American Journal of Applied Sciences 8 (12): 1272-1281, 2011
ISSN 1546-9239

Hardware Approach of a Multipurpose Finite Impulse Response Filter for Real-Time Filtering Applications

[1]Md. Syedul Amin, [2]Md. Mamun and [1]Labonnah F. Rahman
[1]Department of Electrical, Electronic and Systems Engineering,
[2]Institute of Visual Informatics,
University Kebangsaan Malaysia, 43600, Bangi, Selangor, Malaysia

Abstract: Problem statement: Finite Impulse Response (FIR) filters are widely used in various DSP applications. The design of digital FIR filters is a very basic problem in digital signal processing. A FIR filter with multiple operation capability is certainly very useful for any real-time filtering applications. This article presents a multipurpose FIR filter design modeled by the hardware description language VHDL for real-time filtering application. **Approach:** The VHDL has its concept of concurrency to cope with the parallelism of digital hardware. The novel feature is the capability of the design to accomplish up to 127variable filter order and an arbitrary filter frequency response. The coefficients are calculated by Hamming windowing technique. Basing on selection embedded in the design, the model is able to execute highpass, lowpass, bandstop and bandpass filtering operations. It is set at 8-bit signed data processing. To filter the input data in time domain, Linear Constant Coefficient Difference Equation (LCCDE) is used by the filter. **Results:** The design outputs are validated through simulation and compilation. The output results are also compared with the MATLAB implemented calculated output results to test the correctness that proves the effectiveness of the design. **Conclusion:** With the capability of filtering signal in real time mode utilizing arbitrary filter shape, the multipurpose filter proves to be versatile.

Key words: Finite Impulse response (FIR) Filter, hamming window, VHDL, Linear Constant Coefficient Difference Equation (LCCDE), infinite impulse response, filter coefficients, multipurpose digital, execution instruction

INTRODUCTION

Frequency-sensitive linear filters can be divided into two categories: Finite Impulse Response filters (FIR) and Infinite Impulse Response filters (IIR). The design of FIR digital filters is a very basic problem in digital signal processing. As such, a lot of attention for the last 30 years has been received in this field. It is widely used in various DSP applications. Few examples are signal preconditioning, video convolution functions and communications. The FIR filter is chosen for applications which require linear phase or where not producing noise inside the filter is vital. True linear phase can be achieved only in an FIR filter where the impulse response is symmetric. Filters without noise can be achieved only with FIR filters. Because FIR filters can always be designed with a sufficient number of bits in the multipliers where truncation or rounding is not required after the multiplication. In the arena of digital FIR filters designing with the constant fixed-point binary coefficients, significant work has been done (Ma and Taylor, 1990; Lim and Liu, 1988; Dey and Oppenheim, 2008).

Ascertaining filter coefficients is the main task for designing a FIR filter. Usually window method and iterative method is applied for the determination of the coefficient. Iterative method, Ramez-Algorithm permits designing of optimal filters (Kumar *et al*., 2010; Mogaki *et al*., 2007; Parks and McClellan, 1972). Difficulty in implementation by HDL is the drawback of this method. On the contrary, it is easier to implement by the window method (Oppenhaim and Schafer, 1975).

For implementing the digital filtering algorithms, the most common approaches are special purpose digital filtering chips and Application-Specific

Corresponding Author: Md. Syedul Amin, Department of Electrical, Electronic and Systems Engineering, University Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia
Tel: +603-89216316 Fax: +603-89216146

Integrated Circuits (ASICs) for higher rates (Khoo *et al*., 1993; Laskowski and Samueli, 1992; Evans, 1993) or general purpose digital signal processing chips for audio applications.

The Field-Programmable Gate Arrays (FPGA) offers a potential substitute to accelerate the hardware implementation (Coussy *et al*., 2009; Marufuzzaman *et al*., 2010; Reaz *et al*., 2007b; Verma *et al*., 2009). FPGA has the merits of shorter design, higher density and lower cost cycle from the point of computer-aided design (Choong *et al*., 2005; Akter *et al*., 2008; ElGizawy *et al*., 2010). It comprises of a wide variety of building blocks. Each block consists of programmable look-up table and storage registers, where interconnections among these blocks are programmed through the hardware description language (Reaz *et al*., 2004b; 2005b; Iskandarani, 2010). This programmability and simplicity of FPGA made it favorable for prototyping digital system. FPGA allows the users to easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time frame for the hardware becomes shorter by using FPGA (Choong *et al*., 2006; Ibrahimy *et al*., 2006).

In this study we present a model of multipurpose FIR filter by hardware description language VHDL. The aim of the work is to get a proficient structure to ease hardware implementation to attain multi-purpose filtering with variable filter order and arbitrary frequency response and to assess the feasibility of using VHDL for prototyping and quick design. The utilization of Hardware Description Languages (HDL) is steadily increasing, as digital designs become more complex and larger (Pang *et al*., 2006). Previous methods like schematic capture have not been as well suited for reusing design and quick prototyping of large chip designs. Utilizing VHDL for modeling is attractive. Because, a formal description of the system is offered by VHDL. It also permits using definite description styles for covering up various abstraction levels (logic, register transfer and architectural level) used in the design (Reaz *et al*., 2006; 2007a). At first the problem is separated into small pieces in the computation of method. In VHDL, each can be considered as submodule. Synthesis is activated after the software verification of each submodule. It does the translations of HDL code into an equal digital cells' netlist. The synthesis facilitates to integrate the design work. It also gives a higher probability to explore far wider range of architectural substitute (Reaz *et al*., 2004a). This method offers a systematic approach for realization of hardware which allows quick prototyping of the multipurpose FIR filter.

MATERIALS AND METHODS

In this project, a multipurpose digital (FIR) filter is realized using VHDL. It can be modeled by utilizing LCCDE as given by Eq. 1:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{m=0}^{M} b_k y[n-m] \quad (1)$$

The FIR filter involves no feedback. As such, the LCCDE can be described by Eq. 2 as follows (Kumar *et al*., 2010; Mogaki *et al*., 2007; Parks and McClellan, 1972):

$$y[n] = \sum_{m=0}^{M} b_k y[n-m] \quad (2)$$

The FIR filter is a generalization of a running average function as we can find from the above equation. Whenever data fluctuates, averaging is usually done and then smoothed before interpretation. An M-point averaging model method is where each value of the output sequence is the sum of M consecutive input sequence multiplied by its coefficient, b_k (usually less than 1).

By using windowing technique, the multipurpose FIR filter is designed. It is the most suitable algorithm because it is the easiest way for FIR filter realization. By defining the piecewise function with discontinuities at the boundaries between bands, the frequency response can be achieved. By choosing predefine windows, the specifications of the filter can be easily found which matches filter specification. Then, the coefficients can be calculated by mathematical models.

The window selection is based on the stability and causality of all filter types, high pass, low pass, band stop and band pass for windowing algorithm. The specifications which are considered for the windows are the transition width, peak stop band attenuation and peak side lobe amplitude. A tradeoff between transition width and peak stop band attenuation remains. The transition width gets bigger as soon as the window gets higher stop band attenuation. As such, Hamming window which has a tradeoff between peak stop band and transition width attenuation is chosen.

MATLAB's fir2 function is utilized to determine the coefficient number. The interpolation point is set to 128 points and maximum order is limited upto 127 (7 bits) for VHDL realization. The frequency breakpoints and magnitude are divided into 101 sections. With the resolution of 0.01, each of them ranges from 0-1. Values are scaled up with a factor of 100 which is represented with 7 bits as VHDL is unable to function with floating point. To produce corresponding filter coefficients in real time, the filter specifications are supplied into filter model. To filter the signal, LCCDE is employed in time domain. A 256 points FFT generator is utilized to assist generation of the code to automate the design. The algorithm flow is illustrated in Fig. 1 for the filter.

The complication of multipurpose VHDL coding is in the filter coefficient calculation part. All codes are written in one file. Some codes are used again for the similar operation. The codes arrangements are illustrated in Fig. 2.

Process block is the main component for the architecture in the multipurpose FIR filter model. It handles the instruction execution flow as well as external and internal signal porting and filtering process. Like LCCDE concept used in the specific 15-order filter, the coefficients in multipurpose filter are initialized by computation of the filter coefficients basing on MATLAB's fir2 function. Figure 3 shows major part of the architecture.

In the process_block, the main execution instruction flow is dependent on enable_in port. The filter stays in the idle mode when it is "00", specifications are sent to the filter when "01", filter coefficients are calculated at "10" and input data are fed for the purpose of filtering at "11". In the code below, the IF/Else condition for main process is illustrated.

```
If (clock = '1' and enable_in = "00" and clock' event) then
    (idle mode)
ElsIf (clock = '1' and enable_in = "01" and clock' event) then
    (Obtain filter specifications)
Elsif(clock = '1' and enable_in = "10" and clock' event) then
    (computation of filter coefficients)
Elsif(clock = '1' and enable_in = "11" and clock' event) then
    (Filtering process)
End if;
```

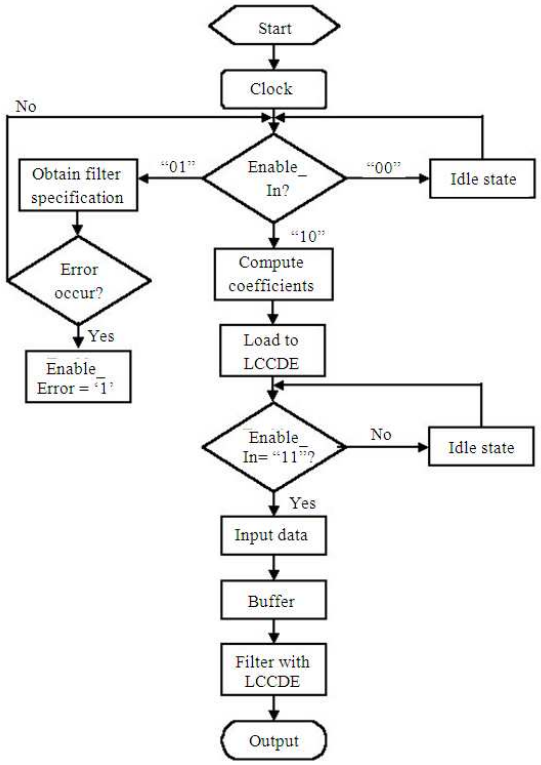

Fig. 1: The multipurpose filter algorithm flow

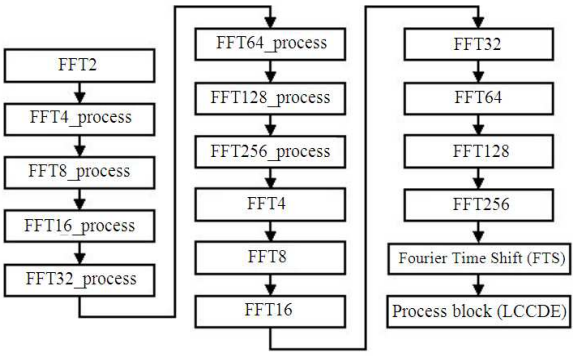

Fig. 2: VHDL code arrangement sequence

Since the filter is supposed to operate in the real time, a buffer is created for receiving the input data even though the filtering process has not started. The size of the buffer is 700 elements of integer type. When the enable_in is set to "01", the filter order is pumped into the filter serially once the clock is triggered from low to high. The maximum of filter order is set to 127 or 2^7 orders. After 7 cycles, when 7 bits are gathered by the buff3, the order is converted into integer as described in the code below.

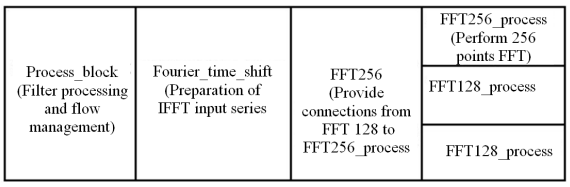

Fig. 3: The architecture inside the multipurpose FIR filter model

```
If counter_in2<=7 then
    buff3 (counter_in2):=order;
    Counter_in2:=counter_in2+1;
Elsif counter_in2=8then
    buff3 (0):='0';
    nn:= conv_integer (buff3(0-7));
End if;
```

At the same time buff1 and buff2 gathered bits of the magnitude and frequency breakpoints respectively. When 7 bits are gathered, the buffer values are converted to integer and assigned to the array aa and ff. To avoid the wrong integer value conversion, the Most Significant Bit (MSB) is set to 0. Hence, altogether first 8 bits are used during the conversion with the value obtained is always positive. The magnitude and frequency breakpoints received by the VHDL model is represented in the code below.

```
buff1 (counter_in1):=amp;
buff2 (counter_in1):=amp;
Counter_in1:=counter_in1+1;
If counter_in1>=8then
    aa(counter_array): conv_integer (buff1(0 to7));
    ff(counter_array): conv_integer (buff2(0 to7));
    buff1(1 to (700-8):= buff1((9) to700);
    buff2(1 to (700-8):= buff2((9) to700);
    counter_array:=counter_array+1;
    counter_in1:=1;
End if;
```

When the enable_in turns "10", values of the Hamming windows are computed. Since the cosine function cannot be synthesized, the cosine function is replaced with an array of constant values. The increment from 0-90° is stored in the constant. The values are up scalded by a factor of 1000 as illustrated in the code below.

```
Constant cosine: array91:=
Array91'(1000,1000,999,999,998,996,995,993,990,988,
985,982,978, 974, 970, 966, 961, 956, 951, 946, 940,
934, 927, 921, 914, 906, 899, 891, 883, 875, 866,
857,848, 839, 829, 819,809, 799, 788, 777, 766,755,
743, 731, 719, 707, 695, 682, 669, 656, 643, 629, 616,
602, 588, 574, 559, 545, 530, 515, 500, 485, 469, 454,
438, 423, 407,391, 375, 358, 342, 326, 309, 292, 276,
259, 242, 225, 208, 191, 174, 156, 139, 122, 105, 87,
70, 52, 35, 17, 0);
```

The degree involves division of the filter order but only power two denominator is synthesized. Since, every fraction is divided by the filter order, the nominator and denominator are both multiplied by 128, which is of power of two. Using the Case statement, an additional nominator is used for different order so every order is divisible by 128 to emulate the division of filter order as described below.

```
hamm3: =intorder;
    For counter in 0-127 loop
    if counter<=hamm3 then
    Case hamm3 is
        When 1=>hamm1:= 360 * counter * 128/128;
        When 2=> hamm1:= 360 * counter * 64/128;
        When 3=> hamm1:= 360 * counter * 43/128;
        When 4=> hamm1:= 360 * counter * 32/128;
        …          ...        ...        …
        When 128=>hamm1:= 360 * counter * 1/128;
        When others => NULL;
End case;
```

After obtaining the degree, it might vary from 0-360° but the cosine constant only ranges from 0-90°. Hence, trigonometry theorem is used so that the cosine value is obtained for the value of degree more than 90. The code to obtain the value of the Hamming window is given below.

```
hamm2: 1;
If hamm1>360 then
    Hamm1:=360;
End if;
        If (hamm1> 90 and (hamm1<180 or hamm1 =
180)) then
        hamm1:= 180-hamm1;
        hamm2:= -1;
Elsif (hamm1> 180 and (hamm1<270 or hamm1 =
270)) then
    hamm1:= hamm1-180;
    hamm2:= -1;
Elsif (hamm1> 270 and (hamm1<360 or hamm1 =
360)) then
    hamm1:= 360-hamm1;
    End if:
hamming1(counter):=(540000-(460*hamm2*
cosine(hamm1)))/1024
```

The difference in the frequency breakpoints are calculated through the codes below. If negative values are obtained, the enable_error port is set high indicating the occurrence of an error.

```
For counter in 0-127 loop
    If counter<= counter_array-2 then
    diff (counter):= ff (counter+1)-ff(counter);
    if diff(counter)<0 then
        enable_error <='1';
    End if:
 End if;
End loop;
```

Next, the filter's frequency response is interpolated to 128 points using same algorithm obtained from fir2 function as described below. The interpolation points are then assigned to the signals before they are ported to the FOURIER_TIME_SHIFT component.

```
Input129 (counter2): = (11*inc *aa(counter 1+1)+11*
    (1000-inc)* aa(counter1))/1024;
Case counter2 is
    When 1=> temp1+0<= input129 (1);
    When 2=> temp1+1<= input129 (2);
    …          …       …..
    When 127=> temp1+126<= input129 (127);
    When 128=> temp1+127<= input129 (128);
End case;
temp1_128<=input 129 (129);
```

In the FOURIER_TIME_SHIFT component, interpolation points are transformed into complex points by multiplying with a complex exponential value similar to the algorithm in fir2 function. Hence, sine and cosine are constant and are declared the same way as declaration of the cosine function earlier. Since, these values are inputted to the fast Fourier transform operation, they are conjugated first so inverse FFT operation is emulated through the codes below.

```
Case counter is
When 0=>        temp_0R<=(temp1*IN_0*
cosine(angle))/1024;
                temp_0I<=((-1)*temp2*IN_0
sine(angle))/1024;
When 1=>        temp_1R<=(temp1*IN_1*
cosine(angle))/1024;
    :           temp_1I<=((-1)*temp2*
IN_1*sine(angle))/1024;
    :           temp_255R<=(temp1*IN_1*
cosine(angle))/1024;
    :           temp_255I<=(temp2*IN_1*
sine(angle))/1024;
    :           …..     …       ...
when 127        temp_127R<=(temp1*IN_127*
cosine(angle))/1024;
                temp_127I<=((-1)*temp2*IN_127*
sine(angle))/1024;
                temp_129R<=(temp1*IN_127*
cosine(angle))/1024;
                temp_129I<=(temp2*IN_127*
sine(angle))/1024;
when 128        temp_128R<=(temp1*IN_128*
sine(angle))/1024;
                temp_128I<=((-1)*temp2*IN_128*
cosine(angle))/1024;
    when others=> null;
End case;
```

When a sequence of data is inputted to the FFT computation, the order of the inputs reshuffled first. Shuffling of input data is performed where the order has to correspond to the bit-reversed indexing of the original sequence. Hence, when the complex series is fed to the 256-point FFT computation, the porting is done according to the bit-reversed indexing as shown below:

```
IN_0R=> temp_0R,
IN_0I=> temp_0I,
IN_1R=> temp_128R,
IN_1I=> temp_128I,
    …          …
IN_254R=> temp_127R,
IN_254I=> temp_127I,
IN_255R=> temp_255,
IN_255I=> temp_255I,
```

After computing the FFT, the results are conjugated to obtain the IFFT series. Since the real part of the results is needed to compute the filter coefficients, conjugation of the results is redundant. Then, the real output is multiplied with Hamming window values to obtain the filter coefficients through the codes below. The number of coefficients is equal to the value filter order plus one.

```
For counter in 0-127 loop
    If counter <=hamm3
    Case counter is
        When 0=>        hamming2(0):=temp2_0*
hamming1(0)/262144;
        When        1=>hamming2(1):=temp2_1*
hamming1(1)/262144;
```

```
        …                ...        ...        …
        When 126=>   hamming2(126):=temp2_126*
hamming1(126)/262144;
        When 127=>   hamming2(127):=temp2_127*
hamming1(127)/262144;
End case;
    End if;
End loop;
```

The computation of the filter coefficients take about 20 cycles. After the 20 cycles, the enable_in is set to "11". Then, the inputs that are stored in the buffer are recalled and the filtering process begins. The concept filtering is similar to the specific 15-order filter except that the filter coefficients are variables. The LCCDE equation used for the filtering purpose through the codes are given below. After computing the filtered data, they are sent to the multipurpose FIR filter output port to be received by other devices.

```
If (enable_in="11") then
    Input1(0):=(arraydata(0)-128);
    Qoutput:=
    ((hamming2(0)) * (input1(0)))/1024+
    ((hamming2(1)) * (input1(1)))/1024+
    ((hamming2(2)) * (input1(2)))/1024+
    …                    ….
    ((hamming2(125)) * (input1(125)))/1024+
    ((hamming2(126)) * (input1(126)))/1024+
    ((hamming2(127)) * (input1(127)))/1024+
End if;
```

RESULTS AND DISCUSSION

To verify the performance and functionality, the multipurpose FIR filter is simulated utilizing VHDL testbench. As the stimuli to the VHDL model consists of a huge data, it is extracted from a text file. For file manipulation, the std.textio.all library is utilized. As such, the data is read line by line from the text file. Then it is fed into the model. The output of the filter is written in another text file after filtering. Together with the signal, the location of the output and input files are declared. Signals from testbench which ported to VHDL model under the test described by VHDL code are shown below:

```
UUT: FIRFilter
    port map
        (order =>order,
        freq=>freq,
        amp=>amp,
        enable_in=>enable_in,
        clock=>clock,
        input=>input,
        output=>output
        );
```

To find out the VHDL model's performance, the output frequency response using MATLAB and VHDL simulations are analyzed and compared. The specification of an order 20, 16, 18 and 20 for Hamming window band stop, high pass, low pass and band pass filter respectively is inserted in the VHDL model for generating filter coefficients. Then the coefficients are further compared with the results generated from the MATLAB for verification. In Table 1, the specifications are given.

The mulitpurpose filter model operating frequency is 1MHz with a period of 1μ sec. Using an external clock, the frequency is pumped in the model. Effectively the square wave duty cycle is 0.5 with a 0.51μ sec period. The output results also operate with the clock at the same frequency. Changes in both input and output occur on the waveform during the transition as only the clock transition from low to high triggers the filter executions.

With 1 μ sec period, each bit of input signal is fed in the model. To get rid of misreading from the previous bit value, a delay of 0.05 μ sec in the bit stream is used. As given in Fig. 4, at the beginning of simulation, the enable_in is pumped with "01". The frequency break points and magnitude are fed to the corresponding ports for the duration of state "01". Thereafter the enable_in is changed to state "10". The calculation time for the filter coefficients at the state "10" is approximately 20 cycles as given in Fig. 5. Hence, the delay is introduced to the enable_in port before the state "11" is introduced to prompt the start of filtering process shown in Fig. 6. The coefficients generated by the multipurpose filter for band stop, high pass, low pass and band pass in the MATLAB and VHDL model are shown in Table 2. It is to be noted that the VHDL coefficients model are scaled up by 1000.

The scaled up filters' coefficients can be clearly viewed from Table 2. These are fairly close to the coefficients generated by the MATLAB. The computed values differ slightly for the approximation and truncation errors. As such, it affects the accuracy of the filter slightly when these coefficients are utilized.

The analytical comparison of the filtering is calculated by examining the signal's frequency response of VHDL and MATLAB model which is given in Fig. 7 and 8.

Table 1: The specifications for low pass, high pass, band pass and band stop

	Low pass filter	High pass filter	Band pass filter	Band stop filter
Order of filter	20	18	16	20
Frequency break Points	0, 0.348, 0.352, 1.0	0, 0.33, 0.34, 1.0	0, 0.44, 0.45, 0.50, 0.51, 1.0	0, 0.44, 0.45, 0.50, 0.51, 1.0
Magnitude of the break points	1, 1, 0, 0	1, 1, 0, 0	0, 0, 1, 1, 0, 0	1, 1, 0, 0, 1, 1
Magnitude of the break points		0, 0, 1, 1		

Table 2: Coefficients generated by VHDL and MATLAB

Low pass filter		High pass filter		Band pass filter		Band stop filter	
VHDL	MATLAB	VHDL	MATLAB	VHDL	MATLAB	VHDL	MATLAB
-3	-0.0030	0	0.0001	6	0.0032	-3	-0.0028
-3	-0.0031	-4	-0.0037	1	0.0005	4	0.0037
0	0.0003	-6	-0.0074	38	0.0209	10	0.0127
15	0.0156	0	-0.0002	-7	-0.0041	-7	-0.0083
52	0.0489	27	0.0252	-26	-0.0223	-33	-0.0381
102	0.0969	39	0.0429	-7	-0.0050	15	0.0200
152	0.1456	2	0.0005	-287	-0.2728	49	0.0541
180	0.1765	-128	-0.1227	15	0.0092	2	0.0028
174	0.1765	-259	-0.2683	538	0.5420	933	0.9395
137	0.1456	662	0.6660	7	0.0092	-4	-0.0049
84	0.0969	-267	-0.2683	-288	-0.2728	49	0.0541
38	0.0489	-124	-0.1227	-6	-0.0050	12	0.0179
10	0.0156	1	0.0005	-26	-0.0223	-33	-0.0381
0	0.0003	40	0.0429	-6	-0.0041	-7	-0.0083
-2	-0.0031	26	0.0252	38	0.0209	10	0.0127
-3	-0.0030	0	-0.0002	1	0.0005	4	0.0037
		-6	-0.0074	6	0.0032	-3	-0.0028
		-4	-0.0037				
		0	0.0001				

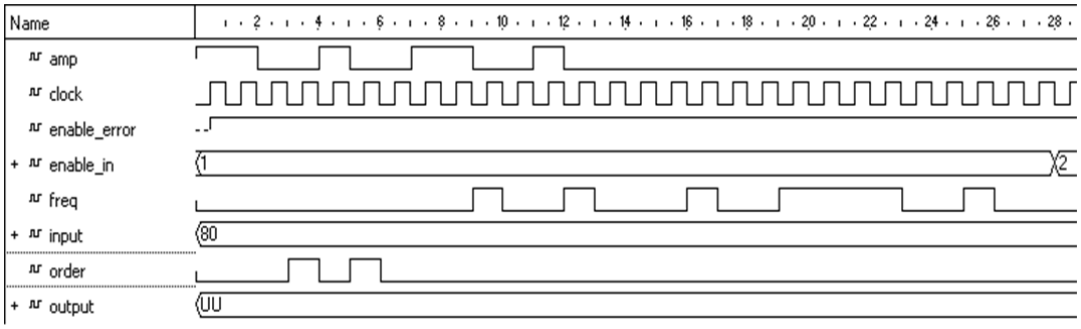

Fig. 4: The simulation waveform of the initialization of VHDL model multipurpose FIR filter

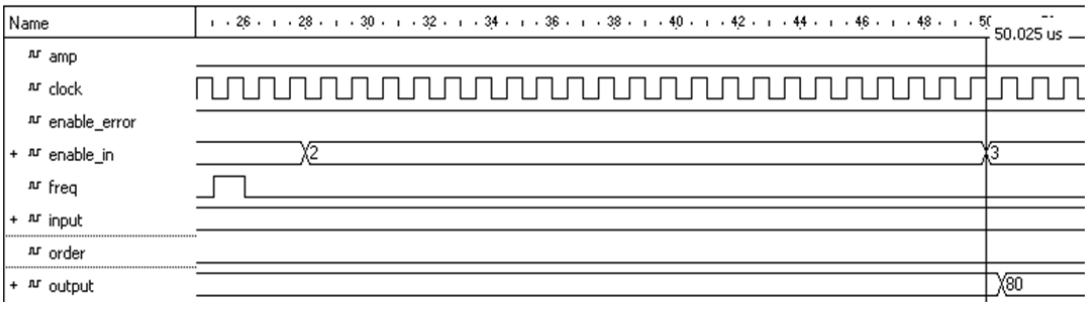

Fig. 5: The simulation waveform of the coefficient computation of VHDL model multipurpose FIR filter

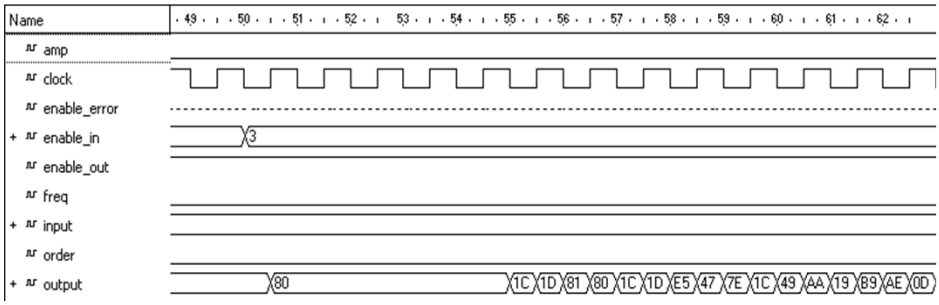

Fig. 6: The simulation waveform of the filtering process of VHDL model multipurpose FIR filter

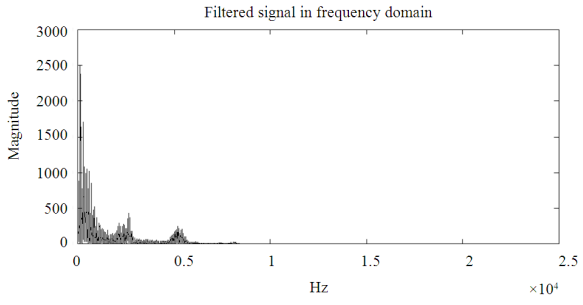

Fig. 7: The frequency response of the VHDL model filtered signal

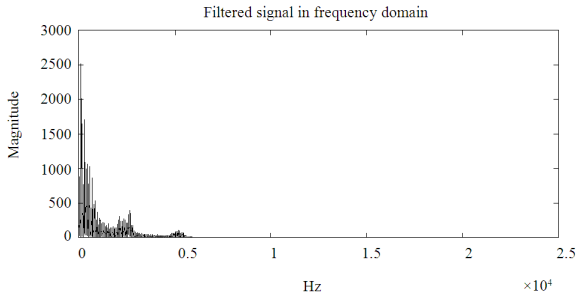

Fig. 8: The frequency response of the filtered signal using MATLAB

Table 3: MSE and SNR Comparison between VHDL model and MTALAB

Filter method	Mean Square Error (MSE)	SNR (dB)
Multipurpose VHDL model filter	0.0291	2.8943
MATLAB	0.0268	3.0162

The filtered signal's Signal to Noise Ratio (SNR) and Mean Square Error (MSE) for the MATLAB model and the proposed multipurpose filter are compared as given in Table 3. Although the same filter specifications and algorithm are used, VHDL model has lower SNR. This happens for truncation and rounding errors in VHDL coding. Before arithmetic operations, all floating points are scaled up as VHDL does not allow floating point data type. The values are scaled up by 1000 when the filter coefficients are entered to the LCCDE. However, all the decimals places are not represented. The decimal places which exceed range are rounded to closest integer. Floating points are truncated during divide operations by division function. As such, it introduces truncation errors. Moreover, approximation errors also occur while estimating sine and cosine. This is the cause of magnitude attenuation of the filtered signal using VHDL model.

CONCLUSION

By simulating and comparing with multipurpose FIR filter MATLAB algorithm, the proposed approach of multipurpose FIR filter design using VHDL is effectively designed, realized and tested. There is significant magnitude attenuation of the filtered signal using VHDL model, which is due to scaled up the floating point. However, with the capability of filtering signal in real time mode utilizing arbitrary filter shape, the multipurpose filter proves to be versatile. For compensating filter flexibility, the accuracy of filtering is sacrificed.

REFERENCES

Akter, M., M.B.I. Reaz, F. Mohd-Yasin and F. Choong, 2008. Hardware implementations of an image compressor for mobile communications. J. Commun. Technol. Elect., 53: 899-910. DOI: 10.1134/S106422690808007X

Choong, F., M.B.I. Reaz and F. Mohd-Yasin, 2005. Power quality disturbance detection using artificial intelligence: A hardware approach. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Apr. 4-8, IEEE Xplore Press, Denver, USA., pp: 146a-146a. DOI: 10.1109/IPDPS.2005.348

Choong, F., M.B.I. Reaz, T.C. Chin and F. Mohd-Yasin, 2006. Design and implementation of a data compression scheme: A partial matching approach. Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation, Jul. 26-28, IEEE Xplore Press, Sydney, Australia, pp: 150-155. DOI: 10.1109/CGIV.2006.94

Coussy, P., D.D. Gajski, M. Meredith and A. Takach, 2009. An introduction to high-level synthesis. IEEE Design Test Comput., 26: 8-17. DOI: 10.1109/MDT.2009.69

Dey, S.R. and A.V. Oppenheim, 2008. Coefficient dither in fixed-point FIR digital filters. Proceedings of 42nd Asilomar Conference on Signals, Systems and Computers, Oct. 26-29, IEEE Xplore Press, Pacific Grove, CA, USA., pp: 556-560. DOI: 10.1109/ACSSC.2008.5074467

ElGizawy, M., A. Noureldin, J. Georgy, U. Iqbal and N. El-Sheimy, 2010. Wellbore surveying while drilling based on kalman filtering. Am. J. Eng. Applied Sci., 3: 240-259. DOI: 10.3844/ajeassp.2010.240.259.

Evans, J.B., 1993. An efficient FIR filter architecture. Proceedings of the IEEE International Symposium Circuits and Systems, May 3-6, IEEE Xplore Press, Chicago, IL, USA., pp: 627-630. DOI: 10.1109/ISCAS.1993.393799

Ibrahimy, M.I., M.B.I. Reaz, M.A.M. Ali and T.H. Khoon *et al*., 2006. Hardware Realization of an Efficient Fetal QRS Complex Detection Algorithm. WSEAS Trans. Circuits Syst., 5: 575-581.

Iskandarani, M.Z., 2010. A novel approach to signal detection of sensor array units using 5-3-1 rule based matched filter algorithm with intelligent identifiers. Am. J. Eng. Applied Sci., 3: 427-432. DOI: 10.3844/ajeassp.2010.427.432

Khoo, K.Y., A. Kwentus and A.N.J. Willson, 1993. An efficient 175 MHz programmable FIR digital filter. Proceedings of the IEEE International Symposium Circuits and Systems, May 3-6, IEEE Xplore Press, Chicago, IL, USA., pp: 72-75. DOI: 10.1109/ISCAS.1993.393660

Kumar, S.T., A. Panda, S.K. Baghmar, S.K. Agrawal and T. Usha, 2010. FIR filter implementation on A FPGA allowing signed and fraction coefficients with coefficients obtained using remez exchange algorithm. Int. J. Adv. Technol., 1: 203-210.

Laskowski, J. and H. Samueli, 1992. A 150-Mhz 43-tap half-band FIR digital filter in 1.2 µ CMOS generated by compiler. Proceedings of the IEEE 1992 on Custom Integrated Circuits Conference, May 3-6, IEEE Xplore Press, pp: 11.4.1-11.4.4. DOI: 10.1109/CICC.1992.591284

Lim, Y.C. and B. Liu, 1988. Design of cascade form FIR filters with discrete valued coefficients. IEEE Trans. Acoustics, Speech Signal Process., 36: 1735-1739. DOI: 10.1109/29.9010

Ma, G.K. and F.J. Taylor, 1990. Multiplier policies for digital signal processing. IEEE ASSP Mag., 7: 6-20. DOI: 10.1109/53.45968

Marufuzzaman, M., M.B.I. Reaz, M.S. Rahman, M.A.M. Ali, 2010. Hardware prototyping of an intelligent current dq PI controller for FOC PMSM drive. Proceedings of the 6th International Conference on Electrical and Computer Engineering, Dec. 18-20, IEEE Xplore Press, Dhaka, Bangladesh, pp: 86-88. DOI: 10.1109/ICELCE.2010.5700559

Mogaki, S., M. Kamada, T. Yonekura, S. Okamoto and Y. Ohtaki *et al*., 2007. Time-stamp service makes real-time gaming cheat-free. Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games, Sep. 19-20, New York, USA., pp: 135-138. DOI: 10.1145/1326257.1326281

Oppenhaim, A.V. and R.W. Schafer, 1975. Digital Signal Processing. 1st Edn., Prentice Hall, Englewood Cliffs, ISBN: 9780132146357, pp: 585.

Pang, W.L., M.B.I. Reaz, M.I. Ibrahimy, L.C. Low and F. Mohd-Yasin *et al*., 2006. Handwritten character recognition using fuzzy wavelet: A VHDL approach. WSEAS Trans. Syst., 5: 1641-1647.

Parks, T.W. and J.H. McClellan, 1972. Chebyshev approximation for nonrecursive digital filters with linear phase. IEEE Trans. Circ. Theory, 19: 189-194. DOI: 10.1109/TCT.1972.1083419

Reaz, M.B.I., F. Choong and F. Mohd-Yasin, 2006. VHDL modeling for classification of power quality disturbance employing wavelet transform, artificial neural network and fuzzy logic. Simulation Trans. Soc. Model. Simulation Int., 82: 867-881. DOI: 10.1177/0037549707077782

Reaz, M.B.I., F. Choong, M.S. Sulaiman and F. Mohd-Yasin, 2007b. Prototyping of wavelet transform, artificial neural network and fuzzy logic for power quality disturbance classifier. J. Electric Power Components Syst., 35: 1-17. DOI: 10.1080/15325000600815431

Reaz, M.B.I., F. Mohd-Yasin, M.S. Sulaiman, K.T. Tho and K.H. Yeow, 2004b. Hardware prototyping of boolean function classification schemes for lossless data compression. Proceedings of the 2nd IEEE International Conference on Computational Cybernetics, Aug. 30-Sep. 01, IEEE Xplore Press, Vienna, Austria, pp: 47-51. DOI: 10.1109/ICCCYB.2004.1437664

Reaz, M.B.I., F. Mohd-Yasin, S.L. Tan, H.Y. Tan and M.I. Ibrahimy, 2005b. Partial encryption of compressed images employing FPGA. Proceedings of the IEEE International Symposium on Circuits and Systems, May 23-26, IEEE Xplore Press, Kobe, Japan, pp: 2385-2388. DOI: 10.1109/ISCAS.2005.1465105

Reaz, M.B.I., M.I. Ibrahimy, F. Mohd-Yasin, C.S. Wei and M. Kamada, 2007a. Single core hardware module to implement encryption in TECB mode. Inform. MIDEM J. Microelect., Elect. Components Mater., 37: 165-171.

Reaz, M.B.I., M.S. Sulaiman, F.M. Yasin, and T.A Leng, 2004a. IRIS recognition using neural network based on VHDL prototyping. Proceedings of the 2004 International Conference on Information and Communication Technologies: From Theory to Applications, Apr. 19-23, IEEE Xplore Press, Damascus, Syria, pp: 463-464. DOI: 10.1109/ICTTA.2004.1307832

Verma, S., K. Ramineni and I.G. Ian, 2009. A control-oriented coverage metric and its evaluation for hardware designs. J. Comput. Sci., 5: 302-310. DOI: 10.3844/jcssp.2009.302.310