Overview of an Adaptive Filter Using Distributed Arithmetic With Offset Binary Coding

R.Swathi, S.Janani Sri, S.Deebika

Assistant Professor, Department of Instrumentation and Control Engineering, Sri Krishna College of Technology, TamilNadu, India.
Assistant Professor, Department of Instrumentation and Control Engineering, Sri Krishna College of Technology, TamilNadu, India.
Assistant Professor, Department of Instrumentation and Control Engineering, Sri Krishna College of Technology, TamilNadu, India.

---***---

Abstract - *Adaptive filters are used in many DSP (Digital Signal Processing) applications. High-throughput, low-area and low-power are the key aspects in VLSI design. Distributed Arithmetic (DA) which is used in the existing design results in high memory requirements due to high size memory based Look Up Table (LUT). So in this paper, Distributed Arithmetic (DA) with Offset Binary Coding (OBC) technique is used to reduce the memory based LUT size into half. An architecture which is pipelined, considered here to get high-throughput, low-area and low-power. LUT updating and the implementation of filtering and weight updating will results in high-throughput. Carry save accumulation is used to reduce the complexity of the area. In order to reduce the power consumption fast bit clock is used in the carry save accumulation. DA techniques with OBC consists of the Same number of adders, multiplexors, smaller size LUT are required when compared to the existing design.*

Key Words: *Adaptive filter, Distributed Arithmetic (DA), Offset Binary Coding (OBC), Least mean square (LMS) algorithm.*

1. INTRODUCTION

Digital signal processing (DSP) is the mathematical manipulation of an information signal to modify or improve it in some way. Adaptive filters are widely used in several digital signal processing (DSP) applications. An adaptive filter is a computational device that attempts to model the relationship between two signals in real time in an iterative manner. An adaptive filter is a system with a linear filter that has a transfer function controlled by variable parameters and a means to adjust those parameters according to an optimization algorithm. An adaptive filters whose weights are updated by the famous Widrow-Hoff least mean square (LMS) algorithm is the most popularly used only due to its simplicity but also due to its satisfactory convergence performance. Adaptive filters are used in echo cancellation, inverse modelling, signal de-noising, channel equalization for communication and networking. Multiplierless Distributed Arithmetic is used to compute the inner product between the fixed and variable vector. But DA requires more memory since the LUT size is more. So, DA with OBC is used to overcome the high memory requirement by reducing the LUT size using the factor 2^N to 2^{N-1}.

2. LEAST MEAN SQUARE ALGORITHM

Least mean squares (LMS) algorithms are used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time. LMS algorithm is used to computes the output of the filter and an error signal value. Error signal value is the difference between the current output of the filter y(n) and the desired output of the filter d(n). The weights W(n) of LMS adaptive filters are updated according to the following equations,

$$W(n+1) = W(n) + \mu\, e(n)\, x(n) \qquad (1)$$

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

$$y(n) = W^T(n)\, x(n) \qquad (3)$$

3. OFFSET BINARY CODING

Offset binary coding is often used in adaptive filters of the digital signal processing (DSP). Offset binary is also referred to as excess-K, which is a digital coding method, where all-zero corresponds to the minimal negative value and all-one to the maximal positive value. Offset binary coding (OBC) is used with Distributed Arithmetic (DA) which is used to reduce LUT size into half. There is no standard for offset binary, but most often the offset K for an n-bit binary word is $K=2^{(n-1)}$. This has the consequence that the "zero" value is represented by a 1 in the most significant bit and zero in all other bits, and in general the effect is conveniently the same as using two's complement except that the most significant bit is inverted. It also has the consequence that in a logical comparison operation, one gets the same result as with a two's complement numerical comparison operation, whereas, in two's complement notation a logical comparison will agree with two's complement numerical comparison operation if and only if the numbers being compared have the same sign. Otherwise the sense of the comparison will be inverted, with all negative values being taken as being larger than all positive values.

Table-1: OBC LUT contents with 2^{N-1} words

X_{0j}	X_{1j}	X_{2j}	X_{3j}	LUT CONTENTS
0	0	0	0	$-(h_0+h_1+h_2+h_3)/2$
0	0	0	1	$-(h_0+h_1+h_2-h_3)/2$
0	0	1	0	$-(h_0+h_1-h_2+h_3)/2$
0	0	1	1	$-(h_0+h_1-h_2-h_3)/2$
0	1	0	0	$-(h_0-h_1+h_2+h_3)/2$
0	1	0	1	$-(h_0-h_1+h_2-h_3)/2$
0	1	1	0	$-(h_0-h_1-h_2+h_3)/2$
0	1	1	1	$-(h_0-h_1-h_2-h_3)/2$
1	0	0	0	$(h_0-h_1-h_2-h_3)/2$
1	0	0	1	$(h_0-h_1-h_2+h_3)/2$
1	0	1	0	$(h_0-h_1+h_2-h_3)/2$
1	0	1	1	$(h_0-h_1+h_2+h_3)/2$
1	1	0	0	$(h_0+h_1-h_2-h_3)/2$
1	1	0	1	$(h_0+h_1-h_2+h_3)/2$
1	1	1	0	$(h_0+h_1+h_2-h_3)/2$
1	1	1	1	$(h_0+h_1+h_2+h_3)/2$

Table 1 shows the reduced ROM based LUT size. The LUT size is reduced to half.

3.1. DERIVATION OF OFFSET BINARY CODING

Let the Inner product be represented by,

$$y = \sum_{i=0}^{N-1} w_i \cdot x_i \tag{1}$$

Two's complement representation,

$$x_i = x_{ko} + \sum_{j=1}^{M-1} x_{kj}\ 2^{-j} \tag{2}$$

$$x_i = \frac{1}{2}\left[x_i - (x_i)\right] \tag{3}$$

$$-x_i = -\overline{x_{ko}} + \sum_{j=1}^{M-1} \overline{x_{kj}}\ 2^{-j} + 2^{-(M-1)} \tag{4}$$

Substituting (2) and (4) in (3)

$$x_i = \frac{1}{2}\left[-\left(x_{ko} - \overline{x_{ko}}\right) + \sum_{j=1}^{M-1}\left(x_{kj} - \overline{x_{kj}}\right)2^{-j} - 2^{-(M-1)}\right]$$

consider d_{kj} as $\left(x_{ko} - \overline{x_{kj}}\right)$ equation (1) becomes,

$$y = \sum_{i=0}^{N} \frac{w_k}{2}\left[-d_{ko} + \sum_{j=1}^{N-1} d_{kj} 2^{-j} - 2^{-(N-1)}\right] \tag{5}$$

$$y = \sum_{k=0}^{N} \frac{w_k d_{ko}}{2} + \sum_{j=1}^{N-1}\left[\sum_{k=0}^{N} \frac{w_k d_{kj}}{2}\right]2^{-j} - \sum_{k=0}^{N} \frac{w_k}{2} 2^{-(N-1)} \tag{6}$$

equation (6) can be rewritten as,

$$y = -F_O + \sum_{j=1}^{N-1} F_j 2^{-j} + F_{N-1} 2^{-(N-1)} \tag{7}$$

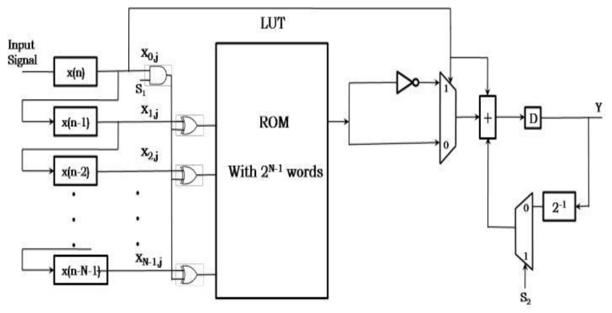

Figure-1 DA based OBC implementation

Figure1.shows the implementation of DA based OBC. Again computation starts from LSB of x_i, i.e., j=0. The XOR gates are used for address decoding, the MUX with constant D_0 provides the initial value to the shift – accumulator and the MUX after the ROM is used to inverse the output of ROM when j=k-1. Two control signal s_1 and s_2 are required, where s_1 is 1 when j = k-1 and 0 otherwise, and s_2 is 1 when j = 0 and 0 otherwise. Instead of this shift-accumulator carry-save accumulation is used to reduce the complexity. Fast bit clock is used in the carry save accumulation to reduce the power consumption.

4. LITERATURE SURVEY

LMS adaptive filters using distributed arithmetic for high throughput

In this paper, DA uses bit-serial operations and LUTs. Only one cycle per bit is used to get high throughput. MAC operation is very common in all Digital Signal Processing Algorithms. The "basic" Distributed Arithmetic technique is bit-serial in nature. DA is basically a bit-level rearrangement of the multiply and accumulate operation. Distributed Arithmetic hides the explicit multiplications by ROM look-ups an efficient technique to implement on Field Programmable Gate Arrays (FPGAs). Distributed Arithmetic is used for computing the inner products between a fixed and a variable data vector. In Distributed Arithmetic, the partial products are pre computed and stored in the memory based LUT. Conventional Distributed Arithmetic (DA) is popular in ASIC design and it features on-chip ROM to achieve high speed and regularity. LUT stores the filter coefficients and the partial products are given as its output. LUT should be recalculated for each adaptation in adaptive filters. LUT with special addressing is used. DA is a method often eliminates the use of multipliers. High throughput is obtained. Throughput improves the low power consumption by decreasing the speed of the clock. In this paper we can notice the importance of DA-based LMS adaptive filters (DAAF) which retains high throughput compared to MMAF (MAC based LMS adaptive filter). Also explains about the DA-LUT[n] updating from DA-LUT [n-1]. In this updating 0 in LSB of LUT [N] is mapped to 0 in MSB of [n-1]. Since DA is used memory requirements is more due the ROM based LUT.

Two high-performance adaptive filter implementation schemes using DA

In this paper DA is used for vector-vector multiplication with a direct application for implementation of convolution which is necessary in digital filters. Two proposed schemes are introduced to store the delayed input samples. The first scheme explains the how the DA stores the sum of weights (coefficients) in the LUTs and uses inputs as addresses. This scheme is well suited for non adaptive filters with constant coefficients. Memory requirements are more in this scheme.The second scheme is OBC (offset binary coding) scheme which are used to reduce the number of LUT entries to reduce the size of the LUT. The LUT size is reduced to its half. So the memory requirements are less in this scheme.

From this paper updating of the weights from t=n to t=n+1 is studied. OBC implementation from DA can also be known with delayed input samples. In this design, low latency and memory requirements improvement are obtained. No use of auxiliary LUT. In the first scheme, exactly the memory usage is half when compared to the previous paper. In the second scheme, memory usage is 30% less is required when compared to the previous paper. But the second proposed scheme requires more addition operations than the first scheme.

High throughput pipelined realization of adaptive FIR filters based on distributed arithmetic

In this paper, carry-save adder with an efficient architecture for high speed based on DA is used to improve the throughput ant to reduce the area-complexity. The proposed design does not require LUT instead half the registers are used to store the different input samples sum. LMS algorithm is used as a optimization algorithm to update weight in an adaptive filters. LUT stores the weight coefficients and the input samples which are given as address. Registers are used instead of D-flip flop. Area-complexity becomes less in this proposed design when compared to the previous paper. Sampling period becomes high in this design. This proposed design17% more hardware is required. High throughput and less energy are obtained. Limited registers are used.

Conjugate Distributed Arithmetic Adaptive FIR Filters and their hardware implementation

In this paper, a new hardware architecture using conjugate distributed arithmetic (CDA) which is suitable for high throughput hardware implementations of LMS adaptive filters is presented. Unlike a traditional distributed arithmetic (DA) implementation where all possible combination sums of the filter coefficients are stored in a look-up-table (LUT), in the CDA architecture, all possible combination sums of the input signal samples are stored in the LUT and updated at the arrival of every sample using an efficient update procedure. The design of CDA adaptive filters and that the practical implementations of CDA adaptive filters which have very high throughput relative to multiply and accumulate architectures were described. Also this design shows that CDA adaptive filters have a potential area and power consumption advantage over DSP microprocessor architectures for a given throughput. Many multiply and accumulate units can be used which results in increased complexity and chip area. Power consumption is also becomes high. But throughput is improved in this proposed design.

Low-power, high-throughput, low-area adaptive filter based on Distributed Arithmetic

In this paper, a novel pipelined architecture for low-power, high-throughput, and low-area implementation of adaptive filter based on distributed arithmetic (DA). An efficient technique for calculating the sum of products or vector dot product or inner product or multiply and accumulate (MAC). MAC operation is very common in all Digital Signal Processing Algorithms. The "basic" Distributed Arithmetic technique is bit-serial in nature. DA is basically a bit-level rearrangement of the multiply and accumulate operation. Throughput rate is increased by a parallel lookup table (LUT) update. Throughput is also achieved by concurrent implementation of filtering and weight-updating. Conventional adder-based shift-accumulation is replaced by a conditional carry-save accumulation of signed partial inner-products to reduce the sampling period. The bit-cycle period amounts to memory access time plus one bit full-adder time (instead of ripple carry addition time) by carry-save accumulation. The use of proposed signed carry-save accumulation also helps to reduce the area-complexity of proposed design. Reduction of power-consumption is achieved by using fast bit-clock for carry-save accumulation but much slower clock for all other operations. The existing designs require an auxiliary control unit for address generation, which is not required in the proposed structure. The inner product computation can be calculated in L cycles of shift-accumulation followed by LUT-read operations.

5. CONCLUSIONS

With the advancement in VLSI design for Digital Signal Processing (DSP) applications high-throughput, low-power, low-area are very important parameters. An efficient architecture is considered which is based on Distributed Arithmetic (DA) with Offset Binary Coding (OBC) to attain those parameters. OBC technique is used to reduce the ROM based LUT size. So the memory requirement is less since the LUT size is reduced by the factor 2^N to $2^{N-1.}$ The size is reduced to half, other half is obtained by changing the signs. Thus the less memory leads to small number of long inner products. Throughput rate is increased by LUT updates. Carry save accumulation is used instead of basic shift-accumulation in order to reduce the complexity. Unlike the existing design, the memory usage is 30% less. So the use of OBC leads to low computational cost. High-throughput, low power consumption, low area-complexity are obtained when compared to the earlier designs.

REFERENCES

[1] S.Haykin and B.Widrow, Least-mean-square adaptive filters. Wiley-Interscience, Hoboken, NJ, 2003.

[2] S. A. White, "Applications of the distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Magazine, vol. 6, no. 3, pp. 5–19, Jul. 1989.

[3] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson, "LMS adaptive filters using distributed arithmetic for high throughput," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 52, no. 7, pp. 1327–1337, Jul. 2005.

[4] R. Guo and L. S. DeBrunner, "Two high-performance adaptive filter implementation schemes using distributed arithmetic," IEEE Transactions on Circuits and Systems-II: Express Briefs, vol. 58, no. 9, pp. 600–604, Sep. 2011.

[5] "A novel adaptive filter implementation scheme using distributed arithmetic," in Asilomar Conference on Signals, Systems and Computers, Nov. 2011, pp. 160–164.

[6] P. K. Meher and S. Y. Park, "High-throughput pipelined realization of adaptive FIR filter based on distributed arithmetic," in IFIP/IEEE International Conference on Very Large Scale Integration, Oct. 2011. pp. 428–433.

[7] W.Huang, V.Krishnan and D.V.Anderson, "Conjugate distributed arithmetic FIR filters and their hardware implementation," IEEE international Midwest symposium, 2006, vol.2, pp. 295-299.

[8] W.Huang and D.V.Anderson, "Adaptive filters using modified sliding-block distributed arithmetic with offset binary coding," IEEE international conference, 2009, pp. 545-548.

[9] Sang Yoon Park and P.K.Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," IEEE Transactions on Circuits and systems II, vol. 60, pp. 346-350.