_logic**;** -- decimated clock by factor of 5 used in comb section

rst **:** **in** std\_logic**;** -- system reset

d **:** **in** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);** -- input data

q **:** **out** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**));** -- output data

**end** **component;**

**begin**

uut**:**cic

**Generic** **Map(** CI\_SIZE **=>** 18**,**

CO\_SIZE **=>** 30**,**

STAGES **=>** 5**)**

**Port** **Map** **(** clk **=>** m\_clk**,**

ce **=>** clk\_enb**,**

ce\_r**=>**clk\_enb\_r**,**

rst**=>**reset**,**

d**=>**data\_in**,**

q**=>**data\_out**);**

--Clock Processs

Rst**:process** --resets counters to 0

**begin**

reset **<=** '1'**;**

**wait** **for** CP**;**

reset **<=** '0'**;**

**wait** **for** CP**;**

**wait;**

**end** **process;**

m\_clock**:process** --free running clock

**begin**

m\_clk **<=** '1'**;**

**wait** **for** PULSE **;**

m\_clk **<=** '0'**;**

**wait** **for** PULSE**;**

**end** **process;**

ce\_r**:process**

**begin**

clk\_enb\_r**<=**'1'**;**

**wait** **for** pulse**;**

clk\_enb\_r**<=**'0'**;**

**wait** **for** CP**\*(**STAGES**-**1**)+**pulse**;**

**end** **process;**

rw\_i**:process(**m\_clk**)**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** TVs **:** text **open** read\_mode **is** "16\_mhz\_dat.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** read\_in**:** line**;** --reads in a line from the text file

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_from\_line**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

row\_cnt **:=** 0**;**

data\_in**<=** **(Others** **=>**'0'**);**

r\_val\_cnt**<=** 1**;**

**elsif(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**(**1**));** --read read line data from file into bit array

data\_in **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**(**1**),**CI\_SIZE**));**

**end** **if;**

**end** **process;**

rw\_o**:process**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** Output**:**text **open** write\_mode **is** "Lab\_2\_Output.txt"**;**

**variable** write\_in**:** line**;** --reads in a line from the text file

**variable** write\_line\_to\_file**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

**else**

-- --1

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --2

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

--3

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --4

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --5

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

--6

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --7

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

--8

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --9

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --10

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

**end** **if;**

**end** **process;**

**end** Behavioral**;**

1. *Tb\_CIC\_16Mhz.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 09/18/2020 07:25:36 PM

-- Design Name:

-- Module Name: tb\_cic - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**use** std**.**textio**.all;**

**entity** tb\_cic **is**

-- Port ( );

**end** tb\_cic**;**

**architecture** Behavioral **of** tb\_cic **is**

--Constant Definitions

**CONSTANT** CP**:** TIME **:=** 12.5ns**;**--clock period

--8 MHz (80 MHz Sample) : 125 ns sample period (12.5 ns clock)

--16 MHz (80 MHz Sample) : 62.5ns sample period (12.5 ns clock)

--24 MHz (80 MHz Sample) : 41.67ns sample period (12.5 ns clock)

**CONSTANT** STAGES**:** INTEGER **:=** 5**;**--number of Filter Stages

--CONSTANT SAMP: INTEGER := 10;

--CONSTANT R\_VAL: INTEGER := SAMP/STAGES; --clock decimator, number of pulses to wait until ce\_r is high

**CONSTANT** CI\_SIZE **:** INTEGER **:=** 18**;**--filter data input size

**CONSTANT** CO\_SIZE**:** INTEGER **:=** 30**;** --filter data output size

--Signal Definitions

**SIGNAL** PULSE**:** TIME**:=**CP**\***0.5**;**--pulse width

**signal** m\_clk**,**reset**:** std\_logic **:=** '0'**;** --master clock, filter reset

**signal** clk\_enb**:** std\_logic **:=** '1'**;** --tied to 1 so that filter is always enabled

**signal** clk\_enb\_r**:** std\_logic **:=** '0'**;** --enables decimated clock pulse

**signal** data\_in**,**temp**:** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);**

**signal** data\_out**:** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**);**

**signal** r\_val\_cnt**:** integer **:=** 1**;**

**component** cic **is**

**generic(**CI\_SIZE **:** integer **:=** CI\_SIZE**;** -- cic input data width

CO\_SIZE **:** integer **:=** CO\_SIZE**;** -- cic output data width

STAGES **:** integer **:=** STAGES**);**

**port** **(** clk **:** **in** std\_logic**;** -- system clock (80 Mhz)

ce **:** **in** std\_logic**;** -- clock enable

ce\_r **:** **in** std\_logic**;** -- decimated clock by factor of 5 used in comb section

rst **:** **in** std\_logic**;** -- system reset

d **:** **in** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);** -- input data

q **:** **out** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**));** -- output data

**end** **component;**

**begin**

uut**:**cic

**Generic** **Map(** CI\_SIZE **=>** 18**,**

CO\_SIZE **=>** 30**,**

STAGES **=>** 5**)**

**Port** **Map** **(** clk **=>** m\_clk**,**

ce **=>** clk\_enb**,**

ce\_r**=>**clk\_enb\_r**,**

rst**=>**reset**,**

d**=>**data\_in**,**

q**=>**data\_out**);**

--Clock Processs

Rst**:process** --resets counters to 0

**begin**

reset **<=** '1'**;**

**wait** **for** CP**;**

reset **<=** '0'**;**

**wait** **for** CP**;**

**wait;**

**end** **process;**

m\_clock**:process** --free running clock

**begin**

m\_clk **<=** '1'**;**

**wait** **for** PULSE **;**

m\_clk **<=** '0'**;**

**wait** **for** PULSE**;**

**end** **process;**

ce\_r**:process**

**begin**

clk\_enb\_r**<=**'1'**;**

**wait** **for** pulse**;**

clk\_enb\_r**<=**'0'**;**

**wait** **for** CP**\*(**STAGES**-**1**)+**pulse**;**

**end** **process;**

rw\_i**:process(**m\_clk**)**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** TVs **:** text **open** read\_mode **is** "16\_mhz\_dat.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** read\_in**:** line**;** --reads in a line from the text file

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_from\_line**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

row\_cnt **:=** 0**;**

data\_in**<=** **(Others** **=>**'0'**);**

r\_val\_cnt**<=** 1**;**

**elsif(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**(**1**));** --read read line data from file into bit array

data\_in **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**(**1**),**CI\_SIZE**));**

**end** **if;**

**end** **process;**

rw\_o**:process**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** Output**:**text **open** write\_mode **is** "Lab\_2\_Output.txt"**;**

**variable** write\_in**:** line**;** --reads in a line from the text file

**variable** write\_line\_to\_file**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

**else**

-- --1

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --2

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

--3

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --4

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --5

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --6

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --7

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

-- --8

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --9

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --10

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

**end** **if;**

**end** **process;**

**end** Behavioral**;**

1. *Tb\_CIC\_24Mhz.vhd*

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 09/18/2020 07:25:36 PM

-- Design Name:

-- Module Name: tb\_cic - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**use** std**.**textio**.all;**

**entity** tb\_cic **is**

-- Port ( );

**end** tb\_cic**;**

**architecture** Behavioral **of** tb\_cic **is**

--Constant Definitions

**CONSTANT** CP**:** TIME **:=** 12.5ns**;**--clock period

--8 MHz (80 MHz Sample) : 125 ns sample period (12.5 ns clock)

--16 MHz (80 MHz Sample) : 62.5ns sample period (12.5 ns clock)

--24 MHz (80 MHz Sample) : 41.67ns sample period (12.5 ns clock)

**CONSTANT** STAGES**:** INTEGER **:=** 5**;**--number of Filter Stages

--CONSTANT SAMP: INTEGER := 10;

--CONSTANT R\_VAL: INTEGER := SAMP/STAGES; --clock decimator, number of pulses to wait until ce\_r is high

**CONSTANT** CI\_SIZE **:** INTEGER **:=** 18**;**--filter data input size

**CONSTANT** CO\_SIZE**:** INTEGER **:=** 30**;** --filter data output size

--Signal Definitions

**SIGNAL** PULSE**:** TIME**:=**CP**\***0.5**;**--pulse width

**signal** m\_clk**,**reset**:** std\_logic **:=** '0'**;** --master clock, filter reset

**signal** clk\_enb**:** std\_logic **:=** '1'**;** --tied to 1 so that filter is always enabled

**signal** clk\_enb\_r**:** std\_logic **:=** '0'**;** --enables decimated clock pulse

**signal** data\_in**,**temp**:** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);**

**signal** data\_out**:** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**);**

**signal** r\_val\_cnt**:** integer **:=** 1**;**

**component** cic **is**

**generic(**CI\_SIZE **:** integer **:=** CI\_SIZE**;** -- cic input data width

CO\_SIZE **:** integer **:=** CO\_SIZE**;** -- cic output data width

STAGES **:** integer **:=** STAGES**);**

**port** **(** clk **:** **in** std\_logic**;** -- system clock (80 Mhz)

ce **:** **in** std\_logic**;** -- clock enable

ce\_r **:** **in** std\_logic**;** -- decimated clock by factor of 5 used in comb section

rst **:** **in** std\_logic**;** -- system reset

d **:** **in** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);** -- input data

q **:** **out** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**));** -- output data

**end** **component;**

**begin**

uut**:**cic

**Generic** **Map(** CI\_SIZE **=>** 18**,**

CO\_SIZE **=>** 30**,**

STAGES **=>** 5**)**

**Port** **Map** **(** clk **=>** m\_clk**,**

ce **=>** clk\_enb**,**

ce\_r**=>**clk\_enb\_r**,**

rst**=>**reset**,**

d**=>**data\_in**,**

q**=>**data\_out**);**

--Clock Processs

Rst**:process** --resets counters to 0

**begin**

reset **<=** '1'**;**

**wait** **for** CP**;**

reset **<=** '0'**;**

**wait** **for** CP**;**

**wait;**

**end** **process;**

m\_clock**:process** --free running clock

**begin**

m\_clk **<=** '1'**;**

**wait** **for** PULSE **;**

m\_clk **<=** '0'**;**

**wait** **for** PULSE**;**

**end** **process;**

ce\_r**:process**

**begin**

clk\_enb\_r**<=**'1'**;**

**wait** **for** pulse**;**

clk\_enb\_r**<=**'0'**;**

**wait** **for** CP**\*(**STAGES**-**1**)+**pulse**;**

**end** **process;**

rw\_i**:process(**m\_clk**)**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** TVs **:** text **open** read\_mode **is** "24\_mhz\_dat.txt"**;**-- "8\_mhz\_dat.txt"; --16\_mhz\_dat.txt -- 24\_mhz\_dat.txt

**variable** read\_in**:** line**;** --reads in a line from the text file

**variable** row\_cnt**:** integer **:=** 0**;**--row to read from

**variable** read\_from\_line**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

row\_cnt **:=** 0**;**

data\_in**<=** **(Others** **=>**'0'**);**

r\_val\_cnt**<=** 1**;**

**elsif(**m\_clk'**event** **AND** m\_clk**=**'1'**)** **then**

**if(not** **endfile(**TVs**))** **then**

row\_cnt **:=** row\_cnt **+** 1**;**

**readline(**TVs**,**read\_in**);** --read line from file

**end** **if;**

**read(**read\_in**,**read\_from\_line**(**1**));** --read read line data from file into bit array

data\_in **<=** std\_logic\_vector**(to\_signed(**read\_from\_line**(**1**),**CI\_SIZE**));**

**end** **if;**

**end** **process;**

rw\_o**:process**

**CONSTANT** COL**:** INTEGER **:=**1**;**

**type** t\_integer\_array **is** **array(**integer **range** **<>** **)** **of** integer**;**

**file** Output**:**text **open** write\_mode **is** "Lab\_2\_Output.txt"**;**

**variable** write\_in**:** line**;** --reads in a line from the text file

**variable** write\_line\_to\_file**:** t\_integer\_array**(**1 **to** COL**);**

**begin**

**if(**reset**=**'1'**)** **then** --if reset

**else**

-- --1

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --2

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

--3

write\_line\_to\_file**(**1**):=** **to\_integer(**signed**(**data\_out**));**

**write(**write\_in**,**write\_line\_to\_file**(**1**));**

**writeline(**Output**,** write\_in**);**

**wait** **for** CP**;**

-- --4

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --5

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

-- --6

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --7

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

-- --8

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --9

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

---- --10

-- write\_line\_to\_file(1):= to\_integer(signed(data\_out));

-- write(write\_in,write\_line\_to\_file(1));

-- writeline(Output, write\_in);

-- wait for CP;

**end** **if;**

**end** **process;**

**end** Behavioral**;**

1. *Cic.vhd*

-- Module Name: cic.vhd

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** cic **is**

**generic(**CI\_SIZE **:** integer **:=** 18**;** -- cic input data width

CO\_SIZE **:** integer **:=** 30**;** -- cic output data width

STAGES **:** integer **:=** 5**);**

**port** **(** clk **:** **in** std\_logic**;** -- system clock (80 Mhz)

ce **:** **in** std\_logic**;** -- clock enable

ce\_r **:** **in** std\_logic**;** -- decimated clock by factor of 5 used in comb section

rst **:** **in** std\_logic**;** -- system reset

d **:** **in** std\_logic\_vector **(**CI\_SIZE**-**1 **downto** 0**);** -- input data

q **:** **out** std\_logic\_vector **(**CO\_SIZE**-**1 **downto** 0**));** -- output data

**end** cic**;**

**architecture** syn **of** cic **is**

-- array definition for integrator and comb section

**type** d\_array\_type **is** **array** **(**STAGES **downto** 0**)** **of** std\_logic\_vector**(**CO\_SIZE**-**1 **downto** 0**);** -- l less stages than combination section

--width is selected such that array\_type+array\_type

--will be the size of d\_array\_type

-- array definition for comb section

**type** array\_type **is** **array** **(**STAGES **downto** 1**)** **of** std\_logic\_vector**(**CO\_SIZE**-**1 **downto** 0**);**

**signal** d\_fs **:** d\_array\_type**;** -- used in the integrator section

**signal** d\_fsr **:** d\_array\_type**;** -- used in the differentiator section, at rate r

**signal** m1 **:** array\_type**;** -- used in the differentiator section, at rate r

**signal** id **:** std\_logic\_vector**(**CO\_SIZE**-**1 **downto** 0**):=** **(others** **=>** '0'**);** -- to use for sign extended version of the input (sign extended d)

**begin**

-- output data

q **<=** d\_fsr**(**STAGES**);** --define output integrated response of input data

-- input data (d input is sign extended to 30 bits)

id**(**CO\_SIZE**-**1 **downto** CI\_SIZE**)** **<=** **(others** **=>** d**(**CI\_SIZE**-**1**));** --sign bits, extends msb

id**(**CI\_SIZE**-**1 **downto** 0**)** **<=** d**;** --data bits

-- integrator section ( shifts in sum of current + prevoious data)

**process(**clk**)**

**begin**

**if(**clk'**event** **and** clk **=** '1'**)** **then** -- when clocked

**if(**rst **=** '1'**)** **then** --on synchronous reset, clear integrator output for all d\_fs regs

d\_fs**(**0**)** **<=** **(others** **=>** '0'**);**

**for** i **in** 1 **to** STAGES **loop**

d\_fs**(**i**)** **<=** **(others** **=>** '0'**);**

**end** **loop;**

**elsif(**ce **=** '1'**)** **then** --if the filter clock is enabled

d\_fs**(**0**)** **<=** id**;** --then load in sign extended input to first d\_fs reg (d\_fs(0)

**for** i **in** 1 **to** STAGES **loop**

d\_fs**(**i**)** **<=** d\_fs**(**i**-**1**)** **+** d\_fs**(**i**);** --integrator output stage bit i gets sum of previous stage and current stage

**end** **loop;**

**end** **if;**

**end** **if;**

**end** **process;**

-- differentiator (comb) section ( shifts in difference of current - prevoious data) (reverse of Integrator)

**process(**clk**)**

**begin**

**if(**clk'**event** **and** clk **=** '1'**)** **then** -- when clocked

**if(**rst **=** '1'**)** **then** --on synchronous reset, clear differentiator output for all d\_fsr regs

--clear temp rate r to 0

d\_fsr**(**0**)** **<=** **(others** **=>** '0'**);**

**for** i **in** 1 **to** STAGES **loop**

m1**(**i**)** **<=** **(others** **=>** '0'**);**

d\_fsr**(**i**)** **<=** **(others** **=>** '0'**);**

**end** **loop;**

**elsif(**ce **=** '1'**)** **then** --if the filter clock is enabled

d\_fsr**(**0**)** **<=** d\_fs**(**STAGES**);** --grab the LSB of the input that has been passed through integrator section

**if** **(**ce\_r **=** '1'**)** **then** --if the comb filter section is clocked

**for** i **in** 1 **to** STAGES **loop** --integrate rest of input

m1**(**i**)** **<=** d\_fsr**(**i**-**1**);** --temp i gets integrated input i-1

d\_fsr**(**i**)** **<=** d\_fsr**(**i**-**1**)** **-** m1**(**i**);** --differentator output stage bit i gets difference between previous stage and current stage

**end** **loop;**

**else** --if the filter clock decimated by value r is not enabled

m1 **<=** m1**;** --hold current temp value (hold next differentiated bit)

**for** i **in** 1 **to** STAGES **loop**

d\_fsr**(**i**)** **<=** d\_fsr**(**i**);** --hold current integrator output

**end** **loop;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** syn**;**