AN FPGA IMPLEMENTATION OF THE LMS ADAPTIVE FILTER FOR ACTIVE VIBRATION CONTROL

Shashikala Prakash[1], Renjith Kumar T.G[2], Subramani H[3]

[1] Sr. Principal Scientist, [2] Scientist Fellow, [3] Project Engineer, STTD, CSIR-NAL, Bangalore – 560017, India,
shaship@nal.res.in**, **renjithkumartg@gmail.com**, **umasubbu28@gmail.com

Abstract

This paper brings out implementation of Least Mean Square (LMS) algorithm using two different architectures. The implementations are made on Xilinx Virtex–4 FPGA as part of realization of an Active Vibration Control system. Both fixed point and floating point data representations are considered. A comparison between the two is brought out on the basis of a Finite State Machine (FSM) model suitable for both fixed & floating point implementations. The floating point LMS algorithm in VHDL (Very High Speed Integrated Circuit (VHSIC) Hardware Description Language), uses the Intellectual Property (IP) cores available from Xilinx Inc. Results from the two architectures with respect to area as well as performance clearly shows floating point implementation to emerge as the better option in all respects.

Index Terms: *Least Mean Square Algorithm, Field programmable gate arrays (FPGA), floating point IP cores, Finite State Machine, Active Vibration Control.*

----------***----------

1. INTRODUCTION

The Active techniques are best suited for vibration control in Aircraft/Aerospace structures where the size, weight, volume and cost are very crucial. Active vibration control can be achieved on typical structures on the basis of principle of super position theorem in which an opposite phase signal is generated using the control software and is superimposed on the structure in order to cancel out or suppress the effect of primary vibrations. But the system characteristics are more dynamic in nature, so adaptive systems are more appropriate and efficient in performance compared to conventional systems with fixed structure. Adaptive systems have the ability to track changes in system parameters with respect to time and provide optimal control over much broader range of conditions. The LMS algorithm is a widely used technique for adaptive filtering. The Least Mean Square algorithm is an adaptive algorithm introduced by Widrow and Hoff in 1960[1-5]. A reference input received / monitored using either accelerometer or smart PZT (lead zirconate titanate) sensor is suitably manipulated in the control filter, and the filter response (which is a spectrum of cancellation force) is applied in an equal and opposite direction using PZT/MFC (Macro Fiber Composite) actuators. The Adaptive control algorithm dynamically updates the filter coefficients based on signal obtained from another set of error sensors. LMS algorithm is best suited for Vibration control because of its simplicity and also various advantages it has over other adaptive algorithms [3]. LMS algorithm is implemented using tapped delay line FIR (finite impulse response) filter structure, and is preferred as it does not require matrix inversion, off-line gradient estimations of data and also for the ease of implementation on finite hardware.

The implementation of adaptive filter could be done using an ASIC (Application-Specific Integrated Circuit) custom chip, general purpose processor (GPP) or DSP processors. Though ASIC meets all the hard constraints, it lacks the flexibility that exists in the other two, and its design cycle is much longer involving huge expenditure. Reconfigurable systems for prototyping of digital system are very advantageous. Using reconfigurable devices like FPGAs for Digital Signal Processor (DSP) applications provides the flexibility of GPP, DSP processors, and the high performance of dedicated hardware using ASIC technology [6]. Modern FPGAs contain many resources that support DSP applications such as embedded multipliers, multiply accumulate units (MAC), and processor cores. These resources are implemented on the FPGA fabric and optimized for high performance and low power consumption. Also many soft IP cores are available from different vendors that provide a support for the basic blocks in many DSP applications [7, 15, 16].

The FPGAs can embed more and more functionality into a small silicon area without compromising any of the performance parameters like speed and accuracy. FPGAs can be viewed as a structural decomposition of an array of configurable logic blocks integrated together to form any complex digital systems. The programming procedure and usage is much similar to its predecessors like microprocessor/

DSP families but the dynamic-reusability and re-configurability of its own individual hardware elements makes it most suited for the embedded systems applications. In the recent years FPGA systems are preferred due to their greater flexibility and higher bandwidth, resulting from the parallel architecture.

There have been many efforts to implement LMS algorithm in FPGA platforms. Mohammad Bahura and Hassan Ezzaidi [8] have presented a sequential architecture on FPGA using Virtex-II-Pro development board and Xilinx system generator (XSG). Here a pipelined LMS based adaptive noise cancellation is implemented to remove power line interference from electrocardiogram (ECG). Fixed point LMS algorithm implementation in FPGA is described in [9] along with the effect of word length in the expected results from the theoretical values. An improved hardware architecture of LMS algorithm implementation in FPGA is presented in [10] and the performance in terms of amount of resources is compared with a Software architecture. A FSM based LMS adaptive filter implemented for Active Noise control systems is described in [11].

All the above mentioned papers use fixed point number representation. An active vibration control system, when implemented on the finite hardware should provide precise results in terms of accuracy and the error level should be as minimal as possible to achieve maximum control. Floating point representation has an edge over the fixed point representation in this context. A comparison between the fixed and floating point approach in terms of its implementation in FPGA hardware has been made in terms of resource utilization, accuracy & speed. To this end, both fixed and floating point method based LMS implementations have been realized using the same FSM model and the performance is tested with respect to the FPGA resources utilized, speed of convergence and accuracy level. In this paper the outcome of the study is presented.

The paper is structured as follows. In Section II LMS algorithm is briefly described. III (A) gives a review of the number representations. III (B) gives the state machine model which is used for fixed and floating point implementations. It also explains how FPGA incorporates parallelism with FSM. III (C) tells about the fixed point implementation, III (D) describes floating point implementation with IP cores, showing how resource re-usability is improved. Results and discussions are given in Section (V).

2. LMS ALGORITHM

2.1 Introduction

The LMS algorithm is an adaptive algorithm using an iterative process, in which the mean square error is minimized by the method of steepest descent by moving in the direction of negative gradient. The block diagram representation is shown in Fig-1. It is simple, easy to implement in finite hardware and it does not require complex calculations such as matrix inversions or correlation functions. Hence it is widely used in active vibration control of aerospace structures.

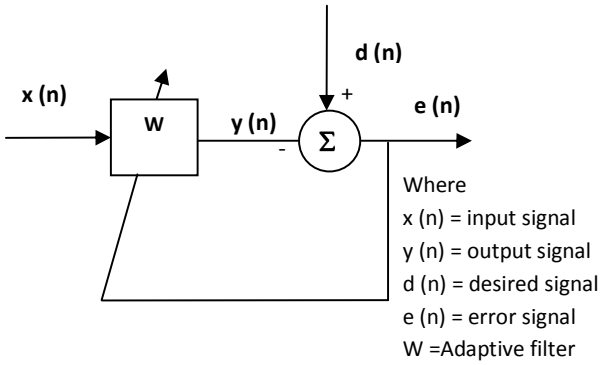


Fig1. Block diagram of an LMS adaptive filter.

The LMS Algorithm consists of three basic processes:

2.1.1 Filtering Process

The output of filter y(n) is a linear combination of input signal X(n) and the weight vector W(n), [where X(n)={x(n),x(n-1),x(n-2),…x(n-N-1)}, and W(n)={w(0),w(1),w(2),…w(N-1) and N, filter length] and the output of the filter is

$$y(n) = X(n) * W(n) \quad (1)$$

* indicates convolution operation

2.1.2 Error Estimation

Error signal, e(n) is obtained by comparing the filter output with the desired response d(n). The aim is to make y(n) as close as possible to d(n), thus error signal is

$$e(n) = d(n) - y(n) \quad (2)$$

2.1.3 Adaptation Process

Mean-square error performance function f (w) is

$$f(w) = E\{|e(n)|^2\} \quad (3)$$

According to the least mean square criterion, the optimal filter parameter w should minimize the error performance function f(w). Using gradient-descent methods, the weight update formula is

$$w(n+1) = w(n) + \mu . e(n) . x(n) \quad (4)$$

Where μ is the adaptive step size parameter and it controls the convergence characteristics of the filter.

3. FPGA IMPLEMENTATION OF LMS ALGORITHM

3.1 Review of Number Representation

FPGA implementation is an overall process of converting a higher level system description into a lower level implementation in terms of signal flow through hardware circuits, as bit streams. Data flow in the form of these streams of bits can be represented either in fixed point or floating point number format and arithmetic.

3.1.1 Fixed Point Number Representation

Fixed point representation assigns a fixed width to integer and fraction. For example, for a 32 bit number can be considered 16 integer bit and 16 fractional bits (shown in Fig-2), which can go up to the range of 2^{16}-1 to -2^{16} (integer part i.e., 65535 to -65536) and with a fractional range of $1/2^{16}$ (i.e., 0.0000152587890625) .

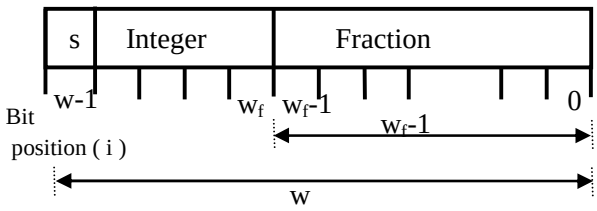


Fig2. Fixed point number representation. Here w and w_f represents total width and fraction width respectively

3.1.2 Floating Point Number Representation

Floating point numbers are, in general, represented approximately to a fixed number of significant digits (the mantissa) and scaled using an exponent. The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

Significant digits × base exponent

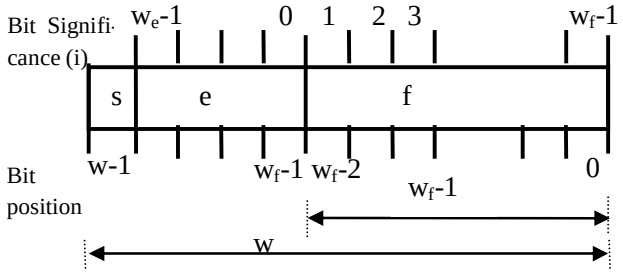


Fig3. IEEE 754 Floating point number representation. Here w, w_e and w_f represents total width, exponent width and fraction width respectively

Floating point representation with its exponent component achieves greater range compared to conventional fixed point representation. For example, consider IEEE 754 floating point format [12] number with 32 bit single precision, implemented as 23 bit mantissa (or fraction, f), 8-bit exponent(e) and sign bit(s) described in Fig-3. The 24 bit mantissa (including sign bit) can achieve a precision of 16M (2^{24}) compared to the 6K (2^{16}) of fixed point format. The remaining 8-bit exponent offers enormously larger dynamic range (in the order of 2^{64}), compared with the fixed-point format.

The LMS algorithm with a behavioral model using fixed point packages is described in our paper [13]. But a structural model system design is preferred since it takes into account the efficiency with which FPGA resources can be configured and interconnected to achieve the desired functionality. Hence a structural model with FSM is developed to explore the parallel processing capabilities of FPGA & this is depicted in the coming chapters.

3.2 Structural Model with FSM.

The LMS algorithm with structural modeling decomposes the system into smaller modules each with unique functionalities. These modules are synthesizable on the hardware using basic functional elements such as Look-up Tables and Gates. These modules are paralleley executable and reusable. The FSM methodology is used to interconnect these individual modules and synchronize their operation in the most efficient way. The FSM design is based on mealy model in which output value depends both on the present state and the input.

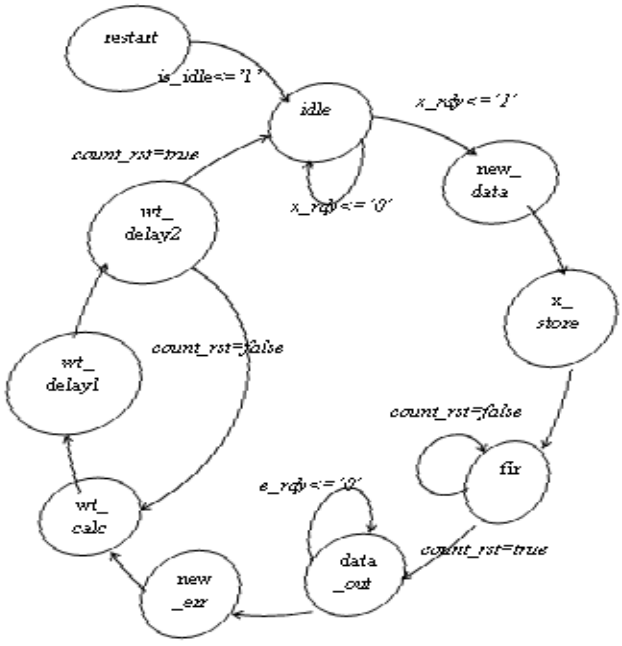


Fig4. FSM diagram of an adaptive filter

The Fig-4 shows the FSM model developed for implementing LMS algorithm on FPGA Hardware in VHDL. The LMS algorithm is divided into 9 states.

One of the main problems faced during design and implementation of such a modular structure was the synchronization of data reading from the buffers along with FIR or weight updation operations. The design becomes all the more complex since the input buffer is circular. So a FSM with Data path model is chosen as a solution to this problem.

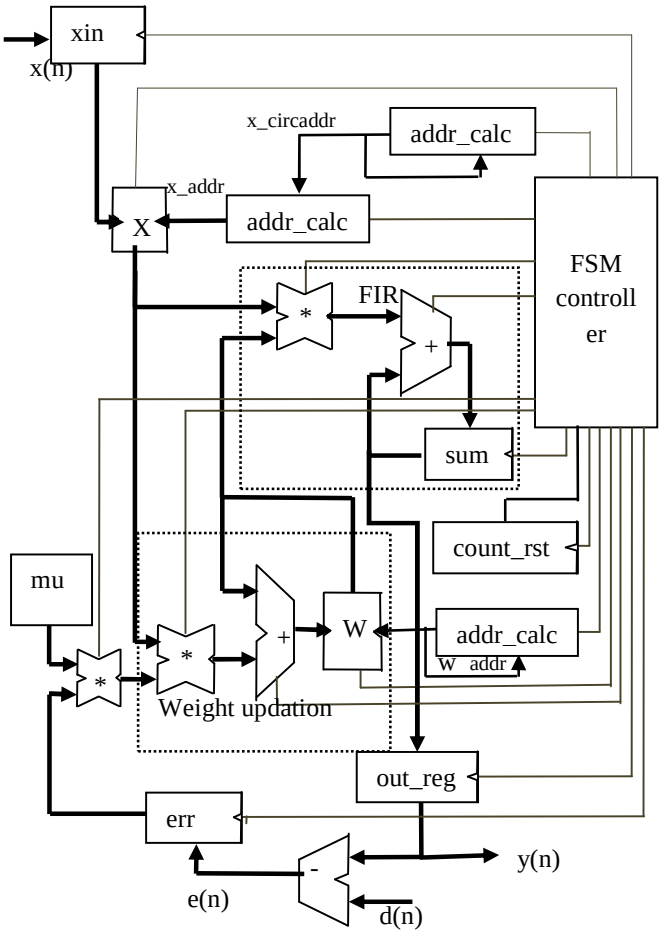


Fig5. LMS adaptive Filter Functional diagram

The FSM with Data path model divides the system design into two parts, a control part (or controller) and an operative part (or data path). The system functionalities such as address generation of input circular buffer/ adaptive filter coefficient buffer, FIR filter output calculation, Weight updation, saving of input and output data in register or memory, and acquiring input and sending out filter output samples are implemented as separate data path modules. The data path unit is implemented with the idea of pre-computing the output values at least half a clock cycle before they are required. The data path modules are invoked either at rising or falling edge of the clock such that its output can be synchronized with other processes or the state machine. The data flow of these pre-computed values in between different processes or process to memory or I/O units is controlled with respect to each of the FSM state. The FSM controller implements FSM with a control unit which generates control signals to activate the different operations in specific states. These modules are implemented on the FPGA using components such as multipliers, adders, incrementers and multiplexers.

Fig-5 shows detailed functional diagram of LMS module. This shows the transfer of data among various storage units (registers) along with various operations done in the system. W represents adaptive filter coefficient buffer and X represents input samples buffer. Data paths are shown as block arrows, and the control signals are shown as lines. + represents addition and * represents multiplication. Fixed and floating point implementation differs only on the data path and the data operation modules. The address generation for input and adaptive filter buffers is the same. The address generation for input and weight buffer, read and write operations are synchronized with the control signals from the FSM states. The address generation modules either reset each of these addresses or do next address calculation with FSM state. The memory address updation is parallelly done for FIR and weight updation modules. The RTL schematic for address generation module is shown in Fig-6.

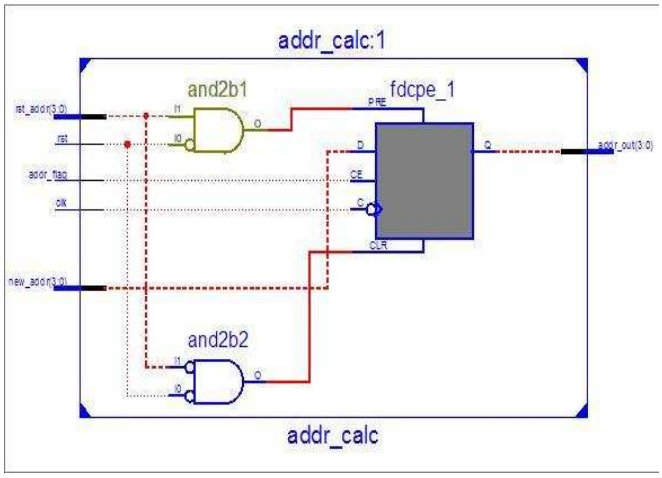


Fig6. Address generation module

3.3 Fixed Point Implementation

LMS algorithm is implemented in FPGA hardware using VHDL with fixed point packages. The usage and other details are available in ref [14]. The implementation uses signed fixed point (**sfixed**) data type with 16 integer bits and 16 bits for

fraction (as shown in Fig-2). The mathematical operations such as multiplication, addition, type conversion and resize (rounding of the results) are defined in the package. The adaptive filter buffer W and the input circular buffer X with a buffer size N are implemented using LUT's in FPGA.

The arrival of new input sample triggers the FSM from idle to *new_data* state. Data is converted to **sfixed** data type. The data is saved in the input circular buffer in *x_store* state and acknowledged. The *FIR* state initiates the MAC (multiply and accumulate) operation which calculates output in N clock cycles. Each MAC result is rounded off to the **sfixed** data type. For that, input sample x (n-k) and filter coefficient w (k) are read parallelly from corresponding buffers. FIR output is sent out at *data_out* state. The error calculation and mu*err multiplication are done in *new_err* state. The weight updation operation is implemented in three states. New weight value is calculated (with rounding off the result) in the *wt_calc* state, updated weight value is stored in the *wt_delay1* state and in the *wt_delay2* state, and next address is generated. Input and filter coefficients are read parallelly. Weight updation process requires 3*N clock cycles. So the LMS algorithm can be implemented in 9 individual states with 5+4N clock cycles.

3.4 Floating Point Implementation

3.4.1 IP Cores

IP cores are pre-made engineering blocks with a reusable unit of logic, cell, or chip layout design which can be integrated to create large System-On-a-Chip (SOC) designs. IP Core designs are available with specifications such as area, size, electrical characteristics, and even the silicon process. IP cores can be used as building blocks within ASIC chip designs or FPGA logic designs. Incorporating IP cores into design accelerates the development cycles in today's time-to-market challenges. The reuse of the verified existing blocks improves design efficiency and also removes the need to go for a new design.

The LMS algorithm is implemented using Floating point IP Cores [15] for math-oriented operations and Block RAM (Random Access Memory) Memory Generator Cores for storage of coefficients [16]. Data is represented with IEEE 754 floating point format throughout the implementation. DSP48 slices are extensively used for all math oriented operations. The cores are configured to get response within a single clock cycle and are included in the Data path of the FSM. The FSM (in Fig-4) is re-designed to generate control signals to reset, clear, and enable add, multiply and divide operations of the floating point cores and also to generate enable and write enable signals for the Block RAM modules. The multiplication IP Core is set with maximum usage i.e. 5 DSP48 slices and addition/subtraction core is set with Full usage (4 DSP48 slices).

The incoming input sample is converted to IEEE 754 single precision (32-bit) floating point format before operation and the floating point output is converted to fixed point before sending out. *FIR* MAC Operation is implemented in two clock cycles (multiplication of input coefficient with the filter weight is initiated in *FIR* state and in *fir_delay* state; the result is added with the previous sum). Another two states (*err_calc* and *mu_err_mult*), are included in the state machine multiplying mu with err (µ*e(n)). Weight updation is carried out in three clock cycles, (µ*e(n)) is multiplied with x(n) in *wt_calc* state, multiplied result is added with w(n) in *wt_delay1* state and the updated weight value is stored in weight buffer in the *wt_delay2* state . The Functional Diagram is same as in the Fig-5 but the math operations are done using floating point cores and memory is replaced with RAM Blocks available in the FPGA.

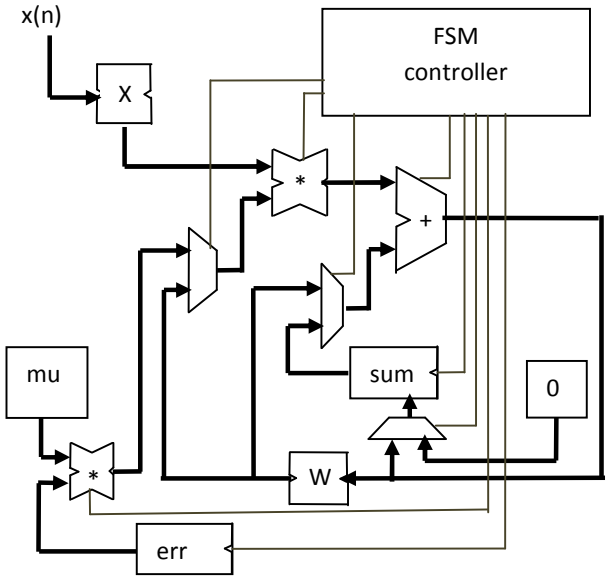


Fig7. Floating point IP core with resource sharing

The VHDL design is optimized by reusing the same floating point cores in FIR and weight updation operation (for multiplication and addition). This is achieved by the structural model implementation. The input to multiplication and addition IP cores are not connected from the registers/memory directly, instead, selected between two input signals using a mux on the basis of the state machine control signal. For example, the floating point multiplier inputs are configured as x(n) and w(n) for FIR operation, but x(n) and e(n) for weight updation operation. Similarly, the floating point adder inputs are configured as previous sum (sum in Fig-5) and current multiplier output (i.e, x(n)*w(n) product) for implementing FIR, but w(n) and the multiplier output (i.e, the product of x(n) and e(n)), for weight updation operation. The output is also redirected accordingly. This is clearly shown in Fig-7. By

this method, a significant reduction in number of DSP48 slices is obtained. FIR requires 2*N and weight updation requires 3*N clock cycles and the algorithm implemented totally requires 7+5*N clock cycles.

4. RESULTS AND DISCUSSIONS

4.1 Simulation Using ISIM

LMS adaptive filter has been implemented in VHDL using Xilinx IP cores using the Xilinx ISE (Integrated Software Environment) tool 12.4 Version and it has been simulated on Xilinx ISIM (ISE Simulator). Timing corrections and sequencing of modules w.r.t. FSM states has been verified by simulating with ISIM. The convergence characteristics with different mu value can be studied with ISIM results based on the mean square error.

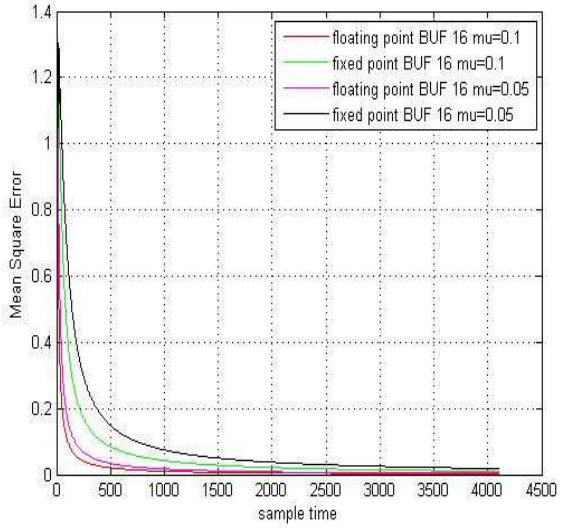


Fig-8 Convergence characteristics with different Mu value (values are taken to MATLAB and plotted). Here BUF is the adaptive filter Buffer Size.

Fig-8 shows the convergence rate of fixed point and floating point implementation with buffer size16 for mu values 0.1 and 0.05. The floating point representation enables greater accuracy, faster convergence compared to fixed point. Moreover, the minimum mean square error with floating point is more precise (up to 0.001), compared to fixed point (up to 0.01). That is, floating point implementation can minimize the residual error which appears with the fixed point implementation (which is not desired in active vibration control systems) by 90%.

Chart -1: Profiling result for Fixed and Floating point with a Buffer Size 16. Here FXP and FLP represent fixed point and floating point respectively

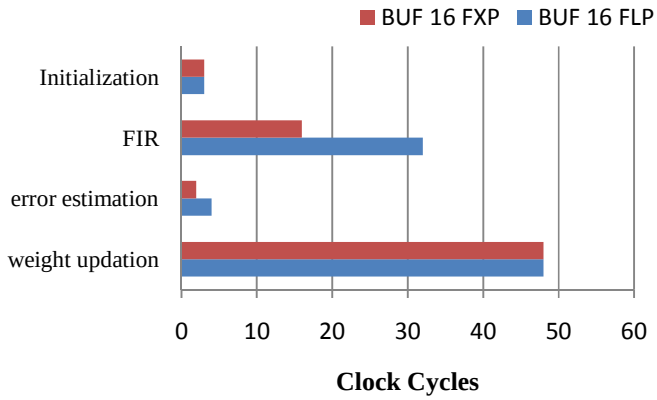
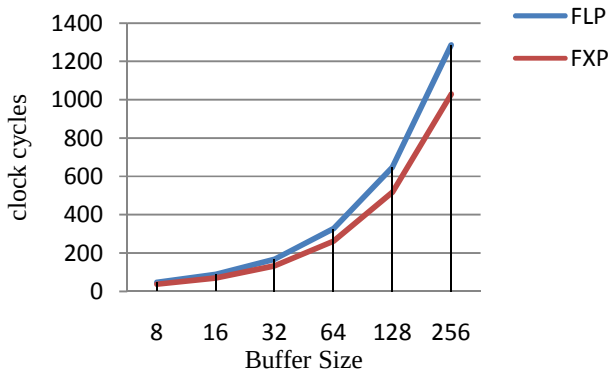


Chart -1 show the no. of clock cycles used for each individual process in LMS for a Buffer size 16. The performance speed characteristic with respect to the buffer size is shown in chart-2.

Chart -2: Speed of performance with respect to Buffer size. Here FXP and FLP represent fixed point and floating point respectively



From these two diagrams we can infer that floating point implementation of LMS algorithm is almost matching in speed with the fixed point implementation.

4.2 Implementing on Xilinx Virtex-4 FPGA Board

The LMS algorithm has been tested on Virtex-4 (XC4VSX55) FPGA board. The board is interfaced with AD5553 14-bit DAC (Digital to Analog Converter) having an output analog voltage range of +/-10V and AD7951 14-bit Successive Approximation register architecture based ADC (Analog to Digital Converter) with input voltage range of +/10V. The ADC/DAC sampling frequency is set up to 600 KHz. An input

sine wave from a function generator is fed to the FPGA board through ADC and the same is taken as the desired signal. Once the filter output is converged to the desired signal, the error minimizes to zero.

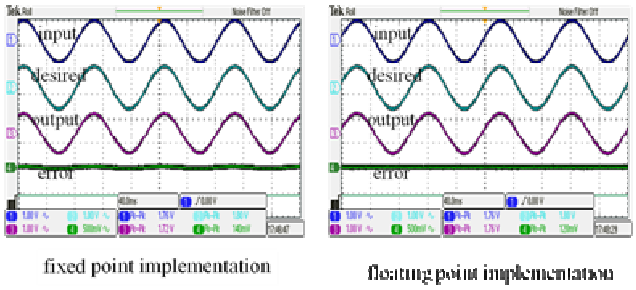
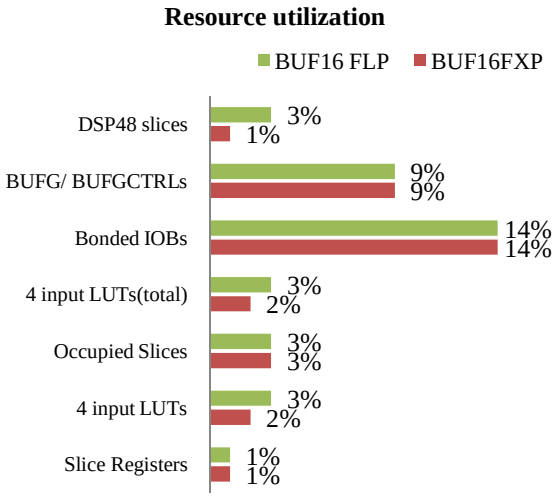


Fig-9 LMS algorithm on Virtex-4 FPGA Board

Fig-9 shows the oscilloscope waveforms and the residual error in fixed point implementation is higher compared to that of floating point implementation.

The percentage of resource utilization of fixed point and floating point implementations (including the ADC/DAC modules) with respect to total available in the Virtex-4 FPGA board is shown as bar chart below (Chart-3). Performance speed and the throughput can be improved by increasing the level of parallelism and pipelining but with the cost of more number of of resources. However this method is chosen by considering the fact that there should be enough resources available to extend this work as a single or multi channel active vibration control system.

Chart -3: Percentage of resource utilization in fixed and floating point implementation for a filter tap length 16. Here FXP and FLP represent fixed point and floating point respectively.



The Virtex-4 FPGA board has a clock frequency of 40 MHz, which is also the clock source to the ADC/DAC. The ADC/DAC processing time is set to 64 clock cycles, that is, each sample processing can take a time up to 1.6 micro seconds. Now the FSM implementation with fixed point takes 5+4N clock cycles as sample processing time and with floating point IP cores it takes 7+5N clock cycles. Calculations show that a direct implementation with a filter tap length up to 8 is possible in both of these cases. If the filter buffer size is more than 8, the state machine based design will automatically up sample the input signal and down sample the output signal in order to meet the timing.(The tap weights are to be set as the powers of 2, i.e. 2^n, n=1,2,…).

The floating point implementation uses 18 DSP48 slices (5x2=10 for multiplication, 4 each for addition and subtraction, fir and weight updation uses the same cores) whereas fixed point implementation uses only 10 slices. Both the designs use 3 to 4 BUFG's. No. of LUT's used as Route-thru in fixed point implementation depends on Buffer size since Memory Buffers are implemented through LUT's. The average fan-out (which is a measurement of power utilization in FPGA) in case of floating point implementation is almost constant but in fixed point design, it is comparatively more, and also varies with Buffer Size.

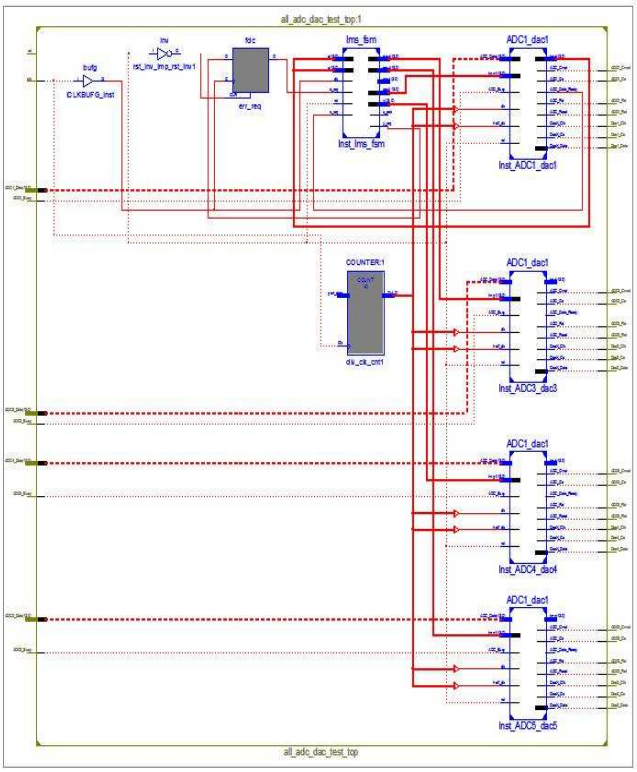


Fig10. Top level RTL schematic of LMS filter implementation

The RTL schematic of fixed and floating point implementations is shown. Fig-10 shows the top level RTL schematic, which is same for both fixed and floating point implementations, Fig-11shows the detailed view of fixed point implementation. It shows the detailed implementation view in terms of slices and LUTs. Here input and coefficient buffer are shown in the left side. Fig-12 shows that of Floating point. It can be clearly depicted that IP cores are appearing as black boxes.

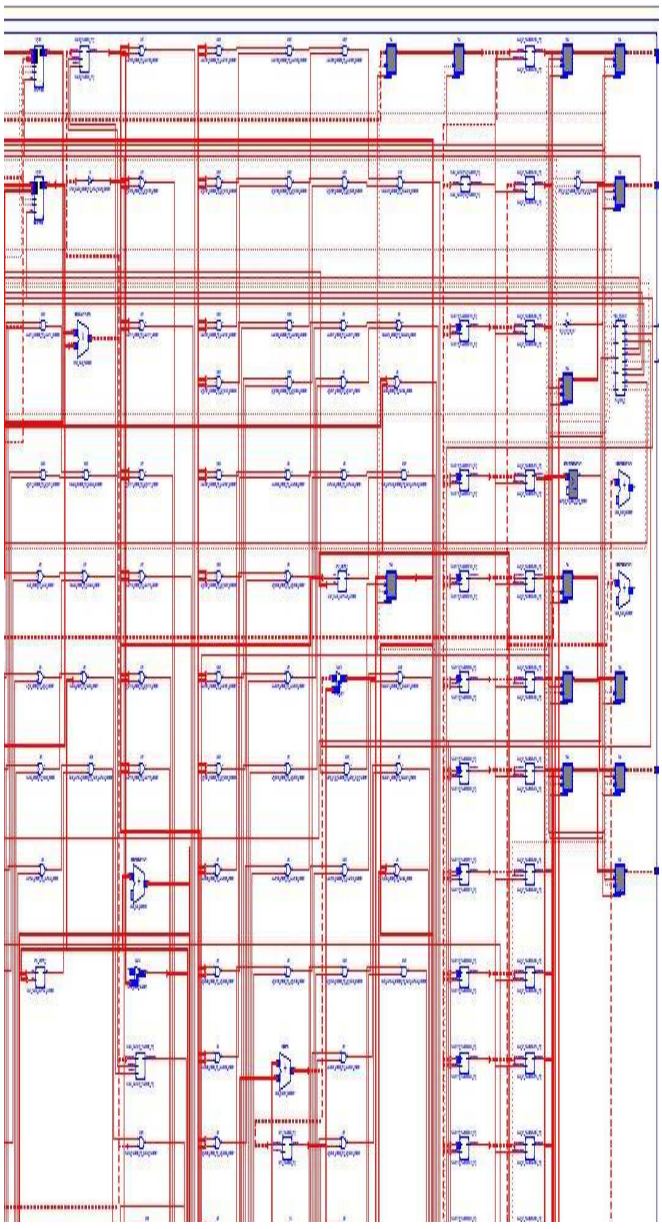


Fig11. RTL schematic of fixed point LMS filter implementation

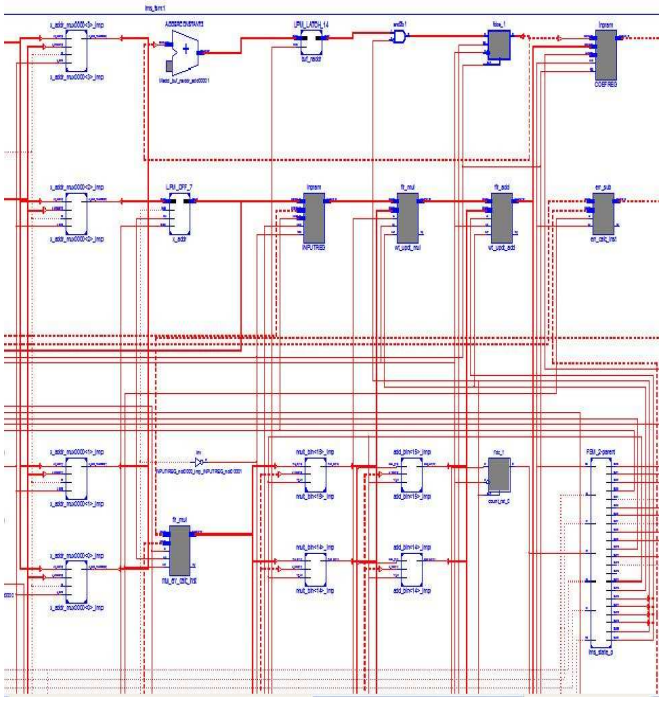


Fig12.RTL schematic of floating point LMS filter implementation

IP Core based design can drastically reduce the design time since the cores are optimized design units and in a ready to use state. Implementation using IP cores is more flexible in controlling the operation of each of the cores with enable or clear signals in the core and also output is indicated with the status signals. A better timing strategy can be achieved by using these features. This method also makes sharing of the available resources much easier. IP cores offer better predictability of chip performance in terms of timing, power, and area in the design stage itself. This helps the designers to manage the level of flexibility with which the design can fit into the FPGA architecture.

CONCLUSIONS

In this paper, LMS adaptive filter has been implemented on Xilinx Virtex-4 FPGA with fixed and floating point data representations using with Xilinx ISE V12.4. The convergence characteristics are studied. Floating point implementation results in a wide dynamic range, better level of accuracy, and also, the overflows are managed within the bounds of the hardware.

It is observed that the residual error with floating point implementation is almost negligible. Also floating point implementation facilitates optimization using IP Cores. This leads to better resource reutilization almost on par with fixed point implementation. Also it takes marginally more number

of clock cycles, but with improved performance. Floating point IP Core also supports division operation which can be synthesized on the FPGA fabric. This facilitates Variable mu calculation as the next step. So floating point implementation is preferred for further development such as adaptive system Identification and control.

ACKNOWLEDGEMENTS

The authors would like to thank CSIR-NAL for supporting the work. Also thanks are due to Director, NAL and Head, STTD for giving us the opportunity to carry out the work at NAL.

REFERENCES

[1] B. Widrow, S.D.Stearns, Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs, N.J., 1985.

[2] K. J. Astrum, B. Witenmark, Adaptive Control, Addison-Wesley Publishing Company, 1995.

[3] Sen M.Kuo, Dennis R.Morgan, Active Noise Control Systems – Algorithms and DSP Implementations, John Wileyand Sons, inc 1996.

[4] John R. Treichler, C. Richard Johnson, Michael G. Larimore, Theory and Design of Adaptive Filters, John Wiley and Sons 1987

[5] S. Haykin, Adaptive Filter Theory, Fourth Edition, Prentice Hall, Upper Saddle River, N.J., 2002

[6] Uwe Meyer-Baese, Digital Signal Processing with Field Programmable Gate Arrays. Springer-Verlag, 2nd edition 2004

[7] Xilinx inc., MicroBlaze Processor Reference Guide Embedded Development Kit for EDK 14.1, April 2012

[8] M. Bahoura , H.Ezzaidi, FPGA-Implementation of a Sequential Adaptive Noise Canceller Using Xilinx System Generator, Inter. Conf. On Microelectronics - Icm, Marrakech, (2009), 213-216

[9] Zheng-Weihu, Zhi-Yuanxie (2009), Modification of Theoretical Fixed-Point LMS Algorithm for Implementation in Hardware, Commerce And Security, 2009. ISECS '09. Second International Symposium On Volume: 2 Digital Object Identifier: 10.1 109/Isecs.2009.40, Page(S): 174 – 178

[10] A. Elhossini, S. Areibi, R. Dony, An FPGA Implementation of the LMS Adaptive Filter for Audio Processing, Reconfig, Pp.1-8, 2006 IEEE International Conference on Reconfigurable Computing And FPGA's (Reconfig 2006), 2006

[11] Wolfgang Fohl,,Jörn Matthies, Bernd Schwarz, A FPGA-Based Adaptive Noise Cancelling System, Proc. of the 12th Int. Conference On Digital Audio Effects (Dafx-09), Como, Italy, September 1-4, 2009

[12] ANSI/IEEE, IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.IEEE-754.

[13] Spoorthi Shekar, Shashikala Prakash, C Gurudasnayak, Renjith Kumar, Ravikiran P.G. Implementation of LMS Adaptive Filter on Virtex-4 FPGA Platform in VHDL., The Fourth National Conference on Information Sciences (NCIS-2012) April 27-28, 2012 Manipal Centre for Information Science (MCIS), Manipal, Karnataka, India

[14] Fixed point package user's guide, David Bishop www.vhdl.org/fphdl/Fixed_ug.pdf

[15] Xilinx LogiCORE IP Floating-Point Operatorv5.0 Data sheet, ds335.pdf www.xilinx.com/

[16] Xilinx LogiCORE IP Block Memory Generator v4.3 data sheet, ds512.pdf www.xilinx.com/4

BIOGRAPHIES

Shashikala Prakash was born in Bangalore, Karnataka, India, in 1956. She received the B.E. & M. E. Degrees in Electronics Engineering from University Visveswaraya College of Engineering, Bangalore in 1978 & 1998 respectively.

From 1980 to 2006, she has been with National aerospace Laboratories in various capacities starting as Junior Scientific Assistant. Since 2006, she has been a Senior Principal scientist handling many projects inactive control involving Microprocessors, DSPs & FPGAs using Smart materials. She has 49 internal publications, 12 National conference papers & 12 International Conference papers including 34 journal publications. She has presented papers in International conferences held at Italy (1989), Sweden (2006) & China (2009). Her research interests include Vibration and its control using smart materials, Real Time Active Vibration Control using Adaptive filters on DSP, FPGA, Microcontrollers, Wireless sensor Networks etc.

Ms. Shashikala Prakash has also contributed in Ground Vibration Testing of full scale aircrafts, Scaled models and also Vibration testing & analysis of various aircraft/aerospace structures. She was the recipient of the CSIR NAL **BEST WOMAN SCIENTIST AWARD** for the year 2002. She is professional member of AeSI, ISAMPE & life member of ISSS.

Renjith Kumar T.G. was born in Alappuzha, Kerala, India, in 1979. He received his M.Sc. & M.Tech. Degrees in Electronics from Cochin University of Science and Technology, Kochi in 2003 and 2010 respectively

From 2011 onwards, He has been working as a Scientist Fellow in National Aerospace Laboratories Bangalore. He has one International Conference paper and two National level conference presentations. His research areas include Real Time Active Vibration Control, Adaptive filters, Digital Signal Processing, FPGA, Robotics, Neural Networks, Microcontrollers and Embedded systems.

Subramani H was born in Kunigal, Karnataka, India 1987. He obtained B.E. in Electronics and Communication Engineering from University Visveswaraya College of Engg, Gubbi in 2010.

Subramani Worked as a Project Engineer during 2012 - 13 in National Aerospace Laboratories, Bangalore before joining as a Hardware Engineer in Dexcel Electronics Designs PVT Ltd in Bengaluru. His areas of research interests include Real Time Active Vibration Control, Adaptive filters, Digital Signal Processing, FPGA, Microcontrollers and Embedded systems.