www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 3 March-2014 Page No. 5114-5117

FPGA Implementation of Fast Fourier Transform (FFT) Based Finite Impulse Response (FIR) filter Using VHDL

Ms.Ritu Sharma, Prof. Ravi Mohan Verma
Department of Electronics & Communications Engineering
RGTU Bhopal, SRIT, Jabalpur

Abstract— The paper describes the development of FIR filters on Field programmable gate array (FPGAs) using FFT Algorithm. FIR filter has been designed and realized by FPGA for filtering the digital signal. The implementation of FIR filter on a Cyclone IV GX FPGA is considered. Presented soft core is the unit to perform the finite impulse response filter based on the Fast Fourier Transform (FFT). It performs the convolution of the unlimited signal sequence with the synthesized impulse response of the length of Ni=$N/2$ samples, where N = 64, 128, 256, 512, 1024. The data and coefficient widths are tunable in the range 8 to 18. The model is capable of performing filtering operations like low pass, high pass, band pass and band stop based on selection that is embedded into the design. The most basic functions required for nearly any signal processor include addition, multiplication and delays. Input data, output data, and coefficient widths are generics. The maximum sampling frequency Fs by N=1024 is less than Fclk/29. IP Corse has been used to filter the input data. The design is coded through VHDL (hardware descriptive language). To verify the designed outputs simulation, compilation and synthesis have been done.

Introduction

Digital signal processing (DSP) is used in a very wide range of applications from high-definition TV, mobile telephony, digital audio, multimedia, digital cameras, radar, sonar detectors, biomedical imaging, global positioning, digital radio, speech recognition, to name but a few!

The increasing costs of silicon technology have put considerable pressure on developing dedicated SoC. An alternative is to use microprocessor style solutions such as microcontrollers, microprocessors and DSP micros, but in some cases, these offerings do not match well to the speed, area and power consumption requirements of many DSP applications. More recently, the field-programmable gate array (FPGA) has been proposed as a hardware technology for DSP systems as they offer the capability to develop the most suitable *circuit architecture* for the computational, memory and power requirements of the application in a similar way to SoC systems. FPGAs are a collection of system components with which the user can create a DSP system. Whilst the prefabricated aspect of FPGAs avoids many of the deep submicron problems met when developing system-on-chip (SoC) implementations, the ability to create an efficient implementation from a DSP system description, remains a highly convoluted problem.

FIR filter has been designed and realized on FPGA for filtering the digital signal. This technique can be applied to any FPGAs. Signal processing is an important area where FPGAs have found many applications in recent years. FPGA contains over a million equivalent logic blocks (logic gates and tens of thousands of flip-flops) [1][2].This means that it is not possible to use traditional methods of logic design involving the drawing of logic diagrams when the digital circuit may contain thousands of gates. The reality is that today digital systems are designed by writing software in the form of hardware description languages (HDLs) [5]. Computer-aided design tools are used to both simulate VHDL design and to synthesize the design to actual
hardware.

The designing of an FIR filter in VHDL for programming it onto an FPGA is explained in this paper. Implementation of project onto an FPGA (including hardware and software parts) VHDL, and basic digital filter concepts are used.

DESIGN FEATURES

Small hardware volume

The proposed Design is intended for the signal filtering with the FIR filter of large impulse response length which exceeds up to Ni = 512 samples.

Dynamically tuneable band pass frequencies

In many applications the user needs the filters which band pass frequencies are tuned dynamically. They are adaptive filtering, software defined radio, ultrasound testing devices, etc. It is not easy problem to perform this mode in the usual FIR or IIR filters. This problem is usually solved by storing a set of coefficients of different filters or by calculating the new coefficient set each time on demand.

Highly pipelined calculations

Each FFT iteration dates are computed by the computational unit, called FFTDPATH, another words, data path for FFT calculations. FFTDPATH calculates the radix-2 FFT butterfly in thehigh pipelined mode. Therefore in each clock cycle one complex number is read from the data RAM and the complex result is written in this RAM. This mode supports the increasing the clock frequency up to 80 MHz and higher.

High precision computations

In the core the block floating point arithmetic is implemented. This means that the data array has the common exponent, and the array is normalized in the mode when the maximum data in the array occupies all the digits of the word. Such mode supports the high calculation precision. Due to this mode, 1024 – point FFT calculations for 16 bit data and coefficients give 70 db signal to noise ratio, which is at least at 20 db higher than calculations with the fixed point arithmetic give.

Combining the band pass filter with differentiators

In many applications the user needs to combine the band pass filter with differentiators. For example, in ultrasound testing devices the transducer has the integrator properties, which have to be compensated by differentiators. Therefore the system needs to put band pass filter and one or two differentiators sequentially. In this situation the proposed Design is the best solution because this mode is implemented in it naturally without additional hardware.

Additional frequency measurements

Often the user needs to investigate the input signal spectrum, for example, to find out the noisy frequency bands. To implement this feature the proposed FFT core has additional output for signal spectrum samples or bins. This output is attached/detached on demand when instantiating the core.

FILTERING ALGORITHM

One channel real signal filter

The sectioned convolution algorithm is used for the one channel complex signal filtering. Consider N = 1024. This algorithm for convolution of the signal a with the impulse responce h looks like the following.

- Input signal is divided into segments ak of the length 512.
- The working array a of the length 1024 is formed as the concatenation of this segment and previous one:
- a = <a_{k-1} , a_k >.
- FFT of the length 1024 for the working array is implemented: A = F(a).
- FFT of the length 1024 for the impulse responce is implemented: H = F(h); note that more than a half of the array h has to be zeroed.
- The signal spectrum and the impulse responce spectrum (frequency responce) are multiplied: A*H
- Inverse FFT of the length 1024 is derived: y = F-1 (A*H).
- 512 resulting samples are selected which are not inferred by the circular convolution effect: $y_k = \{y_p, \ldots, y_{p+511}\}$, p = 256.

The following considerations have to be mentioned. The impulse responce *h* may not be transferred into the frequency responce *H*. Instead the frequency responce *H* can be generated due the parameters of low pass frequency *F*l and high pass frequency *F*h. It has to be symmetric one and has more than 512 zeroed samples.

The initial algorithm is true for the signals, which are represented by the sum of sinusoids which periods are the fractions of the FFT period. If the signal is of common form then it could not be filtered precisely by this algorithm due to the frequency aliasing effect. To minimize this effect the input signal has to be multiplied by some time window W. The resulting filtering algorithm for the real input signal is represented by the diagram on the following Fig.1.

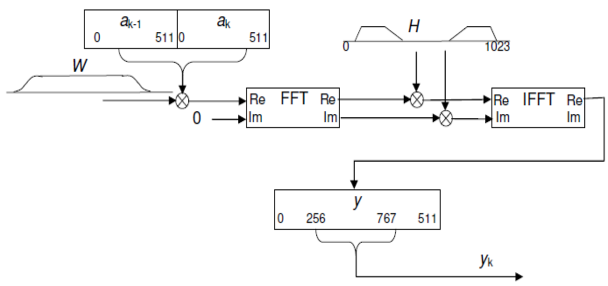

Figure 1. The filtering algorithm for a single channel

Two filters for a single real signal

When filtering a single real signal with two different filters the input signal spectrum is just the same for both filters. But the frequency responce *H*2 of the second filter differs from the frequency responce *H*1 of the first filter. To minimize the algorithm complexity the spectrum symmetry is used. If we have the real signal y1 with the spectrum (*Y*R1 + j*Y*I1) on the real input of FFT, and the real signal y2 with the spectrum (*Y*R2 + j*Y*I2) on the imaginary input of FFT, then after FFT we get the spectrum:

$$Y_R = Y_{R1} - Y_{I2}; \qquad (*)$$

$$Y_I = Y_{I1} + Y_{R2};$$

Therefore if the spectrum of both signals is fore calculated according to (*), then after IFFT we get one signal as the real part, and another signal as the imaginary part of the result. The resulting algorithm diagram is shown on the Fig.2.

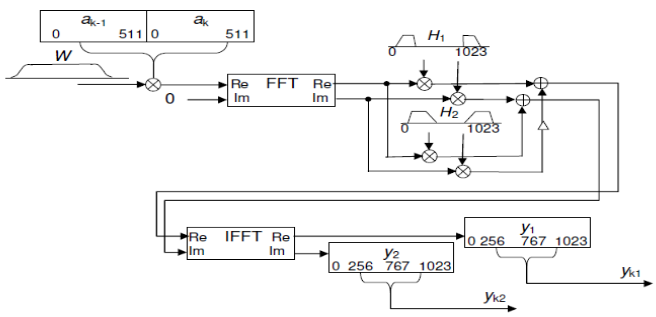

Figure 2. Algorithm of two filters for a single real signal

Two filters for two real signals

The filtering of a single input signal is performed with the abundance of operations because the maginary part of the input data is zeroed. This abundance is minimized when the imaginary part of FFT is data of another input signal (second channel). I.e. the FFT input *x* is formed as:

$x = a + \mathrm{j}b$,

where $a = <a_{k-1}, a_k>$, $b = <b_{k-1}, b_k>$.

After FFT the spectres of channels are restored from the spectrum X due to the formulas:

$A_{Ri}= (X_{Ri} + X_{R(1024-i)})/2$; $A_{Ii}= (X_{Ii} - X_{I(1024-i)})/2$;
$B_{Ri}= (X_{Ii} + X_{I(1024-i)})/2$; $B_{Ii}= - (X_{Ri} - X_{R(1024-i)})/2$;
$i=1,2,\ldots,511$;
$A_{R0}= X_{R0}$; $B_{R0}= X_{I0}$;
$A_{R512}= X_{R\,512}$; $B_{R512}= X_{I512}$;
$A_{I0}= A_{R512}$; $B_{I0}= B_{R512}$;
$A_{I512}= 0$; $B_{I512}= 0$;

where R and I are indexes of the real and imaginary parts respectively. The rest of calculations is performed in the same manner as by the filtering of a single real signal by two filters.

Differentiating

The differentiating of the real signal is equal to multiplying its spectrum at the frequency w to the coefficient jω ($-\pi < \omega < \pi$). By the sectioned convolution it is enough to multiply the real part of the i-th spectrum bin to the coefficient i, and the imaginary part to the coefficient –i, and to swap them.

Time and frequency windows

Frequency window H derives the selective properties of the filter. The rectangle window gives the shortest transitional frequency band. But it is bad because its IFFT has not zeros, and therefore it causes the aliasing effect.
In the Proposed design core the Blackman window is used which has not ripples in the band pass, and provides the suppression range more than 70 db. The time window consists of three parts. The first ant the third parts represent the halves of the Hanning window, and the second part is equal to 1.

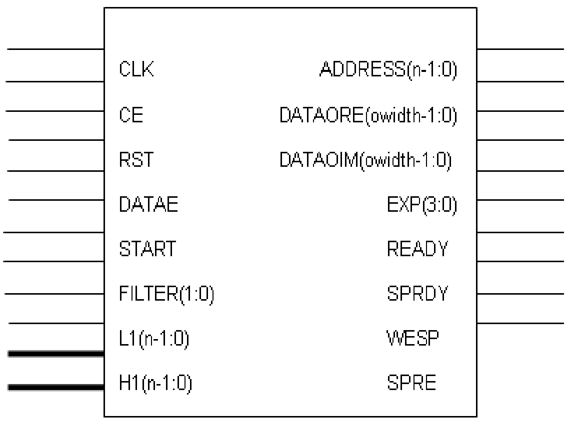

Fig.3 illustrates top level entity of proposed FIR filter

SIGNAL DESCRIPTION

The descriptions of the core signals and generics are represented in the table 1.

Signals	Type	Description
GENERICS		
iwidth	natural	Input data width = 8,…,18
owidth	natural	Output and intermediate data width = 8,…,18
wwidth	natural	Coefficient width = 8,…,6
n	natural	FFT length code:6–64,7–128,8–256,9–512,10–1024
reall	natural	0 – complex, 1 – real Input and output signals
SIGNALS		
CLK	Input	Global clock
RST	Input	Global reset
START	Input	Filter start
DATAE	Input	data enable strobe
FILTER	Input	00 – without filtering, 01 – LPF , LPF+HPF, 10 – LPF+HPF+ differentiator, 11 – LPF+HPF+ double differentiator
L1	Input	Low band pass frequency of the first filter
H1	Input	High band pass frequency of the first filter
L2	Input	Low band pass frequency of the second filter
H2	Input	High band pass frequency of the second filter
DATAIRE	Input	Input data real sample (first channel)
DATAIIM	Input	Input data imaginary sample (second channel)
READY	output	Result ready strobe
DATAOR	output	Output data real sample (first channel)
DATAOI	output	Output data imaginary sample (second channel)
SPRDY	output	Spectrum start output impulse
WESP	output	Spectrum sample strobe
SPRE[owi	output	Spectrum real part sample
SPIM[owi	output	Spectrum imaginary part sample
FREQ	output	Spectrum bin number
SPEXP[3:	output	Spectrum data block exponent

Table 1. Signal Description of Design

DATA REPRESENTATION

Input and output dates are represented by *iwidth* and *owidth* bit two-th complement complex integers, respectively. The spectrum data block exponent is 4-bit positive integer *e*, and the spectrum result Y is equal to $Y=Y_m*2^{\theta}$, where Y_m is the real or imaginary part of the spectrum data. The exponent is the same for each sample of the result array.
The code of the band frequency is equal to the bin number where the filter pass level is equal to –3 db. Codes L1,L2 have to be less than respective codes H1,H2. For instance, for Fs =2500 kHz, N=1024, and LPF with the band pass 400 kHz the code H1=164 because 400*1024/2500 =163.84 . If L1 = 0 or L2 = 0 then the respective HPF is detached.

VERIFICATION

To verify the proposed design before synthesis and after it and after implementation the following files can be additionally used:
FFT_Filter_tb.VHD – the test bench file;
RAMB4_S18_S18.vhd – behavioural model of the Block RAM, can be substituted to the similar Unisim model.
In the testbench the proposed design is instantiated as the component in the standard instantiation. To the core inputs the sine and cosine waves are put with the given frequency which is exchanged in time by the linear law. From the core outputs the results are sampled and analyzed.
The analysis consists in measurement of the complex vector magnitudes, their averaging and logarithm representation. The resulting signals of the testbench are:

res – result magnitude;
reslog – logarithm of the result magnitude (in decibels);
freque – sine wave frequency.

As a result, after modeling one can investigate in the VHDL simulator the frequency response of the filters by the given set of control signals. For instance, Fig. Illustrates the frequency responce of the band pass filter, and the Fig.9 illustrates the same of the band pass filter with the double differentiator.

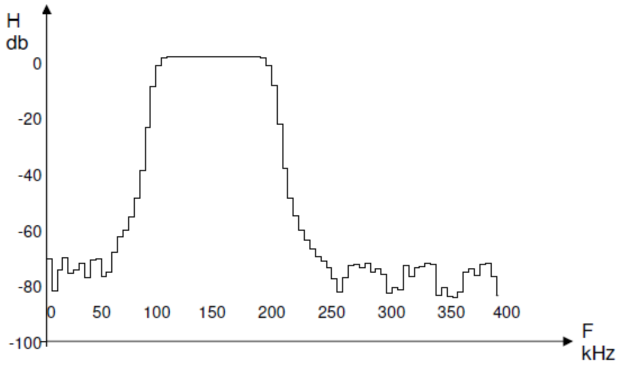

Figure . Frequency response of the band pass filter with the band pass 100 – 200 kHz by Fs=2500 kHz and N=1024. 1 stage width is 5 kHz.

References

[1] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix(CFMR) FFT processor using novel in-place strategy," *IEEE Trans.Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 911–919, May 2005.

[2] P. Y. Tsai and C. Y. Lin, "A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 12, pp. 2290–2302, Dec. 2011.

[3] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. IEEE Int. Parallel Process. Symp.*, Apr. 1996, pp. 766–770.

[4] A.Cortes, I. Velez, and J. F. Sevillano, "Radix *rk* FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.

[4] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug. 2005.

[5] S.-H. Hsiao and W.-R. Shiue, "Design of low-cost and high-throughput linear arrays for DFT computations: Algorithms, architectures, and implementations," *IEEE Trans. Circuits Syst. II, Analog Digital Signal Process.*, vol. 47, no. 11, pp. 1188–1203, Nov. 2000.

[6] Y.-W. Lin and C.-Y. Lee, "Design of an FFT/IFFT processor for MIMO OFDM systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 4, pp. 807–815, Apr. 2007.

[7] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar. 1999.

[8] M. S. Patil, T. D. Chhatbar, and A. D. Darji, "An area efficient and low power implementation of 2048 point FFT/IFFT processor for mobile WiMAX," in *Proc. Int. Conf. Signal Process. Commun.*, 2010, pp. 1–4.