FEASIBILITY OF FIXED-POINT TRANSVERSAL ADAPTIVE FILTERS IN FPGA DEVICES WITH EMBEDDED DSP BLOCKS

Andrew Y. Lin, Karl S. Gugel and José C. Príncipe

Computational NeuroEngineering Laboratory, Dept. of ECE, University of Florida
Emails: alin@ufl.edu, gugel@ecel.ufl.edu, principe@cnel.ufl.edu

ABSTRACT

Transversal adaptive filters for digital signal processing have traditionally been implemented into DSP processors due to their ability to perform fast floating-point arithmetic. However, with its growing die size as well as incorporating the embedded DSP block, the FPGA devices have become a serious contender in the signal processing market. Although it is not yet feasible to use floating-point arithmetic in modern FPGAs, it is sufficient to use fixed-point arithmetic and still achieve tap-weight convergence for adaptive filters. This paper examines the feasibility of implementing an adaptive algorithm, namely the LMS algorithm, based on fixed-point arithmetic, using the Altera Stratix device.

1. INTRODUCTION

Transversal filters have fixed weights and the output of the filters is the convolution of the taps and the filter coefficients. Transversal adaptive filters need an appropriate algorithm to update the filter coefficients and are widely used in the communication industry, as well as in applications such as echo noise cancellation, adaptive beamforming, and channel equalization [4]. An important issue in transversal adaptive filters is the lack of clear methodology to determine the topology before training starts. It is then desirable to speed up the training and allow fast experimentation with various topologies [5].

Transversal adaptive filters have traditionally been implemented using DSP processors. The DSP processors hold the advantage of being able to perform fast floating-point arithmetic, which is essential to fast convergence of adaptive algorithms such as the Least Mean Square (LMS) algorithm. However, the DSP processors lack the flexibility for adjusting to different adaptation requirements from various topologies. Also, the fixed-structure feature of the DSP processors restricts that DSP processors can only perform one arithmetic operation at a time. Such feature restricts the DSP processors' performance.

On the other hand, FPGAs are a form of programmable logic, which offer flexibility to be reconfigured repetitively. Also, since FPGAs consist of Embedded Array Blocks (EABs) organized in rows and columns, a great deal of parallelism can be explored. In transversal adaptive filters, each tap, as well as component for updating each filter coefficient, requires a multiplier and an adder. By instantiating the required numbers of multipliers and adders, the performance of FPGA based filter can increase significantly compared to DSP processors. Additionally, the new generation of FPGAs, the Stratix device family from Altera for example, offers embedded DSP blocks within the device. The embedded DSP blocks have dedicated circuitry to perform additions and multiplications, which are fundamental for any DSP applications.

Without performing arithmetic in floating-points, with sufficient bit length to represent tap-weights, one can achieve convergence in adaptive algorithms by using FPGAs [2]. However, *stalling*, also known as *lockup*, may arise in fixed-point adaptation process. This phenomenon can be avoided by carefully studying the nature of the experiment and choosing bit length accordingly for the filter coefficients [4].

This paper explores the LMS based transversal adaptive filters implemented in the Altera EP1S25F780 device. In section 2, an overview of the Altera's Stratix device family is given. Section 3 outlines the LMS algorithm and its procedures. Detailed hardware implementation of such filter is presented in Section 4. Issues such as bit length selection criterion, fixed point vs. floating-point, and design performance are also discussed.

2. STRATIX DEVICES

The Stratix family is the newest family of programmable logic devices from Altera. The Stratix devices have three times the size of memory blocks compared to traditional FPGAs. The Stratix devices also contain embedded DSP blocks, which have dedicated multiplier, pipeline and accumulator circuitries. With the embedded DSP blocks, the Stratix devices can perform high speed calculation.

Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

2.1. Stratix architecture

Stratix devices contain a two-dimensional row and column based architecture to implement custom logic. A network of varying length and speed, row and column interconnects provide signal interconnections between Logic Array Blocks (LABs), memory blocks, and embedded DSP blocks.

Each LAB consists of 10 Logic Elements (LEs). LABs are grouped into rows and columns across the device.

The memory blocks are RAM based. These memory blocks provide dedicated simple dual-port or single port memory up to 36 bits wide and up to 291MHz access speed.

The DSP blocks can implement multiplications in various bit length with add or subtract features. The blocks also contain 18-bit input shift registers for applications such as Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) filters.

Figure 1 shows the block diagram of a typical Stratix device [1].

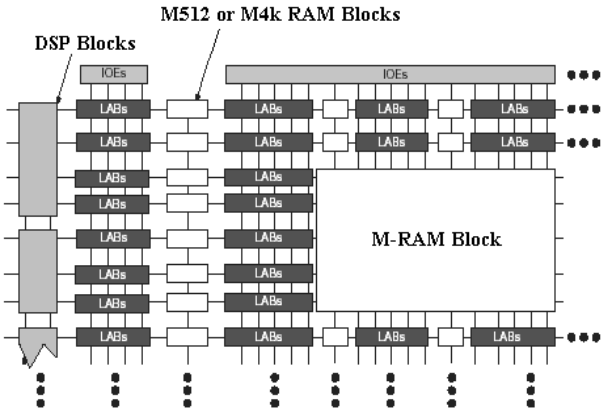

Figure 1. Stratix Device Block Diagram

2.2. Embedded DSP Blocks

The most commonly used DSP functions include multiplication, addition, and accumulation. The Stratix devices provide DSP blocks to meet the arithmetic requirements of these functions. Each Stratix device has two columns of DSP blocks to efficiently implement DSP functions faster than LE-based implementations.

Each DSP block can be configured to support:

- Eight 9 x 9 bit multipliers
- Four 18 x 18 bit multipliers
- One 36 x 36 bit multiplier

DSP block multipliers can optionally feed an adder/subtractor or accumulator within the block. This feature saves LE routing resources and increase performance, since all inter-connections and blocks are all within the DSP block. The DSP block input registers can also be configured as shift registers for FIR filter applications.

Figure 2 is a block diagram for a typical component inside the DSP block.

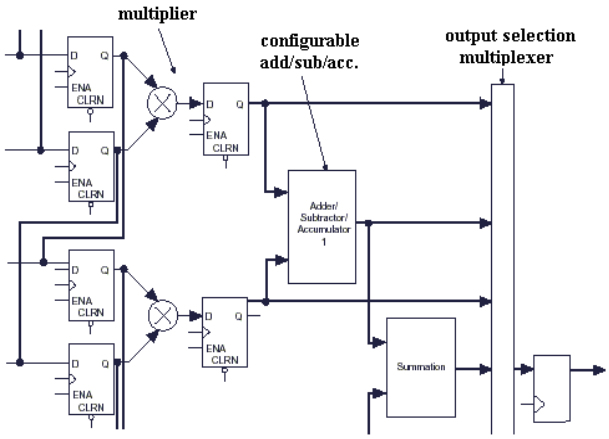

Figure 2. DSP Block Diagram

3. THE LMS ALGORITHM

The LMS algorithm is a Tap Delay Line (TDL) structure and uses Mean Square Error (MSE) as criterion. LMS uses a small step-size parameter, μ, input signal, along with the difference of desired signal and filter output signal to periodically calculate the update of the filter coefficients set.

3.1. LMS Equation

Each filter coefficient adaptation uses its present coefficient value, w[n], to add to the product of the step-size parameter, μ, tap input x[n] and error output e[n], to obtain an updated version of the filter coefficient w[n+1]. All updated filter coefficients are then gathered to perform convolution with the taps to produce a filter output. The filter output y[n] is subtracted from the desired value d[n] to produce an error term, e[n], which is fed back into the filter coefficient update equation to produce consequent coefficient updates. The equations are shown below:

$$e[n] = d[n] - y[n] \quad (1)$$

$$w[n+1] = w[n] + \mu * x[n] * e[n] \quad (2)$$

Note that the only information needed to update filter coefficients are the tap input and the error term. The step-size parameter is pre-determined by the system engineer and is a decimal number between 0 and 1.

3.2. Selecting Step-size and Filter Order

The choice of the step-size parameter and the order of the filter effectively determine the performance of LMS solution. Unfortunately, there is no clear mathematical

Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

analysis to derive the quantities. Only through experiments may we obtain a feasible solution [4]. Nevertheless, the step size parameter, μ, is bounded at the range of $0<\mu<(2/\lambda_{max})$, where λ_{max} is the maximum eigenvalue of the auto-correlation matrix of the filter input [4].

4. HARDWARE IMPLEMENTATION

4.1. Structural Overview

The LMS algorithm uses FIR filter structure. The design shown in Figure 3 depicts a structural view of such FIR filter. As shown in the figure, the main components of the filter consist of *m* Unit Delay Registers and *m+1* Weight Updates. The Unit Delay Registers are simply D Flip-Flops. Each Weight Update component updates the filter coefficient according to equation (2). The filter output is subtracted from the desired signal to produce an error signal. The error signal is a buffer, which is fed back to the Weight Update components to produce next sets of filter coefficients.

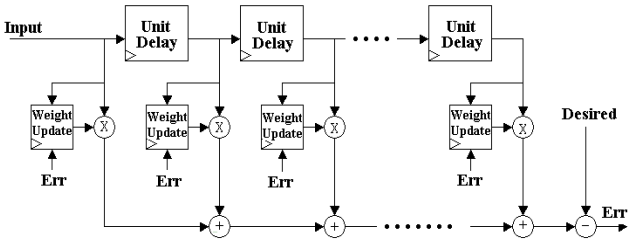

Figure 3. FIR Filter using LMS Algorithm

4.2. Arithmetic Operations

Weight Updates perform logics according to equation (2). The operations needed in equation (2) include two multiplications and one subtraction. Note that the step-size parameter μ is a decimal number, and that multiplying a decimal number is equivalent of dividing its reciprocal. However, in order to avoid implementing complicated and area-consuming division circuitry, or multiplication for floating-point numbers, Arithmetic Shift Right (ASR) operation is used instead to simplify and boost the run-time frequency of the design.

The ASR operation on a 2's complement integer shifts the number n bits to the right (direction of the least significant bit), while preserving the sign bit (the most significant bit). By shifting the number *n* bits to the right, it is equivalent of multiplying this number by 2^{-n}. Therefore, in order to achieve simplicity and feasibility, this design restricts the value of μ to be $\mu = 2^{-n}$, where *n* is a positive integer.

Since all arithmetic operations in this design are done in fix-point, tradeoffs relate to precision exist which must be dealt with delicately. The problem is twofold: first, how to balance between the need of reasonable numeric precision, which is important for accuracy and convergence, and the cost of area in terms of LEs occupied. Second, how to choose a suitable bit length whose dynamic range is large enough to guarantee that saturation, as well as the stalling phenomenon will not occur for a particular application [2].

In this particular design, for simplicity and clarity, the bit lengths for all intermediate results are truncated to be equal to the bit lengths of the operands, by removing the lower significant bits but preserving the sign bit. Result from each operation therefore loses precision however still achieves convergence.

5. RESULTS

Each individual component from figure 3 is created using behavioral architecture of the VHDL. A package VHDL file is also created to hold constants such as number of taps and bit length for the system inputs. By using port map statements in VHDL, in association with the package file, the author is able to dynamically instantiate the fixed-point transversal adaptive filter with various numbers of taps. The bit length of the system is chosen to be 16 bits.

The design is simulated in Altera's Quartus II software. The device chosen is the Stratix EP1S25F780 device, which contains 25,660 LEs, 10 DSP blocks, and has maximum clock rate of 250MHz.

Area and speed are the two main measurements in evaluating system performance of this filter. Figures 4 and 5 show the varying filter orders vs. area and speed plots, respectively. Area is measured by number of LEs occupied. Speed is measured by the maximum allowable clock frequency.

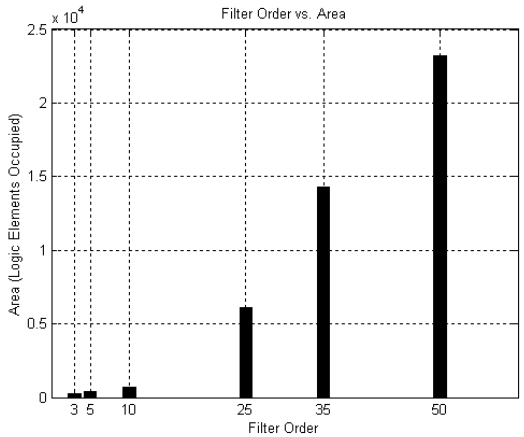

Figure 4. Filter Order vs. Area Plot

Proceedings of The 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications ISBN 0-7695-1929-6/03 $17.00 © 2003 IEEE

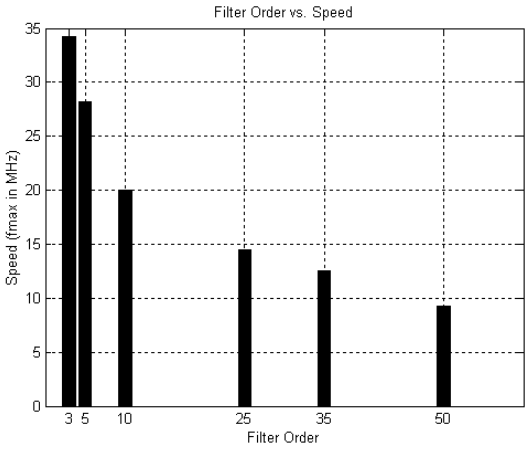

Figure 5. Filter Order vs. Speed Plot

6. DISCUSSION

Refer to Figures 4 and 5, as filter order increases, the performance of the design is reduced considerably. For each additional tap, a separate weight update, multiplier, and adder also have to be instantiated. Therefore, when the number of taps reaches 50, all available LEs are all but occupied. Similarly, for each additional tap, the longest register-to-register path is elongated as well, resulting allowable frequency to plunge.

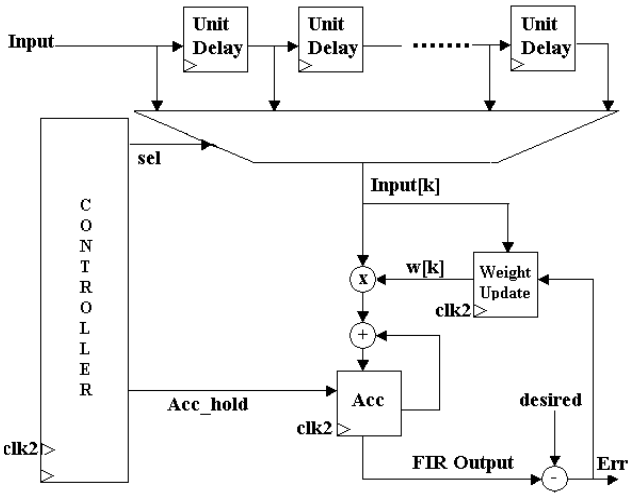

Figure 6. Proposed Structure

Although the design depicted in Figure 3 fully utilize the parallelism advantage of FPGA devices, it becomes impractical for higher order FIR filters, in which instances of multiplier and adders become substantial. For an adaptive filter with order *L*, *2L* multipliers and adders are required. One way to cope with such setback is the use of parallel-serial multipliers [3]. Alternatively, since each tap requires the exact operations, namely multiplication and addition, the author proposes a more sophisticated design. In this design, a single Multiply and Accumulate (MAC) component is used and shared by all taps. The input to the MAC is controlled by a multiplexer. Figure 6 shows the block diagram of this design.

The proposed structure requires two clock signals. One clock signal corresponds to the input data and desired data frequency. The other clock signal, *clk2*, is used in both the controller and the weight update component. The controller, essentially a counter/state machine, controls which tap serves as the input to the MAC. The controller also directs the accumulator to output the final accumulation after all taps have been served as inputs to the MAC. Clearly, clock signal *clk2* has to run fast enough so that accumulation is done prior to the next input cycle, but slow enough so that weight update and the MAC has enough time to propagate their logics.

7. CONCLUSION

The implementation of a fixed-point based transversal adaptive filter in a Stratix device has been presented. The implementation is based on the design shown in Figure 3. Performance is measured in terms of speed (maximum allowable clock frequency) and area (number of LEs occupied). The results indicate that it is feasible to implement fixed-point based adaptive filters in FPGA devices. The system engineer, however, must study the tradeoffs between precision of the data and filter coefficients, speed of convergence, speed of the system, and available LEs, based on the nature of the data and the nature of the topology.

8. REFERENCES

[1]Altera, *Stratix Programmable Logic Device Family Data Sheet,* Data Sheet DS-STXFAMLY-2.1, Altera, Inc., August, 2002

[2]Chew, W.C., Farhang-Boroujeny, B., *FPGA Implementation of Acoustic Echo Cancelling.* TENCON 1999. Proceedings of the IEEE Region 10 Conference, Volume: 1, pp. 263—266.

[3]FU, R. FORTIER, P, *VLSI Implementation of Parallel-Serial LMS Adaptive Filters*, 18th Biennial Symposium on Communications, Kingston, June 1996, pp. 159—162.

[4]Haykin, S. *Adaptive Filter Theory*, 4th edition, Prentice Hall, New Jersey, 2002.

[5]Nichols, K.R., Moussa, M.A., Areibi, S.M., *Feasibility of Floating-point Arithmetic in FPGA based Artificial Neural Networks*, submitted and accepted to 15th International Conference on Computer Applications in Industry and Engineering, November 2002.