International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-2, Issue-6, January 2014

Efficient Programmable Finite Impulse Response Filter Using Xilinx MAC FIR Filter for Software Defined Radio

Baha Ali Nasir

Abstract*: This paper presents a development of FIR filter using polyphase multiply accumulate (MAC) block in system generator to decrease the multiplication process and hence power consumption to provide the SDR requirements. The entrenched system generator offers a very attractive solution that balance high flexibility and performance of FIR filter to minimize the multiplication process. This paper focuses on efficient design of digital FIR filter for software defined radio on an FPGA target device. The simulation results shows an important development in minimization of FIR filter multiplication to utilize the look up table (LUPs) and Slices in FPGA area which decrease the power consumption compared with conventional design.

***Index Terms*: PFIR, MAC, SDR.**

I. INTRODUCTION

The Xilinx Company offer a very good structure to design FIR filter called multiply accumulate (MAC) FIR to serve the Software Defined Radio (SDR) applications. Now days, the Digital Signal Processing (DSP) need to be efficient, low cost and easy in order to provide the user demand. The Xilinx MAC FIR core implements a highly configurable, high-performance, and area efficient FIR filter. Single-rate polyphase decimators and interpolators are supported. Multiple data channel operation is supported for all filter types. Symmetry in the coefficient set is exploited for single MAC implementations to increase overall performance and minimize resource utilization. All internal data-paths provide full-precision arithmetic to avoid the possibility of overflow. The full-precision sum-of-products is presented on the Data Output port. A set of three handshake control signals provides an easy-to-use user interface. The conventional tapped delay line representation of an FIR filter is shown in Figure 1[1].

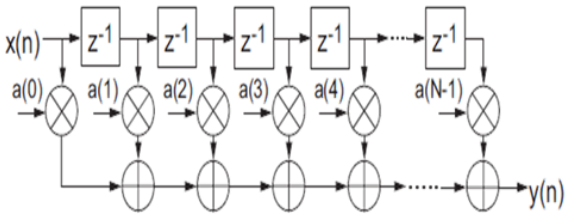

Figure 1: Conventional Tapped-delay Line FIR Filter [1]

Revised Manuscript Received on 30 January 2014.
* Correspondence Author
Baha Ali Nasir*, Electronic Technique Dept. Institute of Technology, Baghdad, Iraq.

The MAC FIR core uses one or more time-shared multiply accumulate (MAC) functional units to service them N sum-of-product calculations in the filter. The core automatically determines the minimum number of MAC engines required to meet the user specified throughput. Figure 2 is a block diagram of a single MAC engine. The figure shows storage for the filter coefficients and the control circuit that sequences the appropriate coefficients and data to the multiply-accumulator for the specified integration period. The descriptions of the MAC engines ports is illustrated in Table 1.

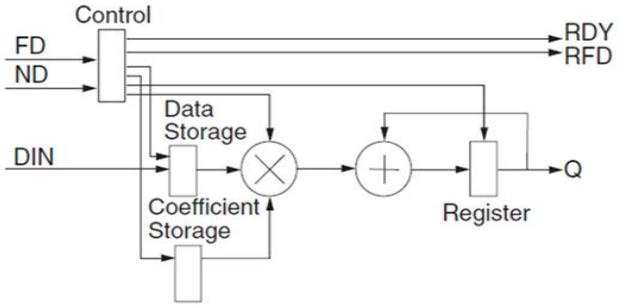

Figure 2: Single MAC Engine Block Diagram [1]
Table 1: Core input output port descriptions [1]

Signal	Description
DIN	**Data Input**: Input data port
ND	**New Data** : When this signal is asserted, the data sample presented on the *DIN* port is loaded into the MAC FIR.
RFD	**Ready for Data** : Indicates when the MAC FIR can accept new data.
RDY	**Ready**: Indicates that a new filter output sample is available.
Registers	The final summation is presented on the output port and can be optionally registered so the output value remains static during successive computation periods.
Control	The control logic manages the flow of data in and out of the input data buffer, the flow of data in and out of the data storage memory, reading of coefficients, and providing the control signals to the MAC core.
Coefficient Storage	The core customization GUI provides the flexibility to allocate the coefficient storage ROM as either distributed Select RAM or block Select RAM
Data Storage	Block memory is the default storage type. Selection of distributed Select RAM memory may be advantageous for several cases
Q	Filter output Port

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

II. PROPOSED PFIR FILTER

The Programmable Finite Impulse Response (PFIR) filter design is required to optimize the software defined radio (SDR) requirements in terms of mask frequency and amount of error in the pass-band region. The error could be optimized by found the minimum filter coefficients that reduce the maximum absolute weighted error. At this time the filter coefficients play important role in the limitation of the error, which is a tradeoff between the filter coefficients and the error should be achieved to optimize the filter design. Digital FIR filter design depends on three important parameters, the filter order, attenuations, and transition width.

The first step in the proposed program is to guess the attenuation and transition width of PFIR depending on SDR requirements. For this, the maximum transition width of PFIR filter is 0.08 π rad/sample to minimize the passband ripple. The pass band and stop band attenuation have been guessed to be 0.01 dB or less to provide minimum ripple. Under these conditions, the program will calculate the PFIR filter orders. If the adjacent rejection and blocker band of the decimation filter is less than or equal to the conventional design, the program should insert extra weights to the PFIR filter response at different stop band frequency with appropriate locations until the maximum attenuation is achieved (i.e. more than -25 dB and 105 dB in the adjacent and blocker band respectively). If the pass band ripple more than or equal to the conventional design, then go to increase the transition width of PFIR filter up to 0.08 π rad/sample until the pass band ripple become less than 0.03dB. If the pass band ripple is less than 0.03db, then save this filter and this is the best approximation. The required conditions is achieved after 6 iteration of this program because the transition width of PFIR filter is changed from (0.03 to 0.08) π rad / sample. Therefore, the simulation results show that the PFIR filter order of 64 should be used to minimize the overall pass band ripple and the accurate location of weights at 100 kHz and 108 kHz should be used to provide maximum attenuation.

Therefore, the stop-band deviation has been guessed to the value of $\delta_s = 0.01$ and to assume that the desired pass-band attenuation is AP≤ - 0.012dB.The normalized transition width of PFIR filter is $\Delta f = 0.08$ rad/sample. The pass-band deviation δ_P is determined by using [2-3].

$$\delta_p = \frac{10^{-\frac{0.012}{20}} - 1}{10^{-\frac{0.012}{20}} + 1} \cong \mathbf{0.0007}$$

Therefore the filter length can be calculated by using [2-3]:

$$M_{PFIR} = \frac{-20log_{10}\left(\sqrt{\delta_p \delta_s}\right) - 13}{14.6\,\Delta f} + 1 = \frac{-20log_{10}\sqrt{0.01 \times 0.\square 007} - 13}{14.6\left(0.\frac{08\pi}{2\pi}\right)} + 1 = 65$$

The filter order is given as:

$N_{PFIR} = M-1 = 65-1 = 64$

This filter is a 65-taps Equiripple linear-phase FIR. The PFIR has to perform another task, i.e., to move the adjacent band rejection of the filter away. An arrangement could be achieved in the filter structure by means of different stop-band attenuation with different weights. The M-file program of design the PFIR filter in MATLAB is illustrated below.

```
N = 64;
Fs = 541666;
F = [0 80e3 100e3 100e3 108e3 Fs/2]/(Fs/2);
A = [1 1 0 0 0  0];
W = [10 1 10];  % Weight the passband 10 times more
than the stopband
pfir = firgr(N,F,A,W);
pfirq = fi(pfir,true,16);
Hpfir = mfilt.firdecim(2,double(pfirq));
 fvtool (Hpfir)
```

Because of the different weights at 100 and 108 kHz, the resulting PFIR filter response given by using fvtool in MATLAB shows different stop-band attenuation values at 100 kHz and 108 KHz. Additionally, the weights at the pass band have been determined to be 10 times more than the weight of the stop-band. The weights were determined to improve the adjacent band rejections. Results show adjacent band rejection improvement of 18% more than conventional filter, as shown in Figure 3. The idea of inserting different weights at 100 kHz and 108 kHz in the filter design is highlight at filter response.

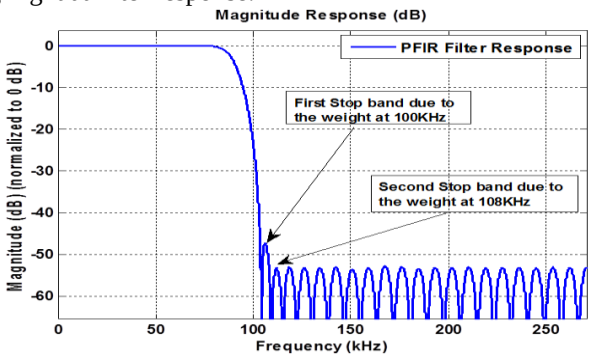

Figure 3: Programmable finite impulse response with different stop-band

III. PFIR FILTER DESIGN VERIFICATIONS

The PFIR filter coefficients have been designed in fixed point values using the FDATool provided by MAC FIR decimator in System Generator as shown in Figure 4 and Figure 5. In Figure 6, the designed filter is connected in parallel with the floating point MATLAB design to verify the PFIR filter coefficients.

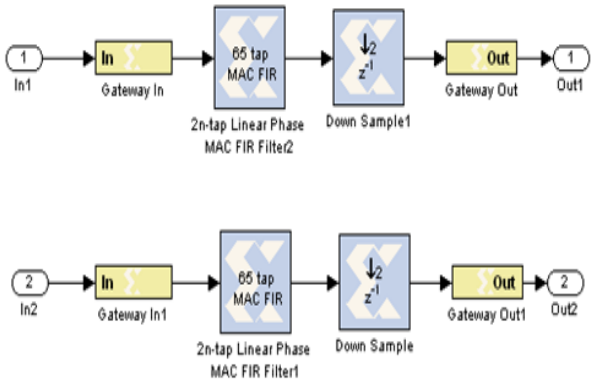

Figure 4: MAC FIR Filter design

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

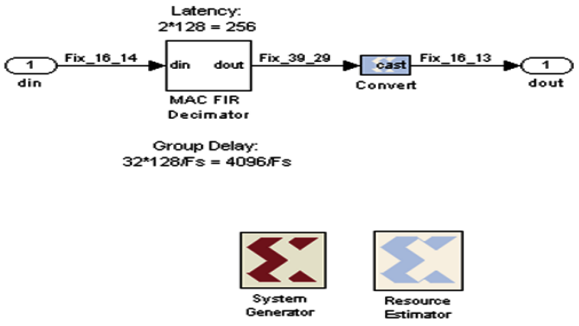

Figure 5: MAC PFIR Decimator

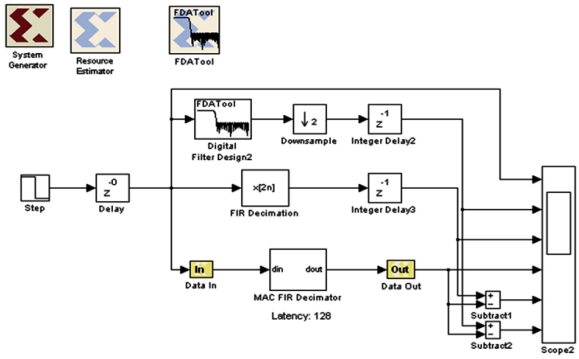

Figure 6: PFIR simulation verification

The resulting response of the conventional decimator and MAC FIR filter simulation in fixed point and floating point design of FIR decimator is shown in Figure7. The resulting response of the three filters shows a slight difference of approximately 2 × 10-5. Quantization errors resulting from the limited number of bits representing the numerical values used in the arithmetic functions such as multiplication and summation in the filtering process have caused this difference.

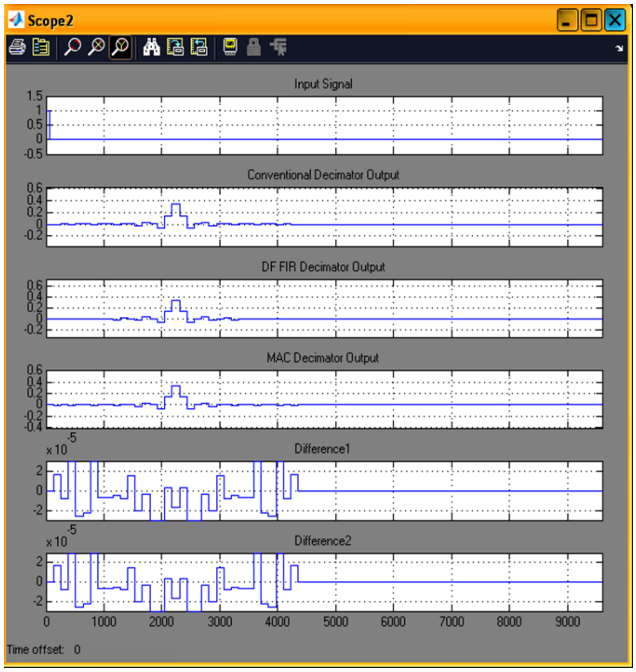

Figure 7: PFIR simulation Filter Response

After the PFIR filter design at fixed point has been verified with the MATLAB design in System Generator, the next step, which is the implementation of the structure in real-time form, is initiated. The complete PFIR design is synthesized to create the Verilog netlist using the output options under System Generator token graphical user interface (GUI) interface and the results of the performance of map, place and route task done, by using the project navigator tool to implement the design in real time. The ISE project file is produced in the netlist directory specified in the System Generator token graphical user interface (GUI) along with the design and its association with timing constraints files [4] and [5].

IV. PFIR IMPLEMENTATION

The implementation steps shown in Figure 8 are read in the constraints file that consists of three major steps: translate, map, and place & route. The translate step essentially flattens the output of the synthesis tool into a large single netlist. A netlist in general, is a large list of gates which is compressed at this stage to remove any hierarchy. The map step groups the logical symbols in the flattened netlist into physical components, specific to the target device. The place and route step places each of these physical components onto the FPGA chip and connects them through the switch matrix and dedicated routing lines [4].

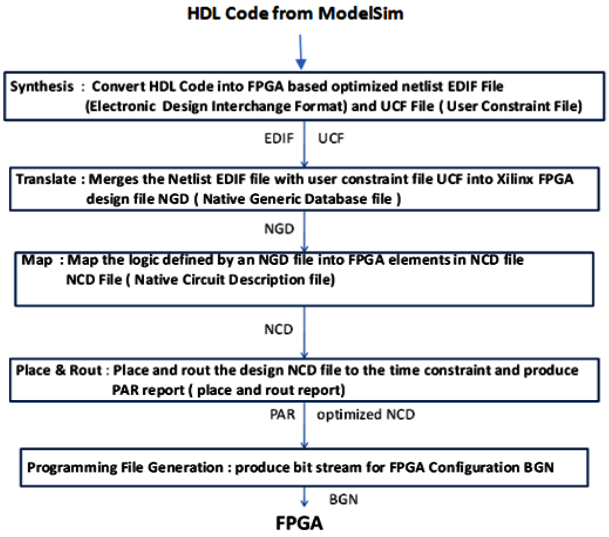

Figure 8: FPGA Implementation steps

Timing characteristics comprise another important factor that affects the performance of FPGA implementation. Thus, the estimated period (required path delay) for the FPGA element must not exceed the requested clock period. For this reason, timing slack = requested period − estimated period, should have a positive value; otherwise, the integrated design has to be reworked [5],[6],[7] and [8]. The synthesis output files generated by the integrated synthesis environments (ISE) software in electronic design interface file (EDIF) and user constraints file (UCF) forms represent the optimized netlist of integrated design, timing constraints, and FPGA pin assignment that have been implemented into the FPGA development board by following the steps listed below [9]. [10]. [11]:

- Translate: Convert the netlist file of integrated design in EDIF format to native generic database (NGD) file, which contains logic description of hierarchical components and Xilinx primitives for the integrated design using NGD build program.
- Map: Perform logical DRC on the NGD file and map the design logic to slices and input-output (I/O) cells in FPGA to create native circuit description (NCD) file.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Place and Route: The design in mapped NCD file is place and route into FPGA based on timing constraints using timing analysis tools with no errors found. All signals are completely routed for transmitter and receiver, respectively.
- Bit generation and program download: Bit generation is used to generate configuration bit-stream file in BIT format form, and subsequently downloaded into FPGA via JTAG cable using the iMPACT program.
- The timing requirement [12] is satisfied as shown in the post-PAR static timing report as illustrated in Tables 2.

Table 2: PFIR Place and Route Timing Report

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_clk_b81e7c3a = PERIOD TIMEGRP "clk_b81e7c3a" 10 ns HIGH 50%	SETUP	3.981ns	6.019ns	0	0
	HOLD	0.502ns		0	0
TS_ce_2_b81e7c3a_group_to_ce_2_b81e7c3a_group = MAXDELAY FROM TIMEGRP "ce_2_b81e7c3a_group" TO TIMEGRP "ce_2_b81e7c3a_group" 20 ns	SETUP	12.101ns	7.899ns	0	0
TS_ce_2_b81e7c3a_group_to_ce_256_b81e7c3a_group = MAXDELAY FROM TIMEGRP "ce_2_b81e7c3a_group" TO TIMEGRP "ce_256_b81e7c3a_group" 20 ns	SETUP	17.645ns	2.355ns	0	0
TS_ce_128_b81e7c3a_group_to_ce_2_b81e7c3a_group = MAXDELAY FROM TIMEGRP "ce_128_b81e7c3a_group" TO TIMEGRP "ce_2_b81e7c3a_group" 20 ns	SETUP	18.051ns	1.949ns	0	0
TS_ce_4_b81e7c3a_group_to_ce_4_b81e7c3a_group = MAXDELAY FROM TIMEGRP "ce_4_b81e7c3a_group" TO TIMEGRP "ce_4_b81e7c3a_group" 40 ns	SETUP	34.058ns	5.942ns	0	0
TS_ce_4_b81e7c3a_group_to_ce_128_b81e7c3a_group = MAXDELAY FROM TIMEGRP "ce_4_b81e7c3a_group" TO TIMEGRP "ce_128_b81e7c3a_group" 40 ns	SETUP	38.279ns	1.721ns	0	0

V. RESULTS AND DISCUSSION

The optimization of timing performance for real-time process reports show zero timing error, and the internal sampling period implementation is satisfied, as shown in Table 3.

Table 3: The timing constraints and internal sampling period of each implementation steps

```
All constraints were met.

Data Sheet report:

All values displayed in nanoseconds (ns)

Clock to Setup on destination clock clk

---------------+---------+---------+---------+---------+
               | Src:Rise| Src:Fall| Src:Rise| Src:Fall|
Source Clock   |Dest:Rise|Dest:Rise|Dest:Fall|Dest:Fall|
---------------+---------+---------+---------+---------+
clk            |   6.259|         |         |         |
---------------+---------+---------+---------+---------+

Timing summary:

Timing errors: 0 Score: 0Constraints cover 32655 paths, 0 nets, and
3983 connections

 Design statistics:

Minimum period:   6.259ns   (Maximum frequency: 159.770MHz)

Maximum path delay from/to any node:   6.259ns

Analysis completed Tue Feb 22 11:26:08 2011

Trace Settings:

Trace Settings

Peak Memory Usage: 225 MB
```

The worst case slack is the difference between constraint clock and best case achievable and should be positive (i.e. the best case achievable must not exceed the constraint clock), for example: if the constraint clock period = 20 ns and the best case achievable = 7.899 ns, then the worst case slack = 20 −7.899= 12.101 ns. Due to the positive value of worst case slacks the ISE software produce zero timing error and zero timing score for each PFIR. Because of the absence of error and warning for design rule checker (DRC), the configuration of the PFIR filter has been successfully implemented in FPGA using the ISE software through the translate, map, place, and route steps. The project status of ISE software generates the PFIR utilizing summary as shown in Table 4. The table lists total number of slices and look-up tables (LUTs) used in this design. In each slice they are two LUTs and two FFs, during the PAR, the ISE software put all necessary LUTs close to each other for minimum propagation of data, for that some LUT inside slice not used and in some slice only FFs is used without LUT so, the number of LUTs cannot be calculated manually, therefore the LUTs could be less or more than Slices depend on software optimization. The device utilization summary generated by ISE software represents the available logic elements in FPGA and logic elements used by the in hand project been designed. The percentage of used logic elements to the available logic elements is calculated as follow:

$$\text{Utilization \%} = \frac{\textbf{used logic elements}}{\textbf{availabl} \square \textbf{ logic elements}} \; X \; \mathbf{100}$$

for examples

Utilized number of Slice Flip Flop = (946/30720) x 100 = 3%
Utilized number 4-input LUTs = (530/30720) x 100 = 1%
Utilized number of occupied Slices = (661/15360) x100 = 4%
Utilized number of bonded IOBs = (31/448) x 100 = 6%

Table 4: Decimation filter project status and device utilization summary

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	946	30,720	3%	
Number of 4 input LUTs	530	30,720	1%	
Logic Distribution				
Number of occupied Slices	661	15,360	4%	
Number of Slices containing only related logic	661	661	100%	
Number of Slices containing unrelated logic	0	661	0%	
Total Number of 4 input LUTs	755	30,720	2%	
Number used as logic	530			
Number used as a route-thru	7			
Number used as Shift registers	218			
Number of bonded IOBs	31	448	6%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			
Number of FIFO16/RAMB16s	2	192	1%	
Number used as FIFO16s	0			
Number used as RAMB16s	2			
Number of DSP48s	2	192	1%	
Total equivalent gate count for design	158,984			
Additional JTAG gate count for IOBs	1,488			

The minimum number of slices and LUTs obtained by the conventional designs are 967 and 819, respectively presented by [13]. This means that the proposed decimatio filter is more efficient than the conventional design in terms of area and power consumption. A comparison between the proposed design and conventional design is shown in Table 5.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

In 2010, two researches have been presented in the same field [13-14]. Each researcher provides different resource of Slices and LUTs. The mythology used by [13] consumes fewer slices compared with [14]. Slices number in FPGA is more important in power consumption than LUTs because each slice contain two LUPs and two FFs. In each slice they are two LUTs and two FFs, during the PAR, the ISE software put all necessary LUTs close to each other for minimum propagation of data, for that some LUT inside slice not used and in some slice only FFs is used without LUT so the number of LUTs cannot be calculated manually, therefore the LUTs could be less than Slices and vice versa

Table 5: FPGA Slices and LUTs comparison

Resource	[11]	[12]	Proposed	Improvements%
Slices	2599	819	661	19%
LUTs	2966	755	755	21%

VI. CONCLUSION

A novel linear-phase Equiripple FIR filter has been developed to improve the performance of SDR receivers to avoid the interference in the received signal. A new technique to decrease the multiplication process by using Xilinx MAC FIR is introduced. The development enhances the SDR transceiver in terms of frequency dynamic range and power consumption. The results show an important utilization in LUTs and Slices compared with conventional design. This paper has provided a tutorial review of PFIR approach to implement programmable narrow-band filters. The powerful and useful results that are realized when MAC FIR is combined with FPGA device technology. In the case of LPF design, the PFIR architecture provides 19% and 21% increase in hardware efficiency (Slices and LUTs respectively) over traditional FPGA implementations. Approach combining of PFIR mechanization and polyphase filter decomposition resulted in important decreasing of FPGA area.

REFERENCES

1. Xilinx Inc. " MAC FIR v 5.1 Product specifications",2007, Available at: www.xilinx.com pp.1-19
2. Chen, "Design of Equiripple Linear-Phase FIR Filters Using MATLAB", IEEE transaction, 2011 available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5997704
3. Sven, "Applied Signal Processing ETB006 FIR Filter Design" Chapter-1 pp.1-5, 2004 http://www.bth.se/tek/asb.nsf/attachments/Assignment_grade4_pdf/$file/Assignment_grade4.pdf
4. Xilinx, Inc., Xilinx ISE 9.2i Software Manuals: Constraints Guide, and Development System Reference Guide". San Jose, California, USA, pp. 1-844, 2007
5. Avnet Memec, Inc.,"Virtex-4 MB Development Board User's Guide",Ver. 3.0, Phoenix, Arizona, USA, pp1-42,2005. Citing Internet sources URL: http://www.files.em.avnet.com/files/177/v4mb_user_guide_3_0.pdf
6. Xilinx, Inc.,"System Generator for DSP User Guide", Release 9.2.01, San Jose, California, USA, pp. 1-346, 2007
7. Xilinx, Inc.,"DSP Design Flows in FPGA Tutorial Slides". San Jose, California, USA, pp. 1-82, 2003.
8. Xilinx Inc., "System Generator for DSP": User Guide, Release 10.1,Snn Jose, Clifornia, USA, PP.1-40: www.xilinx.com, 2008
9. Xilinx, Inc., "Xilinx ISE 9.2i Software Manuals: Constraints Guide, and Development System Reference Guide". San Jose, California, USA, pp. 1-844. 2007.
10. Xilinx, Inc., "Virtex-II Pro and Virtex-II Pro X FPGA User Guide", Ver. 4.2, San Jose, California, USA, pp. 1-559. 2003. Citing Internet sources URL:http://www.xilinx.com/support/documentation/user_guides/ug012.pdf
11. Xilinx, Inc., "Spartan-3 FPGA Family Data Sheet", Ver. 2.5, San Jose, California, USA, pp. 1-217.2006. Citing Internet sources URL: http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf
12. Xilinx, Inc., "Timing constraints User Guide", Ver. 11.1.1, san Jose California, USA, pp. 1-137, 2009. Citing Internet sources URL:http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/ug612.pdf
13. Rajesh and Swapna, "Efficient Hardware Co-Simulation of down Convertor for Wireless communication Systems", International journal of VLSI design & Communication Systems, Vol.1, No.2, pp. 13-21, 2010
14. Changrui, Kong, Xie Shige and Huizhi1, "Design and FPGA Implementation of Flexible and Efficiency Digital Down Converter", 978-1-4244-5900- IEEE, PP.438-441, 2010

AUTHOR PROFILE

Assist Lecturer Eng Baha Ali Nasir Received his B.Sc. Academy of Engineering - Sarajevo - Bosnia's Republic in 1983, and received the M.Sc. degree in Electrical Engineering - University of Belgrade in 1985. Currently he is Assist Lecturer, Researcher and training supervisor, Dep. of Electronic in Institute of Technology Baghdad.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)