FPGA Implementation of Variable Digital Filter using MicroBlaze Processor

M. Rupa[1] and Rayabandi Karthik Kumar[2]
[1]CVR College of Engineering, Department of ECE, Ibrahimpatan, R.R.District, A.P., India
Email: arupa_ece@yahoo.com
[2] CVR College of Engineering, Department of ECE, Ibrahimpatan, R.R.District, A.P., India
Email: karthikrayabandi@gmail.com

***Abstract*—Digital filters have found numerous applications in most digital signal processing systems. Conventional digital filter can only obtain fixed frequency domain characteristics at a time. In order to obtain variable characteristics, the digital filter's type, number of taps and coefficients should be changed constantly such that the desired frequency-domain characteristics can be obtained. Unlike conventional digital filter, Variable Digital Filters (VDFs) can change their filter-type, number of taps and coefficients constantly such that the desired frequency-domain characteristics can be obtained. This paper proposes a method for Variable Digital Filter (VDF) design based on Field Programmable Gate Array and Embedded Micro-Processor (EMP). This VDF is best suited for realization of digital filter algorithms, which are low-pass, high-pass, band-pass and band-stop filter algorithms with variable frequency domain characteristics. The design has been prototyped on an XUPV5-LX110T Evaluation Platform using Integrated Synthesis Environment (ISE) 12.4 Tools all in one design suit from Xilinx.**

***Index Terms*—Variable Digital Filter; Field Programmable Gate Array; Embedded Micro-Processor**

I. Introduction

With the recent rapid advances in communication systems and real-time signal processing, there has been a constant interest in the design and implementation of digital filters with variable frequencydomain characteristics such as variable cutoff frequency, adjustable passband width. The digital filters with variable frequency domain characteristics are referred to as Variable Digital Filters.

Digital filters are mainly two types- Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) digital filter. The response of an IIR filter is a function of current and past input signal samples and past output signal samples. The dependency on past outputs (i.e., recursive) gives rise to the infinite duration of the filters output response even when the input values have been stopped. A Finite Impulse Response (FIR) filter is a filter structure that can be used to implement almost any sort of frequency response digitally. An FIR filter is usually implemented by using a series of delays, multipliers, and adders to create the filter's output. If the output samples of the system depend only on the present input, and a finite number of past input samples, then the filter has a finite impulse response.

In the most general case of a variable filter, all filter magnitude parameters might be subject to changes. Variable magnitude deviations δ_s (in the stopband) and δ_p (in the passband) are, however, not easily achievable – complicated recalculations of the entire transfer function are required and usually many circuit elements must be changed in order to obtain the new δ_s and δ_p.

Fortunately, more often, only frequency parameters – cutoff frequency w_c and stopband edge w_s of low-pass (LP) and high-pass (HP) filters or w_{c1}, w_{c2}, w_{s1}, w_{s2}, center frequency w_0 and bandwidth BW of band-pass (BP) and band-stop (BS) filters – are tuned.

The multipurpose FIR filter is designed using windowing technique. The windowing method is the preferred algorithm because it is simplest method of FIR filter implementation, the selection of the window is based on the causality and stability of all the filter types, low-pass, high-pass, band-pass, and band-stop.

Most of the multipurpose FIR filter designs use commercial tools such as MATLAB's Filter Design and Analysis (FDA) tool for its implementation. MATLAB's fir1 and fir2 function is utilized to compute the filter coefficients [1]. The implementation of such multipurpose FIR filter starts with calculation of coefficients using MATLAB, followed by HDL program and its implementation [1] [2].

This paper proposes a method for implementation of variable digital filter, eliminating the usage of MATLAB functions for coefficient calculations, by using embedded soft RISC processor core. The processor core is optimized for implementation in XILINX FPGAs, and supports both on-chip Block RAM and external memory.

II. Basic Principle Of FIR Filter

The transfer function for an FIR filter of length N is given as:

$$H(z)=\sum_{k=0}^{N-1} h(k)z^{-k}$$

The design of a digital filter involves five steps[3]:

1. Filter Specification:

This may include stating the type of the filter, for example lowpass filter, the desired amplitude and/or phase responses and the tolerances prepared to accept, the sampling frequency, and the wordlength of the input data.

2. Coefficients calculation:

At this step, one determines the coefficients of a transfer function, H(z), which will satisfy the specifications given in step 1. The choice of the coefficient calculation method will be influenced by

 DOI: 10.32377/cvrjst0307

several factors, the most important of which are the critical requirements of step 1.

3. Representation of the filter by a suitable structure (realization):

This involves converting the transfer function obtained in step 2 into suitable filter network or structure.

4. Analysis of the effects of finite word length on filter performance:

Here the effects of quantizing the filter coefficients and the input data as well as the effect of carrying out the filtering operation using fixed wordlength on the filter performance are analyzed.

5. Implementation of filter in software and/or hardware:

This involves producing the software code and/or hardware and performing the actual filtering.

III. Implementation Of Variable Digital Filter

The implementation of Variable Digital Filter consists of

- MicroBlaze (Embedded Micro-Processor)
- User-machine interface
- Shared memory
- Programmable FIR Filter

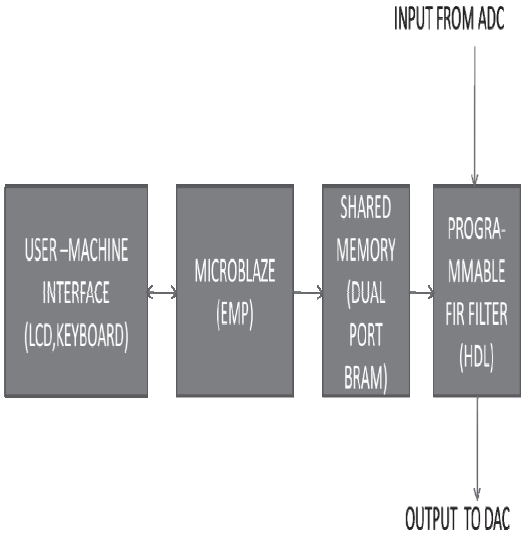

Figure 1. Block Diagram of VDF

A. MicroBlaze (Embedded Micro-Processor)

The *MicroBlaze* soft core processor is used for the computation of coefficients of Variable Digital Filter as per the user specifications. The Embedded Development Kit [4] [5] is used to create a hardware design composed of IP cores and a MicroBlaze soft processor. The design is completed by writing a software application to run on the MicroBlaze processor. The software application controls the functionality of different IP cores added to the processor. The C programming language is used for developing the software application. The below figure 2 shows the block diagram of a MicroBlaze processor based system. The IP cores are connected with MicroBlaze processor using Processor Local Bus v4.6.

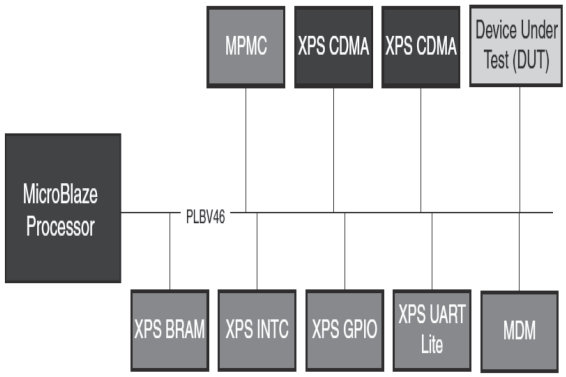

Figure 2. MicroBlaze Processor System[4]

The MicroBlaze processor acts as one of the sub-module in the implementation of VDF.

B. User-machine interface

The implementation of variable Digital Filter uses Liquid crystal display (LCD) and PS/2 keyboard as user-machine interface modules. The LCD module is used to guide the user, such that the required filter specifications can be provided to the processor by using PS/2 keyboard module. The LCD and PS/2 keyboard modules are interfaced with MicroBlaze processor. The MicroBlaze processor monitors the functionality of these modules.

C. Shared memory

A ***Dual Port Block Random Access Memory*** (DP-BRAM) is used to store the coefficient values computed by the MicroBlaze processor. Dual Port BRAM is implemented by adding Block RAM (BRAM) IP Block and XPS BRAM Controller IP to the MicroBlaze system.

The BRAM Block is a configurable memory module that attaches to a variety of BRAM Interface Controllers. Both Port A and Port B of the memory block can be connected to independent BRAM Interface Controllers: Local Memory Bus, Processor Local Bus, and On-Chip Memory.

The XPS BRAM Interface Controller is a Xilinx IP module that incorporates a PLB V4.6 interface. This controller is designed to be byte accessible. Any access size (in bytes) up to the parameterized data width of the BRAM is permitted. The XPS BRAM Interface Controller is the interface between the PLBV4.6 and the BRAM block peripheral. A BRAM memory subsystem consists of the controller along with the actual BRAM components that are included in the BRAM block peripheral.

The filter coefficients computed by the MicroBlaze processor are in the IEEE-754 single precision format, which are stored in the memory using Port A of DP-BRAM.

D. IEEE-754 Single Precision Format

MicroBlaze processor supports single precision IEEE 754 format for representing floating point values. IEEE 754 format is most common standard for representing floating point numbers.

Single precision: 32 bits, consisting of

1. Sign bit (1 bit): The sign bit is as simple as it gets. 0 denotes a positive number; 1 denotes a negative number. Flipping the value of this bit flips the sign of the number.

2. Exponent (8 bits): The exponent field needs to represent both positive and negative exponents. To do this, a *bias* is added to the actual exponent in order to get the stored exponent. For IEEE single-precision floats, this value is 127. Thus, an exponent of zero means that 127 is stored in the exponent field.

3. Mantissa (23 bits): The *mantissa*, also known as the *significand*, represents the precision bits of the number. It is composed of an implicit leading bit and the fraction bits.

The real value assumed by a given 32 bit **binary32** data with a given biased exponent **e** and a **23 bit fraction** is = $-1^{\text{sign}} (1.\, b_{-1}\, b_{-2} \ldots b_{-23})_2 \times 2^{(e-127)}$.

E. Programmable-FIR Filter

A 15-tap Programmable-FIR filter is developed as a sub-module using Verilog HDL language in Xilinx ISE. The FIR filter is named as programmable; because it can change its filter order, type as per the data received from the MicroBlaze processor. Optimized form of FIR filter structure is used for developing the P-FIR filter. The Port B of DP-BRAM is used by the P-FIR filter to read the data from memory. The order of the P-FIR filter can be changed in odd values. A Push button is used to initialize the P-FIR, after the execution of MicroBlaze sub-module.

F. Process Description

An 8 bit data from the output of an ADC is taken as input of the P-FIR filter. The filter coefficients read from the memory are in IEEE 754 single precision format. To simplify the filter computation process, the IEEE 754 single precision value is converted into a Fixed-Point value (Q11.12) using Floating Point IP. Q11.12 represents 1 sign bit, 11 bit integer value and 12 bit fractional value. The multiplications and addition operations are carried according to the optimized structure of the filter. The 12 Least Significant Bits (LSB) are truncated from the final result. The filter output can be used for further signal processing.

A 125 MHz clock is provided as clocking resource to the MicroBlaze processor by the clock generator core using 100 MHz user clock. The Programmable FIR filter is sourced with 33 MHz on-board clock.

IV. RESULTS

The results obtained by implementing Variable Digital Filter using MicroBlaze on Virtex-5 board are shown. The MicroBlaze processor sub-module is executed at the beginning. This sub-module takes filter specifications entered by the user as inputs and computes the respective filter coefficients.

Table I. compares the MicroBlaze computed filter coefficient values with cutoff frequency = 1.5 KHz, sampling frequency = 8 KHz and filter length = 7 for LPF.

TABLE I.
Comparison of filter coefficient values for LPF

Filter coefficients	Low-Pass Filter			
	Theoretical		Obtained	
	Floating-point	Single-precision	Single-precision	Floating-point
h[0]	-0.077381	BD9E79EE	BD9E79E9	-0.07738096
h[1]	0.212798	3E59E7B8	3E59E7A0	0.21279764
h[2]	0.793155	3F4B0C35	3F4B0C31	0.7931548
h[3]	1.083333	3F8AAAA8	3F8AAAA9	1.0833331

The Programmable-FIR filter sub-module is initiated after the execution of MicroBlaze sub-module. This sub-module reads coefficient values computed by the MicroBlaze sub-module from DP-BRAM to configure the filter structure. An 8-bit data is provided as input to the P-FIR filter module. The 24 bit output of P-FIR filter is used for further signal processing. Figure 3 shows the simulation results of P-FIR filter.

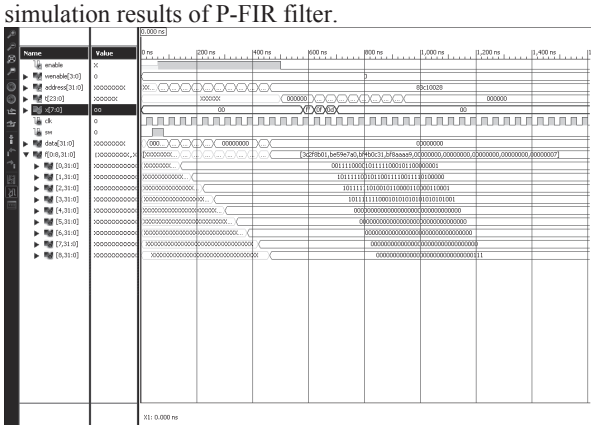

Figure 3. Simulation Result of HPF Filter for fc=1.5 KHz, fs=8 KHz & N = 7

CONCLUSION

Today many embedded products place their solution on several chips making it bigger, more expensive and more power requiring. FPGA boards has become bigger, faster and cheaper and is now able to handle a SoC solution. As the FPGA boards have become bigger and faster they are now able to handle a soft processor which is an Intellectual Property (IP) core implemented using logical primitives. A key benefit is configurability where it is possible to add only what is needed in the design. A trade off is performance, a hard processor is faster but less configurable and more expensive.

With a soft-core processor, a design becomes more flexible, while still keeping high performance parts inside the FPGA. Thus the Xilinx MicroBlaze 32 bit RISC soft-core processor can be easily adapted to the designer's needs and full- featured GNU tool chain is used for software development.

DOI: 10.32377/cvrjst0307

Variable Digital Filter using MicroBlaze processor has been implemented on Virtex-5LX110T board. The processor is operated at a clock frequency of 125 MHz. Variable Digital Filter is assigned to a frequency of 33MHz, and the maximum speed of operation that can be achieved is 56MHz.

REFERENCES

[1] Md.Syedul Amin, Md.Mamun, Labonnah F. Rahaman,"Hardware Approach of a Multipurpose Finite Impulse Response Filter for Real-Time Filtering Applications", American Journal of Applied Sciences.

[2] R.Yousefi, A.Ahmadi, S.M.Fakhraie, "Design Flow for Hardware Implementation of DigitalFilters", International Symposium on Telecommunications, 2008.

[3] [3] E.C.Ifeachor, B.W.Jervis, "Digital Signal Processing-A Practical approach", Prentice Hall, 2nd Edition, 2002.

[4] MicroBlaze Processor Reference Guide *Embedded Development Kit EDK 12.4*

[5] Embedded System Tools Reference Manual EDK 12.4