Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 10, October - 2017

FIR Filter Implementation using Matlab Fdatool and Xilinx Vivado

Rajesh Kumar Dwivedi[1] and Raghav Dwivedi[2]
[1]Department of Physics, Christ Church College, Kanpur
[2]PG scholar, JIIT Noida

Abstract: **Finite impulse filter is a filter structure that can be implemented at almost any sort of frequency digitally. An FIR filter can be implemented using a series of delays, multipliers, adders to create filter output. Although FIR Filter has been implemented successfully using various software's but this paper is actually the implementation of n-tap or n-1 order FIR filter using Matlab Fdatool and Xilinx Vivado. The waveform on Xilinx Vivado is representation of output of FIR which we obtain On Matlab.**

Keywords: Xilinx Vivado, Matlab Fdatool, FIR Filter.

I. INTRODUCTION

Filters are signal conditioners each accept input signal and block some pre-specified frequency component and the output of the filter is the input signal minus the pre-specified frequency component. For example a typical phone line acts as a filter that limits frequencies considerably to the range of frequencies that the human can hear. A digital filter takes digital input, produce digital output and have digital components.

Fir filter are the filter with finite response, they are also called as the recursive filter. Basic diagram of Fir filter is shown under fig. 1.

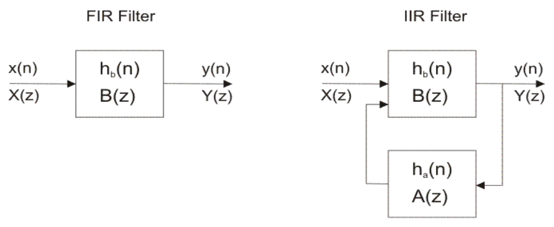


Fig.1

Representation of digital filters

It can be clearly seen iir filter has feedback while fir filter does not, thus fir filter are stable as compared to iir filter. Hence, fir filter are preferred over iir filter. In time domain digital filter is given by convolution sum:

$$Y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

An fir filter of order N is characterized by N+1 coefficients and in general requires N+1 multipliers

And N two input adders. The following figure below show the specification of various fir filters which are the basic block for the Fir filter specification (fig.2).

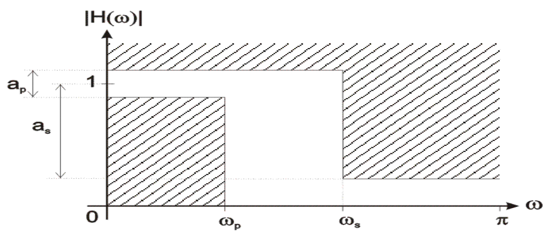


Fig.2: Low-pass digital filter specification

Where

- ω_p – normalized cut-off frequency in the pass band
- ω_s – normalized cut-off frequency in the stop band
- a_p – maximum ripples in the pass band
- a_s- minimum attenuation in the stop band

Similarly for the high-pass, band pass, band stop filter is shown in the following figures 3, 4, 5.

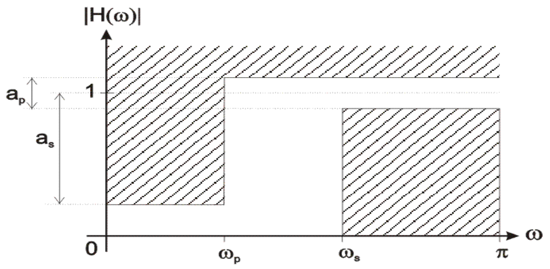


Fig.3: High pass digital filter specification

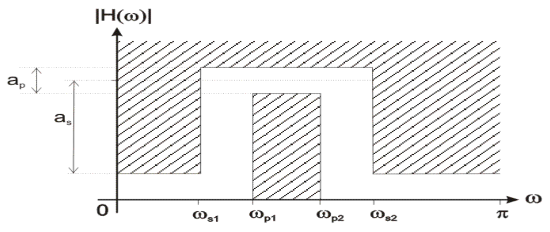


Fig.4: Band-pass digital filter specification

For the band stop filter specification please refer to next diagram as follows

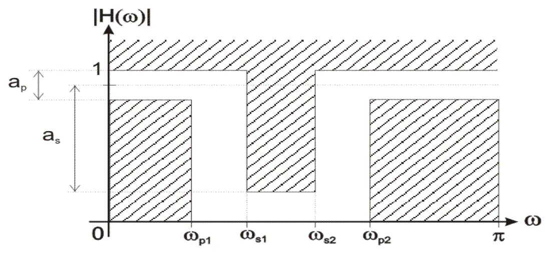


Fig.5: Band-stop digital filter Specification

The Z-transform is performed using with discrete-time signals. It converts a discrete time-domain signal into a frequency-domain representation. It is very suitable for analysing discrete time signals and systems in dsp architectures. The z-transform is derived from the Fourier discrete time transformation and is considered the basic operation in digital filter design process.

The Z-transform is defined as:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

Fir filter can be implemented using various structures which are explained in the following sections.

A. FIR DIRECT FORM

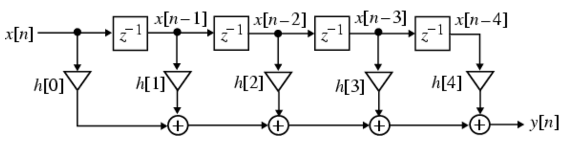


Fig.6: Direct form realization of fir filter

This is called direct form because it is direct implementation of the convolution, the number of delay is the order of filter, and hence this is also called canonical form. Fig.6 shows the structure of direct form fir, y[n] can be obtained from below given equation

$$y[n]=h[0]x[n]+h[1]x[n-1]+\ldots+h[n]x[n-N]=$$

$$\sum_{k=0}^{N} h[k]x[k-n]$$

B. FIR CASCADED DIRECT FORM

The transpose of the direct form shown above is represented in fig.7.

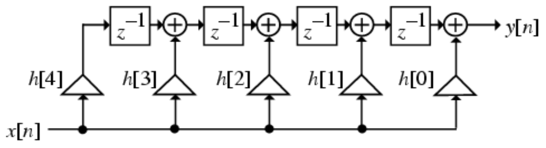


Fig.7: Cascade direct form realization of fir

C. FIR CASCADE FORM

A cascade realization of fir filter with N=6 is shown in fig.8

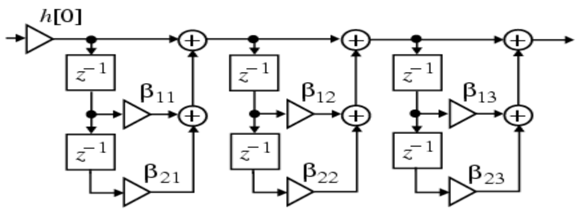


Fig.8: Cascade form realization of fir

Each second-order section in the above structure shown in figure 8 can also be realized in the transposed direct form

D. LINEAR PHASE FIR STRUCTURE

The symmetry property of a linear phase fir filter can be accomplished to reduce the number of multipliers into almost half of that in the direct form implementation. Consider a length-7 type 1 fir filter; its symmetric impulse response is given as:

$$H(z) = h[0]+h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[2]z^{-4} + h[1]z^{-5} + h[0]z^{-6}$$

Rewriting *H (z)* we get

$$H(z) = h[0](1 + z^{-6}) + h[1](z^{-1} + z^{-5}) + h[2](z^{-2} + z^{-4}) + h[3]z^{-3}$$

We obtain the realization as shown below in fig.9

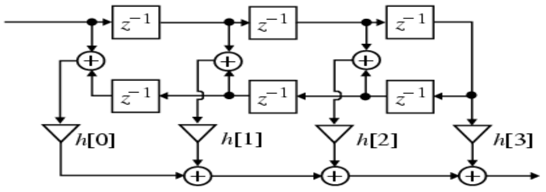


Fig. 8 Linear Phase fir structure

A similar decomposition can be applied to Type-2 Fir Filter transfer function.

II. IMPLEMENTATION OF FIR FILTER

The implementation of fir filter requires the basic three building blocks viz.

1) Multiplication
2) Adder
3) Signal delay

Building blocks of the fir filter can be shown in fig.9.

Multiplier output is *y[n] = βx[n]*

Adder output is *y[n]* = $x_1[n] + x_2[n]$

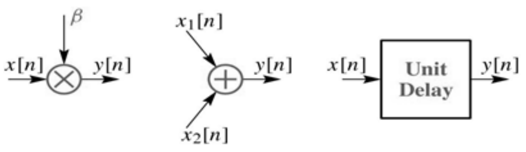


Fig.9: Basic building block of fir filter

Delay output is *y[n] = x [n-1]*

However DSP filter now uses MAC circuit for the multiplication followed by accumulation.

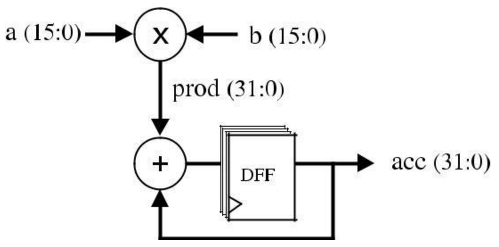


Fig.10: MAC Circuit

A typical MAC structure has been shown above in fig.10 it consists of multiplying two values, then adding the result to the previously accumulated value, which should be re-stored in the registers for future accumulations. Another feature of a MAC circuit is that it must check for overflow operation in the filter, when the number of MAC operations is large.

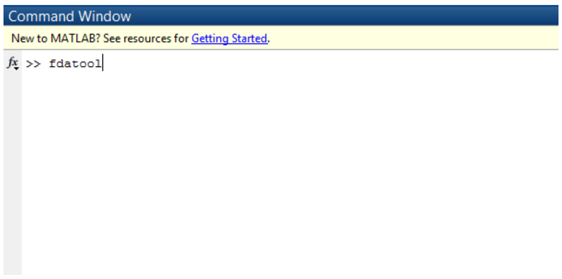
III. MATLAB FDATOOL

MATLAB FDATOOL is a filter design analysis tool used mainly to design the filter on MATLAB platform and then transforms it into corresponding VHDL code. In the following section are the steps how to configure the FIR filter on MATLAB.

III.A GETTING STARTED WITH FDATOOL

a) Type *fdatool or filterDesigner* at the MATLAB command prompt.

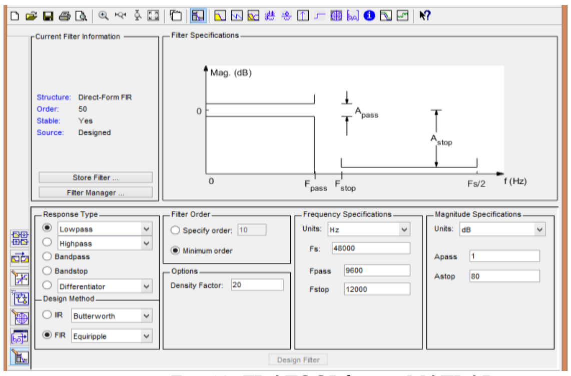
>> fdatool



```
Command Window
New to MATLAB? See resources for Getting Started.
fx >> fdatool
```

Fig. 11 fdatool on MATLAB command window

b) Filter design and analysis toolbox appear as shown in the figure 12

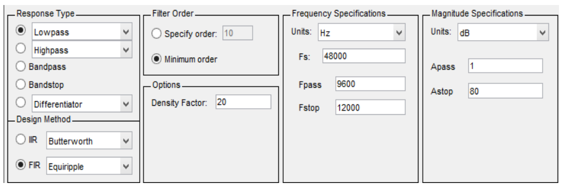


Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Response Type: Lowpass
Filter Order: Minimum order
Frequency Specifications: Units: Hz; Fs: 48000; Fpass: 9600; Fstop: 12000
Magnitude Specifications: Units: dB; Apass: 1; Astop: 80
Design Method: FIR Equiripple
Options: Density Factor: 20

Fig. 12: FDATOOL box in MATLAB

c) In the Filter Design & Analysis Tool dialog box, check that the following filter options are set:



Response Type: Lowpass
Design Method: FIR Equiripple
Filter Order: Minimum order
Options: Density Factor: 20
Frequency Specifications: Units: Hz; Fs: 48000; Fpass: 9600; Fstop: 12000
Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

Fig. 13: FDATOOL options in FDATOOL box

These settings are for the default filter design that the Filter Designer creates for you. If you do not have to change the filter, and design filter is grayed out, you are done and can skip to quantize the Filter.

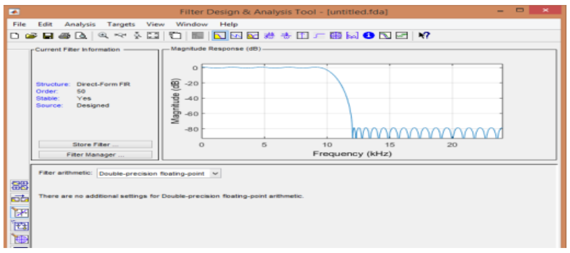
d) If you modified options listed in step c, click Design Filter. The Filter Designer creates a filter for the specified design and displays the following message in the Filter Designer status bar when the task is complete.

Designing Filter... Done.

Quantize filters for HDL code generation. To quantize your filter,

e) Open the basic FIR filter design you created in Design a FIR Filter in Filter Designer.

f) Click the Set Quantization Parameters button in the left-side toolbar. The Filter Designer displays a Filter arithmetic menu in the bottom half of its dialog box.



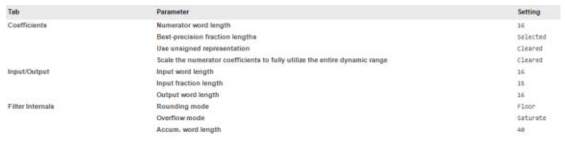
Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Filter arithmetic: Double-precision floating-point

There are no additional settings for Double-precision floating-point arithmetic.

Fig.14: Quantizing the Filter in MATLAB

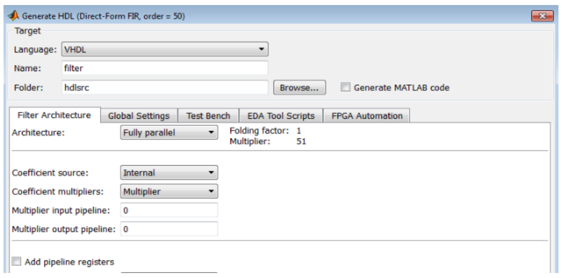
e) Set the quantization parameters as shown in the figure below



Tab	Parameter	Setting
Coefficients	Numerator word length	16
	Best-precision fraction lengths	Selected
	Use unsigned representation	Cleared
	Scale the numerator coefficients to fully utilize the entire dynamic range	Cleared
Input/Output	Input word length	16
	Input fraction length	15
	Output word length	16
Filter Internals	Rounding mode	Floor
	Overflow mode	Saturate
	Accum. word length	40

Fig. 15 Quantization parameters of the filter

g) After quantizing the filter we need to set the coder option and generate VHDL code from the MATLAB. Select target> generate HDL from filter designer dialog box.



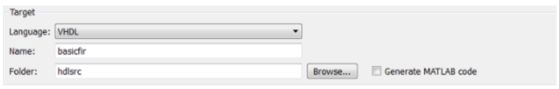
Generate HDL (Direct-Form FIR, order = 50)

Target
Language: VHDL
Name: filter
Folder: hdlsrc
Generate MATLAB code: ☐

Filter Architecture
Architecture: Fully parallel
Folding factor: 1
Multiplier: 51
Coefficient source: Internal
Coefficient multipliers: Multiplier
Multiplier input pipeline: 0
Multiplier output pipeline: 0
Add pipeline registers: ☐

Fig. 16 filter design box in MATLAB

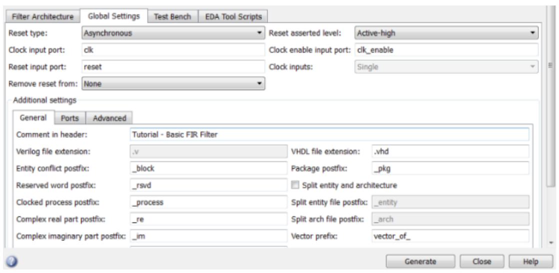
h) Change the default name in the target window to basicfir



Target
Language: VHDL
Name: basicfir
Folder: hdlsrc

Fig.17 Target window in MATLAB

i) Open the global setting tab of the filter and write Tutorial-Basic FIR filter in the comment in header text box

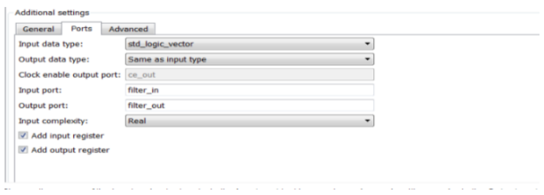


Reset type: Asynchronous
Reset asserted level: Active-high
Clock input port: clk
Clock enable input port: clk_enable
Reset input port: reset
Clock inputs: Single
Remove reset from: None
Additional settings

Comment in header: Tutorial - Basic FIR Filter
Verilog file extension: .v
VHDL file extension: .vhd
Entity conflict postfix: _block
Package postfix: _pkg
Reserved word postfix: _rsvd
Split entity and architecture
Clocked process postfix: _process
Split entity file postfix: _entity
Complex real part postfix: _re
Split arch file postfix: _arch
Complex imaginary part postfix: _im
Vector prefix: vector_of_

Fig. 18 Global setting box for Filter

j) Select the ports in the additional setting portion of the GUI.

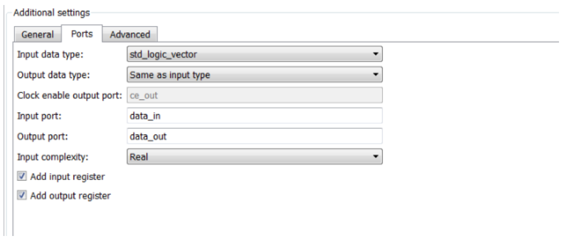


Additional settings

Input data type: std_logic_vector
Output data type: Same as input type
Clock enable output port: ce_out
Input port: filter_in
Output port: filter_out
Input complexity: Real
Add input register
Add output register

Fig. 19 Ports in the additional setting at the GUI

k) Replace filter_in with data_in and filter_out with data_out in the input port text box

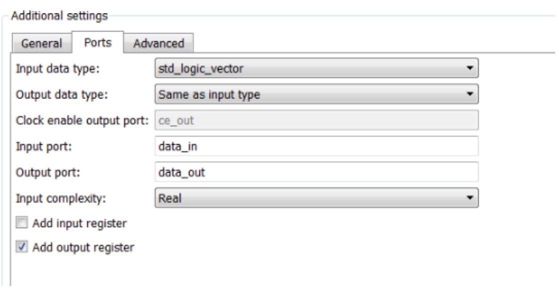


Additional settings

Input data type: std_logic_vector
Output data type: Same as input type
Clock enable output port: ce_out
Input port: data_in
Output port: data_out
Input complexity: Real
Add input register
Add output register

Fig. 20 Changes in the input port text box

l) Clear the add input register option from the input port text box then the box looks like as shown.

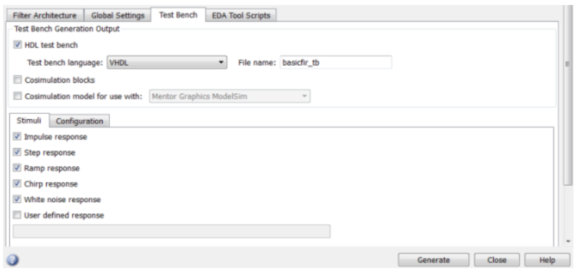


Additional settings

Input data type: std_logic_vector
Output data type: Same as input type
Clock enable output port: ce_out
Input port: data_in
Output port: data_out
Input complexity: Real
Add input register
Add output register

Fig. 21 Input port text box final look after following instruction

m) In the testbench tab, the file name box should have basicfir_tb as its entry onto it.

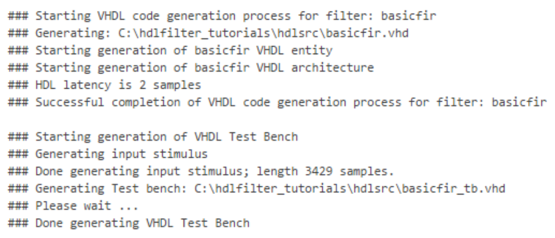


Test Bench Generation Output
HDL test bench
Test bench language: VHDL
File name: basicfir_tb
Cosimulation blocks
Cosimulation model for use with: Mentor Graphics ModelSim

Impulse response
Step response
Ramp response
Chirp response
White noise response
User defined response

Fig. 22 Testbench box in MATLAB

n) Click on generate button to start generate process. The coder displays the message as in the command window shown in the following figure.



```
### Starting VHDL code generation process for filter: basicfir
### Generating: C:\hdlfilter_tutorials\hdlsrc\basicfir.vhd
### Starting generation of basicfir VHDL entity
### Starting generation of basicfir VHDL architecture
### HDL latency is 2 samples
### Successful completion of VHDL code generation process for filter: basicfir

### Starting generation of VHDL Test Bench
### Generating input stimulus
### Done generating input stimulus; length 3429 samples.
### Generating Test bench: C:\hdlfilter_tutorials\hdlsrc\basicfir_tb.vhd
### Please wait ...
### Done generating VHDL Test Bench
```

Fig. 23 Message in Command window

o) After the successful generation of VHDL codes analyse it on XILINX VIVADO. The waveforms are shown in the next segment of this paper.

IV. ANALYSIS OF VHDL CODE ON XILINX VIVADO

The following VHDL code along with the testbench is generated using MATLAB FDATOOLS and the following code is analysed using Xilinx vivado the waveform obtained is then compared with the waveform posted on MATLAB tutorials. The waveform is shown below.

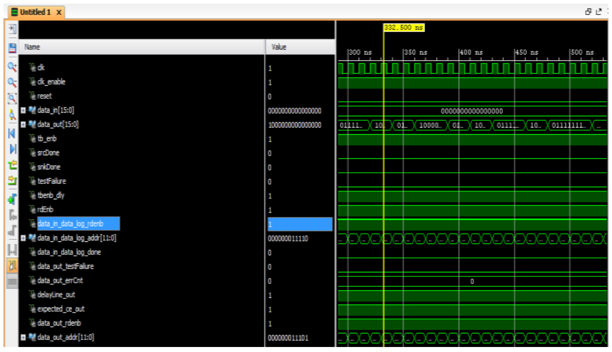


Fig.24 (a) Waveform-I on Xilinx vivado

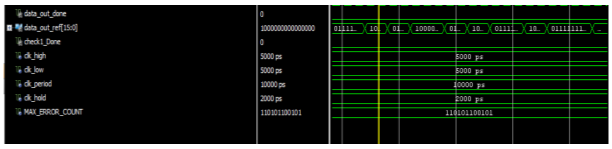


Fig.24 (b) Waveform-II on Xilinx vivado

RTL netlist of the following is generated as shown in the figure below, since the size of the filter is 51 tap so it's impossible to capture in the screenshot but still some glimpse of that 51 tap filter is shown here:

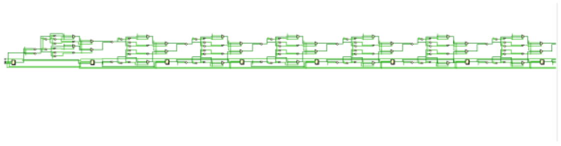
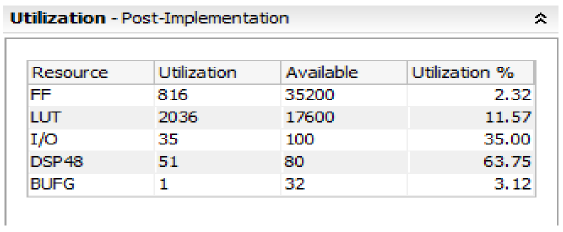


Fig. 25 RTL Netlist of the FIR Filter on Xilinx Vivado

The project summary of the following filter is shown below:



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
FF	816	35200	2.32
LUT	2036	17600	11.57
I/O	35	100	35.00
DSP48	51	80	63.75
BUFG	1	32	3.12

Fig. 26 post-implementation of fir filter

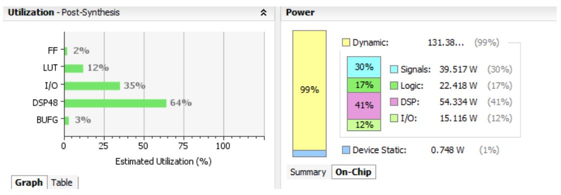


Fig. 27 Post-synthesis and power on-chip of FIR filter

Thus, the following FIR filter is successfully designed using MATLAB FDATOOLS and the XILINX VIVADO. The order of the filter is 50. This type of filter can be made using floating point, fixed point or even the distributed arithmetic coefficient and the MATLAB provides the platform for all the type of algorithm. FDATOOL provide the platform for the fixed point or the floating point coefficient, however for distributed arithmetic we have to follow some other procedure to configure it successfully.

However, with the help of following paper it's easy to recognize the use of MATLAB in transforming the MATLAB code to the VHDL code significantly, this paper will provide all the steps for designing the FIR filter. The filter order can vary for different design.

In the recent development the Xilinx platform has provide all the significant and necessary information for the designing of the filter.

V. FUTURE SCOPE AND APPLICATION

1) With the help of the MATLAB it's almost easy to design the FIR filter as it provide easy and suitable platform for the implementation of the FIR filter. MATLAB along with XILINX are helpful for the fast and error-free implementation of FIR filter.
2) MATLAB provide easy approach to generate VHDL code for any configuration and any order.
3) The following tutorial focuses on the practical aspects of FIR filter design. Practically, it is possible to design any FIR filter using MATLAB.
4) The coefficient in the FIR filter can be fixed-point or floating-point or even distributed arithmetic algorithm etc. can be implemented using MATLAB.
5) It's easy to configure and generate summary for the successful implementation of the FIR filter.
6) Time- efficient implementation of the FIR Filter is possible using FDATOOLS.

REFERENCES

[1] Practical FIR Filter Design in MATLAB, *Ricardo A. Losada, The MathWorks, Inc*, 3 Apple Hill Dr. Natick, MA 01760, USA January 12, 2004.
[2] Introduction to Signal Processing, Prentice Hall, *S. J. Orfanidis*, Upper Saddle River, 1996.
[3] Theory and Application of Digital Signal Processing, *L. R. Rabiner and B. Gold*, Prentice Hall, Englewood Cliffs, 1975.
[4] Design of computationally efficient interpolated FIR filters," *T. Saramaki, Y. Neuvo, and S. K. Mitra*, IEEE Trans. on Circuits and Systems, vol. 35, N0. 1,pp. 70-88, January 1988
[5] Exchange algorithms that complement the Parks-McClellan algorithm for linear-phase FIR filter design, *I. W. Selesnick and C. S. Burrus*, IEEE Trans. on Circuits and Systems II, 44(2):137-142, February 1997.
[6] Sakshat virtual labs, Digital FIR filter design and simulation, *NME-ICT initiative of MHRD*.
[7] MATLAB FDATOOLS, MATHWORKS tutorials
[8] Circuit design and simulation with VHDL, The MIT Press, *Volnei A. pedroni,2010*
[9] XILINX VIVADO tutorial, Xilinx
[10] VHDL Primer by *J.Bhaskar*
[11] Design and verification of FIR filter based on MATLAB and DSP, *Chen-Long Hu*, IEEE 2012 International Conference on Image Analysis and Signal Processing (IASP), 10.1109/IASP.2012.6425042