Eng.&Tech.Vol.26,No.7,2008

Design and Implementation of Programmable FIR Filter Using FPGA

Dr. Hikmat N. Abdullah
Received on:4/7/2006
Accepted on:5/6/2008

Abstract:

This paper presents the design and implementation of a programmable Finite Impulse Response (FIR) Filter using ALTERA Field Programmable Gate Array (FPGA) device. The filter performance is first tested using Filter Design and Analysis (FDA) tool from Mathworks to verify magnitude response and obtain coefficient tables. The test operation includes LPF and BPF filter types with coefficient lengths of 7 and 31 respectively. The FPGA design is carried out by writing VHDL modules for different filter components. The simulation waveforms, synthesis reports and board programming files have obtained using the package QUARTUSII. ALTERA-FLEX10K10 FPGA Family with EPF10K10LC84-3 board is used as a target device for implementation purpose.

الخلاصة:

هذا البحث يعرض تصميم وبناء المرشح ذو الاستجابة النبضية المحددة باستخدام تقنيـة مصفوفة البوابات المبرمجة الواسعة (FPGA) نوع ALTERA . في البدء تـم فحـص أداء المرشح المطلوب بناءه باستخدام أداة تصميم وتحليل أداء المرشحات المتوفرة من Mathworks للحصول على جداول قيم التفرعات وكذلك استجابة المقدار لمرشح ذو مواصفات معينـة. ولقـد تضمنت عملية الفحص مرشحات من نوع LPF و BPF ذات تفرعات بطـول 7 و 31 علـى التوالي. وتم إنجاز هذا التصميم بكتابة نماذج بلغة VHDL لمختلف مكونات المرشح. ولقد تم استخدام البرنامج QUARTUS II للحصول على الأشكال الموجية وتقـارير البنـاء وملفـات البرمجـة. وتـم اختيـار عائلـة الألـواح FPGA-FLEX10K10 وتحديـداً اللـوح EPF10K10LC84-3 لغرض البناء الحقيقي.

I-Introduction :

Considerable attention has beenplaced on the implementation of signal processing algorithms in VSLI, ranging from full custom VLSI to general purpose digital

https://doi.org/10.30684/ etj.26.7.12
University of Technology-Iraq, Baghdad, Iraq/2412-0758

signal processors. A variety of approaches to custom implementation of FIR filters have been pursued [1,2]. In order to attain high performance, parallel implementation strategies such as systolic methods have been applied. Word-parallel, bit parallel processing techniques appear to scale well with improvements in implementation technology and increasing demands for higher performance.

Advances in field programmable gate array (FPGA) technology have enabled FPGAs to be used in a variety of applications. In routing options that are available for connecting logical operators on the array. Many current FPGA architectures are implemented using memory technologies, and hence the advances in that area will be reflected in improved FPGA density and speed.

Previous works like ones in [3-5] describe different approaches to implement digital FIR filter using FPGA technology. However, all these works presented designs for filters suited for specific applications which means that these designs were carried out for fixed number of taps and fixed values of passband and stopband frequencies. This paper presents design and implementation of programmable digital FIR filter

particular, FPGAs prove particularly useful in data path designs, where the regular structure of the array can be utilized effectively. The programmability of FPGA adds flexibility not available in custom approaches, while retaining relatively high system clock rates. The disadvantage of FPGA are primarily related to the limited number of logic operations that can be implemented on a particular device, the constraints on the inputs and outputs to the atomic logic units, and the limited signal

using ALTERA-FPGA. The design allow the user to easily change the filter type and length (and performance as a result). First, MATLAB mathematical computation package in conjunction with digital signal processing toolboxes is used to design filter response and generate coefficient table. Then VHDL modules are written for each filter element for FPGA implementation. QUARTUSII software package provided by ALTERA is used to write VHDL modules, integrate entities on a top-level schematic file, obtain simulation waveforms and synthesis reports.

II- FIR Design Procedure:

FIRs have the advantage of being much more realizable in hardware [6] because they avoid division and feedback paths. FIR filter response y(n) is computed by convolution operation between filter coefficients h(k) and the input data x(k) which can be described by the following equation:

$$y(n) = \sum_{k=1}^{N} x(k)h(n-k) \quad \ldots(1)$$

where N is the number of filter coefficients. Upon performing convolution, one of the two sets of numbers is reversed and "slid past" the other. The resulting stream of numbers is found by taking the sum of the multiplications at each sliding interval. FIRs can be graphically represented by a Direct Form realization as shown in Figure 1.
The process of designing a filter in an FPGA involves two distinct steps[7]: designing the filter response and designing the filter implementation. Over 100 different filter design tools, including toolbox functions available in MATLAB, are available for designing filter response and generating coefficient tables, each with varying levels of sophistication. Graphical filter design tools provides easy-to-use selections for specifying passband, filter order, and design methods, as well as provide plots of the response of the filter to various standard forms of inputs. One of the most popular tool is the FDA tool from the MathWorks shown in Fig.2 which generates a behavioral MATLAB model and coefficient tables.

III- FPGA Implementation of FIR Filter:

Using QUARTUSII software package provided by ALTERA[8], VHDL modules were written for each of filter elements. Fig.3 shows the the schematic design which integrates all design modules (entities). These modules are *taps*, *hvalues*, *state_m* and *acc*.

The module taps receives the input data d(7..0) whose width is 8 bits and store them permanently on a shift register basis of width equals the number of filter taps. According to the value of selection signal "sel(1..0)" produced by module *state_m* (the size of virtual shift register equals $2^{\text{width of selection signal}}$), the stored data

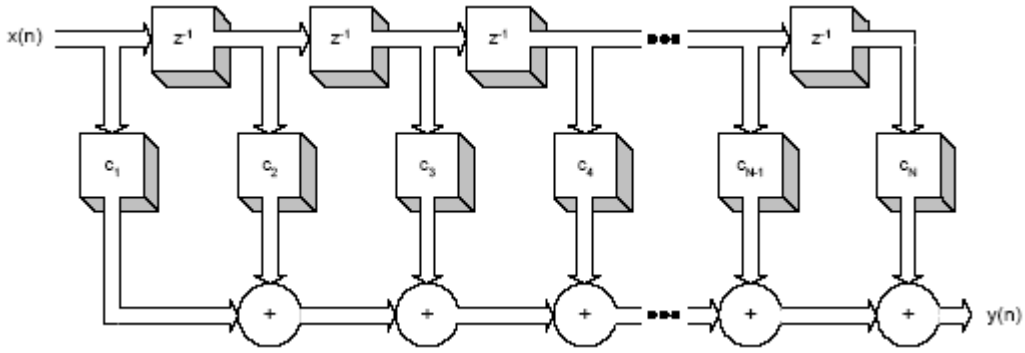

Figure 1: Direct Form realization signal flow graph of an FIR.

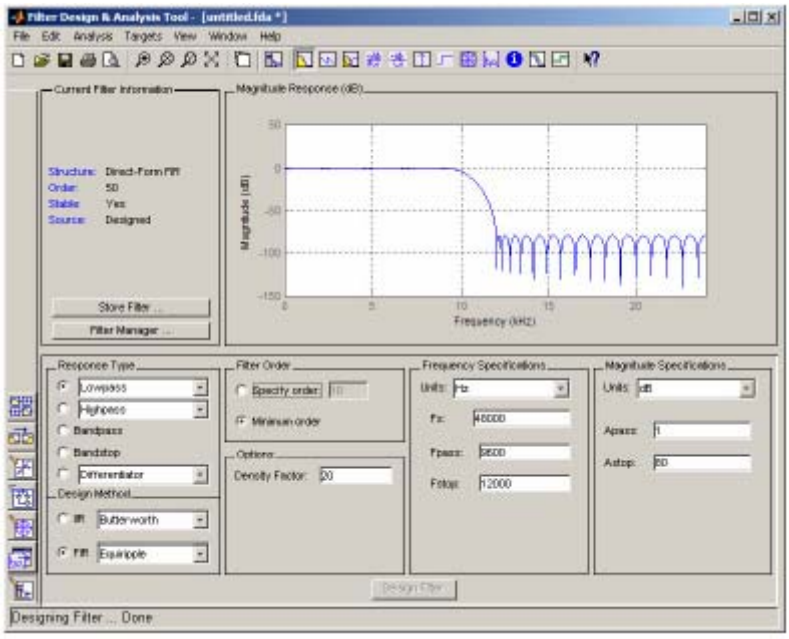

Fig.2 MathWorks Filter Design & Analysis Tool

values d(7..0) are assigned to the output values x(7..0) sequentially one–by-one in each clock pulse to apply them to the multiplier.

When all data values passed to the multiplier, the signal "newt " is activated to receive new data stream.

The *hvalues* module stores the coefficient values of the filter "h(2..0)" and push them to the multiplier sequentially also according to the selection signal produced by *state_m* module. Two other signals are produced by *hvalues* module: "first" to mark that the current multiplication operation is the first one along the the facility of store the produced values and obtain complemented output if necessary.

IV- Top-Level Design:

Using QUARTUSII software package provided by ALTERA, VHDL design modules are written for each of filter entities. At the top level of design, a schematic file is created to assign input and output pins as it look like in the ALTERA-FPGA chip. Fig.8 shows the top level design file for the implemented FIR filter.

V- Verification of Filter Response:

In order to test the designed filter, it must first coefficient tables for each test operation being already available. With FDA tool provided by convolution interval and "follow" to mark that all multiplications have been finished and a new stream would be ready to apply.

The accumulator module *acc* carry out addition process of individual multiplication results. The multiplier is designed directly from Mega Core design facility provided by QUARTUSII [6] package without the need of writing VHDL file manually. In the design, the D flip/flops provides

MathWorks, it is easy to obtain such tables for required performance.

The verification includes 7-tap and 31 tap low pass and bandpass filters. Each of filter frequency response and filter output signal plots have been included for further comparisons. The input signal includes a mix of four different sinusoids having the following frequencies:

- 62.83 rad/sec. (10 Hz)
- 1256.63 rad/sec. (200 Hz)
- 1570.79 rad/sec. (250 Hz)
- 2513.27 rad/sec. (400 Hz)

■ Lowpass Filter Verfication:

The frequency responses significantly differ between the two filter lengths where the 7-tap filter allows considerable energy for unwanted frequencies to pass

through. Whereas the 31-tap filter provides a sharper roll-off at the cutoff frequency and possesses sidelobes lower than the -20 dB mark. Fig.5 illustrates the results of failing to provide a narrower transition band. Energy from the 250 Hz sinusoid were allowed to pass through the filter whereas the 31-tap filter completely eliminates the frequency.

■ Bandpass Filter Verification:

The bandpass filter attempts to only pass frequencies in the 180 Hz - 270 Hz range. A significant difference between the two filter frequency responses can be seen where the 7-tap filter allows much more energy from unwanted frequencies to pass. In addition, the left sidelobe for the lower frequencies does not fall below the -20 dB mark. However, the 31-
tap filter provides much sharper roll-offs for both cutoff frequencies and exhibits a better sidelobe characteristic.

VI- Implementation Results:

■ Simulation :

Once the filter performance is verified and filter coefficient tables are obtained from FDA tool, the next step is to obtain simulation waveforms to verify the correctness of FPGA implementation. Fig.9 shows a sample simulation waveforms for a 7-tap LPF whose magnitude response shown in Fig.4. In Fig.9, the clock period is chosen to be 20 nsec and the data d(7..0) are assumed an unsigned decimal count value start from 0 to 15 and repeats with an increment of one in each clock pulse. The "newt" signal is activated as shown from waveform after every eight successive clock pulses which means a stream of 8 bit data has been stored in the shift register. Since the result in a value of 11 bit length, the "fellow" signal is activated after 11 clock pulses and "yvalid" signal becomes 1 in the next clock cycle to indicate that a new result is available.

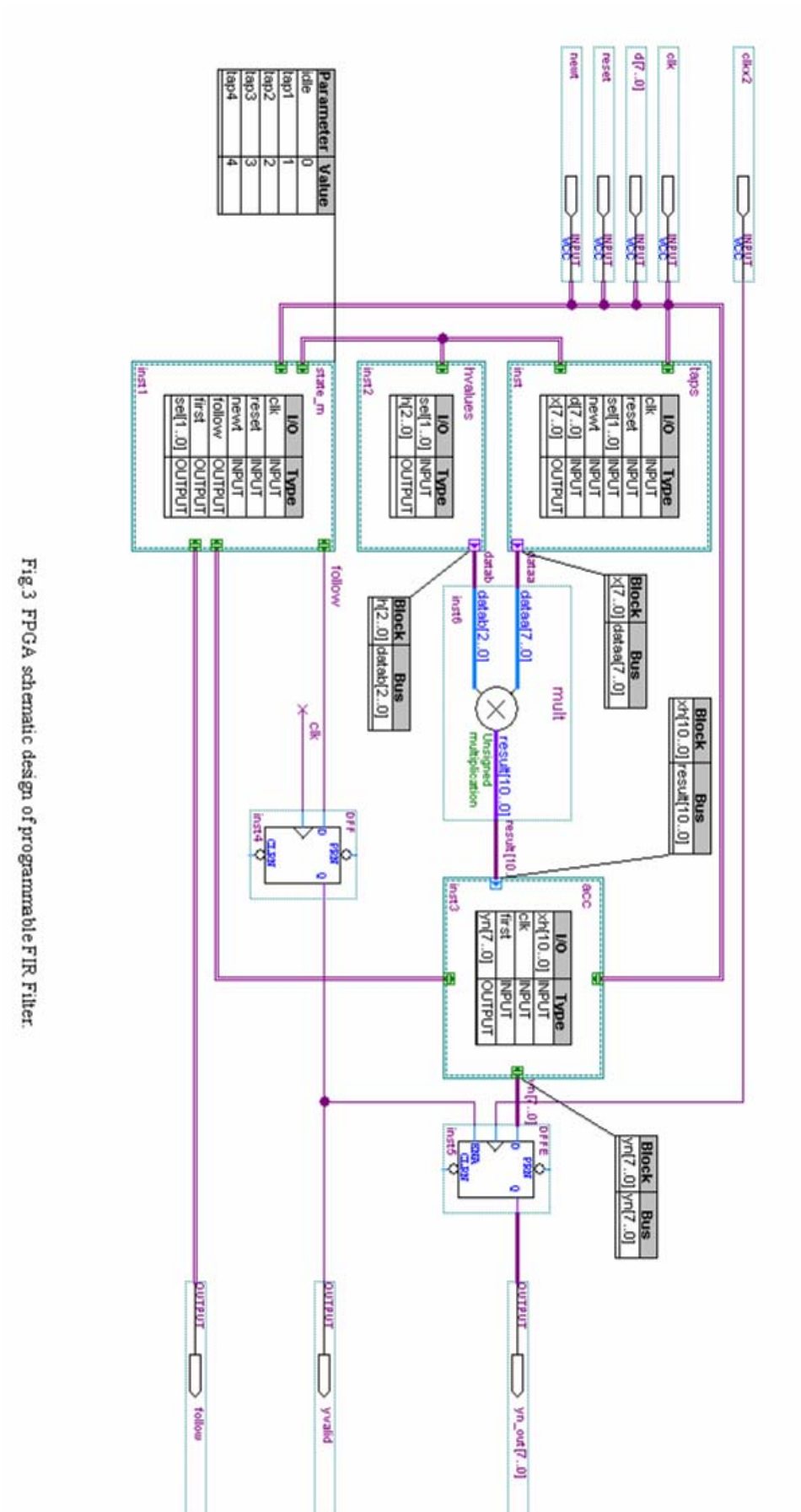

Fig.3 FPGA schematic design of programmable FIR Filter.

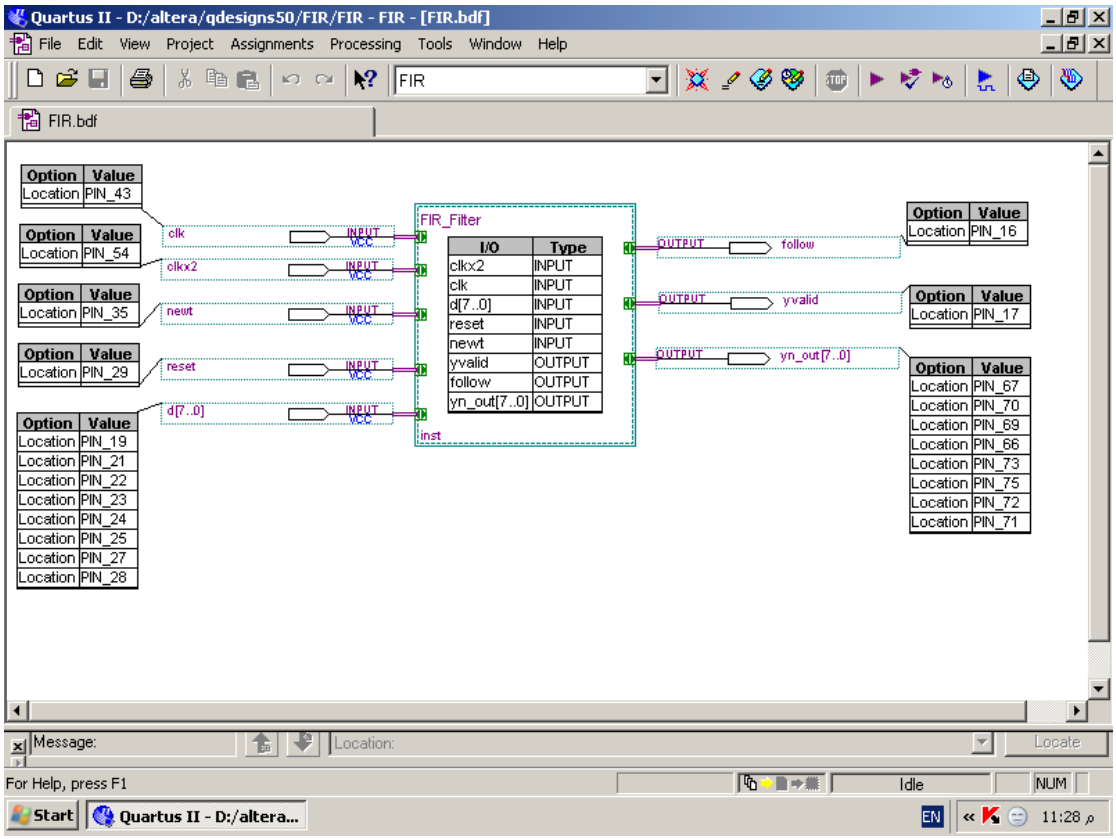

Fig.4 Top-level design of the FPGA implemented FIR Filter.

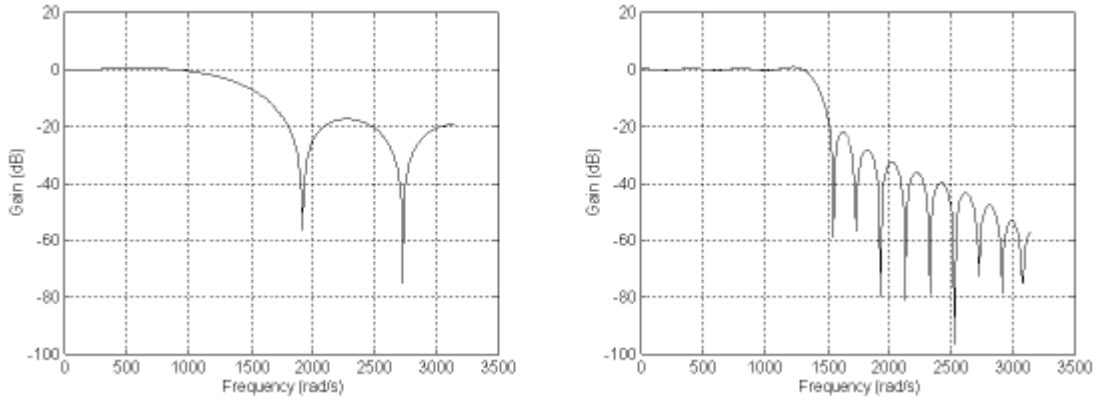

Fig.5 Lowpass filter frequency responses for 7-tap and 31-tap filters with a target cutoff frequency of 1413.7 rad/sec. (225 Hz).

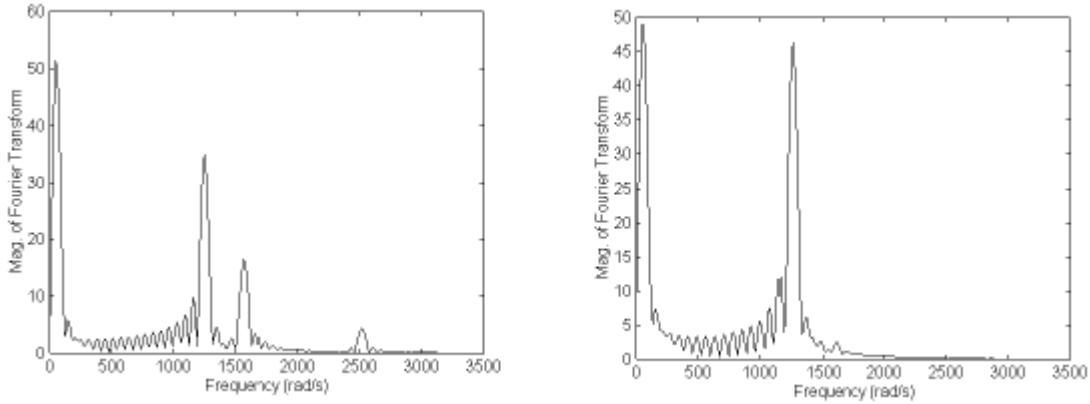

Fig.6 Lowpass filtered signal spectrums for 7-tap and 31-tap filters with a target cutoff frequency of 1413.7 rad/sec. (225 Hz).

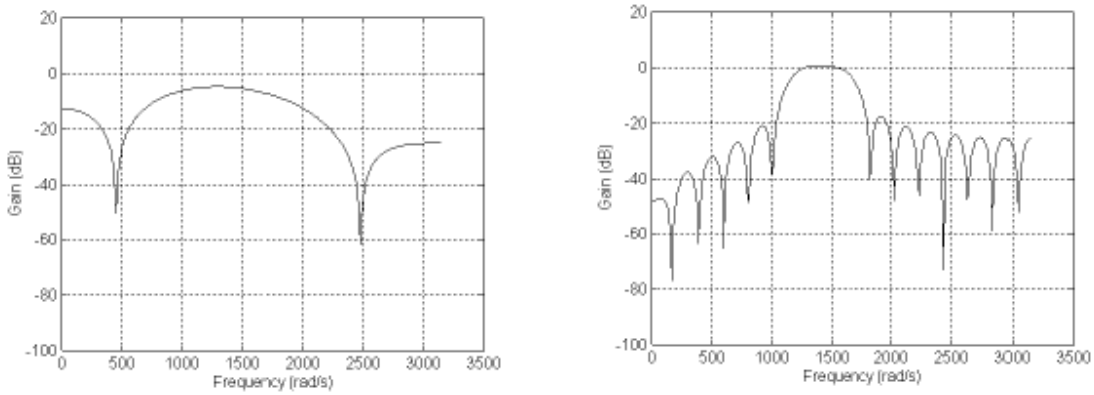

Fig.7 Bndpass filter frequency responses for 7-tap and 31-tap filters with a target cutoff frequency of 1130.97 rad/sec. (180 Hz) and 1696.46 rad/sec. (270 Hz).

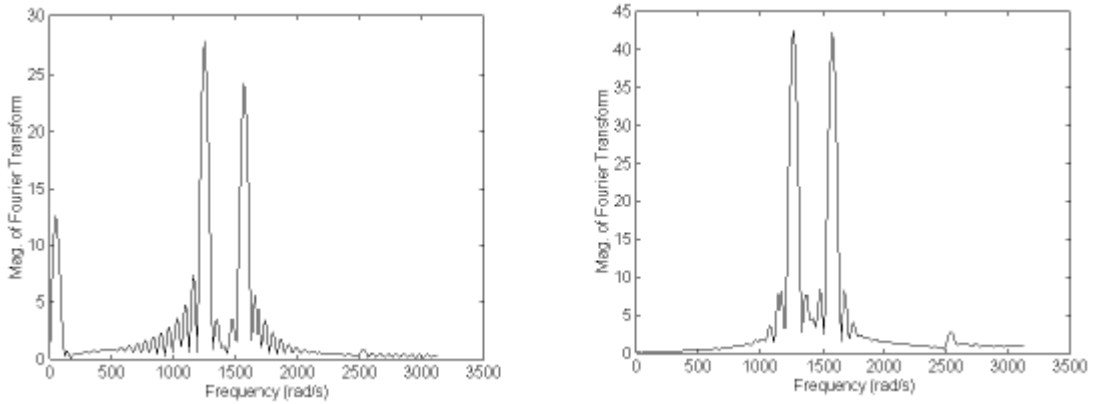

Fig.8 Bandpass filtered signal spectrums for 7-tap and 31-tap Filters.

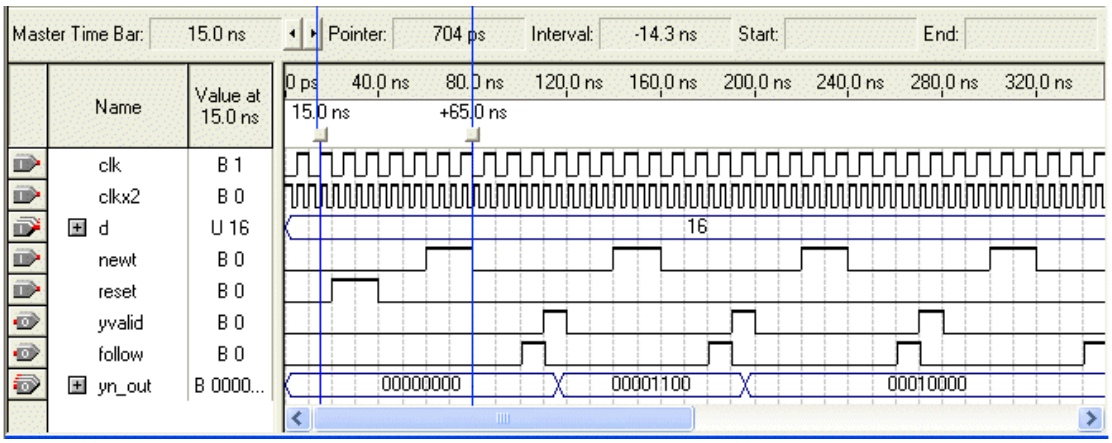

Fig.9 Simulation Waveform of FPGA Implemented FIR Filter (7-tap LPF)

■ Synthesis Reports :

There are a large number of synthesis reports (hardware and software reports) obtained from synthesis operation. They describe all what concern the implementation process like storage resources required, I/O resources required, computation resources required, time delay at different points inside the chip…etc. [9]. Tables 1 through 4 show summary of hardware synthesis reports for 7-tap LPF, 31-tap LPF, 7-tap BPF and 31-tap BPF whose performances were discussed in section V respectively.

It is seen from these reports summary that the FPGA resources requirements increases as the number of filter taps increases and as a result, the maximum possible operating speed decreases. However there is no significant difference in storage resources or speed when the filter type is changed for the same filter coefficient lengths.

■ Implementation :

After obtaining a correct simulation results for the written VHDL modules, the programmable FIR filter has been implemented by generating a program files via QUARTUS II package corresponding to each implementation case. These program files are then downloaded to the FPGA ALTERA-FLEX10K10 board.

The data are applied to this development board internally by writing a VHDL module that produces such an input data. The coded messages are recognized by the aid of the seven segment display provided in the board. This operation should be done at a low data rates in order to recognize the values of coded messages symbols whose displayed in hexadecimal form.

VII- Conclusions:

Digital FIR filter are one of the important element in any DSP design. The usage of FPGA Technology to implement these codes provides many advantages like efficient reconfigurability and universal chip implementation. The paper has described a simple and efficient approach to implement FPGA based programmable filter that suited to different applications unlike most previous approaches on a penalty of reducing computation speed. The design has carried out in such a way that allows the user to easily change filter type and order as for required filter performance. LPF and BPF of different lengths has implemented for

Table 1 Summary of hardware synthesis reports for 7-tap LPF

Maximum period 3.67 ns (Maximum frequency: 272.4 MSPS).
Maximum path delay from the any node: 6.05ns.

Device utilization for EPF10K10LC84-3

Resource	***Used***	***Available***	***Utilization***
IOs	*22*	*160*	*13.7%*
FG Function Generators	*198*	*2592*	*7.6%*
H Function Generators	*92*	*1296*	*7.2%*
CLB Flip-Flops	*209*	*2592*	*8.1%*

Table 2 Summary of hardware synthesis reports for 31-tap LPF

Maximum period 11.22 ns (Maximum frequency: 89.1 MSPS).
Maximum path delay from the any node: 17.24ns.

Device utilization for EPF10K10LC84-3

Resource	***Used***	***Available***	***Utilization***
IOs	*25*	*160*	*15.6%*
FG Function Generators	*770*	*2592*	*29.7%*
H Function Generators	*283*	*1296*	*21.8%*
CLB Flip-Flops	*937*	*2592*	*36.1%*

Table 3 Summary of hardware synthesis reports for 7-tap BPF

Maximum period 3.79 ns (Maximum frequency: 263.8 MSPS).
Maximum path delay from the any node: 3.8 ns.

Device utilization for EPF10K10LC84-3

Resource	***Used***	***Available***	***Utilization***
IOs	*22*	*160*	*13.75%*
FG Function Generators	*205*	*2592*	*7.9%*
H Function Generators	*91*	*1296*	*7%*
CLB Flip-Flops	*224*	*2592*	*8.6%*

Table 4 Summary of hardware synthesis reports for 31-tap BPF

Maximum period 11.22 ns (Maximum frequency: 89.1 MSPS).
Maximum path delay from the any node: 11.22ns.

Device utilization for EPF10K10LC84-3

Resource	***Used***	***Available***	***Utilization***
IOs	*25*	*160*	*15.6%*
FG Function Generators	*767*	*2592*	*29.6%*

H Function Generators	*283*	*1296*	*21.8%*
CLB Flip-Flops	*947*	*2592*	*36.5%*

performance comparison purpose. The complete FPGA design and implementation procedure has been presented starting from top-level schematic design to timing analysis, synthesis reports and downloading programming file.

VIII. References:

1. Borth D. E. et. El., "A flexible Adaptive FIR Filter VLSI IC.", IEEE Journ. Select. Areas. Commun., SAC-6(3): 494-503, April 1998.
2. Powell S. and. Chau, P"Reduced complexity Programmable FIR Filters", IEEE Int. Symp. Circuits and Syst., pages 561-564, May 1999.
3. Gilsin G. and B. Newgard, "16-Tap, 8 bit FIR Filter Applications guide", http://www.xilinx.com/legal.htm, November 1994.
4. Atinirmit, P. "Design and Implementation of an FPGA-Based Adaptive Filter Single-user Receiver", M.Sc. Thesis, Virginia Polytechnic Institute and State University, 1999.
5. Champman, K. P. A. Miller and M. George, "CDMA Matched Filter Implementation in Vertix Devices", http://www.xilinx.com/legal.htm, March 2000.
6. John N. Little and Loern Shure, "Signal Processing Toolbox", The Mathworks, Inc. 2004.
7. "Filter Design Methods For FPGAs", Accel Chip, 2006.
8. "Synthesis and Simulation Design Guide", Altera, Inc. 2004.
9. "QUARTUSII Reference Manual", Altera, Inc., 2005.